Metric and Representation Learning

by

Rishi Sonthalia

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied and Interdisciplinary Mathematics)
in The University of Michigan
2021

Doctoral Committee:

Visiting Professor Anna Gilbert, Co-Chair
Associate Professor Raj Rao Nadakuditi, Co-Chair
Professor Sergey Fomin
Professor Seth Pettie

Rishi Saurabh Sonthalia

rsonthal@umich.edu

ORCID iD: 0000-0002-0928-392X

ACKNOWLEDGEMENTS

Thank you to my amazing advisors Anna Gilbert and Raj Rao Nadakuditi. Without their support, and help this thesis would not be possible at all. Thank you to my parents, my sisters, and my friends for their support as well. Finally, thank you to the people who put together a nice LaTeX template for me to use.

TABLE OF CONTENTS

LIST OF ALGORITHMS

Algorithm

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

All data has some inherent mathematical structure. I am interested in understanding the intrinsic geometric and probabilistic structure of data to design effective algorithms and tools that can be applied to machine learning and across all branches of science.

The focus of this thesis is to increase the effectiveness of machine learning techniques by developing a mathematical and algorithmic framework using which, given any type of data, we can learn an optimal representation. Representation learning is done for many reasons. It could be done to fix the corruption given corrupted data (noisy or missing values) or to learn a low dimensional or simpler representation, given high dimensional data or a very complex representation of the data. It could also be that the current representation of the data does not capture the important geometric features of the data.

One of the many challenges in representation learning is determining ways to judge the quality of the representation learned. In many cases, the consensus is that if d is the natural metric on the representation (such as L_2 distance for Euclidean embeddings), then this metric should provide meaningful information about the data. Many examples of this can be seen in areas such as metric learning, manifold learning, and graph embedding. However, most algorithms that solve these problems learn a representation in a metric space first and then extract a metric.

A large part of my research is exploring what happens if the order is switched, that is, learn the appropriate metric first and the embedding later. The philosophy behind this approach is that understanding the inherent geometry of the data is

the most crucial part of representation learning. Often, studying the properties of the appropriate metric on the input data sets indicates the type of space, we should be seeking for the representation. Hence giving us more robust representations. Optimizing for the appropriate metric can also help overcome issues such as missing and noisy data. My projects fall into three different areas of representation learning.

- Geometric and probabilistic analysis of representation learning methods.
- Developing methods to learn optimal metrics on large datasets.
- Applications.

For the category of geometric and probabilistic analysis of representation learning methods, we have three projects. First, designing optimal training data for denoising autoencoders. Second, formulating a new optimal transport problem and understanding the geometric structure. Third, analyzing the robustness to perturbations of the solutions obtained from the classical multidimensional scaling algorithm versus that of the true solutions to the multidimensional scaling problem.

For learning optimal metric, we are given a dissimilarity matrix $\hat{D}$, some function f and some a subset S of the space of all metrics and we want to find $DinS$ that minimizes $f(D, \hat{D})$. In this thesis, we consider the version of the problem when S is the space of metrics defined on a fixed graph. That is, given a graph G, we let S, be the space of all metrics defined via G. For this S, we consider the sparse objective function as well as convex objective functions. We also looked at the problem where we want to learn a tree. We also show how the ideas behind learning the optimal metric can be applied to dimensionality reduction in the presence of missing data.

Finally, we look at an application to real world data. Specifically trying to reconstruct ancient Greek text.

CHAPTER I

Introduction

Representation learning is a fundamental technique in science. The goal of representation learning is to learn a good representation of the object of interest. Specifically, for a lot of different machine learning and data science problems, we are given some data Z, and we would like to learn a representation X of the data Z and some function $f(\boldsymbol{x}; \boldsymbol{\theta}) : X \to Y$. To do this, we have to learn the representation for two different objects X and f, which is parameterized by $\boldsymbol{\theta}$.

First, we want to learn a representation X of the data Z. That is, given some representation of the data, we want to learn a new representation of the X such that it is easier to work with X instead of Z. Second, we want to learn a good representation of the function f, which is usually parameterized via latent variables $\boldsymbol{\theta}$. For both objects, we will want to learn a representation in some metric space. Hence there are two different geometries at play. The geometry of the data and the geometry of the parameter space. Understanding both is crucial to designing good machine learning algorithms.

1.1 Data Geometry

Data encompasses many different forms of information such as images, text, coordinates in a vector space, graphs, and many more different structures. Learning a

good representation of the data is a crucial pre-processing step. This allows us to do two things.

1. Understanding the geometry of the data helps us gain new insights into the data. This lets us answer many questions about the data generation process and uncover fundamental structures in the data.

2. Learning a good representation helps us improve computational efficiency. This representation may be efficient in terms of query time, in terms of memory, in terms of time taken to learn the function f, in terms of the robustness of the function f, and many other such statistics.

Currently, most methods for learning representation first choose the type of metric space that they want to learn a representation in and then learn a representation. The kind of metric space chosen usually depends on a researcher's prior knowledge of the data. However, being able to infer the type of geometry from the data is a promising area of active research.

Despite the limitations of having to arbitrarily pick the geometry beforehand, current methods have proved to be extremely successful. Hence, we go over the major categories of such methods.

1.1.1 Dimensionality Reduction

The first reason to learn a new representation of the data is to reduce the dimension of the data. Reducing the dimension helps us improve the efficiency of any algorithm that we run on the data, as well as reduces the amount of space used to store the data.

The first major dimensionality reduction technique is Principal Component Analysis or PCA. For PCA, we assume that we are given data matrix Z that is m by n. That is, we have n data points, and each data point has m coordinates. We assume that the data lies on a low dimensional linear space of dimension r, and we want to learn

a representation X such that this best approximates Z. To do this, PCA solves the problem by finding X such that

$$X = \underset{\hat{Z}, rank(\hat{Z}) \leq r}{\arg\min} \|\hat{Z} - Z\|_F^2.$$

This problem is solved by computing the singular value decomposition (SVD) of X and keeping the first r singular values and components.

A related dimensionality reduction technique is known as Multidimensional Scaling or MDS *Torgerson* (1952). Here our input is no longer a data matrix but instead is a metric or a dissimilarity matrix Z. The goal is then to learn a low dimensional Euclidean embedding X such that if D is the Euclidean distance matrix for X, then $\|D - Z\|_F^2$ is small. This is done by first centering the dissimilarity matrix so that the rows and columns have mean zero and then computing the eigenvalue decomposition and keeping the first r eigenvectors, scaled by the square root of the first r eigenvalues. These scaled eigenvectors then form our low dimensional embedding.

In many cases, the data doesn't lie on some low dimensional linear structure but instead lies on some low dimensional non-linear structure. Here we assume that the data lies on some low dimensional manifold that is embedded in high dimensional Euclidean space. Even though we assume that the data lies on a non-linear structure, we are still trying to learn a linear representation of the data. In this case, we employ different techniques such as Isomap *Tenenbaum et al.* (2000b), Local Linear Embeddings (LLE) *Roweis and Saul* (2000), diffusion maps *Belkin and Niyogi* (2003); *Coifman and Lafon* (2006), and Stochastic Neighborhood Embedding (SNE) *Hinton and Roweis* (2002); *Maaten and Hinton* (2008).

For Isomap, we start by creating a k nearest neighbor weighted graph based on the metric on the given data. The idea is that the points that are embedded close together are close together on the manifold as well. We then use this weighted graph

to compute the all pair shortest path metric. That is, the distance between any two nodes is the weight of the shortest path connecting them. The hope is that these paths approximate geodesics on the manifold. We then use MDS to embed this metric into Euclidean space.

In Isomap, we tried to capture the local and global structure of the data, with LLE, as the name suggests, we only try to capture the local structure. The main idea behind the algorithm is that since the data lies on a manifold, it is locally linear. Hence the problem postulates findings weights W_{ij} such that any data point Z_i can be well approximated by $\sum_j W_{ij} Z_j$, where $W_{ij} = 0$ for data points j that are not near Z_i (either some epsilon neighborhood or within the k nearest points). We then learn new coordinates X_i such that $X_i - \sum_j W_{ij} X_j$ is small for all i. We do this by computing the spectral decomposition of $(I - W)^T (I - W)$ and keeping the first r appropriately scaled eigenvectors.

For diffusion maps, imagine placing a Gaussian distribution at each point. That is, given a data point Z_i, imagine there is an isotropic Gaussian distribution with density f_i centered at Z_i. Then for all of the other points, we let $P_{ij} = f_i(d(Z_i, z_j))/C_i$ where d is the given metric on the data set and C_i is an appropriate normalizing constant. Thus, we can think of the matrix P with entries P_{ij} as the transition matrix for a Markov chain defined on the data. We then run a diffusion process on this Markov chain for t and calculate the t step transition probability matrix. The idea here is that small t this transition matrix represents local structure, and as t increases, we get a more global structure. We then embed this by again computing the spectral decomposition.

Finally, with SNE, we note that the above-defined P_{ij} for a fixed i defines a probability distribution. We then imagine there exists a random low dimensional representation X, and we have similar probability distributions Q_{ij} defined. We then optimize, via gradient descent type algorithms, to minimize some divergence between

these distributions, usually the Kullback-Liebler divergence. One can also change the distribution place on each point from Gaussian to other distributions. One common variant called T-SNE places a heavy-tailed t-distribution on the low dimensional points.

Another common technique is to use an autoencoder. Here the idea is to learn two maps F, G such that $G \circ F$ is the identity map on our data. Here F maps from a high dimensional space to a low dimensional space. Most commonly, F, G are parameterized as neural networks. Then we have that $X = F(Z)$. Thus, again, we have learnt a linear representation of our non-linear data.

One final technique that should be mentioned is via the Johnson Lindenstrauss lemma *Joachims et al.* (2009). In the paper, it was shown given Euclidean data with N points, we can find a low dimensional embedding in $O(\log(N)\epsilon^{-2})$ dimensions, such that the multiplicative distortion is at most $(1+\epsilon)/(1-\epsilon)$. Further, this can be easily achieved by random linear maps, such as, as a matrix G whose entries are appropriately scaled i.i.d mean 0 Gaussian.

So far, we have been learning low-dimensional linear structures. What about learning low-dimensional non-linear structures? This is less commonly done. However, there are some exceptions. In recent years, people have been very interested in learning embeddings into low dimensional hyperbolic manifolds and has lead to the development of many different techniques for learning such representations *Nickel and Kiela* (2017, 2018b); *Sala et al.* (2018); *Sonthalia and Gilbert* (2020c).

1.1.2 Denoising Data

Another reason to learn a new representation X of the data Z is to denoise the data. That is, we are assuming that we are given some data Z that is noisy, say has Gaussian noise added to it or has missing entries, then, in that case, we would like to learn the denoised version.

One common technique to do this is via projections or constrained optimization problems. The idea here is that we assume that our data likes in some subspace S. The goal is then to find $X \in S$ such that X is close to Z via some loss function. That is

$$X \arg\min_{\hat{Z}} F(\hat{Z}, Z).$$

For example suppose $F = \|\cdot\|_F^2$ is the Frobenius norm and S is the space of rank r matrices. Then this solution X is again given by computing the SVD of Z and taking the first r components.

One issue with such techniques is that if $\tilde{Z}$ is the original denoised data matrix, then this set up minimizes, $F(X, Z)$ not $F(X, \tilde{Z})$, which could still be large. In the above example, it has been shown that we can solve for $F(X, \tilde{Z})$, even without known $\tilde{Z}$ by shrinking the singular values *Nadakuditi* (2014).

Another type of noisy data is given the mixture of many different data sources. That is, if the rows of $\tilde{Z}$ are the original data points, then we have that $Z = A\tilde{Z}$ for some matrix A (here, we have a linear mixture). In this case, we solve the problem using independent component analysis or ICA *Comon* (1994); *Jutten and Hérault* (1991). For PCA, we computed the SVD, where each singular vector is found such that it maximizes the variance while being orthogonal to the previously found singular vectors. In ICA, we instead find the components that maximize the absolute value of the fourth cumulant.

One can also adapt the idea behind autoencoders to denoise data. Here instead of training the network to learn the identity map, we train it to learn the map from the noisy to denoised data. While autoencoder type structures have been proved to be useful for such tasks, in principle, any network architecture can be used. However, in general, they are all regularized so that the learned map has high regularity.

Another common type of noisy data is missing entries or masked noise models. Here, we do not have the complete data Z, but some version of it with missing

data. One common example of such a setup is matrix completion. Here we are given a low-rank matrix Z with many missing entries, and the goal is to fill in the missing entries. One common strategy for doing this is via singular value thresholding *Candès and Tao* (2010). That is, we fill in zeros, compute the SVD and the nearest rank r approximation. We then fix the known entries, compute the nearest rank r approximation and repeat until the matrix does not change.

This idea of filling in missing entries plays a big role in natural language processing. Here, a network is trained to fill in the missing words in a text. Once the large network is trained for this task, the network minus the last layer is assumed to have learned a good representation of the language. This network is then extended and then fine-tuned on various other tasks. This method forms the basis for model large language models in use today *Vaswani et al.* (2017); *Devlin et al.* (2019).

1.1.3 Learning Embeddings

A lot of data is presented as structured data such as text and is not given as coordinates. Most data science and machine learning techniques assume that data lives in Euclidean space (or something similar). Hence another common representation learning idea is to learn an embedding into Euclidean space.

We have already seen one technique for learning embeddings. That is, MDS, or in general, given some distance or dissimilarity matrix, computing the spectral decomposition and using the scaled eigenvectors as coordinates.

The ideas behind autoencoders and missing data have also been used to learn embeddings of structured objects. Here the goal is again to learn two functions F, G such that F takes some of the structured object (words in a text, nodes in graphs, group elements in groups) and then outputs an embedding. G then takes the embedding and predicts a related object. For example, the word2vec *Mikolov et al.* (2010) has two models. First is continuous bag of words (CBOW), where we

take the words in a window around a target word in a text and then try and predict the targeted word. Second is the skip-gram model, where we take the targeted word and attempt to predict the window around the word. A similar idea can be used for graphs *Narayanan et al.* (2017) and groups *Cerliani* (2020).

Another area where we learn embedding in an area called metric learning. In metric learning, we are given a dataset and want to learn an approximate metric on the data set. The way we learn this metric is to learn a Euclidean embedding and then learn a Mahalanobis metric on the data set. One common technique is to optimize for the matrix M in the Mahalanobis metric. Another technique is to use a neural network that is applied on two inputs, and then the loss function is based on how close the distance between the output vectors is to some target distance.

1.1.4 Learning Combinatorial Structure

Another form of representation learning is to learn a simple combinatorial structure. One common simple combinatorial structure that is learned is a sparse graph and, in many cases, a tree. There are many methods that learn subgraphs. First are methods that, given a graph, extract a sparse subgraph such as minimum spanning tree *Prim* (1957). Or, in general, methods that low learn low stretch trees *Alon et al.* (1995); *Elkin et al.* (2005). The other paradigm is where we are given a metric on the data, and then we learn a tree that represents this data. Examples of such methods include Neighbor Join *Saitou and Nei* (1987); *Abraham et al.* (2007); *Bartal* (1998).

Another type of discrete object that we learn is compressed sensing *Donoho* (2006); *Tropp and Gilbert* (2007) . Here we are given an overcomplete dictionary or basis and data Z such that Z can be well represented by a sparse combination X of our dictionary atoms.

1.2 Parameter Geometry

Now that we the representation for the input, X, how do we represent the function f? Here we assume that f is parameterized by parameters $\boldsymbol{\theta}$. This is either coordinates in some function space or neural networks.

Historically, this was done via linear regression. That is, we assume that f is a linear function on the data. To learn non - linear functions, we would use the kernel trick, which is just learning a different X and then doing linear regression.

Another example of learning function parameterization was to learn an overcomplete set of functions such that the functions we care about can be represented as sparse linear combinations of such basis functions. This is known as dictionary learning. One example of a basis commonly used for discrete functions is the discrete Fourier basis along with bump functions. Another common example is the radial basis functions, which place a Gaussian function at each point.

If the functions that are trying to learn are density functions, then another common method is to learn the moments of distributions. Here we have techniques such as Latent Dirichlet Allocation (LDA) and Gaussian Mixture models GMM. Another similar idea is Gaussian Processes, where for each input, we again learn a Gaussian distribution, but now we can incorporate prior information.

More recently, with the advent of deep neural networks, we parameterize functions as an iterated sequence of linear and non - linear functions, with the coefficients of the linear function forming $\boldsymbol{\theta}$.

With information geometry, we want to understand the manifold of various kinds of probability distributions. Here we are no longer interested in just parameterizing the space of function but also imposing a metric on the space of functions as well. In many cases, we can think of the manifold of probability distributions. Imposing a metric on this space lets us use ideas such as natural gradient descent or manifold gradient descent to learn optimal parameters for our function f.

1.3 Connection between Data and Parameter Geometry

Finally, how do these two concepts relate? The major connection is provided via optimal transport. Let X be a ground space and d be a metric on X, let $\mu, \boldsymbol{u}$ be two distributions or functions on X. Let $\Pi(\mu, \ni)$ be the space of all measures on the product space $X \times X$ such that the marginals are μ and ν. Then, this metric d on the data space X corresponds to a metric W_k (for integers $k \geq 1$ on the function as follows

$$W_k(\mu, \nu)^k = \inf_{\pi \in \Pi(\mu,\nu)} \int d(x, y)^k \mu(x)\nu(y).$$

Better understanding this connection will help us better leverage the various data representation learning and function representation learning methods.

1.4 In this thesis

Now that we have an idea of the types of problems that exist in representation learning, we shall briefly discuss the material covered in this thesis. The types of problems looked at in this thesis fall under 3 categories.

- Geometric and probabilistic analysis of representation learning methods.
- Developing methods to learn optimal metrics on large datasets.
- Learning representations from real data that capture the inherent geometry of the data for applications.

That being said, multiple projects that we are working on span multiple categories. As mentioned, I am also interested in the interplay between theory and applications. Hence my projects will span both theoretical and application work with large interplays between the theoretical and application work.

1.4.1 Learning Good Metrics

Given a dissimilarity matrix $\hat{D}$, we want to find the closest metric D. Specifically, we want to solve the following problem.

$$\begin{aligned} &\text{minimize} && \|\hat{D} - D\|_p \\ &\text{subject to} && D_{ij} + D_{jk} \geq D_{ik} \qquad \forall i, j, k \end{aligned} \tag{1.1}$$

This problem is the archetypal example of a general class of problems that we call metric constrained problems. More generally, we are given some function f and some a subset S of the space of all metrics, and we want to optimize f over S. Solving this problem for different families of functions f and different sets S has resulted in the following sequence of papers by me *Sonthalia and C. Gilbert* (2018); *Fan et al.* (2020); *Sonthalia and Gilbert* (2020a,c).

1.4.2 Chapter II - Sparse Metric Repair.

In this chapter, we look at the sparse version of the problem in Equation 1.1. Here we generalize the problem so that not only is the output expected to be sparse, but the input is expected to be sparse as well. We prove a variety of hardness results in terms of lower bounds and upper bounds. We combinatorially characterize all of the solutions and provide a fixed-parameter tractable algorithm for the problem. This chapter is based on the paper *Fan et al.* (2020).

1.4.3 Chapter III - Geometric Manifold Repair in the Presence of Missing Data.

In this chapter, we show that the ideas from metric repair can be used to do manifold learning in the presence of missing data. We provide an alternate to imputing the high-dimensional data. Our method approximates the metric using the data with missing values, and then repairs this metric. We also show that if we use the repaired

metric to do dimensionality reduction, then this results in favorable performance compared to methods that impute the data such as *Á. Carreira-Perpiñán and Lu* (2011); *Scholz et al.* (2005). This is based on the chapter *Sonthalia and C. Gilbert* (2018).

1.4.4 Chapter IV - Project and Forget: Solving Highly Constrained Convex Optimization Problems.

Solving the problem in Equation 1.1 is hard due to the number of constraints. In this chapter, we show that standard solvers cannot solve the problem for more than 200 data points. In this chapter, we developed a general-purpose optimization technique called PROJECT AND FORGET that can be used to solve problems with a large number of constraints. In the paper, we show that using our solver, we can solve larger instances of metric nearness, correlation clustering, quadratically regularized optimal transport, and information-theoretic metric learning than what previous solvers could solve. As a highlight, we solve a problem with 10^{15} constraints. This chapter is based on the paper *Sonthalia and Gilbert* (2020a).

1.4.5 Chapter V - Learning Good Quality Hyperbolic Representations Quickly.

In this chapter, we present a new algorithm TREEREP which, when given a metric, learns a tree structure and a tree metric on that data. In the paper, we show that on the task of learning low dimensional hyperbolic embeddings, we **outperform** many hyperbolic embedding techniques as *Nickel and Kiela* (2017, 2018a) and *Sala et al.* (2018), **while also being multiple orders of magnitude faster.** This chapter is based on the paper *Sonthalia and Gilbert* (2020c)

1.4.6 Analyzing Representation Learning Methods

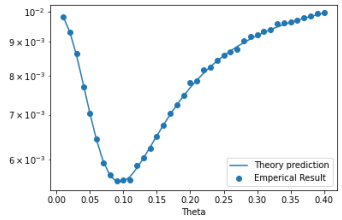
In this category, we have three projects. First, designing optimal training data for denoising autoencoders. Second, formulating a new optimal transport problem and understanding the geometric structure. Third, analyzing the robustness to perturbations of the solutions obtained from the classical multidimensional scaling algorithm versus that of the true solutions to the multidimensional scaling problem.

1.4.7 Chapter VI - Dual Regularized Optimal Transport.

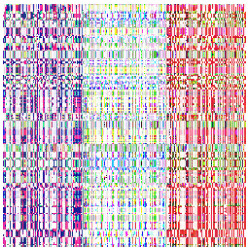
In the paper *Sonthalia and Gilbert* (2020b), we present a new formulation of unbalanced optimal transport called Dual Regularized Optimal Transport (DROT). We argue that regularizing the dual formulation of optimal transport results in a version of unbalanced optimal transport that gives us control over mass creation and destruction. We also show that solutions to DROT are sparse. Additionally, using Project and Forget (see Chapter IV), we can solve the problem for large datasets. This chapter is based on the paper *Sonthalia and Gilbert* (2020b).

1.4.8 Chapter VII - Robustness of cMDS via Spectral Analysis.

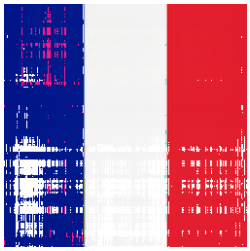
Classical MDS algorithm is only an approximation of the true multidimensional scaling problem as it does not find the embedding whose metric is closest to the original metric (See *Qi and Yuan* (2014a) for details). We are interested in understanding the robustness of the solution obtained from cMDS versus the true solutions. In this chapter, we derive a formula for the error of the cMDS analysis, that is based on the spectral decomposition on a matrix obtained from the input matrix. From the formula, we notice that if the original input to the cMDS is non-Euclidean or is even a perturbed Euclidean distance matrix, then we have this non-desirable behavior, whereas the embedding dimension increases the error eventually increase. In the chapter, we also design a new approximation algorithm that fixes this issue. This



(i) Plot showing Empirical Mean Squared Error averaged over 40 trials and the theoretical predicted MSE.



(ii) Traditional set up: train and test data have the same SNR.



(iii) My set up: train SNR modified.

Figure 1.1: Plot showing results from the analysis of the formula

chapter is based on the paper *Sonthalia et al.* (2021)

1.4.9 Chapter VIII - Probabilistic Analysis of Denoising Autoencoders.

For this project, under some assumptions, I have derived a theoretical formula for the generalization error. (one case can be seen in Equation 1.2).

$$MSE = \frac{\theta_{test}^2}{(1+\theta_{train}^2 M)^2} + \frac{\theta_{train}^2 c + \theta_{train}^4 M}{(1+\theta_{train}^2 M)^2(1-c)} \tag{1.2}$$

This formula then gives us the following counterintuitive insight! The SNR for the training data and test data **should not be the same** unless we have an infinite amount of training data. The formula also provides the optimal SNR for the training data. We can experimentally verify this as seen in Figure 1.1. In Figure 1.1i, we see that the theoretical predicted MSE and experimental MSE match. In Figures 1.1ii and 1.1iii, we show the result of trying to denoise a noisy image of the French flag. For Figure 1.1ii, the SNR of the training data was equal to the test data SNR, as is traditionally advised. On the other hand, for Figure 1.1iii, the SNR of the training data was modified based on insights from my analysis. As we can see, the theoretically suggested modification results in a much better image. This chapter is based on the paper *Sonthalia and Nadakuditi* (2021).

1.4.10 Chapter IX - Reconstructing Ancient Greek Text.

The problem is to reconstruct the original text, given ancient Greek text that is old, in fragments, and has a lot of missing letters. The problem is compounded because Ancient Greek is written without spaces and without diacritics. For this problem, I developed a modular framework that can be used to solve the problem. It was successful in reconstructing large parts of the missing information. For example, when 30% of the characters are missing, the input text is about 54% correct (due to missing diacritics, this is lower than 70%). After reconstructing the text using our framework, the text is 87% correct. More detail can be found in *Sonthalia et al.* (2020). This chapter is based on the paper *Sonthalia et al.* (2020).

CHAPTER II

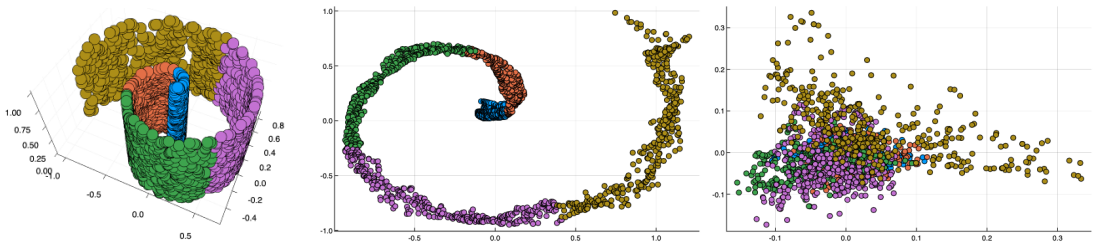
Generalized Metric Repair on Graphs

2.1 Introduction-GMR

Given a set of distances determined by a collection of data points, one of the most basic questions we can ask is whether the distances satisfy a metric. This basic property is fundamental to a large number of computational geometry and machine learning tasks such as metric learning, dimensionality reduction, and clustering (see for example *Wang and Sun* (2015); *Baraty et al.* (2011)). It is fortuitous when the underlying distances arise from a metric space or are at least well modeled by one, as certain tasks become provably easier over metric data (e.g., approximating the optimal TSP tour), and moreover it allows us to use a number of computational tools such as metric embeddings. However, due to noise, missing data, and other corruptions, in practice these distances often do not adhere to a metric.

As a motivating example, consider the following standard manifold learning task (*Belkin and Niyogi* (2003); *Roweis and Saul* (2000); *Tenenbaum et al.* (2000b)). Given a high dimensional data set, we wish to uncover its intrinsic lower dimensional structure, allowing us to visualize and to understand the geometry of the data. Isomap *Tenenbaum et al.* (2000b) is one of the standard embedding tools used to find this lower dimensional structure, and Figure 2.1 shows how Isomap nicely recovers the 2d spiral when embedding a 3d Swiss roll data set. However, as shown on the right in

Figure 2.1, if we perturb even a small fraction of the distances this structure is lost in the embedding produced by Isomap.



(i) Original Swissroll data. (ii) Embedded true distance. (iii) Embedded corrupted distances.

Figure 2.1: (a) 2000 data points in the Swissroll. For (b) and (c) we took the pairwise distance matrix and added $2\mathcal{N}(0,1)$ noise to 5% of the distances. We then constructed the 30-nearest-neighbor graph G from these distances, where roughly 8.5% of the edge weights of G were perturbed. For (b) we used the true distances on G as the input to Isomap. For (c) we used the perturbed distances.

Motivated by the above applications, the problem of minimally fixing the distances to uncover the data metric was previously considered. Specifically, Fan et al. *Fan et al.* (2018d) and Gilbert and Jain *Gilbert and Jain* (2017) respectively formulated the *Metric Violation Distance (MVD)* and the *Sparse Metric Repair (SMR)* problems, where in both cases one is given a *full* distance matrix, and the goal is to modify as few entries as possible so that the repaired distances satisfy a metric.

More generally, however, the underlying distance graph will be incomplete as data may be missing or the constructed distance graph is inherently sparse, as the above manifold example demonstrates. Working directly with this incomplete graph is not only computationally more desirable when the graph is sparse, but also may be necessary to uncover the ground truth. For example, observe that for any graph we can attempt to fill in its missing edges by assigning them weights according to their shortest path distance. Thus naively one could attempt to fix the input graph, by solving MVD/SMR on this complete graph, and afterwards dropping any selected edges that were not in the original graph. Figure 2.2 shows that doing so, however,

can produce radically different and sub-optimal solutions.

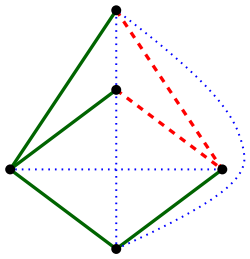


Figure 2.2: Original graph is the four solid green edges of weight 1, and two dashed red edges of weight 4. Added blue dotted edges all have weight 2. The original graph is repaired by increasing the lower left green edge, but the optimal solution in the complete graph decreases the two red edges.

Thus to appropriately capture this more general problem, we define the *Graph Metric Repair* problem as the natural graph theoretic generalization of the MVD and SMR problems:

> Given a positively weighted undirected graph $G = (V, E, w)$ and a set $\Omega \subseteq \mathbb{R}$, find the smallest set of edges $S \subseteq E$ such that by modifying the weight of each edge in S, by adding a value from Ω, the new distances satisfy a metric.

The additional graph structure introduced in the generalized problem lets us incorporate different types of relationships amongst data points and gives us more flexibility in its structure, and hence avails itself to be applicable to a richer class of problems. This general graph structure also elucidates the deep connections to cutting problems, which underlie several results in this paper, and which were not previously observed in *Fan et al.* (2018d); *Gilbert and Jain* (2017). In particular, as discussed below, our problem is closely related to MULTICUT (Problem 2.8.1), a generalization of the standard s-t cut problem to multiple s-t pairs, as well as LB-CUT (Problem 2.9.1), where only s-t paths with lengths up to a given threshold L must be cut.

It should also be noted that Graph Metric Repair, as well as MVD and SMR, are related to a large number of other previously studied problems. A short list includes:

metric nearness, seeking the metric minimizing the sum of distance value changes *Brickell et al.* (2008a); metric embedding with outliers, seeking the fewest points whose removal creates a metric *Sidiropoulos et al.* (2017); matrix completion, seeking to fill missing matrix entries to produce a low rank *Candès and Recht* (2012); and many more. See *Fan et al.* (2018d) for a more detailed discussion of these and other problems.

Contributions and Results: The main contributions of this paper are as follows:

- We transition all previously known structural results about SMR and MVD to the new graph theoretic version. In particular, we provide a *characterization* for the support of solutions to the increase ($\Omega = \mathbb{R}_{\geq 0}$) and general ($\Omega = \mathbb{R}$) versions of the problem. Furthermore, we provide a new structural result showing that the increase only problem reduces to the general one, where it is unknown if such a result holds for SMR and MVD.

- For any fixed constant ς, by parameterizing on the size of the optimal solution, we present a *fixed parameter tractable* algorithm for the case when G is ς-chordal. This not only answers an open question posed by *Fan et al.* (2018d) for complete graphs, but significantly extends it to the larger ς-chordal case (see *Chandran et al.* (2005) for characterizations of such graphs, many of which are the complements of a variety of families of graphs). Moreover, we get an upper bound on the number of optimal supports, as each one is seen by some branch of the algorithm.

- We give polynomial-time approx-preserving reductions from MULTICUT and LB-CUT to graph metric repair. This connection to the well studied MULTICUT problem is interesting in its own right, but by *Chawla et al.* (2006) it also implies graph metric repair is NP-hard, and cannot be approximated within any constant factor assuming the Unique Games Conjecture (UGC).

- We give approximation algorithms, parameterized by different measures of how far the input is from a metric. Significantly, our approximations mirror our hardness results. Call a cycle *broken* if it contains an edge whose weight is larger than the sum of all its other edges, and call the amount of this difference its *deficit*. We give an L-approximation, where L is the maximum number of edges in a broken cycle, while LB-CUT gives $\Omega(\sqrt{L})$-hardness. We give an $O(\kappa \log n)$-approximation, where κ is the number of distinct cycle deficit values, while in general the best known approximation for MULTICUT is $O(\log n)$.

- Finally, we give improved analysis of previous algorithms for the complete graph case. To keep the focus on our main results, this entire section has been moved to Appendix A.3.

2.2 Preliminaries

2.2.1 Notation and problem definition

Let us start by defining some terminology. Throughout the paper, the input is an undirected and weighted graph $G = (V, E, w)$. A subgraph $C = (V', E')$ is called a k-cycle if $|V'| = |E'| = k$, and the subgraph is connected with every vertex having degree exactly 2. We often overload this notation and use C to denote either the cyclically ordered list of vertices or edges from this subgraph. Let $C \setminus e$ denote the set of edges of C after removing the edge e, and $\pi(C \setminus e)$ denote the corresponding induced path between the endpoints of e.

A cycle C is **broken** if there exists an edge $h \in C$ such that $w(h) > \sum_{e \in C \setminus h} w(e)$. In this case, we call the edge h the **heavy** edge of C, and all other edges of C are called **light** edges. We call a set of edges a **light cover** if it contains at least one light edge from each broken cycle. Similarly, we call it a **regular cover** if it contains at least one edge from each broken cycle. We say that a weighted graph $G = (V, E, w)$

satisfies a metric if there are no broken cycles. Finally, let $\mathrm{Sym}_n(\Omega)$ be the set of $n \times n$ symmetric matrices with entries drawn from $\Omega \subseteq \mathbb{R}$. Note that the weight function w can be viewed as an $n \times n$ symmetric matrix (missing edges get weight ∞), and thus for any $W \in \mathrm{Sym}_n(\Omega)$, the matrix sum $w + W$ defines a new weight function. Now we can define the generalized graph metric repair problem as follows. In the following, $\|W\|_0$ is the number of non-zero entries in the matrix W, i.e., the ℓ_0 pseudonorm when viewing the matrix W as a vector.

Problem 2.0.1. Given $\Omega \subseteq \mathbb{R}$ and a positively weighted graph $G = (V, E, w)$ we want to find

$$\arg\min_{W \in \mathrm{Sym}(\Omega)} \|W\|_0 \text{ such that } G = (V, E, w + W) \text{ satisfies a metric, or return NONE,}$$

if no such W exists. Denote this problem as graph metric repair or $\mathrm{MR}(G, \Omega)$.

A matrix W is an *optimal solution* if it realizes the arg min in the above, and is a *solution* (without the *optimal* prefix) if $G = (V, E, w + W)$ satisfies a metric, but $\|W\|_0$ is not required to be minimum. The **support** of a matrix $W \in \mathrm{Sym}(\Omega)$, denoted S_W, is the set of edges corresponding to non-zero entries in W. As we will see in Proposition 2.7, given a support for a solution W, we can easily find satisfying entries. Thus, the main difficulty lies in finding the support. Throughout we use OPT to denote the size of the support of an optimal solution.

We also need the following basic graph theory definitions: K_n is the complete graph on n vertices. C_n is the cycle n vertices. A **chord** of a cycle is an edge connecting two non-adjacent vertices. For a given value ς, a graph G is called a ς-**chordal** if the size of the largest chordless cycle in G is $\leq \varsigma$.

Let the **deficit** of a broken cycle C, denoted $\delta(C)$, be the weight of its heavy edge minus the sum of the weights of all other edges in C. Similarly, $\delta(G)$ denotes the maximum of $\delta(C)$ over all broken cycles. Finally, let $L + 1$ be the maximum number

of edges in a broken cycle (i.e., L counts the light edges). Note δ and L are both parameters measuring the extent to which cycles are broken, δ with respect to weights and L with respect to the number of edges.

In several places we compute all pairs shortest paths (APSP). Let T_{APSP} denote the time to do so, where $T_{\text{APSP}} = O(mn + n^2 \log n)$ using Dijkstra's algorithm and Fibonacci heaps.

2.2.2 Previous results

Fan et al. *Fan et al.* (2018d) and Gilbert and Jain *Gilbert and Jain* (2017) studied the special case of $\text{MR}(G, \Omega)$ where $G = K_n$. Three sub-cases based on Ω were considered, namely $\Omega = \mathbb{R}_{\leq 0}$ (decrease only), $\mathbb{R}_{\geq 0}$ (increase only), and $\mathbb{R}$ (general). Various structural, hardness, and algorithmic results were presented for these cases. In particular, the major results from these previous works are as follows. (Note the notation and terminology here differs slightly from *Fan et al.* (2018d); *Gilbert and Jain* (2017).)

Theorem 2.1. *Fan et al.* (2018d); *Gilbert and Jain* (2017) The problem $\text{MR}(K_n, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\text{APSP}})$ time.

Theorem 2.2. *Fan et al.* (2018d) For a complete positively weighted graph $K_n = (V, E, w)$ and $S \subseteq E$:

1. S is a regular cover if and only if S is the support to a solution to $\text{MR}(K_n, \mathbb{R})$.

2. S is a light cover if and only if S is the support to a solution to $\text{MR}(K_n, \mathbb{R}_{\geq 0})$.

Theorem 2.3. *Fan et al.* (2018d); *Gilbert and Jain* (2017) Given the support S of a solution to $\text{MR}(K_n, \mathbb{R}_{\geq 0})$ or $\text{MR}(K_n, \mathbb{R})$, in polynomial time one can find a weight assignment to the edges in S which is a solution.

Gilbert and Jain (2017) Moreover, for $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$, if $K_n - S$ is connected, then for any edge $uv \in S$, setting the weight of uv to be the shortest distance between u and v in $K_n - S$ is a solution.

Theorem 2.4. *Fan et al.* (2018d) The problems $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(K_n, \mathbb{R})$ are APX-Complete, and moreover permit $O(OPT^{1/3})$ approximation algorithms.

2.3 Transitioning to Graph Metric Repair

In this section we generalize Theorems 2.1, 2.2, and 2.3 to the case when G is any graph, and additionally show that for general graphs $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ reduces to $\mathrm{MR}(G, \mathbb{R})$. Subsequently, in the later sections of paper, we provide a number of new stronger hardness and approximation results for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$ for general graphs, as well as an FPT algorithm for ς-chordal graphs, in effect generalizing and strengthening Theorem 2.4, and answering previously unresolved questions.

For $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ we have the following generalization of Theorem 2.1. Moreover, we observe the hardness proof of *Fan et al.* (2018d) implies if weights are allowed to increase even by a single value, the problem is APX-Complete. The proof of the theorem below follows fairly directly from previous work, and so has been moved to Appendix A.1.1, which contains additional corollaries.

Theorem 2.5. The problem $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\mathrm{APSP}})$ time.

Moreover, the problem becomes hard if even a single positive value is allowed. That is, if $0 \in \Omega$ and $\Omega \cap \mathbb{R}_{>0} \neq \emptyset$ then $\mathrm{MR}(G, \Omega)$ is APX-Complete.

2.3.1 Structural results

Theorem 2.2 suggests that the problem is mostly combinatorial in nature. We shall see that, in general, the difficult part of the problem is finding the support of an optimal solution. Next, we present a characterization of the support of all solutions to

the graph metric repair problem, generalizing Theorems 2.2, 2.3. It should be noted that the proof of the following is significantly simpler than the proof of Theorem 2.2 in *Fan et al.* (2018d). The key insight is:

(i) If the shortest path between two adjacent vertices is not the edge connecting them, then this edge is the heavy edge of a broken cycle.

Theorem 2.6. For any positively weighted graph $G = (V, E, w)$ and $S \subseteq E$:

1. S is a regular cover if and only if S is the support to a solution to $\mathrm{MR}(G, \mathbb{R})$.
2. S is a light cover if and only if S is the support to a solution to $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$.

Proof. First, assume that S is the support of a solution to $\mathrm{MR}(G, \mathbb{R})$ ($\mathrm{MR}(G, \mathbb{R}_{\geq 0})$). Suppose C is a broken cycle in G. If S does not contain any (light) edges from C, then changing (increasing) the weights on S could not have fixed C. Hence, S must be a regular (light) cover thus proving the "if" direction of both parts of the theorem.

For the "only if" direction, we are given a regular (light) cover $S \subseteq E$ which we use to define a graph $\hat{G} = (V, E \setminus S, w)$. Note that since S is either a regular or light cover, S contains at least one edge from all broken cycles of G. Thus, since $\hat{G}$ is G with the edges of S removed, $\hat{G}$ has no broken cycles. Therefore, the shortest path between all adjacent vertices in $\hat{G}$ is the edge connecting them.

Now we define another graph $G' = (V, E, w')$ where $w'(e) = w(e)$ for all $e \in E \setminus S$ and for all $e \in S$, $w'(e)$ is the length of the shortest path between its end points in $\hat{G}$ or $\|w\|_\infty$ (the maximum edge weight in $\hat{G}$) if no path exists.

To prove **1.**, it suffices to show G' satisfies a metric, since G' is G with only weights from edges in S modified. For any edge $e \in E$, if $w'(e)$ is the shortest path between its nodes in G' then e is not a heavy edge in G'. Therefore, edges that are in both G' and $\hat{G}$ and edges that are in G' whose weight was set to length of the shortest path between its end points in $\hat{G}$ are not heavy edges. Thus, we only need to look at edges in G' whose weight is $\|w\|_\infty$. These are edges that connect two disconnected

components in $\hat{G}$. Thus, any cycle in G' with such an edge must involve another edge between components which also has weight $\|w\|_\infty$. However, a cycle with two edges of maximum weight cannot be broken, and thus such edges cannot be heavy in G'. Therefore, there are no heavy edges in G', and so G' satisfies a metric.

To prove **2.**, it now suffices to show that for all $e \in E$, we have that $w'(e) \geq w(e)$. For all $e \in E \setminus S$, we know that $w'(e) = w(e)$. Now, suppose for contradiction that for some $e \in S$, we have $w'(e) < w(e)$. Note if we set $w'(e) = \|w\|_\infty$, then we cannot have $w'(e) < w(e)$. Thus, $w'(e)$ must be the weight of the shortest path between the end points of e in $\hat{G}$. Let P be this shortest path in $\hat{G}$. This implies G has a broken cycle $C = P \cup \{e\}$ for which e is the heavy edge. Since S is a light cover, it has a light edge from each broken cycle. So, S must have a light edge from C, but then P could not have existed in $\hat{G}$, a contradiction. Hence, $w'(e) \geq w(e)$ and we have an increase only solution with such a set S. □

Furthermore, given a weighted graph G and a potential support S_W for a solution W, in $O(T_{\text{APSP}})$ time we can determine if there exists a valid (increase only or general) solution on that support, and if so, find one. This is a generalization of Theorem 2.3, interestingly improving upon the linear programming approach of *Fan et al.* (2018d). Its proof is related to the above theorem, and again uses insight (i), though due to space has been moved to Appendix A.1.2.

Algorithm 1 Verifier

```
function VERIFIER(G = (V, E, w), S)
    M = ||w||_∞, Ĝ = (V, E, ŵ)
    For each e ∈ S set ŵ(e) = M and for each e ∈ E \ S, set ŵ(e) = w(e)
    For each (u, v) ∈ E, update w(u, v) to be length of the shortest path from u to v in Ĝ
    if Only edges in S had weights changed (or increased for increase only case) then
        return w
    else
        return NULL
```

Proposition 2.7. The VERIFIER algorithm, given a weighted graph G and a potential support for a solution S, determines in $O(T_{\text{APSP}})$ time whether there exists a valid (increase only or general) solution on that support and if so finds one.

2.3.2 Reducing MR$(G, \mathbb{R}_{\geq 0})$ to MR$(G, \mathbb{R})$

We now show that MR$(G, \mathbb{R}_{\geq 0})$ reduces to MR$(G, \mathbb{R})$. In later sections, this lets us focus on MR$(G, \mathbb{R})$ for our algorithms and MR$(G, \mathbb{R}_{\geq 0})$ for our hardness results. Note that whether an analogous statement holds for the previously studied $G = K_n$ case, is not known, and the following does not immediately imply this as it does not construct a complete graph.

Theorem 2.8. There is an approximation-preserving, polynomial-time reduction from MR$(G, \mathbb{R}_{\geq 0})$ to MR$(G, \mathbb{R})$.

Proof. Let $G = (V, E, w)$ be an instance of MR$(G, \mathbb{R}_{\geq 0})$. Find the set $H = \{(s_1, t_1), \ldots, (s_{|H|}, t_{|H|})\}$ of heavy edges of all broken cycles by comparing the weight of each edge to the shortest path distance between its endpoints. We now construct an instance, $G' = (V', E', w)$, of MR$(G, \mathbb{R})$. For all $1 \leq i \leq |H|$ and $1 \leq j \leq |E| + 1$, let $Q = \{v_{ij}\}_{i,j}$ be a vertex set, and let $F_l = \{(s_i, v_{ij})\}_{i,j}$ and $F_r = \{(t_i, v_{ij})\}_{i,j}$ be edge sets. Let $V' = V \cup Q$ and $E' = E \cup F_l \cup F_r$, where all (s_i, v_{ij}) edges in F_l have weight $Z = 1 + \max_{e \in E} w(e)$, and for any i all (t_i, v_{ij}) edges in F_r have weight $Z - w((s_i, t_i))$.

Let C be any broken cycle in G with heavy edge (s_i, t_i) for some i. First, observe that $C' = (C \setminus (s_i, t_i)) \cup \{(s_i, v_{ij}), (t_i, v_{ij})\}$ is a broken cycle with heavy edge (s_i, v_{ij}), for any j. To see this, note that $w((s_i, v_{ij})) = Z = w((t_i, v_{ij})) + w((s_i, t_i))$. Thus since C is broken,

$$w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i)) > w((t_i, v_{ij})) + w(C \setminus (s_i, t_i)),$$

and thus by definition C' is broken with heavy edge (s_i, v_{ij}). Hence each broken cycle

C in G, with heavy edge (s_i, t_i), corresponds to $|E| + 2$ broken cycles in G', namely, C itself and the cycles obtained by replacing (s_i, t_i) with a pair $(s_i, v_{ij}), (t_i, v_{ij})$, for any j.

We now show the converse, that any broken cycle C' in G' is either also a broken cycle C in G, or obtained from a broken cycle C in G by replacing (s_i, t_i) with $(s_i, v_{ij}), (t_i, v_{ij})$ for some j. First, observe that for any i, any cycle containing the edge (s_i, v_{ij}) must also contain the edge (t_i, v_{ij}), and moreover, if a cycle containing such a pair is broken, then its heavy edge must be (s_i, v_{ij}) as $w((s_i, v_{ij})) = Z$. Similarly, any cycle containing more than one of these pairs of edges (over all i and j) is not broken, since such cycles then would contain at least two edges with the maximum edge weight Z. So let C' be any broken cycle containing exactly one such (s_i, v_{ij}), (t_i, v_{ij}) pair. Note that C' cannot be the cycle $((s_i, v_{ij}), (t_i, v_{ij}), (s_i, t_i))$, as this cycle is not broken because $w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i))$. Thus, $C = C' \setminus \{(s_i, v_{ij}), (t_i, v_{ij})\} \cup \{(s_i, t_i)\}$ is a cycle, and C' being broken implies C is broken with heavy edge (s_i, t_i), implying the claim. This holds since

$$w(s_i, t_i) = w((s_i, v_{ij})) - w((t_i, v_{ij})) > w(C' \setminus (s_i, v_{ij})) - w((t_i, v_{ij})) = w(C \setminus (s_i, t_i)).$$

Now consider any optimal solution M to the $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ instance G, which by Theorem 2.6 we know is a minimum cardinality light cover of G. By the above, we know that M is also a light cover of G', and hence is also a regular cover of G'. Thus by Theorem 2.6, M is a valid solution to the $\mathrm{MR}(G, \mathbb{R})$ instance. Conversely, consider any optimal solution M' to the $\mathrm{MR}(G, \mathbb{R})$ instance G', which by Theorem 2.6 is a minimum cardinality regular cover of G'. The claim is that M' is also a light cover of G, and hence is a valid solution to the $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ instance. To see this, observe that since all broken cycles in G are broken cycles in G', M' must be a regular cover of all broken cycles in G, and we now argue that it is in fact a light cover. Specifically,

consider all the broken cycles in G which have a common heavy edge (s_i, t_i). Suppose there is some cycle in this set, call it C, which is not light covered by M'. As M' is a regular cover for G', this implies that for any j, the broken cycle described above determined by removing the edge (s_i, t_i) from C and adding edges (s_i, v_{ij}) and (t_i, v_{ij}), must be covered either with (s_i, v_{ij}) or (t_i, v_{ij}). However, as j ranges over $|E| + 1$ values, and these edge pairs have distinct edges for different values of j, M' has at least $|E| + 1$ edges. This is a clear contradiction with M' being a minimum sized cover, as any light cover of G is a regular cover of G', and G only has $|E|$ edges in total. □

2.4 Hardness

Previously, *Fan et al.* (2018d) gave an approximation-preserving reduction from Vertex Cover to both $\mathrm{MR}(K_n, \mathbb{R})$ and $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$. Thus, both are APX-complete, and in particular are hard to approximate within a factor of $2-\epsilon$ for any $\epsilon > 0$, assuming UGC *Khot* (2002). Since these hardness results were proven for complete graphs, they also immediately apply to the general problems $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$. Here we give stronger hardness results for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$ by giving approximation-preserving reductions from MULTICUT and LB-CUT.

Problem 2.8.1 (MULTICUT)**.** Given an undirected unweighted graph $G = (V, E)$ on $n = |V|$ vertices together with k pairs of vertices $\{s_i, t_i\}_{i=1}^{k}$, compute a minimum size subset of edges $M \subseteq E$ whose removal disconnects all the demand pairs, i.e., in the subgraph $(V, E \setminus M)$ every s_i is disconnected from its corresponding vertex t_i.

Chawla *et al. Chawla et al.* (2006) proved that if UGC is true, then it is NP-hard to approximate MULTICUT within any constant factor $L > 0$, and assuming a stronger version of UGC, within $\Omega(\sqrt{\log \log n})$. (The MULTICUT version in *Chawla et al.* (2006) allowed weights, but they remark their hardness proofs extend to the unweighted case.)

Theorem 2.9. There is an approximation-preserving, polynomial-time reduction from MULTICUT to MR$(G, \mathbb{R}_{\geq 0})$.

Proof. Let $G = (V, E)$ be an instance of MULTICUT with k pairs of vertices $\{s_i, t_i\}_{i=1}^{k}$. First, if $(s_i, t_i) \in E$ for any i, then that edge must be included in the solution M. Thus, we can assume no such edges exists in the MULTICUT instance, as assuming this can only make it harder to approximate the optimum value of the MULTICUT instance. We now construct an instance of MR$(G, \mathbb{R}_{\geq 0})$, $G' = (V', E', w)$. Let $V' = V$ and $E' = E \cup \{s_i, t_i\}_{i=1}^{k}$ where the edges in E have weight one and the edges (s_i, t_i), for all $i \in [k]$, have weight $n = |V|$.

If a cycle in G' has exactly one edge of weight n, then it must be broken since there can be at most $n - 1$ other edges in the cycle. Conversely, if a cycle C has no edge or more than one edge with weight n, then C does not have a heavy edge, and so is not broken.

Note that the edges from G are exactly the weight one edges in G', and thus, the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the weight n edges in G' are in one-to-correspondence with the (s_i, t_i) pairs from G. Thus, the cycles in G' with exactly one weight n edge followed by paths of all weight one edges connecting their endpoints, which by the above are exactly the set of broken cycles, are in one-to-one correspondence with paths between (s_i, t_i) pairs from G. Therefore, a minimum cardinality subset of edges which light cover all broken cycles, i.e., an optimal MR$(G, \mathbb{R}_{\geq 0})$ support, corresponds to a minimum cardinality subset of edges from E which cover all paths from s_i to t_i for all i, i.e., an optimal solution to MULTICUT. □

Problem 2.9.1 (LB-CUT)**.** Given a value L and an undirected unweighted graph $G = (V, E)$ with source s and sink t, find a minimum size subset of edges $M \subseteq E$ such that no s-t-path of length less than or equal to L remains in the graph after removing the edges in M.

An instance of LB-CUT with length L, is referred to as an instance of L-LB-CUT. For any fixed L, Lee *Lee* (2017b) showed that it is hard to approximate L-LB-CUT within a factor of $\Omega(\sqrt{L})$. Using a similar reduction as above, we argue the following.

Theorem 2.10. For any fixed value L, there is an approximation-preserving, polynomial-time reduction from L-LB-CUT to $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$.

Proof. Let $G = (V, E)$ be an instance of L-LB-CUT with source s and sink t. First, if $(s, t) \in E$, then that edge must be included in the solution M. Thus we can assume that edge is not in the LB-CUT instance, as assuming this can only make it harder to approximate the optimum value of the LB-CUT instance. We now construct an instance of $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$, $G' = (V', E', w)$. Let $V' = V$ and $E' = E \cup \{(s, t)\}$ where the edges in E have weight 1 and the edge (s, t) has weight $L + 1$.

First, observe that any cycle containing the edge (s, t) followed by $\leq L$ unit weight edges is broken, as the sum of the unit weight edges will be $< L + 1 = w((s, t))$. Conversely, any broken cycle must contain the edge (s, t) followed by $\leq L$ unit weight edges. Specifically, if a cycle does not contain (s, t) then it is unbroken since all edges would then have weight 1. Moreover, if a cycle contains (s, t) and $> L$ other edges, then the total sum of those unit edges will be $\geq L + 1 = w((s, t))$.

Note that the edges from G are exactly the weight one edges in G', and thus the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the edge (s, t) in G' corresponds with the source and sink from G. Thus by the above, the broken cycles in G' are in one-to-one correspondence with s-t-paths with length $\leq L$ in G. Therefore, a minimum cardinality subset of edges which light cover all broken cycles, i.e., an optimal support to $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$, corresponds to a minimum cardinality subset of edges from E which cover all paths from s to t of length $\leq L$, i.e., an optimal solution to LB-CUT. □

In the L-LB-CUT to $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ reduction of Theorem 2.10, one edge (the s, t

pair) has weight $L+1$ and all other edges have unit weight. Moreover, in the reduction from $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to $\mathrm{MR}(G, \mathbb{R})$ of Theorem 2.8, the max edge weight increases by 1. Thus, by these reductions, and previous hardness results, we have the following summarizing theorem.

Theorem 2.11. $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$ are APX-complete, and moreover assuming UGC neither can be approximated within any constant factor.

For any positive integer L, consider the problem defined by the restriction of $\mathrm{MR}(G, \mathbb{R})$ to integer weight instances with maximum edge weight L and minimum edge weight 1, or the further restriction of $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to instances where all weights are 1 except for a single weight L edge. Then assuming UGC these problems are hard to approximate within $\Omega(\sqrt{L})$.

2.5 Fixed Parameter Analysis for ς-Chordal Graphs

Let ς be a fixed constant, and let F_ς be the family of all ς-chordal graphs. Here we provide an FPT for $\mathrm{MR}(G, \mathbb{R})$ for any $G \in F_\varsigma$, parameterized on the optimal solution size OPT.

By Theorem 2.6, we seek a minimum sized cover of all broken cycles. First, we argue below that if G has a broken cycle, then it has a broken chordless cycle. This seems to imply a natural FPT algorithm for constant ς. Namely, find an uncovered broken chordless cycle and recursively try adding each one of its edges to our current solution.* However, it is possible to cover all broken chordless cycles while not covering the broken chorded cycles. These cycles are difficult to cover as they may be much larger than ς, though again by Theorem 2.6 they must be covered.

Consider an optimal solution W, with support S_W. Suppose that we have found

*One might construe this as FPT kernelization. The edges of the broken chordless cycles do form a kernel but its size is not bounded in our parameter. As an example, take $G = K_n$, set one edge weight to $n+1$, and all other weights to 1. There are $2n-3$ edges in the kernel while the optimal solution has size 1.

a subset $S \subsetneq S_W$, covering all broken chordless cycles in G. Intuitively, if we add to each edge in S its weight from W, then any remaining broken chordless cycle must be covered further, in effect revealing which edges to consider from the chorded cycles from the original graph G. The challenge, however, is of course that we don't know W a priori. We argue that despite this one can still identify a bounded sized subset of edges containing an edge from a cycle needing to be covered further.

Lemma 2.12. If G has a broken cycle, then G has a broken chordless cycle.

Proof. Let $C = v_1, \ldots, v_k$ be the broken cycle in G with the fewest edges, with v_1v_k being the heavy edge. If C is chordless, then the claim holds. Otherwise, this cycle has at least one chord v_iv_j. Now there are two paths P_1 and P_2 from v_i to v_j on the cycle. Let P_1 be the path containing the heavy edge of C. If $w(v_i, v_j) > \sum_{e \in P_2} w(e)$, then P_2 together with the edge v_iv_j defines a broken cycle with fewer edges than C. On the other hand, if $w(v_i, v_j) \leq \sum_{e \in P_2} w(e)$ then P_1 together with the edge v_iv_j defines a broken cycle with fewer edges than C. In either case we get a contradiction as C was the broken cycle with the fewest edges. □

Our FPT is shown in Algorithm 2, where we recursively build a potential support S up to our current guess at the optimal size k. The following lemma is key to arguing correctness.

Lemma 2.13. Consider any optimal solution W and its support S_W to an instance of metric repair for $G = (V, E, w) \in F_\varsigma$. If $S \subsetneq S_W$, then $F(G, S, OPT)$ adds at least one edge in $S_W \setminus S$ to P.

Proof. Consider the auxiliary graph $G_S = (V, E, \tilde{w})$, which has the same vertex and edge sets as G, but with the modified weight function:

$$\tilde{w} = \begin{cases} w(e) & e \notin S \\ W(e) + w(e) & e \in S \end{cases}$$

Algorithm 2 FPT

```
1: function F(G, S, k)
2:     if |S| = k then return VERIFIER(G,S)
3:     P = ∅
4:     if there exists a broken chordless cycle C such that C ∩ S = ∅ then P = C
5:     else
6:         for s ⊆ S such that |s| ≤ ς − 1 do
7:             Let C = {Chordless cycles C such that C ∩ S = s}
8:             C_1 ← arg min_{C∈C} Σ_{e∈C\s} w(e)
9:             C_2 ← arg max_{C∈C} w(h) − Σ_{e∈C\(s∪{h})} w(e), where h = arg max_{f∈C\s} w(f)
10:            Add (C_1 ∪ C_2) \ S to P
11:    for e ∈ P do
12:        X = F(G, S ∪ {e}, k)
13:        if X ≠ NULL then return X
14:    return NULL

15: function FPTWRAPPER(G)
16:    for k = 1, 2, ... do
17:        X = F(G, ∅, k)
18:        if X ≠ NULL then return X
```

Since $S \subsetneq S_W$, we have that G_S has a broken cycle. Thus, by Lemma 2.12, G_S has a chordless broken cycle. Suppose there is a chordless broken cycle in G_S that is edge disjoint from S (which occurs if and only if it is also broken in G), in which case, line 4 finds such a cycle. As this is a broken cycle, it must be covered by some edge in $S_W \setminus S$, and thus, we have added some edge in $S_W \setminus S$ to P.

Let us assume otherwise, that any chordless broken cycle in G_S has non-empty intersection with S. Let C be any such chordless broken cycle with $C \cap S \neq \emptyset$. Observe that as C is broken in G_S, it must be that $|C \cap S| < |C|$, as otherwise it would imply W was not a solution. Thus, as $G \in F_\varsigma$, we know that $|C| \leq \varsigma$, and so $|C \cap S| < \varsigma$. This implies in some for loop iteration, $C \in \mathcal{C}$ on line 7.

Let h be the heavy edge, in G_S, of the broken cycle C. We now have two cases:

Case 1: $h \in S$. In this case we have that

$$W(h) + w(h) > \underbrace{\sum_{e \in C \setminus S} w(e)}_{(1)} + \sum_{e \in S} W(e) + w(e).$$

On line 8 we found a cycle C_1 that minimized (1). Thus, since C is broken in G_S, C_1 is also broken in G_S, and so must be covered by some edge in $S_W \setminus S$. Hence, we added some edge in $S_W \setminus S$ to P.

Case 2 $h \notin S$. In this case h has the maximum weight of all edges in $C \setminus s$. We have that

$$\underbrace{w(h) - \sum_{e \in C \setminus (S \cup \{h\})} w(e)}_{(2)} > \sum_{e \in S} W(e) + w(e).$$

On line 9 we found a cycle C_2 maximizing (2). Thus, if C is broken in G_S, then C_2 is broken in G_S, and so must be covered by some edge in $S_W \setminus S$. Hence, we added some edge in $S_W \setminus S$ to P. □

Lemma 2.14. Any time we call F, we have that $|P| \leq 2\varsigma|S|^{\varsigma}$

Proof. Note $|P|$ is upper bounded by ς multiplied by the number of chordless cycles we add. If the conditional on line 4 is true then we add only a single chordless cycle to P. Otherwise, for each $s \subseteq S$ such that $|s| \leq \varsigma - 1$ we find two cycles. There are at most

$$\sum_{i=1}^{\varsigma-1} \binom{|S|}{i} \leq \sum_{i=1}^{\varsigma-1} |S|^i \leq |S|^{\varsigma}$$

many such subsets, and thus we add at most $2|S|^{\varsigma}$ many cycles, implying the claim. □

Theorem 2.15. For any fixed constant ς, Algorithm 2 is an FPT algorithm for $\mathrm{MR}(G, \mathbb{R})$ for any $G \in F_{\varsigma}$, when parameterized by OPT. The running time is $\Theta((2\varsigma OPT^{\varsigma})^{OPT+1} n^{\varsigma})$.

Proof. FPTWRAPPER iteratively calls $F(G, \emptyset, k)$ for increasing values of k until it

returns a non-NULL value. First, we argue that while $k < OPT$, $F(G, \emptyset, k)$ will return NULL. In the initial call to F, we have $S = \emptyset$. F then adds exactly one edge in each recursive call until $|S| = k$, at which point it returns VERIFIER(G, S). Thus, as $k < OPT$, by proposition 2.7, NULL is returned.

Now we argue that when $k = OPT$ an optimal solution is returned. Fix any optimal solution W and its support S_W to the given instance G. By Lemma 2.13, if $S \subsetneq S_W$ (which is true initially as $S = \emptyset$) then at least one edge in $S_W \setminus S$ is added to P. Thus, as F makes a recursive call to $F(G, S \cup \{e\}, k)$ for every edge $e \in P$, in at least one recursive call an edge of S_W is added to S. Thus there is some path in the tree of recursive calls to F in which all $k = OPT$ edges from S_W are added, at which point F returns VERIFIER(G, S), which returns an optimal solution by proposition 2.7. (Note this recursive call may not be reached, if a different optimal solution is found first.)

Now we consider bounding the running time. Observe that in each call to F, a set P is constructed, and then recursive calls to $F(G, S \cup \{e\}, k)$ are made for each $e \in P$. By Lemma 2.14, $|P| \leq 2\varsigma|S|^{\varsigma} \leq 2\varsigma k^{\varsigma}$ at all times. So in the tree of all recursive calls made by any initial call to $F(G, \emptyset, k)$, the branching factor is always bounded by $2\varsigma k^{\varsigma}$, and the depth is k. Thus there are $O((2\varsigma k^{\varsigma})^k)$ nodes in our recursion tree.

Now we bound the time needed for each node in the recursion tree. If VERIFIER is called then it takes $O(T_{\text{APSP}})$ time by proposition 2.7. Otherwise, note that there are $O(n^{\varsigma})$ chordless cycles. Thus it takes $O(\varsigma n^{\varsigma})$ time to enumerate and check them on line 4. Similarly $|\mathcal{C}| = O(n^{\varsigma})$ on line 7, and so the run time of each iteration of the for loop is $O(\varsigma n^{\varsigma})$. There are $O(|S|^{\varsigma}) = O(k^{\varsigma})$ iterations of the for loop, thus the total time per node is $O(\varsigma k^{\varsigma} n^{\varsigma})$.

Thus the total time for each call to $F(G, \emptyset, k)$ is $O((2\varsigma k^{\varsigma})^k \varsigma k^{\varsigma} n^{\varsigma}) = O((2\varsigma k^{\varsigma})^{k+1} n^{\varsigma})$.

Since FPTWrapper calls $F(G, \emptyset, k)$ for $k = 1, \ldots, OPT$, the overall running time is

$$O\left(\left(\sum_{k=1}^{OPT}(2\varsigma k^{\varsigma})^{k+1}\right) \cdot n^{\varsigma}\right) = O((2\varsigma OPT^{\varsigma})^{OPT+1} n^{\varsigma}). \qquad \square$$

As lemma 2.13 holds for any optimal solution, the bound on the recursion tree size in the above proof actually bounds the number of optimal solutions.

Corollary 2.16. If $G \in F_{\varsigma}$ then there are at most $(2\varsigma OPT^{\varsigma})^{OPT}$ subsets $S \subset E$ such that S is the support of an optimal solution to $\mathrm{MR}(G, \mathbb{R})$.

Remark 2.17. Using the approximation-preserving reduction from $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to $\mathrm{MR}(G, \mathbb{R})$ in Theorem 2.8, the above also yields an FPT for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$. This holds since the reduction does not change the optimal solution size, nor ς as it only adds triangles. Alternatively, the above algorithm can be carefully modified to consider light covering broken cycles.

2.6 Approximation Algorithms

In this section we present approximation algorithms for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$.

By Theorem 2.6, the support of an optimal solution to $\mathrm{MR}(G, \mathbb{R})$ is a minimum cardinality regular cover of all broken cycles. This naturally defines a hitting set instance $(E, \mathcal{C})$, where the ground set E is the edges from G, and $\mathcal{C}$ is the collection of the subsets of edges determined by broken cycles. Unfortunately, constructing $(E, \mathcal{C})$ explicitly is infeasible as there may be an exponential number of broken cycles. In general just counting the number of paths in a graph is #P-Hard *Valiant* (1979), though it is known how to count paths of length up to roughly $O(\log n)$ using color-coding. (See *Alon and Gutner* (2010); *Brand et al.* (2018) and references therein.) Moreover, observe our situation is more convoluted as we wish to count only paths corresponding to broken cycles.

Despite these challenges, we argue there is sufficient structure to at least roughly apply the standard greedy algorithms for hitting set. Our first key insight, related to insight (i), is:

(ii) One can always find *some* broken cycle, if one exists, by finding any edge whose weight is more than the shortest path length between its endpoints (using APSP).

Thus we have a polynomial time oracle, returning an arbitrary set in $\mathcal{C}$. Recall the greedy algorithm for hitting set, which repeatedly picks an arbitrary uncovered set, and adds all its elements to the solution. If $L = \max_{c \in \mathcal{C}} |c|$ is the largest set size, this gives an L-approximation, as each time we take the elements of a set, we get at least one element of the optimal solution. Below we apply this approach to approximate $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$.

We would prefer, however, to have an oracle for the number of broken cycles that an edge $e \in E$ participates in as using such an oracle would yield an $O(\log n)$-approximation algorithm for $\mathrm{MR}(G, \mathbb{R})$ (regardless of the size of L) by running the standard greedy algorithm for hitting set which repeatedly selects the element that hits the largest number of uncovered sets. Towards this end, we have the following key insight:

(iii) We can find the *most* broken cycle (i.e., with maximum deficit) and, more importantly, count how many such maximum deficit cycles each edge is in.

To argue that insight (iii) is true, first we observe that the cycle with the largest deficit value corresponds to a shortest path. This in turn, argued over several lemmas, allows us to quickly get a count when restricting to such cycles. Thus, if κ denotes the number of distinct cycle deficit values, the above insight implies an $O(\kappa \log n)$-approximation, by breaking the problem into κ hitting set instances, where for each instance we can run the greedy algorithm.

Algorithm 3 Short Path Cover (SPC) for $\mathrm{MR}(G, \mathbb{R})$

function $\mathrm{SPC}(G = (V, E, w))$
  $H = (V_H = V, E_H = E, w_H = w)$
  while True **do**
    $d, P = \mathsf{APSP}(H)$
    if $\exists\ e = (u, v) \in E_H$ such that $w(e) > d(u, v)$ **then** $E_H = E_H \setminus (P(u, v) \cup \{e\})$
    else return $\textsc{Verifier}(G, E \setminus E_H)$

2.6.1 L-approximation

In this section, we consider the problems defined by restricting $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to the subset of instances where the largest number of light edges in a broken cycle is L. We present an $(L + 1)$-approximation algorithm for $\mathrm{MR}(G, \mathbb{R})$ which runs in $O(T_{\mathrm{APSP}} \cdot OPT)$ time, which also will imply an L-approximation for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ with the same running time.

As mentioned above, the main idea comes from insight (ii). In particular, the following algorithm, Short Path Cover (SPC), can be easily understood by viewing it as running the standard L-approximation for the corresponding instance $(E, \mathcal{C})$ of hitting set, where we have an oracle for finding a set $c \in \mathcal{C}$. In the following, APSP is a subroutine returning a shortest path distance function $d(u, v)$, and a function $P(u, v)$ giving the set of edges along *any* shortest path from u to v.

Theorem 2.18. SPC gives an $(L+1)$-approximation for $\mathrm{MR}(G, \mathbb{R})$ in $O(T_{\mathrm{APSP}} \cdot OPT)$ time.

Proof. First, note that if there is a broken cycle in H, then for some edge $e = (u, v)$, $w(e) > d(u, v)$, and moreover, in this case $P(u, v) \cup \{e\}$ is a broken cycle. Thus, when the algorithm terminates there are no broken cycles in H. Also, for any broken cycle in G, if all of its edges are still in H, then it will be a broken cycle in H. Thus, when the algorithm terminates at least one edge from each broken cycle in G is in $E \setminus E_H$, which by Theorem 2.6 implies $E \setminus E_H$ is a valid support.

Note that removing edges does not create any new broken cycles, thus, any broken cycle in H is also a broken cycle in G. Thus, the support of any optimum solution must contain at least one edge from each broken cycle in H (again by Theorem 2.6), and so every time we remove the edges of a broken cycle $P(u,v) \cup \{e\}$, we remove at least one optimum edge. As the largest broken cycle length is $L+1$, this implies overall we get an $(L+1)$-approximation. The same argument implies the while loop can get executed at most OPT times, and as APSP takes $O(T_{\text{APSP}})$ time, and line 5 takes $O(m)$ time, we obtain the running time in the theorem statement. □

Remark 2.19. If we modify SPC so that in line 5 we only remove $P(u,v)$ from E_H (rather than $P(u,v) \cup \{e\}$), then by the second part of Theorem 2.6, the same argument implies that SPC is an L-approximation for $\text{MR}(G, \mathbb{R}_{\geq 0})$ that runs in $O(T_{\text{APSP}} \cdot OPT)$ time.

Remark 2.20. Theorem 2.11 restricts $\text{MR}(G, \mathbb{R}_{\geq 0})$ and $\text{MR}(G, \mathbb{R})$ to integer instances with max weight L, implying any broken cycle has $\leq L$ edges. As this is a subset of the instances here, SPC is an L or $L+1$ approx for instances that are hard to approximate within $\Omega(\sqrt{L})$.

2.6.2 $O(\kappa \log n)$-approximation

Using insight (iii), our approach is to iteratively cover cycles by decreasing deficit value, ultimately breaking the problem into multiple hitting set instances. We present the algorithm for $\text{MR}(G, \mathbb{R})$ first and then remark on the minor change needed to apply it to $\text{MR}(G, \mathbb{R}_{\geq 0})$.

For any pair of vertices $s, t \in V$, let $d(s,t)$ denote their shortest path distance in G, and $\#\mathsf{sp}(s,t)$ denote the number of shortest paths from s to t. It is straightforward to show that $\#\mathsf{sp}(s,t)$ can be computed in $O(m+n)$ time given all $d(u,v)$ values have been precomputed.

Lemma 2.21 (Proof in Appendix A.2)**.** Let G be a positively weighted graph, where for all pairs of vertices u, v one has constant time access to the value $d(u, v)$. Then for any pair of vertices s, t, the value $\#\mathsf{sp}(s, t)$ can be computed in $O(m + n)$ time.

Recall that for a broken cycle C with heavy edge h, the deficit of C is $\delta(C) = w(h) - \sum_{e \in (C \setminus h)} w(e)$. Moreover, $\delta(G)$ denotes the maximum deficit over all cycles in G. For any edge e, define $N_h(e, \alpha)$ to be the number of distinct broken cycles of deficit α whose heavy edge is e. Similarly, let $N_l(e, \alpha)$ denote the number of distinct broken cycles with deficit α which contain the edge e, but where e is not the heavy edge. While for general α it is not clear how to even approximate $N_l(e, \alpha)$ and $N_h(e, \alpha)$, we argue that when $\alpha = \delta(G)$ these values can be computed exactly.

Lemma 2.22. For any edge $e = (s, t)$, if $w(e) = d(s, t) + \delta(G)$ then $N_h(e, \delta(G)) = \#\mathsf{sp}(s, t)$, and otherwise $N_h(e, \delta(G)) = 0$.

Proof. If $w(e) \neq d(s, t) + \delta(G)$, then as $\delta(G)$ is the maximum deficit over all cycles, it must be that $w(e) < d(s, t) + \delta(G)$, which in turn implies any broken cycle with heavy edge e has deficit strictly less than $\delta(G)$. Now suppose $w(e) = d(s, t) + \delta(G)$, and consider any path $p_{s,t}$ from s to t such that e together with $p_{s,t}$ creates a broken cycle with heavy edge e. If $p_{s,t}$ is a shortest path then $w(e) - w(p_{s,t}) = w(e) - d(s, t) = \delta(G)$, and otherwise $w(p_{s,t}) > d(s, t)$ and so $w(e) - w(p_{s,t}) < w(e) - d(s, t) = \delta(G)$. Thus $N_h(e, \delta(G)) = \#\mathsf{sp}(s, t)$ as claimed. □

As G is undirected, every edge $e \in E$ correspond to some unordered pair $\{a, b\}$. However, often we write $e = (a, b)$ as an ordered pair, according to some fixed arbitrary total ordering of all the vertices. We point this out to clarify the following statement.

Lemma 2.23. Fix any edge $e = (s, t)$, and let $X = \{f = (a, b) \mid w(f) = d(a, s) + w(e) + d(t, b) + \delta(G)\}$, and $Y = \{f = (a, b) \mid w(f) = d(b, s) + w(e) + d(t, a) + \delta(G)\}$.

Then

$$N_l(e, \delta(G)) = \left(\sum_{(a,b) \in X} \#\mathsf{sp}(a, s) \cdot \#\mathsf{sp}(t, b) \right) + \left(\sum_{(a,b) \in Y} \#\mathsf{sp}(b, s) \cdot \#\mathsf{sp}(t, a) \right).$$

Proof. Consider any broken cycle C containing $e = (s, t)$, with heavy edge $f = (a, b)$ and where $\delta(C) = \delta(G)$. Such a cycle must contain a shortest path between a and b, as otherwise it would imply $\delta(G) > \delta(C)$. Now if we order the vertices cyclically, then the subset of C's vertices $\{a, b, s, t\}$, must appear either in the order a, s, t, b or b, s, t, a. In the former case, as the cycle must use shortest paths, $w(f) = d(a, s) + w(e) + d(t, b) + \delta(G)$, and the number of cycles satisfying this is $\#\mathsf{sp}(a, s) \cdot \#\mathsf{sp}(t, b)$. In the latter case, $w(f) = d(b, s) + w(e) + d(t, a) + \delta(G)$, and the number of cycles satisfying this is $\#\mathsf{sp}(b, s) \cdot \#\mathsf{sp}(t, a)$. Note also that the set X from the lemma statement is the set of all $f = (a, b)$ satisfying the equation in the former direction, and Y is the set of all $f = (a, b)$ satisfying the equation in the later direction. Thus summing over each relevant heavy edge in X and Y, of the number of broken cycles of deficit $\delta(G)$ which involve that heavy edge and e, yields the total number of broken cycles with deficit $\delta(G)$ containing e as a light edge. □

Corollary 2.24 (Appendix A.2)**.** Given constant time access to $d(u, v)$ and $\#\mathsf{sp}(u, v)$ for any vertices u and v, $N_h(e, \delta(G))$ can be computed in $O(1)$ time and $N_l(e, \delta(G))$ in $O(m)$ time.

Theorem 2.25. For any positive integer κ, consider the set of $\mathrm{MR}(G, \mathbb{R})$ instances where the number of distict deficit values is at most κ, i.e., $|\{\delta(C) \mid C \text{ is a cycle in } G\}| \leq \kappa$. Then Algorithm 4 gives an $O((T_{\mathrm{APSP}} + m^2) \cdot OPT \cdot \kappa \log n)$ time $O(\kappa \log n)$-approximation.

Proof. Observe that the algorithm terminates only when $\delta(G) = 0$, i.e., only once there are no broken cycles left. As no new edges are added, and weights are never

Algorithm 4 Finds a valid solution for $\mathrm{MR}(G, \mathbb{R})$.

```
1: function APPROX(G = (V, E, w))
2:     Let S = ∅
3:     while True do
4:         For every pair s, t ∈ V compute d(s, t)
5:         Compute δ(G) = max_{e=(s,t)∈E} w(e) − d(s, t)
6:         if δ(G) = 0 then return VERIFIER(G, S)
7:         For every edge (s, t) ∈ E compute #sp(s, t)
8:         For every e ∈ E compute count(e) = N_h(e, δ(G)) + N_l(e, δ(G))
9:         Set f = arg max_{e∈E} count(e)
10:        Update S = S ∪ {f} and G = G \ f
```

modified, this implies that when the algorithm terminates it outputs a valid regular cover S. (The algorithm must terminate as every round removes an edge.) Therefore, by Theorem 2.6, S is a valid $\mathrm{MR}(G, \mathbb{R})$ support, and so we only need to bound its size.

Let the edges in $S = \{s_1, \ldots, s_k\}$ be indexed in increasing order of the loop iteration in which they were selected. Let $G_1, \ldots, G_{k+1}$ be the corresponding sequence of graphs produced by the algorithm, where $G_i = G \setminus \{s_1, \ldots, s_{i-1}\}$. Note that for all i, $G_i = (V, E_i)$ induces a corresponding instance of hitting set, $(E_i, \mathcal{C}_i)$, where the ground set is the set of edges from the $\mathrm{MR}(G, \mathbb{R})$ instance G_i, and $\mathcal{C}_i = \{E_i(C) \mid C \text{ is a broken cycle in } G_i\}$ (where $E_i(C)$ is the set of edges in C).

Let $D = \{\delta(C) \mid C \text{ is a cycle in } G\}$, where by assumption $|D| \leq \kappa$. Note that any cycle C in any graph G_i, is also a cycle in G. Thus as we never modify edge weights, $\delta(G_1), \ldots, \delta(G_{k+1})$ is a non-increasing sequence. Moreover $X = \{\delta(G_i)\}_i \subseteq D$, and in particular $|X| \leq \kappa$. For a given value $\delta \in X$, let $G_\alpha, G_{\alpha+1}, \ldots, G_\beta$ be the subsequence of graphs with deficit δ (which is consecutive as the deficit values are non-increasing). Observe that for all $\alpha \leq i \leq \beta$, the edge s_i is an edge from a cycle with deficit $\delta = \delta(G_i)$. So for each $\alpha \leq i \leq \beta$, define a sub-instance of hitting set $(E'_i, \mathcal{C}'_i)$, where E'_i is the set of edges in cycles of deficit δ from G_i, and $\mathcal{C}'_i$ is the family of sets of edges from each cycle of deficit δ in G_i.

The claim is that for the hitting set instance $(E'_\alpha, \mathcal{C}'_\alpha)$, that $\{s_\alpha, \ldots, s_\beta\}$ is an $O(\log n)$ approximation to the optimal solution. To see this, observe that for any $\alpha \leq i \leq \beta$ in line 8, $count(e)$ is the number of times e is contained in a broken cycle with deficit $\delta = \delta(G_i)$, as by definition $N_h(e, \delta(G_i))$ and $N_l(e, \delta(G_i))$ count the occurrences of e in such cycles as a heavy edge or light edge, respectively. Thus s_i is the edge in E'_i which hits the largest number of sets in $\mathcal{C}'_i$, and moreover, $(E'_{i+1}, \mathcal{C}'_{i+1})$ is the corresponding hitting set instance induced by removing s_i and the sets it hit from $(E'_i, \mathcal{C}'_i)$. Thus $\{s_\alpha, \ldots, s_\beta\}$ is the resulting output of running the standard greedy hitting set algorithm on $(E'_\alpha, \mathcal{C}'_\alpha)$ (that repeatedly removes the element hitting the largest number of sets), and it is well known this greedy algorithm produces an $O(\log n)$ approximation.

The bound on the size of S now easily follows. Specifically, let $I = \{i_1, i_2, \ldots, i_{|X|}\}$ be the collection of indices, where i_j was the first graph considered with deficit $\delta(G_{i_j})$. By the above, S is the union of the $O(\log n)$-approximations to the sequence of hitting set instance $(E'_{i_1}, \mathcal{C}'_{i_1}), \ldots, (E'_{i_{|X|}}, \mathcal{C}'_{i_{|X|}})$. In particular, note that for all i_j, $(E'_{i_j}, \mathcal{C}'_{i_j})$ is a hitting set instance induced from the removal of a subset of edges from the initial hitting set instance $(E_1, \mathcal{C}_1)$, and then further restricted to sets from cycles with a given deficit value. Thus the size of the optimal solution on each of these instances can only be smaller than on $(E_1, \mathcal{C}_1)$. This implies that the total size of the returned set S is $O(OPT \cdot |X| \log n) = O(OPT \cdot \kappa \log n)$.

As for the running time, first observe that by the above, there are $O(OPT \cdot \kappa \log n)$ while loop iterations. Next, the single call to VERIFIER in line 6 takes $O(T_{\text{APSP}})$. For a given loop iteration, computing all pairwise distances in line 4 also takes $O(T_{\text{APSP}})$ time. Computing the graph deficit in line 5 can then be done in $O(m)$ time. For any given vertex pair s, t, computing $\#\mathsf{sp}(s, t)$ takes $O(m + n)$ time by Lemma 2.21. Thus computing the number of shortest paths over all edges in line 7 takes $O(m^2 + mn)$ time. For each edge e, by Corollary 2.24, $count(e) = N_h(e, \delta(G)) + N_l(e, \delta(G))$ can be

computed in $O(m)$ time, and thus computing all counts in line 8 takes $O(m^2)$ time. As the remaining steps can be computed in linear time, each while loop iteration in total takes $O(T_{\text{APSP}} + mn + m^2) = O(T_{\text{APSP}} + m^2)$ time, thus implying the running time bound over all iterations in the theorem statement. □

Remark 2.26. If we modify line 8 to instead set $count(e) = N_l(e, \delta(G))$, by Theorem 2.6, we get the same result for $\text{MR}(G, \mathbb{R}_{\geq 0})$. If instead we used the reduction from $\text{MR}(G, \mathbb{R}_{\geq 0})$ to $\text{MR}(G, \mathbb{R})$ of Theorem 2.8, the graph size increases by a linear factor, giving a slower run time.

CHAPTER III

Manifold Repair In Presence of Missing Data

3.1 Introduction

Many real world data sets can be reasonably modeled as low dimensional manifolds embedded in much higher dimensional spaces, and for these models, a class of techniques play a crucial role in revealing or learning these intrinsic manifolds. This class includes ISOMAP *Tenenbaum et al.* (2000b), Local Linear Embedding (LLE) *Roweis and Saul* (2000), HESSIAN-LLE *Donoho and Grimes* (2003), MAXIMUM VARIANCE UNFOLDING *Weinberger and Saul* (2004), KNN-DIFFUSION *Coifman and Lafon* (2006), and LAPLACIAN EIGENMAP *Belkin and Niyogi* (2003). All of these algorithms have the basic structure shown in Figure 3.1. Given data, we compute a distance matrix (alternatively, we are given a distance matrix), from which we determine neighborhoods about each data point. Some of these algorithms find the K closest points to each data point and others determine the data points in an ϵ neighborhood about each data point. Each algorithm then uses this neighborhood information to compute local Euclidean coordinates in some fashion (i.e., to learn the underlying manifold) and, thus, to determine a low-dimensional representation for the data.

Each of these algorithms assumes that the given or computed distance matrix adheres to a metric. To see the importance of using a distance matrix that satisfies a metric with these algorithms, we show in Figure 3.2 the impact upon the manifold

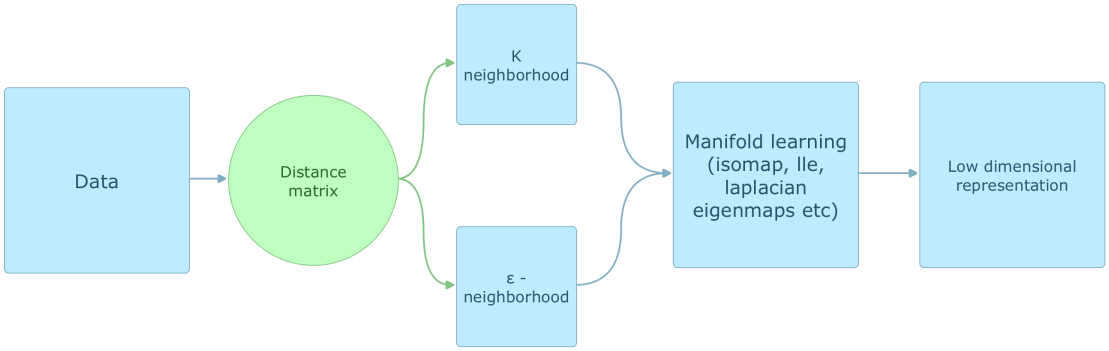
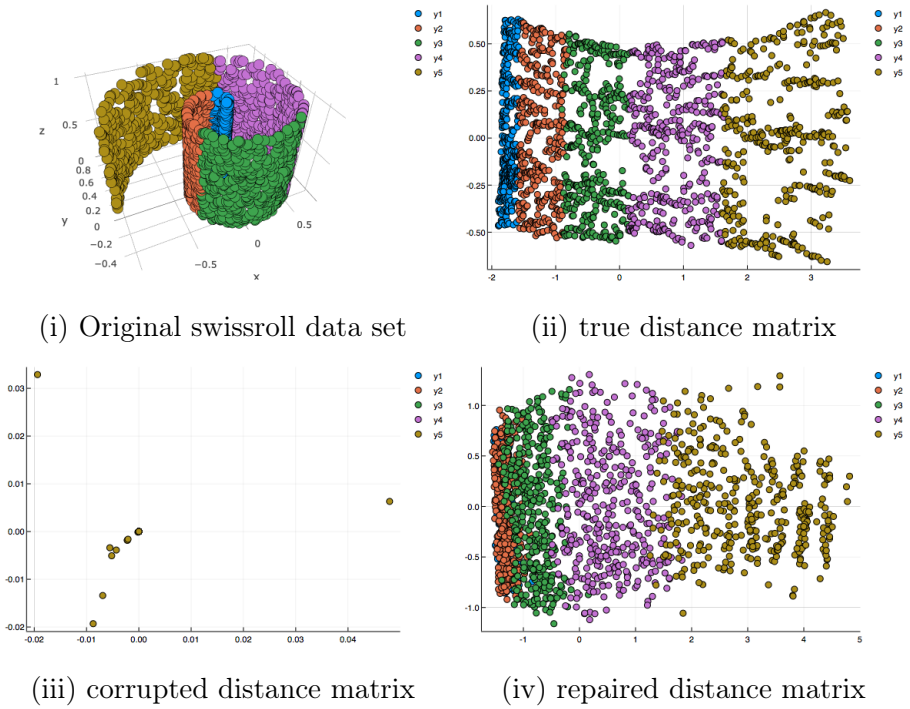


Figure 3.1: General Manifold learning procedure

returned by ISOMAP when we corrupt or perturb the distance matrix so that it no longer satisfies a metric. Note that these algorithms are robust to (small) perturbations in the *data* but not in the *distances* among the data points. In the top left figure 3.2i, we have the original swissroll dataset with 2,000 points. It is a two dimensional manifold embedded in three dimensions. When we run ISOMAP on the true distance matrix, we see in the upper right figure 3.2ii, an "unrolled" version of the intrinsic manifold. To corrupt the distance matrix, we add i.i.d. Gaussian noise $\sim \mathcal{N}(0, 0.01)$ to each non diagonal entry of the distance matrix. We then replace all negative entries with zero and preserve symmetry by averaging the corrupted distance matrix with its transpose. The perturbed distances may not satisfy the triangle inequality and, hence, the corrupted distances may not adhere to a metric.

The lower left figure 3.2iii is the embedding from the corrupted distance matrix. It is considerably different from the original embedding, points are missing, and there is no apparent lower dimensional manifold structure at all. Because the distances do not satisfy a metric, the points all collapse to one location and cannot be distinguished in the figure. In the lower right figure 3.2iv, we first repair the corrupted distance matrix, using a metric repair algorithm from *Gilbert and Jain* (2017), and then embed the data using ISOMAP. The resulting embedding is much closer to the original embedding, with some minor distortion. Thus, we can see that unless the distance matrix satisfies

a metric, dimension reduction or manifold embedding algorithms fail catastrophically.



(i) Original swissroll data set (ii) true distance matrix

(iii) corrupted distance matrix (iv) repaired distance matrix

Figure 3.2: (a) The original swissroll data set (2000 points) and the results from ISOMAP for: (b) the original distance matrix, (c) the corrupted distance matrix, and (d) the repaired distance matrix.

3.1.1 Problem Set Up

This example illustrates the main problem we address: if we have either missing data or missing or corrupted entries in the distance matrix and we assume that the data come from an intrinsic low dimensional manifold, compute a low dimensional representation of the imputed or corrected data set. One such approach for missing data is to complete the data matrix using a matrix completion algorithm. These algorithms assume that the data matrix is approximately low rank and fills in the missing entries accordingly. These algorithms fit the data to a *linear* subspace rather than an intrinsically *nonlinear* embedding and may miss key features of the data. Other methods that learn the intrinsic low dimensional structure in a data set impute missing data values in the original space but one cannot use other, potentially better,

algorithms for the embeddings. Our method aims to repair the distance matrix of the data, so as to extend existing embedding algorithms such as ISOMAP and LAPLACIAN EIGENMAPS to handle missing data or corrupted distances.

To be precise, let X be the high-dimensional data set and D the distance or dissimilarity matrix amongst the data points, and consider the following four problem scenarios:

1. The data set X has corrupted entries,
2. The data set X has missing entries,
3. The dissimilarity matrix D has corrupted entries, or
4. The dissimilarity matrix D has missing entries.

In the first model X has added noise and many of the traditional algorithms are robust and produce satisfactory results. Hence, we shall focus on the second scenario where X has missing entries. The second scenario covers the last two (as missing data corrupt the distances between points) and we sketch the applications of our methods to these scenarios and leave a more in depth analysis of those models as future work.

3.1.2 Previous work

There are two main approaches to filling in missing manifold data that we summarize below. Both methods strive to learn, in an unsupervised fashion, a representation of the data and then to use that learned representation to fill in the missing values. The first method employs a low-dimensional representation as an intermediary step in the overall data representation while the second method directly learns a low-dimensional representation. We will also discuss how matrix completion algorithms could be used for model 4

Non-linear Principle Component Analysis (nlPCA) *Scholz et al.* (2005). This method is a non-linear analog to principle component analysis. The idea is to use a five-layer neural network with architectural dimensions $n \times m \times d \times m \times n$, where n is the input dimension, m is usually bigger than n, and d is the desired dimension of the embedding. We train the neural network to learn the identity map so that the middle layer with d neurons is the low-dimensional representation of the data. To extend to missing data, the network is also trained to reproduce the input data but during the training procedure, any gradients that depend on missing values are disregarded. Then, to fill in the missing data, the data with missing values is input to the network and the output values are used to fill in any missing data.

Missing Data Recovery through Unsupervised Regression (mDRUR) *Á. Carreira-Perpiñán and Lu* (2011). The second major method is the missing data recovery through unsupervised learning (mDRUR). Similarly to the nlPCA algorithm, this algorithm is an extension of a dimensionality reduction algorithm known as dimensionality reduction through unsupervised learning (DRUR). Using the notation of *Á. Carreira-Perpiñán and Lu* (2011), let Y be the high dimensional representation of the data and X the low dimensional representation. Then we have two maps f, F such that $Y = f(X)$ and $X = F(Y)$. To learn the low-dimensional representation, we minimize

$$\arg\min_{X,f,F} \|Y - f(X)\|_F^2 + \|X - F(Y)\|_F^2 + \lambda_f R(f) + \lambda_F R(F)$$

where $R(f)$ and $R(F)$ are regularization terms. To fill in missing data, we first use a linear matrix completion method to fill in the missing values, then we use a spectral method to compute X. Finally, we learn the low-dimensional representation as above and optimize over the missing values of Y to fill in the missing data.

Both algorithms are primarily dimensionality reduction algorithms and, as such, we must first fix the lower dimension and then solve an optimization problem. In most real world applications we do not know the optimal low dimension and, hence, in order to fill in the missing data, we must first find this dimension and then fill in the data, rather than separating these two tasks. Furthermore, the imputed values depend on the computed, specific reduced representation; we cannot avail ourselves of a variety of dimension reduction algorithms and obtain what we hope to be a consistent or robust approximation of the missing values.

Euclidean distance matrix and metric completion. One might be tempted to restrict our distances to Euclidean distances as it is well known *Gower* (1985b) that Euclidean distance matrices (with squared Euclidean distance entries) are low rank matrices with rank at most $r + 2$ if r is the dimension of the space in which the points lie. Hence, the problem of Euclidean matrix completion can be solved using standard low rank matrix completion algorithms. Additionally, this specific problem has been further studied with many successful algorithms in *Al-Homidan and Wolkowicz* (2005); *Bakonyi and Johnson* (1995); *ren Fang and O'Leary* (2012).

In some cases we want our original data to follow a non-Euclidean metric. For example, it is has been shown that for the MNIST dataset if we use the tangent distance metric instead of Euclidean, then k nearest neighbor classifiers have better performance. In this case, if we have the local neighborhood information for the data we can still run ISOMAP and other various algorithms to get lower dimensional representations. Gower *Gower* (1985b) showed that these matrices are either low rank (with the same low rank condition as before) or have full rank. In the case that they have full rank, we can no longer use matrix completion algorithms.

Even when we have a low rank matrix, we only know we can complete these matrices with high probability if the entries present are sampled according to a

certain distribution. Finally, even if we can successfully apply these matrix completion algorithms, there are no guarantees that the resulting distance matrices satisfy a metric.

3.1.3 Our approach and contributions

We focus on the second scenario. We separate the problem into three steps. First we estimate distances between the data points, we then correct these distances so that they adhere to a metric, finally we run a suitable dimension reduction algorithm. We use the Increase Only Metric Repair (IOMR) algorithm in Gilbert and Jain *Gilbert and Jain* (2017) to repair the inaccurate distance matrices. As we can see in Figure 3.1, all of the dimension reduction algorithms depend on the local distances and not on the actual data points themselves. Hence, filling in the missing data is both costly and unnecessary. Instead, we first estimate the distance matrix from the incomplete data, then we correct it. This approach has two advantages over the previous methods

- No parameters: Our algorithm has no parameters that need tuning. Hence making it faster and easier to train compared to nlPCA and mDRUR. Additionally, our algorithm is quadratic is the number of dimensions. Hence, its performance scales well with number of dimensions.

- Accuracy: The manner in which we estimate the distances and then correct them is geared to exactly preserve the local structure.

Remark 3.1. For the most general version of scenario three, fast approximation algorithms for the general metric repair problem do not currently exists. A class of approximation algorithms can be found in Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b).

The rest of the paper is organized as follows, Section 2 presents background knowledge, Section 3 presents our algorithms for correcting a corrupted distance

matrix so as to produce an accurate low dimensional embedding of a data set. We focus specifically on missing data. Section 4 provides our experimental results.

3.2 Background

3.2.1 Manifolds and Geodesic distances

Definition 3.2. $M \subset \mathbb{R}^n$ is called a d dimensional manifold is for all $x \in M$ there exists an $\epsilon > 0$ such that there if a continuous bijective function f from $N = \{y \in M : \|x - y\| < \epsilon\}$ to $\mathbb{R}^d$ such that the inverse is continuous as well.

Intuitively the above definition says that if we look at a d-dimensional manifold M and if we zoom in close enough to any point then it looks like we are in $\mathbb{R}^d$. For example the swiss roll (from the introduction) is a two-dimensional manifold because near any point it looks like a plane. As we can see from the definition itself the local structure of a manifold is important. Hence, all dimensionality reduction algorithms start by computing the local neighborhood of each point. This is done in one of two ways:

- determine the k nearest neighbors, or
- compute the neighbors within some distance ϵ

These local neighborhoods then overlap to describe the general manifold structure. Hence, having this correct local structure is crucial for the success of any of the dimensionality reduction algorithms.

In this paper we will we focus on using Isomap (though we could have picked any of the other algorithms). For Isomap once we have the graph (i.e., two data points are adjacent if one is in the local neighborhood of the other) we then compute the shortest distance (along this graph) between all the points. This usually done using the Floyd Warshall algorithm. We store these distances in a matrix $\tilde{D}$. These

distances are known as geodesic distances. They are the distances we want between our data points in the low dimensional representation.

3.2.2 Multidimensional Scaling

The technique used to go from the inferred distance matrix $\tilde{D}$ to an actual embedding is called multidimensional scaling (MDS) and we include this discussion to complete Section 9.6. An important point to note is that the points we recover are not unique, since rotating and translating the embedding will not change the pairwise distances between the points.

Suppose we have n by n distance matrix $\tilde{D}$. We want to find points $x_1, \ldots, x_n$ in some d dimensional Euclidean space such that $\text{dist}(x_i, x_j) = \tilde{D}_{ij} = \tilde{D}_{ji}$. Let us define

$$S = -\frac{1}{2}\left(I - \frac{1_n 1_n^T}{n}\right)\tilde{D} \circ \tilde{D}\left(I - \frac{1_n 1_n^T}{n}\right)$$

where 1_n is the n dimensional vector of all 1s and $\circ$ is the Hadamard product, which we multiply the two matrices coordinate wise.. Then using the fact that the points are translation invariant (we can assume the centroid of $x_1, \ldots, x_n$ is the origin) using which it can be shown that S_{ij} would then be $x_i \cdot x_j$. This matrix is now positive semi-definite, hence has an eigenvalue decomposition

$$S = U\Lambda U^T$$

Where Λ is a diagonal matrix of the eigenvalues. If we then define $X = U\Lambda^{0.5}$. Then this is our embedding. We get a d dimensional embedding by using only the biggest d eigenvalues.

Algorithm 5 IOMR Fixed

Require: $D \in \mathrm{Sym}_n(\mathbb{R}_{\geq 0})$

function IOMR-FIXED(D)
    $\hat{D} = D$
    for $k \leftarrow 1$ to n **do**
        for $i \leftarrow 1$ to n **do**
            $\hat{D}_{ik} = \max(\hat{D}_{ik}, \max_{j<i}(\hat{D}_{ij} - \hat{D}_{jk}))$
    return $\hat{D} - D$

3.2.3 Metric Repair

Gilbert and Jain in *Gilbert and Jain* (2017) defined the *sparse metric repair* problem. They define $\mathrm{Sym}_n(\mathbb{R}_{\geq 0})$ to be the set of positive real symmetric matrices. More generally, let us define $\mathrm{Sym}_n(S)$ to be the set of symmetric matrices with entries drawn from S. Then, for any matrix $D \in \mathrm{Sym}_n(\mathbb{R}_{\geq 0})$ we say it satisfies a metric if the diagonal of D is all 0s and for all i, j, k we have that $D_{ij} \leq D_{ik} + D_{kj}$.

The sparse metric repair problem seeks a solution to the following optimization problem: Given $D \in \mathrm{Sym}_n(\mathbb{R}_{\geq 0})$ and $S \subset \mathbb{R}$

$$\mathrm{argmin}\|P\|_0 \text{ s.t. } D + P \text{ is a metric and } P \in \mathrm{Sym}_n(S), \tag{3.1}$$

where $\|\cdot\|_p$ is the vector ℓ_0 is the pseudonorm that counts the number of non-zero entries. In *Gilbert and Jain* (2017), they define three variants of the corresponding to three different: $S = \mathbb{R}_{\leq 0}$ decrease only metric repair (DOMR), $S = \mathbb{R}_{\geq 0}$ increase only metric repair (IOMR), and $S = \mathbb{R}$, which is simply called metric repair or MR. In general, for any given set S we shall refer to problem as MR(S)

For the rest of paper we will focus on the increase only case and use the following algorithm from *Gilbert and Jain* (2017):

While Gilbert and Jain showed that IOMR-FIXED worked empirically, they could not provide any guarantees on its performance nor did they demonstrate how it could be used for actual applications. Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b)

provides a more in-depth analysis of a generalized problem. One of their results is to show that the problem of increase only metric repair is NP-Hard for even simple sets S. They also provide several approximation algorithms for this problem and more general variants.

As it was noted in the *Gilbert and Jain* (2017), one could relax Equation equation 3.1 to a convex optimization problem by minimizing $\|P\|_1$ instead of $\|P\|_0$. Gilbert and Jain used various different convex optimization methods, and while these methods produce satisfactory results, they were extremely slow. Hence, we stick with IOMR-Fixed. The speed of the optimization algorithms will not scale well with the number of data points as that the output solution $D' = D + P$ needs to be a metric. In particular, for any i, j, k we need the entries for D' to satisfy the triangle inequality. Thus, if we have n data points, we have $O(n^3)$ constraints. Even for simple applications where we have 1000 data points, the optimization problem has $O(10^9)$ constraints.

3.3 Metric repair on manifolds

In our model scenario, we are given an incomplete data set X and a matrix Q that specifies the support of the *known* entries. We present an algorithm to compute a (potentially corrupted) distance matrix from the incomplete data and then use metric repair to correct the perturbed distances. The idea is to ignore missing entries when calculating the distances between two points as in *Balzano and Bajwa* (2010). Thus, we have the following algorithm.

As we will see in Section 3.4, this algorithm works well in practice and, in this section, we provide theoretical analysis of its performance in a model setting. Before doing so, let us develop some intuition about the algorithm. To analyze how well the algorithm performs, supposed we had a probabilistic model with the following two assumptions:

Algorithm 6 MR-missing

Require: X Input data, Q support of the data ($Q_{ij} = 1$ if and only if X_{ij} is present. $Q_{ij} = 0$ otherwise)

1: **function** MR-MISSING(X,Q)
2: $D = zeros(n, n)$
3: **for** $i \leftarrow 1$ to n **do**
4: **for** $j \leftarrow 1$ to n **do**
5: $D_{ij} = (\sum_{k=1}^{n} Q_{ik} \cdot Q_{jk}(X_{ik} - X_{jk})^2)^{1/2}$
6: P = IOMR-Fixed(D)
7: ISOMAP$(D + P)$

1. if two data points are far apart, then our estimated distance is small with low probability; and,

2. if two data points are far and our estimated distance is small, then, with high probability, we increase this distance during metric repair.

Assuming the above two assumptions hold, we argue that MR-MISSING maintains the local structure of the data. When we compute the distance between two data points so as to ignore missing entries, the estimated distance is smaller than the true distance. Thus, if two points are initially close, then they remain close together. This is beneficial since the crucial structure for all manifold learning algorithms is the local distances. It could happen, that because data are missing, two points x, y that were not close together initially have small inferred distance. Then, by assumption (1), this would happen with small probability and, by assumption (2), we would fix this distance with high probability. Finally, since the metric has been repaired and we have, with high probability preserved the local structure of the data set, ISOMAP produces an embedding consistent with that of the full data set. That is, with high probability, the algorithm preserves the local structures and guarantees that all the distances adhere to a metric, and we conclude the low dimensional embeddings calculated with this repaired distance matrix to preserve most of the manifold structure of the data.

3.3.1 Theory Result

There are two steps to our algorithm. The first step is estimating distances between points with missing coordinates. The second is increasing these distances so that the distances adhere to a metric. In this subsection, we are going to analyze the effectiveness of MR-MISSING by showing that in the following model if two points are well separated then the distance estimated by step 1 is large with high probability.

In this model our data consists of two Gaussians clusters in $\mathbb{R}^n$ with means $\mu_1, \mu_2 \in \mathbb{R}$ (all coordinates for a data point from one cluster have the same mean) and covariances $\Sigma_1 = \Sigma_2 = 0.5I_n$, where I_n is the n dimensional identity matrix.

We assume that, for any data point x, each of its n coordinates is present independently with probability p. Under these conditions, we want to show that with high probability our algorithm preserves local neighborhoods.

For notational connivence let us define $d_p(x, y)$ to be the distance between x, y estimated by step 1 of MR-MISSING when each coordinate is present independently with probability p.

Before we can state and prove our result we need a few lemmas first

Lemma 3.3 (Birgé 2001, *Lucien* (2001))**.** For all $D \geq 1$, if $X = Z_1^2 + \ldots + Z_D^2$, where $Z_i \sim \mathcal{N}(\mu_i, 1)$, and $\lambda = \sum_{i=1}^{D} \mu_i^2$, then, all $0 < c < D + \lambda$, we have that

$$\Pr[X \leq c] \leq e^{-\frac{(D+\lambda-c)^2}{4(D+2\lambda)}}.$$

Lemma 3.4 (Hoeffding's Inequality)**.** If we have n i.i.d. variables $X_1, \ldots, X_n$ such that $X_i = 1$ with probability p and $X_i = 0$ with probability $1 - p$, then, for all $\epsilon > 0$, we have that

$$\Pr\left[\left|\sum_{i=1}^{n} X_i - pn\right| \leq \gamma n\right] \geq 1 - 2e^{-2\gamma^2 n}.$$

Lemma 3.3 allows us to bound the tail of a non-central χ-squared distribution and Lemma 3.4 allows us to bound the probability that we have too much data missing. Combining these two Lemmas, we have the following theorem.

Theorem 3.5. Suppose $X \sim \mathcal{N}(\mu_1 \mathbf{1}, 0.5I)$ and $Y \sim \mathcal{N}(\mu_2 \mathbf{1}, 0.5I)$ are two points in $\mathbb{R}^n$ such that each coordinate of X, Y is missing with probability p. If $\mu = \mu_1 - \mu_2$ and $q = p^2$, then, for all $q(1+\mu^2) > \epsilon > 0$ and $\frac{q(1+\mu^2)-\epsilon}{(1+\mu^2} > \gamma > 0$, we have that

$$\Pr[d_p(x,y) < \epsilon n] \leq e^{-2\gamma^2 n} + \left(e^{-\frac{((q-\gamma)(1+\mu^2)-\epsilon)^2}{4(q-\gamma)(1+2\mu^2)}} \right)^n.$$

Proof. Let $Z = X - Y$. Then $Z_1, \ldots, Z_n$ are i.i.d Gaussian random variables with mean $\mu = \mu_1 - \mu_2$ and variance 1. We know from MR-Missing that we use the entry Z_i to calculate the distance between X, Y if and only if both X_i and Y_i are present. This happens with probability p^2, which we define as $q = p^2$. Thus, we have the entry Z_i with probability q.

Let $q(1+\mu^2) > \epsilon > 0$. We want to show that the probability that the distance calculated by step 1 of MR-Missing is greater than ϵ is small. To do this, we use Hoeffding's inequality to divide into two cases, one in which we observe a large number of coordinates and one a few coordinates. We shall see that the case when we observe a few coordinates occurs with low probability and we obtain a bound on the distance. We will then see that for the case where we see a large number of coordinates, then the distance is large with high probability.

Let $0 < \gamma < \dfrac{q(1+\mu^2) - \epsilon}{(1+\mu^2)}$. Then, by definition of ϵ, we see that $q > \gamma > 0$. Let K be the number of entries we observe and, by Lemma 3.4, we have that

$$\Pr[K \leq (q-\gamma)n] \leq e^{-2\gamma^2 n}.$$

For notational convenience, let $D = (q-\gamma)n$ and consider two cases.

Case 1: Suppose $K \leq D$. That is, we observe fewer than D entries. Then, by

Hoeffding's Inequality, we know that this happens with probability at most $e^{-2\gamma^2 n}$.

Thus we have that

$$\Pr[d_p(x,y) < \epsilon n | K \leq D] \Pr[K \leq D] \leq e^{-2\gamma^2 n}$$

Case 2: Suppose $K \geq D$. That is, we observe a large number of entries. Let us condition on the actual value of K. Suppose that $K = k$ and we have observed entries $Z_{i_1}, \ldots, Z_{i_k}$. We know these entries are i.i.d. with mean μ and we want to bound

$$p_k := \Pr\left[\sum_{j=1}^{k} Z_{i_j}^2 \leq \epsilon n\right].$$

In this case we see that the overall probability that we have a small distance, given that $K \geq D$, is

$$\sum_{k=D}^{n} \Pr[K = k] p_k.$$

The next thing to observe is that p_k is monotone decreasing in k because each Z_i is non-negative with non-zero mean. Thus, we have the following upper bound

$$\begin{aligned}
\Pr[d_p(x,y) < \epsilon n | K \geq D] &\leq \sum_{k=D}^{n} \Pr[K = k] p_k \\
&\leq \sum_{k=D}^{n} \Pr[K = k] p_D \\
&\leq p_D.
\end{aligned}$$

Now, we can use our tail bound for the χ-squared distribution with $c = \epsilon n$, $D = (q - \gamma)n$, and $\lambda = D\mu^2$. Thus, we have

$$\begin{aligned}
p_D &\leq e^{-\frac{((q-\gamma)n(1+\mu^2) - \epsilon n)^2}{4(q-\gamma)n(1+2\mu^2)}} \\
&= \left(e^{-\frac{((q-\gamma)(1+\mu^2) - \epsilon)^2}{4(q-\gamma)(1+2\mu^2)}}\right)^n.
\end{aligned}$$

Combining both cases, we see that the probability that the distance calculated is less than ϵn is at most

$$\Pr[d_p(x,y) < \epsilon n] \leq e^{-2\gamma^2 n} + \left(e^{-\frac{((q-\gamma)(1+\mu^2)-\epsilon)^2}{4(q-\gamma)(1+2\mu^2)}} \right)^n.$$

□

Let us take a closer look at the effect of the various parameters on the above probability:

- *The dimension n of the data*: As n increases, we have an exponential decrease in the probability that two points from the two Gaussian clouds have distance smaller than ϵn. Thus, we expect our algorithm to work better for high dimensional data.

- *The mean squared distance between the clusters $\mu^2 n$*: As μ^2 increases (i.e., as the data are better separated), the probability that they have small distance in the presence of missing data gets smaller. This also allows for a wider range of ϵ and γ.

- *The probability of a coordinate being present p*: First, we note that as p increases, q increases. Then, as q increases (i.e., we have more data present), the probability that two points from the two Gaussian clouds have distance smaller than ϵn decreases. Additionally, for all $q > 0$, we have a feasible range for ϵ. Thus, for any percentage of missing data, if we have enough data points, we can use MR-MISSING for dimensionality reduction.

The final parameter γ has a range of values it can take on and we could optimize over it to get the smallest possible bound.

Finally, as noted before, our method of estimating distances only decreases distances. Thus, points that were close together stay close together. Now by the above the

theorem we see that if points were initially far apart, then our method of estimating distances keeps them far apart with high probability.

Then, since our method of repairing the metric only increases the distance, we see that we maintain the local neighborhood structure with high probability, in this data model.

3.4 Experiments

Let us now verify that our algorithm does will in practice through a variety of experiments. Dimensionality reduction and clustering algorithms are normally used when we have unlabeled date (i.e., in the regime of unsupervised learning). In this case figuring out the ground truth can be difficult. Hence, there are no natural numerical metrics on unlabeled data that we can readily use to evaluate our algorithm. Hence, decided to test the effectiveness of our algorithm on both unlabeled and labeled data in a variety of different experiments.

3.4.1 Unlabeled Data

For unlabeled data, we tested the performance of MR-Missing on synthetic manifolds visually as well as compared our algorithms against nlPCA and mDRUR numerically.

3.4.1.1 Synthetic Manifolds

Let us first define the six manifolds we tested our algorithm on. For notational convenience, let $U(n, m)$ be an $n \times m$ matrix with entries drawn uniformly at random from $[0, 1]$ and let $N(n, m)$ be an $n \times m$ matrix with entries are drawn from a standard Gaussian distribution. Finally, we shall write $f.(X)$ to represent applying f coordinate wise to all elements of X. We generated the six synthetic manifolds as follows:

1. $M_1 = \cos.(U(2000, 2) \cdot N(2, 30))$

2. $M_2 = \cos.(\text{sigmoid}.(U(2000, 2) \cdot N(2, 30)) \cdot N(30, 300))$

3. M_3 is a three dimensional manifold where x, y are drawn from a standard normal and $z = e^{-\sqrt{x^2+y^2}}$

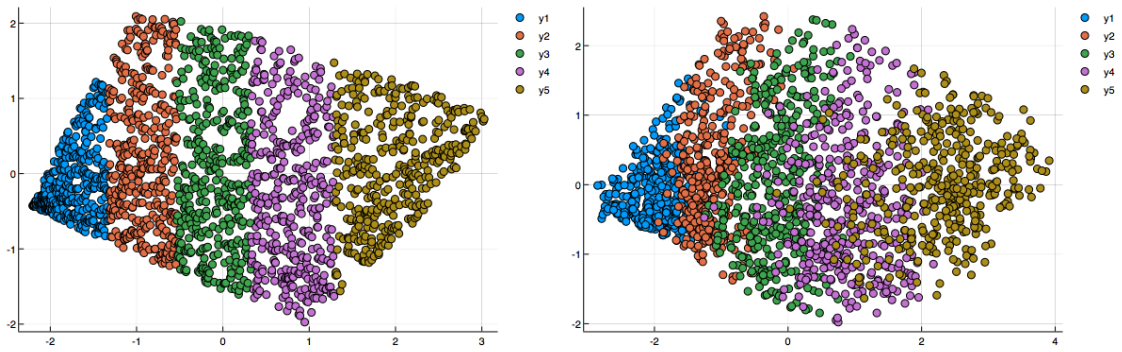
4. M_4 is a three dimensional manifold where x, y are drawn from a uniform on $[-1, 1]^2$ and $z = 20e^{(-(x^2+y^2))}$

5. $M_5 = S^2 = \{x \in \mathbb{R}^3 : \|x\| = 1\}$.

6. M_6 is closed helical curve in three dimensions. Where u is uniformly drawn from $[0, 4\pi]$, and $v = 0.5u$. Then $x = (3 + \cos(u))\cos(v)$, $y = (3 + \cos(u))\sin(v)$, and $z = \sin(u)$

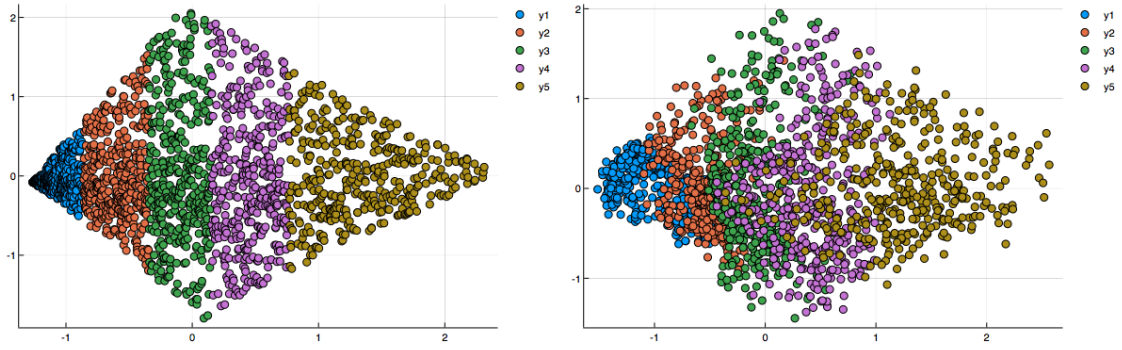
In each case we start with 2000 data points on the high dimensional representations of the six manifolds $M_1, \ldots, M_6$. We then compute low dimensional representations using the full data set and using data set with missing entries as follows. We ran ISOMAP with the true distance matrices to get the low dimensional projections, as depicted on the left hand side of Figures 3.3 and 3.4. We then picked 40% on the entries uniformly at random, declared these entries to be missing, and used MR-MISSING to get the projections depicted on right of Figures 3.3 and 3.4. We used the same ISOMAP parameters across both algorithms. We then compared how there projections looked visually. In most cases our algorithm did well in preserving the general structure of the low dimensional projection as can be seen in Figures 3.3 and 3.4.

In each case, we can see that MR-MISSING does well at preserving not only the general shape of the low dimensional, but also in maintaining the relative ordering of the data points with some minor distortion. Therefore, it is also useful for applications that use these projections for clustering and classification. We test this in the next.



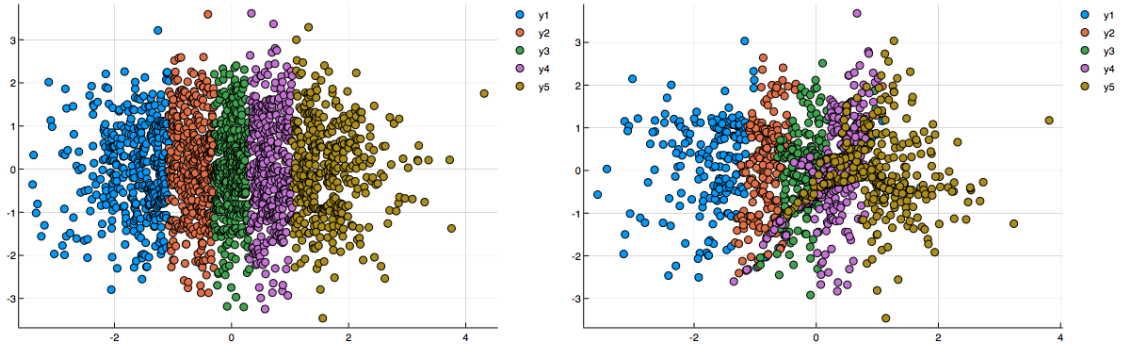
(i) M_1 with full data

(ii) M_1 with missing data



(iii) M_2 with full data

(iv) M_2 with missing data



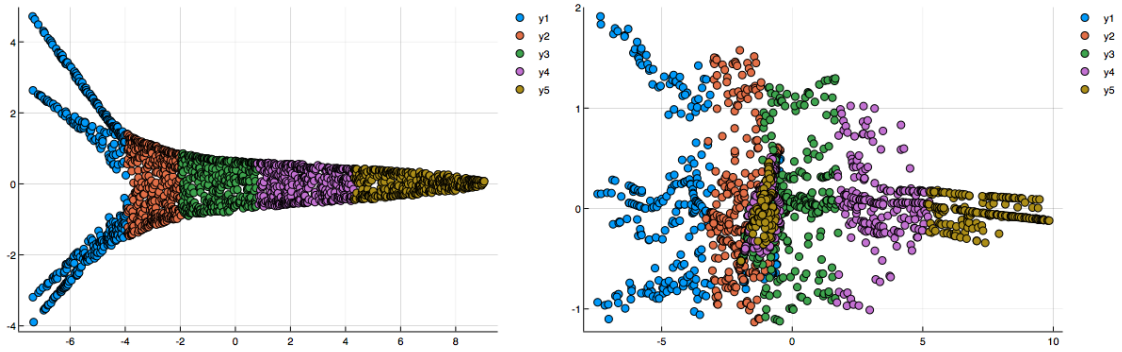
(v) M_3 with full data

(vi) M_3 with missing data

Figure 3.3: The two-dimensional embeddings produced by ISOMAP with complete data (left) versus the two-dimensional embedding produced by ISOMAP where 40% of the data is missing and we use MR-MISSING to correct the distance matrix for the manifolds M_1, M_2, M_3 (right).

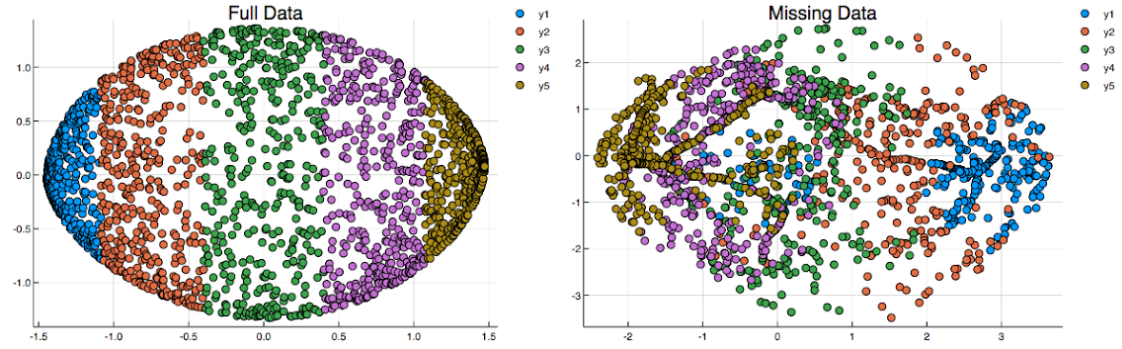
3.4.1.2 MR-missing vs nlPCA vs mDRUR

We compare MR-MISSING against Non Linear PCA (nlPCA) and Missing Data Recovery Through Unsupervised Regression (mDRUR) on the MNIST data set. nlPCA



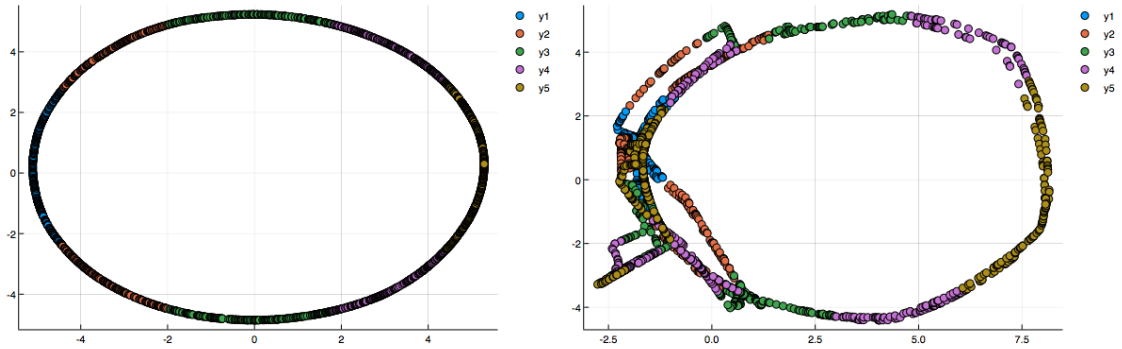
(i) M_4 with full data

(ii) M_4 with missing data



(iii) M_5 with full Data

(iv) M_5 with missing Data



(v) M_6 with full data

(vi) M_6 with missing data

Figure 3.4: The two-dimensional embeddings produced by ISOMAP with complete data (left) versus the two-dimensional embedding produced by ISOMAP where 40% of the data is missing and we use MR-MISSING to correct the distance matrix for the manifolds M_4, M_5, M_6 (right).

and mDRUR are both methods to complete missing data on a manifold, whereas our algorithm is a method to estimate and correct distances so that we can use dimensionality reduction algorithms on data sets with missing entries. So, we must

compare all of these algorithms on low dimensional representations. To that end, for the nlPCA and mDRUR algorithm, we first filled in missing data in the MNIST data set and then used ISOMAP to determine the low dimensional projection. To quantify how well these representations do we compared these representations to the low dimensional representation computed by using ISOMAP with the complete data set. We kept the parameters for ISOMAP fixed across all four projections.

We compared these projections in the following manner. Let P be the ground truth projection and $\hat{P}$ the projection computed by any algorithm on the data set with missing entries. We first aligned the two projections, using the Procrustes method because the representation returned by Multidimensional Scaling is unique only up to rotation and translation. Additionally, since we care only about the relative positions of the points and not the magnitude of the distances between them, we also allowed for scaling. That is, to ascertain the quality of the projection $\hat{P}$, we want to find

$$\text{argmin}_{Q,\alpha,\mu}\|X - \alpha YQ - 1_m\mu^t\|_F$$

where X is the data matrix that we are trying to align Y with. Here, each row is a data point, α is the scaling constant, Q is our orthogonal rotation matrix and μ is our translational vector and 1_m is the vector of all ones in m dimensions.

This problem has a closed form solution in terms of the SVD decomposition of X. Finally, if we let $\tilde{P}$ be our new scaled, rotated, and translated projection that best aligns with the ground truth projection P, then we calculate the relative error between the projections as

$$\frac{\|P - \tilde{P}\|_F}{\|P\|_F}.$$

The data set we used is the first 1000 images of the digit 0,1,2,3,4 from MNIST. For nlPCA we used a $784 \times 800 \times 12 \times 800 \times 784$ structure. For mDRUR we initially filled in the matrix using the singular value projection method. We then calculated

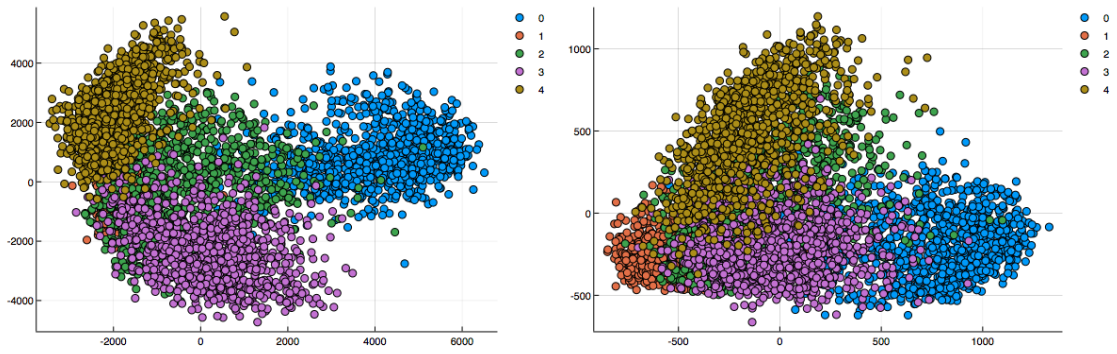
an initial 12 dimensional low representation using Laplacian Eigenmaps. We can see the relative errors in Table 1.

Algorithm	% Missing	2D	3D	4D	10D	12D	20D	50D	100D
nlPCA	40	0.363	0.350	0.385	0.404	0.451	0.514	0.623	0.686
mDRUR	40	0.369	0.363	0.359	0.420	0.427	0.505	0.630	0.717
MR-Missing	40	0.291	0.274	0.263	0.339	0.359	0.438	0.572	0.658
nlPCA	50	0.324	0.330	0.317	0.394	0.441	0.506	0.621	0.685
mDRUR	50	0.497	0.505	0.471	0.518	0.542	0.587	0.707	0.777
MR-Missing	50	0.323	0.317	0.328	0.393	0.417	0.482	0.615	0.707
nlPCA	60	0.366	0.365	0.399	0.405	0.441	0.520	0.635	0.696
mDRUR	60	0.595	0.595	0.573	0.654	0.667	0.712	0.802	0.849
MR-Missing	60	0.369	0.370	0.376	0.436	0.448	0.505	0.653	0.741
nlPCA	70	0.373	0.373	0.391	0.432	0.465	0.533	0.643	0.706
mDRUR	70	0.924	0.874	0.820	0.825	0.830	0.854	0.898	0.920
MR-Missing	70	0.484	0.491	0.595	0.498	0.510	0.573	0.697	0.784

Table 3.1: Table comparing the relative error of the projection of MNIST data obtained via NLPCA vs mDRUR vs MR-missing for various different dimensions and percentage of data missing

We can see that in all cases mDRUR does the worst so we will focus on comparing MR-Missing versus nlPCA. When we have 40% missing data MR-Missing does better than the nlPCA version in all cases. For 50% missing we see that nlPCA does better or as well in some cases with MR-Missing still doing better in a majority of the cases. For 60% missing the algorithms have similar results and nlPCA does better in the case of 70% missing data. Thus, MR-Missing does better for smaller percentage of missing data while for higher percentage of missing data nlPCA does better. Additionally we can see that as the dimension increases both methods have worse errors. We posit that this occurs because to compute a rank k projection $P(\hat{P}, \tilde{P})$ we are using the first k singular values of the distance matrix computed by Isomap. Hence our estimation the distance matrix does better at preserving the larger singular values as compared to the smaller values.

Let us also take a closer look at what our algorithm does in the case of 70% missing data. See the two dimensional representations shown in Figure 3.5.



(i) ISOMAP with actual distances (ii) ISOMAP with repaired distances

Figure 3.5: Two-dimensional projections of the first 1000 images of the digits 0,1,2,3,4 from MNIST using ISOMAP with true distance and ISOMAP with distance obtained from MR-MISSING when 70% of the data is missing.

We see that we have a different looking projection, but the projection still does well in maintaining the clustering of the data, as we predicted theoretically in Section 3.3.1. In the next subsection, we test the effectiveness of these computed low dimensional representations for classification.

3.4.2 Labeled Data

One of the main reasons to find a lower dimensional data representation is to efficiently carry out standard machine learning tasks, such as classification. In this subsection, we test the usefulness of the low dimensional representations produced by MR-MISSING for classification. We calculate a 100-dimensional representation of 500 images of each digit from MNIST. This is our training set. We then used the equation from *Bengio et al.* (2003) to calculate the projections for an additional 100 images of each digit. This is our test set. Then, for classification we trained an SVM (Kernel: RBF, $C = 200$, $\gamma = 0.0000002$) for classification. We then obtained the following classification accuracy for various amounts of missing data shown in Table 3.2. We also ran this classifier with no missing data as a benchmark. As we expect, with missing data, we do not have as high accuracy as we do with complete data, but

from our experiments we see that even in the presence of missing data, ISOMAP with MR-MISSING produces embeddings on which we still have reasonably high accuracy. Particularly in the cases of 40% and 50% missing data, we have accuracy of over 90% and the accuracy does not drop off drastically until we get to 80% missing data.

% missing	0	40	50	60	70	80	90
Accuracy	0.94	0.91	0.90	0.86	0.77	0.20	0.10

Table 3.2: Table showing the accuracy of an SVM trained on the low dimensional projections produced by MR

It is important to note that both the test and the training set had data points missing. We see that MR-MISSING does a good job of maintaining the original clusters of the data.

3.5 Conclusion and Future Work

As we can see MR-MISSING has excellent experimental results. That is, MR-MISSING is a method by which we can use traditional dimensionality reduction algorithms in the presence of missing data. While we have some theoretical justification for this observed performance, more work needs to be done in exploring the effectiveness of our method of estimating distances in more general scenarios.

Additionally, we did not consider in a detailed fashion other corrupted data or distance models. One approach to metric completion (model scenario 4) is to take the given distances, treat these as edges on a graph, and run APSP on this graph to fill in the missing distances. It is possible that APSP modifies some of the given distance information while also filling in the missing values. Thus, a natural question is are there conditions on the given data that guarantee that an APSP algorithm will not change the given data while simultaneously repairing those that are missing? Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b) provide some analysis which suggests a more general version of metric repair may be applied to the metric problem.

Theorem 3.6. (Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b)) Suppose G is a weighted chordal graph such that no three cycle is broken. Then if we run APSP on this graph, the shortest path between any two adjacent vertices, is the edge connecting them.

Corollary 3.7. (Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b)) If the given distances form a graph G, where G is a weighted chordal graph such that no 3 cycle is broken, then this partial distance information can be completed into a metric.

The condition that the given data satisfy a chordal graph appears in Positive Semi-Definite matrix completion and Euclidean distance matrix completion as well. While the first theorem tells us when we can use APSP to complete a metric, it doesn't tell us what properties this new metric satisfies. Hence, the problem of completing a distance matrix for a general metric warrants further investigation.

In the model scenario 3, when we have a corrupted distance matrix, we may not always want to increase distances. In some cases we might want to decrease distances. Hence, we would need a general metric repair algorithm. Fan, et al. *Fan et al.* (2018c) present an algorithm that runs in $\theta(n^6)$ and Gilbert and Sonthalia *Gilbert and Sonthalia* (2018b) present an alternative algorithm that runs in $O(n^5)$. Both of these algorithms are impractical and cannot be used on large data sets. Developing faster algorithms for general metric repair and ascertaining the usefulness of such methods for corrupted distance matrices are two avenues for future work.

CHAPTER IV

Project and Forget: Solving Large Scale Metric Constrained Problem

4.1 Introduction

Given a set of dissimilarity measures amongst data points, many machine learning problems are considerably "easier" if these dissimilarity measures adhere to a metric. Furthermore, learning the metric that is most "consistent" with the input dissimilarities or the metric that best captures the relevant geometric features of the data (e.g., the correlation structure in the data) is a key step in efficient, approximation algorithms for classification, clustering, regression, and feature selection. In practice, these metric learning problems are formulated as convex optimization problems subject to metric constraints, such as the triangle inequality, on all the output variables. Because of the large number of constraints, researchers have been forced to restrict either the kinds of metrics learned or the size of the problem that can be solved. In many cases, researchers have restricted themselves to learning (weighted) Euclidean or Mahalanobis metrics. This approach is, however, far from ideal as the inherent geometry of many data sets necessitates different types of metrics. Therefore, we need to develop optimization techniques that can optimize over the space of all metrics on a data set.

Many of the existing methods for metric constrained problems suffer from significant

drawbacks that hamper performance and restrict the instance size. Gradient based algorithms such as projected gradient descent (e.g., *Beck and Teboulle* (2009); *Nesterov* (1983)) or Riemannian gradient descent require a projection onto the space of all metrics which, in general, is an intractable problem. One modification of this approach is to sub-sample the constraints and then project onto the sampled set (see *Nedić* (2011); *Polyak* (2001); *Wang and Bertsekas* (2013); *Wang et al.* (2015)). For metric constrained problems, however, there are many more constraints than data points, so the condition numbers of the problems are quite high and, as a result, these algorithms tend to require a large number of iterations.

Another standard approach is based on the Lagrangian method. These algorithms augment the objective (or introduce a barrier function) by adding a term for each constraint. Examples of such methods include the interior point method, the barrier method, and the Alternating Direction Method of Multipliers. These methods run into two different kinds of problems. First, computing the gradient becomes an intractable problem because of the large number of constraints. One fix could be to sub-sample the constraints and compute only those gradients, but this approach runs into the same drawbacks as we discussed before. The other option is to incrementally update the Lagrangian, looking at one constraint at a time. Traditionally, these methods require us to cycle through all the constraints. One such method is Bregman cyclic method and we note that the requirement to examine cyclically the constraints is an aspect that is highlighted in various previous works *Censor and Reich* (1998); *Bauschke and Lewis* (2000b); *Censor and Zenios* (1997). This is simply not feasible with metric constraints. Many applications that use this method sidestep the issue either by restricting the number of constraints and solving a heuristic problem (*Davis et al.*, 2007), by solving smaller sized problems (*Dhillon and Tropp*, 2007), or by trying to parallelize the projections (*Ruggles et al.*, 2019).

A third approach is to use traditional active set methods such as Sequential

Linear/Quadratic Programming (*Palacios-Gomez et al.*, 1982; *Boggs and Tolle*, 1995). In general, these methods maintain a set of constraints C that is assumed to be the true set of active constraints. During each iteration, they *completely* solve the sub-problem defined by the constraints in C. They then check which of the constraints in C are inactive and remove those constraints. They also search for and add new violated constraints. These methods run into the problem that each iteration is computationally expensive and, in some cases, may not be tractable because of the large number violated constraints. If the size of C is reduced, then the number of iterations becomes too large, again making the problem intractable.

One final approach is to use cutting planes. The performance of this method is heavily dependent on the cut selection process (see *Dey and Molinaro* (2018); *Poirrier and Yu* (2019) for deep discussions). The discovery of Gomory cuts (*Gomory*, 1960) and other subsequent methods such as branch and bound, has led to the viability of the cutting plane method for solving mixed integer linear programs. This success has, however, not transferred to other problems. In general, if the cuts are not selected appropriately, the algorithm could take an exponential number of iterations; i.e., it might add an exponential number of constraints. To use this method, we must show for each problem that the specific cutting plane selection method results in a feasible algorithm (see *Chandrasekaran et al.* (2012) for an example).

In this paper, we provide an active set algorithm, PROJECT AND FORGET, that uses Bregman projections to solve **convex optimization problems with a large number of (possibly exponentially linear inequality) constraints**. Since our algorithm is based on the cyclic Bregman method, it has the rapid rate of convergence of the Bregman method, along with all the benefits of being an active set method. This method overcomes the weaknesses of both the traditional active set methods and Bregman cyclic method. First, we overcome the drawbacks of the active set methods by allowing the introduction of new and the removal of old constraints *without having*

to completely solve convex programs as intermediate steps. In particular, our algorithm examines each constraint once, before it introduces new constraints and forgets old constraints, thus allowing us to converge rapidly to the true active constraint set. We overcome the drawback of the traditional Bregman method by cycling through the current active set only. Thus, making each iteration much faster. Our new algorithm PROJECT AND FORGET, is the first iterative Bregman projection based algorithm for convex programs that does not require us to cyclically examine all the constraints.

The major contributions of our paper are as follows:

1. For the case when we have linear inequality constraints, we provide a Bregman projection based algorithm that does not need to look at the constraints cyclically. We prove that our algorithm converges to the global optimal solution and that the optimality error (L_2 distance of the current iterate to the optimal) asymptotically decays at an exponential rate. We also show that because of the FORGET step, when the algorithm terminates, the set of constraints remembered are exactly the active constraints.
2. For the case when we have general convex constraints, we provide a Bregman projection based algorithm that does not need to look at the constraints cyclically. We prove that our algorithm converges to the global optimal solution and that when we have quadratic objective function, the optimality error (L_2 distance of the current iterate to the optimal) asymptotically decays at an exponential rate. We also show that because of the FORGET step, when the algorithm terminates, the set of constraints remembered are exactly the active constraints.
3. We solve the weighted correlation clustering problem *Bansal et al.* (2004) on a graph with over $130,000$ nodes. To solve this problem with previous methods, we would need to solve a linear program with over 10^{15} constraints. Furthermore, we demonstrate our algorithms superiority by outperforming the current state of the art in terms of CPU times.

4. We use our algorithm to develop a new algorithm that solves the metric nearness problem *Brickell et al.* (2008c). We show that our algorithm outperforms the current state of the art with respect to CPU time and can be used to solve the problem for non-complete graphs.
5. We also show the generality of our algorithm, by using it to solve the quadratically regularized optimal transport problem. We show, that using our algorithm, we can solve this problem faster and using less memory than many standard solvers.

4.2 Preliminaries

We start by presenting the general version of the problem before focusing on metric constrained problems in later sections.

4.2.1 Convex Programming

Given a strictly convex function $f : \mathbb{R}^d \to \mathbb{R}$, and a finite family of convex sets $\mathcal{F} = \{C_i\}$ we want to find the unique point $x^* \in \bigcap_i C_i =: C$ that solves the following problem.

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && \forall i,\, x \in C_i. \end{aligned} \tag{4.1}$$

We refer to each C_i as a *constraint set* and $C := \bigcap_i C_i$ as the *feasible region*. We shall assume that C is not empty; i.e., there is at least one feasible point. Since we have a large number of constraint sets, we access the constraint sets only through an oracle that has one of the two following separation properties.

Property 4.1. $\mathcal{Q}$ is a deterministic separation oracle for a family of convex sets $\mathcal{F} = \{C_i\}$, if there exists a non-decreasing, continuous function ϕ, with $\phi(y) = 0 \iff y = 0$, such that on input $x \in \mathbb{R}^d$, $\mathcal{Q}$ either certifies $x \in C$ or returns a list $\mathcal{L} \subset \mathcal{F}$

such that

$$\max_{\tilde{C} \in \mathcal{L}} \text{dist}(x, \tilde{C}) \geq \phi(\text{dist}(x, C)),$$

where for a point x and set B, $\text{dist}(x, B) = \inf_{w \in B} \|w - x\|$.

There a few things that we would like to highlight about this definition. First, the list $\mathcal{L}$ *need not contain all violated constraints.* That is, given x, there can be some $C_i \in \mathcal{F}$ such that $x \notin C_i$ *and* $C_i \notin \mathcal{L}$. In fact, there could be many such C_i. However, what we do require is that the maximum distance from x to the constraints in $\mathcal{L}$ is at least some non-decreasing function ϕ of the distance from x to C.

Property 4.2. $\mathcal{Q}$ is a random separation oracle for a family of convex sets $\mathcal{F}$, if there exists a lower bound $\tau > 0$, such that on input $x \in \mathbb{R}^d$, $\mathcal{Q}$ returns a list $\mathcal{L} \subset \mathcal{F}$ such that

$$\forall \tilde{C} \in \mathcal{F},\ Pr[\tilde{C} \in \mathcal{L}] \geq \tau.$$

Remark 4.3. The random separation oracle need not decide whether $x \in C$.

4.2.1.1 Linear Inequality Constraints

In practice, most problems do not have the most general of convex constraints. Indeed, linear inequality constraints are common. In such a case, our constraints sets C_i are half spaces. In particular, we denote each half space by H_i instead of C_i. Additionally, for each H_i, we know that there exists $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ such that

$$H_i = \{x \in \mathbb{R}^d : \langle a_i, x \rangle \leq b_i\}.$$

Thus, if A is the matrix whose rows are given by a_i and b is the vector whose coordinates are b_i, then our feasible region C can be represented as follows:

$$C = \{x \in \mathbb{R}^d : Ax \leq b\}.$$

In this case, we can reformulate Problem 4.1 as follows.

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && Ax \leq b \end{aligned} \tag{4.2}$$

Solving this problem for general convex functions f is too monumental a task. We restrict ourselves to a rich class of functions known as Bregman functions. First, we define the generalized Bregman distance.

Definition 4.4. Given a convex function $f(x) : S \to \mathbb{R}$ whose gradient is defined on S, we define its *generalized Bregman distance* $D_f : S \times S \to \mathbb{R}$ as $D_f(x, y) = f(x) - f(y) - \langle \nabla f(y), x - y \rangle$.

Definition 4.5. A function $f : \Lambda \to \mathbb{R}$ is called a Bregman function if there exists a non-empty convex set S such that $\overline{S} \subset \Lambda$ and the following hold:

(i) $f(x)$ is continuous, strictly convex on $\overline{S}$, and has continuous partial derivatives in S.

(ii) For every $\alpha \in \mathbb{R}$, the partial level sets $L_1^f(y, \alpha) := \{x \in \overline{S} : D_f(x, y) \leq \alpha\}$ and $L_2^f(x, \alpha) := \{y \in S : D_f(x, y) \leq \alpha\}$ are bounded for all $x \in \overline{S}, y \in S$.

(iii) If $y_n \in S$ and $\lim_{n\to\infty} y_n = y^*$, then $\lim_{n\to\infty} D_f(y^*, y_n) = 0$.

(iv) If $y_n \in S$, $x_n \in \overline{S}$, $\lim_{n\to\infty} D_f(x_n, y_n) = 0$, $y_n \to y^*$, and x_n is bounded, then $x_n \to y^*$.

We denote the family of Bregman functions by $\mathcal{B}(S)$. We refer to S as the zone of the function and we take the closure of the S to be the domain of f.

This class of function includes many natural objective functions, including entropy $f(x) = -\sum_{i=1}^n x_i \log(x_i)$ with zone $S = \mathbb{R}^n_+$ (here f is defined on the boundary of S by taking the limit) and $f(x) = \frac{1}{p}\|x\|_p^p$ for $p \in (1, \infty)$. The ℓ_p norms for $p = 1, \infty$ are not Bregman functions but can be made Bregman functions by adding a quadratic term. That is, $f(x) = c^T x$ is a not Bregman function, but $c^T x + x^T Q x$ for any positive

definite Q is a Bregman function.

Definition 4.6. We say that a hyper-plane H_i is *strongly zone consistent* with a respect to a Bregman function f and its zone S, if for all $y \in S$ and for all hyper-planes H, parallel to H_i that lie in between y and H_i, the Bregman projection of y onto H lies in S instead of in $\overline{S}$.

In addition to the restrictions on the functions for which we can solve Problem 4.2, we will need the assumption that all hyper-planes in $\mathcal{H}$ (our family of half spaces) are strongly zone consistent with respect to $f(x)$. This assumption is used to guarantee that when we do a projection the point we project onto lies within our domain. This is also not too restrictive. For example, all hyper-planes are strongly zone consistent with respect to the objective functions $f(x) = 0.5\|x\|^2$ and $f(x) = -\sum_i x_i \log(x_i)$. The final assumption, that we mentioned earlier, is that C is non-empty. This is needed to ensure the algorithm converges.

4.2.1.2 General Convex Constraints

While general convex constraints do not often appear in practical problems, such a formulation is of theoretical interest. When our constraints are simply convex sets rather than linear ones, we must adjust our algorithmic approach and, as a result, our assumptions about the objective function $f(x)$ and the constraints will be slightly different as compared to the linear case. The problem that we are interested in is

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in C_i \quad \forall\, C_i. \end{aligned} \tag{4.3}$$

Before we state our assumptions, we need the following definitions.

Definition 4.7. A function f is Legendre, if f is a closed proper map, $int(domf) \neq \emptyset$, and $\lim_{t\downarrow 0}\langle \nabla f(x+t(y-x)), y-x\rangle = -\infty$ for all $x \in \partial(domf)$ and for all $y \in int(domf)$.

Definition 4.8. A function f is very strictly convex, if f is twice differentiable everywhere and its Hessian is positive definite everywhere.

Definition 4.9. A function f is co-finite is $\lim_{r\to\infty} f(rx)/r = \infty$ for all $x \in domf$.

For this version of the problem we will assume that f is a very strictly convex, co-finite, Legendre function. Note that these are strong conditions and rule out some objective functions such as Burgs entropy $f(x) = \sum_i \log(x_i)$. They do still, however, allow a rich class of functions. Some examples can be seen below:

1. $f(x) = 0.5\|x\|^2$ on $\mathbb{R}^n$.
2. $f(x) = \sum_i x_i \log(x_i) - x_i$ (Boltzman/Shannon entropy) on $\mathbb{R}^n_+$.
3. $f(x) = -\sum_i \sqrt{1 - x_i^2}$ (Hellinger distance) in $[-1, 1]^n$.
4. $f(x) = \sum_i x_i \log(x_i) + (1 - x_i)\log(1 - x_i)$ (Fermi/Dirac entropy) on $[0, 1]^n$.
5. $f(x) = \begin{cases} \frac{1}{2}x^2 + 2x + \frac{1}{2} & x \le -1 \\ -1 - \log(-x) & -1 \le x < 0 \end{cases}$.

Note that the last example Legendre function is not a Bregman function. In addition to the Legendre function assumptions, we also assume that $int(domf) \cap C \neq \emptyset$. See *Bauschke and Lewis* (2000b) for a discussion comparing the two different classes functions presented.

4.2.2 Metric Constrained Problems

The primary motivation of this paper is to solve metric constrained problems and we set up such problems in this section. To define general metric constrained problems, we first define the metric polytope.

Definition 4.10. Let $\text{MET}_n \subset \mathbb{R}^{\binom{n}{2}}$ be the space of all metrics on n points. Given a graph G the metric polytope $\text{MET}_n(G)$ is the projection of MET_n onto the coordinates

given by the edges of G (i.e., we consider distances only between pairs of points that are adjacent in G).

It can be easily seen that for any $x \in \mathbb{R}^{\binom{n}{2}}$, $x \in \mathrm{MET}_n(G)$ if and only if $\forall\, e \in G$, $x(e) \geq 0$ and for every cycle $\mathcal{C}$ in G and $\forall\, e \in \mathcal{C}$, we have that

$$x(e) \leq \sum_{\tilde{e} \in \mathcal{C}, \tilde{e} \neq e} x(\tilde{e}).$$

Therefore, $\mathrm{MET}_n(G)$ can be described as the intersection of exponentially many half-spaces.

Remark 4.11. It is important to note that MET_n is the space of *all* metrics on n points. Hence, when we optimize over MET_n ($\mathrm{MET}_n(G)$) we are optimizing over a much larger and more complex space than the space of Euclidean metrics or the space of all Mahalanobis metrics.

Now that we have the set over which we want to optimize, we give a general formulation for metric constrained optimization problems.

Definition 4.12. Given a strictly convex function f, a graph G, and a finite family of half-spaces $\mathcal{H} = \{H_i\}$ such that $H_i = \{x : \langle a_i, x \rangle \leq b_i\}$, we seek the unique point $x^* \in \bigcap_i H_i \cap \mathrm{MET}(G) =: C$ that minimizes f. That is, if we set A to be the matrix whose rows are a_i and b be the vector whose coordinates are b_i we seek

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & Ax \leq b \\ & x \in MET(G). \end{aligned} \tag{4.4}$$

The constraints encoded in the matrix A let us impose additional constraints, beyond the metric constraints. For example, in correlation clustering, the matrix A encodes $x_{ij} \in [0, 1]$. In general, we will assume that the number of additional

constraints encoded in A (beyond the metric constratins) is relatively small so that the predominant difficulty in solving these optimization problems comes from the metric constraints.

4.2.3 Projections

All of our algorithms will be based on iteratively computing Bregman projections.

Definition 4.13. Given a strictly convex function f, a closed convex set Y, and a point y, the projection of y onto Y with respect to D_f is a point $x^* \in \text{dom}(f)$ such that

$$x^* = \underset{x \in Y \cap \text{dom}(f)}{\arg\min} D_f(x, y).$$

In the case we have linear inequality constraints, we project onto the boundary of the half space ∂H. In this case, the Bregman projection has some additional special properties.

Lemma 4.14. Let x be the point that we project onto $\partial H_i = \{y \in \mathbb{R}^d : \langle y, a_i \rangle = b_i\}$, then there exists a unique x^*, θ such that $\nabla f(x^*) = \nabla f(x) + \theta a_i$ and $\langle x^*, a_i \rangle = b_i$. This unique x^* is also the Bregman projection of x on ∂H_i. Furthermore,

1. $\theta > 0$ if and only if $\langle x, a_i \rangle > b_i$;
2. $\theta < 0$ if and only if $\langle x, a_i \rangle < b_i$;
3. $\theta = 0$ if and only if $\langle x, a_i \rangle = b_i$.

4.3 Project and Forget: Linear Inequalities

To set the stage for subsequent discussions, we present the general structure of our algorithm first and then detail adjustments we make for the different kinds of constraints (linear inequalities versus the more general convex constraints). It is

iterative and, in general, will be run until some convergence criterion has been met. The convergence criterion depends largely on the specific application for which the algorithm is tailored. For this reason, we postpone the discussion of the convergence criterion until the applications section.

The PROJECT AND FORGET algorithm keeps track of three quantities; $x^{(\nu)}$, the vector of variables over which we optimize, $L^{(\nu)}$ a list of the constraints that the algorithm deems active, and $z^{(\nu)}$ a vector of dual variables. Each iteration of the PROJECT AND FORGET algorithm consists of three phases. In the first phase, we query our oracle $\mathcal{Q}$ to obtain a list of constraints L. In the second phase, we merge $L^{(\nu)}$ with L to form $\tilde{L}^{(\nu+1)}$ and project onto each of the constraints in $\tilde{L}^{(\nu+1)}$ one at a time. When we do these projections, we update $x^{(\nu)}$ and $z^{(\nu)}$. Finally, in the third phase, we forget some constraints from $\tilde{L}^{(\nu+1)}$ to yield $L^{(\nu+1)}$.

Algorithm 7 General Algorithm.

function PROJECT AND FORGET(f convex function)
  $L^{(0)} = \emptyset$, $z^{(0)} = 0$. Initialize $x^{(0)}$ so that $\nabla f(x^{(0)}) = 0$.
  while Not Converged **do**
    $L = \mathcal{Q}(x^{\nu})$
    $\tilde{L}^{(\nu+1)} = L^{(\nu)} \cup L$
    $x^{(\nu+1)}, z^{(n+1)} = \text{Project}(x^{(\nu)}, z^{(\nu)}, \tilde{L}^{(\nu+1)})$
    $L^{(\nu+1)} = \text{Forget}(z^{(\nu+1)}, \tilde{L}^{(\nu+1)})$
  return x

4.3.1 Finding Violated (Metric) Constraints

The first step of the method is to find violated constraints and in this subsection we detail how to find violated metric constraints in particular (which are a special case of linear inequality constraints). In many applications, we could do this by searching through the list of constraints until we found a violated constraint. However, in our case, since $\text{MET}_n(G)$ has exponentially many faces, we cannot list all of them, so we seek an efficient separation oracle $\mathcal{Q}$. That is, given a point x, the oracle efficiently return a list L of violated constraints, such that the constraints in L satisfy some

properties. We will assume that $\mathcal{Q}$ satisfies either the Property 4.1 or Property 4.2.

Algorithm 8 Finding Metric Violations.

```
1: function METRIC VIOLATIONS(d)
2:     L = ∅
3:     Let d(i, j) be the weight of shortest path between nodes i and j or ∞ if none
   exists.
4:     for Edge e = (i, j) ∈ E do
5:         if w(i, j) > d(i, j) then
6:             Let P be the shortest path between i and j
7:             Add C = P ∪ {(i, j)} to L
       return L
```

For metric constrained problems, Algorithm 8 finds violated constraints. If the metric constrained problem has additional constraints (i.e $Ax \leq b$), then we augment our oracle accordingly.

Proposition 4.15. METRIC VIOLATION runs $\Theta(n^2 \log(n) + n|E|)$ time and satisfies Property 4.1 with $\phi(y) = \frac{y}{n^{1.5}}$.

Proof. The first step in METRIC VIOLATION is to calculate the shortest distance between all pairs of nodes. This can be done using Dijkstra's algorithm in $\Theta(n^2 \log(n) + n|E|)$ time. Then, if the shortest path between any adjacent pair of vertices is not the edge connecting them, then the algorithm has found a violated cycle inequality. Note that if no such path exists, then all cycle inequalities have been satisfied and the input point x (representing distances) is within the metric polytope. Thus, we have an oracle that separates the polytope.

However, we want an oracle that also satisfies property 4.1. To that end, let us define the deficit of a constraint. Given a point x and a hyper-plane $H_{\mathcal{C},e}$, defined by some cycle $\mathcal{C}$ and an edge e, the deficit of this constraint is given by

$$d(\mathcal{C}, e) = x(e) - \sum_{\tilde{e} \in \mathcal{C}, \tilde{e} \neq e} x(\tilde{e}).$$

If this quantity is positive, then x violates this constraint. In this case, the squared

distance from x to this constraint is $\frac{d(\mathcal{C},e)^2}{|\mathcal{C}|}$ (i.e., we add/subtract $\frac{d(\mathcal{C},e)}{|\mathcal{C}|}$ to each edge weight of $\mathcal{C}$).

Now let x_{apsp} be the all pair shortest path metric obtained from x and $\mathcal{L}$ be the list returned by the oracle. Then

$$\|x - x_{\text{apsp}}\|_2^2 = \sum_{\mathcal{C},e\in\mathcal{L}} d(\mathcal{C},e)^2.$$

Thus, if $H_{\tilde{\mathcal{C}},\tilde{e}}$ is the constraint that maximizes $d(\tilde{\mathcal{C}},\tilde{e})$, then we have that

$$\text{dist}(x,\tilde{C}_{\tilde{\mathcal{C}},\tilde{e}})^2 = \frac{d(\tilde{\mathcal{C}},\tilde{e})^2}{|\tilde{\mathcal{C}}|} \geq \frac{\|x - x_{\text{apsp}}\|_2^2}{|\tilde{\mathcal{C}}||\mathcal{L}|}.$$

Since our oracle returns at most 1 constraint per edge, we have that $|\mathcal{L}| \leq |E| \leq n^2$. This along with the fact that $|\tilde{\mathcal{C}}| \leq n$, gives us that

$$\text{dist}(x,H_{\tilde{\mathcal{C}},\tilde{e}})^2 \geq \frac{\|x - x_{\text{apsp}}\|_2^2}{n^3}.$$

Finally, we know that $x_{\text{apsp}} \in \text{MET}(G)$. Thus, we see that

$$\|x_{\text{apsp}} - x\|_2^2 \geq \text{dist}(x,\text{MET}(G))^2 = \text{dist}(x,C)^2.$$

Putting it all together, we have that

$$\begin{aligned}\max_{\hat{C}\in\mathcal{L}} \text{dist}(x,\hat{C})^2 &\geq \text{dist}(x,H_{\tilde{\mathcal{C}},\tilde{e}})^2 \\ &\geq \frac{\|x - x_{\text{apsp}}\|_2^2}{n^3} \\ &\geq \frac{\text{dist}(x,C)^2}{n^3}.\end{aligned}$$

Taking the square root of both sides gives the needed result. □

4.3.2 Project and Forget Steps

The Project and Forget steps for the algorithm are presented in Algorithm 9. Let us step through the code to obtain an intuitive understanding of its behavior. Let $H_i = \{x : \langle a_i, x\rangle \leq b_i\}$ be a constraint and x the current iterate. The first step is to calculate x^* and θ. Here x^* is the projection of x onto the boundary of H_i and θ is a "measure" of how far x is from x^*. In general, θ can be any real number and so we examine two cases: θ positive or negative.

It can be easily seen from Lemma 4.14 that θ is negative if and only if the constraint is violated. In this case, we have $c = \theta$ because (as we will see in proof) the algorithm always maintains $z_i \geq 0$. Then on line 5, we compute the projection of x onto H_i. Finally, since we corrected x for this constraint, we add $|\theta|$ to z_i. Since each time we correct for H_i, we add to z_i, we see that z_i stores the total corrections made for H_i. On the other hand, if θ is positive, this constraint is satisfied. In this case, if we also have that z_i is positive; i.e., we have corrected for H_i before and we have over compensated for this constraint. Thus, we must undo some of the corrections. If $c = z_i$, then we undo all of the corrections and z_i is set to 0. Otherwise, if $c = \theta$ we only undo part of the correction.

For the Forget step, given a constraint $H_i \in \tilde{L}^{(\nu+1)}$, we check if $z_i^{(\nu+1)} = 0$. If so, then we have not done any net corrections for this constraint and we can forget it; i.e., delete it from $\tilde{L}^{(\nu+1)}$.

If we think of $L^{(\nu)}$ as matrix, with each constraint being a row, we see that at each iteration $L^{(\nu)}$ is a sketch of the matrix of active constraints. Hence, during each iteration we update this sketch by adding new constraints (rows). During the Forget step, we determine which parts of our sketch are superfluous and we erase (forget) these parts (rows) of the sketch.

In general, calculating the Bregman projection (line 3) cannot be done exactly. See *Dhillon and Tropp* (2007) for a general method to perform the calculation on line

Algorithm 9 Project and Forget algorithms.

function PROJECT(x, z, L)
  for $H_i = \{y : \langle a_i, y\rangle = b_i\} \in L$ **do**
    Find x^*, θ by solving $\nabla f(x^*) - \nabla f(x) = \theta a_i$ and $x^* \in H_i$
    $c_i = \min(z_i, \theta)$
    $x \leftarrow x_{new}$, $x_{new} \leftarrow$ such that $\nabla f(x_{new} - \nabla f(x) = c_i a_i$
    $z_i \leftarrow z_i - c_i$
  return x, z
function FORGET(z, L)
  for $H_i = \{x : \langle a_i, x\rangle = b_i\} \in L$ **do**
    if $z_i == 0$ **then** Forget H_i
  return L

3. For example, if $f(x) = x^T Q x + r^T x + s$ where Q positive definite, then for a given hyper-plane $\langle a, x\rangle = b$ and a point x we have that

$$\theta = \frac{\langle a, x\rangle - b}{a^T Q^{-1} a}. \tag{4.5}$$

4.3.3 Truly Stochastic Variant

In some problems, we have constraints defined using only subsets of the data points and we may not have an oracle that satisfies Property 4.1. For such cases, we present a stochastic version of our algorithm. Instead of calling METRIC VIOLATION or an oracle with Property 4.1, we want an oracle with Property 4.2. This version of our algorithm is very similar to the algorithms presented in *Nedić* (2011); *Wang et al.* (2015). The major difference being that we do not need to perform a gradient descent step. Instead, we maintain the KKT conditions by keeping track of the dual variables and doing dual corrections. In practice, using PROJECT AND FORGET with the random oracle tends to produce better results than *Nedić* (2011); *Wang et al.* (2015) because we remember the active constraints that we have seen, instead of hoping that we sample them.

In some cases, we may want a more stochastic variant. With the algorithm as specified, we have to keep track of the constraints that we have seen and carefully

pick which constraints to forget. We can, nevertheless, modify the Forget step to forget all constraints and obtain a truly stochastic version of the algorithm. In this version, at each iteration, we choose a random set of constraints and project onto these constraints only, independently of what constraints were used in previous iterations. We cannot, however, forget the values of the dual variables. This version is similar to that in *Bauschke and Borwein* (1997). However, *Bauschke and Borwein* (1997) only looks at the problem when we have linear equality constraints. To employ such an approach, we could modify our problem to add slack variables and change all of constraints into equality constraints, however these modifications will not yield an equivalent problem. One of the major assumptions of *Bauschke and Borwein* (1997) is that the objective function is strictly convex. Thus, we if add slack variables, then we would need to modify our objective function to be strictly convex on these variables as well. This changes the problem.

4.3.4 Convergence Analysis: Linear Inequality Constraints

Before we can use PROJECT AND FORGET, it is crucial to establish a few theoretical properties. Previous work on the convergence of the Bregman method relies on the fact that the algorithm cyclically visits all of the constraints. For our method, however, this is not the case and so it is not clear that the convergence results for the traditional Bregman method still apply. In particular, it may be the case, that given a sequence of hyper-planes returned by the oracle, the algorithm may not even converge. Fortunately, the proofs for the traditional Bregman method can be adapted in subtle ways, so that we can establish crucial theoretical properties of the PROJECT AND FORGET algorithm.

Theorem 4.16. If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then

1. If the oracle $\mathcal{Q}$ satisfies property 4.1 (property 4.2), then any sequence x^n

produced by the above algorithm converges (with probability 1) to the optimal solution of problem 4.2.

2. If x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive definite, then there exists $\rho \in (0, 1)$ such that

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho \tag{4.6}$$

where $\|y\|_H^2 = y^T H y$. In the case when we have an oracle that satisfies property 4.2, the limit in 4.6 holds with probability 1.

The proof of Theorem 4.16 also establishes another important theoretical property.

Proposition 4.17. If a_i is an inactive constraint, then there exists an N, such that for all $n \geq N$, we have that $z_i^n = 0$. That is, after some finite time, we never project onto inactive constraints ever again.

Corollary 4.18. Under the assumptions for part 2 of Theorem 4.16, the sequence $z^n \to z^*$ also converges.

These properties are important as they permit the following interpretation of our algorithm. The algorithm spends the initial few iterations identifying the active constraints from amongst a large number of constraints. This is the active set part of the algorithm. The algorithm then spends the remainder of the iterations finding the optimal solution with respect to these constraints. Empirically, we notice this phenomenon as well. At first, the error metrics decreases very slowly, while the number of constraints that are being considered grows rapidly. Eventually, we reach a point when the number of constraints that we are currently considering stabilizes, at this point the error metrics start decreasing very rapidly. An example of this phenomenon can be seen in Figure 4.2ii. This behavior is one of the major advantages of our

method. Additionally, the ability to find the set of active constraints without having to solve the problem is another advantage of our algorithm.

Remark 4.19. We note that while these results show that the algorithm converges linearly, ρ is close one. Indeed, the proof bounds $\rho \leq \frac{F}{F+1}$, where F is the number of hyperplanes that the optimal solution lies on. From a heuristic perspective, it is beneficial to use only those hyperplanes that define the facets of the constraint polytope.

For the truly stochastic case, we have the following theorem instead.

Theorem 4.20. If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then with probability 1 any sequence x^n produced by the above truly stochastic algorithm converges to the optimal solution of problem 4.2. Furthermore, if x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semi-definite, then there exists $\rho \in (0, 1)$ such that with probability 1,

$$\liminf_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho. \tag{4.7}$$

Because the proofs of Theorem 4.16 and 4.20 are quite technical and involve two different types of separation oracles, we split them into several parts. In Subsections 4.8.1 and 4.8.2, we prove the first part of Theorem 4.16 for separation oracles with property 4.1 and 4.2, respectively. In Subsection 4.8.3, we prove the second part of Theorem 4.16 (also subdividing this proof into several cases). Finally, in Subsection 4.8.4 we prove Theorem 4.20.

4.4 Project and Forget: General Convex Constraints

In the previous section, we detailed the PROJECT AND FORGET algorithm for half-space (or linear inequality) constraints. In this section, we use similar idea to

develop the appropriate variations for general convex constraints. We draw inspiration from Dijkstra's method and use a slightly different set of assumptions on our objective functions and constraints. These assumptions are detailed in Section 4.2.1.2.

4.4.1 Algorithm

Let x^n be the primal sequence of iterates and q^n an auxiliary sequence. Let P_k denote the Bregman projection operator onto the kth constraint set. Let f^* be the convex conjugate of f. Let $i(k)$ denote the control sequence, and let $p(c,k) = \arg\max_{k'<k} i(k') = c$. We will abbreviate $p(i(k),k)$ as $p(k)$. With this notation established, the PROJECT AND FORGET algorithm for the case of general convex constraints is shown in Algorithm 10.

Algorithm 10 Project and Forget algorithms

1: Initialize $q^0 = 0$, $L = \emptyset$
2: **function** PROJECT($x_0 = x, q, L$)
3: **for** $C_i \in L$ **do**
4: $x^n := (P_i \circ \nabla f^*)(\nabla f(x^{n-1}) + q^{p(n)})$
5: $q^n = \nabla f(x^{n-1}) + q^{p(n)} - \nabla f(x^n)$
return $x^{|L|}$
6: **function** FORGET(x, q, L)
7: **for** $C_i \in L$ **do**
8: **if** $q^{p(i,n)} == 0$ **then** Forget C_i
return L

Lines 4 and 5 of the above algorithm come from Dijkstra's method. To understand their role, note that since f is Legendre, we have that $\nabla f^* = (\nabla f)^{-1}$. Thus, on line 4, we are perturbing x^{n-1} by modifying its gradient with the auxiliary variable $q^{p(n)}$. For example, if $f(x) = x^T Q x$ for some positive definite matrix Q, then line 4 would be

$$x^n = P_i(Q^{-1}(Qx^{n-1} + q^{p(n)})) = P_i(x^{n-1} + Q^{-1}q^{p(n)}).$$

4.4.2 Convergence Analysis

Now that we have defined the above algorithm, the second major theoretical result of this paper is the following.

Theorem 4.21. If f is a closed very strictly convex, co-finite, Legendre function and we are given a point $x^0 \in dom f$, then

1. if the oracle $\mathcal{Q}$ satisfies either property 4.1 or 4.2, then any sequence x^n produced by the above algorithm converges to Bregman projection of x^0 on C with respect to f; and,

2. if f is also a quadratic function, then for large enough ν there exists a $\rho \in (0, 1)$ such that
$$\|x^* - x^{\nu+1}\| \leq \rho \|x^* - x^{\nu}\|.$$

4.5 Applications: Metric Constrained Problems

To demonstrate the effectiveness of our method in solving metric constrained problems, we solve large instances of two different types of such problems: metric nearness and correlation clustering. We focus on these types of problems first as they were the original motivation behind our work. *

4.5.1 Metric Nearness

The first and simplest form of a metric constrained problem is the metric nearness problem. Following *Brickell et al.* (2008c), the metric nearness problem is:

> given a point $x \in \mathbb{R}^{\binom{n}{2}}$, find the closest (in some ℓ_p norm) point $x^* \in \text{MET}_n$ to x.

*All implementations and experiments can be found at `https://github.com/rsonthal/ProjectAndForget`.

This problem is a form of metric learning; see *Brickell et al.* (2008c) for an application to clustering and see *Gilbert and Sonthalia* (2018a) for an application to unsupervised metric learning. Additionally, if we further restrict to finding the closest Euclidean metric, then this problem is a well studied one. See *Qi and Yuan* (2014b); *Cayton and Dasgupta* (2006b), and *Glunt et al.* (1990c) for examples of this problem. Recently this problem has also been looked at from a discrete setting. *Gilbert and Jain* (2017) and *Fan et al.* (2018b) looked to solve the problem with the fewest possible changes. Following this, *Fan et al.* (2018a) generalized the problem to finding the closest point in $\text{MET}(G)$ instead of MET_n.

We use this problem to demonstrate that standard solvers have significant drawbacks on large scale metric constrained problems while PROJECT AND FORGET handles these problems easily. In particular, in addition to comparing against the cyclic Bregman used in *Brickell et al.* (2008c), we compare against commercial solvers CPLEX (*IBM*) and Mosek (*MOSEK ApS*), ADMM based solvers OSQP (*Stellato et al.*, 2017), SCS (*O'Donoghue et al.*, 2019) and COSMO (*Garstka et al.*, 2019), operator splitting and interior point solvers Ipopt (*Nocedal et al.*, 2009), ProxSDP (*Souto et al.*, 2018) and ECOS (*Domahidi et al.*, 2013), and active set solver SLSQP (*Kraft*, 1988). We also use Mosek as an active set solver (MASS) as follows. We go through all n^3 constraints and find the subset S of the violated constraints. We then use Mosek to solve the problem on S. We then remove the inactive constraints and add in the new violated constraints and use Mosek to solve the problem again and iterate.

4.5.1.1 Experimental Set Up:

Before we see the experimental results, let us look at the experimental set up more closely.

Data. For this experiment, we will generate three different types of synthetic data.

We will refer to these as Type I, Type II, and Type III data.

For Type I data, we generate random weighted complete graphs with Gaussian weights. For type II data, for each edge e we set $w(e) = 1$ with probability 0.8 and and $w(e) = 0$ with probability 0.2. For type III data, we let u_{ij} be sampled from the uniform distribution on $[0, 1]$ and v_{ij} from a standard normal, then the weight for an edge $e = ij$ is given by

$$w_{ij} = \left\lceil 1000 \cdot u_{ij} \cdot v_{ij}^2 \right\rceil .$$

Implementation Details. We implemented the algorithm from *Brickell et al.* (2008c). We made a small modification that improves the running time. In *Brickell et al.* (2008c), it is recommended that we store the dual variable z as a sparse vector. However, as we do not want the overhead of handling sparse vectors, we store z as a dense vector.

PROJECT AND FORGET, as presented Algorithm 11, was implemented with two modifications. As we can see from algorithm 8, when the oracle finds violated constraints, it looks at each edge in G and then decides whether there is a violated inequality with that edge. It is cleaner, in theory, to find all such violated constraints at once and then do the project and forget steps. It is, however, much more efficient in practice to do the project and forget steps for a single constraint as we find it. This approach also helps cut down on memory usage. Once our oracle returns a list of constraints (note we have already projected onto these once), we project onto our whole list of constraints again. Thus, for the constraints returned by the oracle, we project onto these constraints twice per iteration. Note this does not affect the convergence results for the algorithm.

The solvers CPLEX, Mosek, OSQP, SCS, COSMO, Ipopt, ProxSDP, ECOS, and MASS all were accessed via Julia's JuMP library. SLSQP was inferfaced with using Scipy.

Convergence criterion. When we presented the algorithm in Section 4.3, we did

Algorithm 11 Pseudo-code for the implementation for Metric Nearness.

$L^0 = \emptyset$, $z^0 = 0$. Initialize x^0 so that $\nabla f(x^0) = 0$.
while Not Converged **do**
    Let $d(i, j)$ be the weight of shortest path between nodes i and j or ∞ if none exists.
    $L = \emptyset$
    for Edge $e = i(, j) \in E$ **do**
        if $w(i, j) > d(i, j)$ **then**
            Let P be the shortest path between i and j.
            Let $C = P \cup \{(i, j)\}$.
            Project onto C and update x, z.
            if $z_C ! = 0$ **then**
                Add C to L.
    $\tilde{L}^{\nu+1} = L^\nu \cup L$
    $x^{\nu+1}, z^{\nu+1} = \text{Project}(x^\nu, z^\nu, \tilde{L}^{\nu+1})$
    $L^{\nu+1} = \text{Forget}(\tilde{L}^{\nu+1})$
return x

not specify a convergence criterion because we wanted the criterion to be application dependent. The convergence criterion for PROJECT AND FORGET and *Brickell et al.* (2008c) is as follows. One variant of the metric nearness problem is the decrease only variant, in which we are not allowed to increase the distances and must only decrease them. This problem can solved in $O(n^3)$ time by calculating the all pairs shortest path metric (*Gilbert and Jain*, 2017). Given x^n as input, let $\hat{x}^n$ be the optimal decrease only metric. For PROJECT AND FORGET, and *Brickell et al.* (2008c), we ran these experiments until $\|\hat{x}^n - x^n\|_2 \leq 10^{-10}$. For convenience, given x let $D(x) = \|\hat{x}^n - x^n\|_2$.

For CPLEX. Mosek, OSQP, SCS, COSMO, Ipopt, ProxSDP, ECOS, SLSQP, we let the convergence criterion be the default criterion. For MASS, we ran it until the constraint set converged.

Comparison Statsitics We compare the solvers on two different kinds of measures. The first is the convergence details of each solver. To determine convergence, we look two statistics. First, we look at the relative objective difference (ROD), which is given by

$$ROD = \frac{\text{OtherSolverObjective - ProjectForgetObjective}}{\text{ProjectForgetObjective}}.$$

Second, if x is the solution returned by PROJECT AND FORGET and $\tilde{x}$ is the solution returned by one of the other solvers, then the feasibility difference (FD) is given by

$$FD = D(\tilde{x}) - D(x).$$

For both metrics, if the quantities are positive, then PROJECT AND FORGET does better.

The second measure we compare is the time taken to solve the optimization problem. *Here we use the solve time reported by the solvers. Note that this does not include the interface time and in many cases the interface time could be significant.*

4.5.1.2 Results

As we can see from Table 4.1, standard solvers run out memory or start taking too long extremely rapidly as a function of instance size. In fact, all times reported for the standard solvers is the solve time returned by the optimizer and does not include the interface time. In many cases, such as ProxSDP, the interface time could be multiple hours. On the other hand, while initially *Brickell et al.* (2008c) is faster than PROJECT AND FORGET as n gets larger, PROJECT AND FORGET starts to dominate. Thus, showing that PROJECT AND FORGET is the only viable algorithm to solve large metric constrained problems.

To make sure that we are running all of the algorithms to the same level of convergence, we take the five best solvers and check their convergence statistics. That is, we check the convergence statistics for commercial solver Mosek, for ADMM based solvers OSQP and SCS, and for operator splitting solver ProxSDP.

Each data point in Table 4.2 is average over 10 trials. When comparing against *Brickell et al.* (2008c), we let n range from a 100 to 1000, i.e., the same values as in Table 4.1. Table 4.2 then shows that both method have similar levels of convergence.

Algorithm	Number of Nodes									
	100	200	300	400	500	600	700	800	900	1000
Ours	13.5	32.7	85.1	170	271	458	720	983	1356	1649
Cyclic Bregman	1.77	10.5	47.1	141	322	558	910	1472	2251	3167
Mosek	11.7	542	Out of Memory							
SCS	1632	19466	Timed Out							
OSQP	64.5	3383	Timed Out							
ProxSDP	353	684	Timed Out							
Ipopt	2792	Timed Out								
ECOS	597	Timed Out								
CPLEX	Out of Memory									
SLSQP	Timed Out									
COSMO	Timed Out									

Table 4.1: Table comparing Project and Forget against a variety of different solvers to solve the Metric Nearness problem for Type 1 graphs in terms of time taken in seconds. All experiments were run on a Computer with 52 GB of memory. All times reported are averaged over 5 instances.

Thus, since Project and forget is faster; it is the superior algorithm.

To compare against the rest of the solvers, as they simply cannot solve the problem for larger values of n, we let n range from 10 to 100. Here, we see that Project and Forget has much smaller feasibility error than the other solvers. Solvers such as SCS and QSQP have a feasibility error of about 10^{-3} to 10^{-5}, however, we run Project and Forget until the feasibilty error is smaller than 10^{-10}, which is many orders of magnitude smaller. For ProxSDP, ROD was consistently around 1 and FD was consistently greater than 2. We conclude that these solvers have not converged.

The only solver, other than Project and Forget, that consistently converges is Mosek, which has higher objective values than Project and Forget in all but one case. The active set method MASS was run until the the feasibility error was at most 1e-10 or the constraint set was stable, in practice we found that the constraint set stabilized first and hence the solver has similar convergence statistics to Mosek.

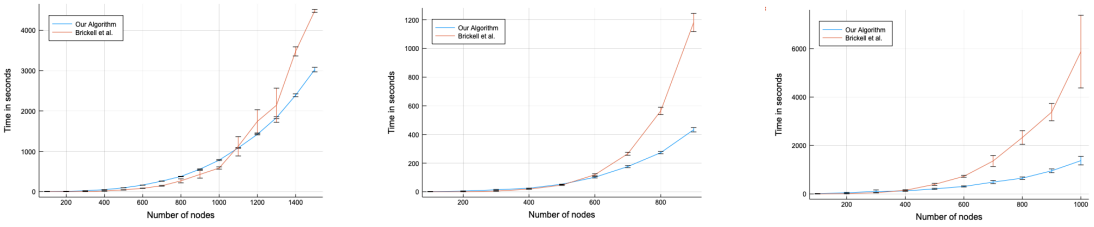
We further tested our method against the cyclic Bregman method on Type II and Type III data. In this case, we got the running times shown in Figure 4.1. For this we

Solver		n 100	200	300	400	500	600	700	800	900	1000
Cyclic Bregman	ROD	-3e-13	-4e-14	4e-15	-2e-14	6e-15	-8e-16	1e-15	-4e-15	5e-15	2e-17
	FD	-6e-12	5e-13	4e-13	-3e-12	1e-12	3e-12	5e-12	3e-12	-2e-12	7e-12
		n 10	20	30	40	50	60	70	80	90	100
Mosek	ROD	-4e-4	2e-9	1e-9	6e-9	7e-9	1e-8	2e-8	2e-8	2e-8	2e-8
	FD	2e-11	2e-11	1e-9	7e-10	3e-10	9e-10	1e-9	9e-10	8e-10	7e-10
OSQP	ROD	-4e-4	5e-7	-9e-7	-2e-7	-7e-8	1e-7	-1e-8	7e-8	4e-8	5e-8
	FD	1e-3	2e-3	1e-3	9e-4	8e-4	5e-4	9e-4	1e-3	8e-4	1e-3
SCS	ROD	-4e-4	-3e-8	8e-7	-8e-6	-5e-7	-1e-8	-2e-8	3e-9	-7e-6	1e-9
	FD	7e-5	1e-4	5e-3	6e-3	5e-4	9e-5	4e-5	4e-5	0.09	4e-5

Table 4.2: Convergence statistics for the metric nearness problem for the different solvers. Each value is the average over 10 trials.

relaxed the convergence criteria to being within 1 of the closest decrease only metric solution.

One thing we learn from relaxing the convergence criteria is that cyclic Bregman method outperforms PROJECT AND FORGET for larger values of n. This is because, that PROJECT AND FORGET only focuses on the set of the active constraints. Thus, the more stringent the convergence criterion, the better our method does compared to the standard cyclic Bregman method used in *Brickell et al.* (2008c).



(i) Type one graphs (ii) Type two graphs (iii) Type three graphs

Figure 4.1: The red line is the mean running time for the algorithm from *Brickell et al.* (2008c). The blue line is the running mean time for our algorithm. All computations were done on a machine with 4 physical cores, each with 13 GB of RAM.

Finally, we must note that when we use the cycle constraints to represent MET(K_n),

there is significant dependency amongst the constraints. That is, many constraints are not independent from other constraints. We suspect that our oracle finds a lot of active constraints that have linear dependencies and so $L^{(\nu)}$ becomes saturated with dependent constraints. It is suprising, however, that after we solve the problem, $L^{(\nu)}$, more often than not, only contains 3-cycle constraints and does not have a lot of dependency. We also see that the size of $L^{(\nu)}$ is about $O(n^2)$.

4.5.2 Weighted Correlation Clustering on General Graphs.

For the correlation clustering problem, we are given a graph $G = (V, E)$ (not necessarily complete) and two non-negative numbers $w^+(e)$ and $w^-(e)$ for each edge e. These numbers indicate the level of similarity and dissimilarity between the end points of e. The goal of correlation clustering is to partition the nodes into clusters so as to minimize some objective function. The most common objective is $\sum_{e\in E} w^+(e)x_e + w^-(e)(1-x_e)$, where $x_e \in \{0,1\}$ indicates whether the end points of the edge e belong to different clusters. This variant of the problem is NP-hard and many different approximation algorithms and heuristics have been developed to solve it. The best approximation results (*Charikar et al.*, 2005; *Emanuel and Fiat*, 2003), however, are obtained by rounding the solution to the following relaxed linear problem

$$\begin{array}{lll} \text{minimize} & \sum_{e\in E} w^+(e)x_e + w^-(e)(1-x_e) & \\ \text{subject to} & x_{ij} \le x_{ik} + x_{kj} & i,j,k = 1,...,n \\ & x_{ij} \in [0,1] & i,j = 1,...,n. \end{array} \tag{4.8}$$

Special cases, such as when the weights are ± 1 and $G = K_n$, have faster algorithms for the same approximation ratio (*Ailon et al.*, 2005).

The LP formulation for correlation clustering in Equation 4.8 has $\Theta(n^3)$ constraints. Hence, solving the LP for large n becomes infeasible quickly in terms of both memory and time. *Veldt et al.* (2019) showed that for instances with $n \approx 4000$, standard

solvers such as Gurobi ran out of memory on a 100 GB machine. On the other hand, *Veldt et al.* (2019) developed a method using which they can feasibly solve the problem for $n \approx 11000$. To do so, they transform Problem 4.8 into Problem 4.9. To do this transformation, for $e \in E$, we define $\tilde{w}(e) = |w^+(e) - w^-(e)|$. For $e \notin E$, we let $\tilde{w}(e) = 0$ Then W is a diagonal matrix whose entries are given by $\tilde{w}$. Then, we define d as follows

$$d_e = \begin{cases} 1 & w^{-1}(e) > w^+(e) \\ 0 & \text{otherwise} \end{cases}.$$

And the transformed problem is as follows.

$$\begin{aligned} \text{minimize} \quad & \tilde{w}^T|x-d| + \tfrac{1}{\gamma}|x-d|^T W|x-d|. \\ \text{subject to} \quad & x \in \text{MET}(K_n). \end{aligned} \tag{4.9}$$

For general γ, the solution to Problem 4.9 approximates the optimal solution to 4.8. However, for large enough γ it has been shown that the two problems are equivalent. As we will see for our instances, Problem 4.9 is at most a 2 approximation to problem 4.8, which is an $O(\log(n))$ approximation of the NP-hard correlation clustering problem.

Finally, we also relax the condition that $x \in \text{MET}_n$ to $x \in \text{MET}(G)$. The formulation of the LP that we solve is as follows:

$$\begin{aligned} \text{minimize} \quad & \tilde{w}^T f + \tfrac{1}{\gamma} f^T \cdot W \cdot f \\ \text{subject to} \quad & x \in \text{MET}(G) \\ & f_{ij} = |x_{ij} - d_{ij}|, \quad (i,j) \in E. \end{aligned} \tag{4.10}$$

While it seems like we have relaxed the problem, however, since our objective function does not depend on the value of x for the edges not in G we see that the two problems are equivalent.

Remark 4.22. When *Veldt et al.* (2019) tested their solver against Gurobi, they did so in an active set manner. That is, they found a set of violated constraints, fed this

set into Gurobi, and solved the problem with this subset of the constraints. They then took the solution from Gurobi and found constraints that the current solution violated and added these constraints and repeated. They did this until Gurobi solved the problem. We view this convoluted process as yet more evidence for standard active set methods not being feasible at large scales.

Proposition 4.23. Let $f(x)$ be a function whose values only depends on the values x_{ij} for $e = (i, j) \in G$ and consider the following constrained optimization problem.

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in \text{MET}(K_n). \end{aligned} \tag{4.11}$$

Let π be the projection from $\text{MET}(K_n)$ to $\text{MET}(G)$ and let $\tilde{f}(\pi(x)) = f(x)$. Then for any optimal solution x^* to the following problem

$$\begin{aligned} &\text{minimize} && \tilde{f}(x) \\ &\text{subject to} && x \in \text{MET}(G). \end{aligned} \tag{4.12}$$

we have that for all $\hat{x} \in \pi^{-1}(x^*)$, $\hat{x}$ is an optimal solution to 4.11.

Proof. Here, we see that if $\tilde{x}$ be a minimizer of Problem 4.11 and x^* is the minimizer of Problem 4.12 then

$$f(\tilde{x}) = \tilde{f}(\pi(\tilde{x})) \geq \tilde{f}(x^*) = f(\pi^{-1}(x^*)),$$

where the middle inequality is true, since $\pi(\tilde{x})$ need not be an optimal solution to Problem 4.12. Thus, any element in $\pi^{-1}(x^*)$ is an optimal solution to Problem 4.11. □

4.5.2.1 Experimental Set up

Before looking at the results, let us see the experimental set up.

Data. We solve the problem for two different types of graphs; dense and sparse.

For dense graphs, we use first take four graphs from the Stanford sparse network repository. Then, following *Veldt et al.* (2019), we use the method from *Wang et al.* (2013) to convert these graphs into instances of weighted correlation clustering on the complete graph. We compare our method against *Ruggles et al.* (2019), a parallel version of *Veldt et al.* (2019), in terms of running time, quality of the solutions, and memory usage.

For much real-world data, the graph G is larger than our previous experiments but is also sparse. Since the weighting of the edges does not affect the size of the linear program that needs to be solved, we tested our algorithm on sparse signed graphs to get an estimate of the running time for the algorithm. The two graphs used for this experiment are much bigger instances than our previous experiments and have 82140 nodes and 131,828 nodes, respectively.

Implementation Details. For the case when $G = K_n$, in addition to the modifications that were done for metric nearness experiment, we made two more modifications to PROJECT AND FORGET. First, we did the project step and the forget step one additional time per iteration. Second, we parallelized the oracle by running Dijkstra's algorithm in parallel. The pseudo-code for this version of Algorithm 7 can be seen in Algorithm 12. For the sparse version, we also used the parallel version of the oracle and during each iteration, but we did the project and the forget step 75 times per iteration.

Note that for both experiments, the additional constraints that were introduced due to the transformation were all projected onto once per iteration and never forgotten. The pseudo-code for this version can be seen in Algorithm 13. We used the implementation provided by the authors of *Veldt et al.* (2019) to run the experiments for their algorithm.

Calculating the approximation ratio. Let $\hat{x}$ be the optimal solution to 4.9,

Algorithm 12 Pseudo-code for the implementation for CC for the dense case.

$L^0 = \emptyset$, $z^0 = 0$. Initialize x^0 so that $\nabla f(x^0) = 0$.
L_a is the list of additional constraints. $z_a^0 = 0$ (dual for additional constraints)
while Not Converged **do**
    Let $d(i, j)$ be the weight of shortest path between nodes i and j or ∞ if none exists. This is found using a parallel algorithm.
    $L = \emptyset$
    for Edge $e = i(, j) \in E$ **do**
        if $w(i, j) > d(i, j)$ **then**
            Let $C = P \cup \{(i, j)\}$. Where P be the shortest path between i and j.
            Project onto C and update x, z.
            if $z_C! = 0$ **then**
                C to L.
    $L^\nu \leftarrow L^\nu \cup L$
    for $i = 1, 2$ **do**
        $x^\nu, z^\nu \leftarrow \text{Project}(x^\nu, z^\nu, L^\nu)$
        $L^\nu \leftarrow \text{Forget}(L^\nu)$
    $x^\nu, z_a^\nu \leftarrow \text{Project}(x^\nu, z_a^\nu, L_a)$
    $x^{\nu+1} = x^\nu, L^{\nu+1} = L^\nu, z^{\nu+1} = z^\nu, z_a^{\nu+1} = z_a^\nu,$
return x

Algorithm 13 Pseudo-code for the implementation for CC for the sparse case.

$L^0 = \emptyset$, $z^0 = 0$. Initialize x^0 so that $\nabla f(x^0) = 0$.
L_a is the list of additional constraints. $z_a^0 = 0$ (dual for additional constraints)
while Not Converged **do**
    Let $d(i, j)$ be the weight of shortest path between nodes i and j or ∞ if none exists. This is found using a parallel algorithm.
    $L = \emptyset$
    for Edge $e = i(, j) \in E$ **do**
        if $w(i, j) > d(i, j)$ **then**
            Let $C = P \cup \{(i, j)\}$. Where P be the shortest path between i and j.
            Add C to L.
    $L^\nu \leftarrow L^\nu \cup L$
    for $i = 1, \ldots, 75$ **do**
        $x^\nu, z^\nu \leftarrow \text{Project}(x^\nu, z^\nu, L^\nu)$
        $L^\nu \leftarrow \text{Forget}(L^\nu)$
    $x^\nu, z_a^\nu \leftarrow \text{Project}(x^\nu, z_a^\nu, L_a)$
    $x^{\nu+1} = x^\nu, L^{\nu+1} = L^\nu, z^{\nu+1} = z^\nu, z_a^{\nu+1} = z_a^\nu,$
return x

then if we let $R = \dfrac{\hat{x}^T W \hat{x}}{2\gamma \tilde{w}^T \hat{x}}$, by *Veldt et al.* (2019), we have that $\hat{x}$ is an $\dfrac{1+\gamma}{1+R}$ approximation to the optimal solution of 4.8. This is the formula we used to calculate the approximation ratios reported in Tables 4.3 and 4.4. For our experiments we used $\gamma = 1$. Note that since the entries of $\hat{x}$ are non negative, and the entries of W and $\tilde{w}$ are non-negative, we have that $R > 0$. Thus, our approximation ratio is at most 2.

Convergence criterion. We ran the experiment until the maximum violation of a metric inequality was at most 0.01. However, the two algorithms, *Ruggles et al.* (2019) and ours, have different metric constraints. Specifically *Ruggles et al.* (2019) only uses all constraints that come from 3 cycles, whereas we use all cycle constraints. Theoretically both sets of constraints define the same polytope, but practically there is a difference. Thus, in practice our algorithm was run to a slightly greater level of convergence than the one from *Ruggles et al.* (2019). Additionally, *Ruggles et al.* (2019) also check a second criteria for convergence, however, in all cases, that criterion was satisfied much before the maximum violation of a metric inequality was at most 0.01.

4.5.2.2 Results

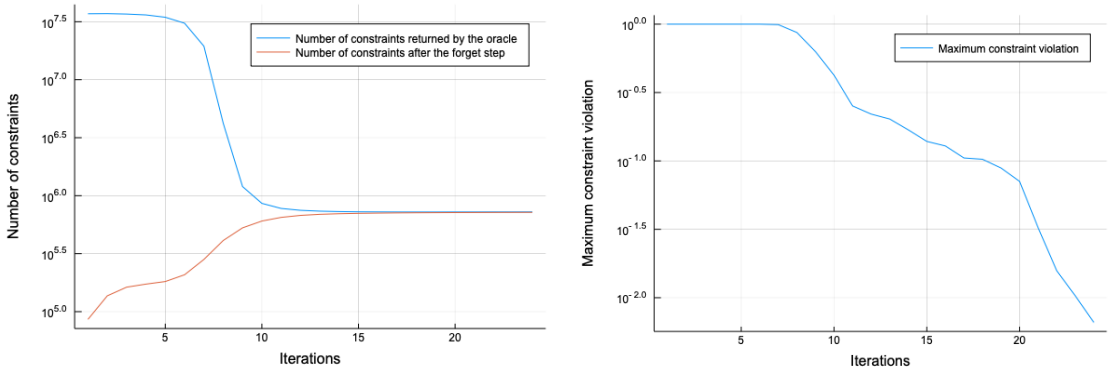
Graph		Time (s)		Opt Ratio		Avg. mem. / iter. (GiB)	
	n	Ours	*Ruggles et al.*	Ours	*Ruggles et al.*	Ours	*Ruggles et al.*
CAGrQc	4158	2098	5577	1.33	1.38	4.4	1.3
Power	4941	1393	6082	1.33	1.37	5.9	2
CAHepTh	8638	9660	35021	1.33	1.36	24	8
CAHepPh	11204	71071	135568	1.33	1.46	27.5	15

Table 4.3: Table comparing PROJECT AND FORGET against *Ruggles et al.* (2019) in terms of time taken, quality of solution, and average memory usage when solving the weighted correlation clustering problem on dense graphs.

For the case of dense graphs, we see from Table 4.3 that our algorithm takes less time to obtain a better approximation ratio, but requires more memory per iteration. Thus, demonstrating the superiority of our method in terms of CPU time. Our

algorithm requires more memory because the initial few iterations find a large number of constraints. Later, the algorithm forgets these constraints until the number of constraints stabilizes at a reasonable level. Hence, our initial memory usage is much larger than our later memory usage. To see how the number of constraints found by the oracle evolves, we plot the number of constraints found by the oracle and the number of constraints after the forget step in Figure 4.2i. As we can see, after the initial few iterations, the number constraints found sharply reduces and has found the true set of active constraints by the 15th iteration. Figure 4.2ii also shows us, as expected, *the exponential decay of the maximum violation of a metric constraint.*

Figure 4.2ii, also highlights another aspect our algorithm. That is, for the initial few iterations the error statistics do not decrease. In fact, we have experimentally seen that the error statistics may actually increase for the first few iterations. Proposition 4.17, which tells us that our algorithm spends the initial few iterations finding the active constraint set, explains this phenomenon. During this time, the algorithm makes minimal progress towards reducing the error statistics. However, once we have found the active constraints, Theorem 4.28 is now applicable and we have an exponential decay of the error statistics. Though in contrast to the theory result, the base of the exponent, is smaller than theoretically predicted.



(i) Number of constraints. (ii) Max Violation.

Figure 4.2: Plots showing the number of constraints returned by the oracle, the number of constraints after the forget step, and the maximum violation of a metric constraint when solving correlation clustering on the Ca-HepTh graph

Remark 4.24. Figure 4.2i highlights the crucial difference between our method and standard active set methods. Standard active set methods would have to initially solve the convex optimization problem with 10^8 constraints. This is not feasible! However, using PROJECT AND FORGET, we only need to compute projections onto each constraint once before we can forget constraints. Thus, we forget constraints more frequently and much earlier.

Graph	n	# Constraints	Time	Opt Ratio	# Active Constraints	Iters.
Slashdot	82140	5.54×10^{14}	46.7 hours	1.78	384227	145
Epinions	131,828	2.29×10^{15}	121.2 hours	1.77	579926	193

Table 4.4: Time taken and quality of solution returned by PROJECT AND FORGET when solving the weighted correlation clustering problem for sparse graphs. The table also displays the number of constraints the traditional LP formulation would have.

For sparse graphs, even if we use *Ruggles et al.* (2019), based on the average time it took for a single iteration for the CA-HepPh graph, it would take *Ruggles et al.* (2019) an estimated two days to complete a single iteration, for a graph with $n \approx 80,000$. This is because each of their iterations takes $\Theta(n^3)$ time. Since most graphs require at least 100 iterations, *Veldt et al.* (2019); *Ruggles et al.* (2019) cannot be used to solve problems of this magnitude. Other methods of solving the LP are also not feasible as they run out of memory on much smaller instances.

We can see the performance of our method PROJECT AND FORGET in Table 4.4. Here, we see that our method has solved these problems in a reasonable amount of time. As we can see from Table 4.4, these instances have over 500 trillion constraints, but the number of active constraints is only a tiny fraction of the total number of constraints. Thus, using our approach, we can solve the weighted correlation clustering problem on much larger graphs than ever before.

There are two reasons as to why PROJECT AND FORGET can be used to solve these problems at these scales. First, while these instances have over 500 trillion

constraints, the number of active constraints is only a tiny fraction of the total number of constraints. Second, due to the sparsity of the graph, our oracle finds violated cycle inequalities relatively quickly (sub-cubic time) and, since we forget inactive constraints, we project onto this relatively small number of constraints in each iteration. Thus, using our approach, we can solve the weighted correlation clustering problem on much larger (10 times bigger) graphs than ever before.

4.6 Applications: General algorithm

We shall now also provide experimental evidence that supports the generality of the algorithm. To do so we will solve three different problems, each with general linear constraints (not simply metric constraints) and a variety of objective functions.

First, we solve the dual of the quadratically regularized optimal transport problem from *Blondel et al.* (2018a). Solving this problem helps highlight various different aspects of our algorithm that are not highlighted by the metric constrained problem. Second, we solve the information theoretic metric learning from *Davis et al.* (2007). Solving this problem highlights the wide variety of objective functions that our method can handle. Finally, we use our algorithm to train L_2 SVMs to highlight cases when the truly stochastic variant of our algorithm would be useful.

4.6.1 Sparse Optimal Transport

Optimal transport is a ubiquitous problem that appears many different areas of machine learning, data science, statistics, mathematics, physics, and finance.

Problem 4.24.1. Given two measure spaces $(\mathcal{X}, \Sigma_X, \mu)$ and $(\mathcal{Y}, \Sigma_Y, \nu)$, and a cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, the Monge-Kanterovich Optimal Transport seeks joint

probability π on $\mathcal{X} \times \mathcal{Y}$ that minimizes

$$\int_{\mathcal{X}\times\mathcal{Y}} c(x,y)d\pi(x,y),$$

subject to constraint that the marginals of π are equal to μ and ν.

In the discrete setting this problem can be formulated as follows:

$$\begin{aligned} &OT(\boldsymbol{a},\boldsymbol{b}) = \min\langle \boldsymbol{C}, \boldsymbol{P}\rangle \\ &\text{Subject to: } \ \boldsymbol{a} = \boldsymbol{P}\mathbf{1}_m,\ \boldsymbol{b} = \boldsymbol{P}^T\mathbf{1}_n \end{aligned} \tag{4.13}$$

It has been shown that optimal solutions to the discrete Monge-Kanterovich problem are sparse (*Brualdi*, 2006). Specifically, at most $n+m-1$ entries of P can be non-zero.

While Problem 4.24.1 can be formulated as a linear program, solving it is fairly challenging as it has a quadratic number of variables and so many alternative formulations have been proposed. One such alternative, called ROT, is presented in *Blondel et al.* (2018b); it uses a different regularizer, most importantly an L_2 regularizer. In this case, they show experimentally that the solutions are still sparse. We show that Project and Forget can also be used to solve ROT.

$$\begin{aligned} &\text{Minimize:} \quad \langle \boldsymbol{C}, \boldsymbol{P}\rangle + \gamma\|\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m\|^2 + \gamma\|\boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_n\|^2 \\ &\text{Subject to:} \quad \forall i \in [n], \forall j \in [m], \quad \boldsymbol{P}_{ij} \geq 0 \end{aligned} \tag{4.14}$$

4.6.1.1 Dual Problem

In particular, we will not solve ROT directly; instead, we will solve the dual formulation of ROT.

$$\begin{aligned} &\text{Min:} \qquad \tfrac{1}{\gamma}\|\boldsymbol{f}\|^2 + \tfrac{1}{\gamma}\|\boldsymbol{g}\|^2 - \langle \boldsymbol{f}, \boldsymbol{a}\rangle - \langle \boldsymbol{g}, \boldsymbol{b}\rangle \\ &\text{Subject to:} \ \boldsymbol{f}_i + \boldsymbol{g}_j \leq \boldsymbol{C}_{ij} \end{aligned} \tag{4.15}$$

Generally, solving the dual problem, while it gives us the optimal value of the

primal objective function, does not give you the optimal value of the primal variables directly. Thus, going from the solution to Problem 4.15 to that of Problem 4.14 is non trivial. This brings us to one aspect of PROJECT AND FORGET that is not highlighted by the metric constrained problems. That is,

> if we use PROJECT AND FORGET to solve **either** the primal problem or the dual problem, then it finds the optimal solution to **both** problems.

This result holds as a result of Proposition 4.17. This proposition states that the variables z converge. Since we maintain the KKT conditions throughout the algorithm, the value that z converges to is the optimal solution of the dual problem and, thus, we solve the dual problem.

4.6.1.2 Sparsity

The reason we want to solve the dual problem instead of the primal is that sparsity impacts PROJECT AND FORGET advantageously. Switching to the dual gives three forms of beneficial sparsity.

First, we have that the primal problem has $O(n^2)$ variables while the dual problem has $O(n)$ variables. In general, having fewer variables is beneficial and while this typically comes at the expense of extra constraints, our method is built for a large number of constraints. Second, we know that the solutions to optimal transport have at most $O(n)$ non-zero entries. Hence, in this case, this means that the dual problem has $O(n)$ active constraints. Since our method is an active set method, having fewer active constraints is a great benefit. Third, the dual constraints are extremely sparse. Each dual constraint involves two variables. This allows the extremely rapid computation of the projection.

4.6.1.3 Experimental Results

In this subsection, we detail the experimental set up and present the results.

Data. The experimental set up is as follows. We take two shifted Gaussian distributions with means ± 15 and variance 10. Then, we split the interval $[-20, 20]$ into n points and create two discrete distributions by sampling the Gaussians on those n points. We use the squared Euclidean distances between the points as the cost function. We set the regularization parameter $1/\gamma = 10^{-3}$. This set up is a very basic example of the optimal transport problem and is an important test example for algorithmic comparisons.

Convergence Details. The feasibility error for Mosek and CPLEX are those reported by the solvers. For PROJECT AND FORGET, we calculate the feasibility error by

$$\max_{i,j} \boldsymbol{f}_i + \boldsymbol{g}_j - \boldsymbol{C}_{ij}.$$

Solvers. For the oracle for the PROJECT AND FORGET, we simply look through all the constraints and the return all of the violated constraints. This oracle clearly satisfies Property 4.1. We solve the dual version of the problem using PROJECT AND FORGET (PF), Mosek, and CPLEX and the primal version of the problem using Mosek, CPLEX, LBFGSB, and projected gradient descent (PGD).

Results. Table 4.5 shows that PROJECT AND FORGET can be used to solve the problem for much larger values of n. Additionally, for the smaller values of n, we see that PROJECT AND FORGET is competitive in terms of solve time. One thing to note from Table 4.6 is that the difference between the Primal objective and the Dual objective for PROJECT AND FORGET is smaller than $1e-7$; however, for Mosek, and CPLEX this is not the case. Thus, while both have similar levels of feasibility error, we see that our solver is still more "converged" since the primal dual gap is smaller.

4.6.2 Information Theoretic Metric Learning

The next problem we consider is metric learning. There are many different versions of this problem (see *Bellet et al.* (2013b); *Suárez Díaz et al.* (2018) for surveys on

Algorithm	501	1001	5001	10001	20001
PF	12	151	1972	5909	21665
LBFGSB	24	162	4080	Out of memory.	
Mosek dual	56	328	1927	Out of memory.	
Mosek primal	5	Out of memory.			
CPLEX primal	105	Out of memory.			
CPLEX dual	Out of memory.				
PGD	Did not converge.				

Table 4.5: Time taken in seconds to solve the quadratic regularized optimal transport problem. All experiments were run on a machine with 52 GB of RAM.

the topic). We focus on a specific instantiation, information theoretic metric learning (ITML) from *Davis et al.* (2007). This formulation allows us to demonstrate the wide range of objective functions that our method can handle.

Given two sets S, D that represent the sets of similar and dissimilar points, we learn the co-variance matrix M of a Gaussian distribution $p(x; M)$. The metric learned is then the Mahalanobis metric associated with M. Concretely, the problem is as follows:

$$\begin{aligned} \text{minimize} \quad & KL\big(p(x;M) \| p(x;I)\big) \\ \text{subject to} \quad & d_A(x_i, x_j) \leq u \qquad (i,j) \in S \\ & d_A(x_i, x_j) \geq l. \qquad (i,j) \in D. \end{aligned} \tag{4.16}$$

Davis et al. (2007) suggest solving this problem by sampling a small subset of the constraints and then using Bregman's cyclic method. However, since our method is solves the problem with the large number of constraints, when we compare against the method from *Davis et al.* (2007), we can no longer compare in terms of convergence statistics or time taken to solve the problem. Hence, as the method from *Davis et al.* (2007) is also based on Bregman projections, we limit both algorithms to the same number of projections. Then, we compare the algorithms on the quality of the solution; that is, the testing accuracy based on the metrics learned.

For each data set, we uniformly at random choose 80% of the data points to be

	Objective		Feasibility Error	
Solver	Dual	Primal	Dual	Primal
	$n = 501$			
Project and Forget	3.8416077	3.8416077	1.7e-09	0
Mosek Dual	3.8414023	3.8414023	3.8e-08	2.8e-10
LBFGSB	n/a	3.8416114	n/a	0
Mosek Primal	3.8303160	3.8303203	3.5e-8	2.2e-11
CPLEX Primal	3.8416376	3.8416076	8.44e-07	1.13e-04
CPLEX Dual	Ran out of memory			
	$n = 1001$			
Project and Forget	1.947532046	1.947532046	2.0e-8	0
Mosek Dual	1.947091229	1.947091203	2.7e-08	8.4e-11
LBFGSB	n/a	1.947548404	n/a	0
Mosek Primal	Ran out of memory			
CPLEX Primal	Ran out of memory			
CPLEX Dual	Ran out of memory			
	$n = 5001$			
Project and Forget	3.946556152e-01	3.946556154e-01	2e-08	0
Mosek Dual	3.880175376e-01	3.880175255e-01	1.2e-08	7.4e-11
LBFGSB	n/a	3.947709104e-01	n/a	0
Mosek Primal	Ran out of memory			
CPLEX Primal	Ran out of memory			
CPLEX Dual	Ran out of memory			
	$n = 10001$			
Project and Forget	0.197687348	0.197687348	2.0e-8 0	
Mosek Dual	Ran out of memory			
LBFGSB	Ran out of memory			
Mosek Primal	Ran out of memory			
CPLEX Primal	Ran out of memory			
CPLEX Dual	Ran out of memory			
	$n = 20001$			
Project and Forget	0.0989355070	0.0989355070	2.0e-8 0	
Mosek Dual	Ran out of memory			
LBFGSB	Ran out of memory			
Mosek Primal	Ran out of memory			
CPLEX Primal	Ran out of memory			
CPLEX Dual	Ran out of memory			

Table 4.6: Table showing the convergence details for the various solvers.

the training set and the remaining to be the test set. We then let the similar pairs S be those pairs that have the same label and the dissimilar pairs D be all of the other pairs. For the algorithm from *Davis et al.* (2007), as suggested in the paper, we randomly sampled $20c^2$ constraints, where c is the number of different classes and run the algorithm so that it performes approximately 10^6 projections. We implement the algorithm in Julia. PROJECT AND FORGET returns $50c^2$ uniformly sampled similarity constraints and $50c^2$ uniformly random dissimilarity constraints and runs until we perform approximately 10^6 projections.

The hyper-parameters were set as follows: $\gamma = 1, u = 1, l = 10$. The pseudo-code for our algorithm can be seen in Algorithm 14. The classification is done using the k nearest neighbor classifier, with $k = 5$.

Algorithm 14 Pseudo-code for the Project and Forget algorithm for ITML.

function PFITML(X, C, γ, u, l, S, D)
  $\lambda^0 = 0$, $\Xi_{ij} = u$ for $(i, j) \in S$ and $\Xi_{ij} = l$ for $(i, j) \in D$. Initialize $C = I$.
  while Not done 1e7 projections **do**
    Randomly sample $50c^2$ (i, j) from S and $50c^2$ from D
    Do projection for these constraints.
    Project onto all remembered constraints.
  return C
function PROJECTION($X, i, j, S, D, u, l, \Xi, \lambda, C$)
  $p = dist_C(X_i, X_j)$
  $\delta = 1$ if $(i, j) \in S$ and $\delta = -1$ if $(i, j) \in D$
  $\alpha = min\left(\lambda_{ij}, \frac{\delta}{2}\left(\frac{1}{p} - \frac{\gamma}{\Xi_{ij}}\right)\right)$
  $\beta = \frac{\delta\alpha}{1-\delta\alpha p}$
  $\Xi_{ij} = \gamma\frac{\Xi ij}{\gamma+\delta\alpha\Xi_{ij}}$
  $\lambda_{ij} = \lambda_{ij} - \alpha$
  $C = C + \beta C(x_i - x_j)(x_i - x_j j)^T C^T$

As shown in Table 4.7, our algorithm results in, for the most part, better test accuracy. The test accuracy is significantly better for the Ionosphere data set. This is the smallest of the data sets, from which we conclude that if PROJECT AND FORGET can solve the entire problem, instead of the approximation from *Davis et al.* (2007), we get a performance boost. As the size of the problem gets larger, since we have

Dataset	|S|	|D|	Project and Forget		ITML	
			Test Accuracy	#Projections	Test Accuracy	#Projections
Banana	4.5e6	4.5e6	89.6%	1.003e6	89.8%	1e6
Ionsphere	2.1e4	1.8e4	88.3%	1.007e6	83.7%	1e6
Coil2000	2.7e7	3.5e6	93.3%	1.02e6	93.2%	1e6
Letter	4.9e6	1.2e8	90.0%	1.08e6	92.6%	1.0004e6
Penbased	3.9e6	3.5e7	98.5%	1.08e6	98.2%	1e6
Spambase	3.5e6	3.2e6	92.5%	1.01e6	90.7%	1e6
Texture	8.8e5	8.8e6	99.8%	1e6	98.7%	1.002e6

Table 4.7: Table comparing the testing accuracy of Project and Forget and ITML. Numbers are averaged over 5 trials.

limited the number of projections, our algorithm has not converged as much. On the other hand, since the size of the problem for *Davis et al.* (2007) does not depend on the size of S and D, its convergence does not change.

4.6.3 Support Vector Machines

Finally, to show the usefulness of the truly stochastic variant, we use our algorithm to train L_2 SVMs. Given training data $x_1, \ldots, x_n$, a feature map ϕ, and labels $y_1, \ldots, y_n \in \{-1, 1\}$, we want to learn a vector w and slack variables ξ_i such that

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\tilde{w}^T w + \tfrac{C}{2}\textstyle\sum_i \xi_i^2 \\ \text{subject to} \quad & y_i(w^T\phi(x_i)) \geq 1 - \xi_i \quad \forall\, i = 1, \ldots n \\ & \xi_i \geq 0, \qquad\qquad i = 1, ..., n. \end{aligned}$$

Here we use an L_2 penalty for the slack variables instead of the usual L_1 penalty. *Tang* (2013) provides insights into when using an L_2 penalty would provide better results.

Joachims et al. (2009) is an cutting plane based method to train L_2 SVMs. However, in contrast with *Joachims et al.* (2009), we do not need to use an oracle that finds the maximum violated constraint. Instead, we used the truly stochastic variant of our

algorithm. Hence, instead of searching through our data points to find large violations, we pick a subset of data points uniformly at random and project onto the constraints defined by these data points.

implementation Details. We used the LIBLINEAR implementation that is available online. We interfaced with LIBLINEAR using an interface written in Julia. The implementation for our algorithm follows the pseudo-code presented in algorithm 15.

Algorithm 15 Pseudo-code for the implementation training an SVM.

```
z^0, w^0 = 0. n is the number of training samples
for iter = 1, ..., MaxIters do
    for i = 1, ..., n do
        x_j = random data point
        Project w onto the constraint defined by x_j.
return w
```

Data. We generated the data matrix X as follows. Each entry X_{ij} was sampled from $\mathcal{N}(0, K^2)$. We then sampled the coordinates of H from $\mathcal{N}(0, 1)$ and used H to label all the points in X. We then added noise so that H did not correctly separate the data. This noise was also sampled from $\mathcal{N}(0, 1)$. We generate data in this manner because a standard pre-processing step is to normalize features (i.e., transform the data so that each feature has mean zero and variance one).

We used $K = 1, 2$ and 5 to generate our data set. The different values for K gave us the different values of s that we report in table 4.8.

Convergence criterion. We considered a few different convergence criteria, but settled upon running the algorithm for 100 projections since whenever we train an SVM, we have a validation set. This number of projections becomes a hyperparamter in the model and, we found experimentally that increasing the number of projections (unless increasing it by several orders of magnitude) beyond 100 did not increase the validation accuracy.

4.6.3.1 Results

Table 4.8 gives us the results for the experiment. In this experiment, we are not checking for convergence, but demonstrating a possible use case of the truly stochastic variant of our algorithm. Because of the way in which we have generated data, it is easy to see that we have good test accuracy only if the current separating hyper-plane is roughly similar to the optimal one. We also see that this method has roughly similar performance to the LibLinear dual solver. However, the primal solver from Liblinear has the best testing accuracy. Thus, we see that the truly stochastic version of the algorithm can potentially be used to get reasonable results quickly.

Parameters				Time			Test Acc.		
n	d	s	C	Ours	Liblinear		Ours	Liblinear	
					Dual	Primal		Dual	Primal
1,000,000	100	6.3%	10^3	1.33 s	547 s	9.87 s	94.7%	94.7%	96.8%
1,000,000	100	12.6%	10^3	1.38 s	1032 s	10.1 s	91.4%	90.2%	93.6%
1,000,000	100	29.5%	10^3	1.58 s	1532 s	8.32 s	78.4%	77.1%	85.2%

Table 4.8: Table comparing the testing accuracy and running times for the truly stochastic variant of or algorithm against LibLinear for binary classification using an L_2 SVM.

4.7 Conclusion and Future work

In conclusion, in this paper, we present a new algorithm Project and Forget that can be used to solve highly constrained convex optimization problems. We show that under some general assumptions, our method has a linear rate of convergence. Additionally, we demonstrate that our method returns both the dual and primal solutions.

The main type of problems we are interested in solving are metric constrained problems. We show that standard solvers cannot solve metric constrained problems at even moderate scales. We then demonstrate that Project and Forget can

solve the problem at large scales. We do this by solving larger instances of the metric nearness and correlation clustering problems.

We also demonstrate the generality of the method. First, we show that a variety of objective functions can be used with our methods, as demonstrated by the information theoretic metric learning experiment. We also demonstrate the role of sparsity and how it helps our method using the optimal transport experiment.

For future work, we hope that our method inspires a new way of learning metrics on data sets. Earlier researchers were either restricted to learning embeddings and then extracting a metric from those embeddings. However, using our method, we can now learn a much wider variety of metrics. We also hope that once we can learn optimal metrics on data, these optimal metrics then suggest the types of spaces the data should be embedded into, rather than users picking the space in which to embed in an ad hoc fasion. We also hope to adapt the general convex constraint version to solve highly constrained semi-definite programs.

4.8 Proofs

4.8.1 Proof of part 1 of Theorem 4.16 for oracles that satisfy property 4.1

We remind the reader of the notation established in Section 4.2. The vector of variables over which we optimize is x, f is the objective function, $H_i = \{y : \langle y, a_i \rangle = b_i\}$ are the hyper-planes that lie on the boundaries of the half-space constraints, $L(x, z)$ is the Lagrangian, z is the dual variable, A is the matrix with rows given by a_i, and b is the vector with rows b_i.

Next, we clarify the indexing of the variables. Algorithm 7 has three steps per iteration and during the PROJECT step there are multiple projections. When we want to refer to a variable after the νth iteration, it will have a superscript with a ν. When

we refer to a variable after the nth, ith, kth projection, we use the superscript n,i,k. Finally, before the nth projection, $i(n)$ will represent the index of the hyper-plane onto which we project.

Finally, let R be the maximum number of constraints that our oracle $\mathcal{Q}$ returns. This is clearly upper bounded by the total number of constraints, which we have assumed is finite. We are now ready to prove the first part of Theorem 4.16.

Theorem 4.16. *(Part 1) If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, $\exists\, x^0 \in S$, such that $\nabla f(x^0) = 0$, and the oracle $\mathcal{Q}$ satisfies property 4.1, then any sequence x^n produced by Algorithm 7 converges to optimal solution of problem 4.2.*

Proof. The proof of this theorem is an adaptation of the proof of convergence for the traditional Bregman method that is presented in *Censor and Zenios* (1997) whose proof entails the following four steps. The main difference between Censor and Zenios' proof and ours is that of the last two steps. We present the entire proof, however, for completeness. To that end, we show the following.

- Step 1. The KKT condition, $\nabla f(x) = \nabla f(x^0) - A^T z$, is always maintained.
- Step 2. The sequence x^n is bounded and it has at least one accumulation point.
- Step 3. Any accumulation point of x^n is feasible (i.e., is in C).
- Step 4. Any accumulation point is the optimal solution.

Step 1. The KKT condition, $\nabla f(x) = \nabla f(x^0) - A^T z$, is always maintained.

We show by induction that for all n, $\nabla f(x^n) = -A^T z^n$. In the base case, $z = 0$, thus, $\nabla f(x^0) = 0 = -A^T z^0$. Assume the result holds for iteration n, then

$$\begin{aligned}\nabla f(x^{n+1}) &= \nabla f(x^n) + c^n a_{i(n)} \\ &= -A^T z^n + A^T c^n e_{i(n)} \\ &= -A^T(z^n - c^n e_{i(n)}) \\ &= -A^T z^{n+1}.\end{aligned}$$

We know that $c^n \leq z^n_{i(n)}$; therefore, we maintain $z^{n+1} \geq 0$ as well.

Step 2. The sequence x^n is bounded and has an accumulation point.

To show that x^n is a bounded, we first show that $\big(L(x^n, z^n)\big)_n$ is a monotonically increasing sequence bounded from above. This observation results from the following string of equalities.

$$\begin{aligned}L(x^{n+1}, z^{n+1}) - L(x^n, z^n) &= f(x^{n+1}) - f(x^n) + \langle z^{n+1}, Ax^{n+1} - b\rangle - \langle z^n, Ax^n - b\rangle \\ &= f(x^{n+1}) - f(x^n) + \langle A^T z^{n+1}, x^{n+1}\rangle - \langle A^T z^n, x^n\rangle - \langle z^{n+1} - z^n, b\rangle \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^{n+1}), x^{n+1}\rangle \langle \nabla f(x^n), x^n\rangle + \langle c^n e_{i(n)}, b\rangle \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^n) + c^n a_{i(n)}, x^{n+1}\rangle + \langle \nabla f(x^n), x^n\rangle + c^n b_{i(n)} \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^n), x^{n+1} - x^n\rangle - \langle c^n a_{i(n)}, x^{n+1}\rangle + c^n b_{i(n)} \\ &= \underbrace{D_f(x^{n+1}, x^n)}_{(1)} + \underbrace{c^n(b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle)}_{(2)}.\end{aligned}$$

Next, we show that both terms (1) and (2) are non-negative. We know that D_f is always non-negative so we only need to consider term (2). There are two cases: (i) if $c^n = \theta^n$, then $x^{n+1} \in H_{i(n)}$ and $b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle = 0$. On the other hand, (ii) if $c^n = z^n_{i(n)}$, then $b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle \geq 0$ and $c^n \geq 0$. We can conclude that the difference between successive terms of $L(x^n, z^n)$ is always non-negative and, hence, it is an increasing sequence.

To bound the sequence, let y be a feasible point (i.e., $Ay \leq b$). (Note that this is the only place we use the assumption that the feasible set is not empty.) Then

$$\begin{aligned} D_f(y, x^n) &= f(y) - f(x^n) - \langle \nabla f(x^n), y - x^n \rangle \\ &= f(y) - f(x^n) + \langle z^n, Ay - Ax^n \rangle \\ &\leq f(y) - f(x^n) + \langle z^n, b - Ax^n \rangle. \end{aligned}$$

Rearranging terms in the inequality, we obtain a bound on the sequence $L(x^n z^n)$ from above:

$$\begin{aligned} L(x^n, z^n) &= f(x^n) + \langle z^n, Ax^n - b \rangle \\ &\leq f(y) - D_f(y, x^n) \\ &\leq f(y). \end{aligned}$$

Since the sequence $(L(x^n, z^n))_{n \in \mathbb{N}}$ is increasing and bounded, it is a convergent sequence and the difference between successive terms of the sequence goes to 0. Therefore,

$$\lim_{n \to \infty} D_f(x^{n+1}, x^n) = 0.$$

From the previous inequality we also have that

$$D_f(y, x^n) \leq f(y) - L(x^n, z^n) \leq f(y) - L(x^0, z^0) =: \alpha.$$

Using part (ii) of the definition of a Bregman function, we see that $L_2^f(y, \alpha)$ is bounded and since $(x^n)_{n \in \mathbb{N}} \in L_2^f(y, \alpha)$, x^n is an infinite bounded sequence. Thus, has an accumulation point.

Step 3. Any accumulation point x^* of x^n is feasible (i.e., is in C).

This is the only step in which we use the fact that our oracle satisfies property 4.1.

Let x^* be some accumulation point for x^n and assume for the sake of contradiction that $Ax^* \not\leq b$. Let $\tilde{A}, \tilde{b}$ be the maximal set of constraints that x^* does satisfy; i.e.,

$$\tilde{A}x^* \leq \tilde{b}.$$

Let (x^{n_k}) be a sub-sequence such that $x^{n_k} \to x^*$ and H be a constraint that x^* violates. Define ϵ as

$$\epsilon := \phi(d(x^*, H)) > 0. \tag{4.17}$$

Because x^n is bounded, x^{n_k} is a convergent sub-sequence $x^{n_k} \to x^*$, and $D_f(x^{n+1}, x^n) \to 0$, by Equation 6.48 from *Censor and Zenios* (1997) we see that for any t,

$$x^{n_k+t} \to x^*.$$

In particular, the proposition holds for all $t \leq 2|\tilde{A}| + 2 =: T$.

Let us consider an augmented sub-sequence $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$, i.e., add in extra terms. Note that if $n_{k+1} - n_k \to \infty$, then this augmented sequence is not the entire sequence. We want to show that infinitely many of the terms in our augmented sequence satisfy a constraint *not* in $\tilde{A}$. Should this hold, then because we have only finitely many constraints, there exists at least one single constraint $\tilde{a}$ that is *not* in $\tilde{A}$, such that infinitely many terms of the augmented sequence all satisfy the single constraint $\tilde{a}$. Finally, because our augmented sequence converges to x^* and we are only looking at closed constraints, we must have that x^* also satisfies the constraint $\tilde{a}$. Thus, we would arrive at a contradiction of the maximality of $\tilde{A}$ and x^* would have to be in the feasible region.

To see that infinitely many of the terms in our augmented sequence satisfy a constraint *not* in $\tilde{A}$, let ν_k be the iteration in which the n_kth projection takes place. Note that we can assume without loss of generality that in any iteration, we project

onto any constraint at most once. If this were not the case and we projected onto constraints more often, we would simply change the value of T to reflect this larger number of projections. Therefore, we have two possibilities for which iteration the $n_k + |\tilde{A}| + 1$st projection takes place and we consider each case below.

Case 1: The $n_k + |\tilde{A}| + 1$st projection, infinitely often, takes places in ν_kth iteration. Since we project onto each constraint at most once, one of the projections between the n_k and $n_k + |\tilde{A}| + 1$st projection must be onto a hyper-plane defined by a constraint *not* represented in $\tilde{A}$ and amongst the terms $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+|\tilde{A}|+1}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$ infinitely often.

Case 2: The $n_k + |\tilde{A}| + 1$st projection, infinitely often, takes place in $\nu_k + 1$st iteration or later.

If this projection happens in $\nu_k + 1$st iteration, consider the iteration in which we do the $n_k + T$th projection. If this projection also takes place in the $\nu_k + 1$st iteration, then we have done at least $|\tilde{A}| + 1$ projections in the $\nu_k + 1$st iteration. Hence, amongst $x^{n_k+|\tilde{A}|+1}, \ldots, x^{n_k+T}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$.

If the $n_k + |\tilde{A}| + 1$st or the $n_k + T$th projection happens in the $\nu_k + 2$nd iteration or later, then between the n_kth and the $n_k + T$th projection, we must have projected onto all constraints returned by oracle in the $\nu_k + 1$st iteration. Therefore, we must have projected onto some hyper-plane defined by $\hat{a}^{n_k}$ (for some constraint $\hat{C}^{n_k}$) such that

$$\hat{d}^{n_k} := d(x^{\nu_k+1}, \hat{C}^{n_k}) \geq \phi(d(x^{\nu_k+1}, C)).$$

Then there exists a sufficiently small $\delta > 0$, depending on $\tilde{A}, \tilde{b}, x^*$, such that if $\|y - x^*\| \leq \delta$, then

$$\tilde{A}y \leq \tilde{b} + \frac{\epsilon}{2}\mathbb{1},$$

where $\mathbb{1}$ is vector of all ones.

Since our augmented sequence converges to x^*, we know that there exists a K,

such that for all $k \geq K$ and $t \leq T$, $\|x^{n_k+t} - x^*\| < \delta$. That is, for all $k \geq K$ and $t \leq T$,

$$\tilde{A}x^{n_k+t} \leq \tilde{b} + \frac{\epsilon}{2}\mathbb{1}. \tag{4.18}$$

Note x^{ν_k+1} is within our augmented sequence so if $\hat{a}$ is infinitely often in $\tilde{A}$, by equation 4.18, we have that infinitely often

$$\frac{\epsilon}{2} \geq \hat{d}^{n_k}.$$

Finally, because the augmented sequence converges to x^*,

$$\begin{aligned}
\epsilon &= \phi(d(x^*, H)) \\
&\leq \phi(d(x^*, C)) \\
&= \lim_{k\to\infty} \phi(d(x^{\nu_k+1}, C)) \\
&\leq \lim_{k\to\infty} \hat{d}^{n_k} \\
&\leq \frac{\epsilon}{2}.
\end{aligned}$$

The first inequality follows from the fact that ϕ is non-decreasing. Thus , we have a contradiction. Therefore, $\hat{a}^{n_k}$ is *not* in $\tilde{A}$ infinitely often and amongst $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$ infinitely often. Thus, there is a constraint $\tilde{a}$ not in $\tilde{A}$ that is satisfied by infinitely terms of our augmented sequence and we have a contradiction.

Step 4. Optimality of accumulation point.

Because we have established the feasibility of all accumulation points, we show next that any accumulation point $x^{n_k} \to x^*$ is optimal.

First, we show that there exists an N, such that for any $k \geq N$, and for any a_i such that

$$\langle a_i, x^* \rangle < b_i,$$

we have $z_i^{n_k} = 0$. To do so, we assume for the sake of contradiction that for some a_i, our sequence $z_i^{n_k}$ is infinitely often not 0. The algorithm then projects onto this constraint infinitely often. Therefore, the point x^{n_k} lies on the hyper-plane defined by a_i, b_i infinitely often. Thus, the limit point x^* must lie on this hyper-plane as well and we have a contradiction.

Now we know that for any constraint a_i, we either have that $\langle a_i, x^* \rangle = b_i$ or we have that $z_i^{n_k} = 0$ for the tail of the sequence. Thus, for sufficiently large k,

$$\begin{aligned}\langle z^{n_k}, Ax^{n_k} - b \rangle &= \langle A^T z^{n_k}, x^{n_k} - x^* \rangle \\ &= \langle -\nabla f(x^{n_k}), x^{n_k} - x^* \rangle \\ &= D_f(x^*, x^{n_k}) - f(x^*) + f(x^{n_k}).\end{aligned}$$

Next, by part (iii) of the definition of a Bregman function,

$$\lim_{k \to \infty} D_f(x^*, x^{n_k}) = 0.$$

Finally,

$$\lim_{k \to \infty} L(x^k, z^k) = \lim_{k \to \infty} f(x^{n_k}) + \langle z^{n_k}, Ax^{n_k} - b \rangle = f(x^*).$$

We also know that $L(x^k, z^k) \leq f(y)$ for any feasible y. Thus, $f(x^*) \leq f(y)$. Hence x^* is an optimal solution. Now since f is strictly convex, this optimal point is unique. Therefore, we have that $(x^n)_{n \in \mathbb{N}}$ has only one accumulation point and $x^n \to x^*$. $\square$

An important fact consequence of this proof is the following proposition:

Proposition 4.17. *If a_i is an inactive constraint, then there exists a N, such that for all $n \geq N$, we have that $z_i^n = 0$. That is, after some finite time, we never project onto inactive constraints ever again.*

4.8.2 Proof of part 1 of Theorem 4.16 for oracles that satisfy property 4.2

In this subsection, we prove part 1 of Theorem 4.16 for oracles that satisfy property 4.2. We make note of the key ideas in this proof as they are useful in the proof of the truly stochastic variant. To be precise, we prove:

Theorem 4.16. *(Part 1) If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, $\exists\, x^0 \in S$, such that $\nabla f(x^0) = 0$, and the oracle $\mathcal{Q}$ satisfies property 4.2, then with probability 1, any sequence x^n produced by Algorithm 7 converges to optimal solution of problem 4.2.*

Proof. Assume that we have an oracle that satisfies property 4.2. A careful reading of the previous proof shows that if we switch out an oracle with property 4.1 for an oracle with property 4.2, then we only need to adjust step 3 of our proof. The crucial part of that step was showing that for our augmented sequence, we had infinitely many terms that satisfied a constraint *not* in $\tilde{A}$. We make the following adjustments to our analysis.

Let ν_k be the iteration in which the n_kth projection takes place. In the previous proof, we used the property of the oracle only when the $n_k + T$th projection took place in the $\nu_k + 2$nd iteration or a later iteration. In this case, the augmented sequence encompasses all of the $\nu_k + 1$st iteration infinitely often.

Let us choose a constraint $\hat{a}$ that is not satisfied by x^*. Because the oracle satisfies property 4.2, for each iteration $\nu_k + 1$, our oracle returns $\hat{a}$ with probability at least $\tau > 0$. By the Borel Cantelli Lemma, we know that during the selected iterations, the constraint $\hat{a}$ is, with probability one, is returned infinitely often by our oracle. Thus, our augmented sequence satisfies this constraint with probability 1 and x^* lies in the feasible region with probability 1. □

A direct consequence of this proof is the proof of the probabilistic version of

Proposition 4.17.

Proposition 4.17. *With probability* 1*, we project onto inactive constraints a finite number of times.*

4.8.3 Proof of part 2 of Theorem 4.16

The discussions in *Iusem and De Pierro* (1990); *Iusem* (1991) almost directly apply to that for our algorithm. For completeness, we present it along with the necessary modifications. As with the traditional Bregman algorithm, we first present the case when $f(x)$ is quadratic. That is,

$$f(x) = r + s^T \cdot x + \frac{1}{2} x^T H x,$$

where H is a positive definite matrix. In this case, it is easy to see that

$$D_f(x, y) = \|x - y\|_H^2 := (x - y)^T H (x - y).$$

4.8.3.1 Proof of part 2 of Theorem 4.16—Quadratic Case

In this section, we will prove the following variation of Theorem 4.16.

Theorem 4.16. *If f is a strictly convex quadratic function, H_i are strongly zone consistent with respect to f, $x^0 = H^{-1}s \in S$, and the oracle $\mathcal{Q}$ satisfies either property 4.1 or 4.2, then there exists $\rho \in (0, 1)$ such that*

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho, \tag{4.6}$$

where $\|y\|_H^2 = y^T H y$. In the case when we have an oracle that satisfies property 4.2, the limit in 4.6 holds with probability 1*.*

We establish some notation ahead of our lemmas. Let I be the set of all active

constraints. That is, if x^* is the optimal solution then

$$I = \{i : \langle a_i, x^* \rangle = b_i\}.$$

Let S be the set of all x that satisfy these constraints (namely $S = \{x : \forall i \in I, \langle a_i, x \rangle = b_i\}$). Let H_x be the hyper-plane, such that H_x represents the constraint in I that is furthest from x. Define

$$\mu = \inf_{x \notin S} \frac{d(x, H_x)}{d(x, S)}.$$

By *Iusem and De Pierro* (1990), we know that $\mu > 0$. Let U be the set of all optimal dual variables z; i.e., $U = \{z : \nabla f(x^*) = -A^T z\}$ and let $I_\nu = \{i : z_i^{\nu+1} \neq 0\}$.

Next, we present a few preliminary lemmas. These lemmas exist in some form or another in *Iusem* (1991); *Iusem and De Pierro* (1990) and we present them suitably modified for our purpose. These lemmas require the following set of assumptions about an iteration ν:

1. $\forall i \notin I$, $z_i^\nu = 0$;
2. for all $i \notin I$, we do not project onto this constraint in the νth iteration; and,
3. there exists $z \in U$, such that for all $i \notin I_\nu$, $z_i = 0$.

Lemma 4.25. Let x^* be the optimal solution for an instance of problem 4.2. For any sequence $x^n \to x^*$ such that x^n, z^n maintain the KKT conditions, there exists an M, such that for all $\nu \geq M$, there exists a $z \in U$, such that for all $i \notin I_\nu$, we have that $z_i = 0$.

Proof. Let $V_\nu = \{z : \forall i \notin I_\nu, z_i = 0\}$. Then assume, for the sake of contradiction, that the result is false. That is, there is a sequence ν_k such that $V_{\nu_k} \cap U = \emptyset$. Then since there finitely many different I_ν (hence finitely many V_ν), we have that one of these

must occur infinitely often. Thus, by taking an appropriate sub-sequence, we assume, without loss of generality, that I_{ν_k} are all equal. Let $V = V_{\nu_k}$ and obtain $V \cap U = \emptyset$.

Since V is a closed subspace, U is a closed set, and $V \cap U = \emptyset$, we must have that $d(V, U) > 0$. But $z^{\nu_k+1} \in V$ and so

$$d(z^{\nu_k+C}, U) \geq d(V, U) > 0.$$

Since $x^\nu \to x^*$ and we maintain the KKT conditions, we have that for any $z \in U$,

$$A^T z^\nu = -\nabla f(x^\nu) \to -\nabla f(x^*) = A^T z.$$

Thus $d(z^\nu, U) \to 0$ which is a contradiction. □

Lemma 4.26. For any sequence $x^n \to x^*$, if for a given ν, we have that the sequence satisfies assumptions (1) and (2), then

$$\|x^{\nu+1} - x^*\|_Q^2 \leq \|x^\nu - x^*\|_Q^2 - \sum_{n=k}^{K} \|x^{n+1} - x^n\|_Q^2$$

where k and K are the indices of the first and last projection that take place in the νth iteration.

Proof. This Lemma is simply a statement about Bregman projections and so its proof requires no modification. □

Before we proceed, we introduce additional notation. Let A_{I_ν}, b_{i_ν} be the sub-matrix of A, b with rows from I_ν and

$$S_\nu = \{x : A_{I_{\nu_k}} x = b_{I_{\nu_k}}\}.$$

Lemma 4.27. For any sequence such that $x^n \to x^*$, if for a given ν, we have that it satisfies assumptions (1), (2), and (3), then we have that $\|x^{\nu+1} - x^*\|_Q = d(x^{\nu+1}, S_\nu)$.

Proof. Consider the constrained problem

$$\min_{x \in S_\nu} \|x^{\nu+1} - x\|_Q^2. \tag{4.19}$$

Then sufficient conditions for a pair (x, z_{I_ν}) to be optimal for this problem are

$$A_{I_\nu} x = b_{I_\nu} \text{ and } x = x^{\nu+1} - Q^{-1} A_{I_\nu}^T z_{I_\nu}.$$

By Proposition 4.17, we see that since x^* is solution to problem 4.2, we have that $A_{I_\nu} x^* = b_{I_\nu}$. Then by assumptions and the manner in which we do projections, we have that there exists $z \in U$, such that for all $i \notin I_\nu$, $z_i = 0$ and

$$x^* = x^{\nu+1} - Q^{-1} A^T (z^{\nu+1} - z).$$

Then since $z_i^{\nu+1} = 0$ for all $i \notin I_\nu$, we have that

$$x^* = x^{\nu+1} - Q^{-1} A_{I_\nu}^T (z_{I_\nu}^{\nu+1} - z_{I_\nu}).$$

Thus, x^* is the optimal solution to 4.19. □

Next for $x \notin S_\nu$, let H_x^ν be the hyper-plane of that is furthest from x and define

$$\mu_\nu = \inf_{x \notin S_\nu} \frac{d(x, H_x^\nu)}{d(x, S_\nu)}.$$

Now we are ready to prove the following theorem.

Theorem 4.28. Let x^* is the optimal solution to problem 4.2. Then given ν that satisfies assumptions (1), (2), and (3), we have that

$$\|x^{\nu+1} - x^*\|_Q^2 \leq \frac{L}{L + \mu^2} \|x^\nu - x^*\|_Q^2$$

where L is the number of projections that happened in νth iteration.

Proof. By Lemma 4.27, for any such ν we have that $x^{\nu+1} \notin S_\nu$ (or we have converged already). Suppose constraint $j \in I_\nu$ defines the hyper-plane $H^\nu_{x^{\nu+1}}$. Then by Lemma 4.27 and definitions of μ_ν, μ we have the following inequality.

$$\begin{aligned}\|x^{\nu+1} - x^*\| &= d(x^{\nu+1}, S_\nu) \\ &\leq \frac{1}{\mu_\nu} d(x^{\nu+1}, H^\nu_{x^{\nu+1}}) \\ &\leq \frac{1}{\mu} d(x^{\nu+1}, H^\nu_{x^{\nu+1}}).\end{aligned}$$

Now since $I_\nu = \{i : z_i^{\nu+1} \neq 0\}$, we know that during the νth iteration we must have projected onto $H^\nu_{x^{\nu+1}}$. Note that this is the only place in the proof where we need the fact that we remember old constraints. Let us say that this happens during the rth projection of the νth iteration.

Note by assumption, we satisfy the assumptions of Lemma 4.26. Let $y^r, y^{\nu+1}$ be the projections of $x^r, x^{\nu+1}$ onto $H^\nu_{x^{\nu+1}}$. Then we see that

$$
\begin{aligned}
d(x^{\nu+1}, H^{\nu}_{x^{\nu+1}})^2 &= \|y^{\nu+1} - x^{\nu+1}\|_Q^2 \\
&\leq \|y^r - x^{\nu+1}\|_Q^2 \\
&\leq \left(\|y^r - x^{r+1}\|_Q + \sum_{i=r+1}^{L} \|x^i - x^{i+1}\|_Q \right)^2 \\
&= \left(\sum_{i=r+1}^{L} \|x^i - x^{i+1}\|_Q \right)^2 \\
&\leq \left(\sum_{i=0}^{L} \|x^i - x^{i+1}\|_Q \right)^2 \\
&\leq L \sum_{i=0}^{L} \|x^i - x^{i+1}\|_Q^2 \\
&\leq L \left(\|x^{\nu} - x^*\|_Q^2 - \|x^{\nu+1} - x^*\|_Q^2 \right).
\end{aligned}
$$

Thus, we get that

$$
\begin{aligned}
\mu^2 \|x^{\nu+1} x^*\|^2 &\leq d(x^{\nu+1}, H^{\nu}_{x^{\nu+1}})^2 \\
&\leq L \left(\|x^{\nu} - x^*\|^2 - \|x^{\nu+1} - x^*\|_Q^2 \right).
\end{aligned}
$$

Rearranging, we get that

$$
\|x^{\nu+1} - x^*\|_Q^2 \leq \frac{L}{L + \mu^2} \|x^{\nu} - x^*\|_Q^2.
$$

□

As a corollary to the above theorem, we have that algorithm 1 converges linearly.

Theorem 4.16. *2) If f is a strictly convex quadratic function, H_i are strongly zone consistent with respect to f, $x^0 = H^{-1}s \in S$, and the oracle $\mathcal{Q}$ satisfies either property*

4.1 or 4.2, then there exists $\rho \in (0,1)$ *such that*

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho \tag{4.6}$$

where $\|y\|_H^2 = y^T H y$*. In the case when we have an oracle that satisfies property 4.2, the limit in 4.6 holds with probability* 1*.*

Proof. Using Proposition 4.17, Lemma 4.25, and that we have finitely many constraints, we see that if ν is large enough, the assumptions for Theorem 4.28 are satisfied. Taking the limit gives us the needed result.

In the case when we have an oracle that satisfies property 4.2, consider the product space of all possible sequences of hyper-planes returned by our oracle. In this product space, we see that with probability 1, we generate a sequence of hyper-planes, such that algorithm 7 converges. For any such sequence of hyper-planes, we have that 4.6 holds. Thus, the limit in 4.6 holds with probability 1 for random separation oracles. □

4.8.3.2 Proof of part 2 of Theorem 4.16—General

The rate of convergence for the general Bregman method was established in *Iusem* (1991). To show this, let $\tilde{f}$ be the 2nd degree Taylor polynomial of f centered at the optimal solution x^*.

$$\tilde{f}(x) = f(x^*) + \nabla f(x^*)^T \cdot x + \frac{1}{2} x^T \cdot \nabla^2 f(x^*) \cdot x.$$

For notational convenience, let H be the Hessian of f at x^*. Then we can see that if replace f with $\tilde{f}$ in 4.2 then the optimal solution does not change. Thus, if had access to $\tilde{f}$ and could use this function to do our projections, then from the quadratic case we have our result.

Thus, to get the general result, if x^ν is our standard iterate and $\tilde{x}^\nu$ is the iterate produced by using $\tilde{f}$ instead of f, then *Iusem* (1991) shows that $\|x^\nu - \tilde{x}^\nu\|$ is $o(\|x^\nu -$

$x^*\|_H$). Specifically, we can extract the following theorem from *Iusem* (1991).

Theorem 4.29. *Iusem* (1991) Let x^* is the optimal solution for problem 4.2 and $\tilde{x}^n$ is the sequence produced by using the same sequence of hyper-planes but with $\tilde{f}$ instead of f. Given a sequence x^n produced by Bregman projections, such that $x^n \to x^*$, and for large enough ν we satisfy assumptions (1), (2), and (3), then $\|x^\nu - \tilde{x}^\nu\|$ is $o(\|x^\nu - x^*\|_H)$

Using this we can get the general result as follows

$$\begin{aligned}
\|x^{\nu+1} - x^*\|_H &\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \|\tilde{x}^{\nu+1} - x^*\|_H \\
&\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \rho\|\tilde{x}^{\nu} - x^*\|_H \\
&\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \rho\|\tilde{x}^{\nu} - x^{\nu}\|_H \\
&\quad + \rho\|x^{\nu} - x^*\|_H.
\end{aligned}$$

Then diving by $\|x^{\nu+1} - x^*\|_H$, and using Theorem 4.29 to take the limit, we get that there exists $\rho \in (0, 1)$ such that

$$\lim_{\nu\to\infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho. \tag{4.6}$$

As with the quadratic case, we see that is an oracle satisfies property 4.2, then 4.6 holds with probability 1. Thus, we have proved Theorem 4.16 in its complete generality.

4.8.4 Proof of Theorem 4.20

In this section we prove Theorem 4.20 in essentially the same manner as we did for Theorem 4.16 and so we outline only what changes are necessary.

Theorem 4.20. *If $f \in \mathcal{B}(S)$, the hyper-planes H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then with probability 1 any*

sequence x^n produced by the algorithm converges to the optimal solution of problem 4.2. Furthermore, if x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semi-definite, then there exists $\rho \in (0,1)$ such that with probability 1,

$$\liminf_{\nu\to\infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho. \tag{4.7}$$

To prove Theorem 4.20, we need to analyze only what goes wrong if the algorithm "forgets" all of the old constraints. First, consider the proof in the case that we converge to the optimal solution. Then, steps 1,2, and 3 are completely unaffected by forgetting old constraints. The only step that is affected is step 4. In a previous proof, we argued that if for some inactive constraint a_i, z_i is non-zero infinitely often, then we projected onto this constraint infinitely often. In our present setting, we cannot conclude this directly as $z_i^\nu > 0$ does not imply that we remember a_i on the νth iteration. However, due to property 4.2, we know that $\mathcal{Q}$ returns a_i with probability at least τ. Thus, again using the Borel Cantelli Lemmas, we see that we have a_i infinitely often and this iteration converges to the optimal.

To prove the second part of the theorem, we recall from Theorem 4.29 that we only need to analyze the case when f is a quadratic function. Indeed, the only place where we used the fact that we remembered old constraints was in the proof of Theorem 4.28 in which we needed to remember old constraints to guarantee that during the νth iteration we project onto the constraint a_i that is furthest from x^ν among those constraints for which $z_i^{\nu+1} > 0$. We cannot guarantee that this happens always but we can guarantee that it happens infinitely often.

Therefore, the conclusion of Theorem 4.28 holds infinitely often instead of for the tail of the sequence and we replace the limit with a limit infimum to obtain the desired result.

4.8.5 General Convex Proof

We will need the following facts from *Bauschke and Lewis* (2000b).

1. A convex function if Legendre if and only if its convex conjugate is Legendre. In this case the gradient mapping

$$\nabla f : int(dom(f)) \to int(domf^*) : x \mapsto \nabla f(x)$$

is a topological isomorphism with inverse mapping $(\nabla f)^{-1} = \nabla f^*$.

2. Suppose f is Legendre on E and a S is a closed convex set in E with $S \cap int(dom(f)) \neq \emptyset$. Suppose further $y \in int(dom(f))$. Then the Bregman projection $P_S^f y$ of y onto S with respect to f is characterized by

$$P_S^f y \in S \cap int(dom(f)) \text{ and } \langle \nabla f(y) - \nabla f(P_S^f y), S - P_S^f y \rangle \leq 0.$$

In addition, $D_f(P_S^f y, y) \leq D_f(s, y) - D_f(s, P_S^f y)$ for all $s \in S \cap dom(f)$. Here when we write $\langle \nabla f(y) - \nabla f(P_S^f y), S - P_S^f y \rangle \leq 0$, we mean that $\langle \nabla f(y) - \nabla f(P_S^f y), s - P_S^f y \rangle \leq 0$, is true for all $s \in S$.

3. Three point identity: Suppose f is Legendre on E. If $x, y \in int(domf)$ and $b \in domf$, then

$$D_f(b, x) + D_f(x, y) - D_f(b, y) = \langle \nabla f(x) - \nabla f(y, x - b) \rangle.$$

4. Suppose f is a very strictly convex function on E. Then for every compact convex subset K of $int(domf)$, there exists reals $0 < \theta$ and $\Theta < +\infty$ such that for every $x, y \in K$ we have that

$$D_f(x, y) \geq \theta \|x - y\|^2 \text{ and } \|\nabla f(x) - \nabla f(y)\| \leq \Theta \|x - y\|.$$

On the nth iteration we projection onto the constraint given by $i(n)$. Thus, we have that

$$x^n \in C_{i(n)} \cap dom(f). \tag{4.20}$$

Then since f is Legendre, using Facts 1 and 2 with $y = \nabla f^*(\nabla f(x^{n-1}) + q^{p(n)})$, $S = C_{i(n)}$ and $P_S^f y = x^n$, we have that

$$\langle \nabla f(x^{n-1}) + q^{p(n)} - \nabla f(x^n), C_{i(n)} - x^n \rangle \leq 0.$$

Then by definition of q^n we have that

$$\langle q^n, x^n - C_{i(n)} \rangle \geq 0. \tag{4.21}$$

Then we can also see that

$$\nabla f(x^{n-1}) - \nabla f(x^n) = q^n - q^{p(n)}. \tag{4.22}$$

Then summing over this telescoping sum and the fact that the q are initialized to be 0 we get that

$$\nabla f(x^0) - \nabla f(x^n) = \sum_{k \in \mathcal{C}} q^{p(k,n)}. \tag{4.23}$$

Here $\mathcal{C}$ stores the indices of all the constraints. To see why this is true, consider what happens if we sum the terms for just one constraint set. If we see the right hand side of Equation 4.22, then we see that only the q from the last time we projected onto that set remains.

Let $c \in int(domf)$. Then we need to prove the following crucial equality.

$$
\begin{aligned}
D_f(c, x^n) &= D_f(c, x^{n+1}) + D_f(x^{n+1}, x^n) - \langle \nabla f(x^{n+1}) - \nabla f(x^n), x^{n+1} - c \rangle && \text{[Fact 3]} \\
&= D_f(c, x^{n+1}) + D_f(x^{n+1}, x^n) + \langle q^{(n+1)} - q^{p(n+1)}, x^{n+1} - c \rangle && \text{[Eq 4.22]} \\
&= D_f(c, x^{n+1}) + D_f(x^{n+1}, x^n) + \langle q^{n+1}, x^{n+1} - c \rangle - \langle q^{p(n+1)}, x^{n+1} - c \rangle \\
&= D_f(c, x^{n+1}) + D_f(x^{n+1}, x^n) + \langle q^{n+1}, x^{n+1} - c \rangle \\
&\qquad - \langle q^{p(n+1)}, x^{n+1} + x^{p(n+1)} - x^{p(n+1)} - c \rangle \\
&= D_f(c, x^{n+1}) + D_f(x^{n+1}, x^n) + \langle q^{n+1}, x^{n+1} - c \rangle \\
&\qquad - \langle q^{p(n+1)}, x^{n+1} - x^{p(n+1)} \rangle + \langle q^{p(n+1)}, x^{p(n+1)} - c \rangle.
\end{aligned}
$$

Using induction we can now prove that for $c \in int(domf)$ and for every $n \geq 0$ we have that

$$
D_f(c, x^0) = D_f(c, x^n) + \sum_{k=1}^{n} \left(D_f(x^k, x^{k-1}) + \langle q^{p(k)}, x^{p(k)} - x^k \rangle \right) + \sum_{k \in \mathcal{C}} \langle x^{p(k,n)} - c, q^{p(k,n)} \rangle.
$$

Using Equation 4.21, we have that all the inner products on the right hand of the above equation are non-negative. Since the Bregman distances are always non-negative, every term on the right hand side in non-negative. Thus, if we take the limit as $n \to \infty$ of the equation we get that $D_f(c, x^n)$ is a convergence sequence. Thus, is bounded. We also get that $S_n = \sum_{k=1}^{n} D_f(x^k, x^{k-1})$ is also a convergent sequence and hence is bounded. Thus, we get that

$$
\lim_{n \to \infty} D_f(x^n, x^{n+1}) = 0.
$$

Then using the previous reasoning we see that x^n is a bounded sequence and has accumulation points. Furthermore by *Bauschke and Lewis* (2000b) we have that all the accumulation points lie inside $domf$ and we have that $x^k - x^{k-1} \to 0$.

Now using Equation 4.23 and adding and subtracting $x^{p(k,n)}$ from $c - x^n$, we get that

$$\langle c-x^n, \nabla f(x^0)-\nabla f(x^n)\rangle = \underbrace{\sum_{k\in\mathcal{C}}\langle c - x^{p(k,n)}, q^{p(k,n)}\rangle}_{S_1(n)} + \underbrace{\sum_{k\in\mathcal{C}}\langle x^{p(k,n)} - x^n, q^{p(k,n)}\rangle}_{S_2(n)}. \quad (4.24)$$

Using Equation 4.21, we see that $S_1(n)$ is non positive. Now we want to prove that

$$\lim_n \sum_{k\in\mathcal{C}} |\langle x^{p(k,n)} - x^n, q^{p(k,n)}\rangle| \stackrel{?}{=} 0. \quad (4.25)$$

Let $K := cl(conv(\{x^n\}))$. That is, K is the smallest closed convex set containing the sequence $\{x^n\}$. Then we have that

$$K = conv(cl(x^n)) = conv(\{x^n\} \cup \{ \text{ cluster points of } x^n\}) \subseteq conv(int(domf)) = int(domf).$$

The first equality comes from the fact that the cluster points of x^n are exactly the points needed to make the set $\{x^n\}$ closed. The second comes from the fact that the clusters points x^n are in the interior of the domain of f. As we previously showed. Finally, since we assumed that the domain of f is convex, we get the last equality.

Then, since f is very strictly convex and K is compact (we showed that the sequence $\{x^n\}$ is bounded), we can use Fact 4 to see that there exists $0 < \theta$ and $\Theta < +\infty$, such that

$$D_f(x^k, x^{k-1}) \geq \theta\|x^k - x^{k-1}\|^2 \text{ and } \|\nabla f(x^k) - \nabla f(x^{k-1})\| \leq \Theta\|x^k - x^{k-1}\|.$$

Then using the fact that $\sum_{k=1}^{\infty} D_f(x^k, x^{k-1}) < \infty$ we have that

$$\sum_{k=1}^{\infty} \|x^k - x^{k-1}\|^2 \leq \frac{1}{\theta}\sum_{k=1}^{\infty} D_f(x^k, x^{k-1}) < \infty.$$

Now consider the following telescoping identity

$$q^n = (q^n - q^{p(n)}) - (q^{p(n)} - q^{p(p(n))}) - \ldots - (q^{p^k(n)} - q^{p^{k+1}(n)}) - \ldots 0.$$

Then, we get that

$$\sum_{k\in\mathcal{C}} \|q^{p(k,n+1)}\| \leq \sum_{k=1}^{n} \|q^k - q^{p(k)}\| = \sum_{k=1}^{n} \|\nabla f(x^k) - \nabla f(x^{k-1})\| \leq \Theta \sum_{k=1}^{n} \|x^k - x^{k-1}\|.$$

Now note that we only forget a constraint if $q^k = 0$ and that in each iteration we look at most $2M$ constraints (where M is number of constraints). Thus, if $q^{p(k,n)} \neq 0$ then, we must have that $p(k,n) \geq n - 4M$. That is, we either have projected onto that constraint in the current iteration or if we haven't projected onto that constraint yet in the current iteration, we must have projected onto it in the previous iteration. Thus, using the above we get that

$$\begin{aligned}
\sum_{k\in\mathcal{C}} |\langle x^{p(k,n)} - x^n, q^{p(k,n)}\rangle &\leq \sum_{k\in\mathcal{C}} \|x^{p(k,n)} - x^n\| \|q^{p(k,n)}\| && (4.26)\\
&\leq \sum_{k\in\mathcal{C}} \|q^{p(k,n+1)}\| \sum_{k=n-8M}^{n} \|x^k - x^{k-1}\| && (4.27)\\
&\leq \Theta \sum_{k=1}^{n} \|x^k - x^{k-1}\| \sum_{k=n-4M}^{n} \|x^k - x^{k-1}\|. && (4.28)
\end{aligned}$$

Then using proposition 3.1 from *Bauschke and Lewis* (2000b) with the fact that $\sum_{k=1}^{\infty} \|x^k - x^{k-1}\|^2 < \infty$ we get that the limit of the right hand side is 0. Thus, Equation 4.25 is true. Now looking at Equation 4.25 again we see that there exists a subsequence k_n such that

$$\sum_{k\in\mathcal{C}} |\langle x^{p(k,k_n)} - x^{k_n}, q^{p(k,k_n)}\rangle| \to 0 \text{ and } 0 \geq \limsup_n \langle c - x^{k_n}, \nabla f(x^0) - \nabla f(x^{k_n})\rangle.$$

Let x^* be the accumulation point for this convergent subsequence (we may need to pass to another subsequence). Now we want to show that x^* is feasible.

Assume for the sake of contradiction that x^* is not feasible. Let $\tilde{C}$ be the set of constraint sets that x^* belongs to, i.e., the set of constraints x^* satisfies. Recall that C is the set of all constraints, and let

$$\epsilon = \phi\left(\inf_{\hat{C}\in C\cap\tilde{C}^c} \operatorname{dist}(x^*, \hat{C})\right).$$

Let N be such that for all $k \geq N$ we have that $\|x^{n_k} - x^*\| < \epsilon/2$. Then by Equation 6.48 from *Censor and Zenios* (1997), since x^n is bounded, $x^{n_k} \to x^*$, and $D(x^{n+1}, x^n) \to 0$ we have that for any t,

$$x^{n_k+t} \to x^*.$$

Thus, in particular, we see that for all $t \leq 2|\tilde{C}| + 2M + 1 =: T$ this is true.

Consider our augmented subsequence $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$ (note if $n_{k+1} - n_k \to \infty$ then this is not all points in the sequence). We want to show that infinitely often one of these values satisfies a constraint not in $\tilde{C}$. Then since we have finitely many constraints, at least one constraint $\tilde{c}$ not in $\tilde{C}$ must be satisfied infinitely often by our augmented sequence. Then since our augmented sequence converges to x^* and we are only looking at closed constraints, we must have x^* satisfies this constraint. Which is a contradiction. Thus, x^* must be feasible.

Let $k > N$. Then we know that x^{n_k} is the variable that we get after the n_kth projection. Let's assume that this happens in iteration ν_k. Note that in any iteration we project onto any constraint at most twice, once if its in our list C^{ν_k} and once if its in the list L returned by the oracle. Thus we have two possibilities for which iteration the $n_k + 2|\tilde{C}| + 1$ projection takes place. Let us case on when that happens.

Case 1: If $n_k + 2|\tilde{C}| + 1$ takes places in ν_kth iteration infinitely often, then in case since we project onto each constraint at most twice, by pigeonhole principle one of the

projections between n_k and $n_k + 2|\tilde{C}| + 1$ must be onto a constraint not represented in $\tilde{C}$. Thus, if we look amongst $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+2|\tilde{C}|+1}$, we must have projected onto a constraint not in $\tilde{C}$ infinitely often.

Case 2: If $n_k + 2|\tilde{C}| + 1$ takes places in $\nu_k + 1$th iteration infinitely often. Then since our oracle returns at most M constraints, we must have that amongst $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$ we must have projected onto all of the constraints returned by the oracle during step 1 of the $\nu_k + 1$th iteration. Thus, we must have projected onto some constraint $\hat{c}^{\nu_k+1}$ such that distance of $x^{\nu+1}$ to $\hat{c}^{\nu_k+1}$ is at least $\phi(d_{\nu_k+1})$ where d_{ν_k+1} is the distance from x^{ν_k+1} to our feasible region C. Then, we have that

$$\operatorname{dist}(x^{\nu_k+1}, \hat{c}^{\nu_k+1}) \geq \phi(d_{\nu_k+1}) \geq \phi\left(\inf_{\hat{C} \in C \cap \tilde{C}^c} \operatorname{dist}(x^{\nu_k+1}, \hat{C})\right).$$

Noting that $x^{\nu+1}$ is within our augmented sequence and taking the limit, we get that

$$\lim_{k\to\infty} \operatorname{dist}(x^{\nu_k+1}, \hat{c}^{\nu_k+1}) \geq \epsilon.$$

Since, we have finitely many constraints, we may assume (by passing to a subsequence) that all $\hat{c}^{\nu_k+1}$ are equal to some constraint $\hat{c}$. Then, we have that

$$\operatorname{dist}(x^*, \hat{c}) \geq \epsilon.$$

Thus, $\hat{c} \notin \tilde{C}$. However, we now project onto $\hat{c}$ infinitely often. Hence x^* must satisfy the constraint $\hat{c}$. Hence we have a contradiction. Thus, x^* is feasible.

Finally, we have that since $D_f(\cdot, \cdot)$ is separately continuous, we have that $D_f(c, x^{k_n}) \to D_f(c, x^*)$ and $D_f(x^{k_n}, x^0) \to D_f(x^*, x^0)$. Thus for an arbitrary feasible c we have that

$$\begin{aligned}\langle c - x^*, \nabla f(x^0) - \nabla f(x^*)\rangle &= D_f(c, x^*) + D_f(x^*, x^0) - D_f(c, x^0) \\ &= \lim_n D_f(c, x^{k_n}) + D_f(x^{k_n}, x^0) - D_f(c, x^0) \\ &= \lim_n \langle c - x^{k_n}, \nabla f(x^0) - \nabla f(x^{k_n})\rangle \\ &\leq 0.\end{aligned}$$

Then by Fact 3, we have that $x^* = P_C^f(x^0)$. Thus, all accumulation points are optimal points. Thus, by strict convexity of f, we see that this is the unique solution.

Now to get the unique solution to constrained minimization problem we need to initialize x^0 such that $\nabla f(x^0) = 0$. This is because x^* is the point that minimized

$$D_f(x, x^0) = f(x) - f(x^0) - \langle \nabla f(x^0), x - x^0\rangle.$$

Thus, if the last term is 0, then second term is a constant and we minimize the first term which is what we want.

4.8.6 Convergence Rate for Quadratic Objective Function

Lemma 4.30. For any C_i such that $x^* \in int(C_i)$ we have that we only look at this constraint finitely often.

Proof. Every time we look at a constraint we project onto its boundary. Thus, if we project onto C_i infinitely often x^* cannot be in the interior of C_i. □

Let I be the set of all active constraints. That is if x^* is the optimal solution then

$$I = \{i : x^* \in \partial C_i\}$$

Let S be the set of all x that satisfy these constraints (namely $S = \{x : \forall i \in I, x \in$

$C_i\}$). Let B_x be the boundary for a constraint in I furthest from x and define

$$\mu = \inf_{x \notin S} \frac{d(x, B_x)}{d(x, S)} > 0$$

Let U be the set of all optimal duals $q's$ that is $U = \{(q_1, \ldots, q_N) : \nabla f(x^0) - \nabla f(x^*) = \sum_i q_i\}$ and let $I_\nu = \{i : q^{p(i,\nu+1)} \neq 0\}$.

Lemma 4.31. For any sequence $x^n \to x^*$ where x^* is the optimal for an instance of problem 4.3, and x^n, q^n maintain the KKT conditions then there exists an M' such that for all $\nu \geq M$" there exists a $z \in U$ such that for all $i \notin I_\nu$ we have that $q_i = 0$

Proof. Let $V_\nu = \{(q_1, \ldots, q - N) : \forall i \notin I_\nu, q_i = 0\}$. Then assume for the sake of contradiction that the result is false. Thus, there is a sequence ν_k such that $V_{\nu_k} \cap U = \emptyset$. Then since there finitely many different I_ν (hence finitely many V_ν) we have that one of these must occur infinitely often. Thus, by taking an appropriate subsequence we assume that without loss of generality that all I_{ν_k} are all equal. Thus, let $V = V_{\nu_k}$. Thus, we have that $V \cap U = \emptyset$.

Then since V is a subspace (hence closed), U is a closed set, and they are disjoint, we must have that $d(V, U) > 0$. But now $(q^{p(1,\nu_k+1)}, \ldots, q^{p(N,\nu_k+1)}) \in V$. Thus,

$$d((q^{p(1,\nu_k+1)}, \ldots, q^{p(N,\nu_k+1)}), U) \geq d(V, U) > 0$$

By Theorem 4.21 we have that $x^\nu \to x^*$. Thus, for any $(q_1, \ldots, q_N) \in U$,

$$\sum_i q^{p(i,\nu_k+1)} = \nabla f(x^0) - \nabla f(x^\nu) \to \nabla f(x^0) - \nabla f(x^*) = \sum_i q_i$$

Thus $d((q^{p(1,\nu_k+1)}, \ldots, q^{p(N,\nu_k+1)}), U) \to 0$ which is a contradiction □

The rest of the proof is same as for Theorem 4.16. As before we get the following corollaries.

Corollary 4.32. If we further have that there exists an $x^0 \in domf$ such that $\nabla f(x^0) = 0$. Then we have that the Bregman projection of x^0 onto C is the solution to Problem 4.3.

Remark 4.33. Similar to before while this gives linear convergence but ρ is very close to one. As we will see later $\rho \leq \frac{F}{F+1}$ where F is number of convex sets whose boundary the optimal solution lies on that are seen by the algorithm.

Remark 4.34. Unlike in the linear case, our proof does not guarantee that a truly stochastic version will converge to the optimal solution. Equation 4.27 does not hold in this case.

CHAPTER V

Tree! I am no Tree! I am a Low Dimensional Hyperbolic Embedding

5.1 Introduction

Extracting hierarchical information from data is a key step in understanding and analyzing the structure of the data in a wide range of areas from the analysis of single cell genomic data *Klimovskaia et al.* (2019), to linguistics *Dhingra et al.* (2018), computer vision *Khrulkov et al.* (2019) and social network analysis *Verbeek and Suri* (2016). In single cell genomics, for example, researchers want to understand the developmental trajectory of cellular differentiation. To do so, they seek techniques to visualize, to cluster, and to infer temporal properties of the developmental trajectory of individual cells.

One way to capture the hierarchical structure is to represent the data as a tree. Even simple trees, however, cannot be faithfully represented in low dimensional Euclidean space *Linial et al.* (1995a). As a result, a variety of remarkably effective hyperbolic representation learning methods, including *Nickel and Kiela* (2017, 2018b); *Sala et al.* (2018), have been developed. These methods learn an embedding of the data points in hyperbolic space by first solving a non-convex optimization problem and then extracting the hyperbolic metric that corresponds to the distances between the

embedded points. These methods are successful because of the inherent connections between hyperbolic spaces and trees. They do not, however, come with rigorous geometric guarantees about the quality of the solution. Also, they are slow.

In this paper, we present a metric first approach to extracting hierarchical information and learning hyperbolic representations. The important connection between hyperbolic spaces and trees suggests that the correct approach to learning hyperbolic representations is the metric first approach. That is, first, learn a tree that essentially preserves the distances amongst the data points and then embed this tree into hyperbolic space.* More generally, the metric first approach to metric representation learning is to build or to learn an appropriate metric first by constructing a discrete, combinatorial object that corresponds to the distances and then extracting its low dimensional representation rather than the other way around.

The quality of a hyperbolic representation is judged by the quality of the metric obtained. That is, we say that we have a good quality representation if the hyperbolic metric extracted from the hyperbolic representation is, in some way, faithful to the original metric on the data points. We note that finding a tree metric that approximates a metric is an important problem in its own right. Frequently, we would like to solve metric problems such as transportation, communication, and clustering on data sets. However, solving these problems with general metrics can be computationally challenging and we would like to approximate these metrics by simpler, tree metrics. This approach of approximating metrics via simpler metrics has been extensively studied before. Examples include dimensionality reduction *Johnson and Lindenstrauss* (1984) and approximating metrics by simple graph metrics *Bartal* (1998); *Peleg and Ullman* (1989).

To this end, in this paper, we demonstrate that methods that learn a tree structure first outperform methods that learn hyperbolic embeddings directly. Additionally, we

*A similar idea is mentioned in *Sala et al.* (2018) for graph inputs rather than general metrics. They do not, however undertake a detailed exploration of the idea.

have developed a novel, extremely fast algorithm TreeRep that takes as input a δ-hyperbolic metric and learns a tree structure that approximates the original metric. TreeRep is a new method that makes use of geometric insights obtained from the input metric to infer the structure of the tree. To demonstrate the effectiveness of our method, we compare TreeRep against previous methods such as *Abraham et al.* (2007) and *Saitou and Nei* (1987) that also recover tree structures given a metric. There is also significant literature on approximating graphs via (spanning) trees with low stretch or distortion, where the algorithms take as input graphs, *not metrics*, and output trees that are subgraphs of the original. We also compare against such algorithms *Alon et al.* (1995); *Chepoi et al.* (2008); *Elkin et al.* (2005); *Prim* (1957). We show that when we are given only a metric and not a graph, then even if we use a nearest neighbor graph or treat the metric as a complete graph TreeRep is not only faster, but produces better results than *Abraham et al.* (2007); *Alon et al.* (1995); *Bartal* (1998); *Chepoi et al.* (2008); *Prim* (1957) and comparable results to *Saitou and Nei* (1987).

For learning hyperbolic representations, we demonstrate that TreeRep is over 10,000 times faster than the optimization methods from *Nickel and Kiela* (2017, 2018b), and *Sala et al.* (2018) while producing better quality results in most cases. This extreme decrease in time, with no loss in quality, is exciting as it allows us to extract hierarchical information from much larger data sets in single-cell sequencing, linguistics, and social network analysis - data sets for which such analysis was previously unfeasible.

The rest of the paper is organized as follows. Section 5.2 contains the relevant background information. Section 5.3 presents the geometric insights and the TreeRep algorithm. In Section 5.4, we compare TreeRep against the methods from *Abraham et al.* (2007); *Alon et al.* (1995); *Chepoi et al.* (2008); *Prim* (1957) and *Saitou and Nei* (1987) in approximating metrics via tree metrics and against methods from *Nickel and*

Kiela (2017, 2018b) and *Sala et al.* (2018) for learning low dimensional hyperbolic embedding. We show that the methods that learn a good tree to approximate the metric, in general, find better hyperbolic representations than those that embed into the hyperbolic manifold directly.

5.2 Preliminaries

The formal problem that our algorithm will solve is as follows[†].

Problem 5.0.1. Given a metric d find a tree structure T such that the shortest path metric on T approximates d.

Definition 5.1. Given a weighted graph $G = (V, E, W)$ the shortest path metric d_G on V is defined as follows: $\forall u, v \in V$, $d_G(u, v)$ is the length of the shortest path from u to v.

5.2.1 δ-Hyperbolic Metrics.

Gromov introduced the notion of δ-hyperbolic metrics as a generalization of the type of metric obtained from negatively curved manifolds *Gromov* (1987).

Definition 5.2. Given a space (X, d), the Gromov product of $x, y \in X$ with respect to a base point $w \in X$ is

$$(x, y)_w := \frac{1}{2} \left(d(w, x) + d(w, y) - d(x, y) \right).$$

The Gromov product is a measure of how close w is to the geodesic $g(x, y)$ connecting x and y.

[†]Note that the input to our problem are metrics and not graphs. Thus, we handle more general inputs as compared to *Alon et al.* (1995); *Elkin et al.* (2005); *Chepoi et al.* (2008), and *Prim* (1957).

Definition 5.3. A metric d on a space X is a δ-hyperbolic metric on X (for $\delta \geq 0$), if for every $w, x, y, z \in X$ we have that

$$(x, y)_w \geq \min\left((x, z)_w, (y, z)_w\right) - \delta. \tag{5.1}$$

In most cases we care about the smallest δ for which d is δ-hyperbolic.

An example of a δ-hyperbolic space is the hyperbolic manifold $\mathbb{H}^k$ with $\delta = \tanh^{-1}\left(1/\sqrt{2}\right)$ *Carnahan* (2010).

Definition 5.4. The hyperboloid model $\mathbb{H}^k$ of the hyperbolic manifold is $\mathbb{H}^k = \{x \in \mathbb{R}^{k+1} : x_0 > 0, x_0^2 - \sum_{i=1}^{k} x_i^2 = 1\}$.

An important case of hyperbolic metrics is when $\delta = 0$. One important property of such metrics is that they come tree spaces.

Definition 5.5. A metric d is a tree metric if there exists a weighted tree T such that the shortest path metric d_T on T is equal to d.[‡]

Definition 5.6. Given a discrete graph $G = (V, E, W)$ the metric graph (X, d) is the space obtained by letting $X = E \times [0, 1]$ such that for any $(e, t_1), (e, t_2) \in X$ we have that $d((e, t_1), (e, t_2)) = W(e) \cdot |t_1 - t_2|$. This space is called a tree space if G is a tree. Here $E \times \{0, 1\}$ are the nodes of G.

Definition 5.7. Given a metric space, (X, d), two points $x, y \in X$, and a continuous function $f : [0, 1] \to X$, such that $f(0) = x$, $f(1) = y$, and there is a λ such that $d(f(t_1), f(t_2)) = \lambda |t_1 - t_2|$, the geodesic $g(x, y)$ connecting x and y is the set $f([0, 1])$.

Definition 5.8. A metric space T is a tree space (or a $\mathbb{R}$-tree) if any pair of its points can be connected with a unique geodesic segment, and if the union of any two geodesic segments $g(x, y), g(y, z) \subset T$ having the only endpoint y in common, is the geodesic segment $g(x, z) \subset T$.

[‡]Note, such metrics may have representations as graphs that are not trees, Section 5.3 has a simple example.

There are multiple definitions of a tree space. However, they are all connected via their metrics. *Bermudo et al.* (2013) tells us that a metric space is 0-hyperbolic if and only if it is an $\mathbb{R}$-tree or a tree space. This result lets us immediately conclude that Definitions 5.6 and 5.8 are equivalent. Similarly, Definition 5.1 implies that Definition 5.5 and 5.6 are equivalent. Hence all three definitions of tree spaces are equivalent. We note that trees are 0-hyperbolic, and that $\delta = \infty$ corresponds to an arbitrary metric. Thus, δ is a heuristic measure for how close a metric is to a tree metric.

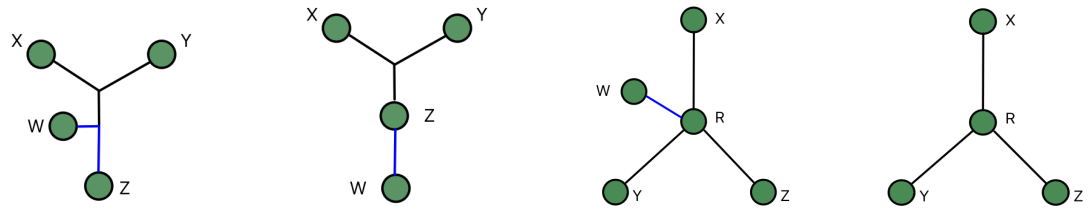
5.2.2 Trees as Hyperbolic Representation.

The problem that is looked at by *Sala et al.* (2018); *Nickel and Kiela* (2017, 2018b) is the problem of learning hyperbolic embeddings. That is, given a metric d, learn an embedding X in some hyperbolic space $\mathbb{H}^k$. We, however, are proposing that if we want to learn a hyperbolic embedding, then we should instead learn a tree. In many cases, we can think of this tree as the hyperbolic representation. However, if we do want coordinates, this can be done as well.

Sala et al. (2018) give an algorithm that is a modification of the algorithm in *Sarkar* (2012) that can, in linear time, embed any weighted tree into $\mathbb{H}^k$ with arbitrarily low distortion (if scaling of the input metric is allowed). The analysis in *Sala et al.* (2018) quantifies the trade-offs amongst the dimension k, the desired distortion, the scaling factor and the number of bits required to represent the distances in $\mathbb{H}^k$. We use these results to consider trees as hyperbolic representations. One possible drawback of this approach is that we may need a large number of bits of precision. Recent work, however, such as *Yu and De Sa* (2019) provides a solution to this issue.

5.3 Tree Representation

To solve Problem 5.0.1 we present an algorithm TreeRep such that Theorem 5.9 holds.



(i) $\hat{T}$ when $\pi x = z$ and $(z,x)_w = (z,y)_w < d(w,z)$.

(ii) $\hat{T}$ when $\pi x = z$ and $(z,x)_w = (z,y)_w = d(w,z)$.

(iii) $\hat{T}$ when $(y,x)_w = (y,z)_w = (x,z)_w \neq 0$

(iv) Universal Tree on x, y, z.

Figure 5.1: Figures showing the tree $\hat{T}$ from Lemma 5.13 for $Zone_2(z)$ (a), $Zone_1(z)$ (b), $Zone_1(r)$ (c), and the Universal tree (d).

Theorem 5.9. Given (X, d), a δ-hyperbolic metric space, and n points $x_1, \ldots, x_n \in X$, TreeRep returns a tree (T, d_T). In the case that $\delta = 0$, $d_T = d$, and T has the fewest possible nodes. TreeRep has a worst case run time $O(n^2)$. Furthermore the algorithm is embarrassingly parallelizable.

Remark 5.10. In practice, we see that the run time for TreeRep is much faster than $O(n^2)$.

To better understand the geometric insights used to develop TreeRep, we first focus on the problem of reconstructing the tree structure from a tree metric. Algorithm 16 and 17 present a high level version of the pseudo-code. The complete pseudo-code for TreeRep is presented in Appendix B.8.

TreeRep is a recursive, divide and conquer algorithm. The first main idea (Lemma 5.11) is that for any metric d on three points, we can construct a tree T with four nodes such that $d_T = d$. We will call such trees *universal trees*. The second main idea (Lemma 5.13) is that adding a fourth point to a universal tree can be done consistently and, more importantly, the additional point falls into one of seven different zones. Thus, TreeRep will first create a universal tree T and then will sort the remaining data points into the seven different zones. We will then do recursion with each of the zones. Finally, we show in Lemma 5.15 that for each node we add in a

zone, the distances to all the other (non-universal) nodes in other zones are consistent, so we simply have to maintain consistency amongst the points within each zone. That is, it's sufficient to maintain local distance consistency.

Lemma 5.11. Given a metric d on three points x, y, z, there exists a (weighted) tree (T, d_T) on four nodes x, y, z, r, such that r is adjacent to x, y, z, the edge weights are given by $d_T(x, r) = (y, z)_x$, $d_T(y, r) = (x, z)_y$ and $d_T(z, r) = (x, y)_z$, and the metric d_T on the tree agrees with d.

Definition 5.12. The tree constructed in Lemma 5.11 is the universal tree on the three points x, y, z. The additional node r is known as a Steiner node.

An example of the universal tree can be seen in Figure 5.1iv. To understand the distinction between the seven different zones, we need to reinterpret Equation 5.1. We know that for any tree metric, and any four points w, x, y, z, we have that

$$(x, y)_w \geq \min\big((x, z)_w, (y, z)_w\big).$$

This inequality implies that the smaller two of the three numbers $(x, y)_w, (x, z)_w$, and $(y, z)_w$ are equal. In this case, knowing which of the quantities are equal tells us the structure of the tree. Specifically, here x, y, z will be the three points in our universal tree T and w will be the point that we want to sort. Then initially, we have four possibilities. The first possibility is that all three Gromov products are equal. This case will define its own zone. If this is not the case, then we have three possibilities depending on which two out of the three Gromov products are equal. Suppose we have that $(x, y)_w = (x, z)_w$, then due to the triangle inequality, we have that $d(w, x) \geq (x, y)_w$. Thus, we will further subdivide this case into two more cases, depending on whether $d(w, x) = (x, y)_w$ or $d(w, x) > (x, y)_w$. Each of these cases will define their own zone. Examples of the different cases can be seen in Figure 5.1. We can also see that there are two different types of zones. The first type is when we

connect the new node directly to an existing node as seen in Figures 5.1ii and 5.1iii. The second type is when we connect w to an edge as seen in Figure 5.1i. The formal definitions for the zones can be seen in Definition 5.14.

Lemma 5.13. Let (X, d) be a tree space. Let w, x, y, z be four points in X and let (T, d_T) be the universal tree on x, y, z with node r as the Steiner node. Then we can extend (T, d_T) to $(\hat{T}, d_{\hat{T}})$ to include w such that $d_{\hat{T}} = d$.

Definition 5.14. Given a data set V (consisting of data points, along with the distances amongst the points), a universal tree T on $x, y, z \in V$ (with r as the Steiner node), let us define the following two zone types.

1. The definition for zones of type one is split into the following two cases.
   (a) $Zone_1(r) = \{w \in V : (x, y)_w = (y, z)_w = (z, x)_w\}$
   (b) For a given permutation π on $\{x, y, z\}$, $Zone_1(\pi x) = \{w \in V : (\pi x, \pi y)_w = (\pi x, \pi z)_w = d(w, \pi x)\}$
2. For a given permutation π on $\{x, y, z\}$, $Zone_2(\pi x) = \{w \in V : (\pi x, \pi y)_w = (\pi x, \pi z)_w < d(w, \pi x)\}$

Using this terminology and our structural lemmas, we can describe a recursive algorithm that reconstructs the tree structure from a 0-hyperbolic metric. Given a data set V we pick three random points x, y, z and construct the universal tree T. Then for all other $w \in V$, sort the w's into their respective zones. Then for each of the seven zones we can recursively build new universal trees. For zones of type 1, pick any two points, w_{i_1}, w_{i_2} and form the universal tree for πx (or r),w_{i_1}, w_{i_2}. If there is only one node in this zone, connect it to πx (or r). For zones of type 2, pick any one point, w_{i_1} and form the universal tree for $\pi x, w_{i_1}, r$. Note that during the recursive step for zones of type 2, we create universal trees with Steiner nodes r as one of the nodes. Hence we need to compute the distance from r to all other nodes sent to that zone. We can calculate this when we first place r. Concretely, if r is

the Steiner node for the universal tree T on x, y, z, then for any w, we will have that $d(w, r) = \max((x, y)_z, (y, z)_x, (z, x)_y)$. The proof for the consistency of this formula is in the proof of Lemma 5.13.

Finally, to complete the analysis, the following lemma proves that we only need to check consistency of the metric within each zone to ensure global consistency.

Lemma 5.15. Given (X, d) a metric tree, and a universal tree T on x, y, z, we have the following

1. If $w \in Zone_1(x)$, then for all $\hat{w} \notin Zone_1(x)$, we have that $x \in g(w, \hat{w})$.
2. If $w \in Zone_2(x)$, then for all $\hat{w} \notin Zone_i(x)$ for $i = 1, 2$, we have that $r \in g(w, \hat{w})$.

5.3.1 TreeRep for General δ-Hyperbolic Metrics.

Having seen the main geometric ideas behind TreeRep, we want to extend the algorithm to return an approximating tree for any given metric. For an arbitrary δ-hyperbolic metric, Lemma 5.13 does not hold. We can, however, modify it and leverage the intuition behind the original proof. Given four points w, x, y, z, we do not satisfy one of the conditions of Lemma 5.13, if all three Gromov products $(x, y)_w, (x, z)_w, (y, z)_w$ have distinct values. Nevertheless, we can still compute the maximum of these three quantities. Furthermore, since we have a δ-hyperbolic metric, the smaller two products will be within δ of each other. Let us suppose that $(x, y)_w$ is the biggest. Then we place w in $Zone_1(x)$ if and only if $d(z, w) = (y, z)_w$ *or* $d(z, w) = (x, z)_w$. Otherwise we place $w \in Zone_2(x)$. Note that when we have tree metric, we have that $d(z, w) = (y, z)_w$ if and only if $d(z, w) = (x, z)_w$.

As shown by Proposition 5.16, when we do this, we are introducing a distortion of at most δ between w and y, z. This suggests that when we do zone 2 recursive steps, we should pick the node that closest to r as the third node for the universal tree. We see experimentally that this significantly improves the quality of the tree returned. Note, we do not have a global distortion bound for when the input is

a general δ-hyperbolic metric. However, as we will see experimentally, we tend to produce trees with low distortion.

Proposition 5.16. Given a δ-hyperbolic metric d, the universal tree T on x, y, z and a fourth point w, when sorting w into its zone ($zone_i(\pi x)$), TreeRep introduces an additive distortion of at most δ between w and $\pi y, \pi z$.

Algorithm 16 Metric to tree structure algorithm.

1: **function** Tree structure(X, d)
2: $T = (V, E, d') = \emptyset$
3: Pick any three data points uniformly at random $x, y, z \in X$.
4: $T =$ recursive_step($T, X, x, y, z, d, d_T,$)
5: **return** T
6: **function** recursive_step($T, X, x, y, z, d, d_T,$)
7: Construct universal tree for x, y, z and sort the other nodes into the seven zones.
8: Recurse for each of the seven zones by calling Zone1_Recursion and Zone2_recursion. **return** T

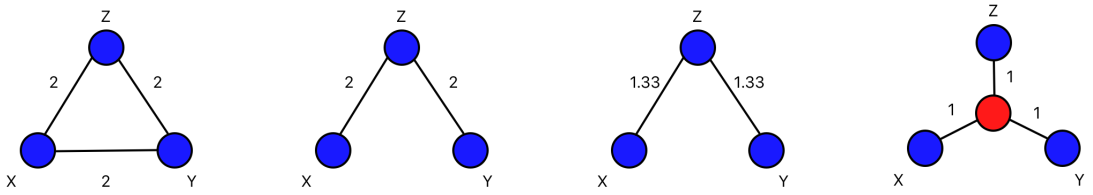
Algorithm 17 Recursive parts of TreeRep.

1: **function** zone1_recursion(T, d_T, d, L, v)
2: **if** Length(L) == 0 **then return** T
3: **if** Length(L) == 1 **then**
4: Let u be the one element in L and add edge (u, v) to E with weight $d_T(u, v) = d(u, v)$
5: **return** T
6: Pick any two u, z from L and remove them from L
7: **return** recursive_step($T, L, v, u, z, d,\ d_T$)
8: **function** zone2_recursion(T, d_T, d, L, u, v)
9: **if** Length(L) == 0 **then return** T
10: Set z to be the closest node to v and delete edge (u, v)
11: **return:** recursive_step($T, L, v, u, z, d,\ d_T$)

5.3.2 Steiner nodes.

A Steiner node is any node that did not exist in the original graph that one adds to it. We give a simple example to illustrate that Steiner nodes are necessary for

reconstructing the correct tree. Additionally, we demonstrate that forming a graph and then computing any spanning tree (as done in *Alon et al.* (1995); *Elkin et al.* (2005); *Prim* (1957)) will not recover the tree structure. Consider 3 points x, y, z such that all pairwise distances are equal to 2. Then, the associated graph is a triangle and any spanning tree is a path. Then, the distance between the endpoints of the spanning tree is not correct; it has been distorted or stretched. The "correct" tree is obtained by adding a new node r and connecting x, y, z to r, and making all the edge weights equal to 1. Thus, we need Steiner nodes when reconstructing the tree structure. Methods such as MST and LS *Alon et al.* (1995) that do not add Steiner nodes will not produce the correct tree, when given a 0-hyperbolic metric, even though such algorithms do come with upper bounds on the distortion of the distances. In this setting, we want to obtain a tree that as accurately as possible represents the metric even at the cost of additional nodes; we do not simply want a tree that is a subgraph of a given graph.



(i) Weighted graph from metric.

(ii) MST

(iii) Solve for closest weights with L_2 penalty.

(iv) Correct tree on x, y, z.

Figure 5.2: Figures showing the example that demonstrates the need fr Steiner nodes.

5.4 Experiments

In this section, we demonstrate the effectiveness of TreeRep. Additional details about the experiments and algorithms can be found in Appendix B.7.[§]

[§]All code can be found at the following link `https://github.com/rsonthal/TreeRep`

For the first task of approximating metrics with tree metrics, we compare TreeRep against algorithms that find approximating trees; Minimum Spanning Trees (MST) *Prim* (1957), LevelTrees (LT) *Chepoi et al.* (2008), Neighbor Join (NJ) *Saitou and Nei* (1987), Low Stretch Trees (LS) *Alon et al.* (1995); *Elkin et al.* (2005), ConstructTree (CT) *Abraham et al.* (2007), and ProbTree (BT) *Bartal* (1998). When comparing against such methods, we show that not only is TreeRep much faster than all of the above algorithms (except MST, and LS), but that TreeRep produces better quality metrics than MST, LS, LT, BT, and CT and metrics that are competitive with NJ. In addition to these methods, other methods such as UPGMA *Sokal et al.* (1958) also learn tree structures. However, these algorithms have other assumptions on the data. In particular, for UPGMA, the additional assumption is that the metric is an ultrametric. Hence we do not compare against such methods.

One important distinction between methods such as LS, MST, and LT and the rest, is that LS, MST, and LT require a graph as the input. This graph is crucial for these methods and hence sets these methods apart from the rest, as the rest only require a metric.

For the second task of learning hyperbolic embeddings, we compare TreeRep against Poincare Maps (PM) *Nickel and Kiela* (2017), Lorentz Maps (LM) *Nickel and Kiela* (2018b), PT *Sala et al.* (2018), and hMDS *Sala et al.* (2018). Since we can embed trees into $\mathbb{H}^k$ with arbitrarily low distortion, we think of trees as hyperbolic representations. When comparing against such methods, we show that TreeRep is not only four to five orders of magnitude faster, but for low dimensions, and in many high dimensional cases, produces better quality embeddings.

We first perform a benchmark test for tree reconstruction from tree metrics. Then, for both tasks, we test the algorithms on three different types of data sets. First, we create synthetic data sets by sampling random points from $\mathbb{H}^k$. Second, we will take real world biological data sets that are believed to have hierarchical structure. Third,

we consider metrics that come from real world unweighted graphs. In each case, we will show that TreeRep is an extremely fast algorithm that produces as good or better quality metrics. We will evaluate the methods on the basis of computational time, and the average distortion, as well as mean average precision (MAP) of the learned metrics.¶

Remark 5.17. TreeRep is a randomized algorithm, so all numbers reported are averaged over 20 runs. The best number produced by TreeRep can be found in the Appendix.

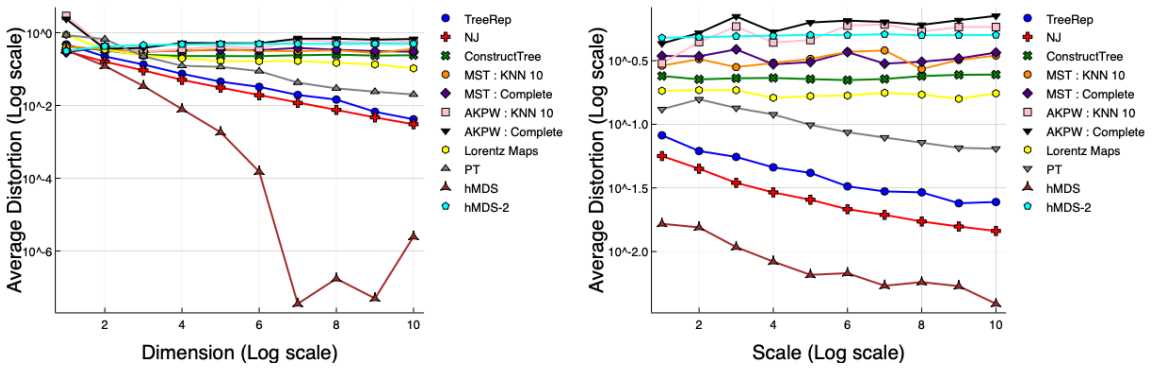
5.4.1 Tree Reconstruction Experiments.

Before experimenting with general δ-hyperbolic metrics, we benchmark our method on 0-hyperbolic metrics. To do this, we generate random synthetic 0-hyperbolic metrics. More details can be found in Appendix B.7. Since TreeRep and NJ are the only algorithms that are theoretically guaranteed to return a tree that is consistent with the original metric, we will run this experiment with these two algorithms only. We compare the two algorithms based on their running times and the number of nodes in the trees. As we can see from Table 5.1, TreeRep is a much more viable algorithm at large scales. Additionally, the trees returned by NJ have double the number of nodes as the original trees. Contrarily, the trees returned by TreeRep have exactly the same number of nodes as the original trees.

Table 5.1: Time taken by Nj and TreeRep to reconstruct the tree structure.

n	11	40	89	191	362	817	1611
TR	0.053	0.23	0.0017	0.0039	0.02	0.08	0.12
NJ	0.084	0.0016	0.0067	0.036	0.18	1.7	15

¶MAP is used in *Nickel and Kiela* (2017, 2018b); *Sala et al.* (2018), while average distortion is used in *Sala et al.* (2018). The definitions are in the appendix.



(i) Varied Dimension. (ii) Varied Scale.

Figure 5.3: Average distortion of the metric learned for 100 randomly sampled points from $\mathbb{H}^k$ for $k = 2^i$ and from $\mathbb{H}^{10}$ for scale $s = 2^i$ for $i = 1, 2, \ldots, 10$.

Table 5.2: Time taken by PT, LM, hMDS, to learn a 10 dimensional embedding for the synthetic data sets and average time taken by TreeRep (TR), MST, and CT.

	TR	NJ	MST	LS	CT	PT	LM	hMDS	hMDS-2
Time	0.002	0.06	0.0001	0.002	0.076	312	971	11.7	0.008

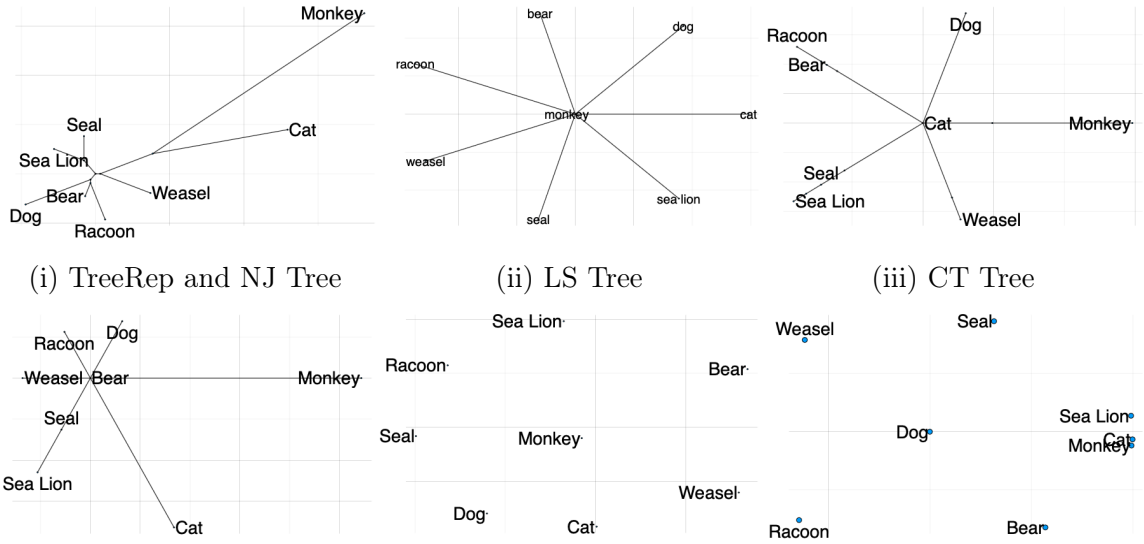
5.4.2 Random points on Hyperbolic Manifold.

We generate two different types of data sets. First, we hold the dimension k constant and scale the coordinates. Second, we hold the magnitude of the coordinates constant and increase the dimension k. Note these metrics do not come with an underlying graph! Hence to even apply methods such as MST, or LS we need to do some work. Hence, we create two different weighted graphs; a complete graph and a nearest neighbor graph.

For both types of data, Figures 5.3i and 5.3ii show that as the scale and the dimension increase, the quality of the trees produced by TreeRep and NJ get better. Contrastingly, the quality of the trees produced by MST, ConstructTree, and LS do not improve. Hence we see that when we do not have an underlying sparse graph that was used to generate the metric, methods such as MST and LS do not perform well. In fact, they have the worst performance. This greater generality of possible inputs is one of the major advantages of our method. Thus, demonstrating

that TreeRep is an extremely fast algorithm that produces good quality trees that approximate hyperbolic metrics. Furthermore, Table 5.2 shows that TreeRep is a much faster algorithm than NJ.

For the second task of finding hyperbolic embeddings, we compare against LM, PT and hMDS. For both LM, PT, and hMDS, we compute an embedding into $\mathbb{H}^k$, where k is dimension of the manifold the data was sampled from. We also use hMDS to embed into $\mathbb{H}^2$, we call this hMDS-2. We can see from Figures 5.3i and 5.3ii that TreeRep produces *much better* embeddings than LM, PT, and hMDS-2. Furthermore, LM and PT are extremely slow, with PT and LM taking 312 and 917 seconds on average, respectively. Thus, showing that TreeRep is 5 orders of magnitude faster than LM and PT, *and produces better quality representations.* On the other hand, since our points come from $\mathbb{H}^k$ if we try embedding into $\mathbb{H}^k$ with hMDS we should theoretically have zero error. However, these are high dimensional representations. We want low dimensional hyperbolic representations. Thus, we compared against hMDS-2 which did not perform well.



(i) TreeRep and NJ Tree (ii) LS Tree (iii) CT Tree

(iv) MST Tree (v) PM Embedding (vi) PT Embedding

Figure 5.4: Tree structure and embeddings for the Immunological distances from *Sarich* (1969).

Table 5.3: Time taken in seconds and the average distortion of the tree metric learned by TreeRep, NJ, MST, and CT and of the 2-dimensional hyperbolic representation learned by PM and PT on the Zeisel and CBMC data set. The numbers for TreeRep (TR) are the average numbers over 20 trials.

	Zeisel							CBMC			
	TR	NJ	MST	LS	CT	PT	PM	TR	NJ	MST	LS
Time	0.36	122.2	0.11	7.2	>14400	8507	12342	2.8	>14400	0.55	30
Distortion	0.117	0.144	0.365	0.250	n/a	0.531	0.294	0.260	n/a	1.09	1.45

5.4.3 Biological Data: scRNA seq and phylogenetic data.

We also test on three real world biological data sets. The first data set consists of immunological distances from *Sarich* (1969). Given these distances, the goal is to recover the hierarchical phylogenetic structure. As seen in Figure 5.4, the trees returned by TreeRep and NJ recover this structure well, with sea lion and seal close to each other, and monkey and cat far away from everything else. Divergently, the trees and embeddings produced by MST, LS, ConstructTree, PM, and PT make less sense as phylogenetic trees.

The second type of data sets are the Zeisel and CBMC sc RNA-seq data set *Zeisel et al.* (2015); *Stoeckius et al.* (2017). These data sets are expected to be a tree as demonstrated in *Dumitrascu et al.* (2019). Here we used the various algorithms to learn a tree structure on the data or to learn an embedding into $\mathbb{H}^2$. The time taken and the average distortion are reported in Table 5.3. In this case, we see that TreeRep has the *lowest* distortion. Additionally, TreeRep is 20 times faster than NJ and is 20,000 to 40,000 times faster than PT and PM. Furthermore, NJ, CT, PT, and PM timed out (took greater than 4 hours) on the CBMC data set. For the CBMC data set, we see that TreeRep is *only* algorithm that produces good quality embeddings in a reasonable time frame. Again we see that if the input is a metric instead of a graph, algorithms such as MST and LS do not do well. We also tried to use hMDS for this experiment, but it either didn't output a metric or it outputted the all zero

metric.

Table 5.4: Table with the time taken in seconds, MAP, and average distortion for all of the algorithms when given metrics that come from unweighted graph. Darker cell colors indicates better numbers for MAP and average distortion. The number next to PT, PM, LM is the dimension of the space used to learn the embedding. The numbers for TreeRep (TR) are the average numbers over 20 trials.

Graph	TR	NJ	MST	LT	CT	LS	PT 2	PT 200	PM 2	LM 2	LM 200	PM 200
						MAP						
Celegan	0.473	0.713	0.337	0.272	0.447	0.313	0.098	0.857	0.479	0.466	0.646	0.662
Diseasome	0.895	0.962	0.789	0.725	0.815	0.785	0.392	0.868	0.799	0.781	0.874	0.886
CS Phd	0.979	0.993	0.991	0.964	0.807	0.991	0.190	0.556	0.537	0.537	0.593	0.593
Yeast	0.815	0.892	0.871	0.742	0.859	0.873	0.235	0.658	0.522	0.513	0.641	0.643
Grid-worm	0.707	0.800	0.768	0.657	-	0.766	-	-	0.334	0.306	0.558	0.553
GRQC	0.685	0.862	0.686	0.480	-	0.684	-	-	0.589	0.603	0.783	0.784
Enron	0.570	-	0.524	-	-	0.523	-	-	-	-	-	-
Wordnet	0.984	-	0.989	-	-	0.989	-	-	-	-	-	-
						Average Distortion						
Celegan	0.197	0.124	0.255	0.166	0.325	0.353	0.236	0.096	0.236	0.249	0.224	0.211
Diseasome	0.188	0.161	0.161	0.157	0.315	0.228	0.227	0.05	0.323	0.328	0.335	0.332
CS Phd	0.204	0.134	0.298	0.161	0.282	0.291	0.295	0.105	0.374	0.378	0.378	0.380
Yeast	0.205	0.149	0.243	0.243	0.282	0.243	0.230	0.089	0.246	0.248	0.234	0.234
Grid-worm	0.188	0.135	0.171	0.202	-	0.234	-	-	0.196	0.203	0.192	0.193
GRQC	0.192	0.200	0.275	0.267	-	0.206	-	-	0.212	0.198	0.193	0.193
Enron	0.453	-	0.607	-	-	0.562	-	-	-	-	-	-
Wordnet	0.131	-	0.336	-	-	0.071	-	-	-	-	-	-
						Time in seconds						
Celegan	0.014	0.28	0.0002	0.086	0.9	0.001	573	1156	712	523	1578	1927
Diseasome	0.017	0.41	0.0003	0.39	15.76	0.001	678	1479	414	365	978	1112
CS Phd	0.037	2.94	0.0007	1.97	226	0.006	1607	4145	467	324	768	1149
Yeast	0.057	8.04	0.0008	8.21	957	0.001	9526	17876	972	619	1334	2269
Grid-worm	0.731	163	0.001	191	-	0.007	-	-	2645	1973	4674	5593
GRQC	0.42	311	0.0014	70.9	-	0.006	-	-	7524	7217	9767	1187
Enron	27	-	0.013	-	-	0.13	-	-	-	-	-	-
Wordnet	74	-	0.18	-	-	0.08	-	-	-	-	-	-

Table 5.5: Graph Statistics

Graph	n	m	δ
Celegan	452	2024	021
Diseasome	516	1188	0.17
CS PhD	1035	1043	0.23
Yeast	1458	1948	≤ 0.32
Grid-worm	3337	6421	≤ 0.38
GRQC	4158	13422	≤ 0.36
Enron	33695	180810	-
Wordnet	74374	75384	-

5.4.4 Unweighted Graphs.

Finally, we consider metrics that come from unweighted graphs. We use eight well known graph data sets from *Rossi and Ahmed* (2015). Table 5.4 records the performance of all the algorithms for each of these data sets. For learning tree metrics to approximate general metrics, we see that NJ has the best MAP, with TreeRep, MST, and LS tied for second place. In terms of distortion, NJ is the best, TreeRep is second, while MST is third and LS is sixth. However, NJ is extremely slow and is not viable at scale. Hence, in this case, we have three algorithms with good performance at large scale; TreeRep, MST, and LS. However, MST and LS did not perform well in the previous experiments.

For the task of learning hyperbolic representations, we see that PM, LM, and PT are much slower than the methods that learn a tree first. In fact, these algorithms were too slow to compute the hyperbolic embeddings for the larger data sets. Additionally, this extra computational effort does not always result in improved quality. In all cases, except for the Celegan data set, the MAP returned by TreeRep is superior to the MAP of the 2-dimensional embeddings produced by PM, LM, and PT. In fact, in most cases, these 2-dimensional embeddings, have worse MAP than all of the tree first methods. Even when they learn 200-dimensional embeddings, PM, LM and PT have worse MAP than TreeRep on most of the data sets. Furthermore, except for PT200,

the average distortion of the metric returned by TreeRep is superior to PT2, PM, an LM. Thus, showing the effectiveness of TreeRep at learning good Hyperbolic representations quickly.

5.5 Broader Impact

There are multiple aspects to the broader impacts of our work, from the impact upon computational biology, specifically, to the impact upon data sciences more generally. The potential impacts on society, both positive and negative, are large. Computational biology is undergoing a revolution due to simultaneous advances in the creation of novel technologies for the collection of multiple and novel sources of data, and in the progress of the development of machine learning algorithms for the analysis of such data. Social science has a similar revolution in its use of computational techniques for the analysis and gathering of data.

Cellular differentiation is the process by which cells transition from one cell type (typically an immature cell) into more specialized types. Understanding how cells differentiate is a critical problem in modern developmental and cancer biology. Single-cell measurement technologies, such as single-cell RNA-sequencing (scRNA-seq) and mass cytometry, have enabled the study of these processes. To visualize, cluster, and infer temporal properties of the developmental trajectory, many researchers have developed algorithms that leverage hierarchical representations of single cell data. To discover these geometric relationships, many state-of-the-art methods rely on distances in low-dimensional Euclidean embeddings of cell measurements. This approach is limited, however, because these types of embeddings lead to substantial distortions in the visualization, clustering, and the identification of cell type lineages. Our work is specifically focused on extracting and representing hierarchical information.

On the more negative side, these algorithms might also be used to analyze social hierarchies and to divine social structure from data about peoples' interactions. Such

tools might encourage, even justify, the intrusive and pervasive collection of data about how people interact and with whom.

Work partially supported by funds from the Michigan Institute for Data Science.

CHAPTER VI

Dual Regularized Optimal Transport

6.1 Introduction

Optimal transport is a ubiquitous problem in areas ranging from economics and the allocation of resources to Riemannian geometry and measure theory. The motivation for and description of the basic problem arises from transporting objects from one set of locations to the another set of locations using a minimal cost transportation plan. Over the past century, but especially the last three decades, considerable work has been done to understand the geometry of the problem and its various formulations. Many different variants of the problem have been posed and algorithmic approaches have been developed to solve these variants. Most importantly for our work, there has also been great interest and activity in applying optimal transport to machine learning, computer vision, and domain transfer tasks. Optimal transport in the setting of machine learning tasks is the starting point of this paper.

6.1.1 Background

There are several versions of the optimal transport problem that we use to motivate our formulation. The original version is that of Monge. The Monge problem, however, has some drawbacks (namely, the transport map must be a function) and, for this reason, we begin with its natural generalization, the Monge-Kantorovich problem.

Problem 6.0.1. Given two probability spaces $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \nu)$, and a cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, the Monge-Kantorovich Optimal Transport seeks a joint probability π on $\mathcal{X} \times \mathcal{Y}$ that minimizes $\int_{\mathcal{X}\times\mathcal{Y}} c(x, y)d\pi(x, y)$, subject to the constraints that the pushforward of the marginals are consistent with the inputs, $\mathcal{P}^{\mathcal{X}}_{\#}\pi = \mu$ and $\mathcal{P}^{\mathcal{Y}}_{\#}\pi = \nu$.

In a finite discrete setting this problem can be formulated as a linear program (see Problem 6.0.2) which, unfortunately, is challenging to solve algorithmically but which does guarantee sparse solutions. Two predominant methods are combinatorial *Bertsekas and Castanon* (1989); *Gabow* (1985); *Duff and Koster* (2001) and PDE based solvers *Benamou and Brenier* (2000). None of these methods, however, scales well. As a result, there are many alternative formulations of the OT problem that are easier to solve, including those formulation types that include regularizing the primal objective function (see, for example, *Cuturi* (2013); *Essid and Solomon* (2017); *Blondel et al.* (2018b); *Lorenz et al.* (2019)) with or without relaxed constraints. These variants are referred to as regularized optimal transport. There is a second class of formulations called unbalanced optimal transport (see for example *Liero et al.* (2017); *Chizat et al.* (2016b); *Blondel et al.* (2018b)). There are a number of proposed efficient algorithms to solve these various formulations, including *Seguy et al.* (2018); *Schmitzer* (2019); *Solomon et al.* (2015); *Frogner et al.* (2015); *Benamou et al.* (2015); *Genevay et al.* (2016); *Alaya et al.* (2019). Despite such algorithmic advances, the regularized optimal transport problems either do not produce sparse transport plans which hampers interpretability for machine learning tasks or they do not perform well in practice. The main drawback with unbalanced optimal transport is that it is unclear how the solution methods balance creation, destruction, and transport of mass, all of which can generate unexpected artifacts.

Besides the two above categories there are a few other general optimal transport formulations. *Alvarez-Melis et al.* (2018) generalizes the formulation of optimal transport to include side information. *Xie et al.* (2019) uses generative neural networks

to solve the problem when there is a map from a latent space to $\mathcal{Z}$ to the spaces $\mathcal{X}, \mathcal{Y}$. Besides general solvers, there are also many more restricted solutions for application specific problems, for examples see *Backurs et al.* (2019); *Petric Maretic et al.* (2019).

6.1.1.1 Our Contribution.

In this paper, we present a new formulation of optimal transport that regularizes the dual problem without relaxing the dual constraints. We refer to this formulation as Dual Regularized Optimal Transport or DROT. We show that this problem has a number of both theoretical and algorithmic properties that the other formulations of the problem do not have. Specifically,

1. the dual of DROT is a form of unbalanced optimal transport whose solution leads to sparse solutions to the optimal transport problem;
2. DROT can be solved efficiently at large scales via PROJECT AND FORGET *Gilbert and Sonthalia* (2020a);
3. with the appropriate choice of the dual regularizer, unlike other optimal transport formulations, we can easily control the level of mass creation versus destruction;

We also provide extensive experimental evidence for our analysis and the performance of PROJECT AND FORGET in a number of settings.

6.2 Preliminaries

For all of our algorithmic discussions, we work in a finite, discrete setting. Let Δ^n denote the $n-1$ dimensional probability simplex. Then, $(\Delta^m, \boldsymbol{a})$ and $(\Delta^n, \boldsymbol{b})$ denote two finite probability spaces and we denote by $\boldsymbol{P}$ the joint distribution on $\Delta^m \times \Delta^n$. Note that $\boldsymbol{P}$ can be represented by an $m \times n$ matrix. The cost function we denote by an $m \times n$ matrix $\boldsymbol{C}$. The vector of all ones of length m is denoted $\mathbf{1}_m$. The Frobenius dot product of two matrices $\boldsymbol{A}, \boldsymbol{B}$ we denote by $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$. For some problem formulations and in an abuse of notation, the distributions $\boldsymbol{a}$ and $\boldsymbol{b}$ on their respective spaces need

not have the same total mass (i.e., they are not strictly probability measures). Finally, given a convex function ϕ, we denote its convex conjugate by ϕ^*.

6.2.1 Background Problem Formulations

In a finite discrete setting the Monge-Kantorovich OT problem can be formulated as the following linear problem.

Problem 6.0.2. Given two probability spaces $(\Delta^m, \boldsymbol{a})$ and $(\Delta^n, \boldsymbol{b})$ and a cost function $\boldsymbol{C}$, we seek the mass transportation map of minimal cost that is consistent with the input distributions:

$$\begin{aligned} \mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}) &= \min \langle \boldsymbol{C}, \boldsymbol{P} \rangle \\ \text{subject to: } \boldsymbol{a} &= \boldsymbol{P}\mathbf{1}_m,\ \boldsymbol{b} = \boldsymbol{P}^T\mathbf{1}_n,\ \boldsymbol{P} \geq 0. \end{aligned} \tag{6.1}$$

One important feature of the solution to Problem 6.0.2 is that it is sparse. Specifically, at most $n+m-1$ entries of $\boldsymbol{P}$ are non-zero *Brualdi* (2006) which means that for applications in machine learning and image processing, the solutions are "interpretable" and they have efficient implementations.

We sketch those problem formulation types that include regularizing the primal objective function with or without relaxed constraints.

6.2.1.1 Regularized and Unbalanced Optimal Transport.

In the first formulation variant (Regularized Optimal Transport or ROT), we use an entropic regularizer without relaxing the constraints. Cuturi *Cuturi* (2013) shows that by adding an entropic regularizer, the ROT problem can be solved quickly with

the Sinkhorn matrix scaling algorithm.

$$\begin{aligned} \text{ROT}(\boldsymbol{a}, \boldsymbol{b}) &= \min \langle \boldsymbol{C}, \boldsymbol{P} \rangle + \gamma \sum_{i,j} \boldsymbol{P}_{ij} \log(\boldsymbol{P}_{ij}) \\ \text{subject to: } & \boldsymbol{a} = \boldsymbol{P}\mathbf{1}_m, \, \boldsymbol{b} = \boldsymbol{P}^T \mathbf{1}_n. \end{aligned} \tag{6.2}$$

This formulation has proven to be extremely useful in practice despite the loss in sparsity of the solution which smooths the transportation plan.

A second natural regularizer is the quadratic function. *Essid and Solomon* (2017); *Blondel et al.* (2018b); *Lorenz et al.* (2019) study this variant and show experimentally that the solutions are sparse. Generalizing further, *Dessein et al.* (2018b) use Bregman functions, a natural extension of *Benamou et al.* (2015).

A second main formulation variant (Unbalanced Optimal Transport or UOT) maintains the regularized primal objective function but relaxes the constraints on the marginal distributions. In a variety of applications, the input distributions do *not* adhere to being probability measures and they have *different* total mass. As a result, *Liero et al.* (2017) formulate transport between densities with different masses, or unbalanced optimal transport. In this variant, we relax the constraint that marginals of the transport must match the given marginals and instead penalize the deviation from the marginals. Similar to *Cuturi* (2013), *Liero et al.* (2017) use entropy based divergences, such as the KL divergence, as the penalty function. *Chizat et al.* (2016b) present matrix scaling algorithms for UOT.

$$\begin{aligned} \text{UOT}(\boldsymbol{a}, \boldsymbol{b}) = \min \langle \boldsymbol{C}, \boldsymbol{P} \rangle &+ \gamma_1 \sum_{i,j} \boldsymbol{P}_{ij} \log(\boldsymbol{P}_{ij}) \\ &+ \gamma_2 KL(\boldsymbol{P}\mathbf{1}_m, \boldsymbol{a}) + \gamma_3 KL(\boldsymbol{P}^T \mathbf{1}_n, \boldsymbol{b}). \end{aligned} \tag{6.3}$$

Blondel et al. (2018b) consider UOT with quadratic penalty terms and also considers an asymmetric version of the problem in which only one marginal constraint has been relaxed. The Monge version of the problem also has a relaxation that is similar to the

unbalanced version of the Monge-Kantorovich problem *Yang and Uhler* (2019).

The main drawback with the current formulations of unbalanced optimal transport, is that it is unclear how the solution methods balance creation, destruction, and transport of mass. These formulations give us control over mass creation and destruction versus transport, by increasing or decreasing the penalty, but we do not have control over the degree of creation versus that of destruction.

6.2.2 Dual regularized optimal transport (DROT)

To build on the various previous OT problem formulations, we devise a new formulation via dual regularization. We add a regularizer term to the dual objective function so that it is strictly concave but we do *not* relax the dual constraints. This may be interpreted as adding a strictly convex regularizer to the primal problem and relaxing the primal constraints, leading to an unbalanced optimal transport problem. We state the discrete version of the problem and note there is a natural continuous version which we do not state.

Problem 6.0.3. Given $\boldsymbol{a}$ and $\boldsymbol{b}$ two vectors of length m and n respectively (representing two distributions on m and n points), an $m \times n$ cost matrix $\boldsymbol{C}$, two strictly convex function φ and ϕ, and a regularization parameter γ, find vectors $\boldsymbol{f}$ and $\boldsymbol{g}$ that maximize

$$\begin{aligned} \mathrm{DROT}(\boldsymbol{a}, \boldsymbol{b}) &= \max \langle \boldsymbol{f}, \boldsymbol{a} \rangle + \langle \boldsymbol{g}, \boldsymbol{b} \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}) + \varphi(\boldsymbol{g})) \\ &\text{subject to: } \boldsymbol{f}_i + \boldsymbol{g}_j \leq \boldsymbol{C}_{i,j}. \end{aligned} \tag{6.4}$$

Let us consider the interpretation of this formulation. We begin with that of *Peyré and Cuturi* (2018). Suppose we have n warehouses and m stores. Let $\boldsymbol{a}$ be the vector whose ith component is the number of items in warehouse i and $\boldsymbol{b}$ be the m dimensional vector for the demand of each store. Let $\boldsymbol{C}$ be the cost to transport items from warehouses to stores. Next, suppose we are an external shipper; we charge $\boldsymbol{f}_i$ to pick up good from warehouse i regardless of where it is delivered and $\boldsymbol{g}_j$ to deliver

goods to store j regardless of the originating warehouse. We want to maximize our income which is given by $\langle \boldsymbol{f}, \boldsymbol{a} \rangle + \langle \boldsymbol{g}, \boldsymbol{b} \rangle$ but our prices must satisfy $\boldsymbol{f}_i + \boldsymbol{g}_j \leq \boldsymbol{C}_{ij}$, some cost constraint. The addition of the regularizer in the objective function, therefore, regularizes the prices we can charge. This is in contrast with the formulation developed in *Liero et al.* (2017) which penalizes the divergence from the input distribution. In many applications, such as domain transfer, color transfer, and economics, regularizing prices (i.e., how profitable is it to transfer both to and from a certain data point) is more natural. For example, we may want to regularize prices and see how this affect this the distributions $\boldsymbol{a}, \boldsymbol{b}$, representing demand and supply.

6.2.3 Extension to Multi-marginal Transport.

We can extend the DROT formulation to multi-marginal optimal transport, in which, instead of transporting mass between two distributions, we learn a joint distribution π over k probability spaces $(\mathcal{X}_1, \mu_1), \ldots (\mathcal{X}_k, \mu_k)$ such that the marginals of π are the μ_i. The (continuous) DROT formulation of the multi-marginal problem is as follows:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{k} \gamma \phi_i(\boldsymbol{f}_i) - \int_{\mathcal{X}_i} \boldsymbol{f}_i d\mu_i \\ \text{subject to:} \quad & (x_1, \ldots, x_k) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_k, \\ & \textstyle\sum_{i=1}^{k} \boldsymbol{f}_i(x_i) \leq c(x_1, \ldots, x_k) \end{aligned} \tag{6.5}$$

An interpretation of the multi-marginal optimal transport problem is that we pick different combinations of variables from the k spaces such that a given combination $x_1, \ldots, x_k$ costs $c(x_1, \ldots, x_k)$. We seek a price list $\boldsymbol{f}_i$ for each of the elements in $\mathcal{X}_i$ such that if demand for each variable is distributed according to μ_i, then we maximize our expected profits. This interpretation of the dual problem is much more amenable to applications and also aligns with a common choice of cost function, $c(x_1, \ldots, x_k) = \sum_{j \neq k} \|x_i - x_j\|_2^2$. *Gangbo and Swiech* (1998) guarantee the uniqueness of such transport plans for precisely such costs.

6.3 Theoretical analysis

In this section, we detail the theoretical analysis of the DROT problem formulation. We begin with an analysis of the features of the solutions. We then discuss the choice of regularizer. We end with a discussion of an algorithmic method for solving Problem 6.4, PROJECT AND FORGET, a general method developed in *Gilbert and Sonthalia* (2020a).

6.3.1 Solution properties

In this section, we analyze the properties of the solutions to the DROT problem. This analysis includes the relation between the solution to the DROT Problem 6.4 and that of other OT formulations (i.e., the approximation quality of the solution), how the solutions depend on the regularization parameter, and finally, what the trade-offs are in the creation and destruction of mass. All formal proofs can be found in the Appendix.

Definition 6.1. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say a function is positive co-finite if for all $x \geq 0$, $f(rx)/r \to \infty$ as $r \to \infty$. Similarly, a function is negative co-finite if for all $x \leq 0$, $f(rx)/r \to \infty$ as $r \to \infty$. A function is co-finite if it is both positive and negative co-finite.

Theorem 6.2. If we add the assumption that ϕ, φ are co-finite Bregman functions to our hypotheses for Problem 6.4, then the following problem is the dual problem to DROT($\boldsymbol{a}, \boldsymbol{b}$). Furthermore, strong duality holds.

$$\begin{aligned} &\min \langle \boldsymbol{C}, \boldsymbol{P} \rangle + \frac{\phi^* \left(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m)\right)}{\gamma} + \frac{\varphi^* \left(\gamma(\boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_n)\right)}{\gamma} \\ &\text{subject to: } \forall i \in [n], \forall j \in [m], \quad \boldsymbol{P}_{ij} \geq 0. \end{aligned} \tag{6.6}$$

If we only have the assumption that ϕ (and similarly for φ) is positively (negatively) co-finite, then we must add the constraint $\boldsymbol{a} - \boldsymbol{P}\mathbf{1} > 0$ ($\boldsymbol{a} - \boldsymbol{P}\mathbf{1} < 0$).

Theorem 6.2 shows us the dual formulation of DROT resembles unbalanced optimal transport problems from *Liero et al.* (2017), but with different types of penalty functions on the transport map. Indeed, if we set ϕ and φ to be quadratic regularizers, then Theorem 6.2 shows that the dual DROT formulation and a formulation in *Blondel et al.* (2018b) are equivalent.

Furthermore, note that if ϕ, φ are positive co-finite functions, then DROT necessarily destroys mass. On the other hand, if ϕ, φ are negative co-finite function, then DROT necessarily creates mass. This matches our intuition exactly. In the objective function for DROT, the regularizer term is $\phi(\boldsymbol{f}) + \varphi(\boldsymbol{g})$ which we seek to minimize. For positive co-finite functions, we do so when both $\boldsymbol{f}$ and $\boldsymbol{g}$ are highly negative. Using the shipping interpretation of the dual problem, $\boldsymbol{f}$ and $\boldsymbol{g}$ represent the prices we charge to ship and a negative price means that we, as shippers, *pay* to do the shipping! Such incentives result in *not* shipping goods or, more abstractly, destroying mass. On the other hand, for negatively co-finite functions, we minimize the objective function when $\boldsymbol{f}, \boldsymbol{g}$ are both highly positive; that is, we are incentivized to ship more goods, or to create mass.

We note that for the dual DROT formulation, it is not necessary that ϕ^*, φ^* attain their minima at 0 (the minimum is attained at 0 if and only if ϕ, φ attain their minima at 0) and, under such conditions, the regularizers actually encourage some deviation from the marginals $\boldsymbol{a}, \boldsymbol{b}$; thus, encouraging the creation or destruction of mass. Note we could also introduce similar incentives in other variants, but such incentives have not been studied before.

The next proposition quantifies how far the solution to DROT is from that of the Monge-Kantorovich formulation.

Proposition 6.3. Let $\boldsymbol{P}^*, \boldsymbol{f}^*, \boldsymbol{g}^*$ be the optimal solutions, primal and dual, to the Monge-Kantorovich formulation (Problem 6.0.2) and let $\boldsymbol{P}^*_{\phi,\varphi}, \boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{g}^*_{\phi,\varphi}$ be the optimal solutions to DROT, Problem 6.4. Then we have that the following are true.

1. The difference between the value of the DROT objective and that of the Monge-Kantorovich formulation is upper and lower bounded by

$$\begin{aligned}\phi\left(\boldsymbol{f}^*_{\phi,\varphi}\right) + \varphi(\boldsymbol{g}^*_{\phi,\varphi}) &\leq \gamma(\mathrm{OT}(\boldsymbol{a},\boldsymbol{b}) - \mathrm{DROT}(\boldsymbol{a},\boldsymbol{b})) \\ &\leq \phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*).\end{aligned}$$

2. We can estimate the quality of the approximation (as a function of the regularizers ϕ and φ) as

$$\gamma\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi}\rangle \leq \phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*) + \phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m)) + \varphi^*(\gamma(\boldsymbol{b} - (\boldsymbol{P}^*_{\phi,\varphi})^T\mathbf{1}_n))$$

3. and

$$\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m)) + \varphi^*(\gamma(\boldsymbol{b} - (\boldsymbol{P}^*_{\phi,\varphi})^T\mathbf{1}_n)) \leq \gamma\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi}\rangle - \phi(\boldsymbol{f}^*_{\phi,\varphi}) - \varphi(\boldsymbol{g}^*_{\phi,\varphi}).$$

These bounds reveal how the various parameters control the problem. Specifically, we can see that error $\mathrm{OT}(\boldsymbol{a},\boldsymbol{b}) - \mathrm{DROT}(\boldsymbol{a},\boldsymbol{b})$ is $O(\gamma^{-1})$. More interestingly, we see how ϕ, φ affect the quality of the approximation. Parts 2, 3 of Proposition 6.3 also give us an interplay between the penalty incurred for not satisfying the marginal constraints and the cost of the transport.

Corollary 6.4. If $\boldsymbol{P}^*_\gamma$ is the solution to $DROT(\boldsymbol{a},\boldsymbol{b})$ for a given γ, and $\boldsymbol{P}^*$ is the solution to $\mathrm{OT}(\boldsymbol{a},\boldsymbol{b})$ then, $\|\boldsymbol{a} - \boldsymbol{P}^*_\gamma\mathbf{1}_m\|$ and $\|\boldsymbol{b} - (\boldsymbol{P}^*_\gamma)^T\mathbf{1}_n\|$, $OT(\boldsymbol{a},\boldsymbol{b}) - DROT(\boldsymbol{a},\boldsymbol{b})$, and $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_\gamma\rangle|$ are all $O(\gamma^{-1})$.

Because the sparsity of solutions to OT problems is critical for some applications, the next series of analysis is the study of the support of solutions to DROT.

Definition 6.5. Given $F : \mathbb{R}^d \times \Theta \to \mathbb{R}$ and G such that for each $\theta \in \Theta$, $G(\theta) \subset \mathbb{R}^d$, we define a parameterized family of optimization problems parameterized by $\theta \in \Theta$

where the function V, $V(\theta) = \max_{x \in G(\theta)} F(x, \theta)$ is the value function and x^*, $x^*(\theta) = \{x \in G(\theta) : F(x, \theta) = V(\theta)\}$ is the optimal policy correspondence.

Definition 6.6. Let $G : \Theta \to \mathbb{P}(\mathbb{R}^d)$ be a function from the parameter space Θ to the power set of $\mathbb{R}^d$. We say that G is upper hemicontinuous at $\theta \in \Theta$ if $G(\theta)$ is nonempty and if, for every open set $U \subset \mathbb{R}^d$ with $G(\theta) \subset U$, there exists a $\delta > 0$ such that for every $\theta' \in N_\delta(\theta)$ (every θ' in some δ-neighborhood of θ), $G(\theta') \subset U$.

Proposition 6.7. Given two discrete measures μ, ν, a cost function c, Bregman regularizers ϕ, φ and $\gamma^{-1} \in [0, \infty)$, the value function V is well defined and continuous on $[0, \infty)$ and the optimal policy correspondence x^* is well defined and continuous on $(0, \infty)$. Furthermore, if ϕ, φ are both positive co-finite or negative co-finite, then the optimal policy correspondence is upper hemicontinuous on $[0, \infty)$.

The implication of the upper-hemicontinuity of the optimal policy correspondence is that any sequence of solutions $(\boldsymbol{f}^*_{\phi,\varphi})_n, (\boldsymbol{g}^*_{\phi,\varphi})_n$ to the DROT Problem 6.4 for a sequence of $(\gamma)_n$, has a convergent sub-sequence. Lower-hemicontinuity implies that all solutions to the OT Problem 6.0.2 can be expressed as limits of sequences of solutions to DROT.

Finally, we show that the transport map $\boldsymbol{P}$ that results from solving DROT is at least as sparse as that from the OT solution. Therefore, except for the Monge-Kantorovich formulation, DROT is the only formulation of optimal transport that has a theoretical result guarantee of solution sparsity. While this is result is for the case when γ is large, as we will see experimentally, we produce sparse solutions for all γ.

Corollary 6.8. Suppose that we have an instance of Problem 6.0.2 such that for any two optimal dual solutions $(\boldsymbol{f}^*_1, \boldsymbol{g}^*_1), (\boldsymbol{f}^*_2, \boldsymbol{g}^*_2)$, we have that $\boldsymbol{f}^*_1 - \boldsymbol{f}^*_2 = c\mathbf{1}$, and $\boldsymbol{g}^*_1 - \boldsymbol{g}^*_2 = -c\mathbf{1}$. Then there exists Γ such that for all $\gamma \geq \Gamma$, if $\boldsymbol{P}^*_\gamma$ is the solution to DROT Problem 6.4 for γ and $\boldsymbol{P}^*$ is any optimal solution to Problem 6.0.2, then we have that $\text{supp}(\boldsymbol{P}^*_\gamma) \subset \text{supp}(\boldsymbol{P}^*)$.

6.3.2 Example regularizers

In this subsection, we focus on three different example regularizers: quadratic, entropic, and exponential. All of these regularizers satisfy the theoretical assumptions of the theoretical analysis in the previous subsection although there are some important differences amongst them.

6.3.2.1 Quadratic.

The quadratic regularizers are $\phi(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$ and similarly for $\varphi(\boldsymbol{g})$. This regularizer is thoroughly studied in *Blondel et al.* (2018b) and, for brevity, we do not discuss it further. We observe that the regularizer is a co-finite Bregman function.

6.3.2.2 Exponential.

Let $\phi(\boldsymbol{f}) = \sum_{i=1}^n e^{\boldsymbol{f}_i}$ and similarly for $\varphi(\boldsymbol{g})$. We observe that ϕ, φ are positively co-finite Bregman functions and, by Theorem 6.2, this formulation of DROT must destroy mass. To be more concrete, the convex dual of ϕ is $\phi^*(\boldsymbol{x}) = \sum_{i=1}^n x_i \log(x_i) - x_i$ with the stipulation that $x_i \geq 0$ and similarly for φ^*. In Theorem 6.2, the variable $\boldsymbol{x}$ in the dual formulation of DROT is $\boldsymbol{x} = \boldsymbol{a} - \boldsymbol{P}\boldsymbol{1}_m$ and the requirement that $x_i \geq 0$ implies

$$\boldsymbol{a}_i - (\boldsymbol{P}\boldsymbol{1}_n)_i \geq 0 \quad \text{or} \quad \boldsymbol{a}_i \geq (\boldsymbol{P}\boldsymbol{1}_n)_i.$$

Hence, the transport process only destroys or preserves mass; it does not create it.

6.3.2.3 Entropy.

Let $\phi(\boldsymbol{f}) = \sum_{i=1}^n \boldsymbol{f}_i \log(\boldsymbol{f}_i) - \boldsymbol{f}_i$ and similarly for $\varphi(\boldsymbol{g})$. The convex dual of ϕ is $\phi^*(x) = \sum_{i=1}^n e^{x_i}$ and similarly for φ^*. In Remark C.1 in the Appendix, we detail the additional stipulations we impose when we use the entropic regularizers. These constraints include that $(\boldsymbol{f})_i, (\boldsymbol{g})_i \geq 0$ which implies that in the dual formulation of

DROT, the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy $\boldsymbol{x} = \boldsymbol{a} - (\boldsymbol{P}\mathbf{1}_m) - \boldsymbol{c}_1$ and $\boldsymbol{y} = \boldsymbol{b} - (\boldsymbol{P}^T\mathbf{1}_n) - \boldsymbol{c}_2$, where $\boldsymbol{c}_1, \boldsymbol{c}_2$ are vectors which non-negative entries. In the optimization problem, we optimize for $\boldsymbol{c}_1, \boldsymbol{c}_2$ as well. We minimize this term in the objective when $\boldsymbol{a} - (\boldsymbol{P}\mathbf{1}_n) - \boldsymbol{c}_1$ is negative, or when $\boldsymbol{a} < (\boldsymbol{P}\mathbf{1}_n) + \boldsymbol{c}_1$ (and similarly for $\boldsymbol{b}$). Because $\boldsymbol{c}_1$ is variable, it is not clear whether we favor creating or destroying mass. As we will see, however, in the experiments, we always favor creating mass in this formulation. This matches our intuition as $\boldsymbol{f}, \boldsymbol{g}$ must be positive.

Algorithm	$n = 501$	$n = 1001$	$n = 5001$	$n = 10001$	$n = 20001$
Project and Forget	6 s	20 s	265 s	1120 s	Out of memory.
LBFGSB	24 s	162 s	4080 s	Out of memory.	
Mosek primal	7 s	27 s	981 s	Out of memory.	
Mosek dual	3 s	Out of memory.			
CPLEX dual	105 s	Out of memory.			
CPLEX primal	Out of memory.				
Projected gradient descent	Did not converge.				

Table 6.1: Time taken in seconds to solve the quadratic regularized problem when the two distributions are Gaussian distributions. Here we set $\gamma = 1000$ and all experiments were run on a machine with 54 GB of RAM.

6.3.3 Efficient algorithm: Project and Forget

While there are many different potential algorithmic techniques that could be used to solve this problem, we adopt a new algorithmic method, Project and Forget *Gilbert and Sonthalia* (2020a), which is a conversion of Bregman's cyclic method into an active set method and, as such, can solve large scale, highly constrained convex optimization problems. Project and Forget is an iterative method with three major steps per iteration: (i) an (efficient) oracle to find violated constraints, (ii) Bregman projection onto the hyperplanes defined by each of the active constraints, and (iii) the forgetting of constraints that no longer require attention.

To adapt Project and Forget for DROT, the three major steps are as follows. First, we use a naive oracle that searches through all of the constraints and adds

to the current list of active constraints any violated constraint. In particular, since each constraint is independently satisfied or not, we can do this search in parallel. In the project step, we observe that the constraints are of the form $\boldsymbol{f}_i + \boldsymbol{g}_j \leq \boldsymbol{C}_{ij}$. To calculate the projection, we first calculate $\boldsymbol{f}_i', \boldsymbol{g}_j', \theta$ as the solutions to the following equations, where $\boldsymbol{e}_i, \boldsymbol{e}_j$ are the i, jth standard basis vectors.

$$\theta \boldsymbol{e}_i := \nabla\phi(\boldsymbol{f}') - \nabla\phi(\boldsymbol{f}) \text{ and } \theta \boldsymbol{e}_j = \nabla\varphi(\boldsymbol{g}') - \nabla\varphi(\boldsymbol{g}).$$

An analytic formula for θ, that only depends on $\boldsymbol{f}_i, \boldsymbol{g}_j, \boldsymbol{C}_{ij}$ for the different regularizers can be seen in the appendix. Once we have calculated θ, we set $c := \min(\boldsymbol{P}_{ij}, \theta)$ and we update $\boldsymbol{P}_{ij} \leftarrow \boldsymbol{P}_{ij} - c$ and $\boldsymbol{f}, \boldsymbol{g}$ as follows

$$\boldsymbol{f} \leftarrow \nabla\phi^{-1}(c\boldsymbol{e}_i + \nabla\phi(\boldsymbol{f})) \text{ and } \boldsymbol{g} \leftarrow \nabla\varphi^{-1}(c\boldsymbol{e}_j + \nabla\varphi(\boldsymbol{g})).$$

In the forget step, if $\boldsymbol{P}_{ij} = 0$, then we forget the related constraint (i.e., remove it from the list of active constraints). Note that $\boldsymbol{P}$ is the dual variable and is the desired transportation plan. One feature of Project and Forget is in addition to calculating the primal variables, we also retain the desired dual variable $\boldsymbol{P}$, the transportation plan.

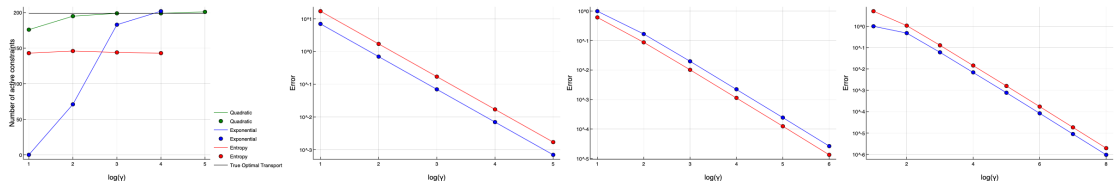
One of the reasons we chose to solve DROT with Project and Forget is for its convergence analysis and rate. Specifically, *Gilbert and Sonthalia* (2020a) show that Project and Forget has a linear rate of convergence and that the rate is at most $\frac{L}{L+\mu^2}$ for some $\mu \in (0, 1]$, where L is the number of active constraints. Corollary 6.8 gives us an estimate of the sparsity of our solutions $\boldsymbol{P}^*_{\phi,\varphi}$ and, hence, an estimate on the number of active constraints. (We note that there are comparatively few active constraints typically). Thus, giving us a reasonable problem specific upper bound on the rate of convergence.

Much of the previous discussion is theoretical in nature; we also performed extensive

comparison experiments to validate our choice of PROJECT AND FORGET. The experimental set up is as follows. We take two shifted Gaussian distributions with means ± 15 and variance 10. Then, split the interval $[-20, 20]$ into n points and create two discrete distributions by sampling the Gaussians on those n points. We use the squared Euclidean distances between the points as the cost function and quadratic regularization. This set up is a basic example of the optimal transport problem and is an important test example for algorithmic comparisons. We solve the dual version of DROT using Mosek, CPLEX, scipy's LBFGSB method, and projected gradient descent. We solve the primal version of DROT using PROJECT AND FORGET, Mosek, and CPLEX. To do a fair comparison, we ran all methods until they reached the same level of convergence (with a feasibility error of 10^{-8}). The convergence details can be seen in the appendix. From Table 6.1, we can see that if we use PROJECT AND FORGET, we can solve the problem for much larger values of n. With PROJECT AND FORGET, our formulation of optimal transport can be scaled up and solved for large number of data points.

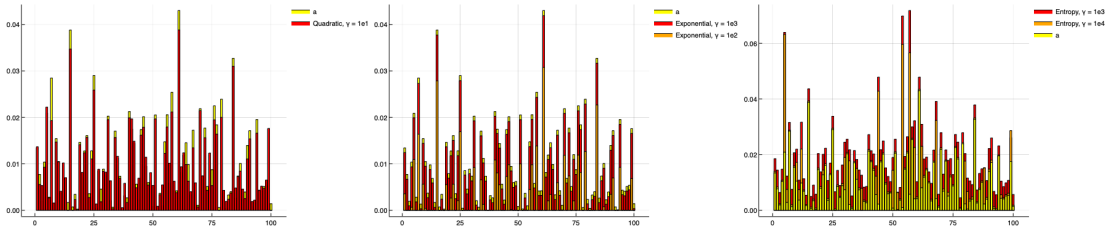
6.4 Experiments

In this section, we provide extensive experimental evidence to support the theoretical results presented in the previous section, to provide the intuition about dual regularized optimal transport (where theoretical analysis is unavailable), and to demonstrate that our new formulation of optimal transport is both different and useful (performing domain transfer tasks, including color transfer and digit classification).*



(i) Sparsity (ii) Entropy (iii) Quadratic (iv) Exponential

Figure 6.1: (i) Sparsity of the solutions for the different regularizers versus the regularization parameter; (ii–iv) Error $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi} \rangle|$ (blue line) and $OT(\boldsymbol{a}, \boldsymbol{b}) - DROT(\boldsymbol{a}, \boldsymbol{b})$ (red line) versus γ for the three different regularizers.



(i) Quadratic (ii) Exponential (iii) Entropy

Figure 6.2: Graphs showing the mass creation and destruction for the different regularizers. The yellow bars represent the true marginal distribution.

6.4.1 Verifying theoretical properties

The first solution property that we verify experimentally is the sparsity of the transport plan. To generate a problem instance for verification, we uniformly sample two distributions $\boldsymbol{a}, \boldsymbol{b}$ from Δ^{100}. Then we sample $\boldsymbol{C}_{ij}$ independently and uniformly from $[0, 1]$. As we can see from Figure 6.1i, in all cases, we find solutions that are sparser than the true optimal transport plan. As the regularization parameter γ increases, the size of the support of our transport plans increases until we reach the true support size. For the entropic and exponential regularizers, for $\gamma = 10^5$, the optimization had not converged so we do not plot those results.

Next, we evaluate how well our objective functions approximate the Wasserstein distance (the objective of Problem 6.0.1) and how well our transport plans approximate the true plans. We construct a simple problem instance (as our previous instance

*All code and data can be found at https://www.dropbox.com/sh/ge52lo3e5vs4dqm/AAC2oFmuS8oLx1ZzrfMThDxua?dl=0

is difficult to calculate for large γ) consisting of two Gaussian distributions with means ± 15 and variance 10. The cost matrix $\boldsymbol{C}$ is given by $\boldsymbol{C}_{ij} = 1$. Then we plot $OT(\boldsymbol{a}, \boldsymbol{b}) - DROT(\boldsymbol{a}, \boldsymbol{b})$ (red line) and $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi} \rangle|$ (blue line) versus γ. Furthermore, the gap between the two lines is $\phi^*(\gamma * (\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m)/\gamma + \varphi^*(\gamma * (\boldsymbol{a} - \boldsymbol{P}^T\mathbf{1}_n)/\gamma$. From our theoretical analysis, we know that all of these quantities should be $O(\gamma^{-1})$. From the plots it is evident that $OT(\boldsymbol{a}, \boldsymbol{b}) - DROT(\boldsymbol{a}, \boldsymbol{b})$ (red line) and $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi} \rangle|$ (blue line) decrease linearly. Finally, since the plots are log-log plot, the plots show that $\phi^*(\gamma * (\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m))/\gamma + \varphi^*(\gamma * (\boldsymbol{a} - \boldsymbol{P}^T\mathbf{1}_n))/\gamma$ also decrease linearly with respect to γ. Thus, the experiments suggest that the theoretical error rate is tight. That is the error is $\Theta(\gamma^{-1})$.

Finally, we test the intuition sketched in our theoretical analysis as to when mass is created versus destroyed. Specifically that, entropy regularization creates mass, the exponential regularization destroys mass, and the quadratically regularized problem does both. To verify this, we uniformly sample two distributions $\boldsymbol{a}, \boldsymbol{b}$ from Δ^{100} and sample $\boldsymbol{C}_{ij}$ independently and uniformly from $[0, 1]$. Then we compute the transport plan and marginals for all three different regularizers for a variety of different values of γ. Figure 6.2 shows that our intuition matches exactly what occurs in practice. The quadratic regularizer both creates and destroys mass; that is, sometimes the yellow bars (bar chart for $\boldsymbol{a}$) are bigger and sometimes the yellow bars are smaller. The exponential regularizer only destroys mass; i.e., the yellow bars are always bigger. Finally, the entropic regularizer only creates mass; i.e., the yellow bars are always smaller. In each case, we see that as γ gets bigger, the marginals of the transport plan better approximate the true marginals.

6.4.2 Domain Transfer

In this section, we explore how our new formulation performs on the task of domain transfer. We also investigate our intuition as to how the different regularizers affect

Figure 6.3: Images produced by doing color transfer using different regularizers (Exponential, Quadratic, Entropy) for DROT and images produced by doing color transfer using other formulations of optimal transport.

the results. The goal of this section is not to present state of the art results for domain transfer, but to demonstrate that creating versus destroying mass gives us different results. And so, being able to decide whether mass is created or destroyed is a desirable attribute in a problem formulation and algorithmic method.

We compare our formulation DROT against other formulations. Specifically,

we compare against standard optimal transport OT, entropic regularization of the primal ROT, and UOT with entropic regularization of the primal with KL divergence controlling the deviation from marginals. All of our DROT formulations are solved using PROJECT AND FORGET. The other formulations are solved using the python optimal transport library with the following algorithms: we solve ROT using the algorithm in *Cuturi* (2013), OT using the algorithm in *Bonneel et al.* (2011), and UOT using the algorithm in *Chizat et al.* (2016b).

For all domain transfer problems we use the squared Euclidean distance as the cost function. Thus, once we have computed our transport plans $\boldsymbol{P}$ (obtained from solving any of the versions of optimal transport), we compute the barycentric projection map to transfer one data set into the domain of the other data set. That is, because we use the squared Euclidean distance as the cost, if $\boldsymbol{a}, \boldsymbol{b}$ are the two data sets, the transport of $\boldsymbol{a}$ to the domain of $\boldsymbol{b}$, denoted $\hat{\boldsymbol{a}}$ is given by, $\hat{\boldsymbol{a}}_i = \frac{\sum_{j=1}^{n} \boldsymbol{P}_{ij} \boldsymbol{b}_j}{\sum_{j=1}^{n} \boldsymbol{P}_{ij}}$.

6.4.2.1 Color Transfer.

Color transfer consists of the first domain transfer experiment. In these experiments, we use the same setup as *Blondel et al.* (2018b). For each picture, we first perform k means to cluster the three dimensional pixels in each image, generating k color centers for each image. These centers are the point masses for the two distributions. The weight of each center is proportional to the number of points assigned to that cluster and the cost matrix is given by the Euclidean squared distance between the color centers. We want to demonstrate two things with this experiment:

1. DROT results in good quality images that look different. The other formulations of OT, which produce similar pictures, as seen in Figure 6.3.
2. The regularization parameter γ, when used with the quadratic regularizer, destroys mass and this is evident in the images but when used with the entropic regularizer, the way in which mass is created is not reflected in the images

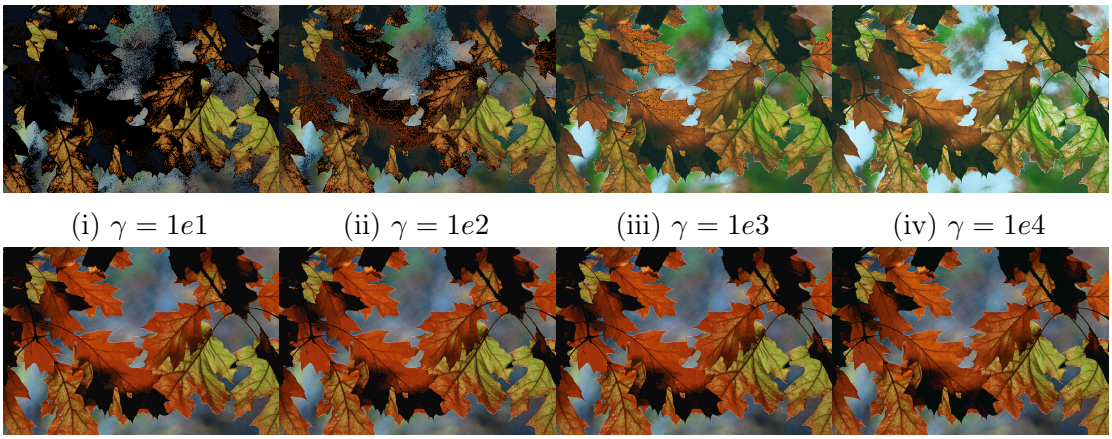
(although it is in a toy example).

For the first demonstration, we can see the performance of the different regularizers in Figure 6.3. If we use the entropic regularizer, then the transferred image is more faithful to the original color distribution. Additionally, we see that entropic regularized images are cleaner and have fewer artifacts.

For the second demonstration, we see from Figure 6.4, that when γ is small and we use the quadratic regularizer, we tend to destroy the mass; i.e., the images are corrupted. Figure 6.4, however, shows that for the entropic regularizer, for all values of γ, the images look identical. We argue that this phenomenon occurs as a result of two different phenomena. First, note that the entries in the cost matrix are less than 1. Because of the entropic regularizer, the critical point of the objective function always has the entries greater than 1. Thus, the solution to the entropic regularized problem will always be on the boundary, regardless of the value of γ. That is, mass transport always occurs. This does not, however, explain why the images look identical. We conjecture a second phenomenon is at play: when we have a convex cost function, we conjecture that, changing γ results in creating mass simply by shifting the distribution upwards (as shown in Figure 6.5). That is, the transport plan maintains the shape of the distribution and just shifts it up. For images, shifting the distribution by a bounded amount does not impact the appearance of the color transfer and the images look similar.

6.4.2.2 MNIST, USPS classification.

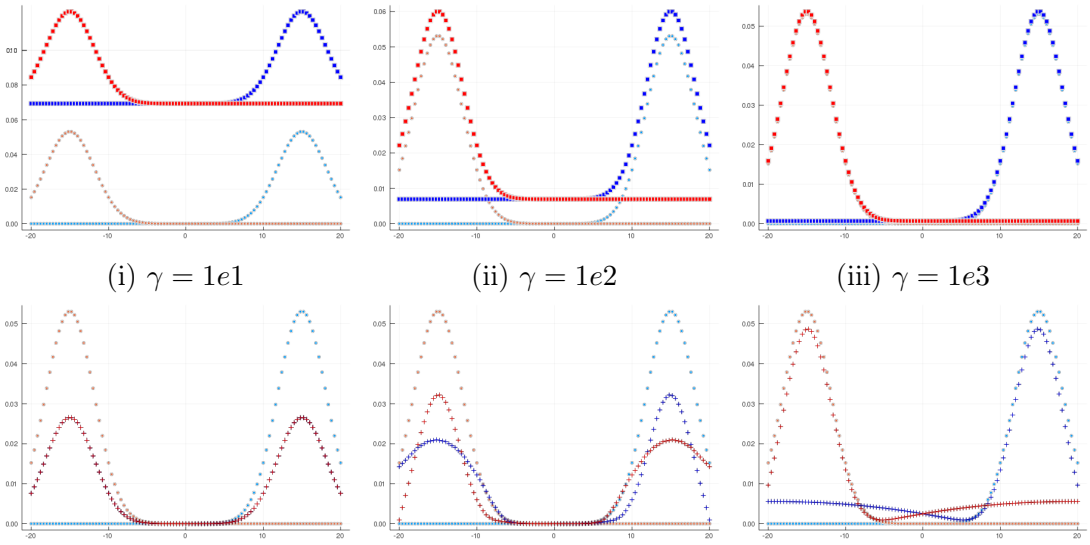
Finally, we use domain adaptation for classification. To test the performance of DROT, we transport between the MNIST training data set and USPS training data sets. First, we pad the USPS images with zeros so that they are are the same size as the MNIST images and the use the squared Euclidean distance as the metric between the two data sets. We then transport the USPS training set to the MNIST domain.



(i) $\gamma = 1e1$ (ii) $\gamma = 1e2$ (iii) $\gamma = 1e3$ (iv) $\gamma = 1e4$

(v) $\gamma = 1e-1$ (vi) $\gamma = 1e0$ (vii) $\gamma = 1e1$ (viii) $\gamma = 1e4$

Figure 6.4: Images produced by doing color transfer for different values of γ. The top row is for the quadratic regularizer, and the bottom row is for the entropic regularizer.



(i) $\gamma = 1e1$ (ii) $\gamma = 1e2$ (iii) $\gamma = 1e3$

(iv) Quadratic, $\gamma = 1e1$ (v) Quadratic, $\gamma = 1e3$ (vi) Quadratic, $\gamma = 1e5$

Figure 6.5: Graphs showing that the entropic regularizer maintains the distribution shape and the quadratic regularizer creates and destroys mass. We used the squared Euclidean distance as the cost function and performed transport from the red distribution to the blue distribution.

First, let us examine the appearance of the transported digits. Figure 6.6 shows what the first 4 digits in the USPS data set look like after they have been transported to the MNIST domain. We can see again that the entropic regularized transport is the most faithful to the original image and has the cleanest new digits. We then use the

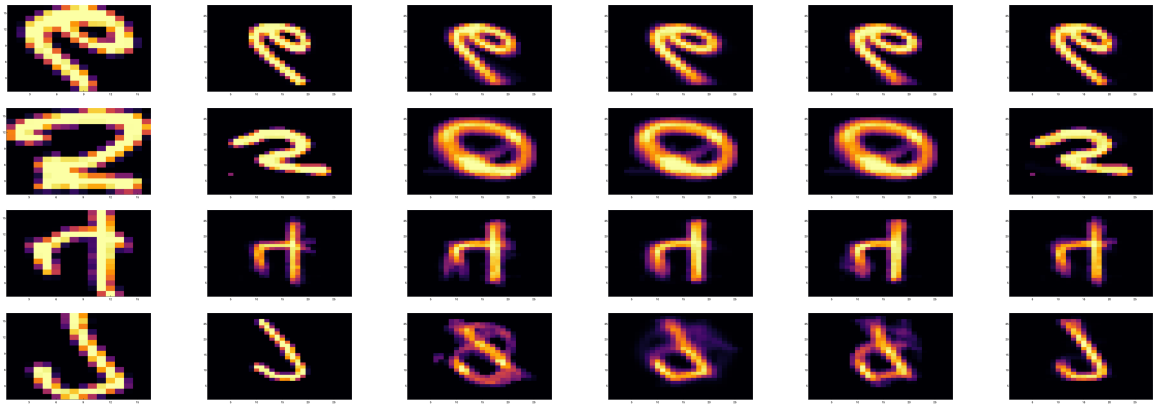


Figure 6.6: Images of the first four digits in the USPS dataset, when transported using to the MNIST domain using various optimal transport problems. DE/DQ refers to entropic/quadratic regularized version of DROT.

Problem	Trained on MNIST	Trained on USPS
Dual Entropy	76.46%	62.54%
Dual Quadratic	65.75%	63.79%
OT	62.04%	65.32%
UOT	75.44%	66.16%
ROT	66.99%	63.87%

Table 6.2: Accuracy using a 1 nearest neighbor classifier after transporting the USPS dataset to the MNIST domain.

transported USPS digits for classification. We try to classify the MNIST digits using a classifier trained on the transported USPS dataset and to classify the transported USPS digits using a classifier trained on the MNIST dataset. Table 6.2 shows that the entropic regularized version performs well.

CHAPTER VII

How can Classical Multidimensional Scaling go Wrong?

7.1 Introduction

Multidimensional scaling (MDS) refers to a class of techniques for analyzing pairwise dissimilarities between arbitrary objects. The goal is to find an embedding of points in a (usually) low dimensional space where the pairwise distances between these points represent, as best as possible, the given pairwise distances between objects *Carroll and Arabie* (1998); *Borg and Groenen* (2005). Apart from the general usefulness of dimensionality reduction, MDS has been used in a wide variety of applications including data visualization, data preprocessing, network analysis, bioinformatics, and data exploration. Due to its long history and being well studied, MDS has many variations such as nonmetric MDS *Shepard* (1962a,b), multi-way MDS *Kroonenberg* (2008), multi-view MDS *Bai et al.* (2017), confirmatory or constrained MDS *Heiser and Meulman* (1983), etc. (See *France and Carroll* (2010); *Cox and Cox* (2008) for surveys).

One of the oldest and most popular methods for solving MDS comes from Torgerson *Torgerson* (1952) who introduced the classical MDS algorithm (cMDS). This algorithm first centers the squares of the given distance matrix and then uses its spectral

decomposition to extract the low dimensional embedding. Part of its popularity is because this decomposition is very fast and can scale to large matrices. However, the simplicity and speed of cMDS comes at the cost of being a heuristic; it does not solve the true MDS problem (compare Equation 7.1 to 7.2). Furthermore, solving the true MDS problem is computationally expensive. So naturally, a practitioner would want to know when they can get away with using cMDS and hence a few questions arise: how bad can the cMDS solution be compared to the true solution? under what conditions does cMDS perform well/poorly? does the embedding dimension matter? what if the given (dis)similarity matrix does not come from points in a Euclidean space? how robust is cMDS (compared to the true solution) to noisy data?

The observation that cMDS is a heuristic or that it can perform poorly in the presence of noisy data is not new. In addition to the variations of MDS mentioned previously, several authors *Cayton and Dasgupta* (2006a); *Mandanas and Kotropoulos* (2016); *Forero and Giannakis* (2012) have proposed even more variations to specifically address robustness with cMDS. However, despite the widespread use of cMDS and the awareness of and variations to remedy non-robustness, there is surprisingly little analysis regarding the above questions. *Cayton and Dasgupta* (2006a) only give heuristic arguments that show cMDS is not robust for non-Euclidean distance matrices. Also mentioned in that paper is the vast body of research on bounding the distortion of embeddings between various spaces. But this work is motivated more by geometric and combinatorial concerns and is not related to the above questions which come from a data analysis perspective.

In this paper we establish theoretical results and give experimental evidence regarding these questions about cMDS.

7.1.1 Problem statements and contributions

Given a distance matrix D whose entries are squared, the MDS problem solves

$$D_t := \underset{D'=EDM(X),X\in\mathbb{R}^{r\times n}}{\arg\min} \|D' - D\|_F^2. \tag{7.1}$$

where given an embedding X, $EDM(X)$ is the corresponding Euclidean distance matrix. That is,

$$EDM(X)_{ij} = \|X_i - X_j\|_F^2,$$

where X_i, X_j are the ith and jth columns of X.

As mentioned, cMDS does not solve 7.1 but instead solves

$$X_{\text{cmds}} := \underset{X\in\mathbb{R}^{r\times n}}{\arg\min} \left\| X^T X - \left(\frac{-VDV}{2}\right)\right\|, \tag{7.2}$$

where V is the centering matrix given by $V := I - \frac{1}{n}J$. Here I is the identity matrix and J is the matrix of all ones.

So we see that $D_{\text{cmds}} := EDM(X_{\text{cmds}})$ is not the solution to the true MDS problem but is only an approximation to it. Thus, the question that we are interested in as follows. Let D_t be the solution to the problem in Equation 7.1, then we are interested in understanding the quantity $err := \|D_t - D_{\text{cmds}}\|$. In particular, can we predict when err is small without having to compute D_t? Doing so provide practitioners with multiple advantages.

1. If err is guaranteed to be small, then we can use the cMDS algorithm without having to worry about loss in quality of the solution.
2. If err is big, we can now make an informed decision about whether benefits of the speed of cMDS algorithm versus the quality of the solution.
3. Understanding when err is big helps us design algorithms to approximate the solution to MDS problem that perform better when cMDS fails.

4. Understanding *err* is also the first step in rigorously quantifying the robustness of the cMDS algorithm.

The main contribution of our paper are as follows.

1. We decompose the error in Equation 7.2 into three terms that depend on the eigenvalues of a matrix obtained from D. Using this analysis, we show that there is a term that tells us that as we increase the dimension that we embed into, eventually, the error starts increasing.
2. Using this analysis, we provide an efficiently computable algorithm that returns a matrix D_l such that if D_t is the true closest EDM, then $\|D_l - D\|_F \leq \|D_t - D\|_F$, and empirically we see that $\|D_l - D_t\|_F \leq \|D_{\text{cmds}} - D_t\|_F$.
3. While D_l is not metric, when given as input to cMDS instead of D, it results in solutions that are empirically better than the cMDS solution. This modified procedure results in a more natural decreasing of the error as the dimension increases.

7.2 Preliminaries and Background

In this section, we lay out the preliminary definitions and necessary key structural characterizations of Euclidean distance matrices.

7.2.1 cMDS algorithm

For completeness, we include the classical multidimensional scaling algorithm in Algorithm 18.

Algorithm 18 Classical Multidimensional Scaling.

1: **function** CMDS(D, r)
2: $X = -V * D * V/2$
3: Compute $\mu_1 \geq \ldots \geq \mu_r > 0$, U as the eigenvalues and eigenvectors of X
4: return $U * diag(\sqrt{\mu_1}, \ldots, \sqrt{\mu_r}, 0, \ldots, 0)$

7.2.2 EDM Matrices

Definition 7.1. $D \in \mathbb{R}^{n\times n}$ is a Euclidean Distance Matrix (EDM) if and only if there exists a $d \in \mathbb{N}$ such that there are points $x_1, \ldots, x_n \in \mathbb{R}^d$ with

$$D_{ij} = \|x_i - x_j\|_2^2.$$

Note that unlike other distance matrices, an EDM consists of the squares of the (Euclidean) distances between the points.

We give three important structural characterizations of the cone of EDM matrices.

1. *Gower* (1985a); *Schoenberg* (1935) show that a symmetric matrix D is an EDM if only if

   $$F := -(I - \mathbf{1}s^T)D(I - s\mathbf{1}^T)$$

   is a positive semi-definite matrix for all s such that $\mathbf{1}^T s = 1$ and $Ds \neq 0$.

2. *Schoenberg* (1938b) showed that D is an EDM if and if $exp(-\lambda D)$ is a PSD matrix with 1s along the diagonal for all $\lambda > 0$. Note here exp is element wise exponentiation of the matrix.

3. Another characterization is given by *Hayden and Wells* (1988) in which D is an EDM if and only if D is symmetric, has 0s on the diagonal, and $\hat{D}$ is negative semi-definite, where $\hat{D}$ is defined as follows

   $$QDQ = \begin{bmatrix} \hat{D} & f \\ f^T & \xi \end{bmatrix}. \tag{7.3}$$

   Here f is a vector and

   $$Q = I - \frac{2}{v^T v} vv^T \text{ for } v = [1, \ldots, 1, 1 + \sqrt{n}]^T. \tag{7.4}$$

Note Q is a unitary matrix.

In addition to characterizations of the EDM cone, we are also interested in the dimension of the EDM.

Definition 7.2. Given an EDM D, the dimensionality of D is the smallest dimension d, such that there exist points $x_1, \ldots, x_n \in \mathbb{R}^d$ with $D_{ij} = \|x_i - x_j\|_2^2$.

Let $\mathcal{E}(r)$ be the set of EDM matrices whose dimensionality is at most r.

Using *Hayden and Wells* (1988)'s characterization of EDMs, *Qi and Yuan* (2014c) show $D \in \mathcal{E}(r)$ if and only if D is symmetric, hollow (i.e., 0s along the main diagonal), and $\hat{D}$ in Equation 7.3 is negative semi-definite with rank at most r.

This characterization provides the following insight and potential algorithm for embedding in $\mathbb{R}^r$ points whose distances are captured by a general distance matrix D. First, we conjugate D by Q to obtain the smaller matrix $\hat{D}$ given in Equation 7.3. Then, we compute the spectral decomposition of the matrix $\hat{D}$ and restrict to rank r. In a sense, the cMDS algorithm attempts to do this; however, it is subtlely different and, as a result, not the optimal way to reduce the dimension. To make more formal this analysis in the next section, let us define $\lambda_1 \leq \ldots \leq \lambda_{n-1}$ to be the eigenvalues of $\hat{D}$.

7.2.3 Conjugation matrices: Q and V

Conjugating distance matrices either by Q (in Equation 7.4) or by the centering matrix V is an important component of understanding both EDMs and the cMDS algorithm. We provide some notation and definitions.

Definition 7.3. Given any symmetric matrix $A \in \mathbb{R}^{n \times n}$, let us define $\hat{A} \in R^{n-1 \times n-1}$, $f(A) \in \mathbb{R}^{n-1}$, and $\xi(A) \in \mathbb{R}$ as follows

$$QAQ = \begin{bmatrix} \hat{A} & f(A) \\ f(A)^T & \xi(A) \end{bmatrix}.$$

One important connection between Q and V is given in *Qi and Yuan* (2014c). Specifically for any symmetric matrix A, we have that

$$VAV = Q \begin{bmatrix} \hat{A} & 0 \\ 0 & 0 \end{bmatrix} Q.$$

Here $\hat{A}$ is the matrix given by Definition 7.3.

7.3 Theoretical Results

Throughout this section we fix the following notation. Let D be a dissimilarity matrix where the entries are squared. Let $\lambda_1 \leq \ldots \leq \lambda_{n-1}$ be the eigenvalues of $\hat{D}$ and U be the eigenvectors. Let $\vec{\lambda}$ be an n-dimensional vector where $\vec{\lambda}_i = \lambda_i \mathbb{1}_{\lambda_i > 0 \text{ or } i > r}$ for $i = 1, \ldots, n-1$ and $\vec{\lambda}_n = 0$. Let $S = Q * \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix}$. Let D_t and D_{cmds} be the true MDS and the classical MDS solutions respectively. Let

$$C_1 = \sum_{i=1}^{n-1} \lambda_i^2 \mathbb{1}_{i>r \text{ or } \lambda_i > 0}$$
$$C_2 = -\sum_{i=1}^{n-1} \lambda_i \mathbb{1}_{\lambda_i > 0 \text{ or } i > r}$$
$$C_3 = \frac{n\|(S \circ S)\vec{\lambda}\|_F^2 - C_2^2}{2}$$

Theorem 7.4. Let r be the dimension we want to embed into and let D_l be the matrix returned by Algorithm 19. Then,

1. $\|D_t - D\|_F^2 \geq \|D_l - D\|_F^2 \geq C_1 + \frac{C_2^2}{r+1}$
2. $\|D_{cmds} - D\|_F^2 = C_1 + C_2^2 + C_3$

Algorithm 19 Lower Bound Algorithm.

function LOWER(D, r)
  Compute $\hat{D}$, $f(D)$ and $\xi(D)$.
  Compute $\lambda_1 \leq \ldots \leq \lambda_{n-1}$, U as the eigenvalues and eigenvectors of $\hat{D}$
  Let $s = 0$.
  for $i = 1 \ldots n-1$ **do**
    if $\lambda_i > 0$ or $i > r$ **then**
      $s \mathrel{+}= \lambda_i$
      Set λ_i to 0
  $E :=$ number of eigenvalues not equal to 0
  $sub := s/(E+1)$
  $\xi(D) \mathrel{+}= sub$, $s \mathrel{-}= sub$
  for $i = 1 \ldots n-1$ **do**
    if $\lambda_{n-i}! = 0$ **then**
      if $\lambda_{n-i} + sub \leq 0$ **then**
        $\lambda_{n-i} \mathrel{+}= sub$, $E \mathrel{-}= 1$, $s \mathrel{-}= sub$
      else
        $E \mathrel{-}= 1$, $s \mathrel{+}= \lambda_{n-1}$, $sub = s/E$
        $\lambda_{n-1} = 0$
  Let $\hat{D} = U * diag(\lambda_1, \ldots, \lambda_{n-1}) * U^T$
  return $Q * \begin{bmatrix} \hat{D} & f(D) \\ f(D)^T & \xi(D) \end{bmatrix} * Q$

7.3.1 Lower bound for $\|D_t - D\|_F^2$

In this section, we are interested in finding a lower bound on $\|D_t - D\|_F$. To do this, we will construct a matrix M where $\|D_t - D\|_F \geq \|M - D\|_F$ and hence this construction provides a new approximation algorithm for the problem. At first, we treat M as being an EDM matrix in order to decompose $\|M - D\|_F$. But we will then relax the EDM conditions on M to obtain the lower bound which we finally denote as D_l.

Let Q be the unitary matrix in Equation 7.4. Then we know that D is an EDM if and only if D is symmetric, is hollow and $\hat{D}$ is negative semi definite. Then conjugating by Q, due to the invariance of the Frobenius norm, we get that

$$\|M - D\|_F = \|QMQ - QDQ\|_F.$$

Now let $\begin{bmatrix} \hat{D} & f(D) \\ f^T(D) & \xi(D) \end{bmatrix} = QDQ$ and let $\hat{D} = U\Lambda U^T$ be the eigen-decomposition. Then consider the matrix

$$R := \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix}.$$

Then since R is a unitary matrix, we again have that

$$\|M - D\|_F = \left\| R^T QMQR - \begin{bmatrix} \Lambda & U^T f(D) \\ f^T(D)U & \xi(D) \end{bmatrix} \right\|_F$$

Now consider

$$QMQ = \begin{bmatrix} \hat{M} & f(M) \\ f^T(M) & \xi(M) \end{bmatrix}$$

If we enforce that M is an EDM, then M needs to be symmetric, hollow, and $\hat{M}$ needs to be negative semi-definite. Let us now conjugate by R to get

$$R^T QMQR = \begin{bmatrix} U^T \hat{M} U & U^T f(M) \\ f^T(M)U & \xi(M) \end{bmatrix}$$

Let $C := U^T \hat{M} U$. Then we have that

$$\begin{aligned} \|M - D\|_F^2 = \sum_{i \neq j} c_{ij}^2 + \sum_{i=1}^{n} (c_{ii} - \lambda_i)^2 \\ + 2\|f(M) - f(D)\|_2^2 + (\xi(M) - \xi(D))^2 \end{aligned}$$

In this case, since $\hat{M}$ was negative semi-definite and U is unitary, we have that $U^T \hat{M} U$ is still negative semi-definite. If we relax the hollow condition on M, then we can set $f(M) = f(D)$. Similarly, we know that we must have $c_{ii} \leq 0$ (C is negative semi-definite). Thus for all $\lambda_i > 0$, the closest M can be to D is by setting the

corresponding $c_{ii} = 0$. Hence if we relax the condition that M must be hollow, then

$$\|D_t - D\|_F^2 \geq \|M - D\|_F^2 \geq \sum_{i=1}^{n-1} \lambda_i^2 \mathbb{1}_{\lambda_i > 0} =: C_1$$

While this is already a lower bound, we can do better. Instead of completely getting rid of the hollow condition, we instead relax it so that M has trace 0. In this case, that is equivalent to $R^T Q M Q R$ having trace 0. Currently, by setting $(U^T \hat{M} U)_{ii}$ to be $min(\lambda_i, 0)$, we have that the trace of M is given by

$$C_2 = -\sum_{i=1}^{n-1} \lambda_i \mathbb{1}_{\lambda_i > 0}.$$

First note that C_2 is negative. Thus, to make the trace 0, we need to increase the remaining eigenvalues and $\xi(M)$. To find the minimum quadratic solution, we would need to increase all of the eigenvalues equally. If N is the number of negative eigenvalues of $\hat{D}$. Thus we have

$$\|D_t - D\|_F^2 \geq \|M - D\|_F^2 \geq \sum_{\lambda_i > 0} \lambda_i^2 + \frac{\left(\sum_{\lambda_i > 0} \lambda_i\right)^2}{N + 1}.$$

So far, we have not had any constraint on the dimensionality of the EDM. Let r be the dimension constraint, that is, we want $M \in \mathcal{E}(r)$. We have that

$$\|M - D\|_F^2 \geq C_1 + \frac{C_2^2}{r + 1}. \tag{7.5}$$

However, we may not be able to increase all the eigenvalues equally. This is because, all of the remaining eigenvalues are negative. It could be possible that if we increase an eigenvalue by $-C_2/(r+1)$ then it might become positive. Hence, we want to increase such eigenvalues by as much as we can and have the rest compensate. We do this on lines 14 through 23 of Algorithm 19.

Theorem 7.5. Let $\kappa(r)$ be the space of symmetric, trace 0 matrices A, such that $\hat{A}$ is negative semi definite of rank at most r. Then we have that Algorithm 19 returns the matrix D_l such that

$$D_l = \underset{A \in \kappa(r)}{\arg\min} \|A - D\|_F^2.$$

Note that Theorem 7.5 and Equation 7.5 prove part 1 of Theorem 7.4.

7.3.2 Expression for $\|D_{\text{cmds}} - D\|_F^2$

In this section we will go through the proof for Theorem 7.4 part 2. The idea is to decompose

$$\begin{aligned} \|D_{cmds} - D\|_F^2 &= \|QD_{cmds}Q - QDQ\|_F^2 \\ &= \|\hat{D} - \hat{D}_{cmds}\|_F^2 \\ &+ 2\|f(D) - f(D_{cmds})\|_F^2 \\ &+ (\xi(D) - \xi(D_{cmds}))^2, \end{aligned}$$

which follows from Definition 7.3. We now relate each of these three terms to C_1, C_2, and C_3.

In the following discussion, let $Y_r := {X_{cmds}}^T X_{cmds}$ and recall that X_{cmds} is the solution to the classical MDS problem given in Equation 7.2.

Lemma 7.6. If G is a positive semi-definite gram matrix, then $-\frac{1}{2}V\text{EDM}(G)V = Q \begin{bmatrix} \hat{G} & 0 \\ 0 & 0 \end{bmatrix} Q$

Proof. First, note that

$$\text{EDM}(G) = diag(G)\mathbf{1}^T - 2G + \mathbf{1}diag(G)^T$$

Then since $diag(G)\mathbf{1}^T$ is a rank 1 matrix where every column is the same, when

we center it using V, we see that $V diag(G)\mathbf{1}^T V = 0$. Similarly, we have that $V\mathbf{1}diag(G)^T V = 0$. Thus, we have that

$$V\text{EDM}(G)V = -2VGV$$

Then dividing by -2 and using the relation between V and Q we get the needed result. □

Lemma 7.7. The actual value obtained by X_{cmds} in Equation 7.2 is $\frac{C_1}{4}$. Specifically, if let $Y_r := X_{cmds}^T X_{cmds}$, then we have that

$$4\|Y_r - (-VDV)/2\|_F^2 = \sum_{i=1}^{n-1} \lambda_i^2 \mathbb{1}_{i>r \text{ or } \lambda_i>0} =: C_1.$$

Proof. To see this, let B be any matrix, then we first note the following

$$\begin{aligned}\|B - (-VDV)/2\|_F^2 &= \left\| B - Q \begin{bmatrix} -\hat{D}/2 & 0 \\ 0 & 0 \end{bmatrix} Q \right\|_F^2 \\ &= \left\| QBQ - \begin{bmatrix} -\hat{D}/2 & 0 \\ 0 & 0 \end{bmatrix} \right\|_F^2\end{aligned}$$

Here the first equality is true, due to the relationship between Q and V, and the second is true, due to the unitary invariance of the Frobenius norm. For the classical MDS algorithm, we minimize the term on the left hand side with the constraint that B is positive semi-definite with rank at most r. Now conjugating a positive semi-definite matrix by unitary matrix results in a positive semi-definite matrix and does not change the rank. Thus, we can solve the MDS problem, by instead minimizing the last term with the restriction that QBQ is positive semi-definite with rank at most r. We know the solution to this is to keep the r biggest positive eigenvalues of $-\hat{D}/2$. Then since

$\lambda_1 \leq \ldots \leq \lambda_{n-1}$, we have that

$$\|Y_r - (-VDV)/2\|_F^2 = \frac{1}{4}\sum_{i=1}^{n-1} \lambda_i^2 \mathbb{1}_{i>r \text{ or } \lambda_i>0} = \frac{C_1}{4}.$$

□

Lemma 7.8. $-\frac{1}{2}\hat{D}_{\text{cmds}} = \hat{Y}_r$.

Proof. We see this via the following calculation.

$$\begin{aligned} Q\begin{bmatrix} -\hat{D}_{\text{cmds}}/2 & 0 \\ 0 & 0 \end{bmatrix} Q &= -VD_{\text{cmds}}V/2 \\ &= -V\text{EDM}(Y_r)V/2 \\ &= Q\begin{bmatrix} \hat{Y}_r & 0 \\ 0 & 0 \end{bmatrix} Q \end{aligned}$$

The first equality is due to the relationship between Q and V expressed in Section 7.2.3. The second is because D_{cmds}, by definition, is given by $EDM(Y_r)$. Finally, the last equality is due to Lemma 7.6. □

Lemma 7.9. If $\text{Tr}(D) = 0$, then

$$(\xi(D) - \xi(D_{\text{cmds}}))^2 = \left(\sum_{i=1}^{n-1} \lambda_i \mathbb{1}_{i>r \text{ or } \lambda_i>0}\right)^2 =: C_2^2.$$

Proof. Since D_{cmds} is an EDM, we have that $\text{Tr}(D_{\text{cmds}}) = 0$. Thus, we have that $\text{Tr}(D - D_{\text{cmds}}) = 0$. Finally, since $\text{Tr}(QAQ) = \text{Tr}(A)$ for any matrix A, we have that

$$0 = \text{Tr}(D - D_{\text{cmds}}) = \text{Tr}(\hat{D} - \hat{D}_{\text{cmds}}) + \xi(D) - \xi(D_{\text{cmds}})$$

Then we know from the construction in Lemma 2, that

$$\mathrm{Tr}(\hat{D} - \hat{D}_{\mathrm{cmds}}) = \sum_{i=1}^{n-1} \lambda_i \mathbb{1}_{i>r \text{ or } \lambda_i>0}.$$

Rearranging and solving gives the needed quantity. □

Lemma 7.10.

$$2\|f(D) - f(D_{cmds})\|_F^2 = \frac{n\|(S \circ S)\vec{\lambda}\|_F^2 - C_2^2}{2} =: C_3.$$

Now we have all the pieces that we need to prove Theorem 7.4 part 2. To see that, write $\|D - D_{cmds}\|_F^2$ as $\|QDQ - QD_{cmds}Q\|_F^2$. Then the difference can be broken down into $\|\hat{D} - \hat{D}_{cmds}\|_F^2$ which is given by Lemma 7.7 and 7.8, plus $(\xi(D) - \xi(D_{cmds}))^2$ which is given by Lemma 7.9 and finally the $2\|f(D) - f(D_{cmds})\|_F^2$ term which is given by Lemma 7.10.

While C_1 and C_2 are intuitive enough to think about C_3 is more obtuse. To simplify it, we consider the following. If δ is an entry of a random n by n unitary matrix, then as n goes to infinity the distribution for $\delta\sqrt{n}$ converges to $\mathcal{N}(0, 1)$ and that the total variation between the two distributions is bounded above by $8/(n-2)$ *Easton* (1989); *Diaconis and Freedman* (1987). Thus, from this we can assume that the variance of an entry of a random n by n orthogonal matrix is about $1/n$. Thus, this lets us heuristically think about $S \circ S \approx \frac{1}{n} 1 1^T$. Then we have that

$$n\|(S \circ S)\vec{\lambda}\|_F^2 \approx \frac{n}{n^2}\|1 1^T \vec{\lambda}\|_F^2 = \frac{n}{n^2}\|1 C_2\|_F^2 = C_2^2.$$

Thus, we can think of C_3 as roughly constant or at worst being $O(C_2^2)$.

7.3.3 Error Analysis for cMDS

Now that we have decomposed the error for the cMDS algorithm, we can analyze this to better understand the failure modes. Through out this discussion let r be the dimension we are embedding into. First, we compare the cMDS error against our lower bound for $\|D_t - D\|_F^2$. Note our lower bound is given by $C_1 + C_2^2/(r+1)$. On the other hand the MDS error is equal to $C_1 + C_2^2 + C_3$. Taking the difference, we get

$$C_3 + \frac{r}{r+1} C_2^2.$$

Using our heuristic from before, we expect that C_3 is a constant or at worst case $O(C_2^2)$. Thus, this tells us that C_2^2 is the term that we should focus our attention towards. To better understand this, let us recall a couple of facts. First, recall that $\lambda_1 \leq \ldots \leq \lambda_{n-1}$ are the eigenvalues of $\hat{D}$ and that

$$C_2 = \sum_{i=1}^{n-1} \lambda_i \mathbb{1}_{i>r \text{ or} \lambda_i>0}.$$

As we increase the embedding dimension, we take more positive eigenvalues of $-VDV/2$. Due to our relation between V and Q, we know that if eigenvalues of $-VDV/2$ are 0 along with $\mu_1 \geq \ldots \geq \mu_{n-1}$, where $\mu_i = -\lambda_i/2$. Thus, as as we increase the embedding dimension, we have that C_2 becomes bigger.

If we now suppose that a significant number of the μs are negative, that is, a significant number of the λs are positive then we could run into an issue. Specifically, as the embedding dimension increases, we want our error to decrease. However, we just said that C_2 gets bigger as the embedding dimension increases. The only way C_2 becoming bigger implies that C_2^2 decreases, is if C_2 is negative. However, if a significant number of the λs are positive, we have that C_2 will eventually be positive and then after that dimension, the error will increase as we increase the dimension!

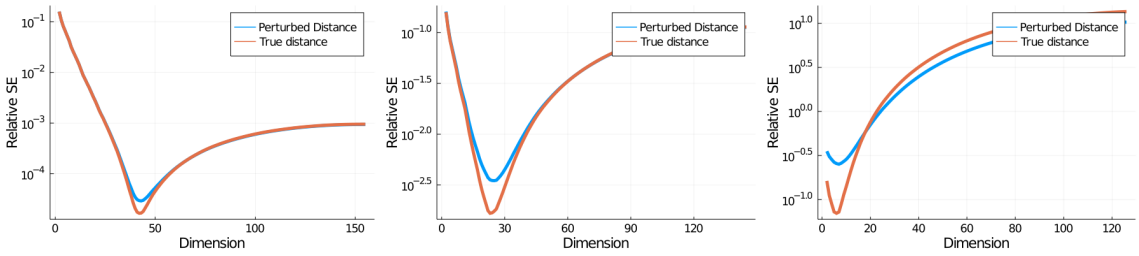
On the other hand, if we give the matrix D_l form Algorithm 19 as input to cMDS, then we see that the error terms C_1 and C_2 vanish. Specifically, if D_{lcmds} is the metric obtained from running cMDS on D_l, then we have that

$$\|D_{lcmds} - D_l\|_F^2 = 2\|f(D_{lcmds}) - f(D_l)\|_F^2.$$

Unfortunately, since D_l is not hollow, we cannot use Lemma 7.10 to estimate this. However, we have gotten rid of the problematic term. Then using the triangle inequality, we have that

$$\begin{aligned}\|D_{lcmds} - D\|_F^2 \leq& \|D_{lcmds} - D_l\|_F^2 + \|D_l - D\|_F^2 \\ \leq& 2\|f(D_{lcmds}) - f(D_l)\|_F^2 + \|D_t - D\|_F^2.\end{aligned}$$

Thus, we see that the approximation error produced by this procedure is now only $2\|f(D_{lcmds}) - f(D_l)\|_F^2$. This gives us a new approximation algorithm for the true MDS problem.

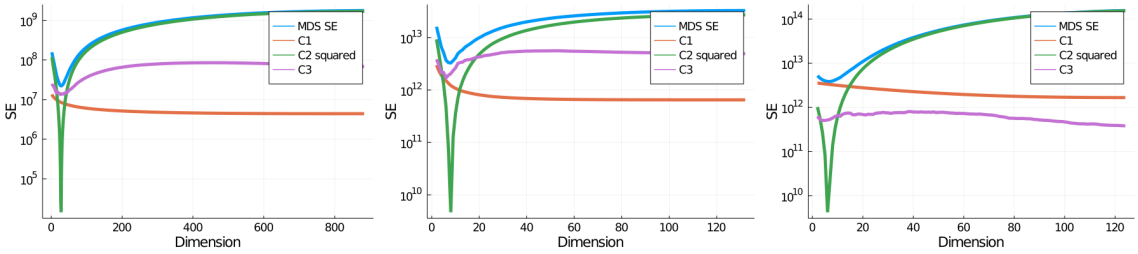


(i) SNR 3e4 (ii) SNR 324 (iii) SNR 3.24

Figure 7.1: Plot showing the relative squared error with respect to original EDM matrix and the perturbed EDM for the cMDS algorithm.

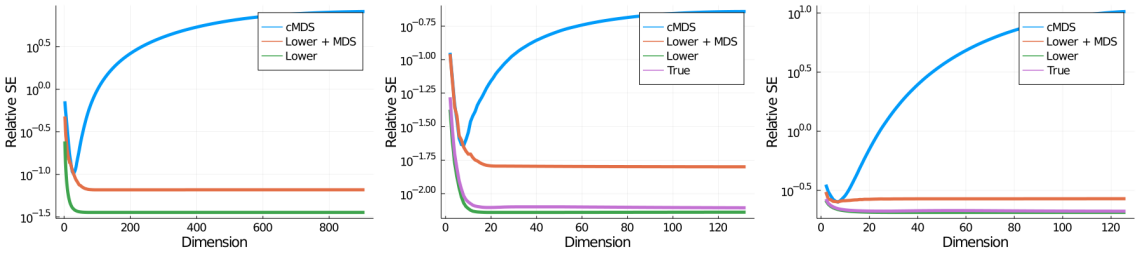
7.4 Experiments

In this section, we look at three important types of input dissimilarity matrices D for which $\hat{D}$ has positive eigenvalues. In all cases, we show that the classical

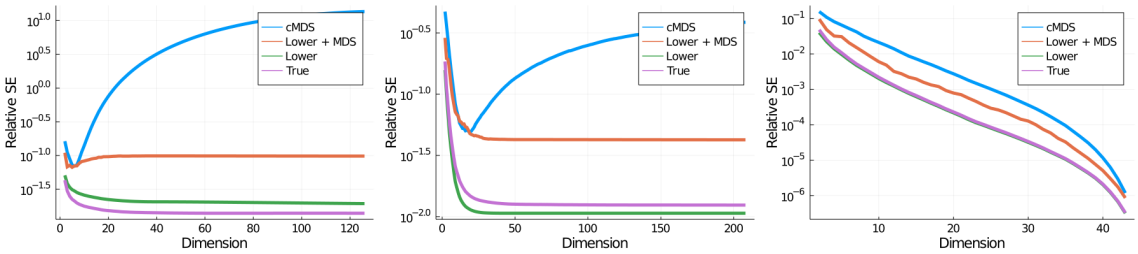


(i) Portugal (ii) Isomap with heart (iii) Perturbed with SNR 3.24

Figure 7.2: Plots showing the cMDS error as well as the three terms that we decompose the error into. For the perturbed EDM input, this is error with respect to the perturbed EDM.



(i) Portugal (ii) Isomap with Heart (iii) Perturbed with SNR 3.24 - Input

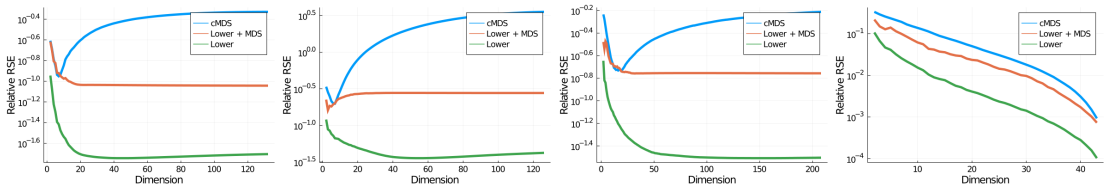


(iv) Perturbed with SNR 3.24 - Original (v) Celegans (vi) Heart

Figure 7.3: Plots showing the relative squared error of the solutions with respect to the input matrix. For the perturbed EDM input, we show the relative squared error with respect to the original EDM (figure (d)) and the perturbed EDM (figure (c)). For the Portugal data set, we couldn't compute the true solution due to computational restraints.

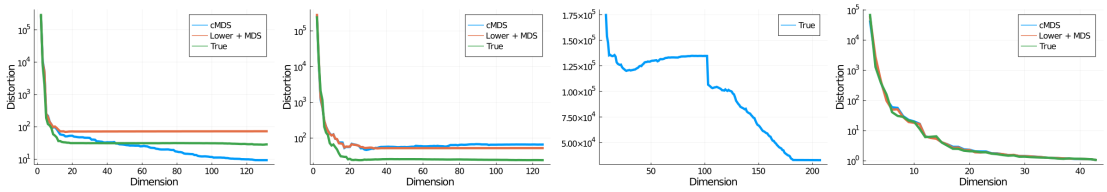
MDS algorithm does not perform as previously expected; instead, it matches our new error analysis. In each case, we demonstrate that our new algorithm Lower + cMDS outperforms cMDS.

Perturbed Euclidean Metrics. Here we take an EDM and perturb it by adding noise. We consider a variety of different noise levels and show that even in the low



(i) Isomap with Heart (ii) Perturbed with SNR 3.24 (iii) Celegans (iv) Heart

Figure 7.4: Plots showing the relative squared error of the various solutions with respect to the true solution.



(i) Isomap on Heart (ii) Perturbed with SNR 3.24 (iii) Celegans (iv) Heart

Figure 7.5: Plot showing the distortion with respect to the input metric. In the case when the input is a perturbed EDM this is the distortion with respect to the original EDM. For the Celegans data set the cMDS solution and the Lower + cMDS solution have infinite distortion and the curves are not plotted.

noise regimes, somewhat counterintuitively, the cMDS error increases as the dimension increases. If D is the input matrix, we perturb it with a matrix G that is symmetric, hollow, and has entries drawn independently from a Gaussian. We define the signal to noise ratio SNR as $\|D\|_F^2/\|G\|_F^2$. We consider 3 different values for the SNR, specifically $3e4$, $3e2$ and 3. We use the heart dataset *Detrano et al.* (1989) and treat each feature as a coordinate. Doing so produces a Euclidean dataset with a distance matrix whose dimensionality is lower than the number of features. Hence, the data live on a lower dimensional structure.

Non-Euclidean Metrics. We know from Section 7.2, that if D is a symmetric, hollow matrix, and $\hat{D}$ is negative semi-definite, then D is an EDM. However, when D is non-euclidean, $\hat{D}$ necessarily has positive eigenvalues. We use two different data sets to illustrate non-Euclidean metrics; Celegans *Rossi and Ahmed* (2015) and Portugal *Rozemberczki et al.* (2019). Celegans is an unweighted metabolic graph for

the Caenorhabditis elegans worm and Portugal is a social network of Twitch steamers in Portugal. We use the all pairs shortest path metric.

Isomap. One common dimensionality reduction technique is Isomap *Tenenbaum et al.* (2000a). In Isomap, a k nearest neighbor graph is constructed from the data, on which the all pairs shortest path metric is computed, and then cMDS is used to embed the metric into Euclidean space. Here, we again use the heart data set.

7.4.1 Results

First, let us see what happens when we perturb an EDM. Here we consider two different measures. Let D be the original input, D_p be the perturbed, and D_{cmds} be the cMDS solution. Then we measure

$$\frac{\|D_p - D_{\text{cmds}}\|_F^2}{\|D_p\|_F^2} \quad \text{and} \quad \frac{\|D - D_{\text{cmds}}\|_F^2}{\|D\|_F^2}.$$

As we can see from Figure 7.1, as the amount of noise increases both quantities eventually increase. The increase in the first quantity suggests that cMDS does not approximate its input well. The increase in the second quantity suggests that cMDS does not do a good job of denoising either. Thus, suggesting that the cMDS objective is not a good one.

Next, we see how cMDS does on graph datasets and with Isomap on Euclidean data. Here we plot the relative squared error ($\|D - D_{\text{cmds}}\|_F^2 / \|D\|_F^2$). Figure 7.3 shows that, as predicted, as the embedding dimension increases, the cMDS error eventually increases as well. For both the Portugal dataset alone and with Isomap on the heart dataset, this error eventually becomes worse than the error when embedding into two dimensions! As we can see from Figure 7.2, this increase is **exactly** due to the C_2^2 term. Also, as heuristically predicted, the C_3 term is roughly constant.

Finally, let us see how the true MDS solution and new approximation algorithm

perform. Here we look at the relative squared error again. As we can see from Figure 7.3, in all cases, we have that the true MDS solution performs much better than cMDS. We can also see from Figure 7.3 (c) and (d) that the true MDS solution and our new approximation solution are closer to the original EDM compared to the perturbed matrix. Thus, they are better at denoising than cMDS. In addition to testing it on all of the previous test cases, we also test its performance for dimensionality reduction on Euclidean data. Figure 7.3 also shows us that our lower bound actually tracks the true MDS solution extremely closely. Finally, we see that our new approximation algorithm Lower + cMDS performs better than just cMDS and, in most cases, fixes the issue of the error increasing with dimension.

We compare the relative squared difference of the Frobenius norm from the true solution to the lower bound matrix, the cMDS solution, and our approximation algorithm solution. As shown in Figure 7.4 both Lower and Lower + cMDS are much better approximations of the true solution!

Finally, to make sure that cMDS is not just scaling the metric, we examine the distortion in the output metric as compared to the input. From figure 7.5, we see that for the heart datatset, for both the heart-isomap input and the perturbed input, they have roughly the same distortion and it is significantly greater than 1. Thus, showing that the squared error seen earlier is not just due to a scaling issue. For the Celegans dataset, both cMDS and our new approximation algorithm Lower + cMDS have infinite distortion so they are not plotted on the graph.

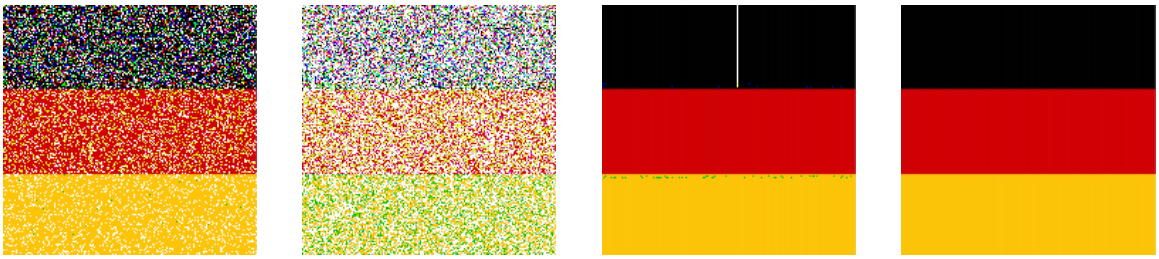
CHAPTER VIII

How to Optimally Train Stacked Linear Denoising Autoencoders?

8.1 Introduction

All real-world datasets are noisy. One way to deal with this noise is to denoise the data. See surveys such as *Tian et al.* (2018, 2020) for many examples of denoising techniques. If one must work with noisy data, then many different techniques such as robust optimization *Gorissen et al.* (2015); *Bertsimas et al.* (2011); *Ben-Tal et al.* (2009), early stopping *Prechelt* (1998), and various forms of activity and weight regularization have been historically employed.

A different approach to dealing with noisy data is to learn good representations of the data. A common way to do this is to add more noise. The archetypal example of this method is stacked denoising autoencoders *Vincent et al.* (2010). Here we take our noisy data, add more noise and train an autoencoder to learn good features. Another popular strategy is to use Dropout *Hinton et al.* (2012); *Wan et al.* (2013); *Srivastava et al.* (2014), where we randomly zero out either neurons or connections. Another idea that is commonly used is data augmentation. In a revolutionary paper, *Krizhevsky et al.* (2012) showed that augmenting the dataset with noisy versions of the images, greatly improved the accuracy. See *Shorten and Khoshgoftaar* (2019) for a survey on



(i) Test Image

(ii) More Noisy Training Image

(iii) Denoised with $\hat{\theta}_{trn} = \hat{\theta}_{tst} = 168$

(iv) Denoised with $\hat{\theta}_{trn} = 30$ as given by Equation 8.2

Figure 8.1: Figure showing the noisy versions as well as the denoised version of the German Flag. Figure (a) shows the test image. Figure (b) shows the noisy image that we should be training on. Figure (c) shows the denoised version when trained with $\hat{\theta}_{trn} = \hat{\theta}_{tst} = 168$. As example of such an image is Figure (a). Figure (d) shows the denoised version when trained with $\hat{\theta}_{trn} = 30$. An example of such an image is in Figure (b). The experiment was run 5 times, and image with the lowest MSE for each denoiser was chosen. Note the way the data matrix was constructed, the data has rank 7.

data augmentation methods. Adding noise has also been shown to have a connection to more traditional techniques such as regularization. For example, it has been shown that adding Gaussian noise is the equivalent of adding a Tikhonov regularizer *Bishop* (1995).

Another area where noise is useful is adversarial learning. *Szegedy et al.* (2014) discovered that adding a small amount of noise to images, such that the difference is not perceptible to the human eye, can completely disrupt a neural network. They then showed that adding such examples to the training set improved the robustness of the neural network. See *Akhtar and Mian* (2018); *Chakraborty et al.* (2018) for surveys on the topic.

This dual nature of noise, both as a hindrance and a helpful tool, lets us conclude that some noise is helpful while too much noise is disruptive. This then leads us to the following principal question. How much noise is too much noise?

The importance of changing $\hat{\theta}_{trn}$ can be seen in Figure 8.1. Here, we took an image of the German flag and then created a noisy version of the flag, as shown in Figure

8.1i. We then train two linear autoencoders to denoise the data. First is trained with the same noise level as the test set. The second is trained on an image (Figure 8.1ii) with more noise, whose SNR is given by our formula. As we can see, training with the SNR adjusted resulted in training a denoiser that performs better.

In this paper, we attempt to answer this question theoretically. The main contributions of our paper are as follows:

1. Under some general assumptions about the noise, we derive an analytical formula for the expected mean squared generalization error for denoising rank one data by a linear autoencoder. This formula depends on the number of data points, the number of features, and the signal to noise ratio (SNR) of the training and test data. This lets us compute the optimal training SNR to denoise a given data set. **For fixed test data SNR, our formula shows that the optimal training SNR is usually not the same as that of the test data, but in fact, lies on a V shaped curve that depends on the number of training points.**
2. Let c be the ratio of the number of features to the number of training points. Then we show via theoretical analysis that the optimal training SNR $\hat{\theta}_{opt-trn}$ depends on c as follows. When $c = 0$, we show that $\hat{\theta}_{opt-trn}$ and the test data SNR are the same. Then as $c \to 1$, we have that $\hat{\theta}_{opt-trn} \to 0$. Finally, as $c \to \infty$, we have that $\hat{\theta}_{opt-trn} \to \infty$. Thus, if we plot $\hat{\theta}_{opt-trn}$ versus c, then we see that the curve is V shaped.
3. We empirically verify our theoretical results and demonstrate that the $\hat{\theta}_{opt-trn}$ versus c curve is V shaped for denoising general data with a linear autoencoder.

In terms of related recent theoretical work, *Pretorius et al.* (2018) derived the learning dynamics of a linear autoencoder in the presence of noise. They also establish some relationships between the noise added and weight decay. However, they do not quantify the optimal amount of noise that should be. *Gnansambandam and Chan* (2020) did look at the problem of what is the optimal amount of noise that should

be added. However, they looked at it from a perspective of minimizing the variance of the performance when the amount of noise in the test dataset deviates from the expected amount.

The rest of this paper will be organized as follows. In Section 8.2, we provide the mathematical framework for the rest of the paper. In Section 8.3, we derive all of our theoretical results. In Section 8.4.1, we empirically verify our theoretical findings. In Section 8.4.2, we show that V shaped $\hat{\theta}_{opt-trn}$ versus v shaped curve exists beyond our theoretical setup.

8.2 Set-Up

In an SDAE, each layer of the initial network is separately pre-trained. Specifically, let X be training data and $\tilde{X}$ be a noisy version of X. Let f and g be the encoder and decoder networks respectively. Then in an SDAE, we learn f, g such that we minimize $L(g(f(\tilde{X})), X)$ for some loss function L. Then the encoder network f, becomes the first layer of the SDAE. Let $X_1 = f(X)$ and $\tilde{X_1}$ be a noisy version of X_1. We then train a new encoder f_1 and a new decoder g_1 to minimize $L(g_1(f_1(\tilde{X_1})), X)$. Then f_1 becomes the next layer of the SDAE. In this manner, we create a stacked network that is then fine tuned on the downstream task.

In *Vincent et al.* (2010), they suggest that f should be parameterized as $f(X) = \texttt{sigmoid}(WX + b)$, whereas for g, they suggest using either an affine decoder, in which case the loss function L is the L_2 loss function or using an affine + sigmoid decoder, in which case the loss function L should be a cross-entropy type loss. In this paper, we restrict ourselves to affine decoders and encoders.

8.2.1 Learning Good Representations

To theoretically analyze how much noise should be added, we **use the expected generalization error of the autoencoder** as the quantity that we want to optimize.

Let Z be some latent variable and let F be a generative model so that our data is generated as

$$X = F(Z).$$

Let A_{tst} be a noise matrix sampled from some noise model so that we have test data of the form

$$Y_{tst} = \theta_{tst} F(Z_{tst}) + A_{tst} \in \mathbb{R}^{M \times N_{tst}}.$$

Let $\hat{\theta}^2_{tst} := \|\theta_{tst} F(Z_{tst})\|_F^2 / \|A_{tst}\|_F^2$ be the test data signal to noise ratio (SNR), and assume that $\hat{\theta}_{tst}$ is sampled form a distribution D_{tst}. Additionally, we also have access to a **limited** number (N_{trn}) of training data. That is, we have

$$X_{trn} = F(Z_{trn}) \in \mathbb{R}^{M \times N_{trn}}.$$

We are now interested in the following question. What is the optimal distribution D_{trn}, such that if $\hat{\theta}_{trn} := \|\theta_{trn} F(Z_{trn})\|_F^2 / \|A_{trn}\|_F^2$ is sampled from D_{trn} and we train our autoencoder to minimize

$$\|\theta_{trn} X_{trn} - g(f(\theta_{trn} X_{trn} + A_{trn}))\|_F^2,$$

then we minimize the expected mean squared generalization error

$$\mathbb{E}_{\hat{\theta}_{trn}, \hat{\theta}_{tst}, A_{tst}, A_{trn}} [\|\theta_{tst} F(Z_{tst}) - g(f(Y_{tst}))\|_F^2 / N_{tst}].$$

In addition to taking the expectation with respect to the test SNR $\hat{\theta}_{tst}$ and the noise in the test data, we also take the expectation with respect to the training data SNR $\hat{\theta}_{trn}$ and the noise in the training data. Since, we added the noise to the training data, our learned encoder f and decoder g are random variables. Hence we take the expectation with respect to the training noise as well. For the rest of this paper, we

suppress the variables over which the expectation is being taken, unless it is not clear from the context. Theta with a hat will always refer to the SNR, while Theta without a hat will be the constant in front of X. Also, let $\hat{\theta}_{opt-trn}$ be the random variable that is the optimal training SNR. We refer to this random variable and its distribution interchangeably.

8.2.2 Assumptions about the noise

We consider a fairly general noise model that captures isotropic Gaussian noise but is applicable beyond that. The specific assumptions on our noise model are as follows. If $A \in \mathbb{R}^{M \times N}$ is a noise matrix, then we assume that the entries of A are independent, have zero mean, have variance $1/M$, and have bounded fourth moment. Additionally, we assume that A is bi-unitarily invariant. That is, if Q is an M by M unitary matrix, A is sampled from the measure space $(\mathbb{R}^{M \times N}, \mu)$, F is the map defined by $A \mapsto QA$ and $F_*(\mu)$ is the push-forward measure defined by this map, then we have that $\mu = F_*(\mu)$. We also want a similar condition to hold for the map $A \mapsto A\tilde{Q}$, where $\tilde{Q}$ is any unitary N by N matrix. The final assumption is that with probability 1, A has full rank. **If a noise matrix A satisfies these assumptions, we use the phrase A satisfies the standard noise assumptions to signal this fact.**

8.2.3 Data Generation Assumptions

For the theoretical part of this paper, we assume that F is a linear function. Specifically, let $u \in \mathbb{R}^M$ be our feature vector. For ease of notation, we assume that $\|u\| = 1$. Then to generate N data points, we sample our latent variable $Z := v \in \mathbb{R}^N$ and form the data matrix $X = F(Z) = uv^T$. For us, we have two such vectors $v_{trn} \in \mathbb{R}^{N_{trn}}$ and $v_{tst} \in \mathbb{R}^{N_{tst}}$. Here, we shall assume that $\|v_{trn}\|^2 = 1$ and $\|v_{tst}\|^2 = 1$. We make no other assumptions on u, v_{trn}, v_{tst} except that they are given and fixed.

Due to our noise assumptions, we have that $\mathbb{E}[\|A_{trn}\|_F^2] = N_{trn}$. Thus, the signal

to noise ratio given by $\hat{\theta}_{trn}^2 = \|X_{trn}\|_F^2/\|A_{trn}\|_F^2 = \theta_{trn}^2/N_{trn}$. Similarly for the test data.

8.2.4 Problem Set Up

To summarize, we have test data of the form $Y_{tst} = \theta_{tst}X_{tst} + A_{tst}$, where A_{tst} satisfies the standard noise assumptions, and $\hat{\theta}_{tst}$ is sampled from some distribution D_{tst}. Then, we want to find $\hat{\theta}_{opt-trn} = \theta_{trn}/\sqrt{N_{trn}}$, such that if W is the linear autoencoder that is the solution to the following problem

$$\text{minimize}_{\hat{W}} \quad \|\theta_{trn}X_{trn} - \hat{W}(\theta_{trn}X_{trn} + A_{trn})\|_F^2, \tag{8.1}$$

then, the expected mean squared error, given by

$$EMSE := \mathbb{E}\left[\frac{\|X_{tst} - WY_{tst}\|_F^2}{N_{tst}}\right]$$

is minimized.

8.3 Theoretical Results

The main theoretical result of the paper is summarized below in Theorem 8.1.

Theorem 8.1. Let $X_{trn} = uv_{trn}^T. \in \mathbb{R}^{M\times N_{trn}}$ and $X_{tst} = uv_{tst}^T \in \mathbb{R}^{M\times N_{tst}}$, where $\|u\|_2 = \|v_{trn}\|_2 = \|v_{tst}\|_2 = 1$. Let A_{trn} and A_{tst} be noise matrices that satisfy the standard noise assumption. Let $Y_{trn} = \theta_{trn}X_{trn} + A_{trn}$ for some $\hat{\theta}_{trn} = \theta_{trn}/\sqrt{N_{trn}}$ and let $Y_{tst} = \theta_{tst}X_{tst} + A_{tst}$, where $\hat{\theta}_{tst} = \theta_{tst}/\sqrt{N_{tst}}$ is sampled from D_{tst}. Then suppose that W minimizes $\|\theta_{trn}X_{trn} - WY_{trn}\|_F$. Finally, let $c = M/N_{trn}$. Then, if $c < 1$, we

have that

$$\mathbb{E}\left[\frac{\|X_{tst} - WY_{tst}\|_F^2}{N_{tst}}\right] = \frac{\mathbb{E}[\hat{\theta}_{tst}^2]}{(1+\hat{\theta}_{trn}^2 M)^2} + \frac{\hat{\theta}_{trn}^2 c + \hat{\theta}_{trn}^4 M}{(1+\hat{\theta}_{trn}^2 M)^2(1-c)} + o(1)$$

and if $c > 1$, we have that

$$\mathbb{E}\left[\frac{\|X_{tst} - WY_{tst}\|_F^2}{N_{tst}}\right] = \frac{\mathbb{E}[\hat{\theta}_{tst}^2]}{(1+\hat{\theta}_{trn}^2 N_{trn})^2} + \frac{\hat{\theta}_{trn}^2}{1+\hat{\theta}_{trn}^2 N_{trn}(c-1)} + o(1).$$

The error term $o(1)$ goes to 0 as $N, M \to \infty$.

Before we prove this result, let us look at some of its consequences. First, if we ignore the error term, we can differentiate the above formula to get that the following formula gives the optimal training SNR.

$$\hat{\theta}_{opt-trn}^2 = \begin{cases} \mathbb{E}[\hat{\theta}_{tst}^2]\left(1 - \frac{c}{2-c}\right) - \frac{c}{M(2-c)} & c < 1 \\ 2\mathbb{E}[\hat{\theta}_{tst}^2](c-1) - \frac{1}{N_{trn}} & c > 1 \end{cases} \tag{8.2}$$

Hence, we can see that if we want to sample $\hat{\theta}_{opt-trn}$ from a distribution D_{trn} then we want D_{trn} to be a constant distribution at the above value. The formulas in Equation 8.2 also describe the V shaped $\hat{\theta}_{opt-trn}$ versus c curve as shown in Figure 8.2.

Second, let us translate this result into the SDAE setup from *Vincent et al.* (2010). In the paper, *Vincent et al.* (2010), showed that **adding** noise resulted in better features. One assumption that is implicit in their set up is that their data is already noise. That is, they are given

$$\hat{X}_{trn} = F(Z_{trn}) + \hat{A}_{trn} \text{ and } \hat{X}_{tst} = F(Z_{tst}) + \hat{A}_{tst}.$$

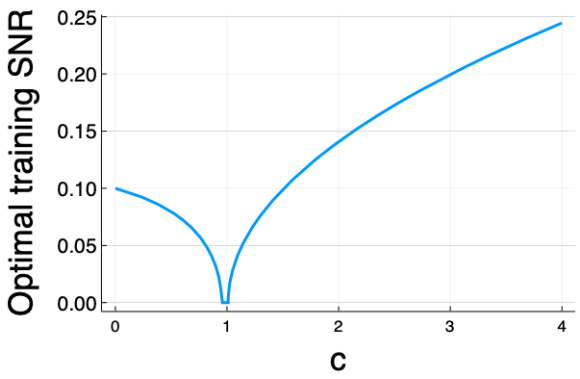


Figure 8.2: Plot showing the equations in Equation 8.2. Here $M = 1000$ and $\hat{\theta}_{tst} = 0.1$.

In this case, they empirically show, using classification accuracy as the yardstick, that training with

$$Y_{trn} = \hat{X}_{trn} + \tilde{A}_{trn} = F(Z_{trn}) + \hat{A}_{trn} + \tilde{A}_{trn},$$

results in improved performance. This interpretation of *Vincent et al.* (2010) is supported by the fact that they show SDAEs have no performance improvement on MNIST and RECT, but show significant performance improvements on noisy variants of MNIST and RECT. That is, their results hold **when they start with noisy datasets**.

Translating this to our setup, we get the following result. For $c < 1$, a lower SNR in the training data as compared to the test data results in better performance. **Thus, we start to provide theoretical justification for the superior performance of stacked denoising autoencoders**.

To begin our theoretical analysis, let us start by first computing the expected generalization error for a given θ_{trn} and θ_{tst}, instead of having distributions for the two values. We prove Theorem 8.1, via the following steps.

1. Derive an analytical formula for W.
2. Using Step 1, decompose the EMSE into various terms.
3. Estimate each term using random matrix theory.

The proofs for all of the lemmas have been moved to the appendix. Here we present

a proof sketch that details most of the high-level ideas.

8.3.1 Step 1: Formula for W

In our current setup, we know that W is the solution to a least-squares problem. Hence $W = X_{trn}Y_{trn}^{\dagger}$. Expanding this out, we get the following formula for W.

Proposition 8.2. Let $h = v_{trn}^T A_{trn}^{\dagger}$, $k = A_{trn}^{\dagger}u$, $s = (I - A_{trn}A_{trn}^{\dagger})u$, $t = v_{trn}(I - A_{trn}^{\dagger}A_{trn})$, $\beta = 1 + \theta_{trn}v_{trn}^T A_{trn}^{\dagger}u$, $\sigma_1 = \theta_{trn}^2\|t\|^2\|k\|^2 + \beta^2$, and $\sigma_2 = \theta_{trn}^2\|s\|^2\|h\|^2 + \beta^2$. If $\beta \neq 0$ and A_{trn} has full rank then

$$W_{opt} = \begin{cases} \frac{\theta_{trn}\beta}{\sigma_1}uh + \frac{\theta_{trn}^2\|t\|^2}{\sigma_1}uk^T A_{trn}^{\dagger} & c < 1 \\ \frac{\theta_{trn}\beta}{\sigma_2}uh + \frac{\theta_{trn}^2\|h\|^2}{\sigma_2}us^T & c > 1 \end{cases}.$$

For the case, when we have Gaussian noise, then with probability 1, A_{trn} has full rank. Additionally, for Gaussian distributions, β is a random variable whose expected value is equal to 1, and that the distribution is highly concentrated. Thus, Proposition 8.2 applies when A_{trn} is isotropic Gaussian noise. We shall show later that any noise matrix that satisfies the standard noise assumptions, in fact, has that the distribution for β is highly concentrated around 1.

8.3.2 Step 2: Decompose the formula for EMSE.

First, we decompose the error into two parts.

Lemma 8.3. If A_{tst} has mean 0 entries and A_{tst} is independent of X_{tst} and W, then

$$\mathbb{E}_{A_{tst}}[\|X_{tst} - WY_{tst}\|_F^2] = \mathbb{E}_{A_{tst}}[\|X_{tst} - WX_{tst}\|_F^2] + \mathbb{E}_{A_{tst}}[\|WA_{tst}\|_F^2].$$

This lets us deal with each term separately. Let us now look at the second term.

Lemma 8.4. If the entries of A_{tst} are independent with mean 0, and variance $1/M$, then we have that $\mathbb{E}_{A_{tst}}[\|WA_{tst}\|^2] = \frac{N_{tst}}{M}\|W\|^2$.

Thus, using Lemmas 8.3 and 8.4, we see that

$$\mathbb{E}_{A_{tst}}[\|X_{tst} - WY_{tst}\|_F^2] = \|X_{tst} - WX_{tst}\|_F^2 + \frac{N_{tst}}{M}\|W\|_F^2.$$

Note that this did not need any assumptions on W or X_{tst}. All that was needed were the assumptions on A_{tst}. Thus, this holds more generally. This decomposition also follows from *Bishop* (1995).

Before we can decompose the first term, we prove the following lemma.

Lemma 8.5. If W is the solution to Equation 8.1, then

$$X_{tst} - WX_{tst} = \begin{cases} \frac{\beta}{\sigma_1}X_{tst} & \text{if } c < 1 \\ \frac{\beta}{\sigma_2}X_{tst} & \text{if } c > 1 \end{cases}.$$

In light of Lemma 8.5 and the fact that $\|X_{tst}\|_F^2 = \theta_{tst}^2$, we see that the expected mean squared generalization error is given by,

$$\mathbb{E}_{A_{tst}}\left[\frac{\|X_{tst} - WY_{tst}\|_F^2}{N_{tst}}\right] = \frac{1}{N_{tst}}\frac{\beta}{\sigma_i}\theta_{tst}^2 + \frac{1}{M}\|W\|_F^2,$$

where σ_i depends on whether $c < 1$ or $c > 1$.

Finally, let us look at the $\|W\|$ term.

Lemma 8.6. If $\beta \neq 0$ and A_{trn} has full rank, then we have that if $c < 1$,

$$\begin{aligned}\|W\|_F^2 = &\frac{\theta_{trn}^2\beta^2}{\sigma_1^2}\mathrm{Tr}(h^T h) + 2\frac{\theta_{trn}^3\|t\|^2\beta}{\sigma_1^2}\mathrm{Tr}(h^T k^T A_{trn}^\dagger) \\ &+ \frac{\theta_{trn}^4\|t\|^4}{\sigma_1^2}\mathrm{Tr}((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger)\end{aligned}$$

and if $c > 1$, then we have that

$$\|W\|_F^2 = \frac{\theta_{trn}^2 \beta^2}{\sigma_2^2} \mathrm{Tr}(h^T h) + 2\frac{\theta_{trn}^3 \|h\|^2 \beta}{\sigma_2^2} \mathrm{Tr}(h^T s^T) + \frac{\theta_{trn}^4 \|h\|^4}{\sigma_2^2} \mathrm{Tr}(ss^T).$$

8.3.3 Step 3: Estimate using random matrix theory.

While the formula given by Lemmas 8.3, 8.4, and 8.6 are correct, we need a simpler formula to analyze the situation. Using ideas from random matrix theory, we can simplify the expression for $\|W\|_F^2$. To do so, we first need to prove Lemmas 8.7 and 8.8.

The main idea behind the lemmas is that due to the bi-invariance of A_{trn}, the expectation of the trace products of various matrices derived from A_{trn} is determined by the expected value of some function χ of the eigenvalues of A. However, instead of directly computing this expected value, we note that for any matrix A, that satisfies the standard noise assumptions, if we let $M, N \to \infty$, with $M/N \to c$, then the eigenvalue distribution converges to the Marchenko - Pastur distribution *Götze and Tikhomirov* (2011, 2003, 2004, 2005); *Bai et al.* (2003). *Götze and Tikhomirov* (2004) showed that the distribution of the eigenvalues converged almost surely with a rate of at least $O(N^{-1/2+\epsilon})$ for any $\epsilon > 0$. Thus, we can use the expected value of the $\chi(\lambda)$ for λ sampled from the Marchenko - Pastur distribution as an approximation. Here we note that the approximation error is a negligible term. Additionally, while we have limited ourselves to having the Marchenko-Pastur distribution as the limiting distribution, in principle using the same argument, we could prove analogous results for any other limiting distribution.

For space reasons, we provide only one instance of the lemmas in the main text. The complete versions can be found in the appendix.

Lemma 8.7. Suppose A is an p by q matrix such that the entries of A are independent

and have mean 0, variance $1/q$, and bounded fourth moment. Let $W_p = AA^T$ and let $W_q = A^T A$. Let $C = p/q$. Suppose λ_p, λ_q are a random eigenvalue of W_p, W_q. Then

1. If $p < q$, then $\mathbb{E}\left[\frac{1}{\lambda_p}\right] = \frac{1}{1-C} + o(1)$.
2. If $p < q$, then $\mathbb{E}\left[\frac{1}{\lambda_p^2}\right] = \frac{1}{(1-C)^3} + o(1)$.
3. If $p > q$, then $\mathbb{E}\left[\frac{1}{\lambda_q}\right] = \frac{C^{-1}}{1-C^{-1}} + o(1)$.
4. If $p > q$, then $\mathbb{E}\left[\frac{1}{\lambda_q^2}\right] = \frac{C^{-2}}{(1-C^{-1})^3} + o(1)$.

Lemma 8.8. Suppose A is an p by q matrix that satisfies the standard noise assumptions. Let x, y be unit vectors in p and q dimensions. Let $C = p/q$. Then

1. $\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger x)] = \begin{cases} \frac{1}{1-C} + o(1) & p < q \\ \frac{q}{p}\frac{C^{-1}}{1-C^{-1}} + o(1) & p > q \end{cases}$.
2. $\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger(AA^T)^\dagger x)] = \begin{cases} \frac{1}{(1-C)^3} + o(1) & p < q \\ \frac{q}{p}\frac{C^{-2}}{(1-C^{-1})^3} + o(1) & p > q \end{cases}$.
3. $\mathbb{E}[\mathrm{Tr}(y^T(A^TA)^\dagger y)] = \begin{cases} \frac{p}{q}\frac{1}{1-C} + o(1) & p < q \\ \frac{C^{-1}}{1-C^{-1}} + o(1) & p > q \end{cases}$.
4. $\mathbb{E}[\mathrm{Tr}(y^T(A^TA)^\dagger(A^TA)^\dagger y)] = \begin{cases} \frac{p}{q}\frac{1}{(1-C)^3} + o(1) & p < q \\ \frac{C^{-2}}{(1-C^{-1})^3} + o(1) & p > q \end{cases}$.

Using these technical lemmas, we can now deal with all of the terms in the expressions in Lemma 8.6. First, let us look at the non-trace terms.

Lemma 8.9. If A_{trn} satisfies the standard noise assumptions, then we have that

1. $\mathbb{E}[\beta] = 1 + o(1)$ and $\mathrm{Var}(\beta) = \frac{\theta_{trn}^2 c}{(max(M, N_{trn})|1-c|))} + o(1)$.
2. If $c < 1$, then $\mathbb{E}[\|h\|^2] = \dfrac{c^2}{1-c} + o(1)$ and $\mathrm{Var}(\|h\|^2) = \dfrac{c^3(2+c)}{N_{trn}(1-c)^3} + o(1)$.
3. If $c > 1$, then $\mathbb{E}[\|h\|^2] = \dfrac{c}{c-1} + o(1)$ and $\mathrm{Var}(\|h\|^2) = \dfrac{c^2(4-c)}{M(c-1)^3} + o(1)$.
4. $\mathbb{E}[\|k\|^2] = \dfrac{c}{1-c} + o(1)$ and $\mathrm{Var}(\|k\|^2) = \dfrac{2-c}{M(1-c)^3} + o(1)$.
5. $\mathbb{E}[\|s\|^2] = \dfrac{c-1}{c} + o(1)$ and $\mathrm{Var}(\|s\|^2) = 2\dfrac{(c-1)^2}{Mc^2} + o(1)$

6. $\mathbb{E}[\|t\|^2] = 1 - c + o(1)$, $\mathrm{Var}(\|t\|^2) = 2\dfrac{(1-c)^2}{N_{trn}} + o(1)$.

Let us now look at the various trace terms as well.

Lemma 8.10. Under standard noise assumptions, we have that

$$\mathbb{E}[\mathrm{Tr}(h^T k^T A_{trn}^\dagger)] = 0$$

and

$$\mathrm{Var}(\mathrm{Tr}(h^T k^T A_{trn}^\dagger)) = \chi_3(c)/N_{trn},$$

where $\chi_3(c) = \mathbb{E}[1/\lambda^3]$, λ is an eigenvalue for AA^T and A is as in Lemma 8.8.

Lemma 8.11. Under standard noise assumptions, we have that

$$\mathrm{Tr}((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger) = \frac{c^2}{(1-c)^3} + o(1)$$

and

$$\mathrm{Var}(\mathrm{Tr}((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger)) = \frac{3}{M}\chi_4(c) - \frac{1}{M}\frac{c^4}{(1-c)^6}$$

where $\chi_4(c) = \mathbb{E}[1/\lambda^4]$, λ is an eigenvalue for AA^T and A is as in Lemma 8.8.

Lemma 8.12. Under the same assumptions as Proposition 8.2, we have that $\mathrm{Tr}(h^T s^T) = 0$.

Lemmas 8.9, 8.10, 8.11, and 8.12 tell us that all of the terms are highly concentrated. Thus, even though such terms may not be uncorrelated, we can use the fact that $|\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]| < |\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}|$, to treat the terms as if they are uncorrelated. Since these variances have now been shown to be small, we will abuse notation by writing an equal to sign while dropping the error terms.

For example, we have that $\sigma_1 = \beta^2 + \theta_{trn}^2 \|t\|^2 \|k\|^2$. Using Lemma 8.9, and our abuse of notation, we have that $\mathbb{E}[\sigma_1] = 1 + \theta_{trn}^2 c$. Similarly, $\mathbb{E}[\sigma_2] = 1 + \theta_{trn}^2$.

Finally, using these lemmas, we can simplify the expressions in Lemma 8.6 to get the following formulas for the expected generalization error. If $M < N_{trn}$, we have that

$$\begin{aligned}\mathbb{E}_{A_{trn}}[\|W\|^2] &= \frac{\theta_{trn}^2}{(1+\theta_{trn}^2 c)^2}\frac{c^2}{(1-c)} + \frac{\theta_{trn}^4(1-c)^2}{(1+\theta_{trn}^2 c)^2}\frac{c^2}{(1-c)^3} \\ &= c^2\frac{\theta_{trn}^2+\theta_{trn}^4}{(1+\theta_{trn}^2 c)^2(1-c)}.\end{aligned}$$

On the other hand, $M > N_{trn}$, we have that

$$\begin{aligned}\mathbb{E}_{A_{trn}}[\|W\|^2] &= \frac{\theta_{trn}^2}{(1+\theta_{trn}^2)^2}\frac{c}{c-1} + \frac{\theta_{trn}^4}{(1+\theta_{trn}^2)^2}\frac{c^2}{(c-1)^2}\frac{c-1}{c} \\ &= \frac{c}{c-1}\frac{\theta_{trn}^2(1+\theta_{trn}^2)}{(1+\theta_{trn}^2)^2} \\ &= \frac{\theta_{trn}^2}{1+\theta_{trn}^2}\frac{c}{c-1}.\end{aligned}$$

If $c < 1$, then

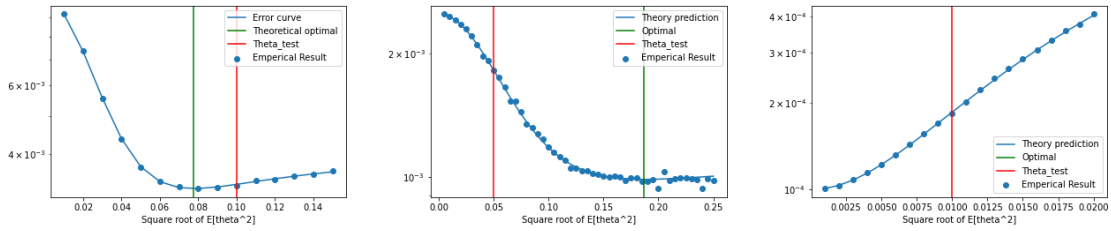
$$\mathbb{E}_{A_{trn},A_{tst}}\left[\frac{\|X_{tst}-WY_{tst}\|}{N_{tst}}\right] = \frac{\theta_{tst}^2}{N_{tst}(1+\theta_{trn}^2 c)^2} + \frac{c^2}{M}\frac{\theta_{trn}^2+\theta_{trn}^4}{(1+\theta_{trn}^2 c)^2(1-c)}$$

and if $c > 1$, then

$$\mathbb{E}_{A_{trn},A_{tst}}\left[\frac{\|X_{tst}-WY_{tst}\|}{N_{tst}}\right] = \frac{\theta_{tst}^2}{N_{tst}(1+\theta_{trn}^2)^2} + \frac{1}{M}\frac{\theta_{trn}^2}{1+\theta_{trn}^2}\frac{c}{c-1}.$$

Recall that $\theta_{trn} = \|X_{trn}\|$ is not the SNR. Our SNRs are given by $\hat{\theta}_{trn}^2 = \frac{\theta_{trn}^2}{N_{trn}}$. We can substitute this into our formulas and get that if $c < 1$ then

$$\mathbb{E}_{A_{trn},A_{tst}}\left[\frac{\|X_{tst}-WY_{tst}\|}{N_{tst}}\right] = \frac{\hat{\theta}_{tst}^2}{(1+\hat{\theta}_{trn}^2 M)^2} + \frac{\hat{\theta}_{trn}^2 c + \hat{\theta}_{trn}^4 M}{(1+\hat{\theta}_{trn}^2 M)^2(1-c)}$$



(i) $c = 0.5$ (ii) $c = 10$ (iii) $c = 2$, $\hat{\theta}_{tst} = 0.01$

Figure 8.3: Figures (a) - (b) show the accuracy of the formula for the expected mean squared error for $c = 0.5, 2$ for fixed value of $\hat{\theta}_{tst} = 0.1$. Figure (c) empirically verifies the existence of a regime where training on pure noise is optimal. Here the red and green lines represent $\mathbb{E}[\hat{\theta}^2_{tst}]$ and $\mathbb{E}[\hat{\theta}^2_{trn}]$ respectively. Each empirical data point is averaged over at least 50 trials.

and if $c > 1$, then.

$$\mathbb{E}_{A_{trn},A_{tst}}\left[\frac{\|X_{tst} - WY_{tst}\|}{N_{tst}}\right] = \frac{\hat{\theta}^2_{tst}}{(1+\hat{\theta}^2_{trn}N_{trn})^2} + \frac{\hat{\theta}^2_{trn}}{1+\hat{\theta}^2_{trn}N_{trn}}\frac{1}{c-1}.$$

Thus, we have derived formulas for the expected mean squared error. It is important to note that this is only accurate when N_{trn} or M is large. Empirically, being bigger than 500 seems to be enough.

Finally, we see that the formula for the expected MSE is linear in $\hat{\theta}^2_{tst}$. Thus, if D_{tst} is no longer a delta distribution, we can replace $\hat{\theta}^2_{tst}$ with $\mathbb{E}[\hat{\theta}^2_{tst}]$. Thus, we have the final piece and we have finished proving Theorem 8.1.

$$\mathbb{E}_{\hat{\theta}_{tst},A_{trn},A_{tst}}\left[\frac{\|X_{tst} - WY_{tst}\|}{N_{tst}}\right] = \frac{\mathbb{E}[\hat{\theta}^2_{tst}]}{(1+\hat{\theta}^2_{trn}M)^2} + \frac{\hat{\theta}^2_{trn}c + \hat{\theta}^4_{trn}M}{(1+\hat{\theta}^2_{trn}M)^2(1-c)}$$

8.3.4 Training with Batches

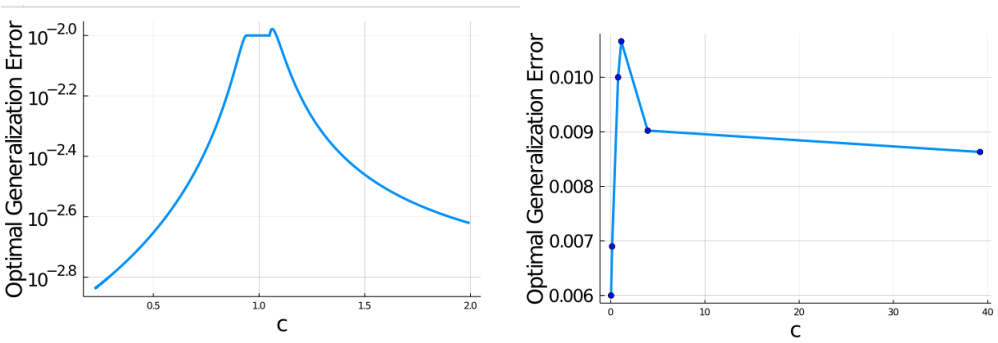
Suppose our training set data consists of K batches $Y_{trn}^{(i)} = \theta_{trn}^{(i)} X_{trn}^{(i)} + A_{trn}^{(i)}$ for $i = 1, \ldots, K$. Here we assume that $\hat{\theta}_{trn}^{(i)}$ is sampled from some distribution D_{trn}. Let us assume that we have a total of N_{trn} data points and each batch has size N_{trn}/K. Then in this case, we are trying to compute W by minimizing $\sum_{i=1}^{K} \|\theta_{trn}^{(i)} X_{trn}^{(i)} - W Y_{trn}^{(i)}\|^2$. This is equivalent to minimizing $\|[X_{trn}^{(1)} \ldots X_{trn}^{(K)}] - W[Y_{trn}^{(1)} \ldots Y_{trn}^{(K)}]\|^2$. Then, we see that $[Y_{trn}^{(1)} \ldots Y_{trn}^{(K)}] = [\theta_{trn}^{(1)} X_{trn}^{(1)} \ldots \theta_{trn}^{(K)} X_{trn}^{(K)}] + [A_{trn}^{(1)} \ldots A_{trn}^{(K)}]$. We still have that the norm squared of the noise matrix is N_{trn}. However, the norm squared of the signal matrix is given by $\sum_{i=1}^{k} (\theta_{trn}^{(i)})^2 = \frac{1}{K} \sum_{i=1}^{k} K(\theta_{trn}^{(i)})^2$. Then we have that

$$\frac{1}{K} \sum_{i=1}^{k} \frac{K}{N_{trn}} (\theta_{trn}^{(i)})^2 = \frac{1}{K} \sum_{i=2}^{K} (\hat{\theta^{(i)}})^2 \approx \mathbb{E}[\hat{\theta}_{trn}].$$

Thus, in this case, we see that if we have batches instead, then we should replace the $\hat{\theta}_{trn}^2$ in the formula with $\mathbb{E}[\hat{\theta}_{trn}^2]$.

8.3.5 Training with no noise

One thing that is important to note is that in the current regime, where we obtain $W = X_{trn} Y_{trn}^{\dagger}$ there is no value of θ_{trn}, even asymptotically, that gives us $W = uu^T$, which is the solution obtained when we train with no noise. We would imagine that as $\theta_{trn} \to \infty$, W should converge to $X_{trn} X_{trn}^{\dagger} = uu^T$. However, the pseudoinverse is not continuous in the limit. In fact, this is also seen from our error formula. For $W = uu^T$, we know that the expected generalization error is $1/M$. However, using our formula as $\theta_{trn} \to \infty$ our error goes to either $\frac{1}{M(1-c)}$ or $\frac{1}{M - N_{trn}}$ depending on whether $c < 1$ or $c > 1$. Thus, without carefully looking at the formula, it is not clear that training with noise is even beneficial. Indeed there are regimes where training with noise is not beneficial. One such regime is when our test data has a high SNR. This further supports the results from *Vincent et al.* (2010), where they showed no improvement



(i) Theoretical Curve (ii) Empirical Curve on MNIST

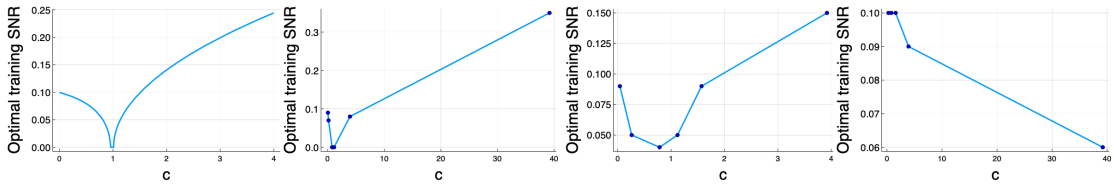
Figure 8.4: Plots showing the optimal generalization error versus c. Figure (a) is the theoretical plot for $\hat{\theta}_{tst} = 0.1$ and $M = 1000$. Figure (b) is empirically computed on MNIST for $\hat{\theta}_{tst} = 0.1$.

on MNIST and RECT. Both of which are datasets with high SNR. However, there exist a lot of regimes where we should train with noise.

8.3.6 c close to 1

We see from our formula that as $c \to 1$, we have that $\hat{\theta}_{trn} \to 0$. In fact, we see that the formulas suggest that $\hat{\theta}^2_{trn}$ should be negative when c is close to 1. We should interpret this as $\hat{\theta}_{opt-trn} = 0$. That is, if we have as many training data points as we have features, then it is optimal to train on a purely noisy sample! This is extremely surprising, as either having more or fewer training samples results in a regime where we do not want to train on only noise!

If we dig deeper and look at the optimal generalization error as c varies (obtained by plugging in the formula for $\hat{\theta}_{opt-trn}$ into the formula for the expected MSE), then we see a surprising result that as c gets closer to 1, the generalization error increases and as c goes to 0 or as c goes to infinity, we have that the generalization error decreases. Thus, if we want to train a linear autoencoder, it might actually be beneficial to reduce the number of training points (if we only have a limited number to begin with). We managed to show that this phenomena exists beyond rank 1 data. Specifically, it exists for MNIST as well. An example of this can be seen in Figure 8.4.



(i) Theoretical Curve (ii) Linear model (iii) LRL model (iv) LSLS model

Figure 8.5: Graphs showing the "V" shape for the $\hat{\theta}_{opt-trn}$ vs c curve. The theoretical curve is based on Equation 8.2 with $\hat{\theta}_{tst} = 0.1$ and $M = 1000$. Figures (b)- (d) are with the MNIST dataset with Gaussian noise so that $\hat{theta}_{trn} = 0.1$ and the optimal $\hat{\theta}_{trn}$ is computed empirically.

8.4 Experiments

Experimentally, we do two things. First, we experimentally verify our theoretical formula and predictions. Second, we move beyond linear autoencoders and linear data and demonstrate the existence of the V shaped optimal $\hat{\theta}_{trn}$ versus c. More details, graphs annd experiments for pre-training SDAEs can be found in the appendix.

8.4.1 Verifying Theoretical Predictions

In our first experiment, we verify the accuracy of our formula for the expected mean squared generalization error given in Theorem 8.1. For $c < 1$, we set $N_{trn} = N_{tst} = 1000$, $M = 500$ and $\hat{\theta}_{tst} = 0.1$. For $c > 1$, we let $M = 1000$ and $N_{trn} = 100$. We can see the results in Figure 8.3. Here we see that our formula fits the expected error almost exactly.

The next thing we empirically verify is that the optimal $\hat{\theta}_{trn}$ can be 0. As shown in Figure 8.3iii, if we let $N_{trn} = 500$, $M = 1000$, and $\hat{\theta}_{tst} = 0.01$, then the best generalization error is achieved when we train with $\hat{\theta}_{trn} = 0$.

The above experiments, were for dirac delta distribution for D_{tst} and a single batch for X_{trn}. Suppose that we have non-constant distributions for D_{tst} and that we multiple batches for the training data, where $\hat{\theta}^{(i)}_{trn}$, the SNR for the ith batch, is sampled from some non-constant distribution D_{trn}. Then to verify that the formulas

are still accurate, we present results for two different combinations of distributions for D_{tst} and D_{trn}. First, D_{trn} is an appropriately scaled uniform distribution and D_{tst} is an exponential distribution. Second, D_{trn} is an exponential distribution and D_{tst} is a Gaussian distribution. As shown in Figures D.1vii and D.1viii, as long as $\mathbb{E}[\hat{\theta}_{trn}^2]$ and $\mathbb{E}[\hat{\theta}_{tst}^2]$ are known, then our formula is accurate.

Figure 8.6: Figure showing the accuracy of the formula when D_{tst} and D_{trn} are distributions. Here $c = 0.1$, and $\hat{\theta}_{tst} = 0.1$

8.4.2 Beyond Linear Data and Linear Autoencoders

Instead of rank 1 data, we now switch to using MNIST. For the test data, we use the standard MNIST test dataset of 10,000 images. Since we are looking at denoising, we add Gaussian noise to the data to produce a dataset with $\hat{\theta}_{tst} = 0.1$. For the training set, we consider the first N_{trn} images for $N_{trn} = 20, 200, 700, 1000, 5000, 20000$. Hence, we get a wide variety of values for c. We trained three different autoencoders on the data. First, a linear autoencoder with the mean squared loss function. Second, a Linear-ReLU-Linear (LRL) autoencoder with the mean squared loss function. Third, a Linear-Sigmoid-Linear-Sigmoid (LSLS) autoencoder with binary cross-entropy as the loss function. As we can see from Figure 8.5ii and Figure 8.5iii, this V shaped curve exists for real-world data and non-linear LRL autoencoders. However, this V shaped curve is not always present, as shown in Figure 8.5iv, as it does not exist for the LSLS architecture.

Finally, we note that when $N_{trn} = 700, 1000$, we have that $c \approx 1$ and for the linear autoencoder, Figure 8.5ii shows that $\hat{\theta}_{opt-trn}$ is extremely close to 0. Thus, empirically verifying that the surprising phenomenon, where we want to train on extremely noisy data, also exists beyond rank 1 data. Further, we are also interested in how the optimal generalization error changes as c changes. This can be seen in Figure 8.4. As we can see, even for MNIST dataset, we have this phenomenon where training on

more data can result in worse performance. Fortunately, for the LRL architecture, we found that training with more data points results in strictly superior performance!

8.5 Future Work

There are many avenues of future work that our paper opens up. One avenue of future work that the authors are interested in, is to determine, either theoretically or empirically - for what architectures, training strategies, loss functions, and generalization error functions does this V curve exist?

CHAPTER IX

Deep Greek: A Framework for Reconstructing Greek Text

9.1 Introduction

Greek papyri are the most ancient form of manuscripts that have preserved Greek texts for us. They date back from the third century BCE to the sixth century CE and mostly come from Egypt because only there the climate was dry enough to preserve them (while in the rest of the Mediterranean world the less dry climate did not allow for the preservation of these ancient manuscripts—hence we have no papyri surviving from mainland Greece unless they had been carbonized).

The problem with working with papyri is that they are fragmentary. This means that the medium (i.e., the papyrus sheet) has holes and breakages because these texts were found in ancient dumps or submerged by layers of dry sand or in mummy cartonnages (plastered layers of papyrus used for masks or panels to cover all or part of the mummified and wrapped body). What we have, therefore, are not complete manuscripts, but rather tiny fragments of what once were full rolls or codices (these were the two different formats of ancient books—the switch between roll and codex occurred around the fourth or fifth century CE, and papyrus was used for both formats). Scholars aiming at publishing an edition of the texts contained in those

fragments are faced with the problem of 'filling the gaps' of highly fragmentary texts.

In addition, according to ancient writing conventions, Greek texts were written without diacritics (i.e., with no accents or breathings—the latter indicated aspiration on initial vowels), without punctuation marks, and without separation between different words (i.e., in *scriptio continua*). Hence, what we normally read as the first line of the *Iliad*:

Μῆνιν ἄειδε, θεά, Πηληϊάδεω Ἀχιλῆος

was written as:

μηνιναειδεθεαπηληιαδεωαχιληος

In fact, in a papyrus we might have something like:

μ...να..δεθεα....ιαδε.αχ...ος

where the dots stand for missing letters (due to lacunae, i.e., breakages in the papyrus, or due to missing fibers or fading of the original ink). Furthermore, the Greek language in papyri is not standard, and there are many irregularities due to ignorance of standard forms (if a standard existed), varying dialects, misspellings and typos, and iotacism (the conversion of several vowel sounds into iota).

In this paper, we take the first step towards literary papyri reconstruction. Specifically, we look at the problem of text reconstruction for texts written in standard classical Greek, preserved by direct tradition (i.e. medieval manuscripts) and available through reliable modern editions. This simplifies the problem by getting rid of issues such as iotacism and misspellings. However, this is already a complex task due to the highly inflected nature of ancient Greek, and due to the fact that we will strip out letters and diacritics from our data to 'recreate' a situation as similar as possible to that of a 'real' papyrus text. The advantage of using a corpus of Greek prose and poetry that is known and transmitted through a reliable tradition is twofold. First, we can train the model with texts which we can choose as belonging to a specific dialect,

time-frame, author, or genre. Second, since we know and can compare the results with the original texts, we can check the accuracy of the model. We hope to use this preliminary work as a stepping stone to working with real papyri in the future.

To address the problem of text reconstruction in this controlled environment, we present a new machine learning model and demonstrate its effectiveness in reconstructing text. Specifically, the main contributions of our paper are the following.

1. Create a data pipeline that breaks the problem into meaningful steps. We also present methods for generating synthetic data to train models for each step. We show that this pipeline results in a 2-7% increase in character accuracy when compared to models that do not use the pipeline. For example, when we have 30% of the characters missing, our pipeline improves the accuracy from 81% to 87%.
2. When we have a small number of missing characters, our data pipeline can be used to efficiently and accurately reconstruct the data. When have 10% of the characters, due to the missing diacritics, about 70% of the input characters are correct. In such cases our reconstructed texts are about 95% accurate.
3. When we have a large number of missing characters, our data pipeline can be used to get a reconstructed version of the text. For example, if we start with texts that are about 45% correct, our pipeline manages to output texts, that are a valid sequence of Greek words, that are around 75% correct.
4. We also test our model on out of sample texts. That is, texts written in a different dialect. We have reasonable performance on these texts already. Thus, suggesting that our trained model can be used for transfer learning.

The rest of the paper is organized as follows. Section 9.2 talks about previous work done on this problem and related problems. Section 9.3 provides details about the data used, the specific scope of the problem, and the general workflow to address the problem. Section 9.4 provides more details on machine learning techniques used to

address our problem. Section 9.5 details how we create the synthetic data and provides more details about the models used. Section 9.6 presents numerical experiments to demonstrate the ability of our method.

9.2 Related Work

The first attempt at solving the text reconstruction problem with neural networks was the work of *Berglund et al.* (2015). In this paper, the authors create the bidirectional RNN model to fill in missing letters in English text. Following this, *Sun et al.* (2017) developed a new sampling technique known as bidirectional beam sampling and made progress in solving the problem in the realm of video captioning. More recently, with the advent of transformer networks *Vaswani et al.* (2017) and the BERT framework *Devlin et al.* (2019), many different architectures have been developed to predict a missing word in a given context. However, these techniques work on the word level, and as we will see, we need to work at the character level.

Recent work by *Assael et al.* (2019b) looks into reconstructing ancient Greek inscriptions. *Assael et al.* (2019b) develop a new model Pythia based on sequence to sequence networks with attention to address this problem. The input to the problem addressd in *Assael et al.* (2019b) is very different from our input. Additionally, the scope of the problem is also very different, as detailed in Section 9.3.1.

9.3 Reconstructing Text

In this section, we discuss the details related to the exact scope of our problem and the data used.

9.3.1 Scope of DeepGreek

Compared with Pythia, the scope of our model is quite different. Pythia is designed specifically to fill in Greek inscriptions. Our **long term goal**, on the other hand, is to fill in Greek literary papyri. Literary papyri cover, in principle, the entire spectrum of Greek literature, so the training data has to be different from what was done for Pythia.

9.3.2 Data Source

Literary texts, both in prose and poetry, can be written in a variety of dialects. The leading Greek dialectal groups are Ionic, Attic, Aeolic, and Doric. Furthermore, from the Hellenistic period on (i.e., ca. from the third century BCE on), Greek developed into a more standard language, roughly corresponding to Attic, called Koine, 'common (language)'. Training our model with all types of dialects would have been too vast as a task. For this first test, we decided to train the model with Attic and Koine Greek. Hence, we selected texts (both prose and poetry), written in Attic or Koine Greek from the fifth century BCE to the third century CE, with the earliest author being Aeschylus (first half of the fifth century BCE) and the latest one Athenaeus (third century CE). While we have restricted our training data, this is not a restriction on the model. In principle, we can expand our training data to include texts written in Ionic, Aeolic, and Doric dialects of Greek. However, for this paper, we will limit ourselves to Attic and Koine Greek.

Our training data consists of texts that have been preserved by the medieval manuscript tradition taken from *Crane* (1987). The full list of texts used can be found in the Appendix. We focus on using such texts as the goal is to study the ability of algorithms to reconstruct the text in the simplified setting of standard classical Greek, before we can build towards being able to look at more complex texts, like those preserved in papyri.

9.3.3 Data Workflow

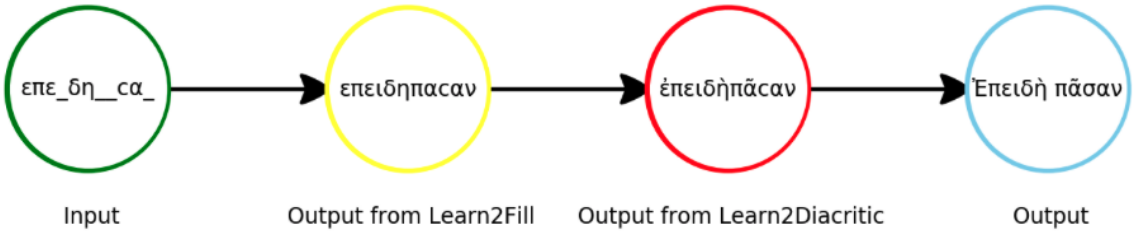


Figure 9.1: The leftmost circle shows the kind of input that we expect, and the rightmost circle shows the reconstructed version of the text that we want. The other circles are intermediate stages of our reconstruction process.

The general workflow by which we will address this problem can be seen in Figure 9.1. The leftmost circle displays the kind of inputs that we expect to get. The final goal is to reconstruct the text into the form seen in the rightmost circle. The in-between circles will be the various stages in which the text is reconstructed. The overall stages are:

1. **Filling in letters.** Given a text without any punctuation, spaces, and diacritics, and with missing letters, we want to fill in the missing base Greek letters. The green circle in Figure 9.1 displays an example of the input text, and the yellow circle displays an example of the desired output. This problem will be the most difficult stage. The model to address this problem will be called Learn2Fill.
2. **Filling in Diacritics.** Given a sequence of Greek characters without any spaces, punctuation, or diacritics, the goal is to fill in the missing diacritics. The yellow circle displays an example of the input text, and the red circle displays an example of the desired output. The model to address this problem will be called Learn2Diacritic.
3. **Filling in spaces.** The final stage is to divide the text into words by adding spaces. Again an example is shown in Figure 9.1. The red circle displays the input, and the blue circle displays the desired output. The model to address

this problem will be called LEARN2SPACE.

Stage 3, is easily addressed by a person with professional knowledge of ancient Greek. Similarly, adding diacritics, which is part of what we do in stage 2, is entirely ruled based and is relatively easily achieved. However, these cannot be done until we have filled in the missing base letters (stage 1).

9.4 Method

To do each stage, we will first train a neural network whose architecture is seen in Figure 9.2. In each case, the input to the neural network will be a string S of length $2n+1$. Here we will think of the $n+1$st letter as the letter that we want to either fill in, put a diacritic on, or decide if there is a space after. Let us call this character C. Let P be the string of the first n characters before C and let Q be the string of the last n characters after C. We will assume that the characters come from an alphabet Σ. Each character will then get a one-hot vector encoding. That is, the ith character is assigned a vector in $\mathbb{R}^{|\Sigma|}$ where the ith coordinate is a one, and the rest of the coordinates are zeros. Thus, neural network takes in matrix of size $2n+1 \times |\Sigma|$. Since we will be using a sequence to sequence model. Our neural network, will output a matrix of size $2n+1 \times |\Sigma|$.

Let us now suppose that we are trying to fill in the base letter, the rest of the stages are similar. Thus, we have some text T of length N that has no diacritics, spaces, and has missing letters. Now in general when we train our model, we will consider fixed n and N will be bigger than $2n+1$. Hence we need to do some work to fill in the whole text. One way in which we could fill in the whole text is that we could give our model an input of size $N \times |\Sigma|$ and then use the output to fill in the missing letters. However, if N is large, such a method would require using the model on inputs that have different distribution (since $N \neq 2n+1$) compared to the training data. One way to fix this would be instead break the text into chunks of size n and

then fill in each independently.

We will instead use beam sampling. To get a handle on beam sampling, let us assume for now that the first n characters of T do not have any missing letters. Now let i be the index of the first missing letter. Then consider the sub-string T_i of length $2n+1$ that starts at the $(i-n)$th character and continues until the $(i+n)$th character. We will give T_i as an input to our neural network and the $n+1$st column of our output will a prediction for the ith letter. So we let S be a list of the $|S|$ most likely characters that could be filled into the ith position along with their respective likelihoods. We then fill in the missing letter with each of the possibilities in S and predict the letter in the $(i+1)$st position using a sub-string T_{i+1}. This way for each entry $s \in S$, we get a list L_s of the possible characters for the $(i+1)$th positions. Thus, for any $l \in L_s$, we can now compute the probability of having the sequence sl as the characters in the ith and the $(i+1)$th place. We now have a lot more than $|S|$ sequences of length two, so we will trim our list of possible sequences to the $|S|$ most likely outcomes. When we predict a letter that we already know, the list L_s will have size 1, while when we are predicting a letter that we do not know, the list L_s will have size $|\Sigma|$. We will continue this until our list S contains sequences of length M. This, M is known as the length of the beam, and $|S|$ is the width of the beam.

Now we had the assumption that the first n characters were present in T. Hence to fill in the first n characters, we will use the above method, but whenever we want to predict a character in the ith place for $i \leq n$, we will let T_i be the string of the first $2n+1$ characters of T and then use the ith column of the output of the neural network as the prediction. Similarly for the last n characters.

9.4.1 Neural Network Architecture

To address this problem, we are going to use a hybrid neural network structure. Specifically, we will use the shallow and wide network shown in Figure 9.2. To get a

better sense of the neural network, let us discuss what each part of the network does.

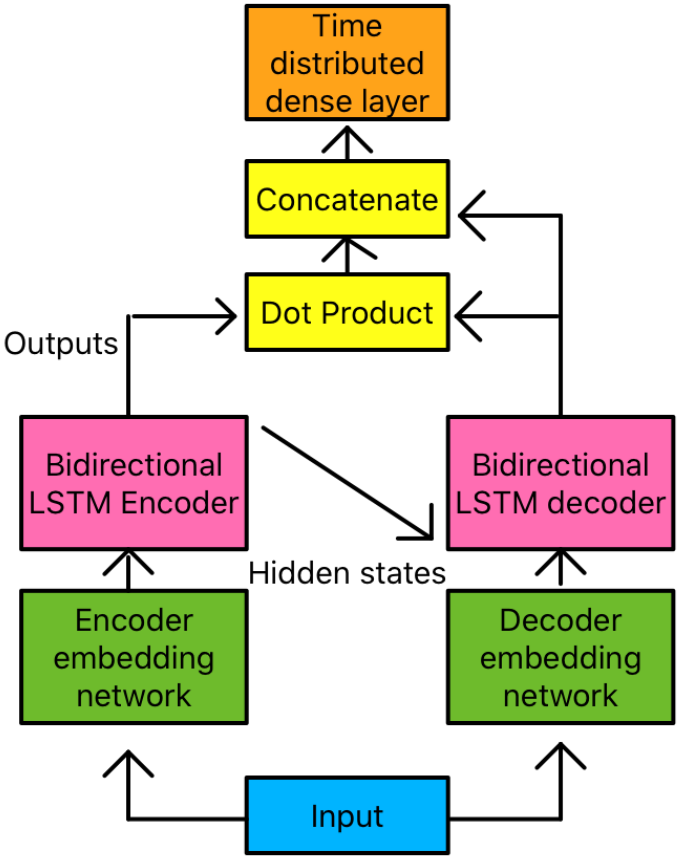


Figure 9.2: Neural Network Architecture for Learn2Fill, Learn2Diacritic, Learn2Space.

9.4.2 Embedding Layer

The first layers of our network are two embedding networks. Recall that out input does not have any spaces. Hence we cannot use word embeddings as Pythia does. Instead we will be use character embeddings. Now there are two different kinds of character embeddings that we could use. The first is a fixed embedding. That is for each letter $s \in \Sigma$ we have a unique vector $v_s \in \mathbb{R}^d$ where d is the embedding dimension.

Due to the highly inflected nature of the language, we do not want to used fixed embedding per character. Hence we followed *Zhang and LeCun* (2015) to learn context dependent character embeddings. *Zhang and LeCun* (2015) showed at we can use a three layer convolutional network to learn context dependent character embeddings. Here we use 1 dimensional convolutions and pad so that the output sequence has the same length as the input sequence.

9.4.3 Encoder and Decoder

The encoder is three bidirectional LSTM layers stacked on top of each each other. While the decoder is four bidirectional LSTM layers stacked on top of each each other. As with sequence to sequence models, the initial hidden state of the first layer of the decoder is the set to be the hidden state for the last encoder layer. Since we are using a bidirectional LSTM for both, this means that the hidden states for the forward LSTM of the last layer of the encoder is are the hidden states for the forward network of first layer of the decoder and similarly for the backward LSTM network of the bidirectional LSTM layer. We also have attention layers between the decoder LSTM layers.

9.5 Creating Training Data

Since our data consists of complete texts with no characters missing, we must synthetically create training data that captures the fragmentary nature of papyri. We also want our training data to have no spaces and diacritics. Furthermore, since we know that we are going to use beam search to fill in the missing letters in a text, our training data should replicate the type of inputs that the neural network would encounter while beam searching.

9.5.1 Filling in Letters

For this problem, we expect our input to be a string whose alphabet has 25 different characters. These 25 characters are the 24 base Greek characters as well as a character "_" to encode missing data. In ancient Greek, there are actually 25 base Greek characters. Specifically, there are two different variants of the letter sigma. Sigma, written as σ, occurs in the middle of the word, whereas ς occurs at the end of the word. Since we want to work with text that has no spaces, we replace both of these with the Greek lunate sigma symbol c. The latter is a way to write sigma both in the middle of words and at the end. It is conventionally used in papyrology because often scholars do not know where one word ends and the next one begins. Lunate sigma, then, makes sigma work like any other letter of the Greek alphabet independently from word-boundaries. An example of the input string, before the one-hot encoding, can be seen below.

$$\underbrace{\text{πιτελουμενδεηδη}}_{P}\underbrace{\text{τ}}_{C}\underbrace{\text{ουδειπνουτωνφαλ}}_{Q}$$

9.5.1.1 Erasing Characters in Input

During beam search, we fill in the text from left to right. Hence we must have that all letters are present in P, while Q has missing letters. C will always be assumed to be missing. Hence we want our input to look like the following:

$$\underbrace{\text{πιτελουμενδεηδη}}_{P}\underbrace{\text{\_}}_{C}\underbrace{\text{ο\_δε\_\_νου\_ωνφ\_\_}}_{Q}$$

Additionally, different parts of the text will have different percentages of missing characters. Thus, during training, we want Learn2Fill to encounter varying levels of missing information in Q. Thus, every time we create a training batch, we sample a p from uniformly from $[0, 2/3]$. Then for each character in Q, with probability p, we change that character to missing. This uniform sampling of missing letters does not

precisely match the distribution of missing letters in papyri. In papyri, we tend to have long runs of missing letters. Hence to account for that, we divide Q into blocks of k characters, and then for each block, we decide independently if that block is present or missing. This procedure helps us concentrate the missing information and better matches the real-world scenario.

9.5.1.2 Corrupting Non Missing Characters

Additionally, when we are using beam search to fill in a text, some of the letters in P would have been filled in by Learn2Fill earlier. Since Learn2Fill is unlikely to be 100% accurate, we need to account for this error. Hence, we should corrupt the letters in P. Specifically, for each character c in P with probability p, we uniformly randomly select a character $\hat{c}$ from Σ and replace c with $\hat{c}$. Here, every time we create a new batch of training data, a new p is sampled uniformly from $[0, 1/5]$. A sample input now looks like the following. The red characters are corrupted characters.

$$\underbrace{\pi\zeta\tau\epsilon\lambda\omega\upsilon\mu\epsilon\nu\delta\epsilon\eta\delta\eta}_{P}\underbrace{\_}_{C}\underbrace{\mathrm{o\_\delta\varepsilon\_\_\nu o\upsilon\_\omega\nu\varphi\_\_}}_{Q}$$

9.5.2 Filling in Diacritics

For this problem, we expect our input to be a string whose alphabet has 139 different characters. This includes all lower case Greek letters except for ᾗ, ᾧ, ᾴ, ῂ, ῷ, ᾇ, ῇ, ᾏ, ᾯ. This is because these letters do not show up in our training texts. Note, however, that other characters with iota subscripts do appear in the text. The output of Learn2Diacritic is a 139 dimensional vector. Thus, the base letter in the output of Learn2Diacritic, need not be the same as the input. Thus, Learn2Diacritic can correct mistakes in the text.

Again, we would like the input to Learn2Diacritic to match the kinds of inputs that would be encountered while doing beam search to fill in the diacritics. When

doing beam search, we will fill in the diacritics from left to right. Thus, our prefix P must already have diacritics filled in. We also expect our inputs to be outputs from LEARN2FILL. Thus, we know that the output is not 100% correct. Hence we corrupt the inputs to LEARN2DIACRITIC during training so that it matches the inputs received during actual use. We use the same procedure detailed in Section 9.5.1.2. Here we sample p uniformly from $[0, 1/5]$. However, for P, we corrupt the letters into any other letter (with diacritics), and for Q, we only have base letters, so we corrupt these letters to other base letters. With probability p, we also corrupt C to some other base letter.

9.5.3 Filling in Spaces

At this point, we have reconstructed most of the missing information, and it is trivial for a human to fill in the spaces. The inputs to our final neural network LEARN2SPACE is a text of length $2n + 1$ characters with no spaces. Again since we expect our outputs from LEARN2ACCENT not to be 100% correct, we corrupt the inputs here as well. The output of the neural network is either a 0 or a 1. It is supposed to be a 0 if there is no space after the $(n + 1)$st character, and it is supposed to be a 1 if there is a space after the $(n + 1)$st character. Here we corrupted 10% of the letters.

9.6 Experimental Results

To evaluate our model, we will have to test our data pipeline in multiple different ways and on multiple different types of data sets. In Sections 9.6.2, 9.6.3, and 9.6.4, we will test the performance of our neural network in solving each of the stages. Here, the inputs will be synthetically created, as described in Section 9.5. For each model, we will take a test string with 40,641 characters and test each model by using the model to address their respective problems on this text using beam search. After we

test each individual stage, we will test stages 1 and 2 together in Section 9.6.5. Finally, we explore whether we can generalize to other types of texts in Section 9.6.6. For all of the experiments we set $n = 10$.

9.6.1 Human Evaluation

As was shown in *Assael et al.* (2019b), humans only got 40% of the missing characters correct when attempting to fill in the missing letters. In our case, we postulate that this is even worse, as we start with much less information (no diacritics and spaces).

9.6.2 Filling in the Missing Letters: Learn2Fill

Block size	Model	Percentage Missing 10%	20 %	30%	40%	50%
1	seq2seq	89%	82%	75%	64%	52%
1	BLSTM	85%	79%	72%	61%	50%
2	seq2seq	77%	69%	60%	50%	41%
2	BLSTM	74%	66%	57%	49%	38%
3	seq2seq	63%	58%	49%	42%	34%
3	BLSTM	63%	56%	47%	40%	33%

Table 9.1: Accuracy of missing letter prediction when using beam search to fill in a ∼40,000 character text T for different percentages and block sizes of missing characters in T for different network architectures.

Since this is the most difficult task in the pipeline, we use this task to compare against other possible neural network architectures. Specifically, we compare against a bidirectional LSTM (BLSTM). To make a fair comparison, we give both models the same amount of context. All models have roughly 10 million parameters. The embedding layers for each model has the same structure, and we use beam sampling with the same width and length to fill in the missing letters. All models were trained with the same compute resources for the same time period. As we can see from Table

9.1, when we will use beam sample to fill in our text string for various percentages and block sizes of missing letters, we see that the sequence to sequence model has the better performance.

Looking at the results more closely, we see that as the block size increases, that is, as the missing information gets more concentrated, the performance of our model decreases. There could be two reasons for this. First, we do not sample the missing characters in blocks when training the network. Thus, when we test the network on a text where the missing letters are sampled in blocks our test data is different from our train data. The other reason could be that as the missing information gets more concentrated the problem becomes harder. To see this imagine the extreme case where all of the missing information is concentrated in one contiguous block. We also see that the concentration of the missing characters has a much more significant impact on the accuracy when compared to the percentage of missing characters. That is having the 10% missing in blocks of 3, the accuracy is much worse than having 30% missing in blocks of 1. This highlights one of the main difficulties with actual papyri - the concentration of missing text.

9.6.3 Filling in Diacritics: Learn2Diacritic

Figure 9.3 shows us the testing accuracy for the Learn2Diacritics neural network. Here we have that $p\%$ of the characters in the prefix P are corrupted to some other letter in our alphabet of size 139. The middle character C is corrupted to one of the base letters with $p\%$ of the time, and the post fix Q has $p\%$ of the characters corrupted to other base letters as well. Thus, we have that at least $p\%$ of the characters in the input are not correct. This does not include the fact that Q has missing diacritics. As we can see from Figure 9.3, once $p \geq 4\%$, we have higher accuracy than $100 - p$. This implies that, in many instances, Learn2Diacritic not only fills in the missing diacritics but corrects corruptions in the base letter. Thus,

LEARN2DIACRITIC will fix some of the mistakes that LEARN2FILL makes. This shows us that our model can be adapted for text emendation.

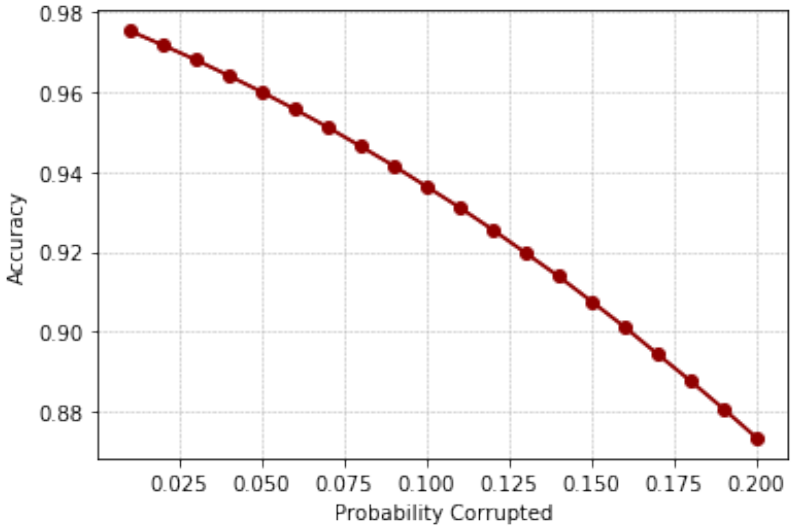


Figure 9.3: LEARN2DIACRITIC testing accuracy for different probability of corrupting letters in P, Q, C.

We then took our ∼40,000 character length test text and tested the accuracy of LEARN2DIACRITIC for different concentrations and percentages of corrupted letters. Similar to the case when we filled in the missing base letters, to get the beam search started, we would need to fill the first n characters differently. Thus, to get around this, we append a known phrase to the beginning and end of our text.

In complete contrast to the situation we have with LEARN2FILL, Figure 9.4 shows that the amount of corrupted information plays a more prominent role than the concentration. We also see that we do not lose any accuracy when we transition from the neural network stage to the stage when we use beam search to fill in the diacritics for a whole text. This is because we make so few errors, and we account for the errors that we do make by employing corruption during the training stage.

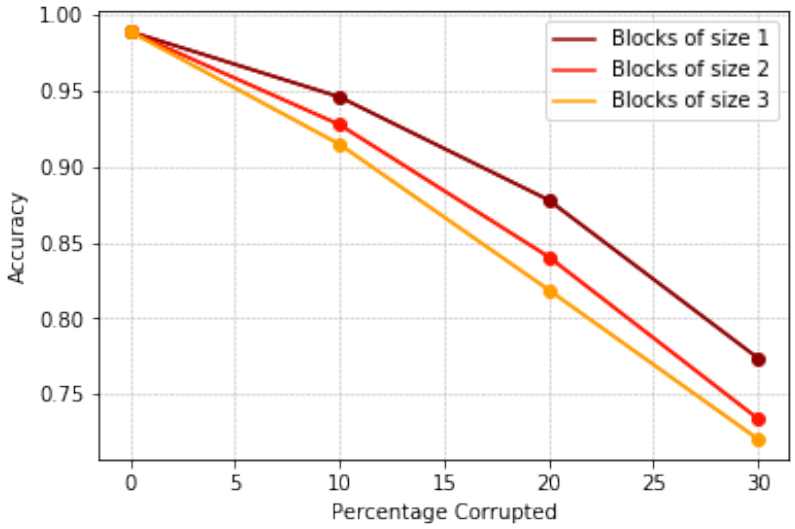


Figure 9.4: Accuracy for LEARN2DIACRITIC for a text T of length ∼40,000 characters for varying block sizes and percentages of corrupted letters in T.

% Missing	10			20			30			40		
Block size	Full Text Character Accuracy											
	Orig.	Split	Comb.	Orig.	Split	Comb.	Orig.	Split	Comb.	Orig.	Split	Comb.
1	69%	97%	94%	62%	93%	89%	54%	87%	81%	46%	79%	74%
2	70%	96%	94%	62%	91%	87%	54%	83%	80%	46%	74%	71%
3	69%	94%	93%	62%	88%	86%	59%	81%	77%	47%	72%	69%

Table 9.2: Percentage of characters correct before (orig) and after having filled in the base letters and the diacritics for our test text for varying percentage of missing text and varying block sizes. Here Split refers to the accuracy of using LEARN2FILL followed by LEARN2DIACRITIC, whereas Comb. refers to LEARN2FILLANDDIACRITIC.

9.6.4 Filling in Spaces: Learn2Space

LEARN2SPACE has a test accuracy of 96%. Note that we do not have spaces in the input to the neural network. Thus, in this case, future outputs of the neural network do not depend on the previous outputs. Hence, beam searching provides no advantage here, and the additional test is not needed.

% Missing	10			20			30			40		
	Full Text Character Accuracy											
	O.	R.	I.	O.	R.	I.	O.	R.	I.	O.	R.	I.
Text 1	69%	97%	28%	63%	96%	33%	56%	91%	35%	48%	84%	36%
Text 2	70%	90%	20%	64%	84%	20%	56%	75%	19%	45%	63%	18%
Text 3	67%	90%	23%	60%	82%	22%	52%	76%	24%	47%	68%	21%
Text 4	66%	78%	12%	61%	72%	11%	54%	64%	10%	47%	56%	9%

Table 9.3: Table showing the percentage of characters correct before (O.) and after (R.) having filled in the base letters and the diacritics for the 4 out of sample texts with varying percentage of missing text. The improve (I.) column is the increase in accuracy between original and the reconstructed text.

9.6.5 Learn2Fill and Learn2Diacritic

Finally, we test our whole pipeline. To do this, we take our ∼40,000 character test string, throw away some letters, all of the diacritics, and the spaces and fill in the missing letters and diacritics.

We also demonstrate why the modular pipeline was needed. To do so, we train a model Learn2FillAndDiacritic that performs stage 1 and stage 2 together. For the comparison to be fair, Learn2FillAndDiacritic has 20 million parameters, since Learn2Fill and Learn2Diacritic each had 10 million parameters. Learn2FillAndDiacritic was also trained for the same amount of time as Learn2Fill and Learn2Diacritic combined.

We can see the accuracy for varying percentages of missing data in Table 9.2. To get a sense of how much information is reconstructed, Table 9.2 also shows the percentage correct for the text that we get as input. As we can see from the table, when we have small amount of missing letters, we reconstruct the text with high accuracy. In the cases, when we have large amounts of missing text, we achieve significant improvement in the percentage correct. Additionally, even when we high amounts of missing data, our output is a sequence of valid Greek words. Thus showing that this method reconstructs text in this setting. Furthermore, we see that

LEARN2FILLANDDIACRITIC has a lower accuracy than the split models.

9.6.6 Other Types of Texts

In addition to testing our model on a test text from our corpus, we would also like to test our model on four texts that differ from our training data. Text 1 is similar to the text in our model (Attic Greek prose of the fifth century). Text 2 is from the same period but is poetry instead of prose (Attic comedy). Text 3 is from the same period, but it is written in a different dialect, not present in our data set (Ionic prose of the fifth century BCE); text 4 is poetry written much earlier than the fifth century BCE and in a very different type of Greek compared to what we have in our data set. Here we tested the accuracy of the letters after filling in the base letter and putting in the diacritics. As we can see in Table 9.2, as we get further away from the type of Greek in our corpus, the accuracy of our model decreases. Thus, we see that if we want to build a general-purpose model to fill in any and all Greek texts, we need to widen our training data to include samples from different types of Greek. This also suggests that we could take our already trained model and fine tune it for a different dialect. The four different Greek texts can be seen in Appendix F.1.1.

9.7 Future Work

A major next step would to be able fill in missing letters when it is unclear how many characters are missing. In actual papyri as we just have a blank space and a researcher must estimate how many characters are missing. We also hope that our work becomes the baseline from which researchers can develop more fine tuned algorithms for various different types of texts. In terms of filling in the base letters, our model still makes mistakes. Often, while the output from our pipeline is a sequence of existing and morphologically correct Greek words, they do not form syntactically meaningful units. Hence we need to train the model at a deeper level, so that it is

able to correctly handle Greek syntax too. This is an avenue for future research.

APPENDICES

APPENDIX A

Generalized Metric Repair on Graphs

A.1 Transitioning to Graph Metric Repair

A.1.1 The decrease only case

For the problem $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$, consider the following simple algorithm, used in previous works for the special case when $G = K_n$.

Algorithm 20 Decrease Metric Repair (DMR)

```
1: function DMR(G = (V, E, w))
2:     Let W = w
3:     For any edge e = uv ∈ E, set W(e) = weight of a shortest path between u and
   v
4:     return W − w
```

Theorem 2.5. *The problem MR$(G, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\mathrm{APSP}})$ time by the* DMR *algorithm.*

Moreover, the problem becomes hard if even a single positive value is allowed. That is, if $0 \in \Omega$ and $\Omega \cap \mathbb{R}_{>0} \neq \emptyset$ then MR(G, Ω) is APX-Complete.

Proof. For the first part, let $e \in G$ be an edge whose edge weight is bigger than the shortest path between the two end points of e. Then in this case e is the heavy edge

in a broken cycle. Hence, any decrease only solution must decrease this edge. Thus all edges decreased by DMR are edges that must be decreased.

By the same reasoning we see that this new weighted graph has no broken cycles. Thus, we see that our algorithm gives a sparsest solution to $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ in $\Theta(T_{\mathrm{APSP}})$ time.

For the second part, the reduction is the same as that of Fan et al. *Fan et al.* (2018d). However, we make the observation that for any value $\alpha > 0$, by appropriately scaling the weights of the reduction in Fan et al. *Fan et al.* (2018d), $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ is still APX-Hard in the extreme case when $\Omega = \{0, \alpha\}$. □

Corollary A.1. For any $G = (V, E, w)$ DMR returns the smallest solution for any ℓ_p norm for $p \in [1, \infty)$.

Proof. The proof of Theorem 2.5 actually shows that there is a unique support for the sparsest solution, i.e., the set of all heavy edges. In fact any decrease only solution must contain all of these edges in its support. We can also see that DMR decreases these by the minimum amount so that the cycles are unbroken. Thus, this solution is in fact the smallest for any ℓ_p norm. □

A.1.2 Structural results

Proposition 2.7. *The* VERIFIER *algorithm (Algorithm 1), given a weighted graph G and a potential support for a solution S, determines in $O(T_{\mathrm{APSP}})$ time whether there exists a valid (increase only or general) solution on that support and if one exists finds one.*

Proof. Let $G = (V, E, w)$ be the original graph and let M be the maximum edge weight from the graph G. The algorithm defines a new graph $\hat{G} = (V, E, \hat{w})$, with the

following weights

$$\hat{w}(e) = \begin{cases} w(e) & e \notin S \\ M & e \in S \end{cases}$$

For each $e = (v_1, v_2) \in E$, line 4 sets $w(e)$ to be the weight of the shortest path in $\hat{G}$ from v_1 to v_2. Thus, at the end of the algorithm $w(e)$ satisfies the shortest path metric of $\hat{G}$. As the algorithm outputs w if and only if only edge weights in S are modified (increased), it suffices to argue S is a regular cover (light cover) if and only if only edge weights in S are modified (increased).

Assume that S is a regular or light cover. We argue line 4 only updates the weights of the edges in S. Note that $G \setminus S$ has no broken cycles. Thus, for any $e = (v_1, v_2) \in G \setminus S$ we have that the shortest path from v_1 to v_2 must be e. Now consider any path P from v_1 to v_2 in $\hat{G}$. If $P \cap S = \emptyset$, then $w(P) \geq w(e)$. On the other hand if $P \cap S \neq \emptyset$, then let $\tilde{e} \in P \cap S$. Then, we have that

$$w(P) \geq w(\tilde{e}) = M \geq w(e)$$

Thus, in either case, $w(P) \geq w(e)$. Hence for all $e \in G \setminus S$ we do not change its weight.

If S is a light cover, we also need to argue that the weights only increased. Let $e = (v_1, v_2) \in S$. Let P be a path of smallest weight in $\hat{G}$. Suppose $P \cap S \neq \emptyset$, then, we have that $w(P) \geq M \geq w(e)$. Thus, in this case we could not have decreased the weight. Thus, assume that $P \cap S = \emptyset$. If we still have that $w(P) \geq w(e)$, then we could not have decreased the weight. Thus, let us further assume that $w(P) < w(e)$. In this case, P along with e form a broken cycle in G, with e as the heavy edge. But then since S is a light cover, we have that $P \cap S \neq \emptyset$. Thus, we have a contradiction and this case cannot occur. Thus, if S is a light cover, then we only increase the edge weights.

Now assume S is not a regular cover (light cover). Then there exists a broken cycle C such that none of its (light) edges are in S. Thus, there is a broken cycle C in $\hat{G}$. Let e be the heavy edge of C, then on line 4 the weight of e will be decreased, and thus our algorithm will return NULL. □

The next theorem shows that once we know the support, the set of all possible solutions on that support is a nice space.

Theorem A.2. For any weighted graph G and support S we have that the set of solutions with support S is a closed convex subset of $\mathbb{R}^{n\times n}$. Additionally, if $G - S$ is a connected graph or we require an upper bound on the weight of each edge, then the set of solutions is compact.

Proof. Let x_{ij} for $1 \leq i, j \leq n$ be our coordinates. Then the equations $x_{ij} = c_{ij}$ for (i, j) not in the support and $x_{ij} \leq x_{ik} + x_{kj}$ define a closed convex set. Thus, we see the first part. For the second part we just need to see that set is bounded to get compactness. If we have that $G - S$ is connected then for all $e \in S$ there is a path between end points of e in $G - S$. Thus, the weight of this path is an upper bound. On the other hand 0 is always a lower bound. Thus, we get compactness if $G - S$ is connected. □

A.2 Approximation Algorithms

Here we give the missing proofs from our $O(\kappa \log n)$-approximation algorithm.

Lemma 2.21. *Let G be a positively weighted graph, where for all pairs of vertices u, v one has constant time access to the value $d(u, v)$. Then for any pair of vertices s, t, the value $\#\mathsf{sp}(s, t)$ can be computed in $O(m + n)$ time.*

Proof. Let $V = \{v_1, v_2, v_3, ..., v_n\}$, and let $N(v_i)$ denote the set of neighbors of v_i. Define $X_i = \{v_j \in N(v_i) \mid w(v_i, v_j) + d(v_j, t) = d(v_i, t)\}$, that is, X_i is the set of

neighbors of v_i where there is a shortest path from t to v_i passing through that neighbor. Thus we have,

$$\#\mathsf{sp}(v_i, t) = \sum_{v_j \in X_i} \#\mathsf{sp}(v_j, t).$$

Note that any shortest path from v_i to t can only use vertices v_j which are closer to t than v_i. So consider a topological ordering of the vertices, where edges are conceptually oriented from smaller to larger $d(v_i, t)$ values. Thus if we compute the $\#\mathsf{sp}(v_i, t)$ values in increasing order of the index i, then each $\#\mathsf{sp}(v_i, t)$ value can be computed in time proportional to the degree of v_i, and so the overall running time is $O(m + n)$. □

Corollary 2.24. *Given constant time access to $d(u, v)$ and $\#\mathsf{sp}(u, v)$ for any pair of vertices u and v, $N_h(e, \delta(G))$ can be computed in $O(1)$ time and $N_l(e, \delta(G))$ in $O(m)$ time.*

Proof. By Lemma 2.22, in constant time we can check whether $w(e) = d(s, t)+\delta(G)$, in which case set $N_h(e, \delta(G)) = \#\mathsf{sp}(s, t)$, and otherwise set $N_h(e, \delta(G)) = 0$. By Lemma 2.23, we can compute $N_l(e, \delta(G))$ with a linear scan of the edges, where for each edge f in constant time we can compute whether $w(f) = d(a, s)+w(e)+d(t, b)+\delta(G)$ and if so add $\#\mathsf{sp}(a, s) \cdot \#\mathsf{sp}(t, b)$ to the sum over X, and if $w(f) = d(b, s)+w(e)+d(t, a)+\delta(G)$ add $\#\mathsf{sp}(b, s) \cdot \#\mathsf{sp}(t, a)$ to the sum over Y. □

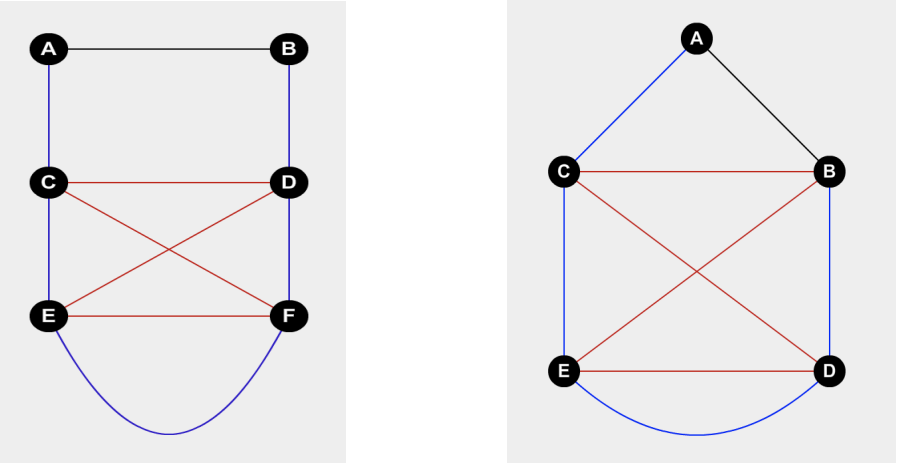
A.3 Improved Analysis for Complete Graphs

Here we consider the special case when $G = K_n$, improving parts of the analysis from *Fan et al.* (2018d); *Gilbert and Jain* (2017). First, we consider the $O(OPT^{1/3})$-approximation algorithm of *Fan et al.* (2018d), which works for both $\mathrm{MR}(K_n, \mathbb{R})$ and $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$. The running time of this algorithm is $\Theta(n^6)$, since at some point it enumerates all cycles of length ≤ 6. With a more careful analysis, we observe

it suffices to consider cycles of length ≤ 5, improving the running time to $\Theta(n^5)$. For $\text{MR}(K_n, \mathbb{R}_{\geq 0})$ we consider a simple, appealing algorithm with good empirical performance from *Gilbert and Jain* (2017), referred to as IOMR-FIXED. We prove that unfortunately it is an $\Omega(n)$ approximation.

A.3.1 5 Cycle Cover

Here we argue the running time of the $O(OPT^{1/3})$-approximation algorithm of *Fan et al.* (2018d), which works for both $\text{MR}(K_n, \mathbb{R})$ and $\text{MR}(K_n, \mathbb{R}_{\geq 0})$, can be improved from $\Theta(n^6)$ to $\Theta(n^5)$. The algorithm presented in *Fan et al.* (2018d) has 3 major steps. The first two steps are used to approximate the support of the optimal solution and then the last step is actually used to find a solution given this support. We shall focus on the first 2 steps as these are where we make modifications.



(i) (ii)

Figure A.1: Left: Embedding from *Fan et al.* (2018d). Right: Our modified embedding for a smaller cycle. Here the black edge is the heavy edge. The blue edges are the light edges and the red edges are the embedded 4 cycle. The curved blue edge indicates that there are more vertices along that path

First Step: In the first step, *Fan et al.* (2018d) find a cover for all broken cycles of length $\leq m$. In particular, the authors use the case when $m = 6$. As described in *Fan et al.* (2018d), we can obtain an $m - 1$ approximation of the optimal cover for all

broken cycles of length $\leq m$ in $O(n^m)$ time. Denote this cover by $S_{\leq m}$.

Second Step: For this step, we need to first define unit cycles. Given a broken cycle C with heavy edge h, let e be a chord of C. Then e divides C into 2 cycles, one that contains h, denoted heavy(C, e) and one that does not contain h denoted light(C, e). We say this cycle is a unit cycle if for all chords e, e is not the heavy edge of light(C, e).

From the definition of a unit cycle, a light cover of all unit cycles light covers all broken cycles. Hence, step 2 of the algorithm from *Fan et al.* (2018d) light covers all unit cycles not covered by $S_{\leq 6}$ as follows. Let C be such a unit cycle. Now we know that C has at least 7 edges. Consider the red C_4 shown in Figure A.1. We know that for each $e \in C_4$, we have that heavy(C, e) is a broken cycle with at most 6 edges. Hence, we must have at least 1 edge in $S_{\leq 6}$. But since C has no light edges in $S_{\leq 6}$, we must have $e \in S_{\leq 6}$. Thus, we know all edges in C_4 are edges in $S_{\leq 6}$. Moreover, observe that either chord of C_4 is a light edge of C. Thus it suffices to compute a cover with least one chord of every four cycle from the edges in $S_{\leq 6}$, a step which the authors in *Fan et al.* (2018d) denote $chord4(S_{\leq 6})$.

In Figure A.1, we observe that the same 4 cycle can be embedded in a 6 cycle instead of a 7 cycle. Thus, our modified algorithm is shown in Algorithm 21.

Algorithm 21 5-Cycle Cover

```
1: function 5 Cycle Cover(G = (V, E, w))
2:     Compute a regular cover of S_{≤5} of all broken cycles with ≤ 5 edges
3:     Compute a cover S_c = chord4(S_{≤5})
4:     return Verifier(G, S_c ∪ S_{≤5})
```

A.3.2 IOMR-fixed

We will now show that IOMR-FIXED is an $\Omega(n)$ approximation algorithm. The algorithm presented in Gilbert and Jain *Gilbert and Jain* (2017) is as follows:

Algorithm 22 IOMR Fixed

Require: $D \in \mathrm{Sym}_n(\mathbb{R}_{\geq 0})$

function IOMR-FIXED(D)
  $\hat{D} = D$
  for $k \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to n **do**
      $\hat{D}_{ik} = \max(\hat{D}_{ik}, \max_{j<i}(\hat{D}_{ij} - \hat{D}_{jk}))$
  return $\hat{D} - D$

Lemma A.3. For every n, there exists a weighted graph G such that IOMR-FIXED repairs $\binom{n-1}{2}$ edge weights while an optimal solutions repairs at most $(n-2)$ edge weights.

Proof. Consider a matrix D where

$$D_{ij} = \begin{cases} 0 & \text{if } i \neq 1, j \neq 1 \\ 2^i & \text{if } j = 1, i > 1 \\ 2^j & \text{if } i = 1, j > 1 \end{cases}$$

This matrix D will be the weight matrix for the input graph K_n.

First, we claim that all entries of the form D_{s1} will never be updated as entries will only be updated the first time they are seen. Thus

$$D_{s1} = \max(D_{s1}, \max_{t<s}(D_{s1} - D_{1t})) = \max(2^s, \max_{t<s}(2^s - 2^t)) = 2^s$$

Now we just have to verify that the rest of the non-diagonal entries are updated. Let us look at the first time an entry D_{rs} is updated. (Here $r < s$.) Then we have that

$$\begin{aligned} \hat{D}_{rs} &= \max(D_{rs}, \max_{t<s}(D_{st} - D_{tr})) = \max_{t<s}(D_{st} - D_{tr}) \qquad [\text{Since } D_{rs} = 0] \\ &\geq D_{s1} - D_{1r} = 2^s - s^r > D_{rs}. \end{aligned}$$

Thus all other non-diagonal entries will be updated the first time seen. Thus, for the solution $W = \hat{D} - D$ that IOMR-FIXED returns, we see that $W_{ij} > 0$ for exactly all $1 < i, j \leq n$ and $i \neq j$. Thus, we repaired $\binom{n-1}{2}$ edge weights.

Finally, a sparser increase only solution W can be obtained as follows. For all $s > 1$ we set

$$W_{1s} = W_{s1} = 2^n - D_{s1}$$

and all other entries of W are 0. This then gives us the desired result. □

Corollary A.4. IOMR-FIXED is an $\Omega(n)$ approximation algorithm.

APPENDIX B

Tree! I am not a Tree! I am a Low Dimensional Hyperbolic Embedding

B.1 Metric First Discussion and Justification

Table 5.4 shows that for most of the data sets, learning a tree structure first and then embedding it into hyperbolic space, yields embeddings with better MAP and average distortion compared to methods that learn the embedding directly. One possible explanation for this phenomenon is that the optimization problems that seek the embeddings directly are not being solved optimally. That is, the algorithms get stuck at some local minimum. Another possibility is that there is a disconnect between the objective being optimized and the statistics calculated to judge the quality of the embeddings.

We propose that there are geometric facts about hyperbolic space that suggest embedding by first learning a tree is the correct approach. The tree-likeness of hyperbolic space has been studied from many different approaches. We present details from *Hamann* (2018); *Dyubina and Polterovich* (2001) and looks at the geometry of $\mathbb{H}^k$ at its two extremes; large scale and small scale. Since $\mathbb{H}^k$ is a manifold, we know

that at small scales hyperbolic space looks like Euclidean space. Additionally, in the Poincare disk, the hyperbolic Riemannian metric is given by $\frac{4}{(1-x^2-y^2)^2}(dx^2+dy^2)$ and is just a re-scaling of the Euclidean metric. Thus, at small scales, hyperbolic space is similar to Euclidean space.

Hence to take advantage of hyperbolic representations (i.e., why learn a hyperbolic representation instead of a Euclidean one), we want to embed data into $\mathbb{H}^k$ at scale. To study the large scale geometry of $\mathbb{H}^k$, we consider the asymptotic cone for hyperbolic space $Con(\mathbb{H}^k)$. In particular, we can think of the asymptotic cone as the "view of our space from infinitely far away". See the more detailed discussion in Appendix B.3 for examples and complete definitions. The following connects $Con(\mathbb{H}^k)$ to $\mathbb{R}$-tree spaces.

Theorem B.1. *Young* (2008) $Con(\mathbb{H}^k)$ is a complete $\mathbb{R}$-tree.

Thus, we see that the large scale structure of hyperbolic space is a tree, indicating a strong connection between learning trees and learning hyperbolic embeddings. Furthermore, it can be shown that $Con(\mathbb{H}^k)$ is a $2^{\aleph_0}$-universal tree. That is, any tree with finitely many nodes can be embedded into $Con(\mathbb{H}^k)$ exactly. However, these are still embeddings into $Con(\mathbb{H}^k)$. We would like to study embeddings into $\mathbb{H}^k$.

Definition B.2. A metric space (T, d_T) admits an isometric embedding at infinity into the space (X, d_X) if there exists a sequence of positive scaling factors $\lambda_i \to \infty$ such that for every point $t \in T$, there exists an infinite sequence $\{x_t^i\}, i = 1, 2, \ldots$ of points in X such that for all $t_1, t_2 \in T$ $\lim_{i\to\infty} d_X(x_{t_1}^i, x_{t_2}^i)/\lambda_i = d_T(t_1, t_2)$

Theorem B.3. *Dyubina and Polterovich* (2001) $Con(\mathbb{H}^k)$ can be isometrically embedded at infinity into $\mathbb{H}^k$.

Thus, we can embed any tree into $\mathbb{H}^k$ with arbitrarily low distortion. A type of converse is also true.

Definition B.4. A (geodesic) ray R is a (isometric) homeomorphic image of $[0, \infty)$, such that for any ball B of finite diameter, R lies outside B eventually.

Hamann (2018) showed that we can construct a rooted $\mathbb{R}$-tree T inside $\mathbb{H}^k$, such that every geodesic ray in $\mathbb{H}^k$ eventually converges to a ray of T. Thus, showing that any configuration of points at scale in $\mathbb{H}^k$ can be approximated by a tree. Additionally, larger the scale points can be better approximated by trees. More details can be found in Appendix B.4. Thus, showing that learning a tree and then embedding this tree into $\mathbb{H}^k$ is equivalent to learning hyperbolic representations at scale.

This provides an explanation for why as the scale and dimension increased, TreeRep found a tree that better approximated the hyperbolic metric in Section 5.4.2. This also provides a justification for why learning a tree first, results in better hyperbolic representations.

B.2 Proofs

B.2.1 Tree Representation Proofs

Lemma 5.11. *Given a metric d on three points x, y, z, there exists a (weighted) tree (T, d_T) on four nodes x, y, z, r, such that r is adjacent to x, y, z, the edge weights are given by $w(x, r) = (y, z)_x$, $w(y, r) = (x, z)_y$ and $w(z, r) = (x, y)_z$, and the metric d_T on the tree agrees with d.*

Proof. The basic structure of this tree can be seen in Figure 5.1iv. To prove that the metrics agree we such need to see the following calculation.

$$\begin{aligned} d_T(x, y) &= w(x, r) + w(r, y) \\ &= (y, z)_x + (x, z)_y \\ &= \frac{1}{2}(d(x, y) + d(x, z) - d(y, z) \\ &\qquad + d(x, y) + d(y, z) - d(x, z)) \\ &= d(x, y) \end{aligned}$$

Here d_T is the metric on the tree T. □

One important fact that we need is that if (X, d) is a metric graph, then for any three distinct points $x, y, z \in X$, the geodesics connecting them intersect at a unique point. As seen in Lemma 5.11, we refer to this point a Steiner point r. It is now important to note that even though r may not be a point in the data set we are given, but $r \in X$ *Bridson and Häfliger* (2013). Thus, in the following lemmas, whenever we find a Steiner point, we will assume that the metric d is defined on r.

Lemma B.5. If d is a tree metric and x, y, w are three points then

1. $(x, y)_w = 0$ if and only if $w \in g(x, y)$
2. $(x, y)_w = d(x, w)$ if and only if $(w, y)_x = 0$.
3. $(x, y)_w = d(y, w)$ if and only if $(w, x)_y = 0$.

Here $g(x, y)$ is the unique path connecting x and y.

Proof. For 1. we see that

$$\begin{aligned} 0 = (x, y)_w = \frac{1}{2}(d(w, x) + d(w, y) - d(x, y)) \\ \Rightarrow d(x, y) = d(w, x) + d(w, y) \end{aligned}$$

Thus we have that $w \in g(x, y)$.

For 2. we see that

$$\begin{aligned} (x, y)_w = d(x, w) &\Rightarrow d(w, x) + d(w, y) - d(x, y) \\ &= 2d(w, x) \\ &\Rightarrow d(w, x) + d(x, y) - d(w, y) = 0 \\ &\Rightarrow 2(w, y)_x = 0 \end{aligned}$$

The proof for 3 is similar to that of 2.

□

Lemma 5.13. *Let (X, d) be a tree space. Let w, x, y, z be four points in X and let (T, d_T) be the universal tree on x, y, z with node r as the Steiner node. Then we can extend (T, d_T) to $(\hat{T}, d_{\hat{T}})$ to include w such that $d_{\hat{T}} = d$.*

Proof. We note that there are four different possible cases for the configuration of x, y, z, w depending on the relationship amongst the Gromov products. Each case determines a different placement of r, as follows:

1. If $(x, y)_w = (x, z)_w = (y, z)_w = 0$, then replace r with w to obtain $\hat{T}$.
2. If $(x, y)_w = (x, z)_w = (y, z)_w = c > 0$, then connect w to r via an edge of weight c to obtain $\hat{T}$.
3. If there exists a permutation $\pi : \{x, y, z\} \to \{x, y, z\}$ such that,

$$(\pi x, \pi y)_w = (\pi x, \pi z)_w = c < (\pi y, \pi z)_w$$

   and $d(\pi x, w) = (\pi x, \pi y)_w$, then connect w to πx via an edge of weight c to obtain $\hat{T}$.
4. If there exists a permutation $\pi : \{x, y, z\} \to \{x, y, z\}$ such that,

$$(\pi x, \pi y)_w = (\pi x, \pi z)_w = c < (\pi y, \pi z)_w$$

   and $d(\pi x, w) > (\pi x, \pi y)_w$, then add a Steiner point $\hat{r}$ on the edge x, r with $d(\pi x, \hat{r}) = d(\pi x, w) - c$ and connect w to $\hat{r}$ via an edge of weight c to obtain $\hat{T}$.

To prove that these extensions of T are consistent, first let us prove that there are exactly four cases. To do that, first note that since we have a 0-hyperbolic metric, at least two of the three Gromov products must be equal. Using the triangle inequality,

we can see that for any three points a, b, c the following holds

$$0 \leq (a, b)_c \leq d(a, c).$$

That is, either we are in the first two cases and three of products are equal, or we have that two of the products are equal. In the case that two of the products are equal, the permutation π tells us which of the two are equal and we further subdivide into the case whether $d(\pi x, w) = (\pi x, \pi y)_w$ or $d(\pi x, w) > (\pi x, \pi y)_w$ as we cannot have $d(\pi x, w) < (\pi x, \pi y)_w$.

Therefore, there are at most four possible configuration cases and it remains to show that the new tree $d_{\hat{T}}$ is consistent with d on the four points. In each case, we present the high level intuition for why these modification result in a consistent tree. The low level details about the metric numbers can easily be checked.

Case 1: If $(x, y)_w = (x, z)_w = (y, z)_w = 0$, then we replaced r with w in $\hat{T}$. In this case, using Lemma B.5, we see that w must lie on all tree geodesics $g(x, y), g(x, z), g(y, z)$. Since the metric comes from a tree, these three geodesics can only intersect at one point r. Thus, we must replace r with w.

To see that the metric is consistent, we need to verify that $d(w, x) = d_{\hat{T}}(r, x)$. To see we have the following:

$$\begin{aligned} d_{\hat{T}}(r, x) &= (y, z)_x \\ &= (y, z)_x + (x, y)_w + (x, z)_w - (y, z)_w \\ &= d(w, x) \end{aligned}$$

Case 2: If

$$(x, y)_w = (x, z)_w = (y, z)_w = c > 0,$$

then we can see that $(x, w)_r = (y, w)_r = (z, w)_r = 0$. In this case, r lies on geodesics

$g(x,y)$, $g(x,z)$, $g(x,w)$, $g(y,w)$, $g(y,z)$, $g(z,w)$. Thus, we must have a star shaped graph with r in the center.

To see that the metric is consistent we just need to verify that $d(w,x) = d_{\hat{T}}(w,x)$. To see that we have the following calculation.

$$\begin{aligned} d_{\hat{T}}(w,x) &= d_{\hat{T}}(w,r) + d_{\hat{T}}(x,r) \\ &= (x,y)_w + (y,z)_x \\ &= (x,y)_w + (x,z)_w - (y,z)_w + (y,z)_x \\ &= d(w,x) \end{aligned}$$

Case 3: In this case suppose condition 4 is true. Without loss of generality assume that π is the identity map. In each case, we have a tree that looks like a tree in Figure 5.1. In this case, we can do the calculations and see that $(w,y)_r = (w,z)_r = 0$. That is, the geodesics $g(w,y), g(w,z), g(y,z), g(x,y), g(x,z)$ all intersect at the same point. Thus, again telling us our tree structure.

To check that the metric is consistent, we need to verify that $d(w,y) = d_{\hat{T}}(w,y) = d_{\hat{T}}(w,r) + d_{\hat{T}}(r,y)$. Before we can do that, let us first verify that

$$d_{\hat{T}}(w,r) = (y,z)_w$$

To verify this we need to the following calculation

$$\begin{aligned} d_{\hat{T}}(w,r) &= d_{\hat{T}}(r,\hat{r}) + d_{\hat{T}}(\hat{r},w) \\ &= c + d_T(x,r) - d_{\hat{T}}(x,\hat{r}) \\ &= c + (y,z)_x - (d(x,w) - c) \\ &= 2c + (y,z)_x - d(x,w) \\ &= (x,y)_w + (x,z)_w + (y,z)_x - d(w,x) \\ &= (y,z)_w \end{aligned}$$

We then can see that

$$\begin{aligned} d_{\hat{T}}(w,y) &= d_{\hat{T}}(w,r) + d_{\hat{T}}(r,y) \\ &= (y,z)_w + (x,z)_y \\ &= (y,z)_w + (x,y)_w - (x,z)_w + (x,z)_y \\ &= d(w,y) \end{aligned}$$

Note $d_{\hat{T}}(w,r) = (y,z)_w$ and the consistency of the metric implies that $d(w,r) = (y,z)_w$. Finally, we can see $(w,y)_r = 0$ as follows.

$$\begin{aligned} 2(w,y)_r &= d(w,r) + d(r,y) - d(w,y) \\ &= (z,y)_w + (x,z)_y - d(w,y) \\ &= \frac{1}{2}(d(w,z) - d(w,y) + d(x,y) - d(x,z) \\ &= (x,z)_w - (x,y)_w \\ &= 0 \end{aligned}$$

Note that this also implies that $(w,x)_r > 0$.

Case 4: In this case, suppose condition 3 is true. Without loss of generality assume

that π is the identity map. Then in this case, we still have that $(w,y)_r = (w,z)_r = 0$, but in addition we have that $(w,y)_x = (w,z)_x = 0$. Thus, again telling us our tree structure.

In this case, to verify that the metric is consistent, we need to check that $d(w,y) = d_{\hat{T}}(w,y) = d_{\hat{T}}(w,x) + d_{\hat{T}}(x,y)$. To see this we have the following calculations.

$$\begin{aligned} d_{\hat{T}}(w,x) + d_{\hat{T}}(x,y) &= (x,y)_w + d(x,y) \\ &= 2(x,y)_w - (x,z)_w + d(x,y) \\ &= d(w,y) + (w,z)_x \end{aligned}$$

Thus, now it suffices to show that $(w,z)_x = 0$, which can be seen using the following calculations.

$$\begin{aligned} (x,z)_w = d(w,x) &\Rightarrow 0 = d(x,w) + d(x,z) - d(w,z) \\ &\Rightarrow (w,z)_x = 0 \end{aligned}$$

This also implies that $(w,z)_r = 0$. □

The proof of Lemma 5.13 shows that there are a number of ways to extend T to include the new point w. To clarify our discussion of the extension of T, we introduce new terminology.

Lemma B.6. Let (X,d) is a metric tree. Let $x,y \in X$ and let $r \in g(x,y)$ if and only if $X \setminus \{r\}$ has at least two disconnected components and x,y are in distinct components.

Proof. Suppose $r \in g(x,y)$. In metric trees, we know that there exist unique simple path between any two points. Therefore, if, after removing r, a path connecting x,y remained (i.e., they are in the same component), then there are two simple paths connecting x,y in X, which is not possible.

Suppose x, y are in two separate components of $X \setminus \{r\}$, then because X is path connected, the geodesic between x and y must pass through r. □

Lemma 5.15. *Given (X, d) a metric tree, and a universal tree T on x, y, z, we have the following*

1. *If $w \in Zone_1(x)$, then for all $\hat{w} \notin Zone_1(x)$, we have that $x \in g(w, \hat{w})$.*
2. *If $w \in Zone_2(x)$, then for all $\hat{w} \notin Zone_i(x)$ for $i = 1, 2$, then we have that $r \in g(w, \hat{w})$.*

Proof. First let us prove statement 1. To do this, let us analyze the possible zones to which $\hat{w}$ belongs.

Case 1: Suppose $\hat{w} \in Zone_1(y)$ (similar for $\hat{w} \in Zone_1(z)$). Then we have that $d(\hat{w}, y) = (x, y)_{\hat{w}}$. This, implies that $(\hat{w}, x)_y = 0$. Thus, by Lemma B.5, we have that $y \in g(\hat{w}, x)$. Similarly we have that $x \in g(w, y)$.

Now since $w \in Zone_1(x)$, we know that $g(x, w) \cap g(x, y) = \{x\}$. Similarly, know that $g(x, y) \cap g(y, \hat{w}) = \{y\}$. Then using Lemma B.6, on removing x, we see that w and y are different connected components. Then since $x \notin g(\hat{w}, y)$, we see that $\hat{w}, y$ is in one connected component. Thus, $w, \hat{w}$ are in different components. Thus, $x \in g(w, \hat{w})$ by Lemma B.6.

Case 2: Suppose $\hat{w} \in Zone_2(y)$ (similar for $\hat{w} \in Zone_2(z)$). Now let r be the Steiner node of the universal tree on x, y, z. In this case we know from Lemma 5.13 that $r \in g(\hat{w}, x)$ and that $g(w, x) \cap g(x, r) = \{x\}$.

Now since $w \in Zone_1(x)$, we know that $g(x, w) \cap g(x, r) = \{x\}$. Similarly, know that $g(x, r) \cap g(r, \hat{w}) = \{r\}$. Then using Lemma B.6, on removing x, we see that w and r are different connected components. Then since $x \notin g(\hat{w}, r)$, we see that $\hat{w}, r$ is in one connected component. Thus, $w, \hat{w}$ are in different components. Thus, $x \in g(w, \hat{w})$ by Lemma B.6.

Case 3: $\hat{w} \in Zone_2(x)$. Let r be the Steiner node for the universal tree on x, y, z.

Now my Lemma 5.13, we know that $x \in g(w, r)$. Thus, again by removing x and using Lemma B.6, r and w are in different. We also have that by Lemma 5.13 $x \notin g(\hat{w}, r)$. Thus $r, \hat{w}$ are in the same connected component of $X \setminus \{x\}$. Thus, w and $\hat{w}$ are in different connected components. Thus, by Lemma B.6, $x \in g(w, \hat{w})$

Thus in all cases, we can see that $x \in g(w, \hat{w})$

Now let us prove statement 2. Without loss of generality assume that

$$\hat{w} \in Zone_i(y)$$

for $i = 1, 2$. Then from Lemma 5.13, we know that $r \notin g(w, x)$ and $r \notin g(\hat{w}, y)$, but $r \in g(x, y)$. Thus, using Lemma B.6 on removing r, x and y and in different components and w is in the same component as x and $\hat{w}$ is in the same component as y. Thus, again using Lemma B.6, we have that $r \in g(w, \hat{w})$.

□

Theorem 5.9. *Given (X, d), a δ-hyperbolic metric space, and n points $x_1, \ldots, x_n \in X$,* TreeRep *returns a tree (T, d_T). In the case that $\delta = 0$, $d_T = d$, and T has the fewest possible nodes.* TreeRep *has worst case run time $O(n^2)$. Furthermore the algorithm is embarrassingly parallelizable.*

Proof. The proof of this theorem follows directly from our structural lemmas. More precisely, we show that for $\delta = 0$, TreeRep returns a consistent metric via induction on n, the number of data points.

Base Case: The case when $n \leq 3$ is covered by Lemma 5.11. And, the case when $n = 4$ is covered by Lemma 5.13.

Inductive Hypothesis: Assume that for all $k \leq n$, our data set of k points is consistent with a 0-hyperbolic metric d, then TreeRep returns a tree (T, d_T) that is

consistent with d on the k points.

Inductive Step: Assume that w is the last vertex attached to T. By the inductive hypothesis, we know that without w, (T, d_T) is consistent on with d so we only need to show that it is consistent with the addition of w.

Now let x, y, z be the universal tree used to sort w in the penultimate recursive step. Let r be the Steiner node. Then by Lemma 5.13, we know that $d_T(w, x) = d(w, x)$, $d_T(w, y) = d(w, y)$, and $d_T(w, z) = d(w, z)$.

Now without loss of generality assume that w was sorted in a zone for x. That is, $w \in Zone_i(x)$ for $i = 1, 2$.

Case 1: If $w \in Zone_1(x)$. Then from Lemma 1, we know that for all $\hat{w} \notin Zone_1(x)$, we have that $x \in g(w, \hat{w})$. Thus, having $d_T(x, w) = d(x, w)$ and $d_T(x, \hat{w}) = d(x, \hat{w})$ is sufficient to show consistency.

Now, since w was placed last there is at most one other point $\tilde{w}$ in $Zone_1(x)$, and $d_T(w, \tilde{w}) = d(w, \tilde{w})$ due to Lemma 5.11.

Case 2: If $w \in Zone_2(x)$. Then from Lemma 2, we know that for all $\hat{w} \notin Zone_i(x)$, for $i = 1, 2$ we have that $r \in g(w, \hat{w})$. Thus, having $d_T(r, w) = d(r, w)$ and $d_T(r, \hat{w}) = d(r, \hat{w})$ is sufficient to show consistency.

Suppose $\hat{w} in Zone_1(x)$. Then from Lemma 1, we have that $x \in g(w, \hat{w})$. Thus, having $d_T(x, w) = d(x, w)$ and $d_T(x, \hat{w}) = d(x, \hat{w})$ is sufficient to show consistency.

Finally, since w was the last node placed there are no other nodes in $Zone_2(x)$.

Thus, we have the the tree returned by TreeRep is consistent with the input metric d.

Notice that whenever we add a Steiner node r we fix the position of at least one data point node. We then look at $O(n)$ Gromov inner products. Thus, we have a worst case running time of $O(n^2)$.

Additionally, the part where we place nodes into their respective zones can be

done in parallel. Thus, if we have K threads then the running time is $O\left(\frac{n^2}{K}\right)$ for the worst running times.

The final part of the theorem is that we return the tree with the smallest possible nodes. Whenever we look at any triangle formed by three points x, y, z, we place a Steiner node r. Now, if none of the distances from x, y, z to r is 0, then this Steiner node must exist in all tree consistent with d. If one of these distances is 0, we contracted that edge and got rid of r. Thus, along with the local consistency argument above this shows that all Steiner nodes that we have placed are necessary (the local consistency argument implies that no two of the Steiner nodes placed could in fact be made into one node due to the nodes beings in different regions). Thus, we have the fewest possible nodes.

□

B.2.2 Tree Approximation Proofs

Proposition 5.16. *Given a δ-hyperbolic metric d, the universal tree T on x, y, z and a fourth point w, when sorting w into its zone $zone_i(\pi x)$,* TreeRep *introduces an additive distortion of δ between w and $\pi y, \pi z$*

Proof. Without loss of generality assume that π is the identity. In this case, we know that $d_T(w, r) = (y, z)_w$, and that $d_T(y, r) = (x, z)_y$. Thus, we have the following:

$$
\begin{aligned}
|d_T(w,y) - d(w,y)| &= |d_T(w,r) + d_T(r,y) - d(w,y)| \\
&= |(y,z)_w + (x,z)_y - d(w,y)| \\
&= \frac{1}{2}|d(w,z) + d(y,x) \\
&\quad - d(w,y) - d(x,y)| \\
&= |(x,y)_w - (x,z)_w| \\
&\leq \delta
\end{aligned}
$$

□

B.3 Geometry: Asymptotic Cones

Definition B.7. An ultrafilter $\mathcal{F}$ on X is a subset of $\mathcal{P}(X)$ such that

1. If $A \in \mathcal{F}$ and $A \subset B$ then $B \in \mathcal{F}$
2. $A, B \in \mathcal{F}$ then $A \cap B \in \mathcal{F}$
3. For any $A \subset X$, exactly 1 of $A, X \setminus A$ is in $\mathcal{F}$
4. $\emptyset \notin \mathcal{F}$.

One way to view $\mathcal{F}$ is as defining a probability measure on X. In particular, we will view the sets in $\mathcal{F}$ to be large and the sets not in $\mathcal{F}$ to be small. Hence, we can define a measure ν such that for all $A \in \mathcal{F}$ we have that $\nu(A) = 1$ and for all $A \notin \mathcal{F}$ we have that $\nu(A) = 0$.

In this way, we can see that ν is a finitely additive measure on X. One common method to define ultrafilters is to take a point $x \in X$ and let $\mathcal{F}$ be the set of all sets that contain x. In this case, the measure ν has a point mass at x and zero mass elsewhere. Such filters are known an principal ultrafilters.

Given a measure ν on $\mathbb{N}$, we can use it to define limits and convergence in X. In

particular, we have that a sequence x_i converges to x, if for all $\epsilon > 0$ we have that

$$\nu\left(\{x_i : |x_i - x| < \epsilon\}\right) = 1$$

We will denote limits of this form as $\lim_\nu x_i = x$.

We will make use of ultrafilters to construct the asymptotic cone. We will do this via looking at a non-principal ultrafilter on $\mathbb{N}$. We consider non-principal ultrafilters as we want to get a view from infinity, and we do not want to be in the case when one particular index in $\mathbb{N}$ has the entire mass. Hence we restrict ourselves to non-principal ultrafilters. One nice characterization of non-principal ultrafilters is that they are exactly the ultrafilters that have no finite sets.

Now that we have mathematical framework in which we can take limits, let us define our asymptotic cone. Let ω be a non-principal ultrafilter on $\mathbb{N}$. Let $\{b_i\}_{i\in\mathbb{N}}$ be a sequence of base points and let $\{\lambda_i\}_{i\in\mathbb{N}}$ be a sequence of scaling factors that go to infinity. Let d be the metric on our space X. Then let

$$X_{\omega,b_i,\lambda_i} = \{\{y_i\} : y_i \in X \text{ and } d(b_i, y_i) \leq const_{\{y_i\}}\lambda_i\}$$

While this space looks huge we will define an equivalence relation and mod out by this relation to obtain better structure on this space. Given two points $y = \{y_i\}, z = \{z_i\} \in X_{\omega,b_i,\lambda_i}$ we say that $y \sim z$ if

$$\lim_\omega \frac{d(y_i, z_i)}{\lambda_i} = 0$$

We can now define our asymptotic cone $Con_\omega(X) = X(\omega, b_i, \lambda_i)/\sim$. We can also define a metric on this space as follows, given $y = \{y_i\}, z = \{z_i\} \in Con_\omega(X)$

$$d_\omega(y, z) := \lim_\omega \frac{d(y_i, z_i)}{\lambda_i}$$

Let us look at a few examples to get a handle on what $Con_\omega(X)$ looks like.

1. Example 1: Let us first consider $X = \mathbb{R}^n$. We know that $\mathbb{R}^n$ is scale invariant. This results in $Con_\omega(\mathbb{R}^n)$ being equivalent to $\mathbb{R}^n$. In fact, if we assume that $b_i \equiv 0$, then the map $x \mapsto \{\lambda_i x\}$ is an isometry from $\mathbb{R}^n$ to $Con_\omega(\mathbb{R}^n)$

2. Example 2: Suppose X is a bounded metric space. In this case $Con_\omega(X)$ is a single point.

Definition B.8. A metric space (X, d_x) can be isometrically embedded into a metric space (Y, d_y) if there exists a map $f : X \to Y$ such that for all $x_1, x_2 \in X$ we have that

$$d_x(x_1, x_2) = d_y(f(x_1), f(x_2))$$

Such a map f is known as an isometry.

Definition B.9. A metric space (X, d) is homogenous if for all $x, y \in X$ there exists an isometry $f : X \to X$ such that $f(x) = y$.

Definition B.10. Given a $\mathbb{R}$-tree T, the valency of a point $x \in T$ in an $\mathbb{R}$-tree is the number of connected components in $T \setminus \{x\}$. Let the valence of a the tree, denoted $val(T)$, be the maximum valence of any point in T.

Definition B.11. A $\mathbb{R}$-tree T is a μ-universal if every $\mathbb{R}$-tree $\hat{T}$ with $val(\hat{T}) \leq \mu$ can be isometrically embedded into T.

Here we can see that we can embed any finite tree into a $2^{\aleph_0}$-universal tree T. Hence, if could isometrically embed T into $Con(\mathbb{H}^n)$ then we can embed any tree into $Con(\mathbb{H}^n)$. This and more turns out to be true.

Theorem B.12. *Dyubina and Polterovich* (2001) Any $2^{\aleph_0}$-universal $\mathbb{R}$-tree can be isometrically embedded into the asymptotic cone for any complete simply connected manifold of negative curvature.

B.4 Geometry: Geodetic Tree

In general, it is rare to be able isometrically embed one space into another. Hence, we have the following weaker definition.

Definition B.13. We say that we can quasi isometrically embed a metric space (X, d_x) into a metric space (Y, d_y) if there exists a map $f : X \to Y$ and real numbers $c, \lambda \in \mathbb{R}$ such that $\lambda \geq 1$, $c > 0$ and for all $x_1, x_2 \in X$ we have that

$$\frac{1}{\lambda} d_x(x_1, x_2) - c \leq d_y(f(x_1), f(x_2)) \leq \lambda d_x(x_1, x_2) + c$$

Such isometries are called (λ, c)-quasi-isometries.

It is has been shown that any δ-hyperbolic metric space (X, d) with bounded growth admits a quasi-isometric embedding into $\mathbb{H}^k$ *Bonk and Schramm* (2000).

Definition B.14. We say that a ray R is quasi geodetic if instead of being an isometric image of $[0, \infty)$, we have that R is an quasi-isometric image of $[0, \infty)$.

Definition B.15. A ray is eventually (quasi) geodetic if it has a subray that is (quasi) geodetic.

Theorem B.16. *Hamann* (2018) For all $\lambda \geq 1$,$c \geq 0$ there is a constant $\kappa = \kappa(\delta, \lambda, c)$, such that for every two points $x, y \in \mathbb{H}^k$, every (λ, c)-quasi-geodesic between them lies in a κ-neighborhood around every geodesic between x and y and vice versa.

Definition B.17. Two geodetic rays π_1, π_2 are equivalent if for any sequence (x_n) of points on π_1, we have $\liminf_{n\to\infty} d(x_n, \pi_2) \leq M$ for an $M < \infty$

Definition B.18. The boundary $\partial\mathbb{H}^k$ of $\mathbb{H}^k$ is the equivalence class of all geodesic rays.

Theorem B.19. *Hamann* (2018) There is an $\mathbb{R}$-tree $T \subset \mathbb{H}^k$ such that the canonical map γ from ∂T to ∂X exists and has the following properties.

1. It is surjective;

2. there is a constant $M < \infty$ depending only on k such that $\gamma^{-1}(\eta)$ has at most M elements for each $\eta \in \partial\mathbb{H}^k$.

Theorem B.20. *Hamann* (2018) Let T be the $\mathbb{R}$-tree in Theorem B.19 with root r. There exist constants $\lambda \geq 1$, $c \geq 0$ such that every ray in T starting at the root is a (λ, c)-quasi-geodetic ray in $\mathbb{H}^k$.

The above two theorems tell us that given any geodesic ray R in $\mathbb{H}^k$ there is exists a ray in T that is equivalent to R (via $\sim$ in Definition B.17). Furthermore this ray in T is (λ, c)-quasi-geodetic ray in $\mathbb{H}^k$. Thus, due to Theorem B.16 any configuration of points at scale in $\mathbb{H}^k$ can be approximated by a tree such that the larger the scale, better the approximation.

B.5 TreeRep Best

So far all numbers for the TreeRep algorithm that we have reported are averages. But due to the speed of the algorithm, we can actually run the experiment multiple times and pick the tree with the best metric.

Table B.1: TreeRep Best Numbers

	No Opt		Heuristic Opt		Full Opt	
Graph	MAP	Distortion	MAP	Distortion	MAP	Distortion
Celegan	0.508	0.173	0.539	0138	0.547	0.119
Diseasome	0.912	0.134	0.911	0.106	0.890	0.092
CS PhD	0.987	0.134	0.984	0.119	0.968	0.121
Yeast	0.841	0.171	0.833	0.150	0.808	0.135
Grid-worm	0.727	0.154	0.728	0.125	-	-
GRQC	0.699	0.175	0.694	0.152	-	-

B.6 Improving Distortion

We have seen that in the case of unweighted graphs TreeRep produces better MAP than PM, LM, and PT. However, PT tends to have better average distortion. Hence, we want to be able to improve the distortion. Once we have learned the tree structure we can set up an optimization problem to learn the edge weights on the tree to improve the distortion. Specifically, since the metric comes from the tree, for any pair of data points, there is exactly one path connecting the two data points. Thus, regardless of the edges weights, this path is the shortest path between the data points. Thus, we can set up an optimization problem of the following form:

$$\arg\min_{w} \|AW - D\|_2.$$

Here W is a vector containing the edge weights, D is a vector containing the original metric, and A is a matrix that encodes all of the paths. This optimization problem however, is unfeasible as n gets longer. So instead we sample some rows of A and solve a heuristic problem. As can be seen from Table B.2, we are still faster than NJ but now have improved our distortion without sacrificing MAP.

Table B.2: MAP and average distortion for the TreeRep and MST after doing the heuristic optimization. The time taken for both optimizations is the same.

Graph	Time	Distortion	MAP	Distortion	MAP
		TreeRep		MST	
Celegans	0.69	0.157	0.504	0.195	0.357
Diseasome	1.56	0.121	0.891	0.111	0.774
CS Phd	1.2	0.152	0.971	0.170	0.989
Yeast	4.2	0.163	0.813	0.171	0.862
Grid Worm	32	0.164	0.707	0.151	0.768
GRQC	68	0.157	0.676	0.159	0.669

B.7 Experiment and Practical Details

B.7.1 MAP and Average Distortion

Definition B.21. Given two metrics d_1, d_2 on a finite set $X = x_1, \ldots, x_n$ the average distortion is:

$$\frac{1}{\binom{n}{2}} \sum_{i=1}^{n} \sum_{j<i} \frac{|d_1(x_i, x_j) - d_2(x_i, x_j)|}{d_2(x_i, x_j)}$$

Smaller average distortion implies greater similarity between d_1 and d_2.

In many cases, the metric learned by the various algorithms will be a scalar multiple of the actual metric, so we will solve for the scale $\alpha := \arg\min_c \|D - c\hat{D}\|_F$, before calculating the average distortion.*

Definition B.22. Let d be a metric on the nodes of a graph $G = (V, E)$. For $v \in V$, let $N(v) = \{u_1, \ldots, u_{deg(v)}\}$ be the neighborhood of v. Then let $B_{v,u_i} = \{u \in V \setminus \{u\} : d(u, v) \leq d(v, u_i)\}$. Then the mean average precision (MAP) is defined to be

$$\frac{1}{n} \sum_{v \in V} \frac{1}{deg(v)} \sum_{i=1}^{|N(v)|} \frac{|N(v) \cap B_{v,u_i}|}{|B_{v,u_i}|}$$

Closer MAP is to 1, the closer d is to approximating d_G.

B.7.2 TreeRep

There are a few practical details that must be discussed in relation to the TreeRep algorithm.

1. Pre-allocate the matrix for the weights of edges of the tree as a dense matrix. Doing this greatly speeds up computations. Note the proof of Lemma 5.13, show that we need at most n Steiner nodes. Thus, the tree has about $2n$ nodes.

*For NJ and LT, computing this α made the average distortion worse, so we report numbers un-scaled. Additionally, computing α is too computationally expensive for bigger data sets and was not done for the Enron and Wordnet data set.

Since the input to the algorithm is a dense $n \times n$ matrix, we already need $O(n^2)$ memory. Thus, having a dense $2n \times 2n$ matrix is still linear memory usage in the size of the input.

2. When doing zone 2 recursions pick the node closest to r as the new z as suggested by Proposition 5.16.
3. The placement of nodes into their respective zones can be done in parallel. For all of the experiments in the paper, we used 8 threads to do the placement for all of the experiments, except that we used 1 thread for the random points from $\mathbb{H}^k$ experiment and for CBMC experiment.
4. All of the numbers reported are averages over 20 iterations. We could have also picked the best over 20 iterations as our algorithm is fast enough for this to be viable.
5. When checking for equality, instead of checking for exact equality, we checked whether two numbers are within 0.1 of each other.
6. It is possible for some of the edge weights to be set to a negative number. In this case, after the algorithm terminated we set those edge weights to 0.

B.7.3 Bartal

We sample 200 trees from the distribution and compute the metric assuming that we are embedding into the distribution restricted to these 200 trees.

B.7.4 Neighbor Join

The following implementation of NJ was used: http://crsl4.github.io/PhyloNetworks.jl/latest/. We set the options so as to not have any negative edge weights.

B.7.5 MST

Prim's algorithm for calculating MST was used. We used the implementation at https://github.com/JuliaGraphs/LightGraphs.jl

B.7.6 LS

Low stretch spanning trees are calculated using Laplacian package in Julia. This code is based an adaptation of *Alon et al.* (1995) by the authors of *Elkin et al.* (2005).

B.7.7 LevelTree and ConstructTree

To the best of the authors knowledge there does not exist a publicly available implementations of these algorithms. Both of these algorithms were implemented by the authors.

Note that LevelTree claims to be a $O(n)$ algorithm, but this only true, once we have calculated the sphere S_n needed for the algorithm. However, it takes $O(n^2)$ time to calculate the spheres S_n (equivalent to solving single source all destination shortest path problem).

B.7.8 PM and LM

The following options were used. The number of epochs was to set to be higher than default. Everything else was left at default. One note about PM and LM is that their objective function is set up to optimize for MAP and not average distortion.

1. -lr 0.3
2. -epochs 1000
3. -burnin 20
4. -negs 50
5. -fresh
6. -sparse

7. -train_threads 2
8. -ndproc 4
9. -batchsize 10

For PM we used `-manifold poincare`, for LM we used `-manifold lorentz`. The code is taken from https://github.com/facebookresearch/poincare-embeddings

B.7.9 PT

The following options were used. We used the `-learn-scale` option as based on the discussion in the appendix of *Sala et al.* (2018) learning the scale results in better quality metrics. Additionally, we add a burnin phase to the optimization. Finally, based on the discussion in *Sala et al.* (2018), the objective function for PT has a lot of shallow local minimas. Thus, we added momentum and used Adagrad for the optimization to try and avoid these local minimums.

1. –learn-scale
2. –burn-in 100
3. –momentum 0.9
4. –use-adagrad
5. –l 5.0
6. –epochs 1000
7. –batch-size 256
8. –subsample 64

The code is taken from https://github.com/HazyResearch/hyperbolics

B.7.10 Hardware

All experiments were run on Google cloud instances. For PM, LM and PT we created a fresh instance for each algorithm. Each instance for an algorithm only had the bare minimum installed to run those algorithms. We used `n1-highmem-8`

instances. The specification of each of the instances are as follows:

1. 8 cores each with 6.5 GB of ram.
2. Ubuntu-1604-xenial-v20190913 operting system.
3. 100 standard persistent disk.

For TreeRep, NJ, CT, LT and MST, we ran all code via a Jupyter notebook interface running Julia 1.1.0. All experiments (except for the experiments with Enron and Wordnet), we done on instances with the same specification as above.

For Enron and Wordnet, we need more memory to store the distance matrices. Thus, used an since with the following specifications.

1. 24 cores each with 6.5 GB of ram.
2. Ubuntu-1604-xenial-v20190913 operting system.
3. 100 standard persistent disk.

B.7.11 Synthetic 0-hyperbolic metrics

To produce random synthetic 0-hyperbolic metrics, we do the following. First, we take a complete binary tree of depth i. We then compute its double tree. Then for each node in this tree we sample a number C from 2 to 10 and replace the node with a clique of size C. We then pick a random node in the tree and compute the breadth first search tree from that node. We then assign edge uniformly randomly, sampled from $[0, 1]$.

B.7.12 Synthetic Data Sets

Here we sampled coordinates from the standard normal $\mathcal{N}(0, 1)$. The final coordinate x_0 is set so that the point lies on the hyperboloid manifold. In the presence of a scale we just multiplied each coordinate by that scale before calculating x_0. We ran TreeRep with 1 thread.

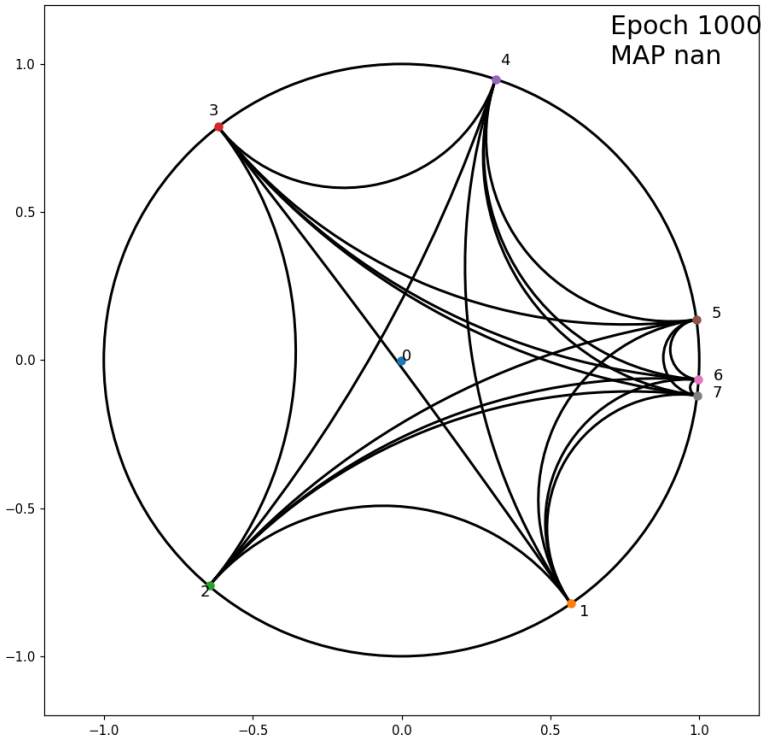


Figure B.1: Figure for Sarich data produced by PT code

B.7.13 Phylogenetic and Single Cell Data

The immunological distances can be seen in Figure B.2. The matrix is symmeterized by averaging across the diagonal. In this case, we ran TreeRep 10 times and picked the tree with the lowest average distortion.

The figures for the trees are produced using an adaptation of Sarkar's construction for Euclidean space. The code from PT also produces a picture. This picture can be seen in Figure B.1. As we can see, this figure is similar to the one in the main text.

For the Zeisel data we did the same pre-processing as done in *Dumitrascu et al.* (2019). For PM and MST, we use 10 nearest neighbor graph. For LS we used the complete graph.

dog	0	32	48	51	50	48	98	148
bear	32	0	26	34	29	33	84	136
raccoon	48	26	0	42	44	44	92	152
weasel	51	34	42	0	44	38	86	142
seal	50	29	44	44	0	24	89	142
sea lion	48	33	44	38	24	0	90	142
cat	98	84	92	86	89	90	0	148
monkey	148	136	152	142	142	142	148	0

Figure B.2: Immunological distances from *Sarich* (1969)

For the CBMC data we did the same pre-processing as done in *Dumitrascu et al.* (2019). For MST and LS we used the complete graph.

B.7.14 Unweighted Graphs

Some of the graphs are disconnected. The largest connected component of each graph was used.

For δ calculation, we normalized the distances so that the maximum distance was 1 and then calculated δ. For Celegans, Diseasome, and Phds,, this calculation is exact.

For Yeast, Grid-worm and GRQC, we fixed the base point to be $w = 1$ and then calculated δ. It is known from theory that for any fixed base point the δ is at least half of the δ for the whole metric *Bridson and Häfliger* (2013). Thus, we get the inequality.

All experiments with a "-" were terminated after 4 hours.

B.7.15 Calculating α

Can be calculated directly using

$$\alpha = \frac{Tr(D' * D}{\|D\|_F^2}$$

Algorithm 23 Recursive parts of TreeRep.

function ZONE1_RECURSION(T, d_T, d, L, v)
  if Length(L) == 0 **then**
    return T
  if Length(L) == 1 **then**
    Set u = pop(L) and add edge (u, v) to E
    Set edge weight $d_T(u, v) = d(u, v)$
    return T
  Set u =pop(L), z =pop(L)
  return RECURSIVE_STEP(T, L, v, u, z, d, d_T)

function ZONE2_RECURSION(T, d_T, d, L, u, v)
  if Length(L) == 0 **then return** T
  Set z= the closest node to v.
  Delete edge (u, v)
  return: RECURSIVE_STEP(T, L, v, u, z, d, d_T)

B.8 Tree Representation Pseudo-code

We can see examples of what happens when we set the new Steiner node for the two different kinds of recursion in Figure B.3

Figure B.3: Figure showing the placement of the Steiner node R' for the Zone 1 and Zone 2 recursion. The nodes in orange are Steiner nodes and the nodes in green come from the data set V.

Algorithm 24 Metric to tree structure algorithm.

function TREE STRUCTURE(X, d)
  $T = (V, E, d') = \emptyset$
  Pick any three data points uniformly at random $x, y, z \in X$.
  $T =$ RECURSIVE_STEP$(T, X, x, y, z, d, d_T,)$
  return T

function RECURSIVE_STEP$(T, X, x, y, z, d, d_T,)$
  Let $Z1(r \rightarrow [], x \rightarrow [], y \rightarrow [], z \rightarrow []), Z2(x \rightarrow [], y \rightarrow [], z \rightarrow [])$ // Dictionaries of list for various zones
  Place an additional node r in V and add edges xr, yr, zr to E
  Set the weights $d_T(x, r) = (y, z)_x$, $d_T(y, r) = (x, z)_y$, and $d_T(z, r) = (x, y)_z$ // If edge weight $= 0$, contract the edge.
  for all remaining data points $w \in X$ **do**
    $a = (x, y)_w$, $b = (y, z)_w$, $c = (z, x)_w$, $m = 0$, $m2 = 0$
    if $a == b == c$ **then**
      push(w, $Z1[r]$)
      Set $d_T(w, r) = (x, y)_w$
    else if $a == maximum(a, b, c)$ **then**
      $\pi = (x \rightarrow z, y \rightarrow y, z \rightarrow x)$
      $m = b$, $m2 = c$
      Set $d_T(w, r) = a$
    else if $b == maximum(a, b, c)$ **then**
      $\pi = (x \rightarrow x, y \rightarrow y, z \rightarrow z)$
      $m = a$, $m2 = c$
      Set $d_T(w, r) = b$
    else if $c == maximum(a, b, c)$ **then**
      $\pi = (x \rightarrow y, y \rightarrow x, z \rightarrow z)$
      $m = a$, $m2 = b$
      Set $d_T(w, r) = c$
    if $d(w, \pi x) == m$ or $d(w, \pi x) == m2$ **then**
      push(w, $Z1[\pi x]$)
    else
      push(w, $Z2[\pi x]$)
  // recurse on each of the zones
  $T =$ ZONE1_RECURSION$(T, d_T, d, Z1[r], r)$
  $T =$ ZONE1_RECURSION$(T, d_T, d, Z1[x], x)$
  $T =$ ZONE1_RECURSION$(T, d_T, d, Z1[y], y)$
  $T =$ ZONE1_RECURSION$(T, d_T, d, Z1[y], z)$
  $T =$ ZONE2_RECURSION$(T, d_T, d, Z2[x], x, r)$
  $T =$ ZONE2_RECURSION$(T, d_T, d, Z2[y], y, r)$
  $T =$ ZONE2_RECURSION$(T, d_T, d, Z2[z], z, r)$
  return T

APPENDIX C

Dual Regularized Optimal Transport

C.1 Proofs

Theorem 6.2. *For the discrete problem, if we add the assumption that ϕ, φ are co-finite Bregman functions then the following problem is the dual problem to DROT($\boldsymbol{a}, \boldsymbol{b}$). Furthermore, strong duality holds for this problem.*

$$\begin{aligned} 6.6 \textit{Minimize:} \quad & \langle \boldsymbol{C}, \boldsymbol{P} \rangle + \frac{\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m)}{\gamma} + \frac{\varphi^*\big(\gamma(\boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_n)\big)}{\gamma} \\ \textit{Subject to:} \quad & \forall i \in [n], \forall j \in [m], \quad \boldsymbol{P}_{ij} \geq 0 \end{aligned} \tag{C.1}$$

If we only have the assumption that ϕ (and similarly for φ) is positively (negatively) co-finite, then we need to add the constraint $\boldsymbol{a} - \boldsymbol{P}\mathbf{1} > 0$ ($\boldsymbol{a} - \boldsymbol{P}\mathbf{1} < 0$).

Proof. Since Bregman functions are strictly convex and we have linear inequality constraints, it is easy to see that DROT($\boldsymbol{a}, \boldsymbol{b}$) is a convex program. Furthermore, strong duality holds if Slater's condition *Slater* (2014) holds. Specifically, given $\boldsymbol{C}$, we need to show the existence of an $\boldsymbol{f}$ and $\boldsymbol{g}$ such that for all i, j we have that

$\boldsymbol{f}_i + \boldsymbol{g}_j < \boldsymbol{C}_{ij}$. To do so, set

$$\boldsymbol{f} = -\|\boldsymbol{C}\|_\infty \mathbf{1}_n \text{ and } \boldsymbol{g} = -\|\boldsymbol{C}\|_\infty \mathbf{1}_m.$$

Thus, we have strong duality.

Let us now compute the dual of the problem. To do so, let $\boldsymbol{P}$ be the dual variables and obtain the Lagrangian $L(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{P})$:

$$\begin{aligned} L(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{P}) = \frac{1}{\gamma}\phi(\boldsymbol{f}) + \frac{1}{\gamma}\varphi(\boldsymbol{g}) - \boldsymbol{f}^T\boldsymbol{a} - \boldsymbol{g}^T\boldsymbol{b} \\ + \langle \boldsymbol{P}, \boldsymbol{f}\mathbf{1}_m^T + \mathbf{1}_n\boldsymbol{g}^T - \boldsymbol{C}\rangle. \end{aligned} \tag{C.2}$$

Now we know that the dual problem is given by

$$\max_{\boldsymbol{P}_{ij}\geq 0} \min_{\boldsymbol{f},\boldsymbol{g}} L(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{P}). \tag{C.3}$$

Let us do some simplifications to get this into the standard form. We first note that the Lagrangian L can be rewritten as

$$\begin{aligned} L(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{P}) = \frac{1}{\gamma}\phi(\boldsymbol{f}) + \frac{1}{\gamma}\varphi(\boldsymbol{g}) - \langle \boldsymbol{f}, \boldsymbol{a} - \boldsymbol{P}\mathbf{1}_n\rangle \\ - \langle \boldsymbol{g}, \boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_m\rangle - \langle \boldsymbol{P}, \boldsymbol{C}\rangle. \end{aligned} \tag{C.4}$$

Now, for fixed $\boldsymbol{P}$ consider the function

$$F(\boldsymbol{f}) = \frac{1}{\gamma}\phi(\boldsymbol{f}) - \langle \boldsymbol{f}, \boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m\rangle.$$

Due to the strict convexity and co-finiteness of ϕ, we have that F is a strictly convex function. and has a unique stationary point that corresponds to its global minimum $\boldsymbol{f}^*$. We can solve for this as follows. For the case when we have positive co-finiteness only,

we need $\boldsymbol{a} - \boldsymbol{P} < 0$ for F to have a stationary point. Note if these conditions are not satisfied then the value of $L(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{P})$ is negative infinity, however if it is satisfied then it is a finite number. Thus, since we have the outer maximization, this is equivalent to adding the constraint.

$$0 = \nabla F(\boldsymbol{f}^*) = \frac{1}{\gamma}\nabla\phi(\boldsymbol{f}^*) - \boldsymbol{a} + \boldsymbol{P}\mathbf{1}_m.$$

Thus, we have that

$$\frac{1}{\gamma}\nabla\phi(\boldsymbol{f}^*) = \boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m.$$

Now from *Bauschke and Borwein* (1998), if we can show that ϕ is *essentially strictly convex* then due to ϕ being co-finite, we have that $\nabla\phi^*\nabla\phi(\boldsymbol{f}) = \boldsymbol{f}$. Hence via Lemma C.2, we have that

$$\boldsymbol{f}* = \nabla\phi^*\left(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m)\right)$$

Performing a similar calculation for $\boldsymbol{g}$ and substituting into Equation C.3, we get the following equation for dual.

$$\begin{aligned}\max_{\boldsymbol{P}_{ij}\geq 0} -\langle C, P\rangle + \frac{1}{\gamma}\phi(\nabla\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m))) \\ -\langle\nabla\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m)), \boldsymbol{a} - \boldsymbol{P}\mathbf{1}_n\rangle \\ +\frac{1}{\gamma}\varphi(\nabla\varphi^*(\gamma(\boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_m))) \\ -\langle\nabla\varphi^*(\gamma(\boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_m)), \boldsymbol{b} - \boldsymbol{P}^T\mathbf{1}_n\rangle\end{aligned}$$

To simplify this, *Amari* (2016) tells us that

$$\psi^*\left(\nabla\psi(x)\right) = x^T\nabla\psi(x) - \psi(x) \tag{C.5}$$

From *Rockafellar* (1970), we know that $\phi^{**} = cl(conv(\phi))$. However, since ϕ is closed and convex, we have that $\phi^{**} = \phi$. Additionally, since we also have that ϕ^* is closed and convex *Rockafellar* (1970), we also have that $\phi^{***} = \phi^*$. Thus, we have that

$$\frac{1}{\gamma}\phi(\nabla\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m))) = \langle \boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m, \nabla\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m))\rangle - \frac{1}{\gamma}\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}\mathbf{1}_m))$$

Substituting back, we get that dual of $\mathrm{DROT}(\boldsymbol{a}, \boldsymbol{b})$ is given by

$$\begin{aligned} &\text{Minimize:} \quad \langle \boldsymbol{C}, \boldsymbol{P}\rangle + \frac{\phi^*(\gamma(\boldsymbol{a}-\boldsymbol{P}\mathbf{1}_m)}{\gamma} + \frac{\varphi^*\left(\gamma(\boldsymbol{b}-\boldsymbol{P}^T\mathbf{1}_n)\right)}{\gamma} \\ &\text{Subject to:} \quad \forall i \in [n], \forall j \in [m], \quad \boldsymbol{P}_{ij} \geq 0 \end{aligned}$$

□

Remark C.1. Our proof of strong duality, as written, does not hold for the entropic regularizer. For the entropic regularized version we need to add the assumption that $\boldsymbol{C}_{ij} > 0$ for all i, j. If this is the case, then letting $\boldsymbol{f}, \boldsymbol{g} = 0$ works. Therefore, for all experiments involving the entropy regularizer, we add a small number to the cost matrix to guarantee that all the costs are positive.

We also need to add the constraint that $\boldsymbol{f}, \boldsymbol{g} \geq 0$ so, in the dual formulation, we add the dual variables $\boldsymbol{c}_1, \boldsymbol{c}_2$ that correspond to these constraints.

Lemma C.2. If ϕ is a Bregman function, then ϕ is *essentially strictly convex*

Proof. From *Rockafellar* (1970), we know that a function ϕ is *essentially strictly convex* if for all convex $S \subset \{x : \nabla\phi(x) \neq 0\} =: \mathrm{dom}(\partial\phi)$, ϕ is strictly convex on S. From *Rockafellar* (1970), we also know that $\mathrm{dom}(\partial\phi) \subset \mathrm{dom}\phi$. Thus, since Bregman functions are strictly convex, we have that ϕ is *essentially strictly convex.* □

Proposition 6.3. *Let $\boldsymbol{P}^*, \boldsymbol{f}^*, \boldsymbol{g}^*$ be the optimal solutions, primal and dual, to the Monge-Kantorovich formulation (Problem 6.0.2) and let $\boldsymbol{P}^*_{\phi,\varphi}, \boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{g}^*_{\phi,\varphi}$ be the optimal*

solutions to DROT, Problem 6.4. Then we have that the following are true.

1. *The difference between the value of the DROT objective and that of the Monge-Kantorovich formulation is upper and lower bounded by*

$$\begin{aligned}\phi\left(\boldsymbol{f}^*_{\phi,\varphi}\right) + \varphi(\boldsymbol{g}^*_{\phi,\varphi}) &\leq \gamma(OT(\boldsymbol{a},\boldsymbol{b}) - DROT(\boldsymbol{a},\boldsymbol{b})) \\ &\leq \phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*).\end{aligned}$$

2. *We can estimate the quality of the approximation (as a function of the regularizers ϕ and φ) as*

$$\begin{aligned}\gamma\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi}\rangle \leq &\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*) + \\ &\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m)) + \\ &\varphi^*(\gamma(\boldsymbol{b} - (\boldsymbol{P}^*_{\phi,\varphi})^T\mathbf{1}_n))\end{aligned}$$

3. *and*

$$\begin{aligned}&\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m)) + \varphi^*(\gamma(\boldsymbol{b} - (\boldsymbol{P}^*_{\phi,\varphi})^T\mathbf{1}_n)) \leq \\ &\gamma\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_{\phi,\varphi}\rangle - \phi(\boldsymbol{f}^*_{\phi,\varphi}) - \varphi(\boldsymbol{g}^*_{\phi,\varphi}).\end{aligned}$$

Proof. Let us first prove the lower bound for part 1. To do this note that since $\boldsymbol{f}^*_{\phi,\varphi}$ and $\boldsymbol{g}^*_{\phi,\varphi}$ satisfy the constraints $\boldsymbol{f}^*_{\phi,\varphi}\mathbf{1}_m^T + \mathbf{1}_n(\boldsymbol{g}^*_{\phi,\varphi})^T \leq \boldsymbol{C}$, we have that

$$\langle \boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{a}\rangle + \langle \boldsymbol{g}^*_{\phi,\varphi}, \boldsymbol{b}\rangle \leq \langle \boldsymbol{f}^*, \boldsymbol{a}\rangle + \langle \boldsymbol{g}^*, \boldsymbol{b}\rangle = OT(\boldsymbol{a},\boldsymbol{b})$$

Then subtracting $\frac{1}{\gamma}\phi\left(\boldsymbol{f}^*_{\phi,\varphi}\right) + \frac{1}{\gamma}\varphi\left(\boldsymbol{g}^*_{\phi,\varphi}\right)$ from both sides and rearranging gives us the the lower bound.

For the upper bound, note that $\boldsymbol{f}^* \mathbf{1}_m^T + \mathbf{1}_n (\boldsymbol{g}^*)^T \leq \boldsymbol{C}$, hence we have that

$$\frac{1}{\gamma}\phi(\boldsymbol{f}^*) + \frac{1}{\gamma}\varphi(\boldsymbol{g}^*) - (\boldsymbol{f}^*)^T \boldsymbol{a} - (\boldsymbol{g}^*)^T \boldsymbol{b} \geq -DROT(\boldsymbol{a}, \boldsymbol{b}).$$

Thus, rearranging gives us the upper bound.

Now for part 2, we have that

$$\langle \boldsymbol{C}, \boldsymbol{P}^* \rangle = \langle \boldsymbol{f}^*, \boldsymbol{a} \rangle + \langle \boldsymbol{g}^*, \boldsymbol{b} \rangle.$$

Then we subtract $(\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*))/\gamma$ from both sides to get

$$\langle \boldsymbol{C}, \boldsymbol{P}^* \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*) = \langle \boldsymbol{f}^*, \boldsymbol{a} \rangle + \langle \boldsymbol{g}^*, \boldsymbol{b} \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*).$$

Then we have that

$$\langle \boldsymbol{f}^*, \boldsymbol{a} \rangle + \langle \boldsymbol{g}^*, \boldsymbol{b} \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*) \leq DROT(\boldsymbol{a}, \boldsymbol{b}).$$

Thus, we get that

$$\begin{aligned} \langle \boldsymbol{C}, \boldsymbol{P}^* \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*) \leq \langle C, \boldsymbol{P}^*_{\phi,\varphi} \rangle \\ + \frac{\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m))}{\gamma} \\ + \frac{\varphi^*(\gamma(\boldsymbol{b} - (\boldsymbol{P}^*_{\phi,\varphi})^T \mathbf{1}_n))}{\gamma} \end{aligned}$$

Rearranging the above equation gives us part 2

For part 3, note that

$$\langle \boldsymbol{C}, \boldsymbol{P}^* \rangle = \langle \boldsymbol{f}^*, \boldsymbol{a} \rangle + \langle \boldsymbol{g}^*, \boldsymbol{b} \rangle \geq \langle \boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{a} \rangle + \langle \boldsymbol{g}^*_{\phi,\varphi}, \boldsymbol{b} \rangle.$$

Then we subtract $(\phi(\boldsymbol{f}^*_{\phi,\varphi}) + \varphi(\boldsymbol{g}^*_{\phi,\varphi}))/\gamma$ from both sides to get

$$\langle \boldsymbol{C}, \boldsymbol{P}^* \rangle - \frac{1}{\gamma}(\phi(\boldsymbol{f}^*_{\phi,\varphi}) + \varphi(\boldsymbol{g}^*_{\phi,\varphi})) \geq DROT(\boldsymbol{a}, \boldsymbol{b})$$

Substituting in the primal objective for DROT and rearranging gives us part 3. □

Corollary 6.4. *If $\boldsymbol{P}^*_\gamma$ is the solution to $DROT(\boldsymbol{a}, \boldsymbol{b})$ for a given γ, and $\boldsymbol{P}^*$ is the solution to* $\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b})$ *then, $\|\boldsymbol{a} - \boldsymbol{P}^*_\gamma \mathbf{1}_m\|$ and $\|\boldsymbol{b} - (\boldsymbol{P}^*_\gamma)^T \mathbf{1}_n\|$, $OT(\boldsymbol{a}, \boldsymbol{b}) - DROT(\boldsymbol{a}, \boldsymbol{b})$, and $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_\gamma \rangle|$ are all $O(\gamma^{-1})$.*

Proof. Note that at the optimal point, by the KKT conditions, we have stationarity. So we have that

$$\frac{1}{\gamma}\nabla\phi(\boldsymbol{f}^*_{\phi,\varphi}) = \boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m \Rightarrow \|\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m\| = \frac{1}{\gamma}\|\nabla\phi(\boldsymbol{f}^*_{\phi,\varphi})\|$$

Now due to the convexity of ϕ, and part 1 of proposition 1, we have that $\phi(\boldsymbol{f}^*_{\phi,\varphi})$ is bounded from above. Then again due to the convexity of ϕ, this implies that $\|\nabla\phi(\boldsymbol{f}^*_{\phi,\varphi})\|$ is bounded from above, Thus, $\|\boldsymbol{a} - \boldsymbol{P}^*_\gamma \mathbf{1}_m\|$ is $O(\gamma^{-1})$.

Similarly, noting that convex functions are bounded from below, due to Proposition 6.3 part 1, we have that $OT(\boldsymbol{a}, \boldsymbol{b}) - DROT(\boldsymbol{a}, \boldsymbol{b})$ is $O(\gamma^{-1})$.

Finally, since $\|\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m\|$ is bounded, we have that $\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m$ lives in a bounded set whose diameter is $O(\gamma^{-1})$. Thus, $\phi^*(\gamma(\boldsymbol{a} - \boldsymbol{P}^*_{\phi,\varphi}\mathbf{1}_m))$ is bounded. Thus, using similar reasoning to before and Proposition 6.3 parts 2,3, we have that $|\langle \boldsymbol{C}, \boldsymbol{P}^* - \boldsymbol{P}^*_\gamma \rangle|$ is $O(\gamma^{-1})$. □

Proposition 6.7. *Given two discrete measures μ, ν, a cost function c, and Bregman regularizers ϕ, φ and $\gamma^{-1} \in [0, \infty)$, the value function V is well defined and continuous on $[0, \infty)$ and the optimal policy correspondence x^* is also well defined and continuous on $(0, \infty)$. Furthermore, if ϕ, φ are both positive co-finite or both negative co-finite, then the optimal policy correspondence is upper hemicontinuous on $[0, \infty)$.*

Proof. Let us start by defining a new problem DROT_n as follows. Here we add the following new constraints: $-n \leq \boldsymbol{f}_i, \boldsymbol{g}_j$. In this case, we have that the feasible region is bounded and closed and hence is compact.

We are going to show continuity using Berge's maximal theorem. Hence we need to show the assumptions for Berge's theorem are true. Here let $K_n = \{[\boldsymbol{f}, \boldsymbol{g}] \in \mathbb{R}^{2n} : -n \leq \boldsymbol{f}_i, \boldsymbol{g}_j\}$, then we have

$$X_n = \{[\boldsymbol{f}, \boldsymbol{g}] \in \mathbb{R}^{2n} : \boldsymbol{f}_i + \boldsymbol{g}_j \leq \boldsymbol{C}_{ij}\} \cap K_n$$

This X_n will be the feasible region for the problem DROT_n. Now let $\Theta = [0, \infty)$. Now define $T : X_n \times \Theta \to \mathbb{R}$ that is defined as follows.

$$T(\boldsymbol{f}, \boldsymbol{g}, \gamma^{-1}) = \langle \boldsymbol{f}, \boldsymbol{a} \rangle + \langle \boldsymbol{g}, \boldsymbol{b} \rangle - \gamma^{-1}\phi(\boldsymbol{f}) + \gamma^{-1}\varphi(\boldsymbol{g})$$

Finally, let us define $G_n(\theta) = X_n$ for all $\theta \in \Theta$. In this case, we have that the value function is

$$V_n(\theta) = \max_{x \in G_n(\theta)} T(x, \theta),$$

and the optimal policy correspondence is

$$x_n^*(\theta) = \{x \in G_n(\theta) : T(x, \theta) = V_n(\theta)\}.$$

The first few assumption for Berge's maximal theorem are that T is a continuous function, Θ is closed and X_n is closed. These are clearly true. Thus, we just need to show that G is compact valued and continuous. First, we see that X_n is compact. Hence G is compact valued. Thus, we just need to show that G is continuous.

We shall do this by showing that G_n is upper and lower hemicontinuous.

For upper hemicontinuity, we need to show that for all $\theta \in \Theta$ that for every

sequence $(\theta_j)_{j\in\mathbb{N}}$) with $\theta_j \to \theta$ and every sequence $(x_j)_{j\in\mathbb{N}}$ with $x_j \in G_n(\theta_j)$ for all j, there exists a convergent sub-sequence x_{j_k} such that $x_{j_k} \to x \in G_n(\theta)$. In this case, since $G_n(\theta_j) = X_n$ for all θ_j, we have that $x_j \in X_n$. Then since X_n is compact, we have a convergent sub-sequence.

For lower hemicontinuity, we need to show that for all $\theta \in \Theta$, for every open set $X' \subset X_n$ with $G_n(\theta) \cap X' \neq \emptyset$, there exists a $\delta > 0$ such that for every $\theta' \in N_\delta(\theta)$, $G_n(\theta') \cap X' \neq \emptyset$. In this case, since $G_n(\theta) = X_n$ for all θ, this is trivially true.

Thus, G_n is compact valued and continuous. Thus, by the Berge's maximal theorem, we have that V_n is well defined and continuous. Also we have that x_n^* is upper hemicontinuous. Now we have that for a fixed $\theta \in (0,\infty)$, $[\boldsymbol{f},\boldsymbol{g}] \to T(\boldsymbol{f},\boldsymbol{g},\theta)$ is a strictly concave function and $G_n(\theta)$ is a convex. Thus, we have that there has a unique maximizer. Thus, $x_n^*(\theta)$ is a singleton set. Thus, being upper hemicontinuous implies continuity and that the function $\theta \mapsto [\boldsymbol{f},\boldsymbol{g}] \in x_n^*(\theta)$ is a continuous function.

Let V, x^* be the value function and optimal policy correspondence for DROT. Then we need to show that V, x^* are continuous at all $\theta \in (0,\infty)$. To do this let $\theta \in (0,\infty)$ and let $\boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{g}^*_{\phi,\varphi}$ be the optimal solutions. Then we know there exists an n such that $[\boldsymbol{f}^*_{\phi,\varphi}, \boldsymbol{g}^*_{\phi,\varphi}] \in int(K_n)$. Thus, due to the continuity of x_n^* there is a ball B around θ, such that $x_n^*(B) \subset int(K_n)$ and $x_n^* = x^*$ on B. Thus, $V = V_n$ on B. Thus, V, x^* is continuous on $(0,\infty)$. Finally part 1 of Proposition 6.3 shows that V is continuous at 0.

The final detail that we need to prove is the fact that x^* is upper hemicontinuous at 0. First, suppose both ϕ and φ are negative co-finite. Then since convex functions are bounded from below and

$$\phi(\boldsymbol{f}^*_{\phi,\varphi}) + \varphi(\boldsymbol{g}^*_{\phi,\varphi}) \leq \phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*).$$

We see that $\phi(\boldsymbol{f}^*_{\phi,\varphi})$, $\varphi(\boldsymbol{g}^*_{\phi,\varphi})$ are bounded from above. Thus, since the two functions

are negative co-finite, there exists an N such that, $N \leq \boldsymbol{f}, \boldsymbol{g}$. Thus, we see that, $x^* = x_N^*$. Thus, we have upper hemi-continuous at 0.

Let us now suppose that both ϕ and φ are negative co-finite. Then since convex functions are bounded from below and

$$\phi(\boldsymbol{f}_{\phi,\varphi}^*) + \varphi(\boldsymbol{g}_{\phi,\varphi}^*) \leq \phi(\boldsymbol{f}^*) + \varphi(\boldsymbol{g}^*).$$

We see that $\phi(\boldsymbol{f}_{\phi,\varphi}^*)$, $\varphi(\boldsymbol{g}_{\phi,\varphi}^*)$ are bounded from above. Thus, since the two functions are negative co-finite, there exists an N such that, $N \geq \boldsymbol{f}, \boldsymbol{g}$.

Now we know that at $\gamma^{-1} \to 0$, we have that

$$\|\langle \boldsymbol{f}^* - \boldsymbol{f}_{\phi,\varphi}^*, \boldsymbol{a}\rangle + \langle \boldsymbol{g}^* - \boldsymbol{g}_{\phi,\varphi}^*, \boldsymbol{b}\rangle\| \to 0.$$

Thus now assume for the sake of contradiction that

$$\boldsymbol{f}_{\phi,\varphi}^* \to -\infty$$

as $\gamma^{-1} \to 0$. Then we have that

$$\langle \boldsymbol{f}^* - \boldsymbol{f}_{\phi,\varphi}^*, \boldsymbol{a}\rangle \to -\infty$$

as $\gamma^{-1} \to 0$. Thus, we must have that

$$\langle \boldsymbol{g}^* - \boldsymbol{g}_{\phi,\varphi}^*, \boldsymbol{b}\rangle \to \infty$$

as $\gamma^{-1} \to 0$. But then this would imply that $\boldsymbol{g}_{\phi,\varphi}^* \to \infty$ as $\gamma^{-1} \to 0$. This is a contradiction. Thus $\boldsymbol{f}_{\phi,\varphi}^*$ is bounded from below.

Similarly, we have that $\boldsymbol{g}_{\phi,\varphi}^*$ is bounded from below. Thus, there exists an N such that $x^* = x_N^*$. Thus, x^* is upper hemicontinuous at 0.

□

Corollary 6.8. *Suppose that we have an instance of Problem 6.0.2 such that for any two optimal dual solutions* $(\boldsymbol{f}_1^*, \boldsymbol{g}_1^*), (\boldsymbol{f}_2^*, \boldsymbol{g}_2^*)$*, we have that* $\boldsymbol{f}_1^* - \boldsymbol{f}_2^* = c\mathbf{1}$*, and* $\boldsymbol{g}_1^* - \boldsymbol{g}_2^* = -c\mathbf{1}$*. Then there exists* Γ *such that for all* $\gamma \geq \Gamma$*, if* $\boldsymbol{P}_\gamma^*$ *is the solution to DROT Problem 6.4 for* γ *and* $\boldsymbol{P}^*$ *is any optimal solution to Problem 6.0.2, then we have that* $supp(\boldsymbol{P}_\gamma^*) \subset supp(\boldsymbol{P}^*)$*.*

Proof. First, we note that the solution to the optimal transport problem is now unique upto constants. Then due to the existence of strictly complementary solutions. We see that all solution must be strictly complementary.

Let $\boldsymbol{f}^*, \boldsymbol{g}^*$ be the optimal solutions to the regularized problem. Then we know that $\text{supp}(\boldsymbol{P}^*) = \{i, j : \boldsymbol{f}_i^* + \boldsymbol{g}_j^* < \boldsymbol{C}_{i,j}\}$. Then there is an $\epsilon > 0$, such that for any non active constraint we have that

$$\boldsymbol{f}_i^* + \boldsymbol{g}_j^* - \boldsymbol{C}_{ij} < -\epsilon$$

Then let $V = \{\boldsymbol{f}, \boldsymbol{g} : \|\boldsymbol{f}^* - \boldsymbol{f}\| < \epsilon/3, \|\boldsymbol{g} - \boldsymbol{g}^*\| < \epsilon/3\}$. Then by upper continuity we know that there exists a $\delta > 0$ such that for all $\gamma^{-1} < \delta$ we have that $\boldsymbol{f}_{\phi,\varphi}^*, \boldsymbol{g}_{\phi,\varphi}^* \in V$. Thus, by complementary slackness we have the needed result.

□

C.2 Algorithmic Details

C.2.1 Calculating θ

For quadratic, we have that $\theta = \frac{\boldsymbol{C}_{ij} - \boldsymbol{f}_i - \boldsymbol{g}_j}{2\gamma}$, in the case of entropy we have that

$$\theta = \log\left(\frac{\boldsymbol{C}_{ij}}{\boldsymbol{f}_i + \boldsymbol{g}_j}\right)/\gamma,$$

in the case of exponential it is given by

$$\theta = -\frac{e^{\boldsymbol{f}_i} + e^{\boldsymbol{g}_j} \pm \sqrt{(e^{\boldsymbol{f}_i} + e^{\boldsymbol{g}_j})^2 - 4(e^{\boldsymbol{f}_i+\boldsymbol{g}_j} - e^{\boldsymbol{C}_{ij}})}}{2}$$

For exponential, this is done by solving the Lagrange multiplier problem. However, this problem doesn't always a solution in this set up. Such a situation arises when we set γ to be large. Hence, we don't make γ too large in any of our experiments.
In the case, we want to mix, then this calculation becomes more difficult. For example, if ϕ is quadratic and φ is entropy, then θ is the root of $e^x + x + \boldsymbol{f}_i + \boldsymbol{g}_j - \boldsymbol{C}_{ij}$.

C.3 Experiment Details

All experiments were run on a machine with 8 cores and 56 GB of memory.

C.3.1 Solver choice

For this experiment, we took two Gaussian distributions with means ± 15 and variance 10. We then sampled n equidistant points on $[-20, 20]$ and formed two discrete distributions on these n by sampling from the Gaussians. The cost matrix $\boldsymbol{C}$ is given by the squared Euclidean distance. We then solved the quadratic regularized version of the problem with $\gamma = 1e3$.

The feasibility error for Mosek and CPLEX are those reported by the solvers. For project and forget, we calculate the feasibility error by

$$\max_{i,j} \frac{\boldsymbol{f}_i + \boldsymbol{g}_- \boldsymbol{C}_{ij}}{2\gamma}.$$

As we can see from Table C.1, the solvers have roughly reached the same level of convergence. One thing of note, is that the Mosek solver consistently has very different objective values compared to the other solvers.

Solver	n	Objective Primal	Objective Dual	Feasibility Error Primal	Feasibility Error Dual
Project and Forget	501	3.8416076	3.8416077	7.8e-09	0
Mosek Primal	501	3.8414023	3.8414023	3.8e-08	2.8e-10
LBFGSB	501	n/a	3.8416114	n/a	0
Mosek Dual	501	3.8303160	3.8303203	3.5e-8	2.2e-11
CPLEX Dual	501	3.8416376	3.8416076	8.44e-07	1.13e-04
CPLEX Primal	501	Ran out of memory			
Project and Forget	1001	1.947531924	1.947532070	9.3e-9	0
Mosek Primal	1001	1.947091229	1.947091203	2.7e-08	8.4e-11
LBFGSB	1001	n/a	1.947548404	n/a	0
Mosek Dual	1001	Ran out of memory			
CPLEX Dual	1001	Ran out of memory			
CPLEX Primal	1001	Ran out of memory			
Project and Forget	5001	3.94655624e-01	3.946556176e-01	1.42e-09	0
Mosek Primal	5001	3.880175376e-01	3.880175255e-01	1.2e-08	7.4e-11
LBFGSB	5001	n/a	3.947709104e-01	n/a	0
Mosek Dual	5001	Ran out of memory			
CPLEX Dual	5001	Ran out of memory			
CPLEX Primal	5001	Ran out of memory			

Table C.1: Table showing the convergence details for the various solvers.

All experiments were run on a machine with 8 cores and 56 GB of memory.

C.3.2 Verifying theoretical properties

Here all experiments were run until the project and forget feasibility error was smaller than 1e-15.

C.3.3 Color Transfer

Here we used $k = 4096$ clusters. For the quadratic regularizer $\gamma = 1e4$, for the entropic regularizer $\gamma = 1e4$, for the exponential regularizer, $\gamma = 10^{log_10(e^10)} \approx 10^{4.34}$. Here we picked γ that looked best for the first set of images and used the same γ for the second set.

For ROT we set $\gamma = 1e-2$. For UOT we set the regularizer $\gamma_1 = 1e-2$, and we

set the penalty $\gamma_3 = \gamma_2 = 1e1$.

C.3.4 MNIST-USPS

For the quadratic regularizer, we set $\gamma = 1e7$, the entropic regularizer we set $\gamma = 1e5$. These were the smallest γ's at which transport happened. For ROT and UOT we set $\gamma = \gamma_1 = \gamma_2 = \gamma_3 = 1$.

Note γ was finalized before we looked at any of the digits or the prediction accuracy. It was chosen whenever the transport plan $\boldsymbol{P}$ had non trivial number of non-zero entries.

APPENDIX D

How to Optimally Train a Stacked Linear Denoisng Autoencoder?

D.1 Proofs

In this section we present all of the proofs for the results in the main text. Here we present the proofs in the same order they appear in the text.

D.1.1 Step 1: Formula for W_{opt}

Proposition 8.2. *Let* $h = v_{trn}^T A_{trn}^\dagger$, $k = A_{trn}^\dagger u$, $s = (I - A_{trn}A_{trn}^\dagger)u$, $t = v_{trn}(I - A_{trn}^\dagger A_{trn})$, $\beta = 1 + \theta_{trn} v_{trn}^T A_{trn}^\dagger u$, $\sigma_1 = \theta_{trn}^2 \|t\|^2 \|k\|^2 + \beta^2$, *and* $\sigma_2 = \theta_{trn}^2 \|s\|^2 \|h\|^2 + \beta^2$. *If* $\beta \neq 0$ *and* A_{trn} *has full rank then*

$$W_{opt} = \begin{cases} \frac{\theta_{trn}\beta}{\sigma_1} uh + \frac{\theta_{trn}^2 \|t\|^2}{\sigma_1} uk^T A_{trn}^\dagger & c < 1 \\ \frac{\theta_{trn}\beta}{\sigma_2} uh + \frac{\theta_{trn}^2 \|h\|^2}{\sigma_2} us^T & c > 1 \end{cases}.$$

Proof. Let us first proof the case when $c > 1$. Here we know that u is arbitrary. Here we have that A_{trn} has full rank. Thus, since $c > 1$, we have that $M > N_{trn}$, thus A_{trn}

has rank N_{trn}. Thus, the rows of A_{trn} span the whole space. Thus, v_{trn} lives in the range of A_{trn}^T. Finally, since $\beta \neq 0$, we want Theorem 5 from *Meyer* (1973).

Here let us further define

$$p_2 = -\frac{\theta_{trn}^2 \|s\|^2}{\beta} A_{trn}^\dagger h^T - \theta_{trn} k \text{ and } q_2^T = -\frac{\theta_{trn}\|h\|^2}{\beta} s^T - h$$

and finally $\sigma_2 = \theta_{trn}^2 \|s\|^2 \|h\|^2 + \beta^2$. Then we have from *Meyer* (1973) that

$$(A_{trn} + \theta_{trn} u v_{trn}^T)^\dagger = A_{trn}^\dagger + \frac{\theta_{trn}}{\beta} A_{trn}^\dagger h^T s^T - \frac{\beta}{\sigma_2} p_2 q_2^T$$

In our case, we only care about $\theta_{trn} u v_{trn}^T (A_{trn} + \theta_{trn} u v_{trn}^T)^\dagger$. Thus let us multiply this through and see what we get.

$$\begin{aligned}
\theta_{trn} u v_{trn}^T (A_{trn} + \theta_{trn} u v_{trn}^T)^\dagger &= \theta_{trn} u v_{trn}^T (A_{trn}^\dagger + \frac{\theta_{trn}}{\beta} A^\dagger h^T u^T - \frac{\beta}{\sigma_2} p_2 q_2^T) \\
&= \theta_{trn} u h + \frac{\theta_{trn}^2 \|h\|^2}{\beta} u s^T + \frac{\theta_{trn}\beta}{\sigma_2} u v_{trn}^T \left(\frac{\theta_{trn}^2 \|s\|^2}{\beta} A_{trn}^\dagger h^T + \theta_{trn} k \right) q_2^T \\
&= \theta_{trn} u h + \frac{\theta_{trn}^2 \|h\|^2}{\beta} u s^T + \frac{\theta_{trn}^3 \|s\|^2 \|h\|^2}{\sigma_2} u q_2^T + \frac{\theta_{trn}^2 \beta}{\sigma_2} u h u q_2^T
\end{aligned}$$

Then we have that

$$\frac{\theta_{trn}^3 \|s\|^2 \|h\|^2}{\sigma_2} c q_2^T = -\frac{\theta_{trn}^4 \|s\|^2 \|h\|^4}{\sigma_2 \beta} u s^T - \frac{\theta_{trn}^3 \|s\|^2 \|h\|^2}{\sigma_2} u h \tag{D.1}$$

and

$$\frac{\theta_{trn}^2 \beta}{\sigma_2} u h u q_2^T = -\frac{\theta_{trn}^3 \|h\|^2}{\sigma_2} u h u s^T - \frac{\theta_{trn}^2 \beta}{\sigma_2} u h u h. \tag{D.2}$$

Using that $\beta - 1 = \theta_{trn} v_{trn}^T A_{trn}^\dagger u = \theta_{trn} hu$, we get that

$$\frac{\theta_{trn}^2 \beta}{\sigma_2} uhuq_2^T = -\frac{\theta_{trn}^2 \|h\|^2 (\beta - 1)}{\sigma_2} us^T - \frac{\theta_{trn} \beta (\beta - 1)}{\sigma_2} uh. \tag{D.3}$$

Substituting back in and collecting like terms we get that

$$\begin{aligned}\theta_{trn} uv_{trn}^T (A_{trn} + \theta_{trn} uv_{trn}^T)^\dagger = \theta_{trn} u \left(1 - \frac{\theta_{trn}^2 \|s\|^2 \|h\|^2}{\sigma_2} - \frac{\beta(\beta-1)}{\sigma_2}\right) h \\ + \theta_{trn}^2 u \left(\frac{\|h\|^2}{\beta} - \frac{\theta_{trn}^2 \|s\|^2 \|h\|^4}{\sigma_2 \beta} - \frac{\|h\|^2 (\beta - 1)}{\sigma_2}\right) s^T\end{aligned}$$

We can then simplify the constants as follows.

$$1 - \frac{\theta_{trn}^2 \|s\|^2 \|h\|^2}{\sigma_2} - \frac{\beta(\beta-1)}{\sigma_2} = \frac{\sigma_2 - \theta_{trn}^2 \|s\|^2 \|h\|^2 - \beta^2 + \beta}{\sigma_2} = \frac{\beta}{\sigma_2}$$

and

$$\frac{\|h\|^2}{\beta} - \frac{\theta_{trn}^2 \|s\|^2 \|h\|^4}{\sigma_2 \beta} - \frac{\|h\|^2 (\beta - 1)}{\sigma_2} = \frac{\|h\|^2 (\sigma_2 - \theta_{trn}^2 \|s\|^2 \|h\|^2 - \beta(\beta - 1)}{\beta \sigma_2} = \frac{\|h\|^2 \beta}{\beta \sigma_2} = \frac{\|h\|^2}{\sigma_2}.$$

This gives us the result for $c < 1$.

If $c > 1$, then we have that $M < N_{trn}$. Thus, the rank of A_{trn} is M the range of A_{trn} is the whole space. Thus, u lives in the range of A_{trn}. In this case, we then want Theorem 3 from *Meyer* (1973). In this case, we define

$$p_1 = -\frac{\theta_{trn}^2 \|k\|^2}{\beta} t^T - k \text{ and } q_1^T = -\frac{\theta_{trn} \|t\|^2}{\beta} k^T A_{trn}^\dagger - h.$$

Then in this case, we have that

$$(A_{trn} + \theta_{trn} uv_{trn}^T)^\dagger = A_{trn}^\dagger + \frac{\theta_{trn}}{\beta} t^T k^T A_{trn}^\dagger - \frac{\beta}{\sigma_1} p_1 q_1^T.$$

Then we simplify the equation as we did before! □

D.1.2 Step 2: Formula for the Expected MSE

Lemma 8.3. *If A_{tst} has mean 0 entries and A_{tst} is independent of X_{tst} and W, then*

$$\mathbb{E}_{A_{tst}}[\|X_{tst} - WY_{tst}\|_F^2] = \mathbb{E}_{A_{tst}}[\|X_{tst} - WX_{tst}\|_F^2] + \mathbb{E}_{A_{tst}}[\|WA_{tst}\|_F^2].$$

Proof. Using the fact that for any two matrices $\|G-H\|_F^2 = \|G\|_F^2+\|H\|_F^2-2\text{Tr}(G^TH)$, we get that

$$\begin{aligned}\|X_{tst} - WY_{tst}\|^2 &= \|X_{tst} - WX_{tst} - WA_{tst}\|_F^2 \\ &= \|X_{tst} - WX_{tst}\|_F^2 + \|WA_{tst}\|^2 - 2\text{Tr}((X_{tst} - WX_{tst})^TWA_{tst}).\end{aligned}$$

Then since the trace is linear, and X_{tst}, W are independent of A_{tst}, and A_{tst} has mean 0 entries, we see that

$$\mathbb{E}_{A_{tst}}[\text{Tr}((X_{tst} - WX_{tst})^TWA_{tst})] = 0.$$

Thus, we have the needed result. □

Lemma 8.4. *If the entries of A_{tst} are independent with mean 0, and variance $1/M$, then we have that $\mathbb{E}_{A_{tst}}[\|WA_{tst}\|^2] = \frac{N_{tst}}{M}\|W\|^2$.*

Proof. To see this, we note if we look at $A_{tst}A_{tst}^T$, then this is a M by M, for which the expected value of the off diagonal entries is equal to 0, while the expected value of each diagonal entry is N_{tst}/M. That is, $\mathbb{E}_{A_{tst}}[A_{tst}A_{tst}^T] = \frac{N_{tst}}{M}I_M$.

Then note that

$$\|WA_{tst}\|^2 = \text{Tr}(A_{tst}^TW^TWA_{tst}) = \text{Tr}(W^TWA_{tst}A_{tst}^T) = \text{Tr}(W^TWA_{tst}A_{tst}^T).$$

Using the fact that the trace is linear again, we see that

$$\mathbb{E}_{A_{tst}}[\mathrm{Tr}(W^T W A_{tst} A_{tst}^T)] = \mathrm{Tr}(W^T W \mathbb{E}_{A_{tst}}[A_{tst} A_{tst}^T]) = \frac{N_{tst}}{M}\mathrm{Tr}(W^T W) = \frac{N_{tst}}{M}\|W\|_F^2.$$

□

Lemma 8.5. *If W is the solution to Equation 8.1, then*

$$X_{tst} - WX_{tst} = \begin{cases} \frac{\beta}{\sigma_1} X_{tst} & \text{if } c < 1 \\ \frac{\beta}{\sigma_2} X_{tst} & \text{if } c > 1 \end{cases}.$$

Proof. To see this, we have the following calculation for when $N_{trn} > M$.

$$\begin{aligned} X_{tst} - WX_{tst} &= X_{tst} - \frac{\theta_{trn}\theta_{tst}\beta}{\sigma_1} uhuv_{tst}^T - \frac{\theta_{trn}^2\theta_{tst}\|t\|^2}{\sigma_1} ck^T A_{trn}^\dagger uv_{tst}^T \\ &= X_{tst} - \frac{\theta_{trn}\theta_{tst}\beta}{\sigma_1} uv_{trn}^T A_{trn}^\dagger uv_{tst}^T - \frac{\theta_{trn}^2\theta_{tst}\|t\|^2}{\sigma_1} uk^T A_{trn}^\dagger uv_{tst}^T. \end{aligned}$$

First, we note that $\beta = 1 + \theta_{trn} v_{trn}^T A_{trn}^\dagger u$. Thus, we have that $\theta v_{trn}^T A_{trn}^\dagger u = \beta - 1$. Thus, substituting this into the second term, we get that

$$X_{tst} - WX_{tst} = X_{tst} - \frac{\theta_{tst}\beta(\beta-1)}{\sigma_1} uv_{tst}^T - \frac{\theta_{trn}^2\theta_{tst}\|t\|^2}{\sigma_1} uk^T A_{trn}^\dagger uv_{tst}^T.$$

For the third term, we note that $k = A_{trn}^\dagger u$. Thus, we have that $k^T A_{trn}^\dagger u = k^T k = \|k\|^2$. Substituting this into the expression, we get that

$$X_{tst} - WX_{tst} = X_{tst} - \frac{\theta_{tst}\beta(\beta-1)}{\sigma_1} uv_{tst}^T - \frac{\theta_{trn}^2\theta_{tst}\|t\|^2\|k\|^2}{\sigma_1} uv_{tst}^T.$$

Noting that $X_{tst} = \theta_{tst} u v_{tst}^T$, we get that

$$X_{tst} - W X_{tst} = X_{tst} \left(1 - \frac{\beta(\beta - 1)}{\sigma_1} - \frac{\theta_{trn}^2 \|t\|^2 \|k\|^2}{\sigma_1} \right).$$

To simplify the constants, we note that $\sigma_1 = \theta_{trn}^2 \|t\|^2 \|k\|^2 + \beta^2$. Thus, we get that

$$\frac{\sigma_1 + \beta - \beta^2 - \theta_{trn}^2 \|t\|^2 \|k\|^2}{\sigma_1} = \frac{\beta}{\sigma_1}.$$

For the case when $N_{trn} < M$, we note that the first term of W is the same (modulo replacing σ_1 for σ_2) as it is for the case when $c > 1$. Thus, we just need to deal with the last term. Here we see that the last term is

$$\frac{\theta_{trn}^2 \theta_{tst} \|h\|^2}{\sigma_2} u s^T u v_{tst}^T.$$

Here we note that $s = (I - A_{trn} A_{trn}^\dagger) u$. Thus, in particular, s is the projection of u onto the kernel of A_{trn}^T. Thus, we have that $u = s + \hat{s}$, where $s \perp \hat{s}$. This then tells us that $s^T u = \|s\|^2$. Thus, for this term, we get that it is equal to

$$\frac{\theta^2 \|h\|^2 \|s\|^2}{\sigma_2} X_{tst}.$$

For this term we note that $\sigma_2 = \beta^2 + \theta^2 \|h\|^2 \|u\|^2$. Thus, doing the same simplification as before, we see that for the case when $N_{trn} < M$, we have that

$$X_{tst} - W X_{tst} = \frac{\beta}{\sigma_2} X_{tst}.$$

□

In light of Lemma 8.5 and the fact that $\|X_{tst}\|_F^2 = \theta_{tst}^2$. We see that if we look at

the expected MSE, we have that,

$$\mathbb{E}_{A_{tst}}\left[\frac{\|X_{tst} - W(X_{tst} + A_{tst})\|}{N_{tst}}\right] = \frac{\beta}{N_{tst}\sigma_i}\theta_{tst}^2 + \frac{1}{M}\|W\|_F^2,$$

where σ_i depends on whether $c < 1$ or $c > 1$.

Finally, let us look at the $\|W\|$ term.

Lemma 8.6. *If $\beta \neq 0$ and A_{trn} has full rank, then we have that if $c < 1$,*

$$\|W\|_F^2 = \frac{\theta_{trn}^2\beta^2}{\sigma_1^2}Tr(h^T h) + 2\frac{\theta_{trn}^3\|t\|^2\beta}{\sigma_1^2}Tr(h^T k^T A_{trn}^\dagger) + \frac{\theta_{trn}^4\|t\|^4}{\sigma_1^2}Tr((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger)$$

and if $c > 1$, then we have that

$$\|W\|_F^2 = \frac{\theta_{trn}^2\beta^2}{\sigma_2^2}Tr(h^T h) + 2\frac{\theta_{trn}^3\|h\|^2\beta}{\sigma_2^2}Tr(h^T s^T) + \frac{\theta_{trn}^4\|h\|^4}{\sigma_2^2}Tr(s s^T).$$

Proof. To deal with the term $\mathrm{Tr}(W^T W)$ we are again going to have to look at whether N_{trn} is bigger than or smaller than M. First, let us start by looking at the case when $N_{trn} > M$. Here we have that

$$\begin{aligned}
\|W\|_F^2 &= \mathrm{Tr}(W^T W) \\
&= \mathrm{Tr}\left(\left(\frac{\theta_{trn}\beta}{\sigma_1}uh + \frac{\theta_{trn}^2\|t\|^2}{\sigma_1}uk^T A_{trn}^\dagger\right)^T\left(\frac{\theta_{trn}\beta}{\sigma_1}uh + \frac{\theta_{trn}^2\|t\|^2}{\sigma_1}uk^T A_{trn}^\dagger\right)\right) \\
&= \frac{\theta_{trn}^2\beta^2}{\sigma_1^2}\mathrm{Tr}(h^T u^T uh) + 2\frac{\theta_{trn}^3\|t\|^2\beta}{\sigma_1^2}\mathrm{Tr}(h^T u^T uk^T A_{trn}^\dagger) + \frac{\theta_{trn}^4\|t\|^4}{\sigma_1^2}\mathrm{Tr}((A_{trn}^\dagger)^T k u^T uk^T A_{trn}^\dagger) \\
&= \frac{\theta_{trn}^2\beta^2}{\sigma_1^2}\mathrm{Tr}(h^T h) + 2\frac{\theta_{trn}^3\|t\|^2\beta}{\sigma_1^2}\mathrm{Tr}(h^T k^T A_{trn}^\dagger) + \frac{\theta_{trn}^4\|t\|^4}{\sigma_1^2}\mathrm{Tr}((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger).
\end{aligned}$$

Where the last inequality is true due to the fact that $\|u\|^2 = 1$. How about when $N_{trn} < M$. Then we have the following string of equalities instead.

$$
\begin{aligned}
\|W\|_F^2 &= \mathrm{Tr}(W^T W) \\
&= \mathrm{Tr}\left(\left(\frac{\theta_{trn}\beta}{\sigma_2}uh + \frac{\theta_{trn}^2\|h\|^2}{\sigma_2}us^T\right)^T\left(\frac{\theta_{trn}\beta}{\sigma_2}uh + \frac{\theta_{trn}^2\|h\|^2}{\sigma_2}us^T\right)\right) \\
&= \frac{\theta_{trn}^2\beta^2}{\sigma_2^2}\mathrm{Tr}(h^T u^T uh) + 2\frac{\theta_{trn}^3\|h\|^2\beta}{\sigma_2^2}\mathrm{Tr}(h^T u^T us^T) + \frac{\theta_{trn}^4\|h\|^4}{\sigma_1^2}\mathrm{Tr}(su^T us^T) \\
&= \frac{\theta_{trn}^2\beta^2}{\sigma_2^2}\mathrm{Tr}(h^T h) + 2\frac{\theta_{trn}^3\|h\|^2\beta}{\sigma_2^2}\mathrm{Tr}(h^T s^T) + \frac{\theta_{trn}^4\|h\|^4}{\sigma_2^2}\mathrm{Tr}(ss^T).
\end{aligned}
$$

□

D.1.3 Step 3: Estimate using random matrix theory.

Lemma 8.7. *Suppose A is an p by q matrix such that the entries of A are independent and have mean 0, variance $1/q$, and bounded fourth moment. Let $W_p = AA^T$ and let $W_q = A^T A$. Let $C = p/q$. Suppose λ_p, λ_q are a random eigenvalue of W_p, W_q. Then*

1. *If $p < q$, then $\mathbb{E}\left[\frac{1}{\lambda_p}\right] = \frac{1}{1-C} + o(1)$.*
2. *If $p < q$, then $\mathbb{E}\left[\frac{1}{\lambda_p^2}\right] = \frac{1}{(1-C)^3} + o(1)$.*
3. *If $p > q$, then $\mathbb{E}\left[\frac{1}{\lambda_q}\right] = \frac{C^{-1}}{1-C^{-1}} + o(1)$.*
4. *If $p > q$, then $\mathbb{E}\left[\frac{1}{\lambda_q^2}\right] = \frac{C^{-2}}{(1-C^{-1})^3} + o(1)$.*

Proof. Suppose A is an p by q matrix such that the entries of A are independent and have mean 0, variance $1/q$, and bounded fourth moment. Then we know that $W_p = AA^T$ is an p by p Wishart matrix with $c = C$. If we send p, q to infinity such that p/q remains constant, then we have the eigenvalue distribution F_p converges to the Marchenko Pastur distribution F in probability.

From *Rao and Edelman* (2008), we know there exists a bi variate polynomial $L(m, z) = czm^2 - (1 - c - z)m + 1$ such that the zeros of $L(m, z)$ given by $L(m(z), z)$ are such that

$$
m(z) = \int \frac{1}{\lambda - z} dF(\lambda) = \mathbb{E}_\lambda\left[\frac{1}{\lambda - z}\right].
$$

For the Marchenko-Pastur distribution, we have that for $z = 0$, we get that $m(z) = 1/(1-c)$. Thus, for λ_p is an eigenvalue value of W_p, we have that

$$\mathbb{E}\left[\frac{1}{\lambda_p}\right] = \frac{1}{1-c} + o(1).$$

For $\mathbb{E}_\lambda\left[\frac{1}{(\lambda-z)^2}\right]$ we need to calculate $m'(0)$. Using the implicit function theorem, we know that

$$m'(z) = -1\left(\frac{\partial L}{\partial m}(m(z), z)\right)^{-1}\frac{\partial L}{\partial z}(m(z), z).$$

Here we can see that $\partial L/\partial m = 2czm + c + z - 1$. Thus, at $(1/(1-c), 0)$, this is equal to $c - 1$. Also $\partial L/\partial z = cm^2 + m$. Again at $(1/(1-c), 0)$ this is equal to $\frac{c}{(1-c)^2} + \frac{1}{1-c} = \frac{1}{(1-c)^2}$. Thus, we have that

$$m'(0) = \frac{1}{(1-c)^3}.$$

On the other hand if $q < p$, then $W_q := A^T A$ is not a Wishart matrix here, because it is scaled by the wrong constant. However, multiplying it by $1/C$ gives us the correct scaling. Thus, $A^T A/$ is a Wishart matrix with $c = 1/C$ Thus, for λ_q is an eigenvalue value of W_q, we have that

$$\mathbb{E}\left[\frac{1}{\lambda_q}\right] = \frac{C^{-1}}{1-C^{-1}} + o(1).$$

□

Lemma 8.8. *Suppose A is an p by q matrix that satsifies the standard noise assumptions. Let x, y be unit vectors in p and q dimensions. Let $C = p/q$. Then*

1. $\mathbb{E}[Tr(x^T(AA^T)^\dagger x)] = \begin{cases} \frac{1}{1-C} + o(1) & p < q \\ \frac{q}{p}\frac{C^{-1}}{1-C^{-1}} + o(1) & p > q \end{cases}$.

2. $\mathbb{E}[Tr(x^T(AA^T)^\dagger(AA^T)^\dagger x)] = \begin{cases} \frac{1}{(1-C)^3} + o(1) & p < q \\ \frac{q}{p}\frac{C^{-2}}{(1-C^{-1})^3} + o(1) & p > q \end{cases}$.

3. $\mathbb{E}[Tr(y^T(A^TA)^\dagger y)] = \begin{cases} \frac{p}{q}\frac{1}{1-C} + o(1) & p < q \\ \frac{C^{-1}}{1-C^{-1}} + o(1) & p > q \end{cases}$.

4. $\mathbb{E}[Tr(y^T(A^TA)^\dagger(A^TA)^\dagger y)] = \begin{cases} \frac{p}{q}\frac{1}{(1-C)^3} + o(1) & p < q \\ \frac{C^{-2}}{(1-C^{-1})^3} + o(1) & p > q \end{cases}$.

Proof. Let $A = U\Sigma V^T$ be the SVD. Then we have that $(AA^T)^\dagger = U(\Sigma^2)^\dagger U^T$. Then since A is bi-unitary invariant, we have that U is a uniformly random unitary matrix. Thus, $a = x^TU$ is a uniformly random unit vector. Note with probability 1, the rank of A is full and that the non-zero eigenvalues of A^TA and AA^T are the same.
If $p < q$, then we have that

$$\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger x)] = \sum_{i=1}^{p} a_i^2 \frac{1}{\sigma_i^2}.$$

Using Lemma 8.7, we have that $\mathbb{E}[1/\sigma_i^2] = 1/(1-C) + o(1)$. Thus, we have that

$$\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger x)] = \sum_{i=1}^{p} \frac{1}{p}\frac{1}{1-C} + o(1).$$

On the other hand, if $p > q$, from Lemma 8.7, we have that $\mathbb{E}[1/\sigma_i^2] = C^{-1}/(1-C^{-1}) + o(1)$. Thus,

$$\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger x)] = \sum_{i=1}^{q} \frac{1}{p}\frac{C^{-1}}{1-C^{-1}} + o(1).$$

Similarly, if we had we looking at $\mathrm{Tr}(x^T(AA^T)^\dagger(AA^T)^\dagger x)$, we would have a $1/\sigma_i^4$

term instead. Thus, if $p < q$, we would have that

$$\mathbb{E}[\mathrm{Tr}(x^T(AA^T)^\dagger(AA^T)^\dagger x)] = \frac{1}{(1-C)^3} + o(1).$$

A similar calculation holds for the others. □

Now we have the following Lemma in the main text. However, here instead of having one big proof, we will separate each term out into its own lemma.

Lemma 8.9. *If A_{trn} satisfies the standard noise assumptions, then we have that*

1. $\mathbb{E}[\beta] = 1 + o(1)$ *and* $\mathrm{Var}(\beta) = \frac{\theta_{trn}^2 c}{(max(M,N_{trn})|1-c|))} + o(1)$.
2. *If* $c < 1$, *then* $\mathbb{E}[\|h\|^2] = \dfrac{c^2}{1-c} + o(1)$ *and* $\mathrm{Var}(\|h\|^2) = \dfrac{c^3(2+c)}{N_{trn}(1-c)^3} + o(1)$.
3. *If* $c > 1$, *then* $\mathbb{E}[\|h\|^2] = \dfrac{c}{c-1} + o(1)$ *and* $\mathrm{Var}(\|h\|^2) = \dfrac{c^2(4-c)}{M(c-1)^3} + o(1)$.
4. $\mathbb{E}[\|k\|^2] = \dfrac{c}{1-c} + o(1)$ *and* $\mathrm{Var}(\|k\|^2) = \dfrac{2-c}{M(1-c)^3} + o(1)$.
5. $\mathbb{E}[\|s\|^2] = \dfrac{c-1}{c} + o(1)$ *and* $\mathrm{Var}(\|s\|^2) = 2\dfrac{(c-1)^2}{Mc^2} + o(1)$
6. $\mathbb{E}[\|t\|^2] = 1 - c + o(1)$, $\mathrm{Var}(\|t\|^2) = 2\dfrac{(1-c)^2}{N_{trn}} + o(1)$.

Lemma D.1. β term.

Proof. First, we calculate the expected value of β. To do so, let $A_{trn} = U\Sigma V^T$ be the SVD. Then since A_{trn} is bi-unitarily invariant, we have that U, V are uniformly random unitary matrices. Since u, v_{trn} are fixed. We have that $a := v_{trn}^T V \in \mathbb{R}^{Ntrn}$ and $b := U^T u \in \mathbb{R}^M$ are uniformly random unit vectors. In particular, we have that $\mathbb{E}[a_i] = 0, \mathbb{E}[b_i] = 0, \mathrm{Var}(a_i) = 1/N_{trn}, \mathrm{Var}(b_i) = 1/M$.

Thus, if σ_i are the singular values for A_{trn}, then we have that

$$\beta = 1 + \theta_{trn} \sum_{i=1}^{\min(M,N_{trn})} \frac{1}{\sigma_i} a_i b_i.$$

Thus, if you take the expectation you get that

$$\mathbb{E}[\beta] = 1.$$

On the other hand, lets look at the variance. For the variance, we need to compute $\mathbb{E}[\beta^2]$. Now if we let $T := \theta_{trn} v_{trn}^T A_{trn}^\dagger u$. Then we have that

$$\beta^2 = 1 + T^2 + 2T.$$

Thus, again if we take the expectation, we get that

$$\mathbb{E}[\beta^2] = 1 + \mathbb{E}[T^2].$$

Again due to the fact that a, b are independent have have mean 0 entries, the cross terms in $\mathbb{E}[T^2]$. Thus, we have that

$$\mathbb{E}[T^2] = \theta trn^2 \mathbb{E}\left[\sum_{i=1}^{\min(M,N_{trn})} \frac{1}{\sigma_i^2} a_i^2 b_i^2\right] = \theta trn^2 \frac{1}{MN_{trn}} \mathbb{E}\left[\sum_{i=1}^{\min(M,N_{trn})} \frac{1}{\sigma_i^2}\right].$$

Now we need to case on whether $M > N_{trn}$ or $M < N_{trn}$. Now to use Lemma 8.7, we note that $q = M$ and $p = N_{trn}$.

Suppose we have that $M > N_{trn}$, then in this case, we have that $q > p$. Thus, we have that

$$\mathbb{E}\left[\frac{1}{\sigma_i^2}\right] = \frac{1}{1-C} + o(1),$$

where $C = p/q = N_{trn}/M = 1/c$. Thus, we have that

$$\mathbb{E}\left[\frac{1}{\sigma_i^2}\right] = \frac{1}{1-1/c} + o(1) = \frac{c}{c-1} + o(1).$$

Thus, we have that

$$\mathbb{E}[T^2] = \theta trn^2 \frac{c}{M(c-1)} + o\left(\frac{1}{M}\right).$$

Thus, we have

$$\mathrm{Var}(\beta) = \theta trn^2 \frac{c}{M(c-1)} + o\left(\frac{1}{M}\right).$$

On the other hand, if $M < N_{trn}$. Then we have that $q < p$. Thus, we have that

$$\mathbb{E}\left[\frac{1}{\sigma_i^2}\right] = \frac{C^{-1}}{1 - C^{-1}} + o(1),$$

where $C = p/q = N_{trn}/M = 1/c$. Thus, we have that

$$\mathbb{E}\left[\frac{1}{\sigma_i^2}\right] = \frac{c}{1-c} + o(1).$$

Thus, we have that

$$\mathbb{E}[T^2] = \theta trn^2 \frac{1}{N_{trn}} \left(\frac{c}{1-c} + o(1)\right) = \frac{c}{N_{trn}(1-c)} + o\left(\frac{1}{N_{trn}}\right).$$

Thus, we have

$$\mathrm{Var}(\beta) = \theta trn^2 \frac{c}{N_{trn}(1-c)} + o\left(\frac{1}{N_{trn}}\right).$$

□

Lemma D.2. $\|h\|^2$ term.

Proof. We want to do a calculation similar to that in Lemma 1. Here we have that

$$\|h\|^2 = \mathrm{Tr}(h^T h) = \mathrm{Tr}((A_{trn}^\dagger)^T v_{trn} v_{trn}^T A_{trn}^\dagger) = \mathrm{Tr}(v_{trn}^T A_{trn}^\dagger (A_{trn}^\dagger)^T v_{trn}) = \mathrm{Tr}(v_{trn}^T (A_{trn}^T A_{trn})^\dagger v_{trn}).$$

To use Lemma 8.8, we note that $A = A_{trn}^T$, $q = M$, $p = N_{trn}$. Let us now suppose that

$M < N_{trn}$. Then again taking the expectation, we see that

$$\mathbb{E}[\|h\|^2] = \frac{M}{N_{trn}}\left(\frac{c}{1-c} + o(1)\right) = \frac{c^2}{1-c} + o(1).$$

For the expectation of $\|h\|^4$, let $A_{trn} = U\Sigma V^T$ be the svd. Then $h = v_{trn}^T V \Sigma^\dagger U^T$. Let $a = v_{trn}^T V$ and note that a is a uniformly random unit vector. Thus, we have that

$$\|h\|^2 = \sum_{i=1}^{M} \frac{1}{\sigma_i^2} a_i^2.$$

For the expectation of $\|h\|^4$, we note that

$$\|h\|^4 = \sum_{i=1}^{M}\sum_{j=1}^{M} \frac{1}{\sigma_i^2\sigma_j^2} a_i^2 a_j^2 = \sum_{i=1}^{M} \frac{1}{\sigma_i^4} a_i^4 + \sum_{i\neq j} \frac{1}{\sigma_i^2}\frac{1}{\sigma_j^2} a_i^2 a_j^2.$$

Taking the expectation of the first term, we get

$$\sum_{i=1}^{M} \mathbb{E}\left[\frac{1}{\sigma_i^4}\right]\mathbb{E}[a_i^4] = \frac{3M}{N_{trn}^2}\left(\frac{c^2}{(1-c)^3} + o(1)\right) = 3\frac{c^3}{N_{trn}(1-c)^3} + o(1).$$

Taking the expectation of the second term, we get

$$M(M-1)\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]^2 \mathbb{E}[a_i^2]^2 = M(M-1)\frac{1}{N_{trn}^2}\left(\frac{c^2}{(1-c)^2} + o(1)\right) = \frac{c^4}{(1-c)^2} - \frac{c^3}{N_{trn}(1-c)^2} + o(1).$$

Thus, we have that

$$\mathbb{E}[\|h\|^4] = \frac{c^4}{(1-c)^2} + \frac{c^3(2+c)}{N_{trn}(1-c)^3} + o(1).$$

Thus, the variance is

$$\mathrm{Var}(\|h\|^2) = \frac{c^3(2+c)}{N_{trn}(1-c)^3} + o(1).$$

For $M > N_{trn}$, we instead have that

$$\mathbb{E}[\|h\|^2] = \frac{N_{trn}}{N_{trn}}\left(\frac{c}{c-1} + o(1)\right) = \frac{c}{c-1} + o(1).$$

For the expectation of $\|h\|^4$, we note that

$$\|h\|^4 = \sum_{i=1}^{N_{trn}}\sum_{j=1}^{N_{trn}} \frac{1}{\sigma_i^2\sigma_j^2}a_i^2a_j^2 = \sum_{i=1}^{N_{trn}} \frac{1}{\sigma_i^4}a_i^4 + \sum_{i\neq j} \frac{1}{\sigma_i^2}\frac{1}{\sigma_j^2}a_i^2a_j^2.$$

Taking the expectation of the first term, we get

$$\sum_{i=1}^{N_{trn}} \mathbb{E}\left[\frac{1}{\sigma_i^4}\right]\mathbb{E}[a_i^4] = \frac{3N_{trn}}{N_{trn}^2}\left(\frac{c^2}{(c-1)^3} + o(1)\right) = 3\frac{c^2}{N_{trn}(c-1)^3} + o(1).$$

Taking the expectation of the second term, we get

$$\begin{aligned} N_{trn}(N_{trn}-1)\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]^2\mathbb{E}[a_i^2]^2 &= N_{trn}(N_{trn}-1)\frac{1}{N_{trn}^2}\left(\frac{c^2}{(c-1)^2} + o(1)\right) \\ &= \frac{c^2}{(c-1)^2} - \frac{c^2}{N_{trn}(c-1)^2} + o(1). \end{aligned}$$

Thus, we have that

$$\mathbb{E}[\|h\|^4] = \frac{c^2}{(c-1)^2} + \frac{c^2(4-c)}{N_{trn}(c-1)^3} + o(1).$$

Thus, the variance is

$$\mathrm{Var}(\|h\|^2) = \frac{c^2(4-c)}{N_{trn}(c-1)^3} + o(1).$$

□

Lemma D.3. $\|k\|^2$ term.

Proof. First note that k only appears in the formula when $c < 1$. Thus, we can focus

on this case. As with h, we have that

$$\|k\|^2 = \mathrm{Tr}(u^T(A_{trn}^\dagger)^T A_{trn}^\dagger u) = \mathrm{Tr}(u^T(A_{trn}A_{trn}^T)^\dagger u).$$

Again using Lemma 8.8, with $q = M, p = N_{trn}, A = A_{trn}, y = u$. Thus, since we have $q = M < N_{trn} = p$, we get that

$$\mathbb{E}[\|k\|^2] = \frac{c}{1-c} + o(1).$$

To calculate the variance, we need to calculate the expectation of $\|k\|^4$. Here be again let $A = U\Sigma V^T$ be the SVD. Then let $b := U^T u$. Then we have that

$$\|k\|^2 = \sum_{i=1}^{M} \frac{1}{\sigma_i^2} b_i^2.$$

Thus, we see that

$$\|k\|^4 = \sum_{i=1}^{M} \frac{1}{\sigma_i^4} b_i^4 + \sum_{i \neq j} \frac{1}{\sigma_i^2}\frac{1}{\sigma_j^2} b_i^2 b_j^2.$$

Taking the expectation of the first term we get

$$3\frac{M}{M^2}\frac{c^2}{(1-c)^3} = \frac{3c^2}{M(1-c)^3}.$$

Taking the expectation of the second term we get

$$\frac{M(M-1)}{M^2}\frac{c^2}{(1-c)^2} = \frac{c^2}{(1-c)^2} - \frac{c^2}{M(1-c)^2}.$$

Thus, we have that

$$\mathbb{E}[\|k\|^4] = \frac{c^2}{(1-c)^2} + \frac{2-c}{M(1-c)^3} + o(1).$$

Thus, we have that

$$\mathrm{Var}(\|k\|^2) = \frac{2-c}{M(1-c)^3} + o(1).$$

□

Lemma D.4. $\|s\|^2$ term.

Proof. First, we note that s only appears when $M > N_{trn}$. Thus, we only need to deal with that case. For this term, we note that $(I - A_{trn}A_{trn}^{\dagger})$ is a projection matrix onto a uniformly random $M - N_{trn}$ dimensional subspace. Thus, we have that

$$\mathbb{E}[\|s\|^2] = \frac{M - N_{trn}}{M} = \frac{c-1}{c}.$$

Similarly, we have that

$$\begin{aligned}
\mathbb{E}[\|s\|^4] &= \sum_{i=1}^{M} \mathbb{E}[s_i^4] + \sum_{i \neq j} \mathbb{E}[s_i^2]\mathbb{E}[s_j^2] \\
&= 3M\frac{(M-N_{trn})^2}{M^4} + M(M-1)\frac{(M-N_{trn})^2}{M^4} + o(1) \\
&= 3\frac{(c-1)^2}{Mc^2} + \frac{(c-1)^2}{c^2} - \frac{(c-1)^2}{Mc^2} + o(1) \\
&= \frac{(c-1)^2}{c^2} + \frac{2(c-1)^2}{Mc^2} + o(1).
\end{aligned}$$

Thus, we have that

$$\mathrm{Var}(\|s\|^2) = 2\frac{(c-1)^2}{Mc^2} + o(1).$$

□

Lemma D.5. $\|t\|^2$ term.

Proof. First, we note that t only appears when $M < N_{trn}$. Thus, we only need to deal with that case. For this term, we note that $(I - A_{trn}^{\dagger}A_{trn})$ is a projection matrix onto

a uniformly random $N_{trn} - M$ dimensional subspace. Thus, we have that

$$\mathbb{E}[\|t\|^2] = \frac{N_{trn} - M}{N_{trn}} = 1 - c.$$

Similarly, we have that

$$\begin{aligned}
\mathbb{E}[\|t\|^4] &= \sum_{i=1}^{N_{trn}} \mathbb{E}[t_i^4] + \sum_{i \neq j} \mathbb{E}[t_i^2]\mathbb{E}[t_j^2] \\
&= 3N_{trn}\frac{(N_{trn} - M)^2}{N_{trn}^4} + N_{trn}(N_{trn} - 1)\frac{(N_{trn} - M)^2}{N_{trn}^4} + o(1) \\
&= 3\frac{(1-c)^2}{N_{trn}} + (1-c)^2 - \frac{(1-c)^2}{N_{trn}} + o(1) \\
&= (1-c)^2 + \frac{2(1-c)^2}{N_{trn}} + o(1).
\end{aligned}$$

Thus, we have that

$$\mathrm{Var}(\|t\|^2) = 2\frac{(1-c)^2}{N_{trn}} + o(1).$$

□

Now we could just use the the fact that $|\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]| < \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}$. Another way to do this is via using big O in probability. Which is defined as follows:

Definition D.6. We save that a sequence of random variables X_n is $O_P(a_n)$, if there exists an N such that for all $\epsilon > 0$, there exists a constant L such that for all $n \geq N$, we have that $\Pr[|X_n| > La_n] < \epsilon$.

Using this, we can have see that σ terms are as follows. Here the big O for terms come from the variance.

Lemma D.7. σ_1 term.

Proof. We know that we only use σ_1 when $M < N_{trn}$. In this case, we have that

$\sigma_1 = \beta^2 + \theta_{trn}^2 \|t\|^2 \|k\|^2$.

$$\sigma_1 = 1 + O_P\left(\frac{\theta_{trn}^2 c}{N_{trn}(1-c)}\right) + \theta_{trn}^2\left((1-c) + O_P\left(2\frac{(1-c)^2}{N_{trn}}\right)\right)\left(\frac{c}{1-c} + O_P\left(\frac{2-c}{M(1-c)^3}\right)\right)$$

Thus, we get that $\sigma_1 = 1 + \theta_{trn}^2 c + O_P(E_{\sigma_1})$, where

$$E_{\sigma_1} = \frac{\theta_{trn}^2 c}{N_{trn}(1-c)} + 2\frac{\theta_{trn}^2 c(1-c)}{N_{trn}} + \frac{\theta_{trn}^2(2-c)}{M(1-c)^2} + \frac{2\theta_{trn}^2(2-c)}{N_{trn}M(1-c)}.$$

□

Lemma D.8. σ_2 term.

Proof. We know that we only use σ_2 when $M > N_{trn}$. In this case, we have that $\sigma_2 = \beta^2 + \theta_{trn}^2 \|s\|^2 \|h\|^2$.

$$\sigma_1 = 1 + O_P\left(\frac{\theta_{trn}^2 c}{N_{trn}(1-c)}\right) + \theta_{trn}^2\left(\frac{c-1}{c} + O_P\left(2\frac{(c-1)^2}{Mc^2}\right)\right)\left(\frac{c}{c-1} + O_P\left(\frac{c^2(4-c)}{N_{trn}(c-1)^3}\right)\right)$$

Thus, we get that $\sigma_2 = 1 + \theta_{trn}^2 + O_P(E_{\sigma_2})$, where

$$E_{\sigma_2} = \frac{\theta_{trn}^2 c}{N_{trn}(1-c)} + 2\frac{\theta_{trn}^2(c-1)}{Mc} + \frac{\theta_{trn}^2 c(4-c)}{N_{trn}(c-1)^2} + \frac{2\theta_{trn}^2(4-c)}{N_{trn}M(c-1)}.$$

□

Then the trace terms.

Lemma 8.10. *Under standard noise assumptions, we have that*

$$\mathbb{E}[\,Tr(h^T k^T A_{trn}^\dagger)] = 0$$

and

$$\mathrm{Var}(\,Tr(h^T k^T A_{trn}^\dagger)) = \chi_3(c)/N_{trn},$$

where $\chi_3(c) = \mathbb{E}[1/\lambda^3]$, λ *is an eigenvalue for* AA^T *and* A *is as in Lemma 8.8.*

Proof. First we note that

$$\mathrm{Tr}(h^T k^T A_{trn}^\dagger) = \mathrm{Tr}((A_{trn}^\dagger)^T v_{trn} u^T (A_{trn}^\dagger)^T A_{trn}^\dagger) = u^T (A_{trn}^\dagger)^T A_{trn}^\dagger A_{trn}^\dagger)^T v_{trn}).$$

Again let $A_{trn} = U\Sigma V^T$ be the SVD. Then, we have the middle terms depending on A_{trn} simplifies to

$$(A_{trn}^\dagger)^T A_{trn}^\dagger (A_{trn}^\dagger)^T = U(\Sigma^\dagger)^T \Sigma^\dagger (\Sigma^\dagger)^T V^T.$$

Thus, again letting $b = u^T U$ and $a = V^T v_{trn}$. We see that

$$\mathrm{Tr}(h^T k^T A_{trn}^\dagger) = \sum_{i=1}^{M} a_i b_i \frac{1}{\sigma_i^3}.$$

Now if take the expectation, since a, b are independent and mean 0, we see that

$$\mathbb{E}_{A_{trn}}[\mathrm{Tr}(h^T k^T A_{trn}^\dagger)] = 0.$$

Let us also compute the variance. Here we have that

$$\mathbb{E}[\mathrm{Tr}(h^T k^T A_{trn}^\dagger)^2] = \sum_{i=1}^{M} \mathbb{E}\left[\frac{1}{\sigma_i^6}\right] \mathbb{E}[a_i^2]\mathbb{E}[b_i^2] + 0.$$

Now for the Marchenko Pastur distribution we have that the expectation of $1/\lambda^3 = \chi_3(c)$. where χ_3 is some function. Thus, we have that

$$\mathbb{E}[\mathrm{Tr}(h^T k^T A_{trn}^\dagger)^2] = \frac{1}{N_{trn}} \chi_3(c) + o(1).$$

□

Lemma 8.11. *Under standard noise assumptions, we have that*

$$Tr((A_{trn}^\dagger)^T k k^T A_{trn}^\dagger) = \frac{c^2}{(1-c)^3} + o(1)$$

and

$$\text{Var}(\, Tr((A_{trn}^{\dagger})^T k k^T A_{trn}^{\dagger})) = \frac{3}{M}\chi_4(c) - \frac{1}{M}\frac{c^4}{(1-c)^6}$$

where $\chi_4(c) = \mathbb{E}[1/\lambda^4]$, λ *is an eigenvalue for* AA^T *and* A *is as in Lemma 8.8.*

Proof. Now using Lemma 8.8, we see that

$$\mathbb{E}_{A_{trn}}[\text{Tr}((A_{trn}^{\dagger})^T k k^T A_{trn}^{\dagger})] = \frac{c^2}{(1-c)^3}.$$

Similar to proofs before, we have that

$$\mathbb{E}_{A_{trn}}[\text{Tr}((A_{trn}^{\dagger})^T k k^T A_{trn}^{\dagger})^2] = \sum_{i=1}^{M} \frac{3}{M^2}\chi_4(c) + \sum_{i \neq j} \frac{1}{M^2}\frac{c^4}{(1-c)^6} + o(1).$$

Where $\chi_4(c) = \mathbb{E}[1/\lambda^4]$ for the Marchenko Pastur distribution. Thus, we have that

$$\text{Var}(\text{Tr}((A_{trn}^{\dagger})^T k k^T A_{trn}^{\dagger})) = \frac{3}{M}\chi_4(c) - \frac{1}{M}\frac{c^4}{(1-c)^6} + o(1).$$

□

Lemma 8.12. *Under the same assumptions as Proposition 8.2, we have that* $Tr(h^T s^T) = 0$.

Proof. Here we note that $h^T = (A_{trn}^{\dagger})^T v_{trn}$ and $s^T = u^T(I - A_{trn}A_{trn}^{\dagger})^T$. Thus, we have that

$$\begin{aligned}
\text{Tr}(h^T s^T) &= \text{Tr}((A_{trn}^{\dagger})^T v_{trn} u^T - (A_{trn}^{\dagger})^T v_{trn} u^T (A_{trn}A_{trn}^{\dagger})^T) \\
&= \text{Tr}(v_{trn}^T A_{trn}^{\dagger} u) - \text{Tr}(u^T (A_{trn}A_{trn}^{\dagger})^T (A_{trn}^{\dagger})^T v_{trn}) \\
&= \text{Tr}(v_{trn}^T A_{trn}^{\dagger} u) - \text{Tr}(v_{trn}^T A_{trn}^{\dagger} A_{trn} A_{trn}^{\dagger} u) \\
&= \text{Tr}(v_{trn}^T A_{trn}^{\dagger} u) - \text{Tr}(v_{trn}^T A_{trn}^{\dagger} u) \\
&= 0
\end{aligned}$$

□

As we can see that if we take the expectation of $\|W\|$ over A_{trn}, since the variance of each of the terms is small, we can approximate $\mathbb{E}[XY]$ with $\mathbb{E}[X]\mathbb{E}[Y]$. Then we get the following.

If $M < N_{trn}$, we have that

$$\begin{aligned}\mathbb{E}_{A_{trn}}[\|W\|^2] &= \frac{\theta_{trn}^2}{(1+\theta_{trn}^2 c)^2}\frac{c^2}{(1-c)} + 0 + \frac{\theta_{trn}^4(1-c)^2}{(1+\theta_{trn}^2 c)^2}\frac{c^2}{(1-c)^3} \\ &= c^2\frac{\theta_{trn}^2+\theta_{trn}^4}{(1+\theta_{trn}^2 c)^2(1-c)}.\end{aligned}$$

On the other hand, $M > N_{trn}$, we have that

$$\begin{aligned}\mathbb{E}_{A_{trn}}[\|W\|^2] &= \frac{\theta_{trn}^2}{(1+\theta_{trn}^2)^2}\frac{c}{c-1} + \frac{\theta_{trn}^4}{(1+\theta_{trn}^2)^2}\frac{c^2}{(c-1)^2}\frac{c-1}{c} \\ &= \frac{c}{c-1}\frac{\theta_{trn}^2(1+\theta_{trn}^2)}{(1+\theta_{trn}^2)^2} \\ &= \frac{\theta_{trn}^2}{1+\theta_{trn}^2}\frac{c}{c-1}.\end{aligned}$$

Now combining everything together, we get that

$$\mathbb{E}_{A_{trn},A_{tst}}\left[\frac{\|X_{tst} - W(X_tst + A_{tst})\|}{N_{tst}}\right] = \begin{cases} \frac{\theta_{tst}^2}{N_{tst}(1+\theta_{trn}^2 c)^2} + \frac{1}{M}c^2\frac{\theta_{trn}^2+\theta_{trn}^4}{(1+\theta_{trn}^2 c)^2(1-c)} & c < 1 \\ \frac{\theta_{tst}^2}{N_{tst}(1+\theta_{trn}^2 c)^2} + \frac{1}{M}\frac{\theta_{trn}^2}{1+\theta_{trn}^2}\frac{c}{c-1} & c > 1 \end{cases}.$$

D.1.4 Proof of Theorem

We can see that the main text has how to put all of the pieces together to prove the main Theorem. We don't replicate that here.

D.1.5 Formula for $\hat{\theta}_{opt-trn}$

As stated in the main text, we only need to take the derivative. So, we don't present that calculation here as it is fairly straightforward.

D.2 Experiments

We now present extra details and graphs to support the experiments.

D.2.1 Flag Experiment

Here, we took a 176 by 179 pixel image of the German flag. It was a png hence has 4 channels. We concatenate all of the channels to get a 4×176 by 179 matrix. We then replicate each data point 4 times, so we get a 708 by 718 data matrix. In principle this matrix should be rank 1, but due to real world data issues, the matrix was rank 7.

Here we added Gaussian noise sampled from a standard Gaussian distribution. This results in us having $\hat{\theta}_{tst} \approx 168$. Using these numbers we compute $\hat{\theta}_{opt-trn} \approx 30$. This is what we use to train the second denoiser.

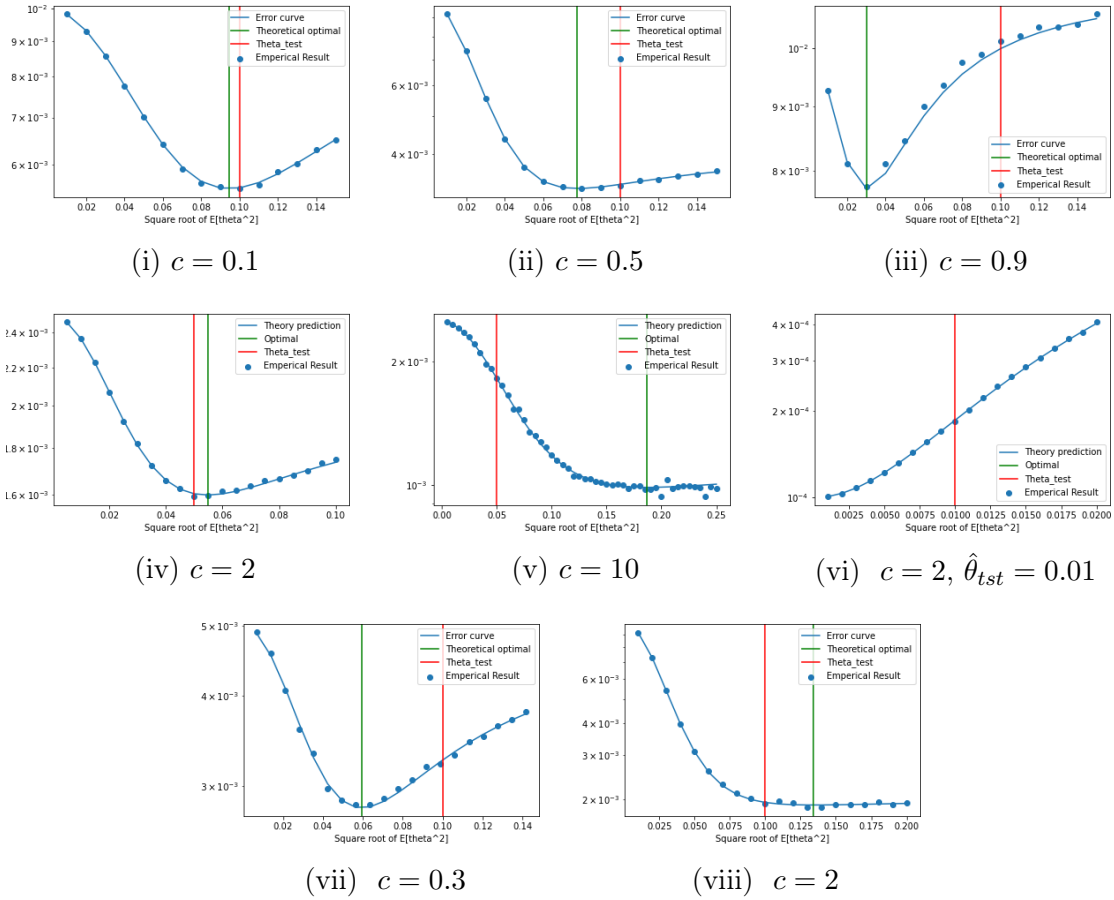
Here the traditional autoencoder has an error of approximately 220 while our decoder has an error of approximately 190. The best MSE for traditional was 200 and the best MSE for our autoencoder was 160.

These experiments were run on a free Google Colab instance. The experiment was run multiple times. In many cases, we have that the traditional autoencoder has fewer vertical lines that are wrong, but at the top of the yellow strip, there are usually a horizontal strip of corrupted pixels that doesn't exist for our autoencoder.

D.2.2 Linear Autoencoder

Here we provide more examples of c and how our theoretical formula matches the experimental performance exactly.

Each empirical point is the average over 50 trials. These were run on a laptop with 8gb of RAM and an i3 processors. The average time to produce any of these plots is about 10 to 30 minutes.



(i) $c = 0.1$ (ii) $c = 0.5$ (iii) $c = 0.9$

(iv) $c = 2$ (v) $c = 10$ (vi) $c = 2$, $\hat{\theta}_{tst} = 0.01$

(vii) $c = 0.3$ (viii) $c = 2$

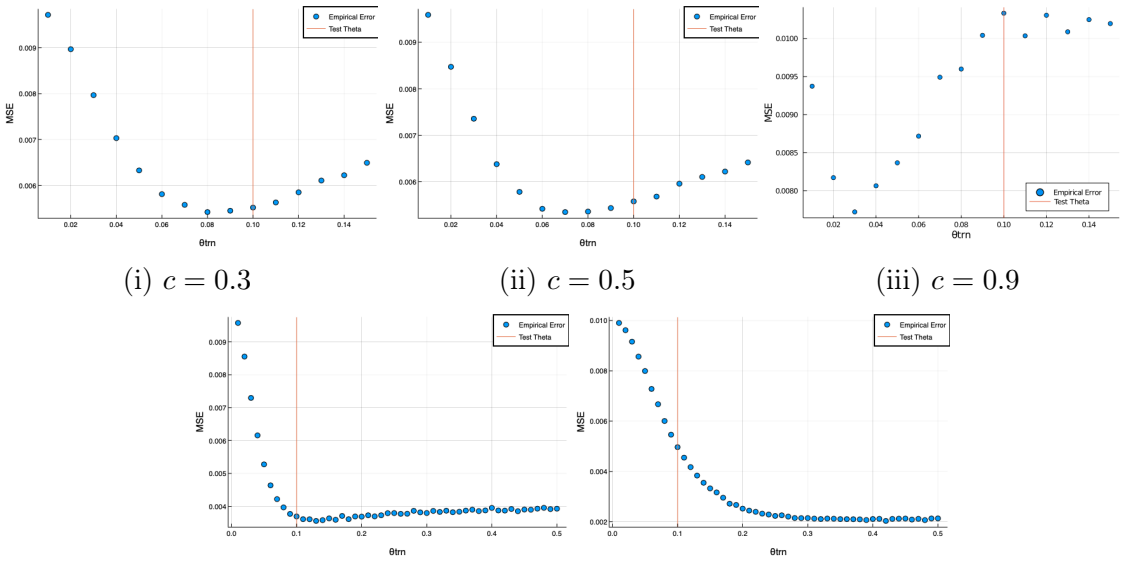
Figure D.1: Figures (a) - (e) showing the accuracy of the formula for the expected mean squared error for $c = 0.1, 0.5, 0.9, 2, 10$ for fixed value of $\hat{\theta}_{tst}$. Figure (f) empirically verifies the existence of a regime where training on pure noise is optimal. Figures (g) and (h) show the accuracy of the formula when D_{tst} (exponential in (g) and Gaussian in (h)) and D_{trn} (uniform in (g) and exponential in (h)) are non constant distributions. Here the red and green lines represent $\mathbb{E}[\hat{\theta}^2_{tst}]$ and $\mathbb{E}[\hat{\theta}^2_{trn}]$ respectively. Each empirical data point is averaged over at least 50 trials.

D.2.3 Rank 2 Data

Let us now demonstrate that the V shaped curve exists beyond rank 1 data and linear autoencoders. We will do this by gradually making the set up more complicated until we can no longer recreate this phenomena. First, we consider rank 2 data is of the following form. Let W_{data} be some fixed matrix, then our data is generated by

$$X = \texttt{relu}(W_{data}\texttt{relu}(uv^T).$$

Where a different v is sampled for the training and test data. the results for this can be seen in Figure D.2. As we can from the figure, we have the exact same qualitative trend for c that we saw before. That is, as c goes from 0 to 1, we have that $\hat{\theta}_{trn}$ goes from $\hat{\theta}_{tst}$ to 0, and then as $c \to \infty$, we have that $\hat{\theta}_{trn}$ goes to infinity as well.



(i) $c = 0.3$ (ii) $c = 0.5$ (iii) $c = 0.9$

(iv) $c = 2$ (v) $c = 10$

Figure D.2: Rank 2

D.2.4 MNIST Data

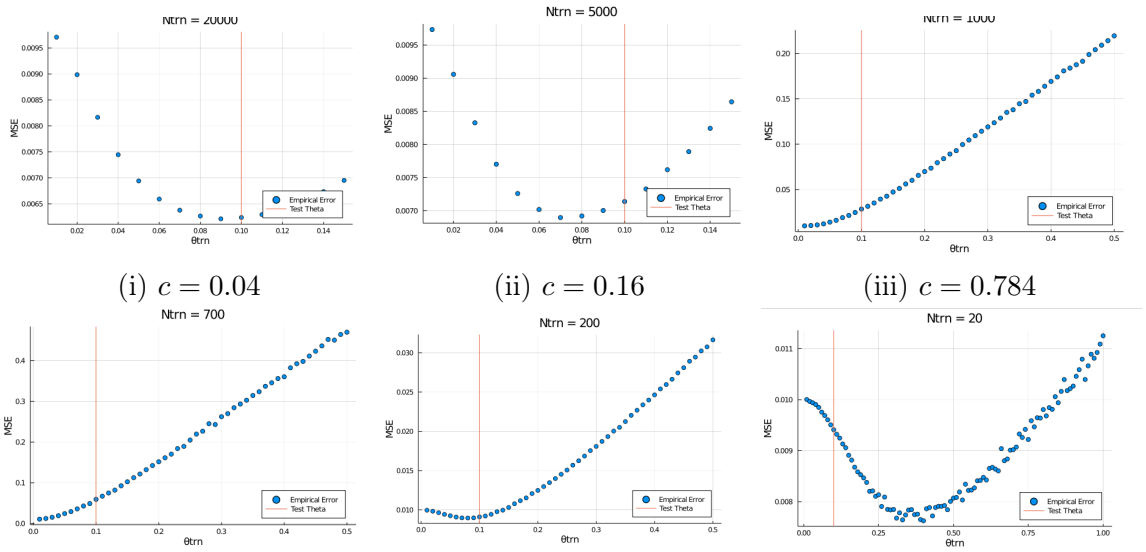
D.2.4.1 Linear Autoencoder

This experiment was run on laptop with 8gb of RAM and an i3 processor. Each empirical data point on each was run 5 times.

D.2.4.2 Non-linear Autoencoder

Here, we trained each network for 1500 epochs. During each epoch we computed a gradient using the whole data set. We used Adam as the optimizer with the code written in Pytorch. Each data point was generated over 20 trials.

These experiments take a little bit more time to run and the one with bigger

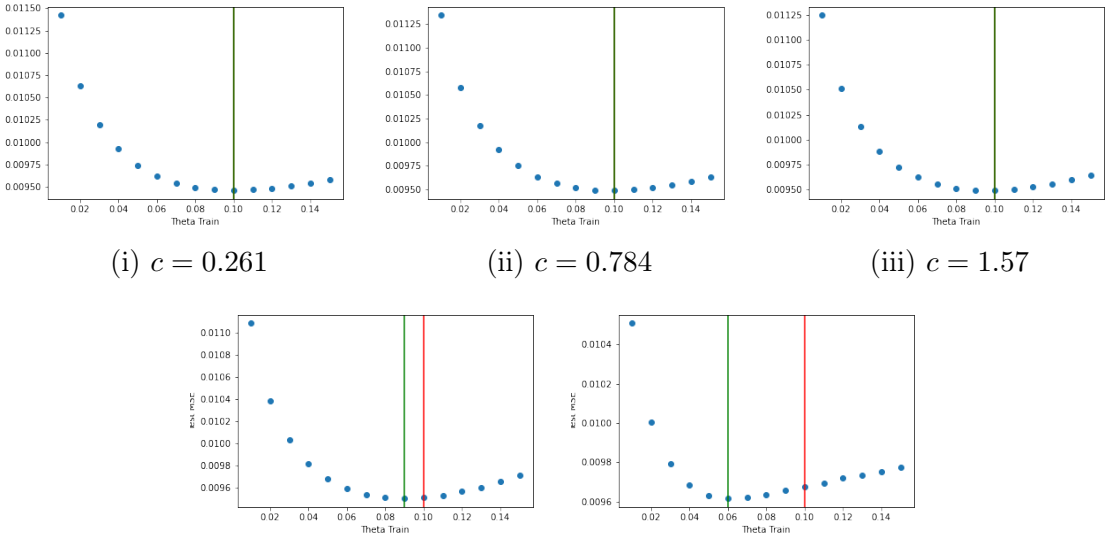


(i) $c = 0.04$ (ii) $c = 0.16$ (iii) $c = 0.784$

(iv) $c = 1.12$ (v) $c = 3.92$ (vi) $c = 39.2$

Figure D.3: MNIST

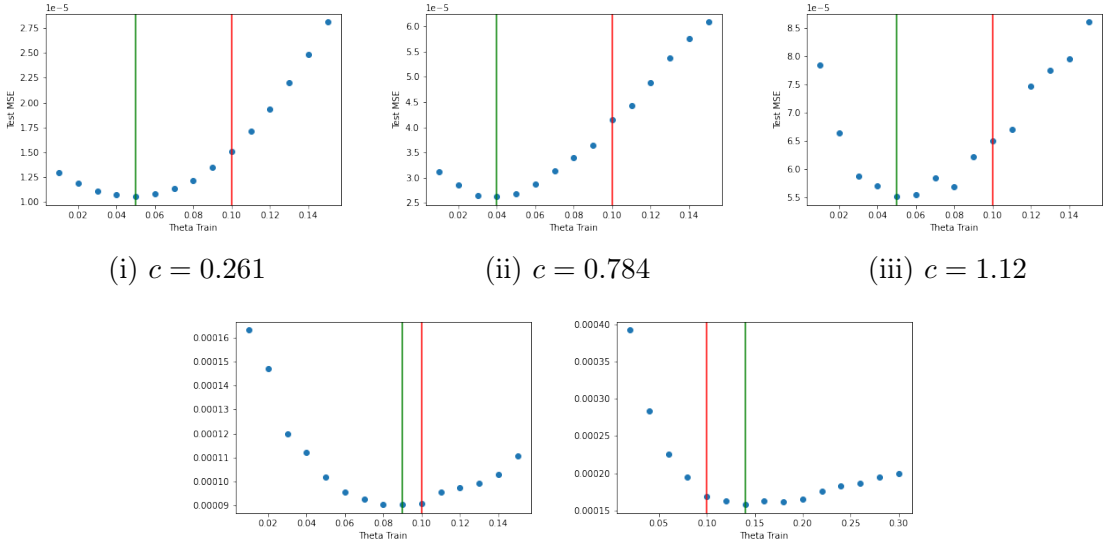
amounts of data can take upto 5 hours on a google cloud instance with 16gb RAM. Here we used a Telse P4 gpu.



(i) $c = 0.261$ (ii) $c = 0.784$ (iii) $c = 1.57$

(iv) $c = 3.92$ (v) $c = 39.2$

Figure D.4: MNIST - LSLS model



(i) $c = 0.261$ (ii) $c = 0.784$ (iii) $c = 1.12$

(iv) $c = 1.57$ (v) $c = 3.92$

Figure D.5: MNIST - LRL model

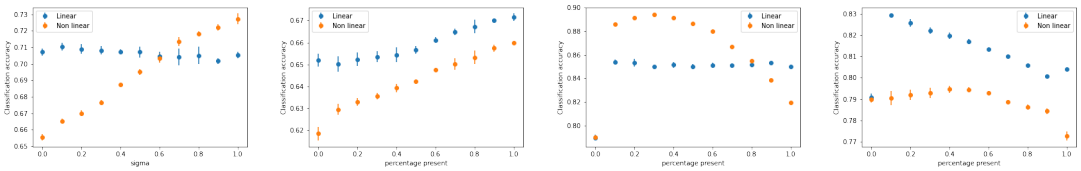
D.3 Pre-training SDAEs

From the previous section, we see that we understand the phenomenon that occurs when using a linear autoencoder. However, we do not fully understand the phenomenon for a non-linear autoencoder. Hence, we want to see if understanding the linear version is enough. That is, suppose we use the linear autoencoders to initialize an SDAE, how does this compare to the non-linear version? To test this, we consider two different noise models. First is Gaussian noise, and the second is salt and pepper noise added only to the background. For salt and pepper noise, each pixel of the background is set to 1 with probability 0.25, is set to -1 with probability 0.25, and we leave it to be 0 with probability 0.5. Let A be the noise matrix for either noise model.

For each noise model, let X_{trn}, X_{tst} be the true MNIST data. Then the corrupted datasets are $\hat{X}_{trn} = X_{trn} + A_{trn}$ and $X_{tst} = \hat{X}_{tst} + A_{tst}$. The final goal for the SDAE is to classify the images in the dataset $\hat{X}_{tst}$. However, to pre-train the SDAE, we train autoencoders on data that has even more noise. Let $\tilde{X}_{trn}$ be the noisy version of $\hat{X}_{trn}$. For the Gaussian noise case, we just add more Gaussian noise, for the salt and

pepper noise case, we use a masking noise. We consider two training objectives for the autoencoders; $L(X, \tilde{X})$ and $L(\hat{X}, \tilde{X})$. That is, is our target is the noisy $\hat{X}$ or the noiseless version X. For the linear decoder, we use the mean squared loss. For the non-linear autoencoder, we use an affine + sigmoid encoder and decoder and use the cross entropy loss. To train these networks, we re-sample the noise to go from $\hat{X}$ to $\tilde{X}$ every iteration and train the models using the Adam optimizer for 1000 epochs. We then stack two such layers, add a new layer for classification and fine tune the model on $\hat{X}_{trn}$ for the classification task for an 1000 epochs. We then test on $\hat{X}_{tst}$. Here we used the basic MNIST dataset as our data. So we have 12000 training samples and 50,000 test samples. Figure D.6 shows the results for the 4 experiments.

We see that in some cases, the linear version does better and in some cases the non-linear one is better. Thus, showing that further work needs to be done to understand the non-linear case.



(i) Autoencoder target X (ii) Autoencoder target $\hat{X}$ (iii) Autoencoder target X (iv) Autoencoder target $\hat{X}$

Figure D.6: Plots comparing the classification accuracy for SDAEs pre-trained with linear versus non-linear autoencoders. The results are averaged over 5 trials Figures (a) - (b) are with Gaussian noise, where as Figures (c) - (d) are with salt and pepper noise.

APPENDIX E

How Can Classical Multidimensional Scaling go Wrong?

E.1 Proofs

Lemma 7.10.

$$2\|f(D) - f(D_{\text{cmds}})\|_F^2 = \frac{n\|(S \circ S)\vec{\lambda}\|_F^2 - C_2^2}{2} =: C_3.$$

Proof. First, we note that if Λ is the diagonal matrix whose diagonal is given by the first $n-1$ entries of $\vec{\lambda}$ then we have that

$$U\Lambda U^T = \hat{D} - \hat{D}_{\text{cmds}}.$$

This is true due to Lemmas 7.6 and 7.8. Then we can see the following spectral

decomposition as well

$$Q \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U^T & 0 \\ 0 & 1 \end{bmatrix} Q = Q \begin{bmatrix} \hat{D} - \hat{D}_{\text{cmds}} & 0 \\ 0 & 0 \end{bmatrix} Q.$$

From *Hayden and Wells* (1988), we have that if F is a symmetric matrix, then there are unique f, ξ such that

$$Q \begin{bmatrix} \hat{F} & f \\ f^T & \xi \end{bmatrix} Q$$

is hollow. The unique f, ξ are given by

$$\begin{bmatrix} 2f \\ \xi \end{bmatrix} = \sqrt{n} Q \text{diag} \left(Q \begin{bmatrix} \hat{F} & 0 \\ 0 & 0 \end{bmatrix} Q \right).$$

Now, note that D and D_{cmds} are already hollow, thus, we have that

$$\begin{bmatrix} 2f(D) \\ \xi(D) \end{bmatrix} = \sqrt{n} Q \text{diag} \left(Q \begin{bmatrix} \hat{D} & 0 \\ 0 & 0 \end{bmatrix} Q \right).$$

$$\begin{bmatrix} 2f(D_{\text{cmds}}) \\ \xi(D_{\text{cmds}}) \end{bmatrix} = \sqrt{n} Q \text{diag} \left(Q \begin{bmatrix} \hat{D}_{\text{cmds}} & 0 \\ 0 & 0 \end{bmatrix} Q \right).$$

Taking the difference, we get that

$$\begin{bmatrix} 2f(D) - 2f(D_{\text{cmds}}) \\ \xi(D) - \xi(D_{\text{cmds}}) \end{bmatrix} = n Q \text{diag} \left(Q \begin{bmatrix} \hat{D} - \hat{D}_{\text{cmds}} & 0 \\ 0 & 0 \end{bmatrix} Q \right)$$

Then using the theorem on diagonalization and the Hadamard product from *Million*

(2007), we know that

$$\operatorname{diag}\left(Q\begin{bmatrix}\hat{D}-\hat{D}_{\text{cmds}} & 0\\ 0 & 0\end{bmatrix}Q\right)=(S\circ S)\operatorname{diag}(\vec{\lambda}).$$

Thus, using Lemma 7.9 taking the norm and noting that Q is unitary, we get that

$$\left\|\begin{bmatrix}2f(D)-2f(D_{\text{cmds}})\\ \xi(D)-\xi(D_{\text{cmds}})\end{bmatrix}\right\|_2^2=n\|(S\circ S)\operatorname{diag}(\vec{\lambda})\|_F^2.$$

Thus, we have that

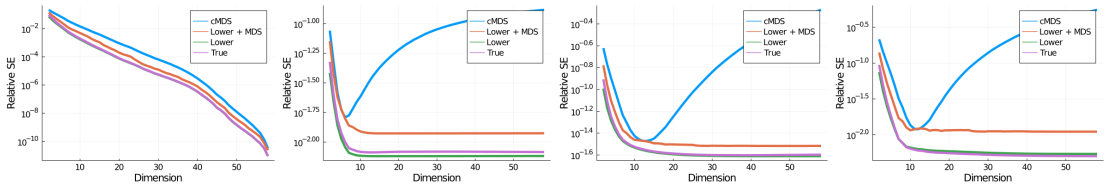
$$2\|f(D)-f(D_{\text{cmds}})\|^2=\frac{n\|(S\circ S)\operatorname{diag}(\vec{\lambda})\|_F^2-C_2^2}{2}.$$

□

E.2 Extra Datasets

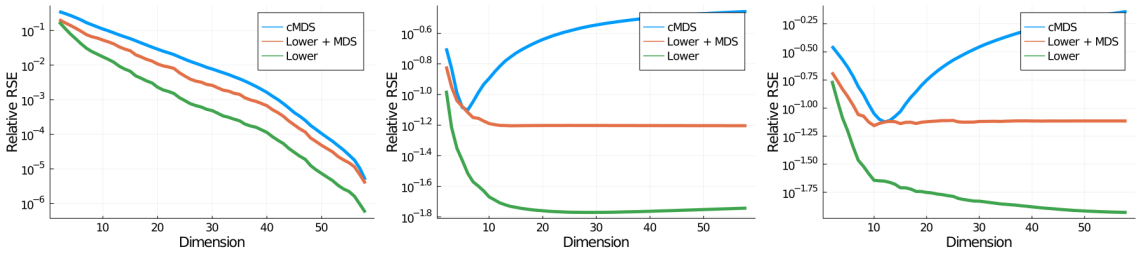
We perform the same experiments as in Section 9.6 on another data set called Sonar *Gorman and Sejnowski* (1988). The results are shown in Figures E.1, E.2, E.3, and E.4. In addition, we compute the *additive* distortion for this data set. As we can see, all of the trends for this data set are consistent with the results in Section 9.6. We note that the overall behavior of the additive distortion somewhat matches that of the distance to the true solution. In other words, the data set in the main paper is not a special one. We also performed the same experiments for the Arrhythmia *Guvenir et al.* (1997), Glass *Evett and Spiehler* (1989), Musk *Dietterich et al.* (1993), Orl, Libras *Dias et al.* (2009), and Ionosphere *Sigillito et al.* (1989). All data sets exhibited the same behavior.

All data and code can be found at:



(i) Sonar (ii) Isomap (iii) Perturbed - Input (iv) Perturbed - Original

Figure E.1: Distance to Input Matrix.



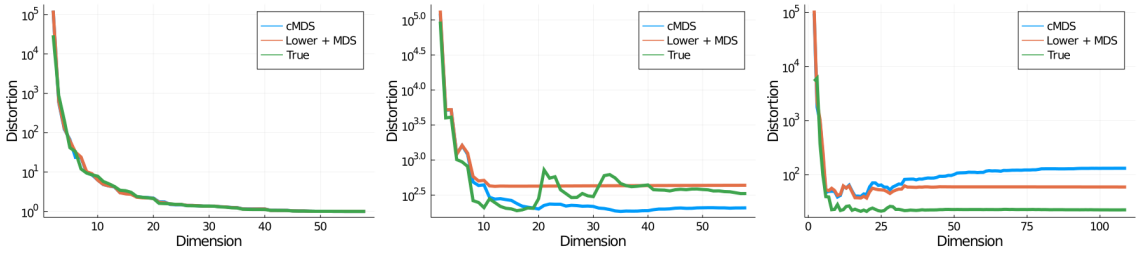
(i) Sonar (ii) Isomap (iii) Perturbed

Figure E.2: Distance to true solution

https://www.dropbox.com/sh/rf3cr4teta79do3/AADtURRC-AhFGivB6-xs6ZEQa?dl=0

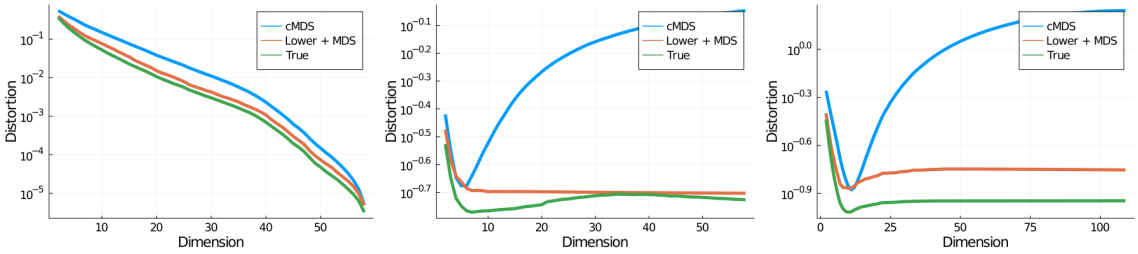
E.3 Computing True MDS solution

The algorithm in *Hayden and Wells* (1988) does not use any dimension constraint and so we use the result from *Qi and Yuan* (2014c) to adapt it. The algorithm in *Hayden and Wells* (1988) is Bregman's cyclic method or Dykstra's method of alternate projections. Instead of projecting $\hat{D}$ onto the cone of negative semi definite cone, we



(i) Sonar (ii) Isomap (iii) Perturbed

Figure E.3: Multiplicative Distortion



(i) Sonar (ii) Isomap (iii) Perturbed

Figure E.4: Average Additive Distortion

project onto $\mathcal{E}(r)$.

We performed each true MDS computation until the change in $\hat{D}$ is at most 0.1, where the difference is measured as the Frobenius norm.

APPENDIX F

Deep Greek: Reconstructing Greek Test

F.1 Appendix

F.1.1 Four Texts

The four texts are the following.

1. Plato, Phaedrus - 231e3-232b5.
2. Aristophanes, Lysistrata - 306-325.
3. Herodotus, 3.40.
4. Homer, Iliad 24.522-542.

F.1.2 Neural Network implementation details

F.1.2.1 The Character Encoding Layers

The character encoding layers are one 1D convolutions layers. In each layer, we pad so that the length of the output sequence does not change, and all of the kernels

Εἰ τοίνυν τὸν νόμον τὸν καθεστηκότα δέδοικας,
μὴπυθομένων τῶν ἀνθρώπων ὄνειδός σοι γένηται,
εἰκός ἐστιτοὺς μὲν ἐρῶντας, οὕτως ἂν οἰομένους
καὶ ὑπὸ τῶν ἄλλωνζηλοῦσθαι ὥσπερ αὐτοὺς ὑφ'
αὑτῶν, ἐπαρθῆναι τῷ λέγεινκαὶ φιλοτιμουμένους
ἐπιδείκνυσθαι πρὸς ἅπαντας ὅτι οὐκἄλλως αὐτοῖς
πεπόνηται· τοὺς δὲ μὴ ἐρῶντας, κρείττουςαὐτῶν
ὄντας, τὸ βέλτιστον ἀντὶ τῆς δόξης τῆς
παρὰ τῶνἀνθρώπων αἱρεῖσθαι. ἔτι δὲ τοὺς μὲν ἐρῶντας
πολλοὺςἀνάγκη πυθέσθαι καὶ ἰδεῖν ἀκολουθοῦντας
τοῖς ἐρωμένοιςκαὶ ἔργον τοῦτο ποιουμένους,
ὥστε ὅταν ὀφθῶσι διαλεγόμενοι ἀλλήλοις,
τότε αὐτοὺς οἴονται ἢ γεγενημένης ἢ μελλούσης
ἔσεσθαι τῆς ἐπιθυμίας συνεῖναι· τοὺς δὲ μὴ
ἐρῶνταςοὐδ' αἰτιᾶσθαι διὰ τὴν συνουσίαν ἐπιχειροῦσιν,
εἰδότες ὅτιἀναγκαῖόν ἐστιν ἢ διὰ φιλίαν
τῳ διαλέγεσθαι ἢ δι' ἄλλην τινὰ ἡδονήν.

Figure F.1: Text 1 mentioned in Section 9.6.5

Τουτὶ τὸ πῦρ ἐγρήγορεν θεῶν ἕκατι καὶ ζῇ. Οὔκουν
ἄν, εἰ τὼ μὲν ξύλω θείμεσθα πρῶτον αὐτοῦ,
τῆς ἀμπέλου δ' εἰς τὴν χύτραν τὸν φανὸν ἐγκαθέντες
ἅψαντες εἶτ' εἰς τὴν θύραν κριηδὸν ἐμπέσοιμεν,
κἂν μὴ καλούντων τοὺς μοχλοὺς χαλῶσιν
αἱ γυναῖκες, ἐμπιμπράναι χρὴ τὰς θύρας καὶ τῷ
καπνῷ πιέζειν. θώμεσθα δὴ τὸ φορτίον. Φεῦ τοῦ
καπνοῦ, βαβαιάξ. Τίς ξυλλάβοιτ' ἂν τοῦ ξύλου
τῶν ἐν Σάμῳ στρατηγῶν; Ταυτὶ μὲν ἤδη τὴν ῥάχιν
θλίβοντά μου πέπαυται. Σὸν δ' ἔργον ἐστίν,
ὦ χύρτα, τὸν ἄνθρακ' ἐξεγείρειν, τὴν λαμπάδ'
ἡμμένην ὅπως παρὼν ἐμοὶ προσοίσει. Δέσποινα
Νίκη, ξυγγενοῦ τῶν τ' ἐν πόλει γυναικῶν τοῦ
νῦν παρεστῶτος θράσους θέσθαι τροπαῖον ἡμᾶς.
Λιγνὺν δοκῶ μοι καθορᾶν καὶ καπνόν, ὦ γυναῖκες,
ὥσπερ πυρὸς καομένου· σπευστέον ἐστὶ θᾶττον.
Πέτου πέτου, Νικοδίκη, πρὶν ἐμπεπρῆσθαι Καλύκην
τε καὶ Κρίτυλλαν περιφυσήτω ὑπό τ' ἀνέμων
ἀργαλέων ὑπό τε γερόντων ὀλέθρων.

Figure F.2: Text 2 mentioned in Section 9.6.5

Καί κως τὸν Ἄμασιν εὐτυχέων μεγάλως ὁ Πολυκράτηςοὐκ
ἐλάνθανε, ἀλλά οἱ τοῦτ' ἦν ἐπιμελές. Πολλῷ
δὲ ἔτιπλέονός οἱ εὐτυχίης γινομένης γράψας
ἐς βυβλίον τάδεἐπέστειλε ἐς Σάμον· «Ἄμασις
Πολυκράτεϊ ὧδε λέγει.Ἡδὺ μὲν πυνθάνεσθαι ἄνδρα
φίλον καὶ ξεῖνον εὖ πρήσσοντα,ἐμοὶ δὲ αἱ σαὶ
μεγάλαι εὐτυχίαι οὐκ ἀρέσκουσι, τὸ θεῖον ἐπισταμένῳ
ὡς ἔστι φθονερόν. Καί κως βούλομαι καὶ
αὐτὸςκαὶ τῶν ἂν κήδωμαι τὸ μέν τι εὐτυχέειν
τῶν πρηγμάτων,τὸ δὲ προσπταίειν, καὶ οὕτω διαφέρειν
τὸν αἰῶνα ἐναλλὰξπρήσσων ἢ εὐτυχέειν
τὰ πάντα· οὐδένα γάρ κω λόγῳ οἶδαἀκούσας ὅστις
ἐς τέλος οὐ κακῶς ἐτελεύτησε πρόρριζος,εὐτυχέων
τὰ πάντα. Σύ νυν ἐμοὶ πειθόμενος ποίησον
πρὸςτὰς εὐτυχίας τοιάδε. Φροντίσας τὸ ἂν εὕρῃς
ἐόν τοιπλείστου ἄξιον καὶ ἐπ' ᾧ σὺ ἀπολομένῳ
μάλιστα τὴν ψυχὴνἀλγήσεις, τοῦτο ἀπόβαλε
οὕτω ὅκως μηκέτι ἥξει ἐς ἀνθρώ-πους. Ἢν τε μὴ
ἐναλλὰξ ἤδη τὠπὸ τούτου αἱ εὐτυχίαι τοιτῇσι
πάθησι προσπίπτωσι, τρόπῳ τῷ ἐξ ἐμέο ὑποκειμένῳἀκέο.»

Figure F.3: Text 3 mentioned in Section 9.6.5

ἀλλ' ἄγε δὴ κατ' ἄρ' ἔζευ ἐπὶ θρόνου, ἄλγεα δ'
ἔμπηςἐν θυμῷ̃ κατακεῖσθαι ἐάσομεν ἀχνύμενοί περ·οὐ
γάρ τις πρῆξις πέλεται κρυεροῖο γόοιο·ὡς
γὰρ ἐπεκλώσαντο θεοὶ δειλοῖσι βροτοῖσιζώειν
ἀχνυμένοις· αὐτοὶ δέ τ' ἀκηδέες εἰσί.δοιοὶ γάρ
τε πίθοι κατακείαται ἐν Διὸς οὔδειδώρων οἷα
δίδωσι κακῶν, ἕτερος δὲ ἑάων·ᾧ μέν κ' ἀμμίξας
δώῃ Ζεὺς τερπικέραυνος,ἄλλοτε μέν τε κακῷ̃
ὅ γε κύρεται, ἄλλοτε δ' ἐσθλῷ̃·ᾧ̃ δέ κε τῶν λυγρῶν
δώῃ, λωβητὸν ἔθηκε, καί ἑ κακὴ βούβρωστις
ἐπὶ χθόνα δῖαν ἐλαύνει,φοιτᾷ̃ δ' οὔτε θεοῖσι
τετιμένος οὔτε βροτοῖσιν.ὣς μὲν καὶ Πηλῆϊ θεοὶ
δόσαν ἀγλαὰ δῶραἐκ γενετῆς· πάντας γὰρ ἐπ'
ἀνθρώπους ἐκέκαστοὅλβῳ̣ τε πλούτῳ̣ τε, ἄνασσε δὲ
Μυρμιδόνεσσι,καί οἱ θνητῷ̃ ἐόντι θεὰν ποίησαν
ἄκοιτιν.ἀλλ' ἐπὶ καὶ τῷ̃ θῆκε θεὸς κακόν, ὅττί
οἱ οὔ τιπαίδων ἐν μεγάροισι γονὴ γένετο κρειόντων,ἀλλ'
ἕνα παῖδα τέκεν παναώριον· οὐδέ
νυ τόν γεγηράσκοντα κομίζω, ἐπεὶ μάλα τηλόθι
πάτρηςἧμαι ἐνὶ Τροίῃ, σέ τε κήδων ἠδὲ σὰ τέκνα.

Figure F.4: Text 4 mentioned in Section 9.6.5

have length 3. The layer has 256 filters. The character embedding layers are not shared between the three recurrent parts of the network. Each recurrent part has its own character embedding network.

F.1.2.2 The Encoder LSTM Network

The encoder LSTM network has 3 LSTM layers. Each LSTM layer has 256 units. In this case, we unroll our network for n steps. Each layer returns the whole sequence. The last layer also returns the hidden states.

F.1.2.3 The Decoder LSTM Network

The encoder LSTM network has 4 LSTM layers. Each LSTM layer has 256 units. In this case, we unroll our network for n steps. Each layer returns the whole sequence. The first layer hidden states is initialized using the last layer from the encoder network.

F.1.2.4 Training Details

We trained each network on ∼11,00 batches, each having have 1024 samples. We used one GPU, and each epoch on a single GPU took 35 minutes for $n = 10$.

LEARN2FILL was trained for 50 epochs. LEARN2ACCENT was trained for 13 epochs and LEARN2SPACE was trained for 1 epochs.

Our training data is a corpus of 70 texts, and we used an 80-10-10 train-validation-test split when training the neural networks. The list of tests used are as follows:

1. Aelius Aristides - Ars Rhetorica
2. Aeschines - Speeches
3. Aeschylus - Agamemnon
4. Aeschylus - Eumenides
5. Aeschylus - Libation Bearers
6. Aeschylus - Persians

7. Aeschylus - Prometheus Bound
8. Aeschylus - Seven Against Thebes
9. Aeschylus - Suppliant Women
10. Andocides - Speeches
11. Appian - The Civil War
12. Appian - The Foreign Wars
13. Aristophanes - Clouds
14. Aristophanes - Peace
15. Aristotle - Athenian Constitution
16. Aristotle - Economics
17. Aristotle - Eudemian Ethics
18. Aristotle - Metaphysics
19. Aristotle - Nicomachean Ethics
20. Aristotle - Politics
21. Aristotle - Rhetoric
22. Aristotle - Virtues and Vices
23. Arrian - Anabasis
24. Arrian - Cynegeticus
25. Arrian - Indica
26. Arrian - Periplus Ponti Euxini
27. Arrian - Tactica
28. Athenaeus - The Deipnosophists: Book 1-15
29. Chrysostom Dio - Orationes
30. Demades = On the Twelve Years
31. Demosthenes - Speeches 1 - 61
32. Diodorus Siculus - Bibliotheca Historica Books I - V, XVIII - XX
33. Dionysius of Halicarnassus - Ad Ammaeum

34. Dionysius of Halicarnassus - Antiquitates Romanae Books I - XX
35. Dionysius of Halicarnassus - De Antiquis Oratoribus
36. Dionysius of Halicarnassus - De Compositione Verborum
37. Dionysius of Halicarnassus - De Demosthene
38. Dionysius of Halicarnassus - De Dinarcho
39. Dionysius of Halicarnassus - De Isaeo
40. Dionysius of Halicarnassus - De Isocrate
41. Dionysius of Halicarnassus - De Lysia
42. Dionysius of Halicarnassus - De Thucydide
43. Dionysius of Halicarnassus - De Thucydidis Idiomatibus (Epistula ad Ammaeum)
44. Dionysius of Halicarnassus - Epistula ad Pompeium Geminum
45. Dionysius of Halicarnassus - Libri Secundi de Antiquis Oratoribus Reliquia
46. Euripides - Bacchae
47. Euripides - Electra
48. Euripides - Hecuba
49. Euripides - Helen
50. Euripides - Heracles
51. Euripides - Ion
52. Euripides - Iphigenia in Aulis
53. Euripides - Iphigenia in Tauris
54. Euripides - Orestes
55. Euripides - Phoenissae
56. Euripides - Rhesus
57. Euripides - Supplicants
58. Euripides - The Trojan Women
59. Hyperides - Speeches
60. Lysias - Speeches

61. Plato - Republic
62. Polybius - Histories
63. Sophocles - Antigone
64. Sophocles - Ajax
65. Sophocles - Electra
66. Sophocles - Oedipus at Colonus
67. Sophocles - Oedipus Tyrannus
68. Sophocles - Philoctetes
69. Sophocles - Trachiniae
70. Xenophon - Anabasis

F.1.3 Phrase Appended for Filling in Diacritics

This is the phrase that we append when solving stage 2. We found that switching up the phrase did not have much of an impact on the performance.

ἐπειδὴπσανπόλινὁρῶμεν

BIBLIOGRAPHY

BIBLIOGRAPHY

Abraham, I., Y. Bartal, T. Chan, K. Dhamdhere, A. Gupta, J. Kleinberg, O. Neiman, and A. Slivkins (2005), Metric embeddings with relaxed guarantees, in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 83–100.

Abraham, I., M. Balakrishnan, F. Kuhn, D. Malkhi, V. Ramasubramanian, and K. Talwar (2007), Reconstructing Approximate Tree Metrics, in *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pp. 43–52, ACM, New York, NY, USA.

Abraham, I., C. Gavoille, A. Gupta, O. Neiman, and K. Talwar (2014), Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs, in *Symposium on Theory of Computing (STOC)*, pp. 79–88.

Abraham, I., S. Chechik, and S. Krinninger (2017), Fully dynamic all-pairs shortest paths with worst-case update-time revisited, in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 440–452.

Agarwal, A., N. Alon, and M. Charikar (2007), Improved approximation for directed cut problems, in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 671–680.

Ailon, N., M. Charikar, and A. Newman (2005), Aggregating Inconsistent Information: Ranking and Clustering, in *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pp. 684–693, ACM, New York, doi:10.1145/1060590.1060692.

Akhtar, N., and A. Mian (2018), Threat of adversarial attacks on deep learning in computer vision: A survey, *IEEE Access*, *6*, 14,410–14,430.

Al-Homidan, S., and H. Wolkowicz (2005), Approximate and exact completion problems for euclidean distance matrices using semidefinite programming, *Linear Algebra and its Applications*, *406*, 109 – 141, doi:https://doi.org/10.1016/j.laa.2005.03.021.

Alaya, M. Z., M. Berar, G. Gasso, and A. Rakotomamonjy (2019), Screening sinkhorn algorithm for regularized optimal transport, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, pp. 12,169–12,179, Curran Associates, Inc.

Alon, N., and S. Gutner (2010), Balanced families of perfect hash functions and their applications, *ACM Trans. Algorithms*, *6*(3), 54:1–54:12.

Alon, N., R. M. Karp, D. Peleg, and D. West (1995), A graph-theoretic game and its application to the k-server problem, *SIAM J. Comput.*, *24*(1), 78–100, doi:10.1137/S0097539792224474.

Alvarez-Melis, D., T. Jaakkola, and S. Jegelka (2018), Structured optimal transport, in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, vol. 84, pp. 1771–1780, PMLR, Playa Blanca, Lanzarote, Canary Islands.

Amari, S. (2016), *Information Geometry and Its Applications*, Applied Mathematical Sciences, Springer Japan.

Amato, G., F. Falchi, and L. Vadicamo (2016), Visual recognition of ancient inscriptions using convolutional neural network and fisher vector, *J. Comput. Cult. Herit.*, *9*(4), 21:1–21:24, doi:10.1145/2964911.

Arabie, P., M. S. Aldenderfer, D. Carroll, and W. S. DeSarbo (1987), *Three Way Scaling: A Guide to Multidimensional Scaling and Clustering*, vol. 65, Sage.

Assael, Y., T. Sommerschield, and J. Prag (2019a), Restoring ancient text using deep learning: a case study on greek epigraphy.

Assael, Y., T. Sommerschield, and J. Prag (2019b), Restoring ancient text using deep learning: a case study on greek epigraphy, *EMNLP 2019*.

Avadesh, M., and N. Goyal (2018), Optical character recognition for sanskrit using convolution neural networks, *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 447–452.

Avis, D., P. Hayden, and M. M. Wilde (2010), Leggett-Garg Inequalities and the Geometry of the Cut Polytope, *Phys. Rev. A (3)*, *82*(3), 030,102, 4, doi:10.1103/PhysRevA.82.030102.

Backurs, A., Y. Dong, P. Indyk, I. Razenshteyn, and T. Wagner (2019), Scalable nearest neighbor search for optimal transport.

Bahdanau, D., K. Cho, and Y. Bengio (2014), Neural machine translation by jointly learning to align and translate, *CoRR*, *abs/1409.0473*.

Bai, S., X. Bai, L. J. Latecki, and Q. Tian (2017), Multidimensional scaling on multiple input distance matrices, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31.

Bai, Z., B. Miao, and J. Yao (2003), Convergence rates of spectral distributions of large sample covariance matrices, *SIAM J. Matrix Anal. Appl.*, *25*, 105–127.

Baier, G., T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella (2010), Length-bounded cuts and flows, *ACM Trans. Algorithms*, *7*(1), 4:1–4:27.

Bakonyi, M., and C. Johnson (1995), The euclidian distance matrix completion problem, *SIAM Journal on Matrix Analysis and Applications*, *16*(2), 646–654, doi:10.1137/S0895479893249757.

Balas, E., and W. Pulleyblank (1983), The Perfectly Matchable Subgraph Polytope of a Bipartite Graph, in *Proceedings of the symposium on the matching problem: theory, algorithms, and applications (Gaithersburg, Md., 1981)*, vol. 13, pp. 495–516, doi:10.1002/net.3230130405.

Balzano, L., and W. U. Z. Bajwa (2010), Column subset selection with missing data, *NIPS Workshop on Low-Rank Methods for Large-Scale Machine Learning*.

Banham, M., and A. Katsaggelos (2012), Digital image restoration.

Banjac, G., P. Goulart, B. Stellato, and S. Boyd (2019), Infeasibility detection in the alternating direction method of multipliers for convex optimization, *Journal of Optimization Theory and Applications*, *183*(2), 490–519, doi:10.1007/s10957-019-01575-y.

Bansal, N., A. Blum, and S. Chawla (2004), Correlation Clustering, *Mach. Learn.*, *56*(1-3), 89–113, doi:10.1023/B:MACH.0000033116.57574.95.

Barahona, F. (1993), On Cuts and Matchings in Planar Graphs, *Math. Programming*, *60*(1, Ser. A), 53–68, doi:10.1007/BF01580600.

Barahona, F., and A. R. Mahjoub (1994), Compositions of Graphs and Polyhedra. I. Balanced Induced Subgraphs and Acyclic Subgraphs, *SIAM J. Discrete Math.*, *7*(3), 344–358, doi:10.1137/S0895480190182666.

Baraty, S., D. A. Simovici, and C. Zara (2011), The impact of triangular inequality violations on medoid-based clustering, in *Foundations of Intelligent Systems*, edited by M. Kryszkiewicz, H. Rybinski, A. Skowron, and Z. W. Raś, pp. 280–289, Springer Berlin Heidelberg, Berlin, Heidelberg.

Bartal, Y. (1998), On Approximating Arbitrary Metrics by Tree Metrics, in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 161–168, ACM, New York, NY, USA.

Bauschke, H., and J. J. Borwein (1998), Legendre Functions and the Method of Random Bregman Projections, *Journal of Convex Analysis*, *4*.

Bauschke, H. H., and J. M. Borwein (1997), Legendre Functions and the Method of Random Bregman Projections, *J. Convex Anal.*, *4*(1), 27–67.

Bauschke, H. H., and A. S. Lewis (2000a), Dykstras Algorithm with Bregman Projections: A Convergence Proof, *Optimization*, *48*(4), 409–427, doi:10.1080/02331930008844513.

Bauschke, H. H., and A. S. Lewis (2000b), Dykstra's Algorithm with Bregman Projections: a Convergence Proof, *Optimization*, *48*(4), 409–427, doi:10.1080/02331930008844513.

Beck, A., and M. Teboulle (2009), A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems, *SIAM J. Imaging Sci.*, *2*(1), 183–202, doi:10.1137/080716542.

Belkin, M., and P. Niyogi (2003), Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Comput.*, *15*(6), 1373–1396, doi:10.1162/089976603321780317.

Bellet, A., A. Habrard, and M. Sebban (2013a), A survey on metric learning for feature vectors and structured data, *CoRR*, *abs/1306.6709*.

Bellet, A., A. Habrard, and M. Sebban (2013b), A Survey on Metric Learning for Feature Vectors and Structured Data, *ArXiv*, *abs/1306.6709*.

Ben-Tal, A., L. Ghaoui, and A. Nemirovski (2009), *Robust Optimization*, Princeton Series in Applied Mathematics, Princeton University Press.

Benamou, J.-D., and Y. Brenier (2000), A computational fluid mechanics solution to the monge-kantorovich mass transfer problem, *Numerische Mathematik*, *84*, 375–393.

Benamou, J.-D., G. Carlier, M. Cuturi, L. Nenna, and G. Peyré (2015), Iterative bregman projections for regularized transportation problems, *SIAM Journal on Scientific Computing*, *37*(2), A1111–A1138, doi:10.1137/141000439.

Benesty, J., J. Chen, and Y. Huang (2010), Study of the widely linear wiener filter for noise reduction, *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 205–208.

Bengio, Y., J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet (2003), Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering, in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pp. 177–184, MIT Press, Cambridge, MA, USA.

Berglund, M., T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. T. Karhunen (2015), Bidirectional recurrent neural networks as generative models, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pp. 856–864, Curran Associates, Inc.

Bermudo, S., J. M. Rodríguez, J. M. Sigarreta, and J.-M. Vilaire (2013), Gromov Hyperbolic Graphs, *Discrete Mathematics*, *313*(15), 1575 – 1585.

Bertsekas, D. P., and D. Castanon (1989), The auction algorithm for the transportation problem, *Annals of Operations Research*, *20*, 67–96.

Bertsimas, D., D. Brown, and C. Caramanis (2011), Theory and applications of robust optimization, *SIAM Rev.*, *53*, 464–501.

Bishop, C. M. (1995), Training with noise is equivalent to tikhonov regularization, *Neural Comput.*, *7*(1), 108–116, doi:10.1162/neco.1995.7.1.108.

Blasius, T., T. Friedrich, A. Krohmer, and S. Laue (2018), Efficient Embedding of Scale-Free Graphs in the Hyperbolic Plane, *IEEE/ACM Transactions on Networking*, *26*(2), 920–933.

Blondel, M., V. Seguy, and A. Rolet (2018a), Smooth and sparse optimal transport, in *AISTATS*.

Blondel, M., V. Seguy, and A. Rolet (2018b), Smooth and Sparse Optimal Transport, in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, vol. 84, pp. 880–889, PMLR.

Bodwin, G. (2018), On the structure of unique shortest paths in graphs, *CoRR*, *abs/1804.09745*.

Boggs, P., and J. Tolle (1995), Sequential Quadratic Programming, *Acta Numerica*, *4*, 1–51, doi:10.1017/S0962492900002518.

Bonk, M., and O. Schramm (2000), Embeddings of Gromov Hyperbolic Spaces, *Geometric & Functional Analysis GAFA*, *10*(2), 266–306.

Bonneel, N., M. van de Panne, S. Paris, and W. Heidrich (2011), Displacement interpolation using lagrangian mass transport, *ACM Trans. Graph.*, *30*(6), 1–12, doi:10.1145/2070781.2024192.

Borg, I., and P. J. Groenen (2005), *Modern multidimensional scaling: Theory and applications*, Springer Science & Business Media.

Bourgain, J. (1985), On lipschitz embedding of finite metric spaces in hilbert space, *Israel Journal of Mathematics*, *52*(1-2), 46–52.

Brand, C., H. Dell, and T. Husfeldt (2018), Extensor-coding, in *Symposium on Theory of Computing (STOC)*, pp. 151–164.

Brickell, J., I. Dhillon, S. Sra, and J. Tropp (2008a), The metric nearness problem, *SIAM Journal on Matrix Analysis and Applications*, *30*(1), 375–396.

Brickell, J., I. Dhillon, S. Sra, and J. Tropp (2008b), The metric nearness problem, *SIAM J. Matrix Analysis Applications*, *30*(1), 375–396, doi:10.1137/060653391.

Brickell, J., I. S. Dhillon, S. Sra, and J. A. Tropp (2008c), The Metric Nearness Problem, *SIAM J. Matrix Anal. Appl.*, *30*(1), 375–396, doi:10.1137/060653391.

Bridson, M., and A. Häfliger (2013), *Metric Spaces of Non-Positive Curvature*, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg.

Brualdi, R. A. (2006), *Combinatorial Matrix Classes*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, doi:10.1017/CBO9780511721182.

Buades, A., B. Coll, and J. Morel (2005), A non-local algorithm for image denoising, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *2*, 60–65 vol. 2.

Buss, J., and J. Goldsmith (1993), Nondeterminism within p^*, *SIAM Journal on Computing*, *22*(3), 560–572, doi:10.1137/0222038.

Cai, H., V. W. Zheng, and K. Chang (2018), A comprehensive survey of graph embedding: Problems, techniques, and applications, *IEEE Transactions on Knowledge and Data Engineering*, *30*(09), 1616–1637, doi:10.1109/TKDE.2018.2807452.

Campbell, L. (2013), *Historical Linguistics: An Introduction*, Edinburgh University Press.

Candès, E., and B. Recht (2012), Exact matrix completion via convex optimization, *Commun. ACM*, *55*(6), 111–119.

Candès, E., and T. Tao (2010), The power of convex relaxation: Near-optimal matrix completion, *IEEE Transactions on Information Theory*, *56*, 2053–2080.

Carnahan, S. (2010), It is Well Known that Hyperbolic Space is δ-Hyperbolic, but what is Delta?, MathOverflow.

Carroll, J. D., and P. Arabie (1998), Multidimensional scaling, *Measurement, judgment and decision making*, pp. 179–250.

Carroll, J. D., and J.-J. Chang (1970), Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition, *Psychometrika*, *35*(3), 283–319.

Cayton, L., and S. Dasgupta (2006a), Robust euclidean embedding, in *Proceedings of the 23rd international conference on machine learning*, pp. 169–176.

Cayton, L., and S. Dasgupta (2006b), Robust euclidean embedding, in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, p. 169–176, Association for Computing Machinery, New York, NY, USA, doi:10.1145/1143844.1143866.

Censor, Y., and S. Reich (1998), The Dykstra Algorithm with Bregman Projections, *Communications in Applied Analysis*, *2*, 407–419.

Censor, Y., and S. Zenios (1997), *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press.

Cerliani, M. (2020), Group2vec for advance categorical encoding.

Chakraborty, A., M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay (2018), Adversarial attacks and defences: A survey, *ArXiv*, *abs/1810.00069*.

Chan, T., K. Dhamdhere, A. Gupta, J. Kleinberg, and A. Slivkins (2009), Metric embeddings with relaxed guarantees, *SIAM J. Comput.*, *38*(6), 2303–2329.

Chandran, L. S., V. V. Lozin, and C. R. Subramanian (2005), Graphs of low chordality, *Discrete Mathematics and Theoretical Computer Science*, *7*, 25–36.

Chandrasekaran, K., L. A. Végh, and S. Vempala (2012), The Cutting Plane method is Polynomial for Perfect Matchings, in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science—FOCS 2012*, pp. 571–580, IEEE Computer Soc., Los Alamitos, CA.

Charikar, M., V. Guruswami, and A. Wirth (2005), Clustering with Qualitative Information, *J. Comput. System Sci.*, *71*(3), 360–383, doi:10.1016/j.jcss.2004.10.012.

Chawla, S., R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar (2006), On the hardness of approximating multicut and sparsest-cut, *Computational Complexity*, *15*(2), 94–114.

Chekuri, C., and V. Madan (2017), Approximating multicut and the demand graph, in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 855–874.

Chen, Y., and T. Pock (2017), Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 1256–1272.

Chepoi, V., and F. Dragan (2000), A Note on Distance Approximating Trees in Graphs, *European Journal of Combinatorics*, *21*(6), 761 – 766.

Chepoi, V., F. Dragan, B. Estellon, M. Habib, and Y. Vaxès (2008), Diameters, Centers, and Approximating Trees of δ-Hyperbolic Geodesic Spaces and Graphs, in *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, pp. 59–68, ACM, New York, NY, USA.

Chizat, L., G. Peyré, B. Schmitzer, and F.-X. Vialard (2015), Unbalanced optimal transport: Dynamic and kantorovich formulation, *arXiv: Optimization and Control*.

Chizat, L., G. Peyré, B. Schmitzer, and F.-X. Vialard (2016a), Scaling Algorithms for Unbalanced Transport Problems, *Mathematics of Computation*, *87*, doi:10.1090/mcom/3303.

Chizat, L., G. Peyré, B. Schmitzer, and F.-X. Vialard (2016b), Scaling Algorithms for Unbalanced Transport Problems, *Mathematics of Computation*, *87*, doi:10.1090/mcom/3303.

Cho, K. (2013), Boltzmann machines and denoising autoencoders for image denoising, *CoRR*, *abs/1301.3468*.

Choi, J. H., O. A. Elgendy, and S. H. Chan (2019), Optimal combination of image denoisers, *IEEE Transactions on Image Processing*, *28*(8), 4016–4031, doi:10.1109/TIP.2019.2903321.

Christofides, N. (1976a), Worst-case analysis of a new heuristic for the travelling salesman problem, *Tech. Rep. 388*, Graduate School of Industrial Administration, Carnegie Mellon University.

Christofides, N. (1976b), Worst-case analysis of a new heuristic for the traveling salesman problem.

Chung, F., M. Garrett, R. Graham, and D. Shallcross (2001), Distance realization problems with applications to internet tomography, *J. Comput. Syst. Sci.*, *63*(3), 432–448.

Chuzhoy, J., and S. Khanna (2009), Polynomial flow-cut gaps and hardness of directed cut problems, *J. ACM*, *56*(2), 6:1–6:28.

Coifman, R. R., and S. Lafon (2006), Diffusion maps, *Applied and Computational Harmonic Analysis*, *21*(1), 5 – 30, doi:https://doi.org/10.1016/j.acha.2006.04.006, special Issue: Diffusion Maps and Wavelets.

Comon, P. (1994), Independent component analysis, a new concept?, *Signal Process.*, *36*, 287–314.

Cox, M. A., and T. F. Cox (2008), Multidimensional scaling, in *Handbook of data visualization*, pp. 315–347, Springer.

Crane, R. R. (1987), Perseus digital library.

Crank, J., and P. Nicolson (1947), A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, *P. Camb. Philos. Soc.*, *43*, 50–64.

Cuturi, M. (2013), Sinkhorn Distances: Lightspeed Computation of Optimal Transport, in *Advances in Neural Information Processing Systems 26*, edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, pp. 2292–2300, NeurIPs.

Davis, C., and W. M. Kahan (1970), The rotation of eigenvectors by a perturbation. iii, *SIAM Journal on Numerical Analysis*, *7*(1), 1–46, doi:10.1137/0707001.

Davis, J. V., B. Kulis, P. Jain, S. Sra, and I. S. Dhillon (2007), Information-Theoretic Metric Learning, in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 209–216, ACM, New York, NY, USA.

Dessein, A., N. Papadakis, and J.-L. Rouas (2018a), Regularized Optimal Transport and The Rot Mover's Distance, *J. Mach. Learn. Res.*, *19*(1), 590–642.

Dessein, A., N. Papadakis, and J.-L. Rouas (2018b), Regularized optimal transport and the rot mover's distance, *Journal of Machine Learning Research*, *abs/1610.06447*.

Detrano, R., A. Jánosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher (1989), International application of a new probability algorithm for the diagnosis of coronary artery disease., *The American journal of cardiology*, *64 5*, 304–10.

Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018), BERT: pre-training of deep bidirectional transformers for language understanding, *CoRR*, *abs/1810.04805*.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019), Bert: Pre-training of deep bidirectional transformers for language understanding, in *NAACL-HLT*.

Dey, S. S., and M. Molinaro (2018), Theoretical Challenges Towards Cutting-Plane Selection, *Math. Program.*, *170*(1, Ser. B), 237–266, doi:10.1007/s10107-018-1302-4.

Deza, M., and M. Laurent (1994), Applications of Cut Polyhedra. I, II, *J. Comput. Appl. Math.*, *55*(2), 191–216, 217–247, doi:10.1016/0377-0427(94)90020-5.

Dhillon, I., S. Sra, and J. Tropp (2005), Triangle fixing algorithms for the metric nearness problem, in *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, edited by L. K. Saul, Y. Weiss, and L. Bottou, pp. 361–368, MIT Press.

Dhillon, I. S., and J. A. Tropp (2007), Matrix Nearness Problems with Bregman Divergences, *SIAM J. Matrix Anal. Appl.*, *29*(4), 1120–1146, doi:10.1137/060649021.

Dhingra, B., C. J. Shallue, M. Norouzi, A. M. Dai, and G. E. Dahl (2018), Embedding Text in Hyperbolic Spaces.

Diaconis, P., and D. Freedman (1987), A dozen de finetti-style results in search of a theory, *Annales De L Institut Henri Poincare-probabilites Et Statistiques*, *23*, 397–423.

Dias, D. B., R. B. Madeo, T. Rocha, H. H. Bíscaro, and S. M. Peres (2009), Hand movement recognition for brazilian sign language: A study using distance-based neural networks, *2009 International Joint Conference on Neural Networks*, pp. 697–704.

Dietterich, T. G., A. Jain, R. Lathrop, and T. Lozano-Perez (1993), A comparison of dynamic reposing and tangent distance for drug activity prediction, in *NIPS*.

Dokmanic, I., R. Parhizkar, J. Ranieri, and M. Vetterli (2015), Euclidean distance matrices: Essential theory, algorithms, and applications, *IEEE Signal Processing Magazine*, *32*(6), 12–30.

Domahidi, A., E. Chu, and S. Boyd (2013), ECOS: An SOCP solver for embedded systems, in *European Control Conference (ECC)*, pp. 3071–3076.

Donahue, J., L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell (2017), Long-term recurrent convolutional networks for visual recognition and description, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 677–691, doi:10.1109/TPAMI.2016.2599174.

Donoho, D. (2006), Compressed sensing, *IEEE Transactions on Information Theory*, *52*, 1289–1306.

Donoho, D. L., and C. Grimes (2003), Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data.

Dua, D., and C. Graff (2017), UCI machine learning repository.

Duff, I., and J. Koster (2001), On algorithms for permuting large entries to the diagonal of a sparse matrix, *SIAM J. Matrix Anal. Appl.*, *22*, 973–996.

Dumitrascu, B., S. Villar, D. G. Mixon, and B. E. Engelhardt (2019), Optimal Marker Gene Selection for Cell Type Discrimination in Single Cell Analyses, *bioRxiv*.

Dunning, I., J. Huchette, and M. Lubin (2017), Jump: A modeling language for mathematical optimization, *SIAM Review*, *59*(2), 295–320, doi:10.1137/15M1020575.

Dyubina, A., and I. Polterovich (2001), Explicit Constructions of Universal $\mathbb{R}$-Trees and Asymptotic Geometry of Hyperbolic Spaces, *Bulletin of the London Mathematical Society*, *33*(6), 727?734.

Easton, M. L. (1989), *Chapter 7: Random orthogonal matrices*, *Regional Conference Series in Probability and Statistics*, vol. Volume 1, pp. 100–107, Institute of Mathematical Statistics and American Statistical Association, Haywood CA and Alexandria VA, doi:10.1214/cbms/1462061037.

Elfving, T. (1989), An Algorithm for Maximum Entropy Image Reconstruction from Noisy Data, *Math. Comput. Modelling*, *12*(6), 729–745, doi:10.1016/0895-7177(89)90358-0.

Elkin, M., Y. Emek, D. A. Spielman, and S.-H. Teng (2005), Lower-stretch spanning trees, in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, p. 494–503, Association for Computing Machinery, New York, NY, USA, doi:10.1145/1060590.1060665.

Emanuel, D., and A. Fiat (2003), Correlation Clustering—Minimizing Disagreements on Arbitrary Weighted Graphs, in *Algorithms—ESA 2003*, *Lecture Notes in Comput. Sci.*, vol. 2832, pp. 208–220, Springer, Berlin, doi:10.1007/978-3-540-39658-1_21.

Essid, M., and J. Solomon (2017), Quadratically-Regularized Optimal Transport on Graphs, *SIAM J. Scientific Computing*, *40*, A1961–A1986.

Evett, I., and E. Spiehler (1989), Rule induction in forensic science, *Knowledge Based Systems*, pp. 152–160.

Fan, C., A. C. Gilbert, B. Raichel, R. Sonthalia, and G. V. Buskirk (2018a), Generalized metric repair on graphs, *ArXiv*, *abs/1807.07619*.

Fan, C., B. Raichel, and G. V. Buskirk (2018b), Metric violation distance: Hardness and approximation, in *SODA*.

Fan, C., B. Raichel, and G. Van Buskirk (2018c), Metric Violation Distance: Hardness and Approximation, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 196–209, doi:10.1137/1.9781611975031.14.

Fan, C., B. Raichel, and G. Van Buskirk (2018d), Metric violation distance: Hardness and approximation, in *Symposium on Discrete Algorithms (SODA)*, pp. 196–209.

Fan, C., A. Gilbert, B. Raichel, R. Sonthalia, and G. V. Buskirk (2020), Generalized metric repair on graphs, in *SWAT*.

Fan, R.-E., P.-H. Chen, and C.-J. Lin (2005), Working Set Selection Using Second Order Information for Training Support Vector Machines, *J. Mach. Learn. Res.*, *6*, 1889–1918.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin (2008), LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research*, *9*, 1871–1874.

Fedus, W., I. Goodfellow, and A. M. Dai (2018), MaskGAN: Better text generation via filling in the _____, in *International Conference on Learning Representations*.

Fefferman, C., S. Ivanov, M. Lassas, and H. Narayanan (2019), Reconstruction of a riemannian manifold from noisy intrinsic distances.

Feige, U. (1998), A threshold of ln n for approximating set cover, *J. ACM*, *45*(4), 634–652, doi:10.1145/285055.285059.

Feldman, U., E. Landi, and N. A. Schwadron (2005), On the sources of fast and slow solar wind, *J. Geophys. Res.-Space*, *110*(A9), 7109–+, doi:10.1029/2004JA010918.

Ferradans, S., N. Papadakis, G. Peyré, and J.-F. Aujol (2014), Regularized discrete optimal transport, *SIAM Journal on Imaging Sciences*, *7*(3), 1853–1882, doi:10.1137/130929886.

Fiorini, S., S. Massar, S. Pokutta, H. R. Tiwary, and R. de Wolf (2012), Linear vs. Semidefinite Extended Formulations: Exponential Separation and Strong Lower Bounds, in *STOC'12—Proceedings of the 2012 ACM Symposium on Theory of Computing*, pp. 95–106, ACM, New York, doi:10.1145/2213977.2213988.

Fisk, L. A. (2003), Acceleration of the solar wind as a result of the reconnection of open magnetic flux with coronal loops, *J. Geophys. Res.-Space*, *108*, 1157–+, doi:10.1029/2002JA009284.

Fisk, L. A., and N. A. Schwadron (2001), Origin of the Solar Wind: Theory, *Space Sci. Rev.*, *97*, 21–33.

Fisk, L. A., N. A. Schwadron, and T. H. Zurbuchen (1998), On the Slow Solar Wind, *Space Sci. Rev.*, *86*, 51–60, doi:10.1023/A:1005015527146.

Fisk, L. A., T. H. Zurbuchen, and N. A. Schwadron (1999), Coronal Hole Boundaries and their Interactions with Adjacent Regions, *Space Sci. Rev.*, *87*, 43–54, doi:10.1023/A:1005153730158.

Flum, J., and M. Gorhe (2006), *Parameterized Complexity Theory*, Springer.

Forero, P. A., and G. B. Giannakis (2012), Sparsity-exploiting robust multidimensional scaling, *IEEE Transactions on Signal Processing*, *60*(8), 4118–4134.

France, S. L., and J. D. Carroll (2010), Two-way multidimensional scaling: A review, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(5), 644–661.

Frogner, C., C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio (2015), Learning with a Wasserstein Loss, in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, p. 2053–2061, MIT Press, Cambridge, MA, USA.

Fukushima, K., and S. Miyake (1982), Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position, *Pattern Recognition*, *15*(6), 455 – 469, doi:https://doi.org/10.1016/0031-3203(82)90024-3.

Gabow, H. (1985), Scaling algorithms for network problems, *J. Comput. Syst. Sci.*, *31*, 148–168.

Ganea, O., G. Becigneul, and T. Hofmann (2018), Hyperbolic Neural Networks, in *Advances in Neural Information Processing Systems 31*, pp. 5345–5355, Curran Associates, Inc.

Gangbo, W., and A. Swiech (1998), Optimal maps for the multidimensional monge-kantorovich problem.

Garg, N., V. Vazirani, and M. Yannakakis (1996), Approximate max-flow min-(multi)cut theorems and their applications, *SIAM J. Comput.*, *25*(2), 235–251.

Garstka, M., M. Cannon, and P. Goulart (2019), COSMO: A conic operator splitting method for large convex problems, in *European Control Conference*, doi:10.23919/ECC.2019.8796161.

Geiss, J., G. Gloeckler, and R. von Steiger (1995), Origin of the Solar Wind From Composition Data, *Space Sci. Rev.*, *72*, 49–60, doi:10.1007/BF00768753.

Genevay, A., M. Cuturi, G. Peyré, and F. Bach (2016), Stochastic optimization for large-scale optimal transport, in *Advances in Neural Information Processing Systems 29*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, pp. 3440–3448, Curran Associates, Inc.

Gilbert, A. C., and L. Jain (2017), If it ain't broke, don't fix it: Sparse metric repair, *ArXiv e-prints*.

Gilbert, A. C., and L. Jain (2017), If it ain't broke, don't fix it: Sparse metric repair, *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 612–619.

Gilbert, A. C., and L. Jain (2017), If it Ain't Broke, Don't Fix it: Sparse Metric Repair, in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 612–619, doi:10.1109/ALLERTON.2017.8262793.

Gilbert, A. C., and R. Sonthalia (2018a), Unsupervised Metric Learning in Presence of Missing Data, *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 313–321.

Gilbert, A. C., and R. Sonthalia (2018b), Generalized metric repair on graphs, *in submission*.

Gilbert, A. C., and R. Sonthalia (2018c), Unsupervised metric learning in presence of missing data, in *56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018, Monticello, IL, USA, October 2-5, 2018*, pp. 313–321.

Gilbert, A. C., and R. Sonthalia (2020a), Project and Forget: Solving Large Scale Metric Constrained Problems.

Gilbert, A. C., and R. Sonthalia (2020b), Project and Forget: Solving Large Scale Metric Constrained Problems.

Gloeckler, G., and J. Geiss (1998), Interstellar and Inner Source Pickup Ions Observed with SWICS on ULYSSES, *Space Sci. Rev.*, *86*, 127–159, doi:10.1023/A:1005019628054.

Gloeckler, G., and J. Geiss (2001), Composition of the Local Interstellar Cloud from Observations of Interstellar Pickup Ions, in *Joint SOHO/ACE workshop "Solar and Galactic Composition"*, *American Institute of Physics Conference Series*, vol. 598, edited by R. F. Wimmer-Schweingruber, pp. 281–289.

Gloeckler, G., T. H. Zurbuchen, and J. Geiss (2003), Implications of the observed anticorrelation between solar wind speed and coronal electron temperature, *J. Geophys. Res.-Space*, *108*, 1158–+, doi:10.1029/2002JA009286.

Glunt, W., T. L. Hayden, S. Hong, and J. Wells (1990a), An alternating projection algorithm for computing the nearest euclidean distance matrix, *SIAM J. Matrix Anal. Appl.*, *11*(4), 589–600.

Glunt, W., T. L. Hayden, S. Hong, and J. Wells (1990b), An Alternating Projection Algorithm for Computing the Nearest Euclidean Distance Matrix, *SIAM J. Matrix Anal. Appl.*, *11*(4), 589–600, doi:10.1137/0611042.

Glunt, W., T. L. Hayden, S. Hong, and J. Wells (1990c), An alternating projection algorithm for computing the nearest euclidean distance matrix, *SIAM Journal on Matrix Analysis and Applications*, *11*(4), 589–600, doi:10.1137/0611042.

Glunt, W., T. L. Hayden, S. Hong, and J. Wells (1990d), An alternating projection algorithm for computing the nearest Euclidean distance matrix, *SIAM J. Matrix Anal. Appl.*, *11*(4), 589–600, doi:10.1137/0611042.

Gnansambandam, A., and S. Chan (2020), One size fits all: Can we train one denoiser for all noise levels?, in *ICML*.

Gombosi, T. I. (1998), *Physics of the Space Environment*, 339 pp., Cambridge University Press, Cambridge, UK.

Gomory, R. E. (1960), An Algorithm for the Mixed Integer Problem.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), Generative adversarial nets, in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, pp. 2672–2680, Curran Associates, Inc.

Gorissen, B. L., İhsan Yanıkoğlu, and D. den Hertog (2015), A practical guide to robust optimization, *Omega*, *53*, 124 – 137, doi:https://doi.org/10.1016/j.omega.2014.12.006.

Gorman, R. P., and T. Sejnowski (1988), Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks*, *1*, 75–89.

Gosling, J. T., J. Birn, and M. Hesse (1995), Three-dimensional magnetic reconnection and the magnetic topology of coronal mass ejection events, *Geophys. Res. Lett.*, *22*, 869–872.

Götze, F., and A. Tikhomirov (2003), Rate of convergence to the semi-circular law, *Probability Theory and Related Fields*, *127*, 228–276.

Götze, F., and A. Tikhomirov (2004), Rate of convergence in probability to the marchenko-pastur law, *Bernoulli*, *10*, 503–548.

Götze, F., and A. Tikhomirov (2005), The rate of convergence for spectra of gue and lue matrix ensembles, *Central European Journal of Mathematics*, *3*, 666–704.

Götze, F., and A. Tikhomirov (2011), On the rate of convergence to the marchenko–pastur distribution, *arXiv: Probability*.

Gower, J. (1985a), Properties of euclidean and non-euclidean distance matrices, *Linear Algebra and its Applications*, *67*, 81–97, doi:10.1016/0024-3795(85)90187-9.

Gower, J. C. (1975), Generalized procrustes analysis, *Psychometrika*, *40*(1), 33–51, doi:10.1007/BF02291478.

Gower, J. C. (1982), Euclidean distance geometry, *Math. Sci.*, *7*(1), 1–14.

Gower, J. C. (1985b), Properties of Euclidean and non-Euclidean distance matrices, *Linear Algebra and Its Applications*, *67*(C), 81–97, doi:10.1016/0024-3795(85)90187-9.

Graham, R. L., and P. M. Winkler (1985), On Isometric Embeddings of Graphs, *Trans. Amer. Math. Soc.*, *288*(2), 527–536.

Grohe, M. (2020), word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*.

Gromov, M. (1987), *Hyperbolic Groups*, pp. 75–263, Springer New York, New York, NY.

Grötschel, M., L. Lovász, and A. Schrijver (1981), The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica*, *1*(2), 169–197, doi:10.1007/BF02579273.

Gulcehre, C., et al. (2019), Hyperbolic Attention Networks, in *International Conference on Learning Representations*.

Guvenir, H. A., B. Açar, G. Demiroz, and A. Çekin (1997), A supervised machine learning algorithm for arrhythmia analysis, *Computers in Cardiology 1997*, pp. 433–436.

Hajiaghayi, M. T., R. Khandekar, and G. Kortsarz (2017), Fpt hardness for clique and set cover with super exponential time in k.

Hamann, M. (2018), On the Tree-Likeness of Hyperbolic Spaces, *Mathematical Proceedings of the Cambridge Philosophical Society*, *164*(2), 345–361, doi:10.1017/S0305004117000238.

Hayden, T., and J. Wells (1988), Approximation by matrices positive semidefinite on a subspace, *Linear Algebra and its Applications*, *109*, 115 – 130, doi:https://doi.org/10.1016/0024-3795(88)90202-9.

Heiser, W. J., and J. Meulman (1983), Constrained multidimensional scaling, including confirmation, *Applied Psychological Measurement*, *7*(4), 381–404.

Hinton, G. E., and S. Roweis (2002), Stochastic neighbor embedding, in *NIPS*.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2012), Improving neural networks by preventing co-adaptation of feature detectors, *ArXiv*, *abs/1207.0580*.

Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural Comput.*, *9*(8), 1735–1780, doi:10.1162/neco.1997.9.8.1735.

Hoeffding, W. (1963), Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association*, *58*(301), 13–30.

Hundhausen, A. J., H. E. Gilbert, and S. J. Bame (1968), Ionization State of the Interplanetary Plasma, *J. Geophys. Res.*, *73*, 5485–5493.

Hwang, K., and W. Sung (2016), Character-level language modeling with hierarchical recurrent neural networks, *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5720–5724.

IBM (), Cplex.

Indyk, P. (1999), Sublinear Time Algorithms for Metric Space Problems, in *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*, pp. 428–432, ACM, New York, doi:10.1145/301250.301366.

Indyk, P., and J. Matoušek (2004), Low-distortion embeddings of finite metric spaces, in *Handbook of Discrete and Computational Geometry*, pp. 177–196, CRC Press.

Ipavich, F. M., et al. (1998), Solar wind measurements with SOHO: The CELIAS/MTOF proton monitor, *J. Geophys. Res.*, *103*, 17,205–17,214, doi: 10.1029/97JA02770.

Iusem, A. N. (1991), On Dual Convergence and the Rate of Primal Convergence of Bregman's Convex Programming Method, *SIAM J. Optim.*, *1*(3), 401–423, doi:10.1137/0801025.

Iusem, A. N., and A. R. De Pierro (1990), On the Convergence Properties of Hildreth's Quadratic Programming Algorithm, *Math. Programming*, *47*(1, (Ser. A)), 37–51, doi:10.1007/BF01580851.

Jackson, J. D. (1999), *Classical Electrodynamics*, 3rd ed., 808 pp., John Wiley & Sons, United States of America.

Jahren, B. (2012), Geometric Structures in Dimension Two.

Jaimovich, A., G. Elidan, H. Margalit, and N. Friedman (2006), Towards an Integrated Protein-Protein Interaction Network: A Relational Markov Network Approach, *Journal of computational biology : a journal of computational molecular cell biology*, *13 2*, 145–64.

Joachims, T., T. Finley, and C.-N. J. Yu (2009), Cutting-Plane Training of Structural SVMs, *Machine Learning*, *77*, 27–59.

Johnson, W. (1984), Extensions of lipschitz mappings into hilbert space, *Contemporary mathematics*, *26*, 189–206.

Johnson, W. B., and J. Lindenstrauss (1984), Extensions of Lipschitz mappings into a Hilbert space, in *Conference in modern analysis and probability (New Haven, Conn., 1982)*, *Contemp. Math.*, vol. 26, pp. 189–206, Amer. Math. Soc., Providence, RI.

Jutten, C., and J. Hérault (1991), Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture, *Signal Process.*, *24*, 1–10.

Karp, R. (1972), Reducibility among combinatorial problems, in *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pp. 85–103.

Khot, S. (2002), On the power of unique 2-prover 1-round games, in *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 767–775.

Khot, S., and O. Regev (2003), Vertex cover might be hard to approximate to within 2-ϵ.

Khot, S., and O. Regev (2008a), Vertex cover might be hard to approximate to within 2-epsilon, *J. Comput. Syst. Sci.*, *74*(3), 335–349, doi:10.1016/j.jcss.2007.06.019.

Khot, S., and O. Regev (2008b), Vertex cover might be hard to approximate to within 2-epsilon, *J. Comput. Syst. Sci.*, *74*(3), 335–349.

Khrulkov, V., L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky (2019), Hyperbolic Image Embeddings.

Kleinman, D., and M. Athans (1968), The design of suboptimal linear time-varying systems, *IEEE Transactions on Automatic Control*, *13*(2), 150–159.

Klimovskaia, A., D. Lopez-Paz, L. Bottou, and M. Nickel (2019), Poincaré Maps for Analyzing complex Hierarchies in Single-Cell Data, *bioRxiv*, doi:10.1101/689547.

Kolar, M., and H. Liu (2012), Marginal regression for multitask learning, in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, vol. 22, edited by N. D. Lawrence and M. Girolami, pp. 647–655, PMLR, La Palma, Canary Islands.

Kolouri, S., S. R. Park, M. Thorpe, D. Slepcev, and G. K. Rohde (2017), Optimal Mass Transport: Signal Processing and Machine-Learning Applications, *IEEE Signal Processing Magazine*, *34*(4), 43–59, doi:10.1109/MSP.2017.2695801.

Kraft, D. H. (1988), A software package for sequential quadratic programming.

Krioukov, D., F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá (2010), Hyperbolic Geometry of Complex Networks, *Physical Review E*, *82*(3), doi:10.1103/physreve.82.036106.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012), Imagenet classification with deep convolutional neural networks, *Communications of the ACM*, *60*, 84 – 90.

Kroonenberg, P. M. (2008), *Applied multiway data analysis*, vol. 702, John Wiley & Sons.

Kruskal, J. B. (1964), Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika*, *29*(1), 1–27, doi:10.1007/BF02289565.

Kuhnle, A., V. Crawford, and M. Thai (2018), Network resilience and the length-bounded multicut problem: Reaching the dynamic billion-scale with guarantees, in *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 81–83.

Laurent, M. (1998a), A connection between positive semidefinite and euclidean distance matrix completion problems, *Linear Algebra and its Applications*, *273*(1), 9 – 22.

Laurent, M. (1998b), A Connection Between Positive Semidefinite and Euclidean Distance Matrix Completion Problems, *Linear Algebra Appl.*, *273*, 9–22, doi:10.1016/S0024-3795(98)90126-4.

Laurent, M. (1998c), A connection between positive semidefinite and Euclidean distance matrix completion problems, *Linear Algebra and Its Applications*, *273*(1-3), 9–22, doi:10.1016/S0024-3795(97)83714-7.

LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel (1990), Handwritten digit recognition with a back-propagation network, in *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky, pp. 396–404, Morgan-Kaufmann.

Lee, E. (2017a), Improved hardness for cut, interdiction, and firefighter problems, in *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 92:1–92:14.

Lee, E. (2017b), Improved hardness for cut, interdiction, and firefighter problems, in *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 92:1–92:14.

Leskovec, J., and A. Krevl (2014), SNAP Datasets: Stanford Large Network Dataset Collection.

Lichtenstein, D. (1982), Planar formulae and their uses, *SIAM Journal on Computing*, *11*(2), 329–343.

Liero, M., A. Mielke, and G. Savaré (2017), Optimal Entropy-Transport Problems and a New Hellinger–Kantorovich Distance Between Positive Measures, *Inventiones mathematicae*, *211*(3), 969–1117, doi:10.1007/s00222-017-0759-8.

Lin, B., A. Monod, and R. Yoshida (2018), Tropical Foundations for Probability Statistics on Phylogenetic Tree Space, *ArXiv e-prints*.

Lin, T., T. Guo, and K. Aberer (2017), Hybrid neural networks for learning the trend in time series, in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 2273–2279, AAAI Press.

Linial, N. (2002), Finite Metric Spaces: Combinatorics, Geometry and Algorithms, in *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, SCG '02, pp. 63–63, ACM, New York, NY, USA.

Linial, N., E. London, and Y. Rabinovich (1995a), The Geometry of Graphs and Some of its Algorithmic Applications, *Combinatorica*, *15*(2), 215–245.

Linial, N., E. London, and Y. Rabinovich (1995b), The geometry of graphs and some of its algorithmic applications, *Combinatorica*, *15*(2), 215–245, doi:10.1007/BF01200757.

Lorenz, D. A., P. Manns, and C. Meyer (2019), Quadratically Regularized Optimal Transport.

Lucien, B. (2001), *An alternative point of view on Lepski's method*, *Lecture Notes–Monograph Series*, vol. Volume 36, pp. 113–133, Institute of Mathematical Statistics, Beachwood, OH, doi:10.1214/lnms/1215090065.

Maaten, L. V. D., and G. E. Hinton (2008), Visualizing data using t-sne, *Journal of Machine Learning Research*, *9*, 2579–2605.

Mandanas, F. D., and C. L. Kotropoulos (2016), Robust multidimensional scaling using a maximum correntropy criterion, *IEEE Transactions on Signal Processing*, *65*(4), 919–932.

Matoušek, J. (2013), *Lecture notes on metric embeddings*, available at: `http://kam.mff.cuni.cz/~matousek/ba-a4.pdf`.

McComas, D. J., et al. (2000), Solar wind observations over Ulysses' first full polar orbit, *J. Geophys. Res.*, *105*, 10,419–10,434, doi:10.1029/1999JA000383.

Meyer, C. D., Jr. (1973), Generalized inversion of modified matrices, *SIAM Journal on Applied Mathematics*, *24*(3), 315–323, doi:10.1137/0124033.

Mikolov, T., M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur (2010), Recurrent neural network based language model, in *INTERSPEECH*.

Mikolov, T., K. Chen, G. S. Corrado, and J. Dean (2013), Efficient estimation of word representations in vector space, in *ICLR*.

Million, E. (2007), The hadamard product elizabeth million april 12 , 2007 1 introduction and basic results.

MOSEK ApS (), Parametric fusion.

Nadakuditi, R. R. (2014), Optshrink: An algorithm for improved low-rank signal matrix denoising by optimal, data-driven singular value shrinkage, *IEEE Transactions on Information Theory*, *60*(5), 3002–3018, doi:10.1109/TIT.2014.2311661.

Narayanan, A., M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal (2017), graph2vec: Learning distributed representations of graphs, *ArXiv*, *abs/1707.05005*.

Nedić, A. (2011), Random Algorithms for Convex Minimization Problems, *Math. Program.*, *129*(2, Ser. B), 225–253, doi:10.1007/s10107-011-0468-9.

Neelakantan, A., L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens (2015), Adding gradient noise improves learning for very deep networks, *ArXiv*, *abs/1511.06807*.

Nesterov, Y. E. (1983), A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$, *Dokl. Akad. Nauk SSSR*, *269*(3), 543–547.

Nickel, M., and D. Kiela (2017), Poincaré Embeddings for Learning Hierarchical Representations, in *NIPS*.

Nickel, M., and D. Kiela (2018a), Learning continuous hierarchies in the lorentz model of hyperbolic geometry, *ArXiv*, *abs/1806.03417*.

Nickel, M., and D. Kiela (2018b), Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry, in *ICML*.

Nocedal, J., A. Wächter, and R. A. Waltz (2009), Adaptive barrier update strategies for nonlinear interior methods, *SIAM Journal on Optimization*, *19*(4), 1674–1693, doi:10.1137/060649513.

O'Donoghue, B., E. Chu, N. Parikh, and S. Boyd (2016), Conic optimization via operator splitting and homogeneous self-dual embedding, *Journal of Optimization Theory and Applications*, *169*(3), 1042–1068.

O'Donoghue, B., E. Chu, N. Parikh, and S. Boyd (2019), SCS: Splitting conic solver, version 2.1.2, `https://github.com/cvxgrp/scs`.

Palacios-Gomez, F., L. Lasdon, and M. Engquist (1982), Nonlinear Optimization by Successive Linear Programming, *Management Science*, *28*(10), 1106–1120.

Palaniappan, S., and R. Adhikari (2017), Deep learning the indus script, *CoRR*, *abs/1702.00523*.

Pan, X., D. S. Papailiopoulos, S. Oymak, B. Recht, K. Ramchandran, and M. I. Jordan (2015), Parallel Correlation Clustering on Big Graphs, in *NIPS*.

Panagopoulos, M., C. Papaodysseus, P. Rousopoulos, D. Dafi, and S. Tracy (2009), Automatic writer identification of ancient greek inscriptions, *IEEE Trans. Pattern Anal. Mach. Intell.*, *31*(8), 1404–1414, doi:10.1109/TPAMI.2008.201.

Pandove, D., S. Goel, and R. Rani (2018), Correlation Clustering Methodologies and their Fundamental Results, *Expert Systems*.

Parker, E. (1959), Extension of the Solar Corona into Interplanetary Space, *J. Geophys. Res.*, *64*, 1675–1681.

Parker, E. N. (1958), Dynamics of the Interplanetary Gas and Magnetic Fields., *ApJ*, *128*, 664–676.

Peleg, D., and J. D. Ullman (1989), An Optimal Synchronizer for the Hypercube, *SIAM Journal on Computing*, *18*(4), 740–747.

Petric Maretic, H., M. El Gheche, G. Chierchia, and P. Frossard (2019), Got: An optimal transport framework for graph comparison, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, pp. 13,876–13,887, Curran Associates, Inc.

Peyré, G., and M. Cuturi (2018), Computational Optimal Transport.

Poirrier, L., and J. Yu (2019), On the Depth of Cutting Planes, *arXiv e-prints*, arXiv:1903.05304.

Polyak, B. T. (2001), Random Algorithms for Solving Convex Inequalities, in *Inherently parallel algorithms in feasibility and optimization and their applications (Haifa, 2000)*, *Stud. Comput. Math.*, vol. 8, pp. 409–422, North-Holland, Amsterdam, doi:10.1016/S1570-579X(01)80024-0.

Poole, B., J. Sohl-Dickstein, and S. Ganguli (2014), Analyzing noise in autoencoders and deep networks, *ArXiv*, *abs/1406.1831*.

Prechelt, L. (1998), *Early Stopping - But When?*, pp. 55–69, Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/3-540-49430-8_3.

Pretorius, A., S. Kroon, and H. Kamper (2018), Learning dynamics of linear denoising autoencoders, *ArXiv*, *abs/1806.05413*.

Prim, R. C. (1957), Shortest Connection Networks and Some Generalizations, *The Bell System Technical Journal*, *36*(6), 1389–1401.

Qi, H., and X. Yuan (2014a), Computing the nearest euclidean distance matrix with low embedding dimensions, *Mathematical Programming*, *147*, 351–389.

Qi, H., and X. Yuan (2014b), Computing the nearest euclidean distance matrix with low embedding dimensions, *Mathematical Programming*, *147*, 351–389.

Qi, H.-D., and X. Yuan (2014c), Computing the nearest euclidean distance matrix with low embedding dimensions, *Mathematical Programming*, *147*(1), 351–389.

Rao, N. R., and A. Edelman (2008), The polynomial method for random matrices, *Foundations of Computational Mathematics*, *8*, 649–702.

Redko, I., N. Courty, R. Flamary, and D. Tuia (2019), Optimal transport for multi-source domain adaptation under target shift, in *Proceedings of Machine Learning Research*, *Proceedings of Machine Learning Research*, vol. 89, pp. 849–858, PMLR.

ren Fang, H., and D. P. O'Leary (2012), Euclidean distance matrix completion problems, *Optimization Methods and Software*, *27*(4-5), 695–717.

Rockafellar, R. (1970), *Convex Analysis*, Princeton Landmarks in Mathematics and Physics, Princeton University Press.

Rossi, R. A., and N. K. Ahmed (2015), The network data repository with interactive graph analytics and visualization, in *AAAI*.

Rothvoss, T. (2014), The Matching Polytope has Exponential Extension Complexity, in *STOC'14—Proceedings of the 2014 ACM Symposium on Theory of Computing*, pp. 263–272, ACM, New York.

Roweis, S. T., and L. K. Saul (2000), Nonlinear dimensionality reduction by locally linear embedding, *Science*, *290*(5500), 2323–2326, doi:10.1126/science.290.5500.2323.

Rozemberczki, B., C. Allen, and R. Sarkar (2019), Multi-scale attributed node embedding.

Ruggles, C., N. Veldt, and D. F. Gleich (2019), A Parallel Projection Method for Metric Constrained Optimization, *arXiv e-prints*, arXiv:1901.10084.

Sahni, S., and T. Gonzalez (1976), P-complete approximation problems, *J. ACM*, *23*(3), 555–565, doi:10.1145/321958.321975.

Saitou, N., and M. Nei (1987), The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees., *Molecular biology and evolution*, *4 4*, 406–25.

Sala, F., C. De Sa, A. Gu, and C. Re (2018), Representation Tradeoffs for Hyperbolic Embeddings, *Proceedings of the 35th International Conference on Machine Learning*, pp. 4460–4469.

Santambrogio, F. (2014), Introduction to Optimal Transport Theory, doi:10.1017/CBO9781107297296.002.

Sarich, V. M. (1969), Pinniped Phylogeny, *Systematic Biology*, *18*(4), 416–422.

Sarkar, R. (2012), Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane, in *Proceedings of the 19th International Conference on Graph Drawing*, GD'11, pp. 355–366, Springer-Verlag, Berlin, Heidelberg.

Schmitzer, B. (2019), Stabilized sparse scaling algorithms for entropy regularized transport problems, *SIAM J. Scientific Computing*, *41*, A1443–A1481.

Schoenberg, I. J. (1935), Remarks to maurice fréchet's article "sur la définition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de hilbert". annals of mathematics 36(3.

Schoenberg, I. J. (1938a), Metric Spaces and Positive Definite Functions, *Trans. Amer. Math. Soc.*, *44*(3), 522–536, doi:10.2307/1989894.

Schoenberg, I. J. (1938b), Metric Spaces and Positive Definite Functions, *Transactions of the American Mathematical Society*, *44*(3), 522, doi:10.2307/1989894.

Scholz, M., F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig (2005), Non-linear pca: a missing data approach, *Bioinformatics*, *21*(20), 3887–3895, doi:10.1093/bioinformatics/bti634.

Schrieber, J., D. Schuhmacher, and C. Gottschlich (2017), DOTmark – A Benchmark for Discrete Optimal Transport, *IEEE Access*, *5*, 271–282.

Seguy, V., B. B. Damodaran, R. Flamary, N. Courty, A. Rolet, and M. Blondel (2018), Large scale optimal transport and mapping estimation, in *International Conference on Learning Representations*.

Shepard, R. N. (1962a), The analysis of proximities: multidimensional scaling with an unknown distance function. i., *Psychometrika*, *27*(2), 125–140.

Shepard, R. N. (1962b), The analysis of proximities: Multidimensional scaling with an unknown distance function. ii, *Psychometrika*, *27*(3), 219–246.

Shorten, C., and T. Khoshgoftaar (2019), A survey on image data augmentation for deep learning, *Journal of Big Data*, *6*, 1–48.

Sidiropoulos, A., D. Wang, and Y. Wang (2017), Metric embeddings with outliers, in *Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 670–689.

Sierksma, G. (1996), Linear and integer programming - theory and practice, in *Pure and applied mathematics*.

Sietsma, J., and R. J. F. Dow (1991), Creating artificial neural networks that generalize, *Neural Networks*, *4*, 67–79.

Sigillito, V., S. Wing, L. Hutton, and K. Baker (1989), Classification of radar returns from the ionosphere using neural networks.

Slater, M. (2014), *Lagrange Multipliers Revisited*, pp. 293–306, Springer Basel, Basel, doi:10.1007/978-3-0348-0439-4_14.

Sokal, R., C. Michener, and U. of Kansas (1958), *A Statistical Method for Evaluating Systematic Relationships*, University of Kansas science bulletin, University of Kansas.

Solomon, J., F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas (2015), Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains, *ACM Trans. Graph.*, *34*(4), doi:10.1145/2766963.

Sontag, D., and T. S. Jaakkola (2008), New Outer Bounds on the Marginal Polytope, in *Advances in Neural Information Processing Systems 20*, edited by J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, pp. 1393–1400, Curran Associates, Inc.

Sontag, D., T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss (2008), Tightening LP Relaxations for MAP Using Message Passing, in *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pp. 503–510, AUAI Press, Arlington, Virginia, United States.

Sontag, D., D. K. Choe, and Y. Li (2012), Efficiently Searching for Frustrated Cycles in MAP Inference, in *Proceedings of the Twenty-Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pp. 795–804, AUAI Press, Corvallis, Oregon.

Sontag, D. A., and T. S. Jaakkola (2007), On Iteratively Constraining the Marginal Polytope for Approximate Inference and MAP, *Semantic Scholar*.

Sonthalia, R., and A. C. Gilbert (2018), Unsupervised metric learning in the presence of missing data, *Proceedings of Allerton Conference on Communication, Computing, and Control*.

Sonthalia, R., and A. Gilbert (2020a), Project and forget: Solving large-scale metric constrained problems, in *Submitted to JMLR*.

Sonthalia, R., and A. C. Gilbert (2020b), Dual regularized optimal transport, in *Submitted to ICML 2021*.

Sonthalia, R., and A. C. Gilbert (2020c), Tree! i am no tree! i am a low dimensional hyperbolic embedding, in *NeurIPS*.

Sonthalia, R., and R. R. Nadakuditi (2021), How to optimally train stakced linear denoising autoencoders?, in *Submitted to ICML 2021*.

Sonthalia, R., F. Schironi, and R. R. Nadakuditi (2020), Deepgreek: A framework for greek text reconstruction, in *Submitted to NAACL*.

Sonthalia, R., G. V. Buskirk, B. Raichel, and A. C. Gilbert (2021), How can, in *Submitted to ICML 2021*.

Souto, M., J. D. Garcia, and Á. Veiga (2018), Exploiting low-rank structure in semidefinite programming by approximate operator splitting, *arXiv preprint arXiv:1810.05231*.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014), Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Stein, C. (1956), Inadmissibility of the usual estimator for the mean of a multivariate normal distribution, in *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pp. 197–206, University of California Press, Berkeley, Calif.

Stellato, B., G. Banjac, P. Goulart, A. Bemporad, and S. Boyd (2017), OSQP: An operator splitting solver for quadratic programs, *ArXiv e-prints*.

Stoeckius, M., C. Hafemeister, W. Stephenson, B. Houck-Loomis, P. K. Chattopadhyay, H. Swerdlow, R. Satija, and P. Smibert (2017), Simultaneous epitope and transcriptome measurement in single cells, *Nature Methods*, *14*(9), 865–868, doi: 10.1038/nmeth.4380.

Sun, Q. H., S. Lee, and D. Batra (2017), Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7215–7223.

Sutskever, I., J. Martens, and G. Hinton (2011), Generating text with recurrent neural networks, in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 1017–1024, Omnipress, USA.

Sutskever, I., O. Vinyals, and Q. V. Le (2014), Sequence to sequence learning with neural networks, in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, pp. 3104–3112, Curran Associates, Inc.

Suárez Díaz, J. L., S. García, and F. Herrera (2018), A Tutorial on Distance Metric Learning: Mathematical Foundations, Algorithms and Software, *arxiv eprint*.

Swanson, K., L. Yu, and T. Lei (2020), Rationalizing text matching: Learning sparse alignments via optimal transport, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5609–5626, Association for Computational Linguistics, Online, doi:10.18653/v1/2020.acl-main.496.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus (2014), Intriguing properties of neural networks, *CoRR*, *abs/1312.6199*.

Taguchi, Y.-H., and Y. Oono (2005), Relational patterns of gene expression via non-metric multidimensional scaling analysis, *Bioinformatics*, *21*(6), 730–740.

Takeda, H., S. Farsiu, and P. Milanfar (2007), Kernel regression for image processing and reconstruction, *IEEE Transactions on Image Processing*, *16*, 349–366.

Tang, Y. (2013), Deep Learning Using Support Vector Machines, *ICML 2013 Challenges in Representation Learning*, *abs/1306.0239*.

Tenenbaum, J., V. de Silva, and J. Langford (2000a), A global geometric framework for nonlinear dimensionality reduction., *Science*, *290 5500*, 2319–23.

Tenenbaum, J. B., V. d. Silva, and J. C. Langford (2000b), A global geometric framework for nonlinear dimensionality reduction, *Science*, *290*(5500), 2319–2323, doi:10.1126/science.290.5500.2319.

Tenenbaum, J. B., V. d. Silva, and J. C. Langford (2000c), A global geometric framework for nonlinear dimensionality reduction, *Science*, *290*(5500), 2319–2323, doi:10.1126/science.290.5500.2319.

The Packard Humanities Institute (2005), PHI greek inscriptions, `https://inscriptions.packhum.org/`, accessed on 2019-04-24.

Tian, C., Y. Xu, L. Fei, and K. Yan (2018), Deep learning for image denoising: A survey, *ArXiv*, *abs/1810.05052*.

Tian, C., L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin (2020), Deep learning on image denoising: An overview, *Neural networks : the official journal of the International Neural Network Society*, *131*, 251–275.

Torgerson, W. S. (1952), Multidimensional scaling i: Theory and method.

Torgerson, W. S. (1958), Theory and methods of scaling.

Tropp, J., and A. Gilbert (2007), Signal recovery from random measurements via orthogonal matching pursuit, *IEEE Transactions on Information Theory*, *53*, 4655–4666.

Valiant, L. (1979), The complexity of enumeration and reliability problems, *SIAM J. Comput.*, *8*(3), 410–421.

van Dijk, D., et al. (2018), Recovering gene interactions from single-cell data using data diffusion, *Cell*, doi:https://doi.org/10.1016/j.cell.2018.05.061.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017), Attention is all you need, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp. 5998–6008, Curran Associates, Inc.

Vazirani, V. (2001), *Approximation Algorithms*, Springer-Verlag New York, Inc., New York, NY, USA.

Veldt, N., D. Gleich, A. Wirth, and J. Saunderson (2019), Metric-Constrained Optimization for Graph Clustering Algorithms, *SIAM Journal on Mathematics of Data Science*, *1*, 333–355, doi:10.1137/18M1217152.

Verbeek, K., and S. Suri (2016), Metric Embedding, Hyperbolic Space, and Social Networks, *Computational Geometry*, *59*, 1 – 12.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010), Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, *J. Mach. Learn. Res.*, *11*, 3371–3408.

Vinyals, O., A. Toshev, S. Bengio, and D. Erhan (2015), Show and tell: A neural image caption generator, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, doi:10.1109/CVPR.2015.7298935.

von Steiger, R., and J. Geiss (1993), Solar wind composition and expectations for high solar latitudes, *Advances in Space Research*, *13*, 63–74, doi:10.1016/0273-1177(93)90392-O.

von Steiger, R., J. Geiss, G. Gloeckler, and A. B. Galvin (1995), Kinetic Properties of Heavy Ions in the Solar Wind From SWICS/Ulysses, *Space Sci. Rev.*, *72*, 71–76, doi:10.1007/BF00768756.

Wan, L., M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus (2013), Regularization of neural networks using dropconnect, in *Proceedings of the 30th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 28, edited by S. Dasgupta and D. McAllester, pp. 1058–1066, PMLR, Atlanta, Georgia, USA.

Wang, F., and J. Sun (2015), Survey on distance metric learning and dimensionality reduction in data mining, *Data Mining and Knowledge Discovery*, *29*(2), 534–564.

Wang, L., Z. Cao, Y. Xia, and G. de Melo (2016), Morphological segmentation with window lstm neural networks, in *AAAI Conference on Artificial Intelligence*.

Wang, M., and D. P. Bertsekas (2013), Incremental Constraint Projection-Proximal Methods for Nonsmooth Convex Optimization, *http://www.mit.edu/ dimitrib/*.

Wang, M., Y. Chen, J. Liu, and Y. Gu (2015), Random Multi-Constraint Projection: Stochastic Gradient Methods for Convex Optimization with Many Constraints, *ArXiv*, *abs/1511.03760*.

Wang, Y., L. Xu, Y. Chen, and H. Wang (2013), A Scalable Approach for General Correlation Clustering, in *ADMA*.

Wang, Z., and T. Oates (2015), Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, in *AAAI Workshops*.

Weinberger, K. Q., and L. K. Saul (2004), Unsupervised learning of image manifolds by semidefinite programming, in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 988–995, doi:10.1109/CVPR.2004.1315272.

Wiener, N. (1949), Extrapolation, interpolation, and smoothing of stationary time series, with engineering applications.

Wimmer-Schweingruber, R. F. (2003), Solar Wind Composition, in *Solar Wind Ten*, *American Institute of Physics Conference Series*, vol. 679, edited by M. Velli, R. Bruno, F. Malara, and B. Bucci, pp. 577–582.

Xie, Y., M. Chen, H. Jiang, T. Zhao, and H. Zha (2019), On scalable and efficient computation of large scale optimal transport, in *ICML*.

Yang, K. D., and C. Uhler (2019), Scalable Unbalanced Optimal Transport Using Generative Adversarial Networks, in *International Conference on Learning Representations*.

Yannakakis, M. (1981a), Computing the minimum fill-in is np-complete, *SIAM Journal on Algebraic Discrete Methods*, *2*, 77–79.

Yannakakis, M. (1981b), Edge-deletion problems, *SIAM Journal on Computing*, *10*(2), 297–13.

Yanover, C., T. Meltzer, and Y. Weiss (2006), Linear Programming Relaxations and Belief Propagation—An Empirical Study, *J. Mach. Learn. Res.*, *7*, 1887–1907.

Young, R. (2008), Notes on Asymptotic Cones.

Yu, T., and C. M. De Sa (2019), Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models, in *Advances in Neural Information Processing Systems 32*, pp. 2021–2031, Curran Associates, Inc.

Yu, Y., T. Wang, and R. J. Samworth (2014), A useful variant of the Davis–Kahan theorem for statisticians, *Biometrika*, *102*(2), 315–323, doi:10.1093/biomet/asv008.

Zeisel, A., et al. (2015), Cell Types in the Mouse Cortex and Hippocampus Revealed by Single-cell rna-seq, *Science*, *347*(6226), 1138–1142.

Zhang, X., and Y. LeCun (2015), Text understanding from scratch, *ArXiv*, *abs/1502.01710*.

Zhang, X., J. Zhao, and Y. LeCun (2015), Character-level convolutional networks for text classification, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pp. 649–657, Curran Associates, Inc.

Á. Carreira-Perpiñán, M., and Z. Lu (2011), Manifold learning and missing data recovery through unsupervised regression, *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 1014–1019.