Machine Learning for Flow Cytometry Data Analysis

by

Gyemin Lee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering : Systems)
in The University of Michigan
2011

To my parents Chung Hun Lee and Sang Ok Yoon,
and my brother Gye Myoung Lee

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisor Professor Clayton D. Scott. Without his guidance, stimulus and support over the years, the present work would not have been possible. His knowledge and insight were crucial for me to mature as a researcher. I am sincerely grateful to Dr. William G. Finn, who motivated me to pursue this exciting research project. Discussions with him were always a great resource about flow cytometry, and he inspired me to develop a number of interesting research ideas. I would also like to thank my dissertation committee members, Professor Jeffrey A. Fessler, Professor Alfred O. Hero III and Professor Long Nguyen. Their helpful suggestions and comments contributed greatly to my doctoral work.

I am much obliged to Dr. Lloyd Stoolman, Josh Jacques, Usha Kota and the Department of Pathology at the University of Michigan for providing me with data and helpful discussions. I am also grateful to the National Science Foundation and the Edwin R. Riethmiller Fellowship for their support to complete this work.

Many thanks to my friends Juseop Lee, Manhee Jeong, Se Young Chun, Joonki Noh, JooSeuk Kim, Yoo Jin Choi, Sung Jin Hwang, Hyun Jeong Cho, and Jae Young Park. Their genuine friendship and sincere advices always strengthen my will to keep going forward during difficult and stressful times. I have also greatly benefited from many friends in KEECS, and I thank Seunghyun Oh for his assistance during my year-long service to this wonderful community. I would also like to thank my friends, Youngkuk Hwang, Seungchul Lee, Yong Keun Hwang, Injae Hong, Holly Sohhyun Chung, and Min Sik Kim, who worked together with me as KSAG executives and given me unforgettable memories in Ann Arbor. I am also thankful to all the members in Professor Scott's research group, Gowtham Bellala, Joyce Liu, Takanori Watanabe, and Robert Vandermeulen, who have provided me with invaluable thoughts and feedback.

Finally, I would like to give my utmost gratitude to my parents and my brother. Their constant, unreserved support and love have always given me the strength to face various challenges in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Machine Learning for Flow Cytometry Data Analysis

by

Gyemin Lee

Chair: Clayton D. Scott

This thesis concerns the problem of automatic flow cytometry data analysis. Flow cytometry is a technique for rapid cell analysis and widely used in many biomedical and clinical laboratories. Quantitative measurements from a flow cytometer provide rich information about various physical and chemical characteristics of a large number of cells. In clinical applications, flow cytometry data is visualized on a sequence of two-dimensional scatter plots and analyzed through a manual process called “gating”. This conventional analysis process requires a large amount of time and labor and is highly subjective and inefficient. In this thesis, we present novel machine learning methods for flow cytometry data analysis to address these issues.

We first begin by a method for generating a high dimensional flow cytometry dataset from multiple low dimensional datasets. We present an imputation algorithm based on clustering and show that it improves upon a simple nearest neighbor based approach that often induces spurious clusters in the imputed data. This technique enables the analysis of multi-dimensional flow cytometry data beyond the fundamental measurement limits of instruments.

We then present two machine learning methods for automatic gating problems. Gating is a process of identifying interesting subsets of cell populations. Pathologists make clinical decisions by inspecting the results from gating. Unfortunately, this process is performed manually in most clinical settings and poses many challenges in high-throughput analysis.

The first approach is an unsupervised learning technique based on multivariate mixture models. Since measurements from a flow cytometer are often censored and truncated, standard model-fitting algorithms can cause biases and lead to poor gating results. We propose novel algorithms for fitting multivariate Gaussian mixture models to data that is truncated, censored, or truncated and censored.

Our second approach is a transfer learning technique combined with the low-density separation principle. Unlike conventional unsupervised learning approaches, this method can leverage existing datasets previously gated by domain experts to automatically gate a new flow cytometry data. Moreover, the proposed algorithm can adaptively account for biological variations in multiple datasets.

We demonstrate these techniques on clinical flow cytometry data and evaluate their effectiveness.

CHAPTER 1

Introduction

Flow cytometry is a technique widely used in many clinical and biomedical laboratories for rapid cell analysis. It has proven its usefulness in a number of tasks, especially hematopathology, the study of blood-related diseases. In modern clinical laboratories, pathologists use flow cytometry on a regular basis in the diagnosis of diseases such as acute leukemia, chronic lymphoproliferative disorder and malignant lymphomas.

Flow cytometry provides quantitative measurements of physical and chemical properties of individual cells. The measured attributes include cell size, granularity and expression level of different antigens on the cell surface. While recent technological advances make it possible to measure up to 20 different attributes simultaneously, the dimensionality of flow cytometry data ranges from 7–12 in most clinical applications. A single session produces multidimensional readouts of 10,000 to 1,000,000 cells. In clinical settings, these cells are typically prepared from the peripheral blood, lymph node or bone marrow of a patient.

Traditionally, however, the analysis of flow cytometry data heavily relies on manual processing. For diagnostic evaluation, pathologists visualize the flow cytometry data set with a series of two-dimensional scatter plots and inspect the shape, range, and other distributional characteristics of cell populations. Depending on specific diseases, pathologists use specialized software tools to draw line segments to select and identify interesting subsets of cells. They repeat this operation on a sequence of two-dimensional scatter plots until they purify the cell population to a homogeneous group. This process is performed manually in clinical settings.

This analytic process is labor-intensive and causes many problems in flow cytometry. Since pathologists or technologists resort to their experience and knowledge to choose the sequence of scatter plots and select cell populations on the plots, it is highly operator-dependent and subjective. Hence, the analysis results can differ and lead to different clinical decisions depending upon the individual who performs it. This problem causes more challenges when the analysis involves multiple data sets. Furthermore, biological variability from different patient health conditions and technical variability from instrument calibration also pose difficulties in the high throughput flow cytometric analysis.

In this thesis, we present novel machine learning methods to automate flow cytometry data analysis and provide robust analytic tools. We evaluate these techniques with clinical flow cytometry datasets.

1.1 Thesis Outline and Contributions

The goal of the work presented in this thesis is applying machine learning techniques to the analysis of flow cytometry data using statistical data modeling and learning principles.

Although analysis of flow cytometry data has traditionally considered one or two markers at a time, there has been increasing interest in multidimensional analysis. However, flow cytometers are limited in the number of markers they can jointly observe, which is typically a fraction of the number of markers of interest. For this reason, practitioners often perform multiple assays based on different, overlapping combinations of markers. In Chapter 2, we address the challenge of imputing the high dimensional jointly distributed values of marker attributes based on overlapping marginal observations. We show that simple nearest neighbor based imputation can lead to spurious subpopulations in the imputed data and introduce an alternative approach based on nearest neighbor imputation restricted to a cell's subpopulation. This requires us to perform clustering with missing data, which we address with a mixture model approach and novel EM algorithm. Since mixture model fitting may be ill-posed in this context, we also develop techniques to initialize the EM algorithm using domain knowledge.

As explained above, flow cytometry data, whether it is straight from a measurement instrument or synthesized from smaller data sets (Chapter 2), is analyzed via the process of gating. However, a flow cytometer can measure a limited range of signal strength and records each marker value within a fixed range. If a measurement falls outside this range, the value is censored and replaced by an indicator. Moreover, a large number of cell measurements can be excluded from recording by the judgment of an instrument operator. While these data deformations (truncation and censoring) exist in flow cytometry data, these issues have not been explicitly addressed in the literature. Therefore, automatic gating approaches based on mixture model fitting can result in biases in the parameter estimates and poor gating results when these data deformations are not properly handled. Chapter 3 focus on these two types of incomplete measurements of flow cytometry data. We present EM algorithms for fitting multivariate Gaussian mixture models to data that is truncated, censored, or truncated and censored. We show that truncation and censoring can be naturally handled together through their relation to the multivariate truncated Gaussian distribution.

While a vast amount of approaches are proposed to automate the gating process, most of them fall under the framework of unsupervised learning and mixture model fitting. However, these approaches have a number of difficulties. The non-elliptical shape of cell clusters requires complex parametric models and leads to complex inference problems. Furthermore, depending on the clustering procedure used, cluster labeling may be needed to find similar cell populations between multiple samples after automating gating. Finally, these algorithms are unsupervised, and do not fully leverage expert knowledge. In Chapter 4, we view the automatic gating problem as a transfer learning problem. By reformulating the problem, we can leverage existing data sets previously gated by experts, while accounting for biological variation, to automatically gate a new flow cytometry data. In particular, we show how expert knowledge can be adaptively transferred to a new data set by optimizing a low-density separation criterion.

Chapter 5 is not directly related to flow cytometry data analysis, but to the problems of multiple set estimation. One-class and cost-sensitive support vector machines (SVMs) are state-of-the-art machine learning methods for estimating density level sets and solving weighted classification problems, respectively. However, the solutions of these SVMs do not necessarily produce set estimates that are nested as the parameters controlling the density level or cost-asymmetry are continuously varied. Such nesting not only reflects the true sets being estimated, but is also desirable for applications requiring the simultaneous estimation of multiple sets, including clustering, anomaly detection, and ranking. We propose nested versions of one-class and cost-sensitive SVMs in this chapter. These methods are compared to conventional (non-nested) SVMs on synthetic and benchmark data sets, and are shown to exhibit more stable rankings and decreased sensitivity to parameter settings.

Based on the research on flow cytometry, the following publications were produced:

(1) G. Lee, W. Finn, and C. Scott. Statistical File Matching of Flow Cytometry Data. In *Journal of Biomedical Informatics*, 44(4):663–676, 2011.

(2) G. Lee and C. Scott. EM Algorithms for Multivariate Gaussian Mixture Models with Truncated and Censored Data. In review for *Computational Statistics and Data Analysis*.

(3) G. Lee, L. Stoolman, and C. Scott. Transfer Learning for Auto-gating of Flow Cytometry Data. To appear in *Journal of Machine Learning Research: Workshop and Conference Proceedings*, vol. 7, 2011.

(4) G. Blanchard, G. Lee, and C. Scott. Generalizing from Several Related Classification Tasks to a New Unlabeled Sample. To appear in *Advances in Neural Information Processing Systems (NIPS)*, 2011.

In addition to the above publications, I have produced the following works outside the topic of flow cytometry:

(5) G. Lee and C. Scott. The One Class Support Vector Machine Solution Path. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP 2007), 2:II-521–II-524, April 2007.

(6) G. Lee and C. Scott. Nested support vector machines. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP 2008), pages 1985-1988, April 2008.

(7) G. Lee and C. Scott. Nested support vector machines. In *IEEE Transactions on Signal Processing*, 58(3):1648–1660, 2010.

(8) G. Blanchard, G. Lee, and C. Scott. Semi-supervised novelty detection. In *Journal of Machine Learning Research*, 11:2973–3009, Nov. 2010.

CHAPTER 2

Statistical File Matching of Flow Cytometry Data

2.1 Introduction

Flow cytometry is a technique for quantitative cell analysis [63]. It provides simultaneous measurements of multiple characteristics of individual cells. Typically, a large number of cells are analyzed in a short period of time – up to thousands of cells per second. Since its development in the late 1960s, flow cytometry has become an essential tool in various biological and medical laboratories. Major applications of flow cytometry include hematological immunophenotyping and diagnosis of diseases such as acute leukemias, chronic lymphoproliferative disorders and malignant lymphomas [9].

Flow cytometry data has traditionally been analyzed by visual inspection of one-dimensional histograms or two-dimensional scatter plots. Clinicians will visually inspect a sequence of scatter plots based on different pairwise marker combinations and perform gating, the manual selection of marker thresholds, to eliminate certain subpopulations of cells. They identify various pathologies based on the shape of cell subpopulations in these scatter plots. In addition to traditional inspection-based analysis, there has been recent work, reviewed below, on automatic cell gating or classification of pathologies based on multidimensional analysis of cytometry data.

Unfortunately, flow cytometry analysis is limited by the number of markers that can be simultaneously measured. In clinical settings, this number is typically five to seven, while the number of markers of interest may be much larger. To overcome this limitation, it is common in practice to perform multiple assays based on different and overlapping combinations of markers. However, many marker combinations are never observed, which complicates scatter plot-based analysis, especially in retrospective studies. In addition, automated multidimensional analysis is not feasible because all cell measurements have missing values.

To address these issues, we present a statistical method for file matching, which imputes higher dimensional flow cytometry data from multiple lower dimensional data files. While Pedreira et al. [51] proposed a simple approach based on Nearest Neighbor (NN) imputation, this method is prone to induce spurious clusters, as we demonstrate below. Our method can improve the file matching of flow cytometry and is less likely to generate false clusters. The result is a full dataset, where arbitrary pairs can be viewed together, and multidimensional methods can be applied.

In the following, we explain the principles of flow cytometry and introduce the file matching problem in the context of flow cytometry data. We then present a file matching approach which imputes a cell's missing marker values with the values of the nearest neighbor among cells of the same type. To implement this approach, we develop a method for clustering with missing data. We model flow cytometry data with a latent variable Gaussian mixture model, where each Gaussian component corresponds to a cell type, and develop an expectation-maximization (EM) algorithm to fit the model. We also describe ways to incorporate domain knowledge into the initialization of the EM algorithm. We compare our method with nearest neighbor imputation on real flow cytometry data and show that our method offers improved performance. Our MATLAB implementation is available online at `http://www.eecs.umich.edu/~cscott/code/cluster_`

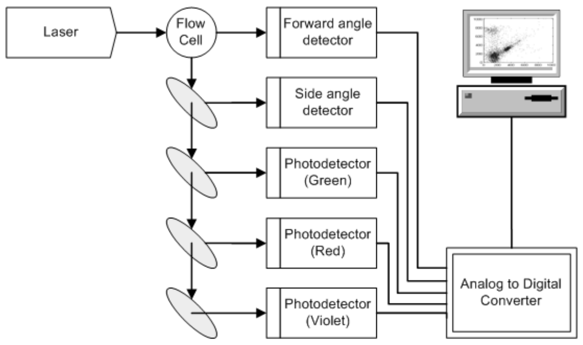


Fig. 2.1: A flow cytometer system. As a stream of cells passes through a laser beam, photo-detectors detect forward angle light scatter, side angle light scatter and light emissions from fluorochromes. Then the digitized signals are analyzed in a computer.

`nn.zip`.

2.2 Background

In this section, we explain the principles of flow cytometry. We also define the statistical file matching problem in the context of flow cytometry data and motivate the need for an improved solution.

2.2.1 Flow cytometry

In flow cytometry analysis for hematological immunophenotyping, a cell suspension is first prepared from peripheral blood, bone marrow or lymph node. The suspension of cells is then mixed with a solution of fluorochrome-labeled antibodies. Typically, each antibody is labeled with a different fluorochrome. As the stream of suspended cells passes through a focused laser beam, they either scatter or absorb the light. If the labeled antibodies are attached to proteins of a cell, the associated fluorochromes absorb the laser and emit light with a corresponding wavelength (color). Then a set of photo-detectors in the line of the light and perpendicular to the light capture the scattered and emitted light. The signals from the detectors are digitized and stored in a computer system. Forward scatter (FS) and side scatter (SS) signals as well as the various fluorescence signals are collected for each cell (see Fig. 2.1).

For example, in a flow cytometer capable of measuring five attributes, the measurements of each cell can be represented with a 5-dimensional vector $\mathbf{x} = (x^{(1)}, \cdots, x^{(5)})$ where $x^{(1)}$ is FS, $x^{(2)}$ is SS and $x^{(3)}, \cdots, x^{(5)}$ are the fluorescent markers. We use "marker" to refer to both the biological entities and the corresponding measured attributes. Then the measurements of N cells are represented by vectors $\mathbf{x}_1, \cdots, \mathbf{x}_N$ and form a $N \times 5$ matrix.

The detected signals provide information about the physical and chemical properties of each cell analyzed. FS is related to the relative size of the cell, and SS is related to its internal granularity or complexity. The fluorescence signals reflect the abundance of expressed antigens on the cell surface. These various attributes are used for identification and quantification of cell subpopulations. FS and SS are always measured, while the marker combination is a part of the experimental design.

Flow cytometry data is usually analyzed using a sequence of one dimensional histograms and two or three dimensional scatter plots by choosing a subset of one, two or three markers. The analysis typically involves manually selecting and excluding cell subpopulations, a process called "gating", by thresholding and drawing boundaries on the scatter plots. Clinicians routinely diagnose by visualizing the scatter plots.

Recently, some attempts have been made to analyze directly in high dimensional spaces by mathematically modeling flow cytometry data. In [8, 15], a mixture of Gaussian distributions is used to model cell populations, while a mixture of t-distributions with a Box-Cox transformation is used in [44]. A mixture of skew t-distributions is studied in [53]. The knowledge of experts is sometimes incorporated as prior information [38]. Instead of using finite mixture models, some recent approaches proposed information preserving dimension reduction to analyze high dimensional flow cytometry data [12, 13].

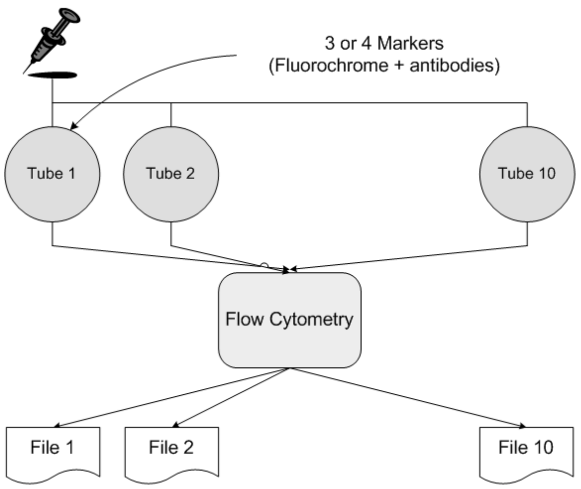


Fig. 2.2: Flow cytometry analysis on a large number of antibody reagents within a limited capacity of a flow cytometer. A sample from a patient is separated into multiple tubes with which different combinations of fluorochrome-labeled antibodies are stained. Each output file contains at least two variables, FS and SS, in common as well as some variables that are specific to the file.

However, standard techniques for multi-dimensional flow cytometry analysis are not yet established.

2.2.2 Statistical file matching

The number of markers used for selection and analysis of cells is constrained by the number of measurable fluorochrome channels (colors) in a given cytometer, which in turn is a function of the optical physics of the laser light source(s) and the excitation and emission spectra of the individual fluorochromes used to label antibodies to targeted surface marker antigens. Recent innovations have enabled measuring near 20 cellular attributes, through the use of multiple lasers of varying energy, multiple fluorochrome combinations, and complex color compensation algorithms. However, instruments deployed in clinical laboratories still only measure 5-7 attributes simultaneously [52].

There may be times in which it would be useful to characterize cell populations using more colors than can be simultaneously measured on a given cytometry platform. For example, some lymph node biopsy samples may be involved partially by lymphoma, in a background of hyperplasia of lymphoid follicles within the lymph node. In such cases, it can be useful to exclude the physiologic follicular lymphocyte subset based on a known array of marker patterns (for example, CD10 expression, brighter CD20 expression than non-germinal center B-cells, and CD38 expression) and evaluate the non-follicular lymphocyte fraction for markers known to be useful in the diagnosis of non-Hodgkin lymphomas (for example, CD5, CD19, CD23, kappa immunoglobulin light chain, and lambda immunoglobulin light chain). Unless an 8-color (10 channel) flow cytometer is available, this analysis cannot be done seamlessly. In such case, the markers must be inferred indirectly, potentially resulting in dilution of the neoplastic lymphoma clone by normal background lymphocytes. Likewise, recent approaches to the analysis of flow cytometry data are built around the treatment of datasets as individual high-dimensional distributions or shapes, again limited only by the number of colors available in a given flow cytometry platform. Given the considerable expense of acquiring cytometry platforms capable of deriving high-dimensionality datasets, the ability to virtually combine multiple lower-dimensional datasets into a single high-dimensional dataset could provide considerable advantage in these situations.

When it is not possible to simultaneously measure all markers of interest, it is common to divide a sample into several “tubes” and stain each tube separately with a different set of markers (see Fig. 2.2) [56]. For example, consider an experiment with two tubes: Tube 1 containing 5000 cells is stained with CD45, CD5 and CD7, and Tube 2 containing 7000 cells is stained with CD45, CD10 and CD19. File 1 and File 2 record the FS, SS and marker measurements in the format of 5000×5 and 7000×5 matrices.

In the sequel, we present a method that combines two or more tubes and generates flow cytometry data in which all the markers of interest are available for the union of

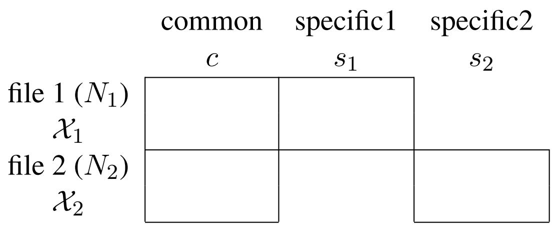


Fig. 2.3: Data structure of two incomplete data files. The two files have some overlapping variables c and some variables s_1 and s_2 that are never jointly observed. File matching combines the two files by completing the missing blocks of variables.

cells. Thus, we obtain a single higher dimensional dataset beyond the current limits of the instrumentation. Then pairs of markers that are not measured together can still be visualized through scatter plots, and methods of multidimensional analysis may potentially be applied to the full dataset.

This technique, called file matching, merges two or more datasets that have some commonly observed variables as well as some variables unique to each dataset. We introduce some notations to generalize the above example. In Fig. 2.3, each unit (cell) $\mathbf{x}_n$ is a row vector in $\mathbb{R}^d$ and belongs to one of the data files (tubes) $\mathcal{X}_1$ or $\mathcal{X}_2$, where each file contains N_1 and N_2 units, respectively. While variables c are commonly observed for all units, variables s_2 are missing in $\mathcal{X}_1$ and s_1 are missing in $\mathcal{X}_2$, where s_1, s_2 and c indicate specific and common variable sets. If we denote the observed and missing components of a unit $\mathbf{x}_n$ with o_n and m_n, then $o_n = c \cup s_1$ and $m_n = s_2$ for $\mathbf{x}_n \in \mathcal{X}_1$ and $o_n = c \cup s_2$ and $m_n = s_1$ for $\mathbf{x}_n \in \mathcal{X}_2$.

Continuing the previous example, suppose that the attribute measurements are arranged as in Fig. 2.3 in the order of FS, SS, CD45, CD5, CD7, CD10 and CD19. Then each individual cell is seen as a row vector in $\mathbb{R}^7$ with two missing variables. Thus, $\mathcal{X}_1$ is a matrix with $N_1 = 5000$ rows and $\mathcal{X}_2$ is a matrix with $N_2 = 7000$ rows, and the common and specific attribute sets are $c = \{1, 2, 3\}$, $s_1 = \{4, 5\}$ and $s_2 = \{6, 7\}$.

A file matching algorithm imputes the blocks of missing variables. Among imputation methods, conditional mean or regression imputations are most common. As shown in Fig. 2.4, however, these imputation algorithms tend to shrink the variance of the data. Thus, these approaches are inappropriate in flow cytometry where the shape of cell subpopulations is important in clinical analysis. More discussions on missing data analysis and file matching can be found in [42] and [54].

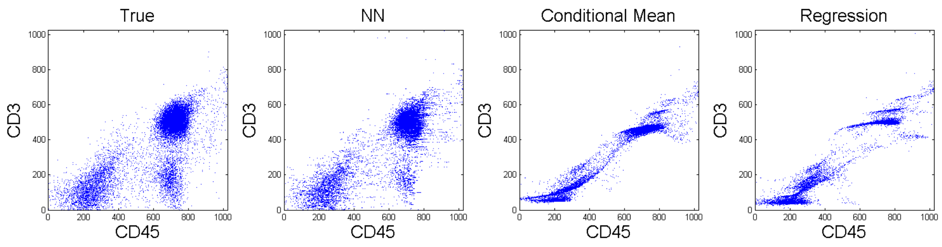


Fig. 2.4: Examples of imputation methods: NN, conditional mean and regression. The NN method relatively well preserves the distribution of imputed data, while other imputation methods such as conditional mean and regression significantly reduce the variability of data.

A recent file matching technique in flow cytometry was developed by Pedreira et al. [51]. They proposed to use Nearest Neighbor (NN) imputation to match flow cytometry data files. In their approach, the missing variables of a unit, called the recipient, are imputed with the observed variables from a unit in the other file, called the donor, that is most similar. If $\mathbf{x}_i$ is a unit in $\mathcal{X}_1$, the missing variables of $\mathbf{x}_i$ are set as follows:

$$\mathbf{x}_i^{m_i} = \mathbf{x}_j^{*\,m_i} \text{ where } \mathbf{x}_j^* = \arg\min_{\mathbf{x}_j \in \mathcal{X}_2} \|\mathbf{x}_i^c - \mathbf{x}_j^c\|_2.$$

Here $\mathbf{x}_i^{m_i} = (x_i^{(p)}, p \in m_i)$ and $\mathbf{x}_i^c = (x_i^{(p)}, p \in c)$ denote the row vectors of missing and common variables of $\mathbf{x}_i$, respectively. Note that the similarity is measured by the distance in the projected space of jointly observed variables. This algorithm is advantageous over other imputation algorithms using conditional mean or regression, as displayed in Fig. 2.4. It generally preserves the distribution of cells, while the other methods cause the variance structure to shrink toward zero.

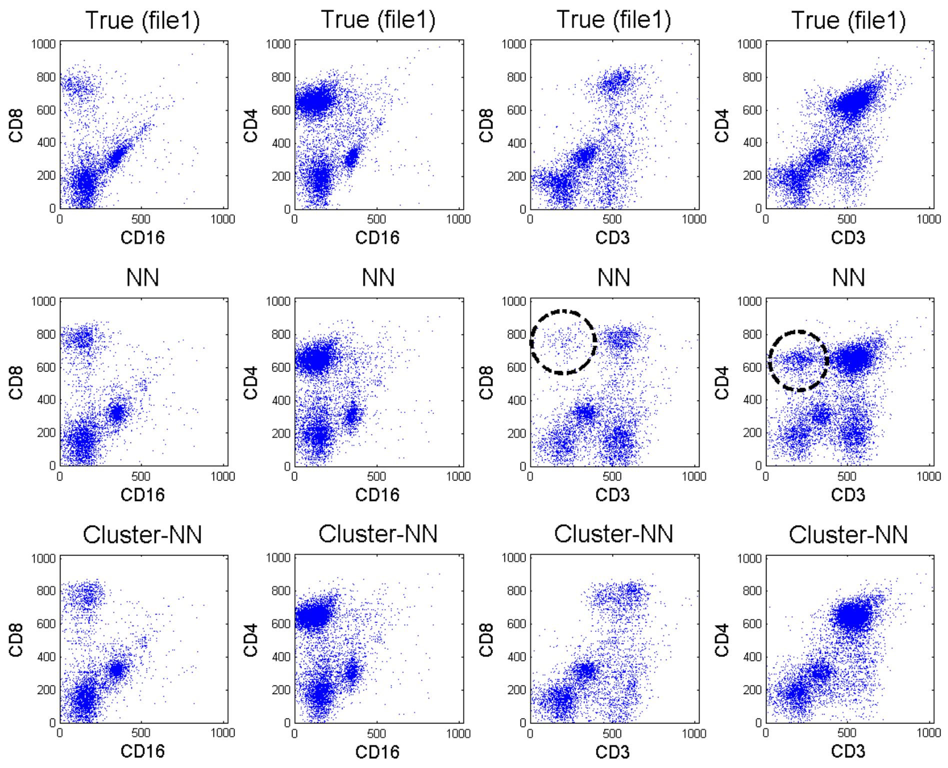


Fig. 2.5: Comparison of results for two imputation methods to the ground truth cell distribution. Figures show scatter plots on pairs of markers that are not jointly observed. The middle row and the bottom row show the imputation results from the NN and the proposed Cluster-NN method, respectively. The results from the NN method show spurious clusters in the right two panels. The false clusters are indicated by dotted circles in the CD3 vs. CD8 and CD3 vs. CD4 scatter plots. On the other hand, the results from our proposed approach better resemble the true distribution on the top row.

However, the NN method sometimes introduces spurious clusters into the imputation results and fails to replicate the true distribution of cell populations. Fig. 2.5 shows an example of false clusters from the NN imputation algorithm (for the detailed experiment setup, see Section 2.4). We present a toy example to illustrate how NN imputation can fail and motivate our approach.

2.2.3 Motivating toy example

Fig. 2.6 shows a dataset in $\mathbb{R}^3$. In each file, only two of the three features are observed: c and s_1 in file 1 and c and s_2 in file 2. Each data point belongs to one of two clusters, but its label is unavailable. This example is not intended to simulate flow cytometry data, but rather to illustrate one way in which NN imputation can fail, and how our approach can overcome this limitation.

When imputing feature s_1 of units in file 2, the NN algorithm produces four clusters whereas there should be two, as shown in Fig. 2.6 (d). This is because the NN method uses only one feature and fails to leverage the information about the joint distribution of variables that are not observed together. However, if we can infer the cluster membership of data points, the NN imputation can be applied within the same cluster. Hence, we seek a donor from subgroup (1) for the data points in (3) and likewise we seek a donor from (2) for the points in (4) in the example. Then the file matching result greatly improves and better replicates the true distribution as in Fig. 2.6 (e).

In this example, as in real flow cytometry data, there is no way to infer cluster membership from the data alone, and incorrect labeling can lead to poor results (Fig. 2.6 (f)). Fortunately, in flow cytometry we can incorporate domain knowledge to achieve an accurate clustering.

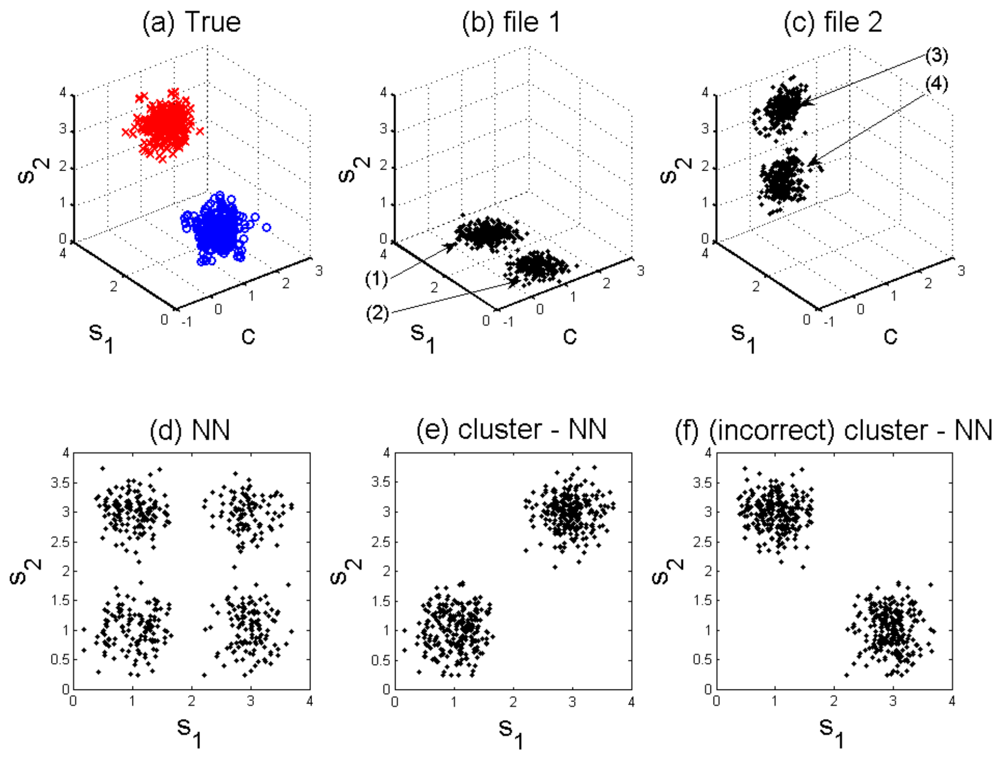


Fig. 2.6: Toy example of file matching. Two files (b) and (c) provide partial information of data points (a) in $\mathbb{R}^3$. The variable c is observed in both files while s_1 and s_2 are specific to each file. The NN method created false clusters in the s_1 vs. s_2 scatter plot in (d). On the other hand, the proposed Cluster-NN method, which applies NN within the same cluster, successfully replicated the true distribution. If the clusters are incorrectly paired, however, the Cluster-NN approach can fail, as in (f).

Algorithm 2.1 Cluster-NN algorithm

```
Input: Two files $\mathcal{X}_1$ and $\mathcal{X}_2$ to be matched
1: 1. Jointly cluster the units in $\mathcal{X}_1$ and $\mathcal{X}_2$.
2: 2. Perform Nearest Neighbor imputation within the same cluster.
Output: Statistically matched complete files $\widehat{\mathcal{X}_1}$ and $\widehat{\mathcal{X}_2}$
```

2.3 Methods

2.3.1 Cluster-based imputation of missing variables

We first focus on the case of matching two files. The case of more than two files is discussed in Section 2.5. For the present section, we assume that there is a single underlying distribution with K clusters, and each $\mathbf{x} \in \mathcal{X}_1$ and each $\mathbf{x} \in \mathcal{X}_2$ is assigned to one of these clusters. Let $\mathcal{X}_1^k$ and $\mathcal{X}_2^k$ denote the cells in $\mathcal{X}_1$ and $\mathcal{X}_2$ from the kth cluster, respectively.

Suppose that the data is configured as in Fig. 2.3. To impute the missing variables of a recipient unit in $\mathcal{X}_1$, we locate a donor among the data points in $\mathcal{X}_2$ that have the same cluster label as the recipient. When imputing incomplete units in $\mathcal{X}_2$, the roles change. The similarity between two units is evaluated on the projected space of jointly observed variables, while constraining both units to belong to the same cluster. Then we impute the missing variables of the recipient by patching the corresponding variables from the donor. More specifically, for $\mathbf{x}_i \in \mathcal{X}_1^k$, we impute the missing variables by

$$\mathbf{x}_i^{m_i} = \mathbf{x}_j^{*m_i} \text{ where } \mathbf{x}_j^* = \arg\min_{\mathbf{x}_j \in \mathcal{X}_2^k} \|\mathbf{x}_i^c - \mathbf{x}_j^c\|_2.$$

Algorithm 2.1 describes the proposed Cluster-NN imputation algorithm.

In social applications such as survey completion, file matching is often performed on the same class such as gender, age, or county of residence [54]. Unlike our algorithm, however, the information for labeling each unit is available in those applications and the class inference step is unnecessary.

2.3.2 Clustering with missing data

To implement the above approach, it is necessary to cluster the flow cytometry data. Thus, we concatenate two input files $\mathcal{X}_1$ and $\mathcal{X}_2$ into a single dataset as in Fig. 2.3. We model the data with a mixture model with each component of the mixture corresponding to a cluster. We emphasize that we are jointly clustering $\mathcal{X}_1$ and $\mathcal{X}_2$, not each file separately. Thus, each $\mathbf{x}$ in the merged dataset is assigned to one of the K mixture model components.

In a mixture model framework, the probability density function of a d-dimensional data vector $\mathbf{x}$ takes the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, p_k(\mathbf{x})$$

where π_k are mixing weights of K components and p_k are component density functions. In flow cytometry, mixture models are widely-used to model cell populations. Among mixture models, Gaussian mixture models are common [8, 15, 38], while distributions with more parameters, such as t-distributions, skew normal or skew t-distributions, have been recently proposed [44, 53]. While non-Gaussian models might provide a better fit, there is a trade-off between bias and variance. More complicated models tend to be more challenging to fit. Furthermore, even with an imperfect data model, we may still achieve an improved file matching.

Clustering amounts to fitting the parameters of the mixture model to the data points in $\mathcal{X}_1$ and $\mathcal{X}_2$. Given the model, a data point $\mathbf{x}$ is assigned to cluster k for which the posterior probability is maximized. Here we explain the mixture model that we used to model the cell populations (Section 2.3.2) and present an EM algorithm for inferring the model parameters, which determine the cluster membership of each data point (Section 2.3.2).

Mixture of PPCA

Fitting multidimensional mixture models require estimating a large number of parameters, and obtaining reliable estimates becomes difficult when the number of components or the dimension of the data increase. Here we adopt a probabilistic principal component analysis (PPCA) mixture model as a way to concisely model cell populations.

PPCA was proposed by [71] as a probabilistic interpretation of PCA. While conventional PCA lacks a probabilistic formulation, PPCA specifies a generative model, in which a data vector is linearly related to a latent variable. The latent variable space is generally lower dimensional than the ambient variable space, so the latent variable provides an economical representation of the data. Our motivations for using PPCA over a full Gaussian mixture model are that the parameters can be fit more efficiently (as demonstrated in Section 2.4), and in higher dimensional settings, a full Gaussian mixture model may have too many parameters to be accurately fit.

The PPCA model is built by specifying a distribution of a data vector $\mathbf{x} \in \mathbb{R}^d$ conditional on a latent variable $\mathbf{t} \in \mathbb{R}^q$, $p(\mathbf{x}|\mathbf{t}) = \mathcal{N}(\mathbf{W}\mathbf{t} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$ where $\boldsymbol{\mu}$ is a d-dimensional vector and $\mathbf{W}$ is a $d \times q$ linear transform matrix. Assuming the latent variable $\mathbf{t}$ is normally-distributed, $p(\mathbf{t}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, the marginal distribution of $\mathbf{x}$ becomes Gaussian $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$. Then the posterior distribution can be shown to be Gaussian as well: $p(\mathbf{t}|\mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1})$ where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$ is a $q \times q$ matrix.

The PPCA mixture model is a combination of multiple PPCA components. This model offers a way of controlling the number of parameters to be estimated without completely sacrificing the model flexibility. In the full Gaussian mixture model, each Gaussian component has $d(d+1)/2$ covariance parameters if a full covariance matrix is used. The number of parameters can be reduced by constraining the covariance matrix to be isotropic or diagonal. However, these are too restrictive for cell populations since the correlation structure between variables cannot be captured. On the other hand, the PPCA mixture model lies between those two extremes and allows control of the number of parameters through specification of q, the dimension of the latent variable.

A PPCA mixture can be viewed as a Gaussian mixture with structured covariances. In Gaussian mixtures, various approaches constraining covariance structures have been proposed [26], where each cluster is required to share parameters to have the same orientation, volume or shape. However, in the PPCA model, the geometry of each cluster is allowed to vary between clusters, and the cluster parameters for different clusters are not constrained to be related to one another. Therefore, the PPCA mixture model is preferable in flow cytometry where cell populations typically have different geometric characteristics.

In a mixture of PPCA model, each PPCA component explains local data structure or a cell subpopulation, and the collection of component parameters $\theta_k = \{\pi_k, \boldsymbol{\mu}_k, \mathbf{W}_k, \sigma_k^2\}$, $k = 1, \cdots, K$, defines the model. An EM algorithm can learn the model by iteratively estimating these parameters. More details on the PPCA mixture and the EM algorithm for data without missing values are explained in [70].

Missing data EM algorithm

The concatenated dataset of $\mathcal{X}_1$ and $\mathcal{X}_2$ contains only partial observations of $N = N_1 + N_2$ units. Hence, we cannot directly apply the EM algorithm for a PPCA mixture to infer the model parameters. In the present section, we devise a novel EM algorithm for the missing data.

Even though our file matching problem has a particular pattern of missing variables, we develop a more general algorithm that allows for an arbitrary pattern of missing variables. Our development assumes values are "missing at random," meaning that whether a variable is missing or not is independent of its value [42]. We note that [31] presented an EM algorithm for a Gaussian mixture with missing data, and [70] presented EM algorithms for a PPCA mixture when data is completely observed. Therefore, our algorithm may be viewed as an extension of the algorithm of [31] to PPCA mixtures, or the algorithm of [70] to data with missing values.

Denoting the observed and missing variables by o_n and m_n, each data point can be divided as $\mathbf{x}_n = \begin{bmatrix} \mathbf{x}_n^{o_n} \\ \mathbf{x}_n^{m_n} \end{bmatrix}$. Recall that, in the file matching problem, o_n indexes the union of common variables and the observed specific variables, and m_n indexes the unobserved specific variables so that $x_n^{(i)}$, $i \in o_n$, are observed variables and $x_n^{(i)}$, $i \in m_n$, are missing variables. This is only for notational convenience and does not imply that the vector $\mathbf{x}_n$ is re-arranged to this form.

Thus, we are given a set of partial observations $\{\mathbf{x}_1^{o_1}, \cdots, \mathbf{x}_N^{o_N}\}$. To invoke the EM machinery, we introduce indicator variables $\mathbf{z}_n$. One and only one entry of $\mathbf{z}_n$ is nonzero and $z_{nk} = 1$ indicates that the kth component is responsible for generating $\mathbf{x}_n$. We also include the missing variables $\mathbf{x}_n^{m_n}$ and the set of latent variables $\mathbf{t}_{nk}$ for each component to form the complete data $(\mathbf{x}_n^{o_n}, \mathbf{x}_n^{m_n}, \mathbf{t}_{nk}, \mathbf{z}_n)$ for $n = 1, \cdots, N$ and $k = 1, \cdots, K$.

We derive an EM algorithm for the PPCA mixture model with missing data. The key difference from the EM algorithm for completely observed data is that the conditional expectation is taken with respect to $\mathbf{x}^o$ as opposed to $\mathbf{x}$ in the expectation step.

To develop an EM algorithm, we employ and extend the two-stage procedure as described in [70]. In the first stage of the algorithm, the component weights π_k and the component center $\boldsymbol{\mu}_k$ are updated:

$$\widehat{\pi}_k = \frac{1}{N} \sum_n \langle z_{nk} \rangle, \tag{2.1}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_n \langle z_{nk} \rangle \begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix}}{\sum_n \langle z_{nk} \rangle} \tag{2.2}$$

where $\langle z_{nk} \rangle = P(z_{nk} = 1 | \mathbf{x}_n^{o_n})$ is the responsibility of mixture component k for generating the unit $\mathbf{x}_n$ and $\langle \mathbf{x}_n^{m_n} \rangle = \mathbb{E}[\mathbf{x}_n^{m_n} | z_{nk} = 1, \mathbf{x}_n^{o_n}]$ is the conditional expectation. Note that we are not assuming the vectors in the square bracket are arranged to have this pattern. This notation can be replaced by the true variable ordering.

In the second stage, we update $\mathbf{W}_k$ and σ_k^2:

$$\widehat{\mathbf{W}}_k = \mathbf{S}_k \mathbf{W}_k (\sigma_k^2 \mathbf{I} + \mathbf{M}_k^{-1} \mathbf{W}_k^T \mathbf{S}_k \mathbf{W}_k)^{-1}, \tag{2.3}$$

$$\widehat{\sigma}_k^2 = \frac{1}{d} \mathrm{tr} \left(\mathbf{S}_k - \mathbf{S}_k \mathbf{W}_k \mathbf{M}_k^{-1} \widehat{\mathbf{W}}_k^T \right) \tag{2.4}$$

from local covariance matrix $\mathbf{S}_k$:

$$\mathbf{S}_k = \frac{1}{N \widehat{\pi}_k} \sum_n \langle z_{nk} \rangle \langle \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right) \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right)^T \rangle.$$

The new parameters are denoted by $\widehat{\pi}_k$, $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\mathbf{W}}_k$ and $\widehat{\sigma}_k^2$. These update rules boil down to the update rules for completely observed data when there are no missing variables. We derive the EM algorithm in detail in Section 2.A.

After model parameters are estimated, the observations are divided into groups according to their posterior probabilities:

$$\underset{k=1,\cdots K}{\arg\max}\, p\left(z_{nk} = 1 | \mathbf{x}_n^{o_n}\right),$$

so each unit (cell) is classified into one of K cell subpopulations. Note that this posterior probability is computed in the E-step. This gives the desired clustering.

Cell Type	CD markers
granulocytes	CD45+, CD15+
monocytes	CD45+, CD14+
helper T cells	CD45+, CD3+
cytotoxic T cells	CD45+, CD3+, CD8+
B cells	CD45+, CD19+ or CD45+, CD20+
Natural Killer cells	CD16+, CD56+, CD3-

Table 2.1: Types of human white blood cells. The T cells, B cells and NK cells are called lymphocytes. Each cell type is characterized by a set of expressed cluster of differentiation (CD) markers. The CD markers are commonly used to identify cell surface molecules on white blood cells. The '$+/-$' signs indicate whether a certain cell type has the corresponding antigens on the cell surface.

Domain knowledge and initialization of EM algorithm

Because of the missing data, fitting a PPCA mixture model is ill-posed, in the sense that several local maxima of the likelihood may explain the data equally well. For example, in the toy example in Section 2.2.3, there is no way to know the correct cluster inference based solely on the data. However, we can leverage domain knowledge to select the number of components and initialize model parameters.

In flow cytometry, from the design of fluorochrome marker combinations and knowledge about the blood sample composition, we can anticipate certain properties of cell subpopulations. For example, Table 2.1 summarizes white blood cell types and their characteristic cluster of differentiation (CD) marker expressions. The six cell types suggests choosing $K = 6$ when analyzing white blood cells.

The CD markers indicated are commonly used in flow cytometry to identify cell surface molecules on leukocytes (white blood cells) [75]. However, this information is qualitative and needs to be quantified. Furthermore, the appropriate quantification depends on the patient and flow cytometry system.

To achieve this, we use one-dimensional histograms. In a histogram, two large peaks are generally expected depending on the expression level of the corresponding CD marker.

If a cell subpopulation expresses a CD marker, denoted by '$+$', then it forms a peak on the right side of the histogram. On the other hand, if a cell subpopulation does not express the marker, denoted by '$-$', then a peak can be found on the left side of the histogram. We use the locations of the peaks to quantify the expression levels.

These quantified values can be combined with the CD marker expression levels of each cell type to specify the initial cluster centers. Thus, each element of $\boldsymbol{\mu}_k$ of a certain cell type is initialized by either the positive quantity or the negative quantity from the histogram. In our implementation, these are set manually by visually inspecting the histograms. Then we initialize the mixture model parameters $\{\pi_k, \boldsymbol{\mu}_k, \mathbf{W}_k, \sigma_k^2\}$ as described in Algorithm 2.2.

Cell populations are partitioned into K groups by the distance to each component center. The component weight π_k is proportional to the size of each partition. From the covariance matrix estimate $\mathbf{C}_k$, parameters $\mathbf{W}_k$ and σ_k^2 are initialized by taking the eigen-decomposition.

Algorithm 2.2 Parameter initialization of an EM algorithm for missing data.

```
Input: $\mathcal{X}_1$, $\mathcal{X}_2$ data files; $K$ the number of components; $q$ the dimension of the latent
  variable space; $\boldsymbol{\mu}_k$ the initial component means.
  for $k = 1$ to $K$ do
    1. Using distance $\|\mathbf{x}_n^{o_n} - \boldsymbol{\mu}_k^{o_n}\|_2$, find the set of data points $\mathcal{X}^k$ whose nearest compo-
    nent mean is $\boldsymbol{\mu}_k$
    2. Initialize observable submatrices of $\mathbf{C}_k$ with sample covariances of data in $\mathcal{X}^k$,
    and the remaining entries with random draws from a standard normal distribution.
    3. Make $\mathbf{C}_k$ positive definite by replacing negative eigenvalues with a tenth of the
    smallest positive eigenvalue.
    4. Set $\pi_k = |\mathcal{X}^k|/(N_1 + N_2)$
    5. Set $\mathbf{W}_k$ with the $q$ principal eigenvectors of $\mathbf{C}_k$
    6. Set $\sigma_k^2$ with the average of remaining eigenvalues of $\mathbf{C}_k$
  end for
Output: $\{\pi_k, \boldsymbol{\mu}_k, \mathbf{W}_k, \sigma_k^2\}$ for $k = 1, \cdots, K$
```

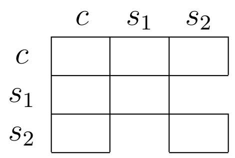


Fig. 2.7: Structure of covariance matrix $\mathbf{C}$. The sub-matrices $\mathbf{C}_k^{s_1,s_2}$ and $\mathbf{C}_k^{s_2,s_1}$ cannot be estimated from a sample covariance matrix because these variables are never jointly observed.

An important issue in file matching arises from the covariance matrix. When data is completely observed, a common way of initializing a covariance matrix is using a sample covariance matrix. In the case of file matching, however, it cannot be evaluated since some sets of variables are never jointly observed (see Fig. 2.7). Hence, we build $\mathbf{C}_k$ from variable to variable with sample covariances, whenever possible. For example, we can set $\mathbf{C}_k^{c,s_1}$ with the sample covariance of data points in $\mathcal{X}_1$ where variables c and s_1 are available. On the other hand, the submatrix $\mathbf{C}_k^{s_1,s_2}$ cannot be built from observations. In our implementation, we set the submatrix $\mathbf{C}_k^{s_1,s_2}$ randomly from a standard normal distribution. However, the resulting matrix may not be positive definite. Thus, we made $\mathbf{C}_k$ positive definite by replacing negative eigenvalues with a tenth of the smallest positive eigenvalue. Once a covariance matrix $\mathbf{C}_k$ is obtained, we can initialize $\mathbf{W}_k$ and σ_k^2 by taking the eigen-decomposition of $\mathbf{C}_k$.

2.4 Results

We apply the proposed file matching technique to real flow cytometry datasets and present experimental results. Three flow cytometry datasets were prepared from lymph node samples of three patients. These datasets were provided by the Department of Pathology at the University of Michigan.

We consider two experimental settings. In the first experiment (Section 2.4.1), we artificially create two incomplete data files from a single tube and compare the imputed results

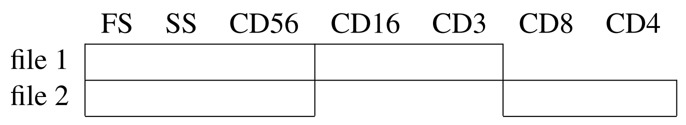


Fig. 2.8: File structure used in the single tube experiments in Section 2.4.1. FS, SS and CD56 are common in both files, and a pair of CD markers are observed in only one of the files. The blank blocks correspond to the unobserved variables. The blocks in file 1 are matrices with N_1 rows and the blocks in file 2 are matrices with N_2 rows.

to the original true dataset. In the second experiments (Section 2.4.2), we investigate multiple tubes where each file is derived individually from two different tubes and the imputed results are compared to separate reference data.

2.4.1 Single tube experiments

From each patient sample, a dataset is obtained with seven attributes: FS, SS, CD56, CD16, CD3, CD8 and CD4. Two files are built from this dataset, and two attributes from each file are made hidden to construct hypothetical missing data. Hence, CD16 and CD3 are available only in file 1, and CD8 and CD4 are available only in file 2, while FS, SS and CD56 are commonly available in both files. Fig. 2.8 illustrates the resulting data pattern where the blocks of missing variables are left blank.

For each white blood cell type, its expected marker expressions (CD markers), relative size (FS) and relative granularity (SS) are presented in Table 2.2. The '$+/-$' signs indicate whether a certain type of cells expresses the markers or not. For example, helper T cells express both CD3 and CD4 but not others. As explained in Section 2.3.2, we quantify this qualitative knowledge with the help of one dimensional histograms. Two dominant peaks corresponding to the positive and negative expression levels are picked from each histogram, and their measurement values are set to the expression levels. Fig. 2.9 and Table 2.3 summarize this histogram analysis. When two negative peaks are present as in CD8, the stronger ones are chosen in our implementation. In flow cytometry, it is known

Cell type	FS	SS	CD56	CD16	CD3	CD8	CD4
granulocytes	$+$	$+$	$-$	$+$	$-$	$-$	$-$
monocytes	$+$	$-$	$-$	$+$	$-$	$-$	$-$
helper T cells	$-$	$-$	$-$	$-$	$+$	$-$	$+$
cytotoxic T cells	$-$	$-$	$-$	$-$	$+$	$+$	$-$
B cells	$-$	$-$	$-$	$-$	$-$	$-$	$-$
Natural Killer cells	$-$	$-$	$+$	$+$	$-$	$-$	$-$

Table 2.2: Cell types and their corresponding marker expressions for data in the single tube experiments. '$+$' or '$-$' indicates whether a certain cell type expresses the CD marker or not.

that two types of cells with the same '$-$' marker can cause slightly different measurement levels. However, this difference between '$-$' peaks is often small and less significant compared to the difference between '$+$' and '$-$' peaks. When we tried experiments (not presented) by choosing weaker peaks, we could not observe meaningful changes in the results.

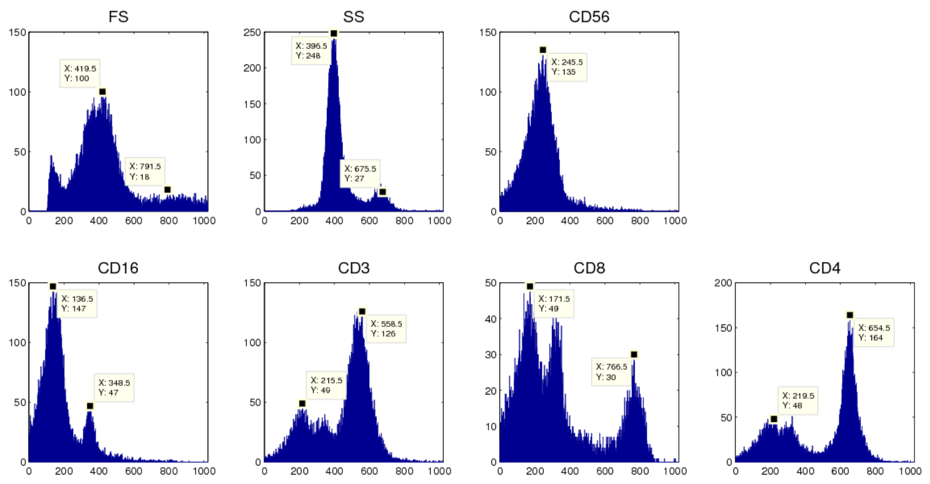


Fig. 2.9: Histogram of each marker in the single tube experiments (Section 2.4.1). The peaks are selected manually and are indicated in each panel.

Following the procedure delineated in Section 2.3, two incomplete data files are completed. A mixture of PPCA is fitted with six components because six cell types are expected on this dataset. The latent variable dimension of each PPCA component is fixed

	FS	SS	CD56	CD16	CD3	CD8	CD4
+	800	680	500	350	550	750	650
−	400	400	240	130	200	170	200

Table 2.3: The positive and negative expression levels are extracted from the histograms in Fig. 2.9. These values are used to initialize the EM algorithm.

to two. The convergence of the missing data EM algorithm is determined when the relative change of log-likelihood value is less than 10^{-10} or the number of iterations reaches 5000. Fig. 2.10 shows the evolution as iteration continues. The likelihood value increases sharply during the dozens of steps in the beginning and then converges.

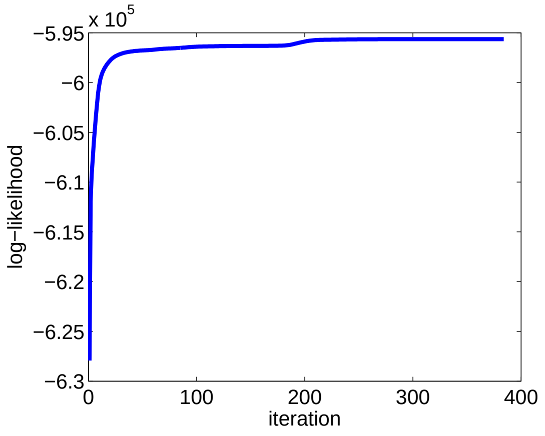


Fig. 2.10: Typical convergence of the proposed missing data EM algorithm.

The synthesized data after file matching is displayed in Fig. 2.5. The figure shows scatter plots of specific variables: CD16, CD3, CD4 and CD8. Note that these marker pairs are not available from any of the two incomplete data files, while other marker pairs are directly obtainable from the observed cells. The imputation results from the NN and the Cluster-NN methods are compared in the figure. For reference, the figure also presents scatter plots of the ground truth dataset. As can be seen, the results from the Cluster-NN better coincide with the true distributions. By contrast, the NN method generates spurious clusters in the CD3-CD8 and CD3-CD4 scatter plots, and the results are far from the true distributions. These false clusters are indicated in Fig. 2.5. We quantify the quality of the

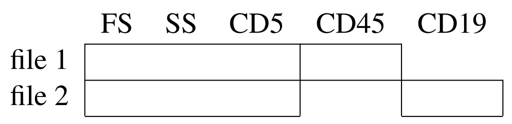


Fig. 2.11: Data pattern used in the multiple tube experiments in Section 2.4.2. Both files contain FS, SS and CD5 commonly, and each file contains one of CD45 and CD19. All marker attributes are available in a separate reference file.

imputed values below in Section 2.4.3.

2.4.2 Multiple tube experiments

In this second experiment, we involve multiple tubes and demonstrate the file matching of flow cytometry data.

Two tubes from each of the three patient samples are stained individually with different marker combinations: CD5/CD45 and CD5/CD19. For comparison with actually measured data, an additional tube is conjugated with markers CD5/CD45/CD19. This additional tube dataset is used only for evaluation of imputation results and is not involved during the file matching. Fig. 2.11 illustrate the pattern of datasets used in the experiments.

As opposed to the previous single tube experiments, the experiments on multiple tubes impose another complication. It is well-known that in flow cytometry, the instrument can drift over time. This technical variation causes the shifts in population positions. To minimize the effects from this variation, data files can be preprocessed with normalization techniques [25, 32] before applying file matching algorithms.

However, the rate of this drift is typically very slow and on a much larger scale than the time for one set of tubes. Furthermore, operators are careful to calibrate each tube (based on the same sample) in the same way to minimize such variation. For these reasons, technical variation within a batch of tubes corresponding to the same patient/sample is much less of an issue in flow cytometry, compared to technical variation between data gathered at different times. Since no noteworthy population shift was found from the

Cell type	FS	SS	CD5	CD45	CD19
granulocytes	+	+	−	+	−
monocytes	+	−	−	+	−
helper T cells	−	−	+	+	−
cytotoxic T cells	−	−	+	+	−
B cells	−	−	−	+	+
Natural Killer cells	−	−	−	+	−

Table 2.4: Types of white blood cells and their corresponding markers expressions for data in the multiple tube experiments..

histogram analysis in Fig. 2.12, we proceeded without any normalization.

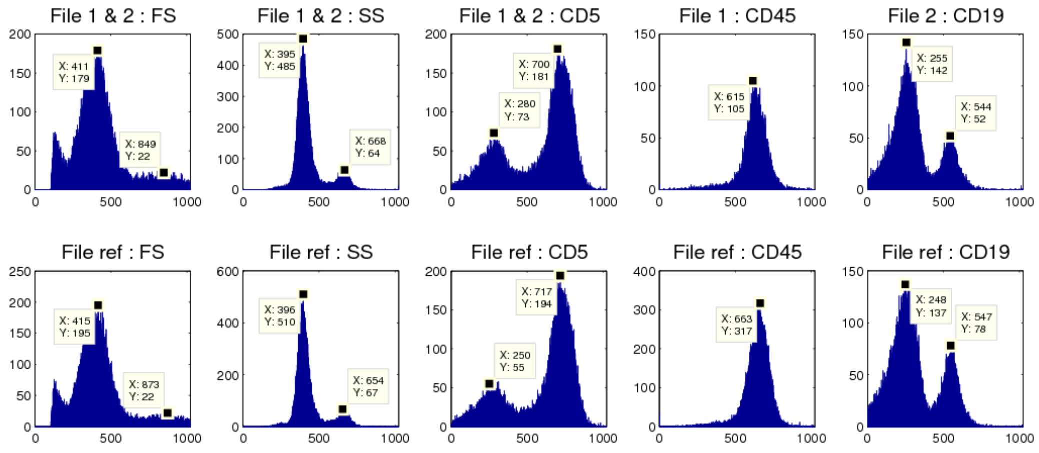


Fig. 2.12: The top row shows histograms from the two incomplete files. Histograms from the reference file are shown in the bottom row. The peaks of each marker are indicated. No noticeable population shift across files was observable.

For datasets in multi-tube experiments, Table 2.4 shows the relative marker expression levels of various types of white blood cells. Their corresponding numerical measurement levels are found from this table and the histograms in Fig. 2.12, and given in Table 2.5. Since all white blood cells express CD45, its negative level is left blank in the table.

	FS	SS	CD5	CD45	CD19
+	850	670	700	615	545
−	410	395	280	–	255

Table 2.5: The positive and negative expression levels are obtained from the histograms in Fig. 2.12. Since all white blood cells express CD45, the negative level is left blank.

Similarly to the above experiments, the two incomplete data files are imputed using the Cluster-NN algorithm as explained in Section 2.3. In this experiment, a PPCA mixture model with five components is fitted to the missing data. We choose five components because the two types of T cells share the same row in Table 2.4. The dimension of the latent variable of each component, q, is set to two.

Fig. 2.13 displays the cell distributions of imputed data files. The presented marker pair CD45-CD19 is not originally available in any of the two files in experiments. The corresponding scatter plot from the separate reference file is also drawn. While the imputed results from the NN method and the Cluster-NN method look similar, a horizontal drift of cells in high CD19 subpopulation can be observed in the NN result. This spread of cells is not present in the reference plot and the Cluster-NN result.

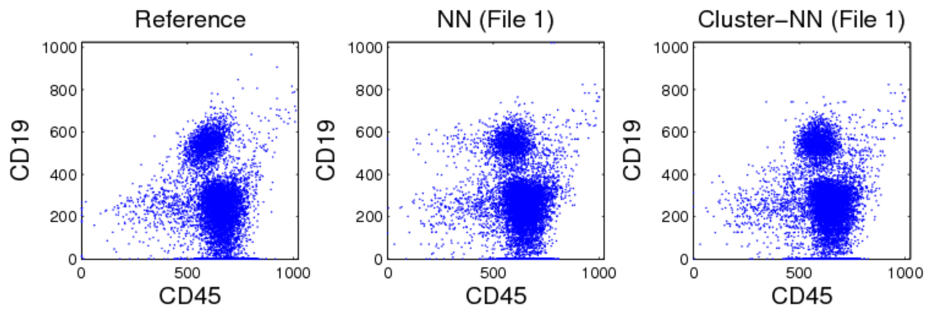


Fig. 2.13: Comparison of two imputation results with the actual measurements in the reference file. The result from the NN method shows a horizontal drift of cells in high CD19 population. This is not observed in the Cluster-NN result and the reference file.

2.4.3 Evaluation method

To quantitatively evaluate the previous results, we use Kullback-Leibler (KL) divergence. The KL divergence between two distribution $f(\mathbf{x})$ and $g(\mathbf{x})$ is defined by

$$KL(g \,\|\, f) = \mathbb{E}_g \left[\log g - \log f \right].$$

Let f be a true distribution responsible for the observations and g be its estimate.

The KL divergence is asymmetric and $KL(f \,\|\, g)$ and $KL(g \,\|\, f)$ have different meanings. We prefer $KL(g \,\|\, f)$ to $KL(f \,\|\, g)$ because the former more heavily penalize the over-estimation of the support of f. This allows us to assess when an imputation method introduces spurious clusters.

For the single tube and the multiple tube experiments, we evaluated the KL divergence of the imputation results. We randomly permuted each dataset ten times, and divided into incomplete data files and evaluation sets. Then we computed the KL divergence for each permutation, and reported their averages and standard errors in Table 2.6. The details of dividing datasets and computing the KL divergence are explained in Section 2.B.

As can be seen, the KL divergences from Cluster-NN are substantially smaller than those from NN in the first set of experiments on a single tube. Therefore, the Cluster-NN yielded a better replication of true distribution. In the second series of experiments, the differences in KL divergence between algorithms were minor. While we could observe the spread of cells in the NN results (see Fig. 2.13), their effect on the KL divergence was sometimes small due to their relatively small number.

2.4.4 Computational considerations

Here we consider computational aspects of the PPCA mixture model and its EM algorithm.

As we described above in Section 2.3.2, through the PPCA mixtures, we can control the number of model parameters without losing the model flexibility. When combining more tubes, this ensures that there is sufficient data for parameter estimation with higher dimensionality. Another advantage of using PPCA mixtures is the execution time of the EM algorithm. Under Windows 7 system equipped with two Intel(R) Xeon(R) 2.27 GHz processors and RAM 12GB, the average convergence time with PPCA mixtures was about

ID	NN (file 1)	Cluster-NN (file 1)	NN (file 2)	Cluster-NN (file 2)
Patient1	2.90 ± 0.05	1.55 ± 0.05	2.66 ± 0.03	1.12 ± 0.04
Patient2	4.54 ± 0.07	1.22 ± 0.03	4.12 ± 0.08	0.92 ± 0.03
Patient3	4.46 ± 0.10	2.40 ± 0.11	4.18 ± 0.11	2.30 ± 0.07

(a) Single tube experiments

ID	NN (file 1)	Cluster-NN (file 1)	NN (file 2)	Cluster-NN (file 2)
Patient1	0.51 ± 0.01	0.46 ± 0.02	0.41 ± 0.01	0.40 ± 0.01
Patient2	0.64 ± 0.01	0.62 ± 0.03	0.80 ± 0.03	0.78 ± 0.04
Patient3	0.88 ± 0.05	0.78 ± 0.07	0.80 ± 0.02	0.65 ± 0.03

(b) Multiple tube experiments

Table 2.6: The KL divergences are computed for ten permutations of each flow cytometry dataset. The averages and standard errors are reported in the table. For both the NN and Cluster-NN algorithm, the file matching results are evaluated. (a) In the single tube experiments, the KL divergences of Cluster-NN are closer to zero than those of NN. Thus, the results from Cluster-NN better replicated the true distribution. (b) In the multiple tube experiments, the Cluster-NN consistently performed better than the NN. However, the differences between two algorithms are small.

23 seconds in the above single tube experiment. On the contrary, it took nearly 200 seconds on average to fit full Gaussian mixtures. That is, fitting a PPCA mixture model took approximately eight times less relative to one based on a full Gaussian mixture model. This computational improvement is highly desirable because demands for high-throughput analysis are sharply increasing in flow cytometry.

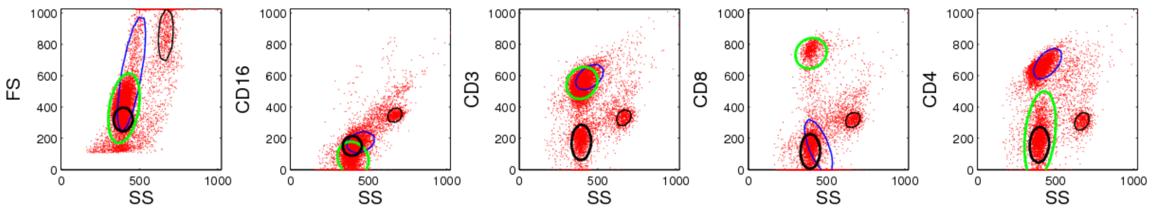


Fig. 2.14: The scatter plots of a dataset used in the single tube experiments (Section 2.4.1) are drawn on several marker pairs. The fitted mixture components are shown as well on each panel. For clarity, four among the six components are displayed.

During the series of experiments, we have chosen the number of mixture components based on the number of cell types. Then mixture models are learned from the partially observed data. Fig. 2.14 illustrates how the clustering behaves on a dataset used in the single tube experiments (Section 2.4.1). Component contours are overlaid on the scatter plots over a few observed marker pairs. Most contours can be successfully identified with important cell subpopulations in the dataset, while there are some cases where we could not find the corresponding cell types.

Although many mixture model-based analysis in flow cytometry rely on criteria such as Akaike information criterion or Bayesian information criterion to select the number of components [15, 44, 53], these approaches assume completely observed data, whereas most of the data are missing in file matching. In practice, a good rule of thumb is to set the number of mixture components with the number of cell types. Fig. 2.15 shows the effect of the number of components. For a range of K, where K is the number of components, we repeated the single tube experiments in Section 2.4.1. Six points are first selected from from Fig. 2.9 and Table 2.2, and then used for $K = 6$. For models with more or less than 6 components, each centroid is initialized by random drawing from a Gaussian distribution centered at one of the six points. Once cluster centers are initialized, the rest of the parameters are initialized by following the method described in Algorithm 2.2. The best performance is given when $K = 7$, with the performance slightly better than the performance when $K = 6$. For values of K less than 6, the performance was much worse, and for values greater than 7, the performance gradually degraded as the number of components was increased.

2.5 Discussion

In this chapter, we demonstrated the use of a cluster-based nearest neighbor (Cluster-NN) imputation method for file matching of flow cytometry data. We applied the proposed algorithm on real flow cytometry data to generate a dataset of higher dimension by merging

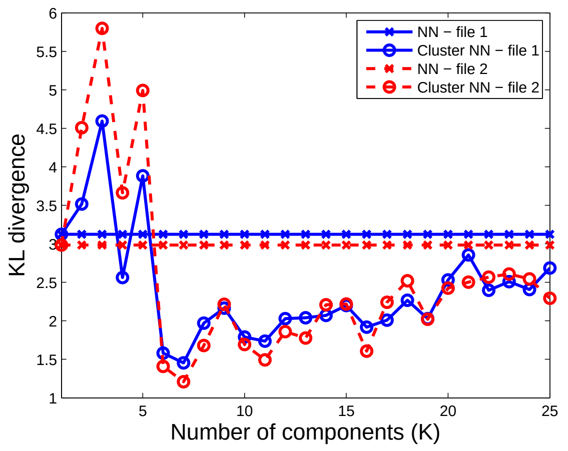


Fig. 2.15: The KL divergence of Cluster-NN imputation results over the number of components of a PPCA mixture model. As the NN method does not involve clustering, the KL divergence remains constant. The best performance of Cluster-NN is achieved near $K = 7$.

two data files of lower dimensions. The resulting matched file can be used for visualization and high-dimensional analysis of cellular attributes.

While the presented imputation method focused on the case of two files, it can be generalized to more than two files. We envision two possible extensions of the Cluster-NN imputation method. For concreteness, suppose that five files $\mathcal{X}_1, \cdots, \mathcal{X}_5$ are given and a missing variable of $\mathbf{x}_i \in \mathcal{X}_1$ is available in $\mathcal{X}_2$ and $\mathcal{X}_3$.

Method 1 The first approach fits a single mixture of PPCA model to the all units in the five files using the missing data EM algorithm in Section 2.3.2. According to their posterior probabilities, units in each file are clustered into classes. If $\mathbf{x}_i$ belongs to $\mathcal{X}_1^k$, then the similarities are computed between $\mathbf{x}_i$ and units in $\mathcal{X}_2^k$ and $\mathcal{X}_3^k$. Then the most similar unit is chosen to be the donor.

Method 2 In the second method, a pair of files are considered at a time by selecting and limiting the search for a donor to one of $\mathcal{X}_2$ and $\mathcal{X}_3$. One can pick a file with more cells, say $\mathcal{X}_3$. Thus, the donor candidates are found among units in $\mathcal{X}_3$. Then the PPCA mixture model is trained with the cells in $\mathcal{X}_1$ and $\mathcal{X}_3$ using the missing data EM algorithm. After units in $\mathcal{X}_1$ and $\mathcal{X}_3$ are labeled, a donor is found from $\mathcal{X}_3^k$ for $\mathbf{x}_i \in \mathcal{X}_1^k$.

Once a donor is elected either from *Method 1* or from *Method 2*, the missing variable of $\mathbf{x}_i$ is imputed from the donor. *Method 2* solves smaller problems involving less number of data points for model fitting, but needs to train mixture models multiple times to impute all the missing variables in the dataset. On the contrary, *Method 1* solves a single large problem involving all data points.

Future research directions include finding ways of automatic domain information extraction. The construction of covariance matrices from incomplete data in the initialization of the EM algorithm is also an interesting problem. We expect that better covariance structure estimation, which will be available from better prior information, will be helpful for better replication of non-symmetric and non-elliptic cell subpopulations in the imputed results.

In the present study, we validated our method with lymphocyte data, where, for certain marker combinations, cell types tend to form relatively well-defined clusters. However, for other samples and marker combinations, clusters may be more elongated or less well-defined due to cells being at different stages of physiologic development. Fig. 2.14 indicates that flow cytometry clusters are often not Gaussian distributed. It may therefore be worth extending the ideas here to incorporate non-elliptical clusters using, for example, skewed Gaussian or skewed multivariate t components [53]. The cluster merging technique of [24] may also be helpful in this regard.

2.A Appendix: Derivation of EM algorithm for mixture of PPCA model with missing data

Suppose that we are given an incomplete observation set. We can divide each unit $\mathbf{x}_n$ as $\mathbf{x}_n = \begin{bmatrix} \mathbf{x}_n^{o_n} \\ \mathbf{x}_n^{m_n} \end{bmatrix}$ by separating the observed components and the missing components. Note that we do not assume that the observed variables come first and the missing variables next, and this should be understood as a notational convenience.

In the PPCA mixture model, the probability distribution of $\mathbf{x}$ is

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}|k)$$

where K is the number of components in the mixture and π_k is a mixing weight corresponding to the component density $p(\mathbf{x}|k)$. We estimate the set of unknown parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, \mathbf{W}_k, \sigma_k^2\}$ using an EM algorithm from the partial observations $\{\mathbf{x}_1^{o_1}, \cdots, \mathbf{x}_N^{o_N}\}$.

To develop an EM algorithm, we introduce indicator variables $\mathbf{z}_n = (z_{n1}, \cdots, z_{nK})$ for $n = 1, \cdots, N$. One and only one entry of $\mathbf{z}_n$ is nonzero, and $z_{nk} = 1$ indicates that the kth component is responsible for generating $\mathbf{x}_n$. We also include a set of the latent variables $\mathbf{t}_{nk}$ for each component and missing variables $\mathbf{x}_n^{m_n}$ to form the complete data $(\mathbf{x}_n^{o_n}, \mathbf{x}_n^{m_n}, \mathbf{t}_{nk}, \mathbf{z}_n)$ for $n = 1, \cdots, N$ and $k = 1, \cdots, K$. Then the corresponding complete data likelihood function has the form

$$\begin{aligned}
\mathcal{L}_C =& \sum_n \sum_k z_{nk} \ln \left[\pi_k p(\mathbf{x}_n, \mathbf{t}_{nk})\right] \\
=& \sum_n \sum_k z_{nk} \Bigg[\ln \pi_k - \frac{d}{2} \ln \sigma_k^2 - \frac{1}{2\sigma_k^2} \mathrm{tr}\left((\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T\right) \\
&+ \frac{1}{\sigma_k^2} \mathrm{tr}\left((\mathbf{x}_n - \boldsymbol{\mu}_k)\mathbf{t}_{nk}^T \mathbf{W}_k^T\right) - \frac{1}{2\sigma_k^2} \mathrm{tr}\left(\mathbf{W}_k^T \mathbf{W}_k \mathbf{t}_{nk} \mathbf{t}_{nk}^T\right) \Bigg],
\end{aligned}$$

where terms independent of the parameters are not included in the second equality. Instead of developing an EM algorithm directly on this likelihood function $\mathcal{L}_C$, we extend the strategy in [70] and build a two-stage EM algorithm, where each stage is a two-step process. This approach monotonically increases the value of the log-likelihood each round [70].

In the first stage of the two-stage EM algorithm, we update the component weight π_k and the component mean $\boldsymbol{\mu}_k$. We form a complete data log-likelihood function with the component indicator variables $\mathbf{z}_n$ and missing variables $\mathbf{x}_n^m$, while ignoring the latent variables $\mathbf{t}_{nk}$. Then we have the following likelihood function:

$$\begin{aligned}\mathcal{L}_1 =& \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln[\pi_k p(\mathbf{x}_n^{o_n}, \mathbf{x}_n^{m_n}|k)] \\ =& \sum_{n}\sum_{k} z_{nk} \left[\ln \pi_k - \frac{1}{2}\ln|\mathbf{C}_k| - \frac{1}{2}\mathrm{tr}\left(\mathbf{C}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T\right)\right]\end{aligned}$$

where terms unrelated to the model parameters are omitted in the second line. We take the conditional expectation with respect to $p(\mathbf{z}_n, \mathbf{x}_n^{m_n}|\mathbf{x}_n^{o_n})$. Since the conditional probability factorizes as

$$p(\mathbf{z}_n, \mathbf{x}_n^{m_n}|\mathbf{x}_n^{o_n}) = p(\mathbf{z}_n|\mathbf{x}_n^{o_n})p(\mathbf{x}_n^{m_n}|\mathbf{z}_n, \mathbf{x}_n^{o_n}),$$

we have the following conditional expectations

$$\begin{aligned}\langle z_{nk}\rangle =& p(k|\mathbf{x}_n^{o_n}) = \frac{\pi_k p(\mathbf{x}_n^{o_n}|k)}{\sum_{k'}\pi_{k'}p(\mathbf{x}_n^{o_n}|k')}, \\ \langle z_{nk}\mathbf{x}_n^{m_n}\rangle =& \langle z_{nk}\rangle\langle \mathbf{x}_n^{m_n}\rangle, \\ \langle \mathbf{x}_n^{m_n}\rangle =& \boldsymbol{\mu}_k^{m_n} + \mathbf{C}_k^{m_n o_n}{\mathbf{C}_k^{o_n o_n}}^{-1}(\mathbf{x}_n^{o_n} - \boldsymbol{\mu}_k^{o_n}), \\ \langle z_{nk}\mathbf{x}_n^{m_n}{\mathbf{x}_n^{m_n}}^T\rangle =& \langle z_{nk}\rangle\langle \mathbf{x}_n^{m_n}{\mathbf{x}_n^{m_n}}^T\rangle, \\ \langle \mathbf{x}_n^{m_n}{\mathbf{x}_n^{m_n}}^T\rangle =& \mathbf{C}_k^{m_n m_n} - \mathbf{C}_k^{m_n o_n}{\mathbf{C}_k^{o_n o_n}}^{-1}\mathbf{C}_k^{o_n m_n} + \langle \mathbf{x}_n^{m_n}\rangle\langle {\mathbf{x}_n^{m_n}}^T\rangle\end{aligned}$$

where $\langle\cdot\rangle$ denotes the conditional expectation. Maximizing $\langle\mathcal{L}_1\rangle$ with respect to π_k, using a Lagrange multiplier, and with respect to μ_k give the parameter updates

$$\widehat{\pi}_k = \frac{1}{N}\sum_n \langle z_{nk}\rangle, \tag{2.5}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_n \langle z_{nk}\rangle \begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n}\rangle \end{bmatrix}}{\sum_n \langle z_{nk}\rangle}. \tag{2.6}$$

In the second stage, we include the latent variable $\mathbf{t}_{nk}$ as well to formulate the complete data log-likelihood function. The new values of $\widehat{\pi}_k$ and $\widehat{\boldsymbol{\mu}}_k$ are used in this step to compute sufficient statistics. Taking the conditional expectation on $\mathcal{L}_C$ with respect to $p(\mathbf{z}_n, \mathbf{t}_{nk}, \mathbf{x}_n^{m_n}|\mathbf{x}_n^{o_n})$, we have

$$\begin{aligned}\langle \mathcal{L}_C\rangle = &\sum_n\sum_k \langle z_{nk}\rangle \Big[\ln\widehat{\pi}_k - \frac{d}{2}\ln\sigma_k^2 - \frac{1}{2\sigma_k^2}\mathrm{tr}\left(\langle(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^T\rangle\right) \\ &+ \frac{1}{\sigma_k^2}\mathrm{tr}\left(\langle(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)\mathbf{t}_{nk}^T\rangle\mathbf{W}_k^T\right) - \frac{1}{2\sigma_k^2}\mathrm{tr}\left(\mathbf{W}_k^T\mathbf{W}_k\langle\mathbf{t}_{nk}\mathbf{t}_{nk}^T\rangle\right)\Big].\end{aligned}$$

Since the the conditional probability factorizes

$$p(\mathbf{z}_n, \mathbf{t}_{nk}, \mathbf{x}_n^{m_n}|\mathbf{x}_n^{o_n}) = p(\mathbf{z}_n|\mathbf{x}_n^{o_n})p(\mathbf{x}_n^{m_n}|\mathbf{z}_n, \mathbf{x}_n^{o_n})p(\mathbf{t}_{nk}|\mathbf{z}_n, \mathbf{x}_n^{o_n}, \mathbf{x}_n^{m_n}),$$

we can evaluate the conditional expectations as follows :

$$
\begin{aligned}
\langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^T \rangle &= \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right) \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right)^T \\
&\quad + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{nk} \end{bmatrix}, \\
Q_{nk} =& \mathbf{C}_k^{m_n m_n} - \mathbf{C}_k^{m_n o_n} \mathbf{C}_k^{o_n o_n^{-1}} \mathbf{C}_k^{o_n m_n}, \\
\langle \mathbf{t}_{nk} \rangle =& \mathbf{M}_k^{-1} \mathbf{W}_k^T (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k), \\
\langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k) \mathbf{t}_{nk}^T \rangle =& \langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^T \rangle \mathbf{W}_k \mathbf{M}_k^{-1}, \\
\langle \mathbf{t}_{nk} \mathbf{t}_{nk}^T \rangle =& \mathbf{M}_k^{-1} \mathbf{W}_k^T \langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^T \rangle \mathbf{W}_k \mathbf{M}_k^{-1} \\
& + \sigma_k^2 \mathbf{M}_k^{-1}.
\end{aligned}
$$

Recall that the $q \times q$ matrix $\mathbf{M}_k = \mathbf{W}_k^T \mathbf{W}_k + \sigma_k^2 \mathbf{I}$. Then the maximization of $\langle \mathcal{L}_C \rangle$ with respect to $\mathbf{W}_k$ and σ_k^2 leads to the parameter updates,

$$
\widehat{\mathbf{W}}_k = \left[\sum_n \langle z_{nk} \rangle \langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k) \mathbf{t}_{nk}^T \rangle \right] \left[\sum_n \langle z_{nk} \rangle \langle \mathbf{t}_{nk} \mathbf{t}_{nk}^T \rangle \right]^{-1}, \tag{2.7}
$$

$$
\begin{aligned}
\widehat{\sigma}_k^2 =& \frac{1}{d \sum_n \langle z_{nk} \rangle} \Bigg[\sum_n \langle z_{nk} \rangle \text{tr} \left(\langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k)^T \rangle \right) \\
& - 2 \sum_n \langle z_{nk} \rangle \text{tr} \left(\langle (\mathbf{x}_n - \widehat{\boldsymbol{\mu}}_k) \mathbf{t}_{nk}^T \rangle \mathbf{W}_k^T \right) \\
& + \sum_n \langle z_{nk} \rangle \text{tr} \left(\mathbf{W}_k^T \mathbf{W}_k \langle \mathbf{t}_{nk} \mathbf{t}_{nk}^T \rangle \right) \Bigg].
\end{aligned} \tag{2.8}
$$

Substituting the conditional expectations simplifies the M-step equations

$$
\widehat{\mathbf{W}}_k = \mathbf{S}_k \mathbf{W}_k (\sigma_k^2 \mathbf{I} + \mathbf{M}_k^{-1} \mathbf{W}_k^T \mathbf{S}_k \mathbf{W}_k)^{-1}, \tag{2.9}
$$

$$
\widehat{\sigma}_k^2 = \frac{1}{d} \text{tr} \left(\mathbf{S}_k - \mathbf{S}_k \mathbf{W}_k \mathbf{M}_k^{-1} \widehat{\mathbf{W}}_k^T \right) \tag{2.10}
$$

ID	N_1	N_2	N_e
Patient1	10000	10000	5223
Patient2	7000	7000	4408
Patient3	3000	3000	3190

Table 2.7: Datasets from three patients in the single tube experiments (Section 2.4.1). Each tube is divided into two data files and an evaluation set. N_1 and N_2 denote the sizes of the two data files, and N_e is the size of the evaluation set.

where

$$\mathbf{S}_k = \frac{1}{N\widehat{\pi}_k} \sum_n \langle z_{nk} \rangle \langle \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right) \left(\begin{bmatrix} \mathbf{x}_n^{o_n} \\ \langle \mathbf{x}_n^{m_n} \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right)^T \rangle.$$

Each iteration of the EM algorithm updates the set of old parameters $\{\pi_k, \boldsymbol{\mu}_k, \mathbf{W}_k, \sigma_k^2\}$ with the set of new parameters $\{\widehat{\pi}_k, \widehat{\boldsymbol{\mu}}_k, \widehat{\mathbf{W}}_k, \widehat{\sigma}_k^2\}$ in (2.5), (2.6), (2.9) and (2.10). The algorithm terminates when the value of the log-likelihood function changes less than a predefined accuracy constant.

2.B Appendix: Computing KL divergences

For each experiment in Section 2.4, we quantify the imputation results using the KL divergence.

2.B.1 Single tube experiments

In the single tube experiments, each dataset corresponding to the different patients is divided into two data files and a separate evaluation set. Table 2.7 summarizes the cell counts in these sets. N_1, N_2 and N_e are the cell counts of the two files and the hold-out set, respectively. After imputing the two files with either the NN or the Cluster-NN method,

ID	N_1	N_2	N_{e1}	N_{e2}	N_3
Patient1	10000	10000	3982	21828	47248
Patient2	8000	8000	14661	3793	28101
Patient3	2000	5000	1817	7228	9795

Table 2.8: Datasets from three patients in the multiple tube experiments (Section 2.4.2). N_1 and N_2 denote the sizes of the two data files, and N_{e1} and N_{e2} denote the sizes of the evaluation sets. N_3 is the number of cells in the additional tube that is treated as the ground truth.

the KL divergences are computed. The empirical estimate of the KL divergence is

$$\begin{aligned} KL(g \,\|\, f) =& \mathbb{E}_g \left[\log g - \log f \right] \\ \approx& \frac{1}{N_e} \sum_{n=1}^{N_e} \left[\log g\left(\widehat{\mathbf{x}}_n\right) - \log f\left(\widehat{\mathbf{x}}_n\right) \right] \\ \approx& \frac{1}{N_e} \sum_{n=1}^{N_e} \left[\log \widehat{g}\left(\widehat{\mathbf{x}}_n\right) - \log \widehat{f}\left(\widehat{\mathbf{x}}_n\right) \right] \end{aligned}$$

where the distributions f and g are replaced by their corresponding density estimates and the expectation is approximated by a finite sum over imputed results $\widehat{\mathbf{x}}_n$ on the hold-out set of size N_e. For $\widehat{f}$ and $\widehat{g}$, we used kernel density estimation on the ground truth data and the imputed data, respectively.

2.B.2 Multiple tube experiments

As explained in Section 2.4.2, three tubes per patient are available in the multiple tube experiments. The third tube of higher dimension is a reference dataset and is not involved during the file matching. Each of the two lower dimensional tubes is split into two halves. The first halves of the two tubes form the incomplete data: file 1 and file 2 with N_1 and N_2 cells, respectively. The second halves of size N_{e1} and N_{e2} form the evaluation sets and their imputed results are used to approximate the expectation of the KL divergence. For each patient, the sizes of these sets are shown in Table 2.8. The reason for splitting each tube in half is so that the data used to approximate the expectation are independent to the data used to estimate density of the imputed result. Therefore, the imputation result of file

1 is evaluated by

$$KL(g_1 \,\|\, f) \approx \frac{1}{N_{e1}} \sum_{n=1}^{N_{e1}} \left[\log \widehat{g}_1 \left(\widehat{\mathbf{x}}_n \right) - \log \widehat{f} \left(\widehat{\mathbf{x}}_n \right) \right]$$

where $\widehat{g}_1$ is the kernel density estimate based on imputed rows from the first half of tube 1. The third tube is treated as the ground truth data and used to obtain the density estimate $\widehat{f}$.

When evaluating the KL divergence of file 2, $\widehat{g}_1$ is replaced by $\widehat{g}_1$, the kernel density estimate on the imputed result of file 2, and the finite sum is taken over the evaluation set of size N_{e2}.

CHAPTER 3

EM Algorithms for Multivariate Gaussian Mixture Models with Truncated and Censored Data

3.1 Introduction

This chapter addresses the problem of fitting Gaussian mixture models on censored and truncated multivariate data. Censoring and truncation arise in numerous applications, for reasons such as fundamental limitations of measuring equipment, or from experimental design. Data are said to be censored when the exact values of measurements are not reported. For example, the needle of a scale that does not provide a reading over 200 kg will show 200 kg for all the objects that weigh more than the limit. Data are said to be truncated when the number of measurements outside a certain range are not reported. For example, an electronic component manufacturer can limit the test duration to 100 hours for life time tests. A data collector might provide only the survival times of the components that failed during the test, but not the number of components tested. In these cases, it is often natural to seek the statistical characteristics of the original (uncensored and untruncated) data instead of the observed (censored or truncated) data.

This work is motivated by the analysis of flow cytometry data. Flow cytometry is an essential tool in the diagnosis of diseases such as acute leukemias, chronic lymphoproliferative disorders, and malignant lymphomas [9, 63]. A flow cytometer measures antigen-based markers associated to cells in a cell population. The analysis of flow cytometry data involves dividing cells into subpopulations and inspecting their characteristics. This clustering process, called gating, is performed manually in practice. To automate gating, researchers recently have been investigating mixture models [8, 15, 38, 44, 53].

However, a flow cytometer measures a limited range of signal strength and records each marker value within a fixed range, such as between 0 and 1023. If a measurement falls outside the range, then the value is replaced by the nearest legitimate value; that is, a value smaller than 0 is censored to 0 and a value larger than 1023 is censored to 1023. Moreover, a large portion of cell measurements can be truncated from recording by the judgment of an operator. Therefore, mixture model fitting that does not account for censoring and truncation can result in biases in the parameter estimates and poor gating results. In flow cytometry, a mixture model fitting algorithm should take censoring and truncation into account to avoid biases. Here we present an Expectation-Maximization (EM) algorithm to fit a multivariate mixture model while accounting for both censoring and truncation.

When censored and truncated data are from an exponential family, Dempster, Laird and Rubin [20] suggested using the EM procedure to find the maximum likelihood estimate. Atkinson [4] derived an EM algorithm for a finite mixture of two univariate normal distributions when data is right-censored. Chauveau [17] also studied a mixture model of univariate censored data, and presented an EM algorithm and its stochastic version. McLachlan and Jones [47] developed an EM algorithm for univariate binned and truncated data. Cadez et al. [10] extended the development of McLachlan and Jones [47] to multivariate case and applied to bivariate blood sample measurements for diagnosis of iron deficiency anemia. To our knowledge, previous work has not addressed censored multivariate data, or continuous (not binned) truncated multivariate data. Furthermore, censoring and truncation have been treated separately. As we will show below, the development of the truncated data EM algorithm and the censored data EM algorithm are closely related to the truncated multivariate Gaussian distribution [46, 67] and we handle these two problems together under the same framework.

Our algorithms make use of recent methods [22, 28–30] for evaluating the cumulative distribution function of a multivariate Gaussian. These algorithms run slower as the dimension increases but, when combined with modern computing resources, they can be used successfully in the kinds of lower dimensional settings where mixture methods tend to be applied. Our MATLAB implementation is available online at `http://www.eecs.umich.edu/~cscott/code/tcem.zip`.

In the following, we briefly review the standard EM algorithm in Section 3.2. Then we consider truncation (Section 3.3), censoring (Section 3.4) and both truncation and censoring (Section 3.5). We derive EM algorithms for each case and discuss how these algorithms improve the standard EM algorithm. We discuss the initialization and the termination of the EM algorithms in Section 3.6. Experimental results are reported in Section 3.7, and Section 3.8 concludes.

3.2 Standard EM Algorithm

In a mixture model, the probability density function of an observation is

$$f(\mathbf{y};\Theta) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{y};\theta_k) \tag{3.1}$$

where π_k are positive mixing weights summing to one, f_k are component density functions parameterized by θ_k, and $\Theta = (\pi_1, \cdots, \pi_K, \theta_1, \cdots, \theta_K)$ is the collection of all model parameters. Each observation is assumed to be from one of the K components. A common choice of the component density is a multivariate normal with mean $\boldsymbol{\mu}_k$ and covariance Σ_k.

Given a set of independent observations $\mathbf{y}^{1:N} := \{\mathbf{y}^1, \cdots, \mathbf{y}^N\}$ in $\mathcal{Y} \subseteq \mathbb{R}^d$, the objective is to fit such a model to the data.

The EM algorithm proposed by Dempster et al. [20] is a widely-applied approach for finding the maximum likelihood estimate of a mixture model. In the EM procedure, the unknown true association of each observation to a component is considered missing, and the expected likelihood of the "complete" data is maximized. Let $\mathbf{z}^n \in \{0,1\}^K$ be the membership indicator variable such that $z_k^n = 1$ if $\mathbf{y}^n$ is generated from f_k and 0 otherwise. Then the complete log-likelihood function becomes

$$\begin{aligned}\mathcal{L}(\Theta) =& \sum_n \sum_k z_k^n \left[\ln \pi_k + \ln f_k(\mathbf{y}^n)\right] \\ =& \sum_n \sum_k z_k^n \Big[\ln \pi_k - \frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} \mathrm{tr}\left(\Sigma_k^{-1} (\mathbf{y}^n - \boldsymbol{\mu}_k)(\mathbf{y}^n - \boldsymbol{\mu}_k)^T\right) \Big] + \text{const}\end{aligned} \tag{3.2}$$

where tr denotes the trace operator of a matrix. The EM algorithm first computes $\mathcal{Q}(\Theta; \Theta^{old}) = \mathbb{E}[\mathcal{L}(\Theta)|\mathbf{y}^{1:N}; \Theta^{old}]$ (E step) and then finds a new Θ such that $\Theta^{new} = \arg\max_\Theta \mathcal{Q}(\Theta; \Theta^{old})$ (M step). The EM algorithm repeats the E step and M step and updates Θ each iteration. An acclaimed property of the EM algorithm is that each round the value of the log-likelihood monotonically increases [35]. The E step simplifies to computing the conditional probabilities

$$\langle z_k^n \rangle := p(z_k^n = 1 | \mathbf{y}^n; \Theta^{old}) = \frac{\pi_k\, f_k(\mathbf{y}^n)}{\sum_l \pi_l\, f_l(\mathbf{y}^n)}.$$

In the M step, we have an update rule in closed form:

$$\widehat{\pi}_k = \frac{1}{N} \sum_n \langle z_k^n \rangle, \tag{3.3}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_n \langle z_k^n \rangle \mathbf{y}^n}{\sum_n \langle z_k^n \rangle}, \tag{3.4}$$

$$\widehat{\Sigma}_k = \frac{\sum_n \langle z_k^n \rangle (\mathbf{y}^n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{y}^n - \widehat{\boldsymbol{\mu}}_k)^T}{\sum_n \langle z_k^n \rangle}. \tag{3.5}$$

The EM algorithm alternates between the E step and the M step until convergence.

When truncation and/or censoring occur, however, the true values of $\mathbf{y}_n$ are not always available and the blindfold use of the standard EM algorithm can result in undesirable parameter estimates.

3.3 Truncated Data EM Algorithm

Truncation restricts the observation to a subset $\mathcal{Y}_T \subseteq \mathcal{Y}$. Thus, the data points outside $\mathcal{Y}_T$ are not available for estimation of Θ. For example, in clinical flow cytometry, cells with low forward scatter (FS) value are not of much pathological interest and are often dropped during data collection to save data storage space. Hence, all the recorded forward scatter values are always greater than a truncation level chosen by an operator.

Here we assume that the observation window $\mathcal{Y}_T$ is a hyper-rectangle in $\mathbb{R}^d$ with two vertices $\mathbf{s} = (s_1, \cdots, s_d)^T$ and $\mathbf{t} = (t_1, \cdots, t_d)^T$ on the diagonal opposites such that every observed event satisfies $\mathbf{s} \le \mathbf{y}^n \le \mathbf{t}$. These inequalities are element-wise, and $s_i = -\infty$ and $t_i = \infty$ mean no truncation below and above, respectively, on the ith coordinate.

The probability density function after truncation is given by $g(\mathbf{y}) = f(\mathbf{y}) / \int_{\mathbf{s}}^{\mathbf{t}} f(\mathbf{y}')d\mathbf{y}'$ for $\mathbf{y} \in [\mathbf{s}, \mathbf{t}]$ and by $g(\mathbf{y}) = 0$ otherwise. Then it can be easily seen that $g(\mathbf{y})$ is also a mixture

$$g(\mathbf{y}) = \sum_{k=1}^{K} \eta_k \, g_k(\mathbf{y}) \, \mathbf{1}_{\{[\mathbf{s},\mathbf{t}]\}}(\mathbf{y}) \tag{3.6}$$

with mixing weights η_k and component density functions g_k:

$$\eta_k = \pi_k \frac{\int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y})d\mathbf{y}}{\int_{\mathbf{s}}^{\mathbf{t}} f(\mathbf{y})d\mathbf{y}} \qquad \text{and} \qquad g_k(\mathbf{y}) = \frac{f_k(\mathbf{y})}{\int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}')d\mathbf{y}'} \mathbf{1}_{\{[\mathbf{s},\mathbf{t}]\}}(\mathbf{y}). \tag{3.7}$$

The indicator function $\mathbf{1}_{\{A\}}(\mathbf{y})$ equals one if $\mathbf{y} \in A$ and zero otherwise. Hence, the component density functions g_k are truncated versions of the original component density functions f_k.

Proceeding similarly as in Section 3.2, we can express the complete data log-likelihood as

$$\begin{aligned}\mathcal{L}_T(\Theta) =& \sum_n \sum_k z_k^n \left[\ln \eta_k + \ln g_k(\mathbf{y}^n)\right] \\ =& \sum_n \sum_k z_k^n \left[\ln \eta_k + \ln f_k(\mathbf{y}^n) - \ln \int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}) d\mathbf{y}\right]. \end{aligned} \tag{3.8}$$

Recall that $\mathbf{z}^n$ are the component membership indicator variables. It is conceivable to define the complete data differently by treating the unknown number of truncated measurements as a random variable and including it into the complete data [20]. However, this approach requires to make an additional assumption on the distribution of the new parameter. It might generalize our approach, but it can be sensitive to the choice of the distribution of the number of truncated sample points [27].

The E step applied to (3.8) requires us to compute

$$\begin{aligned}\mathcal{Q}_T(\Theta; \Theta^{old}) =& \mathbb{E}[\mathcal{L}_T(\Theta) | \mathbf{y}^{1:N}; \Theta^{old}] \\ =& \sum_n \sum_k \langle z_k^n \rangle \left[\ln \eta_k + \ln f_k(\mathbf{y}^n) - \ln \int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}) d\mathbf{y}\right]. \end{aligned}$$

The main difference from (3.2) is the terms of normalizing factors, $\ln \int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}) d\mathbf{y}$, which do not complicate the E step of the EM algorithm, and whose calculation is discussed below. Thus, the E step is simply computing the posterior probability that $\mathbf{y}^n$ belongs to component k

$$\langle z_k^n \rangle := p(z_k^n = 1 | \mathbf{y}^n) = \frac{\eta_k \, g_k(\mathbf{y}^n)}{\sum_l \eta_l \, g_l(\mathbf{y}^n)} = \frac{\pi_k \, f_k(\mathbf{y}^n)}{\sum_l \pi_l \, f_l(\mathbf{y}^n)}. \tag{3.9}$$

As the last equality indicates, this posterior remains unchanged as if $\mathbf{y}^n$ in the truncated data is from the entire sample space $\mathcal{Y}$. Then the M step computes $\widehat{\Theta}$ that maximizes $\mathcal{Q}_T(\Theta; \Theta^{old})$, which is found by taking the derivatives of $\mathcal{Q}_T(\Theta; \Theta^{old})$ with respect to each η_k, $\boldsymbol{\mu}_k$ and Σ_k, and setting to zero. Since η_k should satisfy $\sum_k \eta_k = 1$, a Lagrange multiplier is used to find the maximizer. Using (3.35) and (3.36) in 3.A.2 to calculate the derivatives of the normalizing factors, we have the following M step equations:

$$\widehat{\eta}_k = \frac{1}{N} \sum_n \langle z_k^n \rangle, \tag{3.10}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_n \langle z_k^n \rangle \mathbf{y}^n}{\sum_n \langle z_k^n \rangle} - \mathbf{m}_k, \tag{3.11}$$

$$\widehat{\Sigma}_k = \frac{\sum_n \langle z_k^n \rangle (\mathbf{y}^n - \widehat{\boldsymbol{\mu}}_k)(\mathbf{y}^n - \widehat{\boldsymbol{\mu}}_k)^T}{\sum_n \langle z_k^n \rangle} + H_k \tag{3.12}$$

where $\widehat{\eta}_k$, $\widehat{\boldsymbol{\mu}}_k$ and $\widehat{\Sigma}_k$ denote the new parameters and

$$\mathbf{m}_k = \mathcal{M}^1(\mathbf{0}, \Sigma_k\, ;\, [\mathbf{s} - \boldsymbol{\mu}_k, \mathbf{t} - \boldsymbol{\mu}_k]), \tag{3.13}$$

$$H_k = \Sigma_k - \mathcal{M}^2(\mathbf{0}, \Sigma_k\, ;\, [\mathbf{s} - \boldsymbol{\mu}_k, \mathbf{t} - \boldsymbol{\mu}_k]). \tag{3.14}$$

The notations $\mathcal{M}^1(\boldsymbol{\mu}, \Sigma\, ;\, [\mathbf{s}', \mathbf{t}'])$ and $\mathcal{M}^2(\boldsymbol{\mu}, \Sigma\, ;\, [\mathbf{s}', \mathbf{t}'])$ in (3.13) and (3.14) indicate the first and second moments of a Gaussian with mean $\boldsymbol{\mu}$ and covariance Σ when it is truncated to a hyper-rectangle with vertices $\mathbf{s}'$ and $\mathbf{t}'$. We discuss the computational aspects of these moments in 3.A. Comparing (3.10)-(3.12) with the standard EM equations (3.3)-(3.5) shows that the updates for truncated data are similar to those for untruncated data except the correction terms $\mathbf{m}_k$ and H_k.

The original component weight π_k can be recovered from (3.7). The normal integrals can be evaluated with the help of computational tools for evaluating the multivariate normal cumulative distribution function. Our implementation relies on `mvncdf` function in the MATLAB 7.9.0 statistics toolbox, which uses algorithms developed by Drezner and Wesolowsky [22] and by Genz [28] for bivariate and trivariate Gaussian. The toolbox uses a quasi-Monte Carlo integration algorithm developed by Genz and Bretz [29, 30] for four or more dimensional Gaussians.

3.4 Censored Data EM Algorithm

As discussed above, truncation excludes data points from the dataset, and the number of data points falling outside the measuring range remains unknown. On the contrary, censoring retains such data points while their exact locations remain unknown.

In the following, we investigate censoring on a hyper-rectangle, in which each data point $\mathbf{y}^n$ is censored above at $\mathbf{b} = (b_1, \cdots, b_d)^T$ and below at $\mathbf{a} = (a_1, \cdots, a_d)^T$. [1] Let $\mathcal{Y}_0, \mathcal{Y}_1, \cdots, \mathcal{Y}_C$ be a partition of the overall sample space $\mathcal{Y}$. If $\mathbf{y}^n \in \mathcal{Y}_0$, we observe the exact values of $\mathbf{y}^n$. When $\mathbf{y}^n \in \mathcal{Y}_c, c > 0$, however, censoring occurs and the true values are modified so that

$$x_i^n = y_i^n \mathbf{1}_{\{[a_i,b_i]\}}(y_i^n) + a_i \mathbf{1}_{\{(-\infty,a_i)\}}(y_i^n) + b_i \mathbf{1}_{\{(b_i,\infty)\}}(y_i^n), \qquad \forall i, \forall n$$

are observed. Therefore, instead of $\mathbf{y}^{1:N}$, we obtain a set of observations $\mathbf{x}^{1:N}$, which satisfy $a_i \leq x_i^n \leq b_i$ for $i = 1, \cdots, d$ and $n = 1, \cdots, N$. Note that $a_i = -\infty$ means no censoring below and $b_i = \infty$ means no censoring above.

Chauveau [17] also studied the analysis of censored data. The difference is that in his setup $\mathbf{x}^n = c$ if $\mathbf{y}^n \in \mathcal{Y}_c$, $c > 0$, whereas in ours some coordinates can preserve their exact values. Furthermore, while his primary concern remained on univariate data, our focus extends to multivariate data.

As described above, unless $\mathbf{y}^n \in \mathcal{Y}_0 = \prod_{i=1}^d [a_i, b_i]$, one or more coordinates are censored and its original location is lost. However, we can infer which partition $\mathbf{y}^n$ belongs to from $\mathbf{x}^n$, the censored observation of $\mathbf{y}^n$, by noting when $x_i^n = a_i$ or b_i. Since each data vector may have different censoring patterns, let the censored and uncensored coordinates be indexed by m_n and o_n, respectively, so that $y_i^n, i \in m_n$, are censored values and $y_i^n, i \in$

[1]For univariate data ($d = 1$), left and right censoring are the usual terms.

o_n, are observed values. Then $\mathbf{y}^n$ can be divided into the form $\mathbf{y}^n = \begin{bmatrix} \mathbf{y}^n_{o_n} \\ \mathbf{y}^n_{m_n} \end{bmatrix}$ where $\mathbf{y}^n_{m_n} = (y^n_i, i \in m_n)^T$ and $\mathbf{y}^n_{o_n} = (y^n_i, i \in o_n)^T$ denote the censored and uncensored components of $\mathbf{y}^n$. Note that this does not imply that the vector is arranged to have this pattern and should be understood as a notational convenience. Then the likelihood of $\mathbf{x}^n$ is

$$f(\mathbf{x}^n) = f(\mathbf{y}^n) \qquad \text{for } \mathbf{y}^n \in \mathcal{Y}_0 \tag{3.15}$$

$$\begin{aligned} f(\mathbf{x}^n) &= \int_{\mathcal{X}_{c_n}} f(\mathbf{y}_{m_n}, \mathbf{y}^n_{o_n}) d\mathbf{y}_{m_n} \\ &= f(\mathbf{x}^n_{o_n}) \int_{\mathcal{X}_{c_n}} f(\mathbf{y}_{m_n} | \mathbf{x}^n_{o_n}) d\mathbf{y}_{m_n} \qquad \text{for } \mathbf{y}^n \in \mathcal{Y}_{c_n}, c_n > 0 \end{aligned} \tag{3.16}$$

where the integration is only over the censored coordinates, and $\mathcal{X}_{c_n}$ denote the corresponding integration range. For example, if $x^n_1 = a_1$ and $x^n_2 = b_2$ while other elements are strictly between $\mathbf{a}$ and $\mathbf{b}$, then $\mathcal{Y}_{c_n} = (-\infty, a_1) \times (b_2, \infty) \times \prod_{i=3}^d [a_i, b_i]$, $\mathcal{X}_{c_n} = (-\infty, a_1) \times (b_2, \infty)$ and (3.16) becomes

$$f(\mathbf{x}^n) = f(\mathbf{x}^n_{o_n}) \int_{-\infty}^{a_1} \int_{b_2}^{\infty} f(y_1, y_2 | \mathbf{x}^n_{o_n}) dy_2 \, dy_1.$$

To invoke the EM machinery, we first compute the expected complete log-likelihood

$$\begin{aligned} \mathcal{Q}_C(\Theta; \Theta^{old}) =& \mathbb{E}[\mathcal{L}(\Theta) | \, \mathbf{x}^{1:N}; \Theta^{old}] \\ =& \mathbb{E}\Bigg[\sum_n \sum_k z^n_k \Bigg[\ln \pi_k - \frac{1}{2} \ln |\Sigma_k| \\ & - \frac{1}{2} \mathrm{tr} \left(\Sigma_k^{-1} \left(\begin{bmatrix} \mathbf{y}^n_{o_n} \\ \mathbf{y}^n_{m_n} \end{bmatrix} - \boldsymbol{\mu}_k \right) \left(\begin{bmatrix} \mathbf{y}^n_{o_n} \\ \mathbf{y}^n_{m_n} \end{bmatrix} - \boldsymbol{\mu}_k \right)^T \right) \Bigg] \Bigg| \, \mathbf{x}^{1:N}; \Theta^{old} \Bigg]. \end{aligned}$$

Hence, we need to find posterior probabilities, $p(z^n_k = 1 | \mathbf{x}^n)$, and conditional expectations, $\mathbb{E}[z^n_k \, \mathbf{y}^n_{m_n} | \mathbf{x}^n] = p(z^n_k = 1 | \mathbf{x}^n) \, \mathbb{E}[\mathbf{y}^n_{m_n} | \mathbf{x}^n, z^n_k = 1]$ and $\mathbb{E}[z^n_k \, \mathbf{y}^n_{m_n} {\mathbf{y}^n_{m_n}}^T | \mathbf{x}^n] = p(z^n_k = 1 | \mathbf{x}^n) \, \mathbb{E}[\mathbf{y}^n_{m_n} {\mathbf{y}^n_{m_n}}^T | \mathbf{x}^n, z^n_k = 1]$.

The posterior probability is

$$\langle z_k^n \rangle := p(z_k^n = 1 | \mathbf{x}^n) = \frac{\pi_k \, f_k(\mathbf{x}^n)}{\sum_l \pi_l \, f_l(\mathbf{x}^n)}, \tag{3.17}$$

and it can be computed by (3.15) or (3.16). When one or more coordinates are censored, $f_k(\mathbf{x}^n) = f_k(\mathbf{y}_{o_n}^n) \int_{\mathcal{X}_{c_n}} f_k(\mathbf{y}_{m_n} | \mathbf{y}_{o_n}^n) \, d\mathbf{y}_{m_n}$; thus; it is a product of the probability density function and the cumulative distribution function of Gaussians of lower dimensions, and can be evaluated as explained in 3.A.1.

The conditional expectations are taken with respect to $f_k(\mathbf{y}_m | \mathbf{x})$. Because $f_k(\mathbf{y}_m | \mathbf{y}_o)$ is a normal density function [2] and satisfies

$$f_k(\mathbf{y}_m | \mathbf{x}) = f_k(\mathbf{y}_m | \mathbf{y}_o, \mathbf{y} \in \mathcal{Y}_c) = \frac{f_k(\mathbf{y}_m | \mathbf{y}_o)}{\int_{\mathcal{X}_c} f_k(\mathbf{y}_m | \mathbf{y}_o) \, d\mathbf{y}_m} \mathbf{1}_{\{\mathcal{X}_c\}}(\mathbf{y}_m), \tag{3.18}$$

the conditional density $f_k(\mathbf{y}_m | \mathbf{x})$ is a truncated normal density function over $\mathcal{X}_c$. Then we can calculate the following sufficient statistics of $\mathcal{Q}_C$:

$$\begin{aligned} \langle \mathbf{y}_{m_n}^n | k \rangle :=& \mathbb{E}[\mathbf{y}_{m_n}^n | \mathbf{x}^n, z_k^n = 1] = \mathbb{E}[\mathbf{y}_{m_n}^n | \mathbf{y}_{o_n}^n, \mathbf{y}^n \in \mathcal{Y}_{c_n}, z_k^n = 1] \\ =& \mathcal{M}^1(\boldsymbol{\mu}_{m_n|o_n}^k, \Sigma_{m_n|o_n}^k \, ; \, \mathcal{X}_{c_n}), \end{aligned} \tag{3.19}$$

$$\begin{aligned} \langle \mathbf{y}_{m_n}^n \mathbf{y}_{m_n}^{n\,T} | k \rangle :=& \mathbb{E}[\mathbf{y}_{m_n}^n \mathbf{y}_{m_n}^{n\,T} | \mathbf{x}^n, z_k^n = 1] = \mathbb{E}[\mathbf{y}_{m_n}^n \mathbf{y}_{m_n}^{n\,T} | \mathbf{y}_{o_n}^n, \mathbf{y}^n \in \mathcal{Y}_{c_n}, z_k^n = 1] \\ =& \mathcal{M}^2(\boldsymbol{\mu}_{m_n|o_n}^k, \, \Sigma_{m_n|o_n}^k \, ; \, \mathcal{X}_{c_n}) \end{aligned} \tag{3.20}$$

where $\boldsymbol{\mu}_{m_n|o_n}^k$ and $\Sigma_{m_n|o_n}^k$ are the mean and covariance of $f_k(\mathbf{y}_{m_n}^n | \mathbf{y}_{o_n}^n)$. Recall that $\mathcal{M}^1$ and $\mathcal{M}^2$ denote the first and second moments of a truncated normal distribution (see 3.A.1).

Next, we maximize $\mathcal{Q}_C$ with respect to Θ. Again using a Lagrange multiplier, maximization with respect to π_k gives

$$\widehat{\pi}_k = \frac{1}{N} \sum_n \langle z_k^n \rangle. \tag{3.21}$$

[2]If $\mathbf{y} = (\mathbf{y}_m^T, \mathbf{y}_o^T)^T$ is normally distributed with mean $\boldsymbol{\mu}$ and covariance Σ, then the conditional distribution of its partition, $\mathbf{y}_m | \mathbf{y}_o$, is also normally distributed with mean $\boldsymbol{\mu}_{m|o} = \boldsymbol{\mu}_m + \Sigma_{m,o} \Sigma_{o,o}^{-1} (\mathbf{y}_o - \boldsymbol{\mu}_o)$ and covariance $\Sigma_{m|o} = \Sigma_{m,m} - \Sigma_{m,o} \Sigma_{o,o}^{-1} \Sigma_{o,m}$.

Similarly, maximization with respect to $\boldsymbol{\mu}_k$ and Σ_k leads to

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_n \langle z_k^n \rangle \begin{bmatrix} \mathbf{y}_{o_n}^n \\ \langle \mathbf{y}_{m_n}^n | k \rangle \end{bmatrix}}{\sum_n \langle z_k^n \rangle}, \tag{3.22}$$

$$\widehat{\Sigma}_k = \frac{\sum_n \langle z_k^n \rangle S_k^n}{\sum_n \langle z_k^n \rangle} \tag{3.23}$$

where

$$S_k^n = \left(\begin{bmatrix} \mathbf{y}_{o_n}^n \\ \langle \mathbf{y}_{m_n}^n | k \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right) \left(\begin{bmatrix} \mathbf{y}_{o_n}^n \\ \langle \mathbf{y}_{m_n}^n | k \rangle \end{bmatrix} - \widehat{\boldsymbol{\mu}}_k \right)^T + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & R_k^n \end{bmatrix}, \tag{3.24}$$

$$R_k^n = \langle \mathbf{y}_{m_n}^n \mathbf{y}_{m_n}^n{}^T | k \rangle - \langle \mathbf{y}_{m_n}^n | k \rangle \langle \mathbf{y}_{m_n}^n | k \rangle^T. \tag{3.25}$$

Notice that these equations (3.21)-(3.23) resemble the update equations (3.3)-(3.5) of the standard EM algorithm. In the censored data EM algorithm, the censored elements of $\mathbf{y}^n$ are replaced by the conditional means $\langle \mathbf{y}_{m_n}^n | k \rangle$ and the sample covariance correction R_k^n. When none of the data points are censored, these update equations are equivalent to the standard EM algorithm.

3.5 Truncated and Censored Data EM Algorithm

In this section, we consider truncation and censoring together and present an EM procedure that encompasses the algorithms above.

Truncation reduces the sample space from $\mathcal{Y}$ to $\mathcal{Y}_T$. By restricting the partition regions $\mathcal{Y}_c$ and the integration ranges $\mathcal{X}_c$ to this reduced sample space $\mathcal{Y}_T$, we can see that the likelihood of an observation $\mathbf{x}$ is $g(\mathbf{x}) = f(\mathbf{x}) / \int_{\mathbf{s}}^{\mathbf{t}} f(\mathbf{y})\, d\mathbf{y}$ where the numerator $f(\mathbf{x})$ is defined by (3.15) and (3.16). Then this truncated distribution $g(\mathbf{x})$ is a mixture $g(\mathbf{x}) = \sum_{k=1}^{K} \eta_k\, g_k(\mathbf{x})$ with mixing weights η_k and component density functions g_k, where

$$\eta_k = \pi_k \frac{\int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}) d\mathbf{y}}{\int_{\mathbf{s}}^{\mathbf{t}} f(\mathbf{y}) d\mathbf{y}} \qquad \text{and} \qquad g_k(\mathbf{x}) = \frac{f_k(\mathbf{x})}{\int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y}) d\mathbf{y}}. \tag{3.26}$$

The E step of the EM algorithm begins by finding the expectation

$$\begin{aligned}\mathcal{Q}_{TC}(\Theta;\Theta^{old}) =&\mathbb{E}[\mathcal{L}_T(\Theta)|\mathbf{x}^{1:N};\Theta^{old}]\\ =&\mathbb{E}\left[\sum_n\sum_k z_k^n\left[\ln\eta_k+\ln f_k(\mathbf{y}^n)-\ln\int_{\mathbf{s}}^{\mathbf{t}} f_k(\mathbf{y})d\mathbf{y}\right]\middle|\mathbf{x}^{1:N};\Theta^{old}\right]\end{aligned}$$

conditional on the observed data. This involves computing the posterior probabilities

$$\langle z_k^n\rangle := p(z_k^n=1|\mathbf{x}^n)=\frac{\eta_k\, g_k(\mathbf{x}^n)}{\sum_l \eta_l\, g_l(\mathbf{x}^n)}=\frac{\pi_k\, f_k(\mathbf{x}^n)}{\sum_l \pi_l\, f_l(\mathbf{x}^n)}$$

and the conditional expectations $\langle\mathbf{y}_{m_n}^n|k\rangle := \mathbb{E}[\mathbf{y}_{m_n}^n|\mathbf{x}^n, z_k^n=1]$ and $\langle\mathbf{y}_{m_n}^n\mathbf{y}_{m_n}^{n\,T}|k\rangle := \mathbb{E}[\mathbf{y}_{m_n}^n\mathbf{y}_{m_n}^{n\,T}|\mathbf{x}^n, z_k^n=1]$ with respect to $g_k(\mathbf{y}_m|\mathbf{x})$. Since $g_k(\mathbf{y}_m|\mathbf{x})$ satisfies

$$g_k(\mathbf{y}_m|\mathbf{x})=g_k(\mathbf{y}_m|\mathbf{y}_o,\mathbf{y}\in\mathcal{Y}_c)=\frac{g_k(\mathbf{y}_m|\mathbf{y}_o)}{\int_{\mathcal{X}_c} g_k(\mathbf{y}_m|\mathbf{y}_o)\,d\mathbf{y}_m}=\frac{f_k(\mathbf{y}_m|\mathbf{y}_o)}{\int_{\mathcal{X}_c} f_k(\mathbf{y}_m|\mathbf{y}_o)\,d\mathbf{y}_m}$$

from (3.26) and this equals (3.18), we can deduce that the sufficient statistics $\langle z_k^n\rangle$, $\langle\mathbf{y}_{m_n}^n|k\rangle$ and $\langle\mathbf{y}_{m_n}^n\mathbf{y}_{m_n}^{n\,T}|k\rangle$ retain the same forms of (3.17), (3.19) and (3.20) in Section 3.4.

In the M step, we find the new parameters $\widehat{\eta}_k$, $\widehat{\boldsymbol{\mu}}_k$ and $\widehat{\Sigma}_k$ that maximize $\mathcal{Q}_{TC}$. To take account of the constraint $\sum_k\eta_k=1$, a Lagrange multiplier is used in the maximization with respect to η_k. Combining with the quantities computed in the E step, we obtain the following update equations

$$\widehat{\eta}_k =\frac{1}{N}\sum_n\langle z_k^n\rangle, \tag{3.27}$$

$$\widehat{\boldsymbol{\mu}}_k =\frac{\sum_n\langle z_k^n\rangle\begin{bmatrix}\mathbf{y}_{o_n}^n\\ \langle\mathbf{y}_{m_n}^n|k\rangle\end{bmatrix}}{\sum_n\langle z_k^n\rangle}-\mathbf{m}_k, \tag{3.28}$$

$$\widehat{\Sigma}_k =\frac{\sum_n\langle z_k^n\rangle S_k^n}{\sum_n\langle z_k^n\rangle}+H_k \tag{3.29}$$

where the correction terms $\mathbf{m}_k$ and H_k are given in (3.13) and (3.14), and the matrix S_k^n is given in (3.24). Recall that the original component weight π_k can be obtained from

η_k through (3.26) as in Section 3.3. The remarks on mean and covariance updates in the truncated data EM algorithm and the censored data EM algorithm naturally lead to an observation that (3.28) and (3.29) have the combined forms of (3.11) and (3.22), and, respectively, (3.12) and (3.23).

3.6 Initialization and Termination of EM Algorithms

The initialization is an important issue because the result from an EM algorithm is often sensitive to the initial parameter setting. We suggest to proceed as follows to initialize the parameters for the presented EM algorithms. First, perform the k-means clustering algorithm multiple times with different starting points. Next, compute the mixture model parameter and the corresponding complete data log-likelihood from each k-means clustering result. Finally, choose the parameter that achieves the largest log-likelihood as an initial parameter estimate for the truncated and censored data EM algorithm.

To check the convergence of the EM algorithms, we compute the expectation of the complete data log-likelihood function after each iteration. We terminate the EM algorithms when the relative change of the expected complete data log-likelihood falls below 10^{-10}, or when the number of iteration reaches a fixed number, say 500.

In the following section, we use the described initialization and termination methods for experiments.

3.7 Experiments and Results

We present experimental results to demonstrate the algorithms described above. In the following, we describe experiments on univariate and bivariate synthetic data, and on multi-dimensional flow cytometry data.

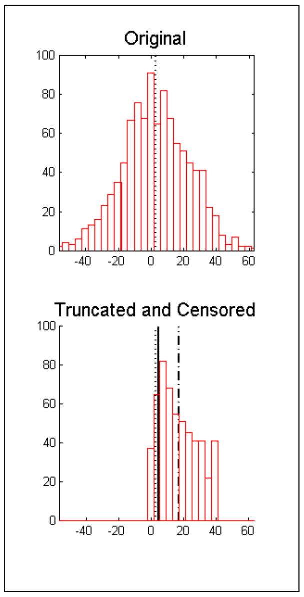
3.7.1 Synthetic Data

In each experiment, we generated datasets from a known distribution and performed censoring and truncation. On the censored and truncated data, we trained mixture models using the standard EM algorithm and the truncated and censored version of the EM algorithm. We also ran the standard EM algorithm on the set of data points in $\mathcal{Y}_0$, that is, the observations that were not censored.

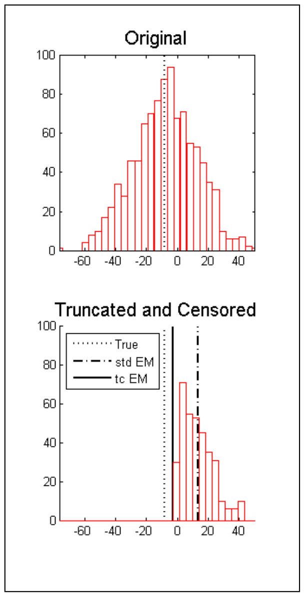
We first investigated three cases of one dimensional data. In cases (a) and (b), 1000 data points were drawn from a Gaussian (a single component mixture) with means 3 and -8, respectively, and with the same standard deviation 20. Values smaller than 0 were discarded (truncation) and those greater than 40 were set to 40 (censoring). Among 1000 data points, the uncensored data points in $\mathcal{Y}_0$ were about 50% in case (a) and 30% in case (b).

Figure 3.1 (a) and (b) show the histograms of each case before and after censoring and truncation. In the figure, the true mean and the estimates are also drawn. As can be seen, the standard EM algorithm always tries to find mean estimates between 0 and 40. Thus, when the true mean is outside this range, the discrepancy between the estimates and the ground-truth can be arbitrarily large. On the other hand, the proposed algorithm finds better estimates. Table 3.1 compares the true parameter values and the estimated parameter values, and numerically supports this observation.

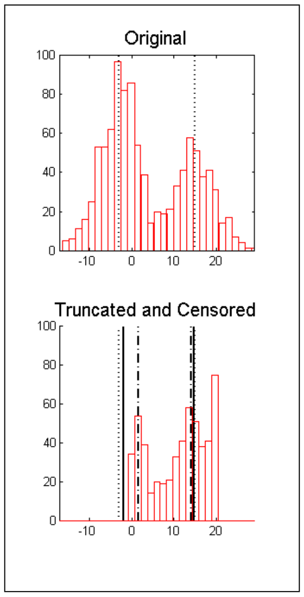
This result is also validated in case (c) in which data points were drawn from a two-component mixture model as illustrated in Figure 3.1 (c). One component with weight 0.6 is centered at -3 and the other component with weight 0.4 is centered at 15 with a common variance 20. The dataset was truncated below at 0 and censored above at 20. Most of the data points from the component on the left were truncated, and nearly 50% of data points were uncensored in each realization. While both algorithms accurately estimated the positive component, the deviations of mean estimates to the true means are evident for the negative component. As shown in Table 3.1, the variance estimates of the proposed method are also much more close to the ground-truth.



(a) A Gaussian at 3



(b) A Gaussian at -8



(c) Two Gaussians

Fig. 3.1: Experiments on 1-dimensional synthetic data. The original data histograms (top) are significantly different from the observed data histograms (bottom) when truncation and censoring occur. All data are truncated at 0 and right-censored at 40 (a),(b) or 20 (c). Dotted lines indicate the true means of each Gaussian component. Solid lines and dash-dot lines are mean estimates from the truncated and censored data EM algorithm and the standard EM algorithm, respectively.

Next we compared the algorithms on multivariate datasets. Two experiments were designed with three-component bivariate Gaussian mixtures. In both cases, an observation (x_1, x_2) was limited to a rectangular window $[0, 25] \times [0, 25]$. In case (a), all three component centers were located within the window ($\pi = (0.5, 0.2, 0.3), \mu_1 = (3, 3), \mu_2 = (13, 3), \mu_3 = (20, 20), \Sigma_1 = diag(20, 5), \Sigma_2 = diag(5, 20), \Sigma_3 = [20, 10; 10, 20]$). On the contrary, in case (b), two centroids were located outside the window ($\pi = (0.5, 0.2, 0.3), \mu_1 = (-3, 3), \mu_2 = (10, -1), \mu_3 = (20, 20), \Sigma_1 = diag(20, 5), \Sigma_2 = diag(5, 20), \Sigma_3 = [20, 10; 10, 20]$). After 1000 data points were drawn in each case, data

	true	standard EM on uncensored	standard EM	truncated and censored EM
1-dim (a)				
μ	3	15.34 ± 0.14	16.80 ± 0.09	2.87 ± 1.31
σ^2	400	105.42 ± 2.18	133.12 ± 1.62	410.65 ± 27.59
1-dim (b)				
μ	-8	12.37 ± 0.13	12.90 ± 0.16	-5.83 ± 1.77
σ^2	400	86.14 ± 2.19	98.65 ± 3.06	352.66 ± 28.84
1-dim (c)				
μ_1	-3	2.08 ± 0.11	2.06 ± 0.10	-1.01 ± 0.36
μ_2	15	13.64 ± 0.11	14.46 ± 0.10	14.89 ± 0.10
σ_1^2	20	2.31 ± 0.23	2.25 ± 0.21	11.68 ± 1.33
σ_2^2	20	14.19 ± 0.69	17.28 ± 0.70	21.79 ± 1.10
π_1	0.6	0.27 ± 0.03	0.24 ± 0.03	0.47 ± 0.09

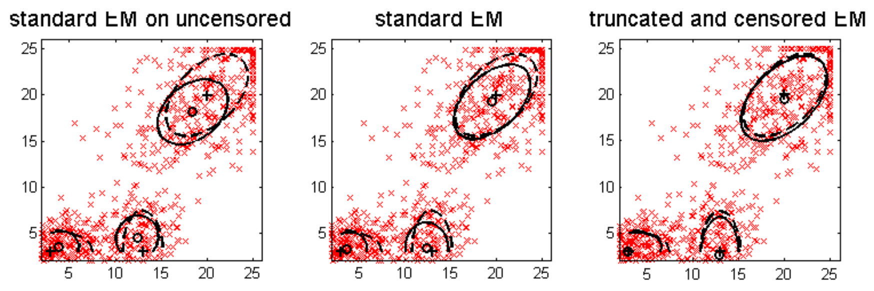
Table 3.1: The true parameters and the estimated parameters are compared for the univariate data experiments. We repeated the experiment ten times, and presented the averages and the standard errors of parameter estimates.

points with $x_1 < 0$ were truncated, and all other values outside the observation window were censored. More than 100 data points were truncated and about 700 data points remained uncensored in case (a), and nearly 400 data points were truncated and about 400 data points remained uncensored in case (b). The data points after truncation and censoring are depicted in Figure 3.2. In the figure, level contours are displayed to compare the estimated distributions to the true distribution. The figure also shows the results when the standard EM algorithm is applied on a subset after the censored data points are excluded from the dataset (standard EM on uncensored). The differences between algorithms are most conspicuous in case (b) where the estimates from the truncated and censored data EM algorithm significantly outperform the estimates from the standard EM algorithms.

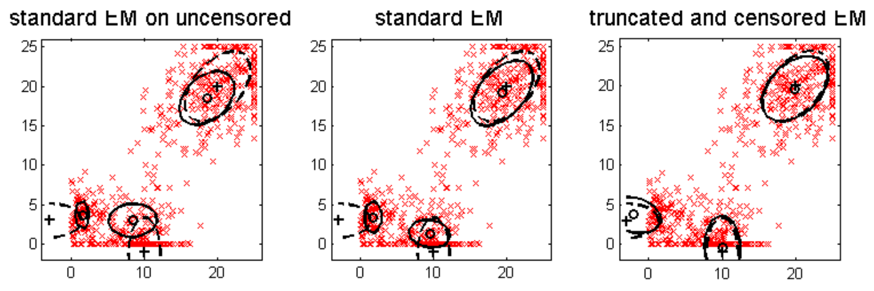
To quantitatively evaluate model estimates, we computed Kullback-Leibler (KL) divergences

$$KL(p \,||\, q) = \mathbb{E}_p \left[\log p - \log q\right] \approx \frac{1}{N_e} \sum_{n=1}^{N_e} \left[\log p\left(\mathbf{x}^n\right) - \log q\left(\mathbf{x}^n\right)\right]$$

between the known true distribution p and estimated distribution q, where the expectation is approximated by a sample mean over $N_e = 10,000$ data points drawn from p. The KL



(a) Three Gaussian components with centroids at $(3, 3)$, $(13, 3)$ and $(20, 20)$



(b) Three Gaussian components with centroids at $(-3, 3)$, $(10, -1)$ and $(20, 20)$

Fig. 3.2: Experiments on 2-dimensional synthetic data. The solid ellipses and 'o's are level-curves and centroids of each component estimates. The dashed ellipses and '+'s are for true mixture components. Small crosses represent data points in the truncated and censored data. x_2 is censored at 0 and 25, and x_1 is truncated at 0 and censored at 25.

	standard EM on uncensored	standard EM	truncated and censored EM
1-dim (a)	1.46 ± 0.03	1.15 ± 0.02	0.02 ± 0.01
1-dim (b)	3.48 ± 0.09	3.07 ± 0.09	0.07 ± 0.03
1-dim (c)	3.82 ± 0.07	3.56 ± 0.07	0.35 ± 0.17
2-dim (a)	0.37 ± 0.02	0.50 ± 0.02	0.03 ± 0.01
2-dim (b)	4.05 ± 0.36	4.21 ± 0.36	0.59 ± 0.20

Table 3.2: The KL divergences between the true densities and the estimated densities are computed for each synthetic dataset. The averages and standard errors across ten samples are reported in the table. The proposed algorithm outperforms the standard EM algorithm.

divergence is non-negative and equals to zero if and only if the estimated distribution is the same as the true distribution. We repeated experiments on the ten different samples and averaged the resulting KL divergences. The computed KL divergences are reported in Table 3.2. For all the investigated datasets, the estimated distributions from the proposed method show significantly smaller KL divergences. Therefore, the truncated and censored data EM algorithm successfully corrects the biases that exist in the standard EM algorithm.

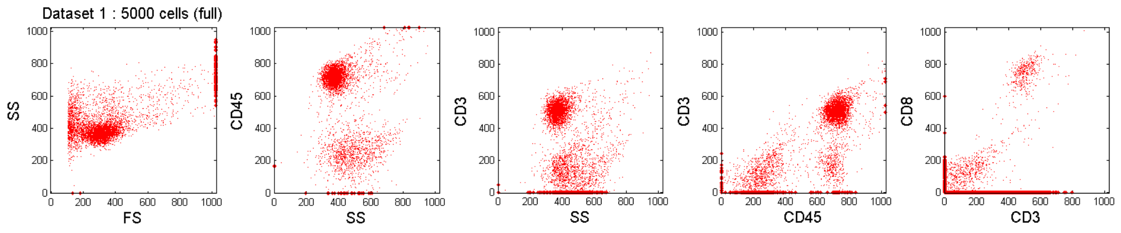
3.7.2 Application to Flow Cytometry Data

We now discuss a real world application. As explained earlier, this work is motivated by flow cytometry analysis. A flow cytometer measures multiple antigen-based markers associated with cells in a cell population.

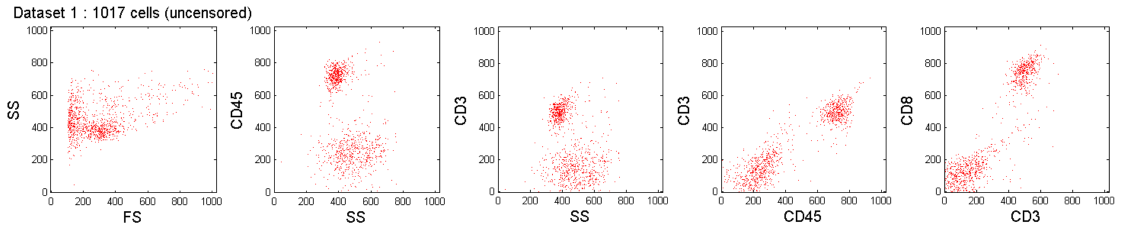
In practice, clinicians usually rely on rudimentary tools to analyze flow cytometry data. They select a subset of one, two or three markers and diagnose by visually inspecting the one dimensional histograms or two or three dimensional scatter plots. To facilitate the analysis, the clinicians often select and exclude cell subpopulations. This process is called "gating" and usually performed manually by thresholding and drawing boundaries on the scatter plots. It is labor-intensive and time-consuming, and limits productivity. Moreover, the results of gating vary by user experience, and replicating the same results by others is difficult.

These difficulties have recently motivated interest in automatic and systematic gating methods. Although standard techniques have not been established yet, mathematical modeling of cell populations with mixture models is favored by many researchers due to the possibility of direct analysis in multi-dimensional spaces. In [8, 15], Gaussian mixtures are used to model cell populations. The use of a mixture of t-distributions combined with a Box-Cox transformation is studied in [44]. A more recent study reported successful applications of a mixture of skew normal distributions or skew t-distributions in [53]. The domain knowledge of field experts is sometimes incorporated in the mixture model [38]. However, while truncation and censoring are present in flow cytometry data, we note that these issues have not been explicitly addressed in the literature.

Here we present the analysis of two flow cytometry datasets. These datasets were provided by the Department of Pathology at the University of Michigan. Each cell contains five marker readings. The markers in the first dataset are FS, SS, CD3, CD8 and CD45. These are intended for finding T-cells, a type of white blood cells, in the blood sample. The second dataset includes FS, SS, CD20, CD5 and CD45, and these markers are for identifying B-lymphocytes. The forward scatter (FS) threshold is set at approximately 100, and cells with low FS values are truncated from these datasets. Each dataset also underwent censoring so that it includes no marker values out of the range from 0 to 1023. The censoring was severe in these datasets, and only 20% and 40% of total observed cells are uncensored as can be seen in the scatter plots in Fig. 3.3 and Fig. 3.4. Furthermore, for each dataset, the distribution of all cells is significantly different from the distribution of the uncensored cells. When we exclude censored cells, the cluster of CD45$^+$ CD3$^-$ cells are lost in the first dataset (Fig. 3.3 fourth column) and the cluster of CD45$^+$ CD20$^+$ cells are lost in the second dataset (Fig. 3.4 fourth column). Thus, analysis based exclusively on the uncensored cell population can be misleading.

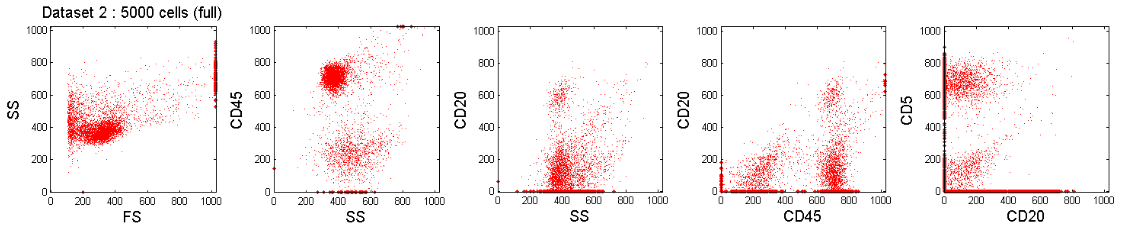


(a) Scatter plots of all observed cells

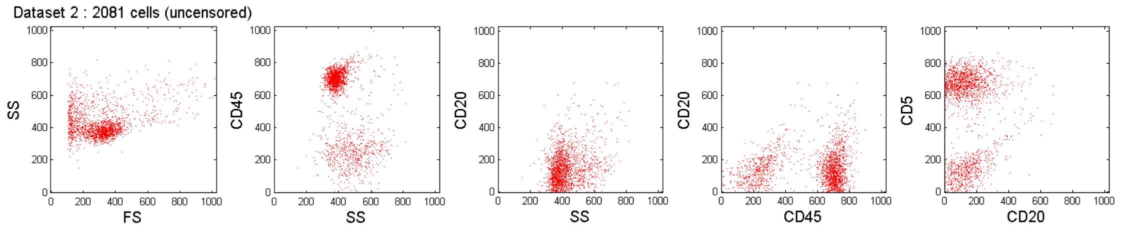


(b) Scatter plots of uncensored cells

Fig. 3.3: The first flow cytometry dataset has markers FS, SS, CD3, CD8 and CD45. These markers are chosen to investigate the T-cells in the blood sample of a patient. Only 20% of cells are uncensored. The CD45^{+} CD3^{-} subpopulation is missing in (b).



(a) Scatter plots of all observed cells



(b) Scatter plots of uncensored cells

Fig. 3.4: The second dataset includes FS, SS, CD20, CD5 and CD45 for B-cell population. The uncensored cells are 40% of the total observed cells. Scatter plots show that the CD45^{+} CD20^{+} cells are missing among the uncensored cells.

We modeled the cell population with a Gaussian mixture and fitted the model to the observed 5000 cells using the standard EM algorithm and the truncated and censored version of EM algorithm. We chose a six-component model ($K = 6$) since, from these datasets, we expect to find several types of cells such as lymphocytes, which mostly consist of T-cells and B-cells, lymphoblasts, and small populations of granulocytes and monocytes. We treated each cell as a point in 5-dimensional space. The k-means algorithm is first performed to initialize each EM algorithm. The convergence is determined when the relative change of the log-likelihood is less than 10^{-10} or the number of iteration reaches 500. Fig. 3.5 shows the evolution of the log-likelihood of the truncated and censored data EM on the flow cytometry datasets. The value increases sharply in the first dozens of steps and then converges. The average times per iteration were 0.01 seconds for the standard EM and 2.50 seconds for the truncated and censored data EM under Windows 7 system equipped with two Intel(R) Xeon(R) 2.27 GHz processors and RAM 12 GB. We repeated this process with 10 different starting points based on different runs of the k-means algorithm, and presented the results that achieved the highest log-likelihood.

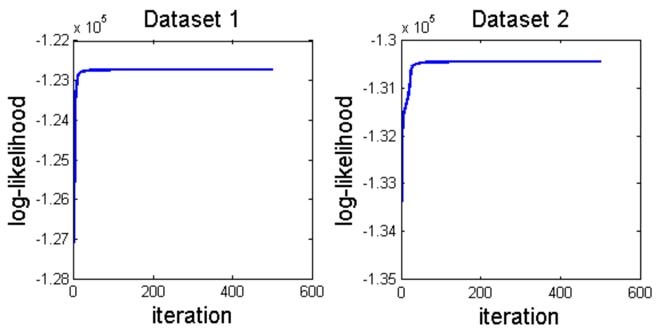
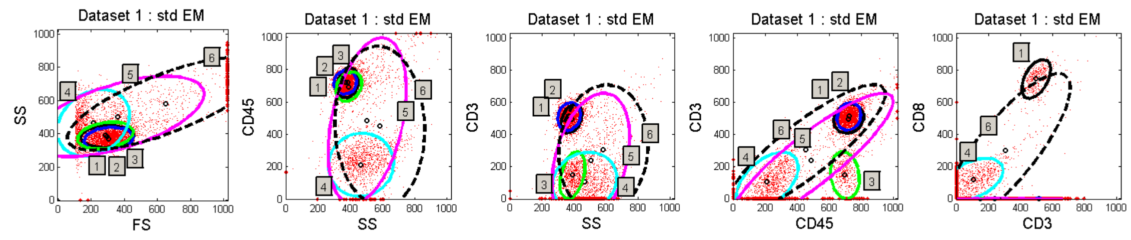
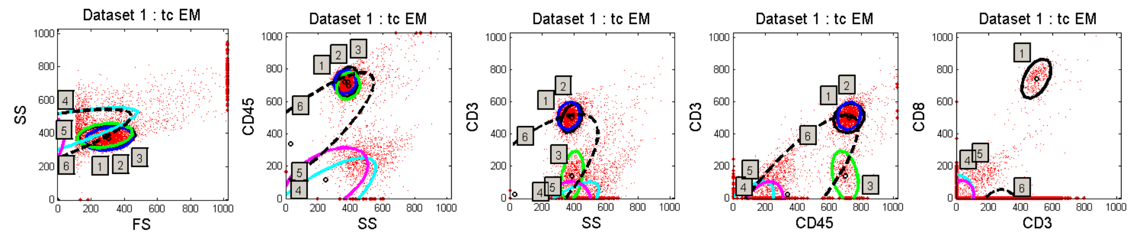


Fig. 3.5: The truncated and censored data EM algorithm terminates when the log-likelihood value changes less than a predefined constant or the number of iteration reaches 500.

The mixture models fitted by the standard EM and the truncated and censored data EM are shown in Fig. 3.6 and Fig. 3.7. In the first dataset, both algorithms generated similar estimates of lymphocyte populations (component 1, 2, and 3), which are the primary interest in the flow cytometry data analysis. On the other hand, the results for lymphoblasts are different (component 4 in Fig. 3.6(a) and component 4, 5 in Fig. 3.6(b)). Because a large number of lymphoblasts were truncated or censored, the component centers from the truncated and censored data EM algorithm were located outside the observation window. In the second flow cytometry dataset, the key difference is that the standard EM failed to find the B-lymphocytes (CD45$^+$ CD20$^+$) while component 3 in Fig. 3.7(b) clearly identified the B-cells. The truncated and censored data EM also estimated that the centers of component 1 and 2 have negative CD20 values because a large amount of CD45$^+$ CD20$^-$ lymphocytes were censored.

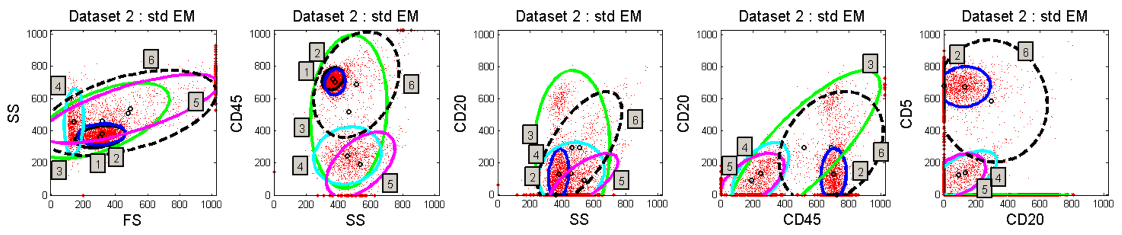


(a) Standard EM algorithm

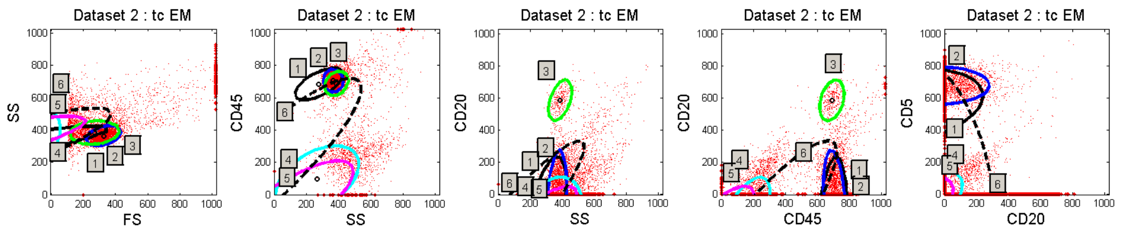


(b) Truncated and censored EM algorithm

Fig. 3.6: For the first flow cytometry dataset, the mixture model fits using the standard EM algorithm and the truncated and censored EM algorithm are shown. The level contour and centroid 'o' of each component are indicated and labeled. Lymphocyte populations (component 1, 2, and 3) were found well by both algorithms.



(a) Standard EM algorithm



(b) Truncated and censored EM algorithm

Fig. 3.7: The results for the second flow cytometry dataset are displayed. While the standard EM result failed to find the CD45$^+$ CD20$^+$ B-lymphocytes, the truncated and censored EM found this cell population (component 3 in (b)).

3.8 Discussion

In this chapter, we addressed the problem of fitting multivariate Gaussian mixture models to truncated and censored data. We presented EM algorithms and showed that their computation can be achieved using the properties of truncated multivariate normal distributions. Simulation results on synthetic datasets showed that the proposed algorithm corrects for the biases caused by truncation and censoring, and significantly outperforms the standard EM algorithm. We also applied the truncated and censored data EM algorithm to automatic gating of flow cytometry data and compared the gating results to the standard EM algorithm. Our results suggest that the proposed algorithm can be effective in identifying clinically important cell populations in flow cytometry data analysis.

Although these algorithms can be readily applied to lower dimensional data, they depend on methods for evaluating a multivariate normal cumulative distribution function, and the algorithms can run slower as the dimension increases. However, ever-growing computing power will lower the hindrance to using these algorithms in the future.

Several criteria such as the Akaike information criterion (AIC) and Bayesian information criterion (BIC) have been proposed to select the number of components for finite mixture models. However, they are based on complete measurements, and it is not straightforward to use them for censored and truncated data. We envision that extending these model selection criteria to the truncated and censored data is an interesting problem for future work.

Another future direction is developing stochastic versions of the truncated and censored data EM algorithm. The stochastic EM (SEM) algorithms are known to be less susceptible to the initialization. While [17] proposed SEM algorithms for censored data, his focus was on univariate data. Therefore, the SEM for multivariate mixture models would be an interesting next step.

3.A Appendix: Truncated Multivariate Normal

We consider here some key properties of truncated multivariate normal distributions used in this paper. The first two moments are derived and the derivatives of normal integrals are related to the moments.

3.A.1 First and Second Moments

Tallis [67] derived the moment generating function of a standardized and truncated normal distribution. Then he derived the first and the second moments from the moment generating function. Here we extend his approach to a normal distribution with arbitrary mean and covariance that is truncated above and below, and show that we can simplify the computation of the first two moments. We note that a similar derivation appeared in [46].

Let $\mathbf{X} \in \mathbb{R}^d$ be normally distributed with a probability density function $\phi_d(\mathbf{x}; \mathbf{0}, \Sigma)$

where

$$\phi_d(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right).$$

We consider the nonzero mean case later in this section. Suppose a truncation of $\mathbf{X}$ below at $\mathbf{a}$ and above at $\mathbf{b}$ and denote

$$\alpha = P(\mathbf{a} \le \mathbf{X} \le \mathbf{b}) = \int_{\mathbf{a}}^{\mathbf{b}} \phi_d(\mathbf{x}; \mathbf{0}, \Sigma)\, d\mathbf{x} = \Phi_d(\mathbf{a}, \mathbf{b}; \mathbf{0}, \Sigma)$$

where the inequality is component-wise and $\Phi_d(\mathbf{a}, \mathbf{b}; \mathbf{0}, \Sigma)$ denotes the normal integration over the rectangle with vertices $\mathbf{a}$ and $\mathbf{b}$. Then the moment generating function is

$$m(\mathbf{t}) = \frac{1}{\alpha} \int_{\mathbf{a}}^{\mathbf{b}} \exp(\mathbf{t}^T\mathbf{x})\, \phi_d(\mathbf{x}; \mathbf{0}, \Sigma)\, d\mathbf{x} = \frac{\exp(\frac{1}{2}\mathbf{t}^T\Sigma\mathbf{t})}{\alpha} \int_{\mathbf{a}-\Sigma\mathbf{t}}^{\mathbf{b}-\Sigma\mathbf{t}} \phi_d(\mathbf{x}; \mathbf{0}, \Sigma)\, d\mathbf{x}. \quad (3.30)$$

We can find the first moment and the second moment from (3.30). We first differentiate (3.30) with respect to t_i and evaluate at $\mathbf{t} = 0$. Then

$$\alpha \left.\frac{\partial m(\mathbf{t})}{\partial t_i}\right|_{\mathbf{t}=0} = \alpha\, \mathbb{E}[X_i] = \sum_{k=1}^{d} \sigma_{i,k}(F_k(a_k) - F_k(b_k)) \quad (3.31)$$

where $\sigma_{i,k} = [\Sigma]_{i,k}$ and

$$\begin{aligned} F_k(x) =& \int_{\mathbf{a}_{-k}}^{\mathbf{b}_{-k}} \phi_d(x, \mathbf{x}_{-k}; \mathbf{0}, \Sigma)\, d\mathbf{x}_{-k} \\ =& \phi_1(x; 0, \sigma_{k,k}) \int_{\mathbf{a}_{-k}}^{\mathbf{b}_{-k}} \phi_{d-1}(\mathbf{x}_{-k}; \boldsymbol{\mu}_{-k|k}(x),\, \Sigma_{-k|k})\, d\mathbf{x}_{-k} \\ =& \phi_1(x; 0, \sigma_{k,k}) \Phi_{d-1}(\mathbf{a}_{-k}, \mathbf{b}_{-k}; \boldsymbol{\mu}_{-k|k}(x),\, \Sigma_{-k|k}). \end{aligned}$$

Here we used $-k$ to denote the set of elements $\{1, \cdots, (k-1), (k+1), \cdots, d\}$ other than the kth. The conditional mean and covariance are $\boldsymbol{\mu}_{-k|k}(x) = \Sigma_{-k,k}\Sigma_{k,k}^{-1} x$ and $\Sigma_{-k|k} = \Sigma_{-k,-k} - \Sigma_{-k,k}\Sigma_{k,k}^{-1}\Sigma_{k,-k}$.

Taking the derivatives of (3.30) with respect t_i and t_j at $\mathbf{t} = 0$ gives the second moment

$$\begin{aligned}\alpha \left.\frac{\partial^2 m(\mathbf{t})}{\partial t_i \partial t_i}\right|_{\mathbf{t}=0} =& \alpha\,\mathbb{E}[X_i X_j] \\ =& \alpha\,\sigma_{i,j} + \sum_{k=1}^{d} \frac{\sigma_{i,k}\,\sigma_{j,k}}{\sigma_{k,k}} \big(a_k\,F_k(a_k) - b_k\,F_k(b_k)\big) \\ &+ \sum_{k=1}^{d} \sigma_{i,k} \sum_{q \neq k} \left(\sigma_{j,q} - \frac{\sigma_{k,q}\sigma_{j,k}}{\sigma_{k,k}}\right) \Big[F_{k,q}(a_k, a_q) + F_{k,q}(b_k, b_q) \\ &- F_{k,q}(a_k, b_q) - F_{k,q}(b_k, a_q)\Big] \end{aligned} \tag{3.32}$$

where

$$\begin{aligned} F_{k,q}(x_k, x_q) =& \int_{\mathbf{a}_{-(k,q)}}^{\mathbf{b}_{-(k,q)}} \phi_d(x_k, x_q, \mathbf{x}_{-(k,q)}; \mathbf{0}, \Sigma)\, d\mathbf{x}_{-(k,q)} \\ =& \phi_2(x_k, x_q, ; \mathbf{0}, \Sigma_{(k,q),(k,q)}) \\ & \cdot \int_{\mathbf{a}_{-(k,q)}}^{\mathbf{b}_{-(k,q)}} \phi_{d-2}(\mathbf{x}_{-(k,q)}\,;\, \boldsymbol{\mu}_{-(k,q)|(k,q)}(x_k, x_q),\, \Sigma_{-(k,q)|(k,q)})\, d\mathbf{x}_{-(k,q)} \\ =& \phi_2(x_k, x_q, ; \mathbf{0}, \Sigma_{(k,q),(k,q)}) \\ & \cdot \Phi_{d-2}(\mathbf{a}_{-(k,q)}, \mathbf{b}_{-(k,q)}\,;\, \boldsymbol{\mu}_{-(k,q)|(k,q)}(x_k, x_q),\, \Sigma_{-(k,q)|(k,q)}). \end{aligned}$$

Likewise, $-(k, q)$ indicates the set of elements except the kth and qth elements. The conditional mean $\boldsymbol{\mu}_{-(k,q)|(k,q)}(x_k, x_q)$ and covariance $\Sigma_{-(k,q)|(k,q)}$ are also defined similarly.

Therefore, we can compute the first moment (3.31) and the second moment (3.32) from a density function and a normal integration, which can be evaluated from the cumulative distribution function and are available in many statistical toolboxes (for example, FORTRAN, R or MATLAB). In particular, `mvncdf` function in MATLAB 7.9.0 evaluates the multivariate cumulative probability using the methods developed by Drezner and Wesolowsky [22] and by Genz [28] for bivariate and trivariate Gaussian. For four or more dimensional Gaussians, it uses a quasi-Monte Carlo integration algorithm developed by Genz and Bretz [29, 30].

Note that $\frac{F_k(x)}{\alpha}$ and $\frac{F_{k,q}(x_k, x_q)}{\alpha}$ are univariate and bivariate marginals of X_k and (X_k, X_q).

Now consider a normal distribution $\phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)$ truncated at $\mathbf{a}^*$ and $\mathbf{b}^*$. Then

$$\begin{aligned}
\mathcal{M}^1(\boldsymbol{\mu}, \Sigma\,;\, [\mathbf{a}^*, \mathbf{b}^*]) :=& \mathbb{E}[\mathbf{Y}] && (3.33)\\
=& \mathbb{E}[\mathbf{X}] + \boldsymbol{\mu},\\
\mathcal{M}^2(\boldsymbol{\mu}, \Sigma\,;\, [\mathbf{a}^*, \mathbf{b}^*]) :=& \mathbb{E}[\mathbf{Y}\mathbf{Y}^T] && (3.34)\\
=& \mathbb{E}[\mathbf{X}\mathbf{X}^T] + \boldsymbol{\mu}\, \mathbb{E}[\mathbf{X}]^T + \mathbb{E}[\mathbf{X}]\, \boldsymbol{\mu}^T + \boldsymbol{\mu}\boldsymbol{\mu}^T\\
=& \mathbb{E}[\mathbf{Y}]\, \mathbb{E}[\mathbf{Y}]^T + \mathbb{E}[\mathbf{X}\mathbf{X}^T] - \mathbb{E}[\mathbf{X}]\, \mathbb{E}[\mathbf{X}]^T
\end{aligned}$$

where $\mathbb{E}[X_i]$ and $\mathbb{E}[X_i X_j]$ are evaluated at $\mathbf{a} = \mathbf{a}^* - \boldsymbol{\mu}$ and $\mathbf{b} = \mathbf{b}^* - \boldsymbol{\mu}$. In Section 3.3, we introduced the notations $\mathcal{M}^1$ and $\mathcal{M}^2$ to denote above expectations (3.33) and (3.34).

For example, consider a univariate random variable Y distributed normally with mean μ and variance σ^2. If it is truncated above at 0 (that is, $a^* = -\infty, b^* = 0$), then

$$\begin{aligned}
\mathbb{E}[Y] =& \mu - \sigma \frac{\phi_1(-\frac{\mu}{\sigma}; 0, 1)}{\Phi_1(-\frac{\mu}{\sigma}; 0, 1)},\\
\mathbb{E}[Y^2] =& \mu^2 + \sigma^2 - \mu\sigma \frac{\phi_1(-\frac{\mu}{\sigma}; 0, 1)}{\Phi_1(-\frac{\mu}{\sigma}; 0, 1)}
\end{aligned}$$

where the fraction of the standard normal density function ϕ and the distribution function Φ is known as the inverse Mills ratio.

3.A.2 Derivatives

Here we consider the derivatives of $\alpha(\boldsymbol{\mu}, \Sigma) := \int_{\mathbf{a}}^{\mathbf{b}} \phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)\, d\mathbf{y}$ with respect to $\boldsymbol{\mu}$ and Σ used in the derivation in Section 3.3, and relate them with the first and the second moments. Taking the derivative of $\alpha(\boldsymbol{\mu}, \Sigma)$ with respect to $\boldsymbol{\mu}$,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}} \int_{\mathbf{a}}^{\mathbf{b}} \phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)\, d\mathbf{y} =& \int_{\mathbf{a}}^{\mathbf{b}} \Sigma^{-1}(\mathbf{y} - \boldsymbol{\mu})\, \phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)\, d\mathbf{y}\\
=& \Sigma^{-1} \left[\alpha\, \mathcal{M}^1(\boldsymbol{\mu}, \Sigma\,;\, [\mathbf{a}, \mathbf{b}]) - \alpha \boldsymbol{\mu}\right]\\
=& \alpha\, \Sigma^{-1}\, \mathcal{M}^1(\mathbf{0}, \Sigma\,;\, [\mathbf{a} - \boldsymbol{\mu}, \mathbf{b} - \boldsymbol{\mu}])
\end{aligned}$$

where the last equality is from (3.33), so we obtain

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ln \int_{\mathbf{a}}^{\mathbf{b}} \phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)\, d\mathbf{y} = \Sigma^{-1}\, \mathcal{M}^1(\mathbf{0}, \Sigma\,;\, [\mathbf{a} - \boldsymbol{\mu}, \mathbf{b} - \boldsymbol{\mu}]). \tag{3.35}$$

Next if we take the derivative with respect to Σ, we have

$$\begin{aligned}\frac{\partial}{\partial \Sigma} \ln \int_{\mathbf{a}}^{\mathbf{b}} \phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma)\, d\mathbf{y} =& \frac{1}{\alpha} \frac{\partial}{\partial \Sigma} \int_{\mathbf{a}-\boldsymbol{\mu}}^{\mathbf{b}-\boldsymbol{\mu}} \phi_d(\mathbf{y}; \mathbf{0}, \Sigma)\, d\mathbf{y} \\ =& \frac{1}{\alpha} \int_{\mathbf{a}-\boldsymbol{\mu}}^{\mathbf{b}-\boldsymbol{\mu}} \left(-\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}\mathbf{y}\mathbf{y}^T\Sigma^{-1} \right) \phi_d(\mathbf{y}; \mathbf{0}, \Sigma)\, d\mathbf{y} \\ =& -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}\, \mathcal{M}^2(\mathbf{0}, \Sigma\,;\, [\mathbf{a} - \boldsymbol{\mu}, \mathbf{b} - \boldsymbol{\mu}])\, \Sigma^{-1} \qquad (3.36)\end{aligned}$$

where we used the following facts [45] in the second equality:

$$\frac{\partial}{\partial \Sigma} \mathrm{tr}(\Sigma^{-1} A) = -(\Sigma^{-1} A \Sigma^{-1})^T, \qquad \frac{\partial}{\partial \Sigma} \ln |\Sigma| = (\Sigma^{-1})^T.$$

CHAPTER 4

Transfer Learning for Automatic Gating of Flow Cytometry Data

4.1 Introduction

Flow cytometry is a technique widely used in many clinical and biomedical laboratories for rapid cell analysis [9, 63]. It plays an important role in the diagnosis of diseases such as acute leukemia, chronic lymphoproliferative disorders and malignant lymphomas. While flow cytometry is used in other contexts, our motivation is hematopathology, the study of blood-related diseases.

Mathematically, a flow cytometry dataset can be represented as a collection $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, where i indexes individual cells, and $\mathbf{x}_i$ is a d-dimensional attribute vector recording physical and chemical attributes of the ith cell. The measured attributes include the cell's size, surface complexity, and a variety of features related to the expression levels of different antigens. The number of cells N can range from 10,000 to 1,000,000 (order of magnitude). The dimension d ranges from 7-12 in most clinical applications and 15-20 in some research applications. In clinical settings, each dataset corresponds to a particular patient, where the cells are typically drawn from a peripheral blood, lymph node, or bone marrow sample.

To make a diagnosis, a pathologist will use a computer to visualize different two-dimensional scatter plots of a flow cytometry dataset. Figure 4.1 shows examples of such scatter plots for two different patients. The attributes in these datasets are denoted here only by their abbreviations (FS, SS, CD45, etc.) for simplicity. These plots illustrate the presence of several clusters of cells within each dataset. They also illustrate the variation of measured data from one patient to another. This variation arises from both biological (e.g., health condition) and technical (e.g., instrument calibration) sources.

Depending on the illness being screened for, the pathologist will typically visualize one type of cell in particular. For example, in the diagnosis of leukemias, lymphocytes are known to be relevant. By visualizing different two-dimensional scatter plots, the pathologist makes a diagnosis based on the shape, range, and other distributional characteristics of these cells. Therefore, a necessary preprocessing step is to label every cell as belonging to the cell type of interest or not, a process known as "gating." This amounts to the assignment of binary labels $y_i \in \{-1, 1\}, i = 1, \ldots, N$, to every cell. In Figure 4.1, the lymphocytes are indicated by an alternate color/shading. Without gating, cells of other types will overlap with the targeted cell type in the two-dimensional scatter plots used for diagnosis. After gating, only the cells of interest are visualized.

This chapter concerns the automation of gating. Unfortunately, in clinical settings gating is still performed manually. It is a labor-intensive task in which a pathologist or technologist visualizes the data from different two-dimensional scatter plots, and uses special software to draw a series of boundaries ("gates"), each of which eliminates a portion of the cells outside of the desired type. In many clinical settings, several "gating" decisions, all currently manual, are required to "purify" a population of interest. For example, one might first draw a series of bounding line segments on pairwise plots of FS, SS and CD45 to select all "cells," which eliminates spurious attribute vectors arising from dead cells, air bubbles, etc. Next, gating on (restricting to) cells, the same three attributes can be used to identify lymphocytes. Finally, a CD20 vs. CD10 plot, gated on lymphocytes, may be used to select B-cell lymphocytes for further analysis. Unfortunately, because of the aforementioned variation in data, this process is difficult to quantify in terms of a small set of simple rules. Instead, the person performing gating must utilize specialized domain knowledge together with iterative refinement. Since modern clinical laboratories can see dozens of cases per day, it would be highly desirable to automate this process.

Because of this need, several researchers have tackled the auto-gating problem. This is evidenced by a recent survey on flow cytometry analysis methods, which revealed that more than 70% of studies focus on automated gating techniques [6]. The vast majority of approaches rely on a clustering/mixture modeling, using a parametric representation for each cell type, together with unsupervised learning (usually an EM algorithm) to fit the model parameters [8, 15, 44, 53]. The mixture modeling approach has a number of difficulties, however. One is that the clusters are typically not elliptical, meaning complex parametric models must be employed, such as skewed Gaussians, leading to challenging inference problems. Another limitation is that clustering is a harder problem than the one that needs to be solved, in the same sense that determining a decision boundary is easier than estimating a density. Finally, these algorithms are unsupervised, and do not fully leverage expert knowledge.

We propose to view auto-gating as a kind of transfer learning problem. In particular, we assume that a collection of expert-gated datasets are available. Although different datasets have different distributions, there is enough similarity, (e.g., lymphocytes show low levels of SS while expressing high levels of CD45) that this expert knowledge can be transferred to the new dataset. Our basic approach is to train a classifier on each expert-gated dataset, and to summarize these classifiers to form a baseline classifier. This baseline is then adapted to the new dataset by optimizing a "low-density separation" criterion.

The transfer learning problem we study is, to our knowledge, a new one, although it has similarities to previously studied transfer learning problems, as well as multi-task learning. These connections are reviewed below.

The rest of the chapter is structured as follows. Section 4.2 gives the problem statement. Related work is described in Section 4.3. Our methodology is described in Section 4.4, and in Section 4.5 we describe the application of our methods to the gating of lymphocytes in peripheral blood samples. Some concluding remarks are given in Section 4.6.

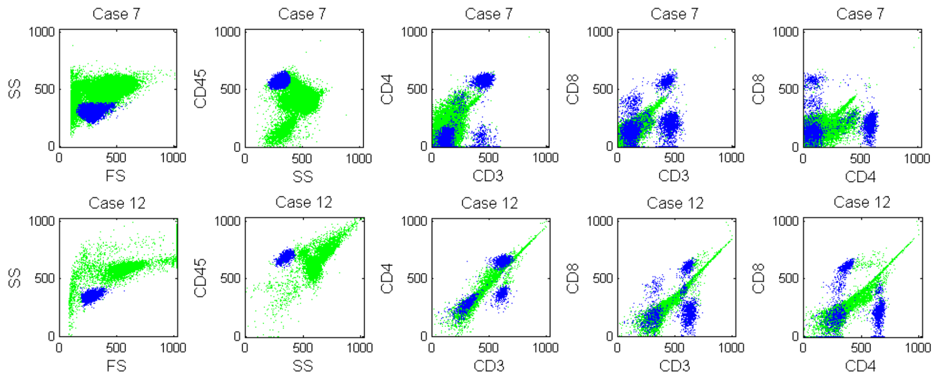


Fig. 4.1: Clinicians analyze flow cytometry data using a series of scatter plots on pairs of attributes. The distribution of cell populations differs from patient to patient, and changes after treatments. Lymphocytes, a type of white blood cell, are marked dark (blue) and others are marked bright (green). These were manually selected by a domain expert.

4.2 Problem Setup

There are M labeled datasets $\mathcal{D}_m = \{(\mathbf{x}_{m,i}, y_{m,i})\}_{i=1}^{N_m}$, $m = 1, \cdots, M$, each a random sample from a distribution $\mathcal{P}_m$. $\mathcal{D}_m$ corresponds to the mth flow cytometry dataset and its labels are determined by experts. There is also an unlabeled dataset $\mathcal{T} = \{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$, a random sample from a new distribution $\mathcal{P}_t$ corresponding to a new flow cytometry dataset. The labels $\{y_{t,i}\}_{i=1}^{N_t}$ are not observed. The goal is to assign labels $\{\widehat{y}_{t,i}\}_{i=1}^{N_t}$ to $\mathcal{T}$ so that the misclassification rate is minimized. All the distributions are different, but defined on the same space $\mathbb{R}^d \times \{-1, +1\}$.

While there are obvious connections to transfer learning (discussed in the next section), this problem might also be described as a supervised learning problem called *set prediction*. Whereas in traditional supervised learning, the inputs are vectors and the outputs are scalar labels, here the inputs are distributions (random samples), and the outputs are subsets of Euclidean space (subsamples). Given several examples of distributions and corresponding subsets, the goal is to generalize to predict the subsets associated with previously unseen distributions.

4.3 Related Work

As a transfer learning problem, our problem is characterized by having multiple source domains (the expert-gated datasets), and a single target domain (the unlabeled dataset). In addition, using the taxonomy of Pan and Yang [50], our setting can be described as follows:

(1) the source and target domains are different, because the marginal distributions of $\mathbf{x}$ are different,

(2) the source and target tasks are different, because each dataset requires a different gating,

(3) there are no labeled examples in the target domain.

To the best of our knowledge, previous work has not addressed this combination of characteristics. Many previous works fall under the heading of *inductive transfer learning*, where at least a few labels are given for the target domain [2, 55]. In *transductive transfer learning*, and the related problems of sample selection bias and covariate shift, the source and target tasks are assumed to be the same [3].

Another closely related area is multi-task learning [14,23,69]. However, our problem contrasts to this line of studies in the sense that our ultimate goal is achieving high performance for the target task only, and not the source tasks. It would be natural, however, to extend our problem to the case where there are multiple unlabeled target datasets, which might be called “transductive set prediction.”

Toedling et al. [72] explore using support vector machines (SVMs) for flow cytometry datasets from multiple patients. They form a single large dataset by merging all the source datasets, and train a classifier on this dataset. However, due to its large size of the combined dataset, the training requires demanding computational and memory resources. Furthermore, this approach ignores the variability among multiple datasets and treats all the datasets as arising from the same distribution. This reduces the problem to standard single-task supervised learning.

4.4 Algorithm

We describe our approach to the problem. In this section, we show how our algorithm summarizes knowledge from the source data and adapts it to the new task.

4.4.1 Baseline Classifier for Summarizing Expert Knowledge

We suppose that the knowledge contained in the source tasks can be represented by a set of decision functions $f_1, \cdots, f_M$. The sign of a decision function f_m provides a class prediction of a data point $\mathbf{x}$: $\widehat{y} = sign(f_m(\mathbf{x}))$. Each f_m is separately learned from each of the M source datasets. Then these decision functions form the pool of knowledge.

In the present work, we consider linear decision functions of the form $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. This decision function defines a hyperplane $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ with a normal vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$.

Algorithm 4.1 Baseline Classifier

```
Input: source task data $\mathcal{D}_m$ for $m = 1, \cdots, M$, regularization parameters $\{C_m\}_{m=1}^M$
1: for $m = 1$ to $M$ do
2:     $(\mathbf{w}_m, b_m) \leftarrow SVM(\mathcal{D}_m,\ C_m)$
3: end for
4: Robust Mean:
       $(\mathbf{w}_0, b_0) \leftarrow Algorithm\ 4.2\,(\{(\mathbf{w}_m, b_m)\}_m)$
Output: $(\mathbf{w}_0, b_0)$ or $f_0(\mathbf{x}) = \langle \mathbf{w}_0, \mathbf{x} \rangle + b_0$
```

The SVM is among the most widely used methods for learning a linear classifier [58]. It finds a separating hyperplane based on the maximal margin principle. We use the SVM to fit a decision function f_m or a hyperplane $(\mathbf{w}_m, b_m)$ to the mth source dataset $\mathcal{D}_m$. Training of a SVM needs a regularization parameter C_m that can be set individually for each source task.

We devise a baseline classifier $f_0 = \langle \mathbf{w}_0, \mathbf{x} \rangle + b_0$, where $(\mathbf{w}_0, b_0)$ is the mean of $(\mathbf{w}_m, b_m)$. Instead of the simple mean, Algorithm 4.1 uses a robust mean to prevent f_0 from being unduly influenced by atypical variations among datasets. Algorithm 4.2 presents the robust estimator as formulated in [11]. Here ψ is a weight function corresponding to a robust loss, and we use the Huber loss function. Note that we also robustly estimate the covariance of the $\mathbf{w}_m$, which is used below in Section 4.4.2.

The learning of f_0 does not involve $\mathcal{T}$ at all. Thus, it is not expected to provide a good prediction for the target task. Next we describe a way to adapt this baseline classifier to the target data based on the low-density separation principle.

4.4.2 Transferring Knowledge to Target Task

Low-density separation is a concept used extensively in machine learning. This notion forms the basis of many algorithms in clustering analysis, semi-supervised classification, novelty detection and transductive learning. The underlying intuition is that the decision

Algorithm 4.2 Robust Mean and Covariance

```
Input: $(\mathbf{w}_m, b_m)$ for $m = 1, \cdots, M$
1: Concatenate: $\mathbf{u}_m \leftarrow [\mathbf{w}_m, b_m], \quad \forall m$
2: Initialize: $\boldsymbol{\mu} \leftarrow mean(\mathbf{u}_m), \quad \mathbf{C} \leftarrow cov(\mathbf{u}_m)$
3: repeat
4:     $d_m \leftarrow \left((\mathbf{u}_m - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{u}_m - \boldsymbol{\mu})\right)^{1/2}$
5:     $w_m \leftarrow \psi(d_m)/d_m$
6:     Update: $\boldsymbol{\mu}^{new} \leftarrow \frac{\sum_m w_m \mathbf{u}_m}{\sum_m w_m}$
               $\mathbf{C}^{new} \leftarrow \frac{\sum_m w_m^2 (\mathbf{u}_m - \boldsymbol{\mu}^{new})(\mathbf{u}_m - \boldsymbol{\mu}^{new})^T}{\sum_m w_m^2 - 1}$
7: until Stopping conditions are satisfied
Output: $\boldsymbol{\mu} = [\mathbf{w}_0, b_0], \quad \mathbf{C}_0 = \mathbf{C}(1:d, 1:d)$
```

Algorithm 4.3 Shift Compensation

```
Input: hyperplane $(\mathbf{w}, b)$, source task data $\{\mathcal{D}_m\}_{m=1}^M$, target task data $\mathcal{T}$
1: $z_{t,i} \leftarrow \langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b, \quad \forall i$
2: for $m = 1$ to $M$ do
3:     $z_{m,i} \leftarrow \langle \mathbf{w}, \mathbf{x}_{m,i} \rangle + b, \quad \forall i$
4:     $e_m \leftarrow \arg\max_z KDE(z, z_{t,i}) \star KDE(z, z_{m,i})$
5: end for
6: $b \leftarrow b - median(e_m)$
Output: $b$
```

boundaries between clusters or classes should pass through regions where the marginal density of $\mathbf{x}$ is low. Thus, our approach is to adjust the hyperplane parameters so that it passes through a region where the marginal density of $\mathcal{T}$ is low.

Preprocessing

The ranges of each attribute are manually determined by an operator in a way that is specific to each dataset. While operators attempt to minimize this source of technical variation, datasets invariably differ by shifting (and possible scaling) along coordinate axis. Rather than aligning all datasets via some global d-dimensional shift/scale transformation, it is sufficient for our purposes to align datasets in the direction of the baseline normal

Algorithm 4.4 Bias Update

```
Input: hyperplane $(\mathbf{w}, b)$, target task data $\mathcal{T}$
 1: Compute:        $z_i \leftarrow \langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b, \quad \forall i$
 2: Build a Grid:   $s_i \leftarrow sort\ (z_i)$
 3: for $j = 1$ to $N_t$ do
 4:     $c_j \leftarrow \sum_i \mathbf{1}_{\{\frac{|z_i - s_j|}{\|\mathbf{w}\|} < 1\}}$
 5: end for
 6: $h \leftarrow kernel\ bandwidth\ (\ \{(s_j, c_j)\}_j\ )$
 7: Smooth:   $\widehat{p}(z) \leftarrow \sum_j c_j k_h(z, s_j)$
 8: $z^* \leftarrow gradient\ descent\ (\widehat{p}(z),\ 0)$
 9: $b^{new} \leftarrow b - z^*$
Output: $b^{new}$ or $f_b(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b^{new}$
```

vector $\mathbf{w}_0$. Specifically, for each dataset, we compute a kernel density estimate (KDE) of the projection onto $\mathbf{w}_0$. Then, we align the target data to each source dataset using maximum cross-correlation (denoted by $\star$ in Algorithm 4.3), and modify the baseline bias by the median of these shifts. This new bias b will serve as the initial bias when adapting the baseline to $\mathcal{T}$.

Varying Bias

We first describe adapting the bias variable to the unlabeled target data $\mathcal{T}$ based on low-density separation. The process is illustrated in Algorithm 4.4.

To assess whether a linear decision boundary is in a low density region, we count data points near the hyperplane. As the hyperplane moves, this number will be large in a high density region and small in a low density region. In particular, we define a margin, as in SVMs, to be a region of a fixed distance from a hyperplane, say Δ, and count data points within this margin. We use $\Delta = 1$. Given a hyperplane $(\mathbf{w}, b)$, basic linear algebra shows

Algorithm 4.5 Kernel Bandwidth

```
Input: grid points and counts $\{(s_k, c_k)\}$
1: $N \leftarrow \sum_k c_k$
2: $\overline{s} \leftarrow \frac{1}{N} \sum_k s_k c_k$
3: $\widehat{\sigma} \leftarrow \left(\frac{1}{N-1} \sum_k c_k (s_k - \overline{s})^2\right)^{1/2}$
4: $h \leftarrow 0.9 \cdot \widehat{\sigma} \cdot N^{-1/5}$
Output: $h$
```

that $\frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|}$ is the signed distance from $\mathbf{x}$ to the hyperplane. Hence, computing

$$\sum_i \mathbf{1}_{\{\frac{|\langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b|}{\|\mathbf{w}\|} < \Delta\}}$$

over a range of b followed by locating a minimizer near the baseline hyperplane gives the desired solution. Algorithm 4.4 implements this on a grid of biases $\{s_j\}$ and builds $\sum_j c_j \, \delta(z - s_j)$ where δ is the Dirac delta. The grid points and the counts at each grid point are denoted by s_j and c_j.

Before searching for the minimizing bias, we smooth these counts over the grid by convolving with a Gaussian kernel $k_h(z, z') = \frac{1}{\sqrt{2\pi} h} \exp\left(-\frac{|z-z'|^2}{2h^2}\right)$. The bandwidth h controls the smoothness of the kernel. This operation yields a smooth function $\widehat{p}(z) = \sum_j c_j k_h(z, s_j)$. Running a gradient descent algorithm on this smoothed function $\widehat{p}(z)$ returns a local minimum near 0 if initialized at 0 (the second parameter in Line 8).

To facilitate a streamlined process for practical use, we automatically select the kernel bandwidth h as shown in Algorithm 4.5. This kernel choice is motivated from the rule of thumb for kernel density estimation suggested in [64].

Varying Normal Vector

We can also adjust the normal vector of a hyperplane. Given a hyperplane having a normal vector $\mathbf{w}$, we let the updated normal vector be of the form $\mathbf{w}^{new} = \mathbf{w} + a_t \mathbf{v}_t$ where $\mathbf{v}_t$ is the direction of change and a_t is the amount of the change. Thus, the new normal vector is from an affine space spanned by $\mathbf{v}_t$.

Now we explain in detail the ways of choosing $\mathbf{v}_t$ and a_t. We find a direction of change from the robust covariance matrix of the normal vectors $\mathbf{w}_1, \cdots, \mathbf{w}_M$ obtained from Algorithm 4.2. We choose the first principal eigenvector for $\mathbf{v}_t$ after making it orthogonal to $\mathbf{w}_0$, the baseline classifier normal vector, because changes in the direction of $\mathbf{w}_0$ do not affect the decision boundary.

To determine the amount of change a_t, we proceed similarly to the method used to update the bias. We count the number of data points inside the margin as the normal vector varies by a regular increment in the direction of $\mathbf{v}_t$. Filtering with a Gaussian kernel smooths these quantities over the range of variation. Then a gradient descent algorithm can spot a_t that leads to a low density solution near the baseline hyperplane. Algorithm 4.6 summarizes this process.

Even though the presented algorithm here confines the varying direction of normal vector to a single vector $\mathbf{v}_t$, we can generalize this to multiple directions. To do this, more than one eigenvector can be chosen in Step 1 of Algorithm 4.7. Then the Gram-Schmidt process generates a set of orthonormal vectors that spans a subspace for a new normal vector. The counting in-margin points in Algorithm 4.6 can be extended to a multivariate grid with little difficulty.

Putting It All Together

Once the normal vector is updated, we build a new hyperplane by combining it with an updated bias so that the hyperplane accords to a low density region of $\mathcal{T}$. The overall scheme is outlined in Algorithm 4.7. In the algorithm, one can repeatedly update the bias and the normal vector of the hyperplane until stopping conditions are met. A simpler method is fixing the number of iterations. In our experience, running one round of the

Algorithm 4.6 Normal Vector Update

```
Input: hyperplane $(\mathbf{w}, b)$, direction of change $\mathbf{v}_t$, target task data $\mathcal{T}$
1: for $a_k = -0.5$ to $0.5$ step $0.01$ do
2:     $\mathbf{w}_k \leftarrow \mathbf{w} + a_k \mathbf{v}_t$
       $c_k \leftarrow \sum_i \mathbf{1}_{\{\left|\frac{\langle \mathbf{w}_k, \mathbf{x}_{t,i}\rangle + b}{\|\mathbf{w}_k\|}\right| < 1\}}$
3: end for
4: $h \leftarrow kernel\ bandwidth\,(\{(a_k, c_k)\}_k)$
5: Smooth:    $g(a) \leftarrow \sum_k c_k k_h(a, a_k)$
6: $a_t \leftarrow gradient\ descent\ (g(a),\ 0)$
7: $\mathbf{w}^{new} \leftarrow \mathbf{w} + a_t \mathbf{v}_t$
Output: $\mathbf{w}^{new}$
```

loop was sufficient for good solutions.

4.5 Experiments

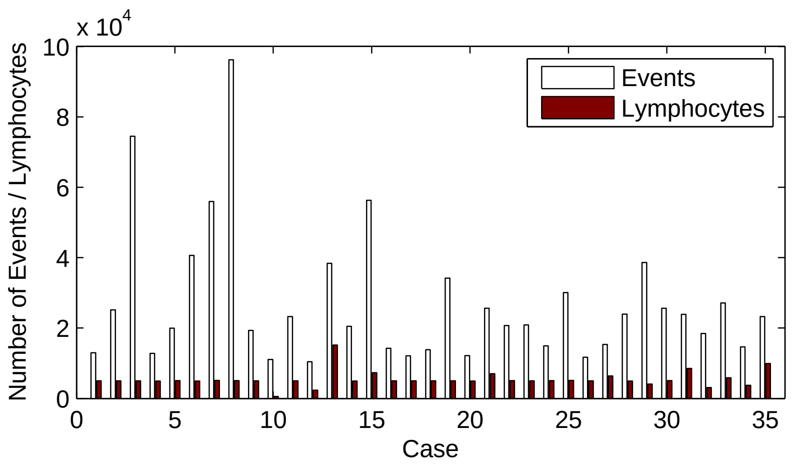


Fig. 4.2: Number of total events and lymphocytes in each of the flow cytometry datasets.

We demonstrate the proposed methods on clinical flow cytometry datasets. Specifically, we apply them to the problem of detecting lymphocytes from peripheral blood samples. Lymphocytes are kinds of white blood cells and play a major role in the human immune system. In diagnosing diseases such as leukemias, identifying these cells is the first step in most clinical analysis.

Algorithm 4.7 Set Estimation based on Low-Density Separation

Input: source task data $\{\mathcal{D}_m\}_{m=1}^M$, target task data $\mathcal{T}$, regularization parameters $\{C_m\}_{m=1}^M$

```
 1: for $m = 1$ to $M$ do
 2:     $(\mathbf{w}_m, b_m) \leftarrow SVM(\mathcal{D}_m, C_m)$
 3: end for
 4: Initialize:
        $((\mathbf{w}_0, b_0), \mathbf{C}_0) \leftarrow Algorithm\ 4.2\ (\{(\mathbf{w}_m, b_m)\}_m)$
        $\mathbf{v}_0 \leftarrow eig(\mathbf{C}_0)$
 5: Normalize:
        $\mathbf{w}_t \leftarrow \mathbf{w}_0/\|\mathbf{w}_0\|, \qquad b_t \leftarrow b_0/\|\mathbf{w}_0\|$
        $\mathbf{v}_t \leftarrow orthonormalize\ \mathbf{v}_0\ with\ respect\ to\ \mathbf{w}_0$
 6: Compensate Shift:
              $b_t \leftarrow Algorithm\ 4.3\ (\mathbf{w}_t, b_t, \{\mathcal{D}_m\}, \mathcal{T})$
 7: Update Bias:
              $b_t \leftarrow Algorithm\ 4.4\ (\mathbf{w}_t, b_t, \mathcal{T})$
 8: repeat
 9:     Update Normal Vector:
              $\mathbf{w}_t \leftarrow Algorithm\ 4.6\ (\mathbf{w}_t, b_t, \mathbf{v}_t, \mathcal{T})$
10:     Update Bias:
              $b_t \leftarrow Algorithm\ 4.4\ (\mathbf{w}_t, b_t, \mathcal{T})$
11: until Stopping conditions are satisfied
```

Output: $(\mathbf{w}_t, b_t)$ or $f_t = \langle \mathbf{w}_t, \mathbf{x} \rangle + b_t$

For the experiments, peripheral blood sample datasets were obtained from 35 normal patients. These datasets are provided by the Department of Pathology at the University of Michigan. The number of events in a dataset ranges from $10,000$ to $100,000$ with a varying portion of lymphocytes among them. Ordinary cells as well as dead cells, cell debris and doublets are referred to as events. Figure 4.2 shows the number of total events and lymphocytes in each dataset. An event in a dataset has six attributes (known as FS, SS, CD45, CD4, CD8 and CD3) and a corresponding binary label ($+1$ for lymphocytes and -1 for others) from the manual gates set by experts (see Figure 4.1).

For the experiments, we adopt a leave-one-out setting: choose a dataset as a target task

$\mathcal{T}$, hide its labels, and treat the other datasets as source tasks $\mathcal{D}_m$. Each of the source task datasets constitutes a binary classification problem with the goal of predicting the correct labels.

On each source dataset $\mathcal{D}_m$, we trained a SVM classifier f_m with the LIBSVM package [16]. Throughout the experiments, we fixed all the regularization parameters C_m to 1. Then we applied the algorithms described in Section 4.4 and evaluated the prediction accuracy on the unlabeled target dataset $\mathcal{T}$. The following transfer learning algorithms are considered:

- f_0 : baseline classifier with no adaptation, referred to as "baseline."
- f_b : classifier adapted to $\mathcal{T}$ by varying the bias-only, referred to as "bias."
- f_t : classifier adapted to $\mathcal{T}$ by varying both the direction and the bias, referred to as "dir. and bias."

In addition to the above classifiers, we compared the error rates from the following classifiers as points of reference:

- `Pooling`

  A SVM is trained after merging all source data as in Toedling et al. [72].

- f_m for $m = 1, \cdots, M$

  Each classifier f_m learned from a source dataset $\mathcal{D}_m$ is applied straight to the target dataset $\mathcal{T}$. Note that this emulates a supervised learning setup with a train sample $\mathcal{D}_m$ and a test sample $\mathcal{T}$ while implicitly assuming $\mathcal{D}_m$ and $\mathcal{T}$ are drawn from the same distribution. A box plot in Figure 4.3 displays the range of results with some '$+$' indicating extreme values. Table 4.1 numbers f_m^{Best}, the best of the 34 error rates.

- `Oracle`

  We also applied the standard SVM with the true labels of the target task data. Its performance is computed by 5-fold cross validation. This quantity gives us a glimpse of the misclassification rate we can expect when a sufficient amount of labeled data are available for the target task.

For each of the datasets, we repeated this experiment and reported their results in Figure 4.3 and Table 4.1.

As can be seen in the figure and table, applying one of the f_m to the target task can result in a wide range of accuracy. The `Pooling` performs poorly on many datasets. The classifier from `Pooling` can be biased toward larger source data. In addition, the `Pooling` also makes the classification problem more difficult. Even if classes are well-separated in each dataset, the separation will be lost in the merged dataset. The baseline classifier f_0 typically improves when we adapt f_0 by changing the bias variable in most cases except Case 14 and Case 23. They further improve by adaptively varying both the direction and the bias. The differences among the f_m^{Best}, `Oracle` and f_t are very small. This reveals that our strategy can successfully replicate what experts do in the field without labeled training set for the target task.

4.6 Conclusion

We cast flow cytometry auto-gating as a novel kind of transfer learning problem. By combining existing ideas from transfer learning, together with a low-density separation criterion for class separation, our approach can leverage expert-gated datasets for the automatic gating of a new unlabeled dataset.

Although linear classifiers are sufficient to gate lymphocytes in peripheral blood, non-linear classifiers may be necessary for other kinds of auto-gating. Our approach accommodates the incorporation of inner-product kernels, which may offer a solution to such problems. It is also quite likely that several other strategies from the transfer learning literature can be adapted to this problem.

Biological and technical variation pose challenges for the analysis of many types of biomedical data. Typically one or both types of variation is accounted for by performing task-independent "normalization" prior to analysis. Our approach to flow cytometry auto-gating can be viewed as a task-dependent approach. The application of transfer learning to overcome biological and/or technical variations in other kinds of biomedical data is an interesting problem for future work.

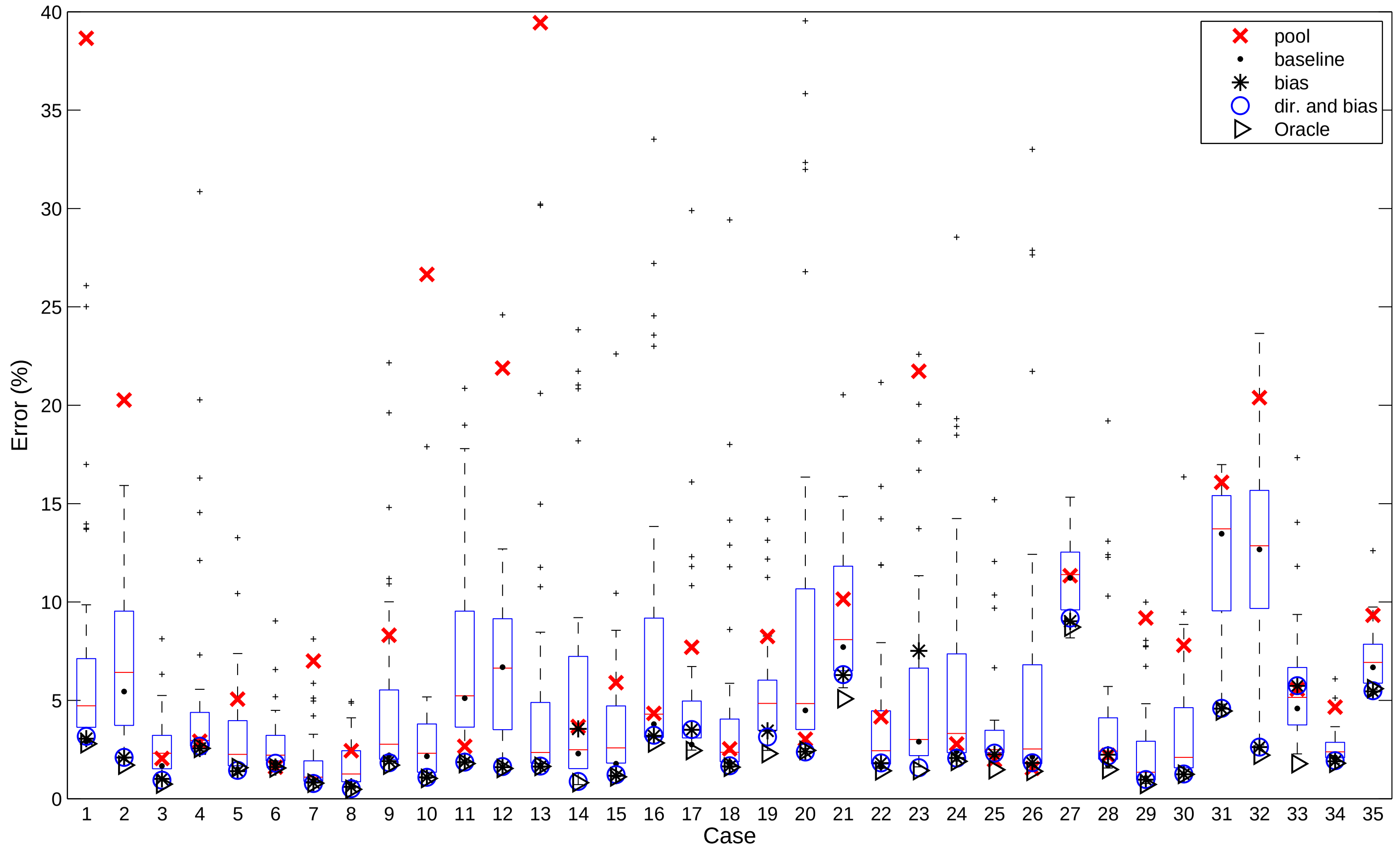


Fig. 4.3: The error rates of various classifiers on each unlabeled dataset. The box plot corresponds to the range of results when one of f_m is applied to $\mathcal{T}$. '+' marks an extreme value that deviates from the others. The results from f_t, `Oracle` and the best of f_m are usually indistinguishable.

CASE	POOL	f_0	f_b	f_t	f_m^{Best}	ORACLE
1	38.65	2.91	3.05	3.17	2.87	2.79
2	20.27	5.44	2.09	2.10	2.06	1.71
3	2.05	1.66	1.00	0.94	0.91	0.74
4	2.93	2.62	2.54	2.67	2.44	2.56
5	5.06	1.50	1.40	1.44	1.41	1.58
6	1.60	1.84	1.60	1.80	1.62	1.56
7	7.00	0.91	0.82	0.77	0.80	0.79
8	2.44	0.65	0.60	0.50	0.52	0.47
9	8.31	2.19	1.91	1.83	1.78	1.71
10	26.65	2.16	1.09	1.09	1.03	1.03
11	2.67	5.11	1.86	1.86	1.77	1.79
12	21.89	6.69	1.60	1.63	1.78	1.54
13	39.44	1.69	1.63	1.65	1.59	1.64
14	3.67	2.29	3.55	0.87	0.71	0.81
15	5.90	1.78	1.16	1.22	1.11	1.11
16	4.34	3.79	3.19	3.23	2.82	2.83
17	7.70	2.75	3.49	3.51	2.47	2.44
18	2.53	1.86	1.64	1.67	1.59	1.60
19	8.25	3.44	3.45	3.14	2.46	2.29
20	3.03	4.48	2.39	2.37	2.56	2.45
21	10.14	7.71	6.28	6.30	5.64	5.08
22	4.16	1.60	1.81	1.82	1.54	1.42
23	21.73	2.89	7.51	1.58	1.61	1.43
24	2.79	2.41	2.06	2.06	1.91	1.89
25	1.98	2.22	2.25	2.32	2.04	1.47
26	1.55	2.13	1.82	1.83	1.42	1.39
27	11.34	11.22	9.02	9.18	8.17	8.72
28	2.21	1.68	2.23	2.17	1.56	1.48
29	9.19	1.06	0.96	0.97	0.77	0.73
30	7.80	1.25	1.24	1.25	1.25	1.24
31	16.08	13.46	4.57	4.59	4.80	4.45
32	20.39	12.66	2.62	2.62	2.72	2.21
33	5.57	4.58	5.74	5.74	2.28	1.77
34	4.66	2.10	1.90	1.93	1.79	1.80
35	9.33	6.68	5.46	5.49	5.56	5.59
AVG	9.81	3.70	2.73	2.49	2.21	2.12
STD ERR	1.68	0.54	0.33	0.30	0.26	0.27

Table 4.1: The error rates (%) of various classifiers on each flow cytometry dataset, with the other 34 treated as labeled datasets. The results from f_t adapted to the unlabeled target data are comparable to the results from `Oracle` trained on labeled target data. For detailed explanations of the experiment setup and the header, readers are referred to the text.

CHAPTER 5

Nested Support Vector Machines

5.1 Introduction

Many statistical learning problems may be characterized as problems of *set estimation*. In these problems, the input takes the form of a random sample of points in a feature space, while the desired output is a subset G of the feature space. For example, in density level set estimation, a random sample from a density is given and G is an estimate of a density level set. In binary classification, labeled training data are available, and G is the set of all feature vectors predicted to belong to one of the classes.

In other statistical learning problems, the desired output is a *family* of sets G_θ with the index θ taking values in a continuum. For example, estimating density level sets at multiple levels is an important task for many problems including clustering [33], outlier ranking [43], minimum volume set estimation [61], and anomaly detection [59]. Estimating cost-sensitive classifiers at a range of different cost asymmetries is important for ranking [36], Neyman-Pearson classification [60], semi-supervised novelty detection [62], and ROC studies [5].

Support vector machines (SVMs) are powerful nonparametric approaches to set estimation [58]. However, both the one-class SVM (OC-SVM) for level set estimation and the standard two-class SVM for classification do not produce set estimates that are *nested*

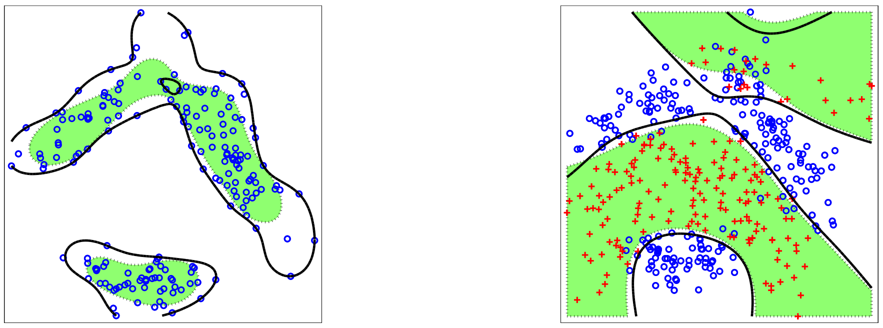

(a) one-class SVM (b) cost-sensitive SVM

Fig. 5.1: Two decision boundaries from a one-class SVM (a) and a cost-sensitive SVM (b) at two density levels and cost asymmetries. The shaded regions indicate the density level set estimate at the higher density level and the positive decision set estimate at the lower cost asymmetry, respectively. These regions are not completely contained inside the solid contours corresponding to the smaller density level or the larger cost asymmetry, hence the two decision sets are not properly nested.

as the parameter λ of the OC-SVM or, respectively, the misclassification cost of the two-class SVM is varied. As displayed in Fig. 5.1, set estimates from the original SVMs are not properly nested. On the other hand, Fig. 5.2 shows nested counterparts obtained from our proposed methods (see Section 5.3, 5.4). Since the true sets being estimated are in fact nested, estimators that enforce the nesting constraint will not only avoid nonsensical solutions, but should also be more accurate and less sensitive to parameter settings and perturbations of the training data. One way to generate nested SVM classifiers is to train a cost-insensitive SVM and simply vary the offset. However, this often leads to inferior performance as demonstrated in [5].

Recently Clémençon and Vayatis [18] developed a method for bipartite ranking that also involves computing nested estimates of cost-sensitive classifiers at a finite grid of costs. Their set estimates are computed individually, and nesting is imposed subsequently through an explicit process of successive unions. These sets are then extended to a complete scoring function through piecewise constant interpolation. Their interest is primarily

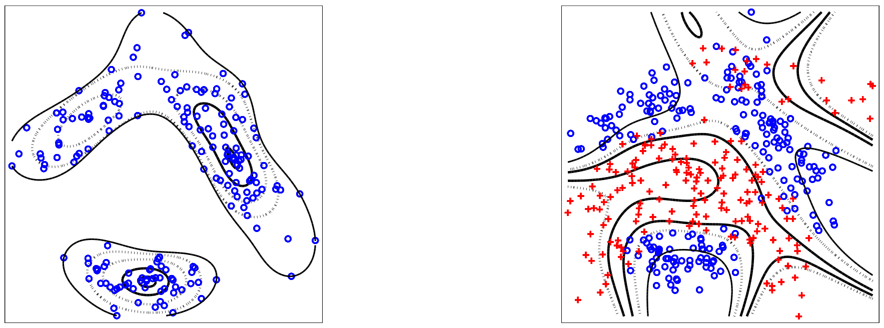

(a) nested OC-SVM (b) nested CS-SVM

Fig. 5.2: Five decision boundaries from our nested OC-SVM (a) and nested CS-SVM (b) at five different density levels and cost asymmetries, respectively. These decision boundaries from nested SVMs do not cross each other, unlike the decision boundaries from the original SVMs (OC-SVM and CS-SVM). Therefore, the corresponding set estimates are properly nested.

theoretical, as their estimates entail empirical risk minimization, and their results assume the underlying Bayes classifiers lies in a Vapnik-Chervonenkis class.

In this chapter, we develop nested variants of one-class and two-class SVMs by incorporating nesting constraints into the dual quadratic programs associated with these methods. Decomposition algorithms for solving these modified duals are also presented. Like the solution paths for conventional SVMs [5, 34, 39], nested SVM solution paths are also piecewise linear in the control parameters, but require far fewer breakpoints. We compare our nested paths to the unnested paths on synthetic and benchmark data sets. We also quantify the degree to which standard SVMs are unnested, which is often quite high. The Matlab implementation of our algorithms is available at `http://www.eecs.umich.edu/~cscott/code/nestedsvm.zip`. A preliminary version of this work appeared in [40].

5.1.1 Motivating Applications

With the multiple set estimates from nested SVMs over density levels or cost asymmetries, the following applications are envisioned.

Ranking : In the bipartite ranking problem [1], we are given labeled examples from two classes, and the goal is constructing a score function that rates new examples according to their likelihood of belonging to the positive class. If the decision sets are not nested as cost asymmetries or density levels varies, then the resulting score function leads to ambiguous ranking. Nested SVMs will make the ranking unambiguous and less sensitive to perturbations of the data. See Section 5.6.3 for further discussion.

Clustering : Clusters may be defined as the connected components of a density level set. The level at which the density is thresholded determines a tradeoff between cluster number and cluster coverage. Varying the level from 0 to ∞ yields a “cluster tree” [66] that depicts the bifurcation of clusters into disjoint components and gives a hierarchical representation of cluster structure.

Anomaly Detection : Anomaly detection aims to identify deviations from nominal data when combined observations of nominal and anomalous data are given. Scott and Kolaczyk [59] and Scott and Blanchard [62] present approaches to classifying the contaminated, unlabeled data by solving multiple level set estimation and multiple cost-sensitive classification problems, respectively.

5.2 Background on CS-SVM and OC-SVM

In this section, we will overview two SVM variants and show how they can be used to learn set estimates. To establish notation and basic concepts, we briefly review SVMs.

Suppose that we have a random sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \{-1, +1\}$ is its class. An SVM finds a separating hyperplane with a normal vector $\mathbf{w}$ in a high dimensional space $\mathcal{H}$ by solving

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_i \xi_i$$

$$\text{s.t.} \quad y_i\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

where λ is a regularization parameter and Φ is a nonlinear function that maps each data point into $\mathcal{H}$ generated by a positive semi-definite kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. This kernel corresponds to an inner product in $\mathcal{H}$ through $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}')\rangle$. Then the two half-spaces of the hyperplane $\{\Phi(\mathbf{x}) : f(\mathbf{x}) \equiv \langle \mathbf{w}, \Phi(\mathbf{x})\rangle = 0\}$ form positive and negative decision sets. Since the offset of the hyperplane is often omitted when Gaussian or inhomogeneous polynomial kernels are chosen [37], it is not considered in this formulation. More detailed discussion on SVMs can be found in [58].

5.2.1 Cost-Sensitive SVM

The SVM above, which we call a cost-insensitive SVM (CI-SVM), penalizes errors in both classes equally. However, there are many applications where the numbers of data samples from each class are not balanced, or false positives and false negatives incur different costs. The cost-sensitive SVM (CS-SVM) handles this issue by controlling the cost asymmetry between false positives and false negatives [49].

Let $I_+ = \{i : y_i = +1\}$ and $I_- = \{i : y_i = -1\}$ denote the two index sets, and γ denote the cost asymmetry. Then a CS-SVM solves

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \gamma\sum_{I_+} \xi_i + (1-\gamma)\sum_{I_-} \xi_i \tag{5.1}$$

$$\text{s.t.} \quad y_i\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

where $\mathbf{w}$ is the normal vector of the hyperplane. When $\gamma = \frac{1}{2}$, CS-SVMs reduce to CI-SVMs.

In practice this optimization problem is solved via its dual, which depends only on a set of Lagrange multipliers (one for each $\mathbf{x}_i$):

$$\min_{\boldsymbol{\alpha}} \frac{1}{2\lambda} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K_{i,j} - \sum_i \alpha_i \tag{5.2}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq \mathbf{1}_{\{y_i<0\}} + y_i \gamma, \quad \forall i.$$

where $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$. The indicator function $\mathbf{1}_{\{A\}}$ returns 1 if the condition A is true and 0 otherwise. Since there is no offset term, a linear constraint $\sum_i \alpha_i y_i = 0$ does not appear in the dual.

Once an optimal solution $\boldsymbol{\alpha}^*(\gamma) = (\alpha_1^*(\gamma), \ldots, \alpha_N^*(\gamma))$ is found, the sign of the decision function

$$f_\gamma(\mathbf{x}) = \frac{1}{\lambda} \sum_i \alpha_i^*(\gamma) y_i k(\mathbf{x}, \mathbf{x}_i) \tag{5.3}$$

determines the class of $\mathbf{x}$. If $k(\cdot, \cdot) \geq 0$, then this decision function takes only non-positive values when $\gamma = 0$, and corresponds to $(0, 0)$ in the ROC. On the other hand, $\gamma = 1$ penalizes only the violations of positive examples, and corresponds to $(1, 1)$ in the ROC.

Bach et al. [5] extended the method of Hastie et al. [34] to the CS-SVM. They showed that $\alpha_i^*(\gamma)$ are piecewise linear in γ, and derived an efficient algorithm for computing the entire path of solutions to (5.2). Thus, a family of classifiers at a range of cost asymmetries can be found with a computational cost comparable to solving (5.2) for a single γ.

5.2.2 One-Class SVM

The OC-SVM was proposed in [57, 68] to estimate a level set of an underlying probability density given a data sample from the density. In one-class problems, all the instances are assumed from the same class. The primal quadratic program of the OC-SVM is

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i \tag{5.4}$$

$$\text{s.t.} \quad \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i.$$

This problem is again solved via its dual in practice:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2\lambda}\sum_i\sum_j \alpha_i\alpha_j K_{i,j} - \sum_i \alpha_i \tag{5.5}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq \frac{1}{N}, \quad \forall i.$$

This formulation is equivalent to the more common ν parametrization [57], and is more convenient for our purposes. We also note that the OC-SVM can be solved by setting $\gamma = 1/2$ and $y_i = 1$ in the CS-SVM. However, our path algorithm for the OC-SVM, which varies λ, is not a special case of our path algorithm for the CS-SVM, which varies γ while holding λ fixed.

A solution $\boldsymbol{\alpha}^*(\lambda) = (\alpha_1^*(\lambda), \ldots, \alpha_N^*(\lambda))$ defines a decision function that determines whether a point is an outlier or not. Here $\alpha_i^*(\lambda)$ are also piecewise linear in λ [39]. From this property, we can develop a path following algorithm and generate a family of level set estimates with a small computational cost. The set estimate conventionally associated with the OC-SVM is given by

$$\widehat{G}_\lambda = \{\mathbf{x} : \sum_i \alpha_i^*(\lambda)k(\mathbf{x}_i, \mathbf{x}) > \lambda\}. \tag{5.6}$$

Vert and Vert [73] showed that by modifying this estimate slightly, substituting $\alpha_i^*(\eta\lambda)$ for $\alpha_i^*(\lambda)$ where $\eta > 1$, (5.6) leads to a consistent estimate of the true level set when a Gaussian kernel with a well-calibrated bandwidth is used. Regardless of whether $\eta = 1$ or $\eta > 1$, however, the obtained estimates are not guaranteed to be nested as we will see in Section 5.6. Note also that when $\alpha_i^*(\lambda) = \frac{1}{N}$, (5.6) is equivalent to set estimation based on kernel density estimation.

5.3 Nested CS-SVM

In this section, we develop the nested cost-sensitive SVM (NCS-SVM), which aims to produce nested positive decision sets $G_\gamma = \{\mathbf{x} : f_\gamma(\mathbf{x}) > 0\}$ as the cost asymmetry γ varies. Our construction is a two stage process. We first select a finite number of cost asymmetries $0 = \gamma_1 < \gamma_2 < \ldots < \gamma_M = 1$ a priori and generate a family of nested decision sets at the preselected cost asymmetries. We achieve this goal by incorporating nesting constraints into the dual quadratic program of CS-SVM. Second, we linearly interpolate the solution coefficients of the finite nested collection to a continuous nested family defined for all γ. As an efficient method to solve the formulated problem, we present a decomposition algorithm.

5.3.1 Finite Family of Nested Sets

Our NCS-SVM finds decision functions at cost asymmetries $\gamma_1, \gamma_2, \ldots, \gamma_M$ simultaneously by minimizing the sum of duals (5.2) at each γ and by imposing additional constraints that induce nested sets. For a fixed λ and preselected cost asymmetries $0 = \gamma_1 < \gamma_2 < \cdots < \gamma_M = 1$, an NCS-SVM solves

$$\min_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M} \sum_{m=1}^{M} \left[\frac{1}{2\lambda} \sum_{i,j} \alpha_{i,m}\alpha_{j,m} y_i y_j K_{i,j} - \sum_i \alpha_{i,m} \right] \tag{5.7}$$

$$\text{s.t.} \quad 0 \le \alpha_{i,m} \le \mathbf{1}_{\{y_i<0\}} + y_i\gamma_m, \quad \forall i, m \tag{5.8}$$

$$y_i\alpha_{i,1} \le y_i\alpha_{i,2} \le \cdots \le y_i\alpha_{i,M}, \quad \forall i \tag{5.9}$$

where $\boldsymbol{\alpha}_m = (\alpha_{1,m}, \ldots, \alpha_{N,m})$ and $\alpha_{i,m}$ is a coefficient for data point $\mathbf{x}_i$ and cost asymmetry γ_m. Then its optimal solution $\boldsymbol{\alpha}_m^* = (\alpha_{1,m}^*, \ldots, \alpha_{N,m}^*)$ defines the decision function $f_{\gamma_m}(\mathbf{x}) = \frac{1}{\lambda}\sum_i \alpha_{i,m}^* y_i k(\mathbf{x}_i, \mathbf{x})$ and its corresponding decision set $\widehat{G}_{\gamma_m} = \{\mathbf{x} : f_{\gamma_m}(\mathbf{x}) > 0)\}$ for each m. In Section 5.7, the proposed quadratic program for NCS-SVMs is interpreted as a dual of a corresponding primal quadratic program.

5.3.2 Interpolation

For an intermediate cost asymmetry γ between two cost asymmetries, say γ_1 and γ_2 without loss of generality, we can write $\gamma = \epsilon\gamma_1 + (1-\epsilon)\gamma_2$ for some $\epsilon \in [0, 1]$. Then we define new coefficients $\alpha_i^*(\gamma)$ through linear interpolation:

$$\alpha_i^*(\gamma) = \epsilon\alpha_{i,1}^* + (1-\epsilon)\alpha_{i,2}^*. \tag{5.10}$$

Then the positive decision set at cost asymmetry γ is

$$\widehat{G}_\gamma = \{\mathbf{x} : f_\gamma(\mathbf{x}) = \frac{1}{\lambda}\sum_i \alpha_i^*(\gamma) y_i k(\mathbf{x}_i, \mathbf{x}) > 0\}. \tag{5.11}$$

This is motivated by the piecewise linearity of the Lagrange multipliers of the CS-SVM, and is further justified by the following result.

Proposition 5.1. *The nested CS-SVM equipped with a kernel such that* $k(\cdot,\cdot) \geq 0$ *(e.g., Gaussian kernels or polynomial kernels of even orders) generates nested decision sets. In other words, if* $0 \leq \gamma_\epsilon < \gamma_\delta \leq 1$*, then* $\widehat{G}_{\gamma_\epsilon} \subset \widehat{G}_{\gamma_\delta}$.

Proof. We prove the proposition in three steps. First, we show that sets from (5.7) satisfy $\widehat{G}_{\gamma_1} \subset \widehat{G}_{\gamma_2} \subset \cdots \subset \widehat{G}_{\gamma_M}$. Second, we show that if $\gamma_m < \gamma < \gamma_{m+1}$, then $\widehat{G}_{\gamma_m} \subset \widehat{G}_\gamma \subset \widehat{G}_{\gamma_{m+1}}$. Finally, we prove that any two sets from the NCS-SVM are nested.

Without loss of generality, we show $\widehat{G}_{\gamma_1} \subset \widehat{G}_{\gamma_2}$. Let $\boldsymbol{\alpha}_1^*$ and $\boldsymbol{\alpha}_2^*$ denote the optimal solutions for γ_1 and γ_2. Then from $k(\cdot,\cdot) \geq 0$ and (5.9), we have $\sum_i \alpha_{i,1}^* y_i k(\mathbf{x}_i, \mathbf{x}) \leq \sum_i \alpha_{i,2}^* y_i k(\mathbf{x}_i, \mathbf{x})$. Therefore, $\widehat{G}_{\gamma_1} = \{\mathbf{x} : f_{\gamma_1}(\mathbf{x}) > 0\} \subset \widehat{G}_{\gamma_2} = \{\mathbf{x} : f_{\gamma_2}(\mathbf{x}) > 0\}$.

Next, without loss of generality, we show $\widehat{G}_{\gamma_1} \subset \widehat{G}_\gamma \subset \widehat{G}_{\gamma_2}$ when $\gamma_1 \leq \gamma \leq \gamma_2$. The linear interpolation (5.10) and the nesting constraints (5.9) imply $y_i\alpha_{i,1}^* \leq y_i\alpha_i^*(\gamma) \leq y_i\alpha_{i,2}^*$, which, in turn, leads to $\sum_i \alpha_{i,1}^* y_i k(\mathbf{x}_i, \mathbf{x}) \leq \sum_i \alpha_i^*(\gamma) y_i k(\mathbf{x}_i, \mathbf{x}) \leq \sum_i \alpha_{i,2}^* y_i k(\mathbf{x}_i, \mathbf{x})$.

Now consider arbitrary $0 \leq \gamma_\epsilon < \gamma_\delta \leq 1$. If $\gamma_\epsilon \leq \gamma_m \leq \gamma_\delta$ for some m, then $\widehat{G}_{\gamma_\epsilon} \subset \widehat{G}_{\gamma_\delta}$ by the above results. Thus, suppose this is not the case and assume $\gamma_1 < \gamma_\epsilon < \gamma_\delta < \gamma_2$ without loss of generality. Then there exist $\epsilon > \delta$ such that $\gamma_\epsilon = \epsilon\gamma_1 + (1-\epsilon)\gamma_2$ and $\gamma_\delta = \delta\gamma_1 + (1-\delta)\gamma_2$. Suppose $\mathbf{x} \in \widehat{G}_{\gamma_\epsilon}$. Then $\mathbf{x} \in \widehat{G}_{\gamma_2}$, hence $f_{\gamma_\epsilon}(\mathbf{x}) = \frac{1}{\lambda}\sum_i(\epsilon\alpha^*_{i,1} + (1-\epsilon)\alpha^*_{i,2})y_i k(\mathbf{x}_i, \mathbf{x}) > 0$ and $f_{\gamma_2}(\mathbf{x}) = \frac{1}{\lambda}\sum_i \alpha^*_{i,2} y_i k(\mathbf{x}_i, \mathbf{x}) > 0$. By adding $\frac{\delta}{\epsilon} f_{\gamma_\epsilon}(\mathbf{x}) + (1 - \frac{\delta}{\epsilon}) f_{\gamma_2}(\mathbf{x})$, we have $f_{\gamma_\delta}(\mathbf{x}) = \sum_i(\delta\alpha^*_{i,1} + (1-\delta)\alpha^*_{i,2})y_i k(\mathbf{x}_i, \mathbf{x}) > 0$. Thus, $\widehat{G}_{\gamma_\epsilon} \subset \widehat{G}_{\gamma_\delta}$. $\square$

The assumption that the kernel is positive can in some cases be attained through preprocessing of the data. For example, a cubic polynomial kernel can be applied if the data support is shifted to lie in the positive orthant, so that the kernel function is in fact always positive.

5.3.3 Decomposition Algorithm

The objective function (5.7) requires optimization over $N \times M$ variables. Due to its large size, standard quadratic programming algorithms are inadequate. Thus, we develop a decomposition algorithm that iteratively divides the large optimization problem into subproblems and optimizes the smaller problems. A similar approach also appears in a multi-class classification algorithm [19], although the algorithm developed there is substantively different from ours. The decomposition algorithm follows:

1. Choose an example $\mathbf{x}_i$ from the data set.

2. Optimize coefficients $\{\alpha_{i,m}\}_{m=1}^M$ corresponding to $\mathbf{x}_i$ while leaving other variables fixed.

3. Repeat 1 and 2 until the optimality condition error falls below a predetermined tolerance.

The pseudo code given in Algorithm 5.1 initializes with a feasible solution $\alpha_{i,m} = \mathbf{1}_{\{y_i<0\}} + y_i\gamma_m,\ \forall i, m$. A simple way of selection and termination is cycling through all the $\mathbf{x}_i$ or picking $\mathbf{x}_i$ randomly and stopping after a fixed number of iterations. However, by checking the Karush-Kuhn-Tucker (KKT) optimality conditions and choosing $\mathbf{x}_i$ most violating the conditions [7], the algorithm will converge in far fewer iterations. In the Appendix, we provide a detailed discussion of the data point selection scheme and termination criterion based on the KKT optimality conditions.

In step 2, the algorithm optimizes a set of variables associated to the chosen data point. Without loss of generality, let us assume that the data point $\mathbf{x}_1$ is chosen and $\{\alpha_{1,m}\}_{m=1}^{M}$ will be optimized while fixing the other $\alpha_{i,m}$. We rewrite the objective function (5.7) in terms of $\alpha_{1,m}$:

$$\begin{aligned}
&\sum_m \left[\frac{1}{2\lambda}\sum_{i,j}\alpha_{i,m}\alpha_{j,m}y_iy_jK_{i,j} - \sum_i \alpha_{i,m}\right] \\
=&\frac{1}{\lambda}\sum_m \left[\frac{1}{2}\alpha_{1,m}^2K_{1,1} + \alpha_{1,m}\left(\sum_{j\neq 1}\alpha_{j,m}y_1y_jK_{1,j} - \lambda\right)\right] + C \\
=&\frac{1}{\lambda}\sum_m \left[\frac{1}{2}\alpha_{1,m}^2K_{1,1} + \alpha_{1,m}\left(\lambda y_1 f_{1,m} - \alpha_{1,m}^{\text{old}}K_{1,1} - \lambda\right)\right] + C \\
=&\frac{K_{1,1}}{\lambda}\sum_m \left[\frac{1}{2}\alpha_{1,m}^2 - \alpha_{1,m}\left(\alpha_{1,m}^{\text{old}} + \frac{\lambda(1-y_1f_{1,m})}{K_{1,1}}\right)\right] + C
\end{aligned}$$

where $f_{1,m} = \frac{1}{\lambda}\left(\sum_{j\neq 1}\alpha_{j,m}y_jK_{1,j} + \alpha_{1,m}^{\text{old}}y_1K_{1,1}\right)$ and $\alpha_{1,m}^{\text{old}}$ denote the output and the variable preceding the update. These values can be easily computed from the previous iteration result. C is a collection of terms that do not depend on $\alpha_{1,m}$.

Then the algorithm solves the new subproblem with M variables,

$$\min_{\alpha_{1,1},\ldots,\alpha_{1,M}} \sum_m \left[\frac{1}{2}\alpha_{1,m}^2 - \alpha_{1,m}\alpha_{1,m}^{\text{new}}\right]$$

$$\text{s.t.}\quad 0 \le \alpha_{1,m} \le \mathbf{1}_{\{y_1<0\}} + y_1\gamma_m, \quad \forall m$$

$$y_1\alpha_{1,1} \le y_1\alpha_{1,2} \le \cdots \le y_1\alpha_{1,M}$$

Algorithm 5.1 Decomposition algorithm for a nested cost-sensitive SVM.

```
Input: ${(\mathbf{x}_i, y_i)}_{i=1}^N$, ${\gamma_m}_{m=1}^M$
1: Initialize:
       $$\alpha_{i,m} \leftarrow \mathbf{1}_{\{y_i<0\}} + y_i\gamma_m, \quad \forall i, m$$
2: repeat
3:     Choose a data point $\mathbf{x}_i$.
4:     Compute:
           $$f_{i,m} \leftarrow \frac{1}{\lambda}\sum_j \alpha_{j,m} y_j K_{i,j}, \quad \forall m$$
           $$\alpha_{i,m}^{\text{new}} \leftarrow \alpha_{i,m} + \frac{\lambda(1 - y_i f_{i,m})}{K_{i,i}}, \quad \forall m$$
5:     Update $\{\alpha_{i,m}\}_{m=1}^M$ with the solution of the subproblem:
           $$\min_{\alpha_{i,1},\dots,\alpha_{i,M}} \sum_m \left[\frac{1}{2}\alpha_{i,m}^2 - \alpha_{i,m}\alpha_{i,m}^{\text{new}}\right]$$
           $$\text{s.t.} \quad 0 \le \alpha_{i,m} \le \mathbf{1}_{\{y_i<0\}} + y_i\gamma_m, \quad \forall m$$
           $$y_i\alpha_{i,1} \le y_i\alpha_{i,2} \le \cdots \le y_i\alpha_{i,M}$$
6: until Accuracy conditions are satisfied
Output: $\widehat{G}_{\gamma_m} = \{\mathbf{x} : \sum_i \alpha_{i,m} y_i k(\mathbf{x}_i, \mathbf{x}) > 0\}, \quad \forall m$
```

where $\alpha_{1,m}^{\text{new}} = \alpha_{1,m}^{\text{old}} + \frac{\lambda(1-y_1 f_{1,m})}{K_{1,1}}$ is the solution if feasible. This subproblem is much smaller and can be solved efficiently via standard quadratic program solvers.

5.4 Nested OC-SVM

In this section, we present a nested extension of OC-SVM. The nested OC-SVM (NOC-SVM) estimates a family of nested level sets over a continuum of levels λ. Our approach here parallels the approach developed for the NCS-SVM. First, we will introduce an objective function for nested set estimation, and will develop analogous interpolation and decomposition algorithms for the NOC-SVM.

5.4.1 Finite Family of Nested Sets

For M different density levels of interest $\lambda_1 > \lambda_2 > \cdots > \lambda_M > 0$, an NOC-SVM solves the following optimization problem

$$\min_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M} \sum_{m=1}^{M} \left[\frac{1}{2\lambda_m} \sum_{i,j} \alpha_{i,m}\alpha_{j,m} K_{i,j} - \sum_i \alpha_{i,m} \right] \tag{5.12}$$

$$\text{s.t.} \quad 0 \le \alpha_{i,m} \le \frac{1}{N}, \quad \forall i, m \tag{5.13}$$

$$\frac{\alpha_{i,1}}{\lambda_1} \le \frac{\alpha_{i,2}}{\lambda_2} \le \cdots \le \frac{\alpha_{i,M}}{\lambda_M}, \quad \forall i \tag{5.14}$$

where $\boldsymbol{\alpha}_m = (\alpha_{1,m}, \ldots, \alpha_{N,m})$ and $\alpha_{i,m}$ corresponds to data point $\mathbf{x}_i$ at level λ_m. Its optimal solution $\boldsymbol{\alpha}_m^* = (\alpha_{1,m}^*, \ldots, \alpha_{N,m}^*)$ determines a level set estimate $\widehat{G}_{\lambda_m} = \{\mathbf{x} : f_{\lambda_m}(\mathbf{x}) > 1\}$ where $f_{\lambda_m}(\mathbf{x}) = \frac{1}{\lambda_m} \sum_i \alpha_{i,m}^* k(\mathbf{x}_i, \mathbf{x})$. In practice, we can choose λ_1 and λ_M to cover the entire range of interesting values of density level (see Section 5.6.2, Appendix 5.C). In Section 5.7, this quadratic program for the NOC-SVM is interpreted as a dual of a corresponding primal quadratic program.

5.4.2 Interpolation and Extrapolation

We construct a density level set estimate at an intermediate level λ between two preselected levels, say λ_1 and λ_2. At $\lambda = \epsilon\lambda_1 + (1-\epsilon)\lambda_2$ for some $\epsilon \in [0, 1]$, we set

$$\alpha_i^*(\lambda) = \epsilon\alpha_{i,1}^* + (1-\epsilon)\alpha_{i,2}^*.$$

For $\lambda > \lambda_1$, we extrapolate the solution by setting $\alpha_i^*(\lambda) = \alpha_{i,1}^*$ for $\forall i$. These are motivated by the facts that the OC-SVM solution is piecewise linear in λ and remains constant for $\lambda > \lambda_1$ as presented in Appendix 5.C. Then the level set estimate becomes

$$\widehat{G}_\lambda = \{\mathbf{x} : \sum_i \alpha_i^*(\lambda) k(\mathbf{x}_i, \mathbf{x}) > \lambda\}. \tag{5.15}$$

The level set estimates generated from the above process are shown to be nested in the next Proposition.

Proposition 5.2. *The nested OC-SVM equipped with a kernel such that* $k(\cdot,\cdot) \geq 0$ *(in particular, a Gaussian kernel) generates nested density level set estimates. That is, if* $0 < \lambda_\epsilon < \lambda_\delta < \infty$*, then* $\widehat{G}_{\lambda_\epsilon} \supset \widehat{G}_{\lambda_\delta}$.

Proof. We prove the proposition in three steps. First, we show that sets from (5.12) satisfy $\widehat{G}_{\lambda_1} \subset \widehat{G}_{\lambda_2} \subset \cdots \subset \widehat{G}_{\lambda_M}$. Second, the interpolated set (5.15) is shown to satisfy $\widehat{G}_{\lambda_m} \subset \widehat{G}_{\lambda} \subset \widehat{G}_{\lambda_{m+1}}$ when $\lambda_m > \lambda > \lambda_{m+1}$. Finally, we prove the claim for any two sets from the NOC-SVM.

Without loss of generality, we first show $\widehat{G}_{\lambda_1} \subset \widehat{G}_{\lambda_2}$. Let $\lambda_1 > \lambda_2$ denote two density levels chosen a priori, and $\boldsymbol{\alpha}_1^*$ and $\boldsymbol{\alpha}_2^*$ denote their corresponding optimal solutions. From (5.14), we have $\sum_i \frac{\alpha_{i,1}^*}{\lambda_1} k(\mathbf{x}_i, \mathbf{x}) \leq \sum_i \frac{\alpha_{i,2}^*}{\lambda_2} k(\mathbf{x}_i, \mathbf{x})$, so the two estimated level sets are nested $\widehat{G}_{\lambda_1} \subset \widehat{G}_{\lambda_2}$.

Next, without loss of generality, we prove $\widehat{G}_{\lambda_1} \subset \widehat{G}_{\lambda} \subset \widehat{G}_{\lambda_2}$ for $\lambda_1 > \lambda > \lambda_2$. From (5.14), we have $\frac{\alpha_{i,1}^*}{\lambda_1} \leq \frac{\alpha_{i,2}^*}{\lambda_2}$ and

$$\begin{aligned}
\frac{\alpha_{i,1}^*}{\lambda_1} = \frac{\lambda \frac{\alpha_{i,1}^*}{\lambda_1}}{\lambda} &= \frac{\epsilon \alpha_{i,1}^* + (1-\epsilon)\frac{\lambda_2}{\lambda_1}\alpha_{i,1}^*}{\lambda} \\
&\leq \frac{\epsilon \alpha_{i,1}^* + (1-\epsilon)\alpha_{i,2}^*}{\lambda} = \frac{\alpha_i^*(\lambda)}{\lambda} \\
&\leq \frac{\epsilon \frac{\lambda_1}{\lambda_2}\alpha_{i,2}^* + (1-\epsilon)\alpha_{i,2}^*}{\lambda} = \frac{\lambda \frac{\alpha_{i,2}^*}{\lambda_2}}{\lambda} = \frac{\alpha_{i,2}^*}{\lambda_2}.
\end{aligned}$$

Hence, $f_{\lambda_1}(\mathbf{x}) \leq f_{\lambda}(\mathbf{x}) \leq f_{\lambda_2}(\mathbf{x})$.

Now consider arbitrary $\lambda_\delta > \lambda_\epsilon > 0$. By construction, we can easily see that $\widehat{G}_{\lambda_\delta} \subset \widehat{G}_{\lambda_\epsilon} \subset \widehat{G}_{\lambda_1}$ for $\lambda_\delta > \lambda_\epsilon > \lambda_1$, and $\widehat{G}_{\lambda_M} \subset \widehat{G}_{\lambda_\delta} \subset \widehat{G}_{\lambda_\epsilon}$ for $\lambda_M > \lambda_\delta > \lambda_\epsilon$. Thus we only need to consider the case $\lambda_1 > \lambda_\delta > \lambda_\epsilon > \lambda_M$. Since above results imply $\widehat{G}_{\lambda_\delta} \subset \widehat{G}_{\lambda_\epsilon}$ if $\lambda_\delta > \lambda_m > \lambda_\epsilon$ for some m, we can safely assume $\lambda_1 > \lambda_\delta > \lambda_\epsilon > \lambda_2$ without loss of generality. Then there exist $\delta > \epsilon$ such that $\lambda_\delta = \delta\lambda_1 + (1-\delta)\lambda_2$ and $\lambda_\epsilon = \epsilon\lambda_1 + (1-\epsilon)\lambda_2$.

Suppose $\mathbf{x} \in \widehat{G}_{\lambda_\delta}$. Then $\mathbf{x} \in \widehat{G}_{\lambda_2}$ and

$$\sum_i (\delta \alpha^*_{i,1} + (1-\delta)\alpha^*_{i,2}) k(\mathbf{x}_i, \mathbf{x}) > \lambda_\delta \tag{5.16}$$

$$\sum_i \alpha^*_{i,2} k(\mathbf{x}_i, \mathbf{x}) > \lambda_2. \tag{5.17}$$

By $\frac{\epsilon}{\delta}\times$ (5.16) + $(1-\frac{\epsilon}{\delta})\times$ (5.17), we have $\sum_i (\epsilon \alpha^*_{i,1} + (1-\epsilon)\alpha^*_{i,2}) k(\mathbf{x}_i, \mathbf{x}) > \lambda_\epsilon$. Thus, $\widehat{G}_{\lambda_\delta} \subset \widehat{G}_{\lambda_\epsilon}$. □

The statement of this result focuses on the Gaussian kernel because this is the primary kernel for which the OC-SVM has been successfully applied.

5.4.3 Decomposition Algorithm

We also use a decomposition algorithm to solve (5.12). The general steps are the same as explained in Section 5.3.3 for the NCS-SVM. Algorithm 5.2 shows the outline of the algorithm. In the algorithm, a feasible solution $\alpha_{i,m} = \frac{1}{N}$ for $\forall i, m$ is used as an initial solution.

Here we present how we can divide the large optimization problem into a collection of smaller problems. Suppose that the data point $\mathbf{x}_1$ is selected and its corresponding coefficients $\{\alpha_{1,m}\}_{m=1}^M$ will be updated. Writing the objective function only in terms of $\alpha_{1,m}$, we have

$$\begin{aligned}
&\sum_m \left[\frac{1}{2\lambda_m} \sum_{i,j} \alpha_{i,m}\alpha_{j,m} K_{i,j} - \sum_i \alpha_{i,m} \right] \\
=&\sum_m \left[\frac{1}{2\lambda_m} \alpha_{1,m}^2 K_{1,1} + \alpha_{1,m} \left(\frac{1}{\lambda_m} \sum_{j \neq 1} \alpha_{j,m} K_{1,j} - 1 \right) \right] + C \\
=&\sum_m \left[\frac{1}{2\lambda_m} \alpha_{1,m}^2 K_{1,1} + \alpha_{1,m} \left(f_{1,m} - \frac{\alpha_{1,m}^{\text{old}}}{\lambda_m} K_{1,1} - 1 \right) \right] + C \\
=&K_{1,1} \sum_m \left[\frac{1}{2\lambda_m} \alpha_{1,m}^2 - \frac{\alpha_{1,m}}{\lambda_m} \left(\alpha_{1,m}^{\text{old}} + \frac{\lambda_m (1 - f_{1,m})}{K_{1,1}} \right) \right] + C
\end{aligned}$$

where $\alpha_{1,m}^{\text{old}}$ and $f_{1,m} = \frac{1}{\lambda_m}\left(\sum_{j\neq 1}\alpha_{j,m}K_{1,j} + \alpha_{1,m}^{\text{old}}K_{1,1}\right)$ denote the variable from the previous iteration step and the corresponding output, respectively. C is a constant that does not affect the solution.

Then we obtain the reduced optimization problem of M variables,

$$\min_{\alpha_{1,1},\dots,\alpha_{1,M}} \sum_m \left[\frac{1}{2\lambda_m}\alpha_{1,m}^2 - \frac{\alpha_{1,m}}{\lambda_m}\alpha_{1,m}^{\text{new}}\right] \tag{5.18}$$

$$\text{s.t.} \quad 0 \leq \alpha_{1,m} \leq \frac{1}{N}, \quad \forall m \tag{5.19}$$

$$\frac{\alpha_{1,1}}{\lambda_1} \leq \frac{\alpha_{1,2}}{\lambda_2} \leq \cdots \leq \frac{\alpha_{1,M}}{\lambda_M} \tag{5.20}$$

where $\alpha_{1,m}^{\text{new}} = \alpha_{1,m}^{\text{old}} + \frac{\lambda_m(1-f_{1,m})}{K_{1,1}}$. Notice that $\alpha_{1,m}^{\text{new}}$ becomes the solution if it is feasible. This reduced optimization problem can be solved through standard quadratic program solvers.

5.5 Computational Considerations

Here we provide guidelines for breakpoint selection and discuss the effects of interpolation.

5.5.1 Breakpoint Selection

The construction of an NCS-SVM begins with the selection of a finite number of cost asymmetries. Since the cost asymmetries take values within the range $[0, 1]$, the two breakpoints γ_1 and γ_M should be at the two extremes so that $\gamma_1 = 0$ and $\gamma_M = 1$. Then the rest of the breakpoints $\gamma_2, \cdots, \gamma_{M-1}$ can be set evenly spaced between γ_1 and γ_M.

On the other hand, the density levels for NOC-SVMs should be strictly positive. Without covering all positive reals, however, λ_1 and λ_M can be chosen to cover practically all the density levels of interest. The largest level λ_1 for the NOC-SVM is set as described in Appendix 5.C where we show that for $\lambda > \lambda_1$, the CS-SVM and OC-SVM remain un-

Algorithm 5.2 Decomposition algorithm for a nested one-class SVM.

```
Input: ${\{\mathbf{x}_i\}}_{i=1}^N$, ${\{\lambda_m\}}_{m=1}^M$
1: Initialize:
       $$\alpha_{i,m} \leftarrow \frac{1}{N}, \quad \forall i, m$$
2: repeat
3:     Choose a data point $\mathbf{x}_i$.
4:     Compute:
           $$f_{i,m} \leftarrow \frac{1}{\lambda_m} \sum_j \alpha_{j,m} K_{i,j}, \quad \forall m$$
           $$\alpha_{i,m}^{\text{new}} \leftarrow \alpha_{i,m} + \frac{\lambda_m(1 - f_{i,m})}{K_{i,i}}, \quad \forall m$$
5:     Update ${\{\alpha_{i,m}\}}_{m=1}^M$ with the solution of the subproblem:
           $$\min_{\alpha_{i,1},\ldots,\alpha_{i,M}} \sum_m \left[ \frac{1}{2\lambda_m} \alpha_{i,m}^2 - \frac{\alpha_{i,m}}{\lambda_m} \alpha_{i,m}^{\text{new}} \right]$$
           $$\text{s.t.} \quad 0 \le \alpha_{i,m} \le \frac{1}{N}, \quad \forall m$$
           $$\frac{\alpha_{i,1}}{\lambda_1} \le \frac{\alpha_{i,2}}{\lambda_2} \le \cdots \le \frac{\alpha_{i,M}}{\lambda_M}$$
6: until Accuracy conditions are satisfied
Output: $\widehat{G}_{\lambda_m} = \{\mathbf{x} : \sum_i \alpha_{i,m} k(\mathbf{x}_i, \mathbf{x}) > \lambda_m\}, \quad \forall m$
```

changed. A very small number greater than 0 is set for λ_M. Then the NOC-SVM is trained on evenly spaced breakpoints between λ_1 and λ_M.

In our experiments, we set the number of breakpoints to be $M = 5$ for NCS-SVMs and $M = 11$ for NOC-SVMs. These values were chosen because increasing the number of breakpoints M had diminishing AUC gains while causing training time increases in our experiments. Thus, the cost asymmetries for the NCS-SVM are $(0, 0.25, 0.5, 0.75, 1)$ and the density levels for NOC-SVM are 11 linearly spaced points from $\lambda_1 = \frac{1}{N} \max_i \sum_j K_{i,j}$ to $\lambda_{11} = 10^{-6}$.

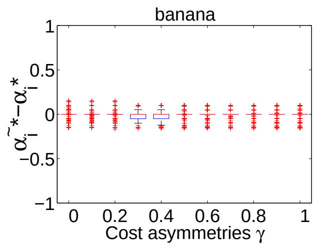
5.5.2 Effects of Interpolation

Nested SVMs are trained on a finite number of cost asymmetries or density levels and then the solution coefficients are linearly interpolated over a continuous range of parameters. Here we illustrate the effectiveness of the linear interpolation scheme of nested SVMs using the two dimensional `banana` data set.

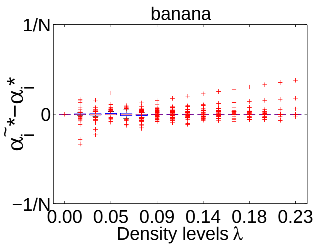
Consider two sets of cost asymmetries, $\widetilde{\boldsymbol{\gamma}} = (0 : 0.25 : 1)$ and $\boldsymbol{\gamma} = (0 : 0.1 : 1)$, with different numbers of breakpoints for the NCS-SVM. Let $\widetilde{\alpha}_i^*(\gamma_m)$ denote the linearly interpolated solution at γ_m from the solution of the NCS-SVM with $\widetilde{\boldsymbol{\gamma}}$, and let $\alpha_i^*(\gamma_m)$ denote the solution from the NCS-SVM with $\boldsymbol{\gamma}$. Fig. 5.3 compares these two solution coefficients $\widetilde{\alpha}_i^*(\gamma_m)$ and $\alpha_i^*(\gamma_m)$. The box plots Fig. 5.3 (a) shows that values of $\widetilde{\alpha}_i^*(\gamma_m) - \alpha_i^*(\gamma_m)$ tend to be very small. Indeed, for most γ_m, the interquartile range on these box plots is not even visible. Regardless of these minor discrepancies, what is most important is that the resulting decision sets are almost indistinguishable as illustrated in Fig. 5.3 (c) and (e). Similar results can be observed in the NOC-SVM as well from Fig. 5.3 (b), (d) and (f). Here we consider two sets of density levels $\widetilde{\boldsymbol{\lambda}}$ with 11 breakpoints and $\boldsymbol{\lambda}$ with 16 breakpoints between $\lambda_1 = \frac{1}{N} \max_i \sum_j K_{i,j}$ and $\lambda_M = 10^{-6}$.

5.5.3 Computational complexity

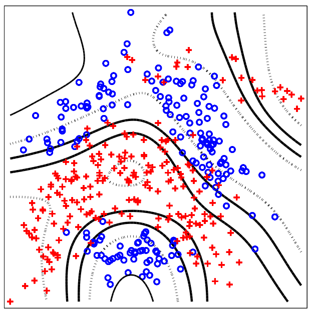
According to Hastie et al. [34], the (non-nested) path following algorithm has $\mathcal{O}(N)$ breakpoints and complexity $\mathcal{O}(m^2N + N^2m)$, where m is the maximum number of points on the margin along the path. On the other hand, our nested SVMs have a controllable number of breakpoints M. To assess the complexity of the nested SVMs, we make a couple of assumptions based on experimental evidence. First, our experience has shown that the number of iterations of the decomposition algorithm is proportional to the number of data points N. Second, we assume that the subproblem, which has M variables, can



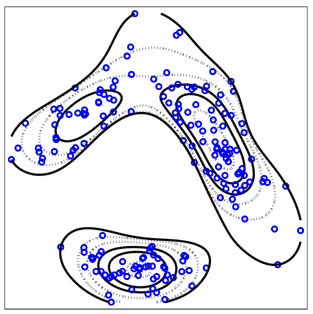
(a) $\widetilde{\alpha}_i^*(\gamma_m) - \alpha_i^*(\gamma_m)$



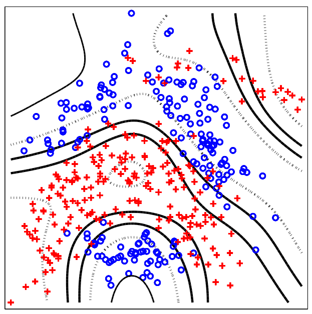
(b) $\widetilde{\alpha}_i^*(\lambda_m) - \alpha_i^*(\lambda_m)$



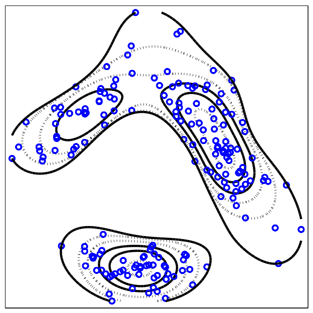
(c) $\widehat{G}_{\gamma_m}(\widetilde{\alpha}_i^*(\gamma_m))$



(d) $\widehat{G}_{\lambda_m}(\widetilde{\alpha}_i^*(\lambda_m))$



(e) $\widehat{G}_{\gamma_m}(\alpha_i^*(\gamma_m))$



(f) $\widehat{G}_{\lambda_m}(\alpha_i^*(\lambda_m))$

Fig. 5.3: Simulation results depicting the impact of interpolation on the coefficients and final set estimates. See Section 5.5.2 for details.

Data set	dim	N_{train}	N_{test}
banana	2	400	4900
breast-cancer	9	200	77
diabetes	8	468	300
flare-solar	9	666	400
german	20	700	300
heart	13	170	100
ringnorm	20	400	7000
thyroid	5	140	75
titanic	3	150	2051
twonorm	20	400	7000
waveform	21	400	4600
image	18	1300	1010
splice	60	1000	2175

Table 5.1: Description of data sets. dim is the number of features, and N_{train} and N_{test} are the numbers of training and test examples.

be solved in $\mathcal{O}(M^2)$ operations. Furthermore, each iteration of the decomposition algorithm also involves a variable selection step. This involves checking all variables for KKT condition violations (as detailed in the Appendices), and thus entails $\mathcal{O}(MN)$ operations. Thus, the computation time of nested SVMs are $\mathcal{O}(M^2N + MN^2)$. In Section 5.6.5, we experimentally compare the run times of the path following algorithms to our methods.

5.6 Experiments and Results

In order to compare the algorithms described above, we experimented on 13 benchmark data sets available online [1] [48]. Their brief summary is provided in Table 5.1. Each feature is standardized with zero mean and unit variance. The first eleven data sets are randomly permuted 100 times (the last two are permuted 20 times) and divided into training and test sets. In all of our experiments, we used the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ and searched for the bandwidth σ over 20 logarithmically spaced points

[1]`http://ida.first.fhg.de/projects/bench/`

from $d_{avg}/15$ to $10\,d_{avg}$ where d_{avg} is the average distance between training data points. This control parameter is selected via 5-fold cross validation on the first 10 permutations, then the average of these values is used to train the remaining permutations.

Each algorithm generates a family of decision functions and set estimates. From these sets, we construct an ROC and compute its area under the curve (AUC). We use the AUC averaged across permutations to compare the performance of algorithms. As shown in Fig. 5.1, however, the set estimates from CS-SVMs or OC-SVMs are not properly nested, and cause ambiguity particularly in ranking. In Section 5.6.3, we measure this violation of the nesting by defining the *ranking disagreement* of two rank scoring functions. Then in Section 5.6.4, we combine this ranking disagreement and the AUC, and compare the algorithms over multiple data sets using the Wilcoxon signed ranks test as suggested in [21].

5.6.1 Two-class Problems

CS-SVMs and NCS-SVMs are compared in two-class problems. For NCS-SVMs, we set $M = 5$ and solved the optimization problem (5.7) at uniformly spaced cost asymmetries $\boldsymbol{\gamma} = (0, 0.25, 0.50, 0.75, 1)$.

In two-class problems, we also searched for the regularization parameter λ over 10 logarithmically spaced points from 0.1 to λ_{max} where λ_{max} is

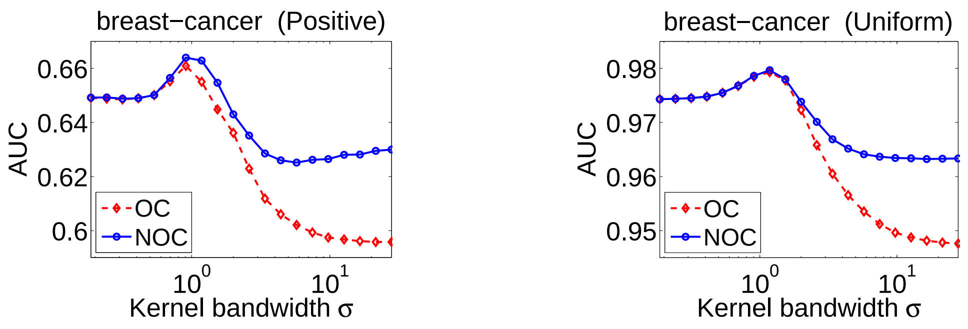
$$\lambda_{max} = \max\left(\max_i \sum_{j \in I_+} y_i y_j K_{i,j},\ \max_i \sum_{j \in I_-} y_i y_j K_{i,j} \right).$$

Values of $\lambda > \lambda_{max}$ do not produce different solutions in the CS-SVM (see Appendix 5.C).

We compared the described algorithms by constructing ROCs and computing their AUCs. The results are collected in Table 5.2. More statistical treatments of these results are covered in Section 5.6.4.

	Two-class		One-class: Positive		One-class: Uniform	
Data Set	CS	NCS	OC	NOC	OC	NOC
banana	0.950 ± 0.009	0.963 ± 0.003	0.919 ± 0.009	0.930 ± 0.007	0.906 ± 0.003	0.911 ± 0.003
breast-cancer	0.733 ± 0.054	0.731 ± 0.056	0.647 ± 0.062	0.654 ± 0.061	0.976 ± 0.006	0.976 ± 0.006
diabetes	0.829 ± 0.016	0.825 ± 0.017	0.722 ± 0.023	0.732 ± 0.022	0.996 ± 0.001	0.996 ± 0.001
flare-solar	0.658 ± 0.040	0.580 ± 0.047	0.601 ± 0.042	0.601 ± 0.043	0.998 ± 0.000	0.998 ± 0.000
german	0.796 ± 0.024	0.788 ± 0.024	0.626 ± 0.031	0.626 ± 0.031	0.991 ± 0.003	0.991 ± 0.003
heart	0.908 ± 0.027	0.907 ± 0.027	0.776 ± 0.037	0.782 ± 0.036	0.986 ± 0.005	0.986 ± 0.005
ringnorm	0.982 ± 0.002	0.955 ± 0.011	0.997 ± 0.000	0.997 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
thyroid	0.962 ± 0.037	0.954 ± 0.037	0.986 ± 0.009	0.987 ± 0.008	0.999 ± 0.000	0.999 ± 0.000
titanic	0.599 ± 0.069	0.597 ± 0.070	0.602 ± 0.068	0.588 ± 0.062	0.761 ± 0.051	0.765 ± 0.041
twonorm	0.997 ± 0.000	0.997 ± 0.000	0.910 ± 0.011	0.912 ± 0.009	1.000 ± 0.000	1.000 ± 0.000
waveform	0.969 ± 0.002	0.967 ± 0.003	0.752 ± 0.019	0.762 ± 0.017	1.000 ± 0.000	1.000 ± 0.000
image	0.991 ± 0.002	0.985 ± 0.004	0.872 ± 0.039	0.854 ± 0.039	1.000 ± 0.000	1.000 ± 0.000
splice	0.950 ± 0.003	0.951 ± 0.004	0.416 ± 0.009	0.415 ± 0.009	0.553 ± 0.012	0.554 ± 0.008

Table 5.2: AUC values for the CS-SVM (CS) and NCS-SVM (NCS) in two-class problems, and OC-SVM (OC) and NOC-SVM (NOC) in one-class problems. In one-class problems, 'Positive' indicates that the alternative hypotheses are from the positive class examples in the data sets, and 'Uniform' indicated that the alternative hypotheses are from a uniform distribution.



(a) Positive (b) Uniform

Fig. 5.4: The effect of kernel bandwidth σ on the performance (AUC). The AUC is evaluated when the alternative class is from the positive class in the data sets (a) and from a uniform distribution (b). The NOC-SVM is less sensitive to σ than the OC-SVM.

5.6.2 One-class Problems

For the NOC-SVM, we selected 11 density levels spaced evenly from $\lambda_{11} = 10^{-6}$ to $\lambda_1 = \frac{1}{N}\max_i \sum_j K_{i,j}$ (see Appendix 5.C). Among the two classes available in each data set, we chose the negative class for training. Because the bandwidth selection step requires computing AUCs, we simulated an artificial second class from a uniform distribution. For evaluation of the trained decision functions, both the positive examples in the test sets and a new uniform sample were used as the alternative class. Table 5.2 reports the results for both cases (denoted by Positive and Uniform, respectively).

Fig. 5.4 shows the AUC of the two algorithms over a range of σ. Throughout the experiments on one-class problems, we observed that the NOC-SVM is more robust to the kernel bandwidth selection than the OC-SVM. However, we did not observe similar results on two-class problems.

5.6.3 Ranking disagreement

The decision sets from the OC-SVM and the CS-SVM are not properly nested, as illustrated in Fig. 5.1. Since larger λ means higher density level, the density level set estimate of the OC-SVM is expected to be contained within the density level set estimate at smaller λ. Likewise, larger γ in the CS-SVM penalizes misclassification of positive examples more; thus, its corresponding positive decision set should contain the decision set at smaller γ, and the two decision boundaries should not cross. This undesired nature of the algorithms leads to non-unique ranking score functions.

In the case of the CS-SVM, we can consider the following two ranking functions:

$$s_+(\mathbf{x}) = 1 - \min_{\{\gamma : f_\gamma(\mathbf{x}) \geq 0\}} \gamma, \qquad s_-(\mathbf{x}) = 1 - \max_{\{\gamma : f_\gamma(\mathbf{x}) \leq 0\}} \gamma. \tag{5.21}$$

For the OC-SVM, we consider the next pair of ranking functions,

$$s_+(\mathbf{x}) = \max_{\{\lambda : \mathbf{x} \in \widehat{G}_\lambda\}} \lambda, \qquad s_-(\mathbf{x}) = \min_{\{\lambda : \mathbf{x} \in \widehat{G}_\lambda\}} \lambda. \tag{5.22}$$

In words, s_+ ranks according to the first set containing a point $\mathbf{x}$ and s_- ranks according to the last set containing the point. In either case, it is easy to see $s_+(\mathbf{x}) \geq s_-(\mathbf{x})$.

In order to quantify the disagreement of the two ranking functions, we define the following measure of *ranking disagreement*:

$$d(s_+, s_-) = \frac{1}{N} \sum_i \max_{j \neq i} \mathbf{1}_{\{(s_+(\mathbf{x}_i) - s_+(\mathbf{x}_j))(s_-(\mathbf{x}_i) - s_-(\mathbf{x}_j)) < 0\}},$$

which is the proportion of data points ambiguously ranked, i.e., ranked differently with respect to at least one other point. Then $d(s_+, s_-) = 0$ if and only if s_+ and s_- induce the same ranking.

With these ranking functions, Table 5.3 reports the ranking disagreements from the CS-SVM and OC-SVM. In the table, d_2 refers to the ranking disagreement of the CS-SVM, and d_p and d_u respectively refer to the ranking disagreement of the OC-SVM when the second class is from the positive samples and from an artificial uniform distribution. As can be seen in the table, for some data sets the violation of the nesting causes severe differences between the above ranking functions.

Data set	$d_2(s_+, s_-)$	$d_p(s_+, s_-)$	$d_u(s_+, s_-)$
banana	0.024	0.498	0.389
breast-cancer	0.013	0.252	0.093
diabetes	0.119	0.020	0.001
flare-solar	0.300	0.657	0.198
german	0.019	0.000	0.000
heart	0.005	0.000	0.000
ringnorm	0.244	0.000	0.000
thyroid	0.002	0.019	0.000
titanic	0.000	0.250	0.231
twonorm	0.006	0.000	0.000
waveform	0.078	0.002	0.001
image	0.307	0.276	0.047
splice	0.105	0.000	0.000

Table 5.3: The measure of disagreement of the two ranking functions from the CS-SVM and OC-SVM. The meaning of each subscript is explained in the text. s_+ and s_- are defined in the equations (5.21) and (5.22).

5.6.4 Statistical comparison

We employ the statistical methodology of Demšar [21] to compare the algorithms across all data sets. Using the Wilcoxon signed ranks test, we compare the CS-SVM and the NCS-SVM for two-class problems, and the OC-SVM and the NOC-SVM for one-class problems.

The Wilcoxon signed ranks test is a non-parametric method testing the significance of differences between paired observations, and can be used to compare the performances between two algorithms over multiple data sets. The difference between the AUCs from the two algorithms are ranked ignoring the signs, and then the ranks of positive and negative differences are added. Table 5.4 and Table 5.5 respectively report the comparison results of the algorithms for two-class problems and one-class problems. Here the numbers under NCS or NOC denote the sums of ranks of the data sets on which the nested SVMs performed better than the original SVMs; the values under CS or OC are for the opposite. T is the smaller of the two sums. For a confidence level of $\alpha = 0.01$ and 13 data sets, the

CS	NCS	T
78	13	13

Table 5.4: Comparison of the AUCs of the two-class problem algorithms: CS-SVM (CS) and NCS-SVM (NCS) using the Wilcoxon signed ranks test (see text for detail.) The test statistic T is greater than the critical difference 9, hence no significant difference is detected in the test.

	OC	NOC	T
Positive	35	56	35
Uniform	21.5	69.5	21.5

Table 5.5: Comparison of the OC-SVM (OC) and NOC-SVM (NOC). In the one-class problems, both cases of alternative hypothesis are considered. Here no significant difference is detected.

difference between algorithms is significant if T is less than or equal to 9 [74]. Therefore, any significant performance difference between the CS-SVM and the NCS-SVM was not detected in the test. Likewise, no difference between the OC-SVM and the NOC-SVM was detected.

However, the AUC alone does not highlight the ranking disagreement of the algorithms. Therefore, we merge the AUC and the disorder measurement, and consider $AUC - d(s_+, s_-)$ for algorithm comparison. Table 5.6 shows the results of the Wilcoxon signed-ranks test using this combined performance measure. From the results, we can observe clearly the performance differences between algorithms. Since the test statistic T is smaller than the critical difference 9, the NCS-SVM outperforms the CS-SVM. Likewise, the performance difference between the OC-SVM and the NOC-SVM is also detected by the Wilcoxon test for both cases of the second class. Therefore, we can conclude that the nested algorithms perform better than their unnested counterparts.

5.6.5 Run time comparison

Table 5.7 shows the average training time for each algorithm. The results for the CS-SVM and OC-SVM are based on our Matlab implementation of solution path al-

CS	NCS	T
4	87	4

	OC	NOC	T
Positive	5	86	5
Uniform	2.5	88.5	2.5

Table 5.6: Comparison of the algorithms based on the AUC along with the *ranking disagreement*. **Left**: CS-SVM and NCS-SVM. **Right**: OC-SVM and NOC-SVM. T is less than the critical values 9, hence the nested SVMs outperforms the original SVMs.

Data set	CS	NCS	OC	NOC
banana	1.01	24.55	0.29	13.03
breast-cancer	0.43	2.42	0.13	9.64
diabetes	2.92	9.80	0.56	75.46
flare-solar	0.17	4.05	0.02	0.85
german	13.25	0.68	4.48	57.69
heart	0.31	7.76	0.07	5.71
ringnorm	3.16	3.43	0.01	2.07
thyroid	0.22	2.74	0.08	6.50
titanic	0.01	0.66	$<$ 0.01	5.69
twonorm	1.89	8.21	0.31	15.29
waveform	1.87	10.42	0.56	26.60
image	40.08	298.98	1.30	64.77
splice	68.43	149.68	0.55	6.06

Table 5.7: Average training times (sec) for the CS-SVM, NCS-SVM, OC-SVM, and NOC-SVM on benchmark data sets. This result is based on our implementation of solution path algorithms for the CS-SVM and OC-SVM.

gorithms [5], [39] available at `http://www.eecs.umich.edu/~cscott/code/svmpath.zip`. We emphasize here that our decomposition algorithm relies on Matlab's `quadprog` function as the basic subproblem solver, and that this function is in no way optimized for our particular subproblem. A discussion of computational complexity was given in Section 5.5.3.

5.7 Primal of Nested SVMs

Although not essential for our approach, we can find a primal optimization problem of the NCS-SVM if we think of (5.7) as a dual problem:

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi}} & \sum_{m=1}^{M} \left[\frac{\lambda}{2}\|\mathbf{w}_m\|^2 + \gamma_m \sum_{I_+} \xi_{i,m} + (1-\gamma_m) \sum_{I_-} \xi_{i,m} \right] \\
\text{s.t.} \quad & \sum_{k=m}^{M} \langle \mathbf{w}_k, \Phi(\mathbf{x}_i) \rangle \geq \sum_{k=m}^{M} (1 - \xi_{i,k}), \quad i \in I_+, \forall m \\
& \sum_{k=1}^{m} \langle \mathbf{w}_k, \Phi(\mathbf{x}_i) \rangle \leq -\sum_{k=1}^{m} (1 - \xi_{i,k}), \quad i \in I_-, \forall m \\
& \xi_{i,m} \geq 0, \quad \forall i, m.
\end{aligned}$$

The derivation of (5.7) from this primal can be found in [41]. Note that the above primal of the NCS-SVM reduces to the primal of the CS-SVM (5.1) when $M = 1$.

Likewise, the primal corresponding to the NOC-SVM is

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi}} & \sum_{m=1}^{M} \left[\frac{\lambda_m}{2}\|\mathbf{w}_m\|^2 + \frac{1}{N} \sum_{i} \xi_{i,m} \right] \\
\text{s.t.} \quad & \sum_{k=m}^{M} \lambda_k \langle \mathbf{w}_k, \Phi(\mathbf{x}_i) \rangle \geq \sum_{k=m}^{M} \lambda_k (1 - \xi_{i,m}), \quad \forall i, m \\
& \xi_{i,m} \geq 0, \quad \forall i, m,
\end{aligned} \tag{5.23}$$

which also boils down to the primal of the OC-SVM (5.4) when $M = 1$.

With these formulations, we can see the geometric meaning of $\mathbf{w}$ and $\boldsymbol{\xi}$. For simplicity, consider (5.23) when $M = 2$:

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi}} & \frac{\lambda_2}{2}\|\mathbf{w}_2\|^2 + \frac{1}{N}\sum_i \xi_{i,2} + \frac{\lambda_1}{2}\|\mathbf{w}_1\|^2 + \frac{1}{N}\sum_i \xi_{i,1} \\
\text{s.t.} \quad & \langle \lambda_2 \mathbf{w}_2, \Phi(\mathbf{x}_i) \rangle \geq \lambda_2 (1 - \xi_{i,2}), \quad \forall i \\
& \langle \lambda_2 \mathbf{w}_2 + \lambda_1 \mathbf{w}_1, \Phi(\mathbf{x}_i) \rangle \geq \lambda_2 (1 - \xi_{i,2}) + \lambda_1 (1 - \xi_{i,1}), \quad \forall i \\
& \xi_{i,m} \geq 0, \quad \forall i, m.
\end{aligned}$$

Here $\xi_{i,1} > 0$ when $\mathbf{x}_i$ lies between the hyperplane $P_{\frac{\lambda_2 \mathbf{w}_2 + \lambda_1 \mathbf{w}_1}{\lambda_2 + \lambda_1}}$ and the origin, and $\xi_{i,2} > 0$ when the point lies between $P_{\mathbf{w}_2}$ and the origin where we used $P_{\mathbf{w}}$ to denote $\{\Phi(\mathbf{x}) : \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle = 1\}$, a hyperplane in $\mathcal{H}$. Note that from the nesting structure, the hyperplane $P_{\frac{\lambda_2 \mathbf{w}_2 + \lambda_1 \mathbf{w}_1}{\lambda_2 + \lambda_1}}$ is located between $P_{\mathbf{w}_1}$ and $P_{\mathbf{w}_2}$. Then we can show that $\frac{\lambda_1 \xi_{i,1} + \lambda_2 \xi_{i,2}}{\|\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2\|}$ is the distance between the point $\mathbf{x}_i$ and the hyperplane $P_{\frac{\lambda_2 \mathbf{w}_2 + \lambda_1 \mathbf{w}_1}{\lambda_2 + \lambda_1}}$.

5.8 Conclusion

In this chapter, we introduced a novel framework for building a family of nested support vector machines for the tasks of cost-sensitive classification and density level set estimation. Our approach involves forming new quadratic programs inspired by the cost-sensitive and one-class SVMs, with additional constraints that enforce nesting structure. Our construction generates a finite number of nested set estimates at a pre-selected set of parameter values, and linearly interpolates these sets to a continuous nested family. We also developed efficient algorithms to solve the proposed quadratic problems. Thus, the NCS-SVM yields a family of nested classifiers indexed by cost asymmetry γ, and the NOC-SVM yields a family of nested density level set estimates indexed by density level λ. Unlike the original SVMs, which are not nested, our methods can be readily applied to problems requiring multiple set estimation including clustering, ranking, and anomaly detection.

In experimental evaluations, we found that non-nested SVMs can yield highly ambiguous rankings for many datasets, and that nested SVMs offer considerable improvements in this regard. Nested SVMs also exhibit greater stability with respect to model selection criteria such as cross-validation. In terms of area under the ROC (AUC), we found that enforcement of nesting appears to have a bigger impact on one-class problems. However, neither cost-sensitive nor one-class classification problems displayed significantly different AUC values between nested and non-nested methods.

The statistical consistency of our nested SVMs is an interesting open question. Such a result would likely depend on the consistency of the original CS-SVM or OC-SVM at fixed values of γ or λ, respectively. We are unaware of consistency results for the CS-SVM at fixed γ [65]. However, consistency of the OC-SVM for fixed λ has been established [73]. Thus, suppose $\widehat{G}_{\lambda_1}, \dots, \widehat{G}_{\lambda_M}$ are (non-nested) OC-SVMs at a grid of points. Since these estimators are each consistent, and the true levels sets they approximate are nested, it seems plausible that for a sufficiently large sample size, these OC-SVMs are also nested. In this case, they would be feasible for the NOC-SVM, which would suggest that the NOC-SVM estimates the true level sets at least as well, asymptotically, at these estimates. Taking the grid of levels $\{\lambda_i\}$ to be increasingly dense, the error of the interpolation scheme should also vanish. We leave it as future work to determine whether this intuition can be formalized.

5.A Appendix: Data Point Selection and Termination Condition of NCS-SVM

On each round, the algorithm in Algorithm 5.1 selects an example $\mathbf{x}_i$, updates its corresponding variables $\{\alpha_{i,m}\}_{m=1}^{M}$, and checks the termination condition. In this appendix, we employ the KKT conditions to derive an efficient variable selection strategy and a termination condition of NCS-SVM.

We use the KKT conditions to find the necessary conditions of the optimal solution of (5.7). Before we proceed, we define $\alpha_{i,0} = 0$ for $i \in I_+$ and $\alpha_{i,M+1} = 0$ for $i \in I_-$ for notational convenience. Then the Lagrangian of the quadratic program is

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha}, \mathbf{u}, \mathbf{v}) = &\sum_m \left[\frac{1}{2\lambda} \sum_{i,j} \alpha_{i,m}\alpha_{j,m} y_i y_j K_{i,j} - \sum_i \alpha_{i,m} \right] \\
&+ \sum_m \sum_i u_{i,m}(\alpha_{i,m} - \mathbf{1}_{\{y_i<0\}} - y_i\gamma_m) \\
&+ \sum_m \sum_{i \in I_+} v_{i,m}(\alpha_{i,m-1} - \alpha_{i,m}) \\
&- \sum_m \sum_{i \in I_-} v_{i,m}(\alpha_{i,m} - \alpha_{i,m+1})
\end{aligned}$$

where $u_{i,m} \geq 0$ and $v_{i,m} \geq 0$ for $\forall i, m$. At the global minimum, the derivative of the Lagrangian with respect to $\alpha_{i,m}$ vanishes

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \alpha_{i,m}} &= y_i f_{i,m} - 1 + u_{i,m} \begin{cases} -v_{i,m} + v_{i,m+1}, & i \in I_+ \\ +v_{i,m-1} - v_{i,m}, & i \in I_- \end{cases} \\
&= 0
\end{aligned} \tag{5.24}$$

where, recall, $f_{i,m} = \frac{1}{\lambda}\sum_j \alpha_{j,m} y_j K_{i,j}$ and we introduced auxiliary variables $v_{i,M+1} = 0$ for $i \in I_+$ and $v_{i,0} = 0$ for $i \in I_-$. Then we obtain the following set of constraints from the KKT conditions

$$y_i f_{i,m} - 1 + u_{i,m} = \begin{cases} v_{i,m} - v_{i,m+1}, & i \in I_+ \\ -v_{i,m-1} + v_{i,m}, & i \in I_- \end{cases} \tag{5.25}$$

$$0 \leq \alpha_{i,m} \leq \mathbf{1}_{\{y_i<0\}} + y_i\gamma_m, \qquad \forall i, \forall m \tag{5.26}$$

$$y_i\alpha_{i,1} \leq y_i\alpha_{i,2} \leq \cdots \leq y_i\alpha_{i,M}, \qquad \forall i \tag{5.27}$$

$$u_{i,m}\left(\alpha_{i,m} - \mathbf{1}_{\{y_i<0\}} - y_i\gamma_m\right) = 0, \qquad \forall i, \forall m \tag{5.28}$$

$$v_{i,m}(\alpha_{i,m-1} - \alpha_{i,m}) = 0, \qquad i \in I_+, \forall m \tag{5.29}$$

$$v_{i,m}(\alpha_{i,m} - \alpha_{i,m+1}) = 0, \qquad i \in I_-, \forall m \tag{5.30}$$

$$u_{i,m} \geq 0, \qquad v_{i,m} \geq 0, \qquad \forall i, m. \tag{5.31}$$

Since (5.7) is a convex program, the KKT conditions are also sufficient [7]. That is, $\alpha_{i,m}$, $u_{i,m}$, and $v_{i,m}$ satisfying (5.25)-(5.31) is indeed optimal. Therefore, at the end of each iteration, we assess a current solution with these conditions and decide whether to stop or to continue. We evaluate the amount of error for $\mathbf{x}_i$ by defining

$$e_i = \sum_m \left| \frac{\partial \mathcal{L}}{\partial \alpha_{i,m}} \right|, \quad \forall i.$$

An optimal solution makes these quantities zero. In practice, when their sum $\sum_i e_i$ decreases below a predetermined tolerance, the algorithm stops and returns the current solution. If not, the algorithm chooses the example with the largest e_i and continues the loop.

Computing e_i involves unknown variables $u_{i,m}$ and $v_{i,m}$ (see (5.24)), whereas $f_{i,m}$ can be easily computed from the known variables $\alpha_{i,m}$. Table 5.8 and Table 5.9 are for determining these $u_{i,m}$ and $v_{i,m}$. These tables are obtained by firstly assuming the current solution $\alpha_{i,m}$ is optimal and secondly solving $u_{i,m}$ and $v_{i,m}$ such that they satisfy the KKT conditions. Thus, depending on the value $\alpha_{i,m}$ between its upper and lower bounds, $u_{i,m}$ and $v_{i,m}$ can be simply set as directed in the tables. For example, if $i \in I_+$, then we find $u_{i,m}$ and $v_{i,m}$ by referring Table 5.8 iteratively from $m = M$ down to $m = 1$. If $i \in I_-$, we use Table 5.9 and iterate from $m = 1$ up to $m = M$. Then the obtained e_i takes a non-zero value only when the assumption is false and the current solution is sub-optimal.

	$\alpha_{i,m-1} < \alpha_{i,m}$	$\alpha_{i,m-1} = \alpha_{i,m}$
$\alpha_{i,m} < \min(\gamma_m, \alpha_{i,m+1})$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(f_{i,m} - 1, 0)$
$\alpha_{i,m} = \gamma_m < \alpha_{i,m+1}$	$u_{i,m} = \max(1 - f_{i,m}, 0)$ $v_{i,m} = 0$	- -
$\alpha_{i,m} = \alpha_{i,m+1} < \gamma_m$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(f_{i,m} - 1 + v_{i,m+1}, 0)$
$\alpha_{i,m} = \alpha_{i,m+1} = \gamma_m$	$u_{i,m} = \max(1 - f_{i,m} - v_{i,m+1}, 0)$ $v_{i,m} = 0$	- -

	$\alpha_{i,M-1} < \alpha_{i,M}$	$\alpha_{i,M-1} = \alpha_{i,M}$
$\alpha_{i,M} < \gamma_M$	$u_{i,M} = 0$ $v_{i,M} = 0$	$u_{i,M} = 0$ $v_{i,M} = \max(f_{i,M} - 1, 0)$
$\alpha_{i,M} = \gamma_M$	$u_{i,M} = \max(1 - f_{i,M}, 0)$ $v_{i,M} = 0$	- -

Table 5.8: The optimality conditions of NCS-SVM when $i \in I_+$. (Upper: $m = 1, 2, \ldots, M-1$, Lower: $m = M$.) Assuming $\alpha_{i,m}$ are optimal, $u_{i,m}$ and $v_{i,m}$ are solved as above from the KKT conditions. Empty entries indicate cases that cannot occur.

	$\alpha_{i,m+1} < \alpha_{i,m}$	$\alpha_{i,m+1} = \alpha_{i,m}$
$\alpha_{i,m} < \min(1-\gamma_m, \alpha_{i,m-1})$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(-f_{i,m} - 1, 0)$
$\alpha_{i,m} = 1-\gamma_m < \alpha_{i,m-1}$	$u_{i,m} = \max(1 + f_{i,m}, 0)$ $v_{i,m} = 0$	- -
$\alpha_{i,m} = \alpha_{i,m-1} < 1-\gamma_m$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(-f_{i,m} - 1 + v_{i,m-1}, 0)$
$\alpha_{i,m} = \alpha_{i,m-1} = 1-\gamma_m$	$u_{i,m} = \max(1 + f_{i,m} - v_{i,m-1}, 0)$ $v_{i,m} = 0$	- -

	$\alpha_{i,2} < \alpha_{i,1}$	$\alpha_{i,2} = \alpha_{i,1}$
$\alpha_{i,1} < 1-\gamma_1$	$u_{i,1} = 0$ $v_{i,1} = 0$	$u_{i,1} = 0$ $v_{i,1} = \max(-f_{i,1} - 1, 0)$
$\alpha_{i,1} = 1-\gamma_1$	$u_{i,1} = \max(1 + f_{i,1}, 0)$ $v_{i,1} = 0$	- -

Table 5.9: The optimality conditions of NCS-SVM when $i \in I_-$. (Upper: $m = 2, \ldots, M$, Lower: $m = 1$.)

5.B Appendix: Data Point Selection and Termination Condition of NOC-SVM

As in NCS-SVM, we investigate the optimality condition of NOC-SVM (5.12) and find a data point selection method and a termination condition.

With a slight modification, we rewrite (5.12),

$$\min_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M} \sum_{m=1}^{M} \left[\frac{1}{2\lambda_m} \sum_{i,j} \alpha_{i,m}\alpha_{j,m}K_{i,j} - \sum_i \alpha_{i,m} \right] \tag{5.32}$$

$$\text{s.t.} \quad \alpha_{i,m} \le \frac{1}{N}, \quad \forall i, m$$

$$0 \le \frac{\alpha_{i,1}}{\lambda_1} \le \frac{\alpha_{i,2}}{\lambda_2} \le \cdots \le \frac{\alpha_{i,M}}{\lambda_M}, \quad \forall i.$$

We then use the KKT conditions to find the necessary conditions of the optimal solution of (5.32). The Lagrangian is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}, \mathbf{u}, \mathbf{v}) = &\sum_{m=1}^{M} \left[\frac{1}{2\lambda_m} \sum_{i,j} \alpha_{i,m}\alpha_{j,m}K_{i,j} - \sum_i \alpha_{i,m} \right] \\ &+ \sum_{m=1}^{M} \sum_i u_{i,m}(\alpha_{i,m} - \frac{1}{N}) - \sum_i v_{i,1} \frac{\alpha_{i,1}}{\lambda_1} \\ &+ \sum_i \sum_{m=2}^{M} v_{i,m} \left(\frac{\alpha_{i,m-1}}{\lambda_{m-1}} - \frac{\alpha_{i,m}}{\lambda_m} \right)\end{aligned}$$

where $u_{i,m} \ge 0$ and $v_{i,m} \ge 0$ for $\forall i, m$. At the global minimum, the derivative of the Lagrangian with respect to $\alpha_{i,m}$ vanishes

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_{i,m}} &= f_{i,m} - 1 + u_{i,m} \begin{cases} -\frac{v_{i,m}}{\lambda_m} + \frac{v_{i,m+1}}{\lambda_m}, & m \neq M \\ -\frac{v_{i,M}}{\lambda_M}, & m = M \end{cases} \\ &= 0.\end{aligned} \tag{5.33}$$

where, recall, $f_{i,m} = \frac{1}{\lambda_m} \sum_j \alpha_{j,m} K_{i,j}$. Then, from the KKT conditions, we obtain the following set of constraints for $\mathbf{x}_i$:

$$f_{i,m} - 1 + u_{i,m} = \begin{cases} \frac{v_{i,m}}{\lambda_m} - \frac{v_{i,m+1}}{\lambda_m}, & m \neq M \\ \frac{v_{i,M}}{\lambda_M}, & m = M \end{cases} \tag{5.34}$$

$$\alpha_{i,m} \leq \frac{1}{N}, \qquad \forall m \tag{5.35}$$

$$0 \leq \frac{\alpha_{i,1}}{\lambda_1} \leq \frac{\alpha_{i,2}}{\lambda_2} \leq \cdots \leq \frac{\alpha_{i,M}}{\lambda_M} \tag{5.36}$$

$$u_{i,m}(\alpha_{i,m} - \frac{1}{N}) = 0, \qquad \forall m \tag{5.37}$$

$$v_{i,m}(\frac{\alpha_{i,m-1}}{\lambda_{m-1}} - \frac{\alpha_{i,m}}{\lambda_m}) = 0, \qquad \forall m \tag{5.38}$$

$$u_{i,m} \geq 0, \qquad v_{i,m} \geq 0, \qquad \forall m. \tag{5.39}$$

Since (5.32) is a convex program, the KKT conditions are sufficient [7]. That is, $\alpha_{i,m}$, $u_{i,m}$, and $v_{i,m}$ satisfying (5.34)-(5.39) is indeed optimal. Therefore, at the end of each iteration, we assess a current solution with these conditions and decide whether to stop or to continue. We evaluate the amount of error for $\mathbf{x}_i$ by defining

$$e_i = \sum_m \left| \frac{\partial \mathcal{L}}{\partial \alpha_{i,m}} \right|, \quad \forall i.$$

An optimal solution makes these quantities zero. In practice, when their sum $\sum_i e_i$ decreases below a predetermined tolerance, the algorithm stops and returns the current solution. If not, the algorithm chooses the example with the largest e_i and continues the loop.

Computing e_i involves unknown variables $u_{i,m}$ and $v_{i,m}$ (see (5.33)), whereas $f_{i,m}$ can be easily computed from the known variables $\alpha_{i,m}$. Table 5.10 are for determining these $u_{i,m}$ and $v_{i,m}$. These tables are obtained by firstly assuming the current solution $\alpha_{i,m}$ is optimal and secondly solving $u_{i,m}$ and $v_{i,m}$ such that they satisfy the KKT conditions. Thus, depending on the value $\alpha_{i,m}$ between its upper and lower bounds, $u_{i,m}$ and $v_{i,m}$ can be simply set by referring Table 5.10 iteratively from $m = M$ down to $m = 1$. Then the obtained e_i takes a non-zero value only when the assumption is false and the current solution is not optimal.

	$\frac{\lambda_m}{\lambda_{m-1}}\alpha_{i,m-1} < \alpha_{i,m}$	$\frac{\lambda_m}{\lambda_{m-1}}\alpha_{i,m-1} = \alpha_{i,m}$
$\alpha_{i,m} < \min(\frac{1}{N}, \frac{\lambda_m}{\lambda_{m+1}}\alpha_{i,m+1})$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(\lambda_m(f_{i,m} - 1), 0)$
$\alpha_{i,m} = \frac{1}{N} < \frac{\lambda_m}{\lambda_{m+1}}\alpha_{i,m+1}$	$u_{i,m} = \max(1 - f_{i,m}, 0)$ $v_{i,m} = 0$	- -
$\alpha_{i,m} = \frac{\lambda_m}{\lambda_{m+1}}\alpha_{i,m+1} < \frac{1}{N}$	$u_{i,m} = 0$ $v_{i,m} = 0$	$u_{i,m} = 0$ $v_{i,m} = \max(\lambda_m(f_{i,m} - 1 + \frac{v_{i,m+1}}{\lambda_m}), 0)$
$\alpha_{i,m} = \frac{\lambda_m}{\lambda_{m+1}}\alpha_{i,m+1} = \frac{1}{N}$	$u_{i,m} = \max(1 - f_{i,m} - \frac{v_{i,m+1}}{\lambda_m}, 0)$ $v_{i,m} = 0$	- -

	$\frac{\lambda_M}{\lambda_{M-1}}\alpha_{i,M-1} < \alpha_{i,M}$	$\frac{\lambda_M}{\lambda_{M-1}}\alpha_{i,M-1} = \alpha_{i,M}$
$\alpha_{i,M} < \frac{1}{N}$	$u_{i,M} = 0$ $v_{i,M} = 0$	$u_{i,M} = 0$ $v_{i,M} = \max(\lambda_M(f_{i,M} - 1), 0)$
$\alpha_{i,M} = \frac{1}{N}$	$u_{i,M} = \max(1 - f_{i,M}, 0)$ $v_{i,M} = 0$	- -

Table 5.10: The optimality conditions of NOC-SVM. (Upper: $m = 1, 2, \ldots, M-1$, and Lower: $m = M$.) Empty entries indicate cases that cannot occur.

5.C Appendix: Maximum value of λ of CS-SVM and OC-SVM

In this appendix, we find the values of the regularization parameter λ over which OC-SVM or CS-SVM generate the same solutions.

First, we consider OC-SVM. The decision function of OC-SVM is given by $f_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x})$, and $f_\lambda(\mathbf{x}) = 1$ forms the margin. For sufficiently large λ, every data point $\mathbf{x}_i$ falls inside the margin ($f_\lambda(\mathbf{x}_i) \leq 1$). Since the KKT optimality conditions of (5.4) imply $\alpha_i = \frac{1}{N}$ for the data points such that $f_\lambda(\mathbf{x}_i) < 1$, we obtain $\lambda \geq \frac{1}{N}\sum_j K_{i,j}$ for $\forall i$. Therefore, if the maximum row sum of the kernel matrix is denoted as $\lambda_{OC} = \max_i \frac{1}{N}\sum_j K_{i,j}$, then for any $\lambda \geq \lambda_{OC}$, the optimal solution of OC-SVM becomes $\alpha_i = \frac{1}{N}$ for $\forall i$.

Next, we consider the regularization parameter λ of in the formulation (5.1) of CS-SVM. The decision function of CS-SVM is $f_\gamma(\mathbf{x}) = \frac{1}{\lambda}\sum_j \alpha_j y_j k(\mathbf{x}_j, \mathbf{x})$, and the margin is $y f_\gamma(\mathbf{x}) = 1$. Thus, if λ is sufficiently large, all the data points are inside the margin and satisfy $y_i f_\gamma(\mathbf{x}_i) \leq 1$. Then $\lambda \geq \sum_{j \in I_+} \gamma y_i y_j K_{i,j} + \sum_{j \in I_-}(1-\gamma) y_i y_j K_{i,j}$ for $\forall i$ because $\alpha_i = \mathbf{1}_{\{y_i < 0\}} + y_i \gamma$ for all the data points such that $y_i f_\gamma(\mathbf{x}_i) < 1$ from the KKT conditions. For a given γ, let

$$\lambda_{CS}(\gamma) = \max_i \left[\gamma \sum_{j \in I_+} y_i y_j K_{i,j} + (1-\gamma) \sum_{j \in I_-} y_i y_j K_{i,j} \right].$$

Then for $\lambda > \lambda_{CS}(\gamma)$, the solution of CS-SVM becomes $\alpha_i = \mathbf{1}_{\{y_i<0\}} + y_i \gamma$ for $\forall i$. Therefore, since $\lambda_{CS}(\gamma) \leq (1-\gamma)\lambda_{CS}(0) + \gamma \lambda_{CS}(1)$ for all $\gamma \in [0, 1]$, values of $\lambda > \max(\lambda_{CS}(0), \lambda_{CS}(1))$ generate the same solutions in CS-SVM.

CHAPTER 6

Conclusions and Future Work

In this thesis, we have applied machine learning techniques to the problem of automatic flow cytometry data analysis. Conventional analysis of flow cytometry data is based on primitive tools and usually performed by hand. This has been recognized as a labor-intensive, time-consuming, and highly subjective process. Furthermore, addressing the challenges of biological and/or technical variations is also an important issue for developing automatic analysis tools for biomedical data. A machine learning-based framework enables the development of high-throughput analysis pipelines for flow cytometry data and may help in establishing standardized and objective analytic procedures.

We began by presenting the work on file matching of flow cytometry data. We proposed cluster-based nearest neighbor imputation method (Cluster-NN). As opposed to the previous approach that only uses common markers of multiple assays, our approach improves the file matching of flow cytometry data and is less likely to induce spurious clusters. We have demonstrated the ability of the proposed algorithm on real flow cytometry data to generate a dataset of higher dimension by merging multiple data files of lower dimensions. The evaluation showed a high degree of agreement between the synthesized dataset and the measured dataset. We emphasize that the use of the proposed imputation algorithm and its associated clustering algorithm is not limited to the file matching problems. Whereas a file matching problem has a particular missing variable pattern, the developed methods are more general and can solve problems that have arbitrary missing variable patterns.

Given a flow cytometry data, a typical analysis process begins by separating cell populations into regions and identifying interesting subsets. We have presented an unsupervised learning technique to automate this gating process. A common approach for this task is clustering flow cytometry data by modeling the entire cell measurements with mixture of distributions. We addressed this problem by fitting multivariate Gaussian mixture models. On the contrary to the standard EM algorithms, our approach can handle the truncated and censored measurements that are present in the flow cytometry data. The performance of the proposed algorithms has been shown for the simulated data as well as the clinical flow cytometry data. The truncated and censored data EM algorithms could effectively correct the biases from the data deformations and outperform the standard EM algorithms. Since interpretation of gating results often depends on many statistical features such as proportions, locations and spreads of cell populations, these results suggest that our methods will be useful for producing more reliable analysis results.

Finally, we have reformulated the automatic gating problem into a transfer learning problem. By combining the transfer learning and the low-density separation principle, our approach required no labeled training data for the target task unlike conventional supervised learning techniques. Furthermore, the presented method could take advantage of previous expert-gated data sets to solve a new gating problem. The ability of the presented set estimation algorithm based on low-density separation was demonstrated on the tasks of automatically identifying lymphocytes in peripheral blood samples. The results provided empirical evidence that our strategy was able to successfully replicate what experts do in the field. While biological and technical variation pose challenges in developing fully automated analysis pipelines, our solution shows how a transfer learning technique can be used to facilitate previous solutions to similar problems and design algorithms that can tune themselves automatically to a new problem with no human intervention.

In future work, we would like to verify the effectiveness of our analytic tools through rigorous experiments with extensive flow cytometry data and relate the results with clinical assessments of pathologists. Exploratory analysis with high-dimensional research-grade flow cytometry data would also be helpful to have insight on the robustness of the algorithms. While we presented empirical evidence in this thesis, more rigorous error analysis of the presented algorithms will support the usefulness of our transfer learning approach. Additionally, we would like to provide more general kernel-based framework for automatic gating problems. This involves discovering kernels that measure similarity between two gating tasks based on their marginal distributions. This kernel will naturally account for biological and technical variations in multiple data sets. Finally, extending the algorithms developed for flow cytometry data analysis to different kinds of biomedical data application is also an interesting future direction.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the roc curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

[2] R. K. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 05)*, pages 1–9, 2005.

[3] A. Arnold, R. Nallapati, and W.W. Cohen. A comparative study of methods for transductive transfer learning. *Seventh IEEE International Conference on Data Mining Workshops*, pages 77–82, 2007.

[4] Scott E. Atkinson. The performance of standard and hybrid EM algorithms for ML estimates of the normal mixture model with censoring. *Journal of Statistical Computation and Simulation*, 44:105–115, 1992.

[5] F. R. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7:1713–1741, 2006.

[6] A. Bashashati and R. R. Brinkman. A survey of flow cytometry data analysis methods. *Advances in Bioinformatics*, 2009:Article ID 584603, 2009.

[7] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonliear Programming: Theory and Algorithms*. John Wiley & Sons, Hoboken, NJ, USA, third edition, 2006.

[8] M. J. Boedigheimer and J. Ferbas. Mixture modeling approach to flow cytometry data. *Cytometry Part A*, 73:421 – 429, 2008.

[9] M. Brown and C. Wittwer. Flow cytometry: Principles and clinical applications in hematology. *Clinical Chemistry*, 46:1221–1229, 2000.

[10] Igor V. Cadez, Padhraic Smyth, Geoff J. MacLachlan, and Christine E. McLaren. Maximum likelihood estimation of mixture densities for binned and truncated multivariate data. *Machine Learning*, 47:7–34, 2002.

[11] N. A. Campbell. Robust procedures in multivariate analysis I: Robust covariance estimation. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29:231–237, 1980.

[12] K. Carter, R. Raich, W.G. Finn, and A. O. Hero. Fine: Fisher information non-parametric embedding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009. to appear.

[13] K. Carter, R. Raich, W.G. Finn, and A. O. Hero. Information preserving component analysis: data projections for flow cytometry analysis. *Journ. of Selected Topics in Signal Processing*, 3:148–158, 2009.

[14] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[15] C. Chan, F. Feng, J. Ottinger, D. Foster, M. West, and T.B. Kepler. Statistical mixture modeling for cell subtype identification in flow cytometry. *Cytometry Part A*, 73:693–701, 2008.

[16] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Didier Chauveau. A stochastic EM algorithm for mixtures with censored data. *Journal of Statistical Planning and Inference*, 46:1–25, 1995.

[18] Stéphan J.M. Clémençon and Nicolas Vayatis. Overlaying classifiers: a practical approach for optimal ranking. *Advances in Neural Information Processing Systems 21*, 21:313–320, 2009.

[19] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[21] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[22] Z. Drezner and G. O. Wesolowsky. On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation*, 35:101–107, 1989.

[23] T. Evgeniou and M. Pontil. Regularized multi–task learning. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 04)*, pages 109–117, 2004.

[24] Greg Finak, Ali Bashashati, Ryan Brinkman, and Raphael Gottardo. Merging mixture components for cell population identification in flow cytometry. *Advances in Bioinformatics*, 2009, 2009.

[25] Greg Finak, Juan-Manuel Perez, Andrew Weng, and Raphael Gottardo. Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*, 11:546, 2010.

[26] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41:578–588, 1998.

[27] A. Gelman. Parameterization and bayesian modeling. *Journal of the American Statistical Association*, 99(466):537–544, 2004.

[28] A. Genz. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing*, 14:251–260, 2004.

[29] A. Genz and F. Bretz. Numerical computation of multivariate t probabilities with application to power calculation of multiple contrasts. *Journal of Statistical Computation and Simulation*, 63:361–378, 1999.

[30] A. Genz and F. Bretz. Comparison of methods for the computation of multivariate t probabilities. *Journal of Computational and Graphical Statistics*, 11:950–971, 2002.

[31] Z. Ghahramani and M.I. Jordan. Supervised learning from incomplete data via an EM approach. *Advances in Neural Information Processing Systems*, 6, 1994.

[32] Florian Hahne, Alireza Hadj Khodabakhshi, Ali Bashashati, Chao-Jen Wong, Randy D. Gascoyne, Andrew P. Weng, Vicky Seyfert-Margolis, Katarzyna Bourcier, Adam Asare, Thomas Lumley, Robert Gentleman, and Ryan R. Brinkman. Per-channel basis normalization methods for flow cytometry data. *Cytometry Part A*, 77A:121–131, 2010.

[33] J. A. Hartigan. Consistency of single linkage for high-density clusters. *J. of the American Stat. Association*, 76:388–394, 1981.

[34] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[35] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

[36] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[37] V. Kecman. *Learning and Soft Computing, Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, Cambridge, MA, 2001.

[38] J. Lakoumentas, J. Drakos, M. Karakantza, G. C. Nikiforidis, and G. C. Sakellaropoulos. Bayesian clustering of flow cytometry data for the diagnosis of B-chronic lymphocytic leukemia. *Journal of Biomedical Informatics*, 42:251–261, 2009.

[39] Gyemin Lee and Clayton Scott. The one class support vector machine solution path. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, volume 2, pages II–521–II–524, 2007.

[40] Gyemin Lee and Clayton Scott. Nested support vector machines. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, pages 1985–1988, 2008.

[41] Gyemin Lee and Clayton Scott. Nested support vector machines. Technical report, 2008. `http://www.eecs.umich.edu/~cscott`.

[42] Roderick Little and Donald Rubin. *Statistical analysis with missing data*. Wiley, 2nd edition, 2002.

[43] R. Liu, J. Parelius, and K. Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Annals of Statistics*, 27:783–858, 1999.

[44] K. Lo, R. R. Brinkman, and R. Gottardo. Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A*, 73:321 – 332, 2008.

[45] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, second edition, 1999.

[46] B. G. Manjunath and S. Wilhelm. Moments calculation for the double truncated multivariate normal density (working paper). 2009.

[47] G. J. McLachlan and P. N. Jones. Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics*, 44:571–578, 1988.

[48] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12:181–201, 2001.

[49] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT Artificial Intelligence Laboratory, Mar 1997.

[50] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[51] Carlos E. Pedreira, Elaine S. Costa, Susana Barrena, Quentin Lecrevisse, Julia Almeida, Jacques J. M. van Dongen, and Alberto Orfao. Generation of flow cytometry data files with a potentially infinite number of dimensions. *Cytometry A*, 73A:834–846, 2008.

[52] S.P. Perfetto, P. K. Chattopadhyay, and M. Roederer. Seventeen-colour flow cytometry: unravelling the immune system. *Nature Reviews Immunology*, 4:648–655, 2004.

[53] S. Pyne, X. Hu, K. Wang, E. Rossin, T. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. De Jager, and J. P. Mesirov. Automated high-dimensional flow cytometric data analysis. *PNAS*, 106:8519–8524, 2009.

[54] Susanne Rässler. *Statistical matching: a frequentist theory, practical applications, and alternative Bayesian approaches*. Number 168 in Lecture Notes in Statistics. Springer, 2002.

[55] A. Rettinger, M. Zinkevich, and M. Bowling. Boosting expert ensembles for rapid concept recall. *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 06)*, 1:464–469, 2006.

[56] ML Sánchez, J Almeida, B Vidriales, MC López-Berges, MA García-Marcos, MJ Moro, A Corrales, MJ Calmuntia, JF San Miguel, and A Orfao. Incidence of phenotypic aberrations in a series of 467 patients with B chronic lymphoproliferative disorders: basis for the design of specific four-color stainings to be used for minimal residual disease investigation. *Leukemia*, 16:1460–1469, 2002.

[57] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1472, 2001.

[58] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[59] C. Scott and E. D. Kolaczyk. Annotated minimum volume sets for nonparametric anomaly discovery. In *IEEE Workshop on Statistical Signal Processing*, pages 234–238, 2007.

[60] C. Scott and R. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Trans. Inf. Theory*, 51:3806–3819, 2005.

[61] C. Scott and R. Nowak. Learning minimum volume sets. *Journal of Machine Learning Research*, 7:665–704, 2006.

[62] Clayton Scott and Gilles Blanchard. Novelty detection: Unlabeled data definitely help. *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 5:464–471, 2009.

[63] H. Shapiro. *Practical Flow Cytometry*. Wiley-Liss, 3rd edition, 1994.

[64] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[65] I. Steinwart. How to compare different loss functions and their risks. *Constructive Approximation*, 26:225–287, 2007.

[66] W. Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification*, 20(5):25–47, 2003.

[67] G. M. Tallis. The moment generating function of the truncated multi-normal distribution. *Journal of the Royal Statistical Society. Series B (Methodological)*, 23:223–229, 1961.

[68] D.M.J. Tax and R.P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[69] S. Thrun. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

[70] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysis. *Neural Computation*, 11:443–482, 1999.

[71] M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, 6:611–622, 1999.

[72] J. Toedling, P. Rhein, R. Ratei, L. Karawajew, and R. Spang. Automated in-silico detection of cell populations in flow cytometry readouts and its application to leukemia disease monitoring. *BMC Bioinformatics*, 7:282, 2006.

[73] R. Vert and J. Vert. Consistency and convergence rates of one-class SVMs and related algorithms. *Journal of Machine Learning Research*, 7:817–854, 2006.

[74] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, pages 1:80–83, 1945.

[75] Heddy Zola, Bernadette Swart, Ian Nicholson, Bent Aasted, Armand Bensussan, Laurence Boumsell, Chris Buckley, Georgina Clark, Karel Drbal, Pablo Engel, Derek Hart, Vaclav Horejsi, Clare Isacke, Peter Macardle, Fabio Malavasi, David Mason, Daniel Olive, Armin Saalmueller, Stuart F. Schlossman, Reinhard Schwartz-Albiez, Paul Simmons, Thomas F. Tedder, Mariagrazia Uguccioni, and Hilary Warren. CD molecules 2005: human cell differentiation molecules. *Blood*, 106:3123–3126, 2005.