Supporting Information for "Automated High-frequency Geomagnetic Disturbance Classifier: A Machine Learning Approach to Identifying Noise while Retaining High-Frequency Components of the Geomagnetic Field"

Brett A. McCuen[1], Mark B. Moldwin[1],Mark J. Engebretson[2], Erik S. Steinmetz[2],

[1]Department of Climate and Space Sciences and Engineering, University of Michigan, Ann Arbor, MI, USA

[2]Department of Physics, Augsburg University, Minneapolis, MN, USA

Contents of this file

1. Text S1: Information on four machine learning algorithms and the process of tuning/testing each

2. Table S1: Details and Accuracy scores for four machine learning algorithms

Introduction

The supporting information in this file describes three machine learning algorithms that were tuned and tested in the same manner as the support vector machine (SVM) described in the main article. We provide background information on these other three classification algorithms as well as details on the tuning process, the hyper-parameters that were cross-validated, and the accuracy scores on the test data set to show evidence for the determination that the SVM classifier performed the best out of the four algorithms.

Text S1.

For this study, four machine learning algorithms were analyzed to determine the model that performed the highest accuracy score on the test data. Each of the four algorithms were tuned with the tuning process outlined below and tested on a dB/dt test set from the eight stations for the year of 2016. The machine learning classification algorithms that were analyzed are the decision tree, random forest, support vector machine (SVM), and Gaussian process classifier (GPC).

A Gaussian process classifier (GPC) is similar to an SVM in that it also relies on a kernel function, but uses this kernel differently. A GPC classifies data by predicting possible functions that fit the training data (Rasmussen & Williams, 2006). The initial assumptions for the functions that may possibly fit the training data are determined by the kernel function which represents the probability distribution over these functions. The model assumes a prior probability distribution of these functions via the kernel function and then updates this probability distribution based on information from the training set and using Bayesian inference methodology. In the tuning process, four different kernel functions were analyzed to determine the kernel function that performs the best on the training data: the radial basis function (RBF), the dot product function, the Matern function and the rational quadratic function.

A decision tree predicts classes by splitting the data into subsamples based on true/false requirements until every sample is isolated into a class (Song & Lu, 2015). Every time the model splits the dataset, a node is created and each group that the data was split into is referred to as a leaf. The loss function for a decision tree is a function that informs the model of the quality of each split, usually by comparing the class distribution before and

after the split via the gini or entropy functions- these are the "criterion" hyper-parameters in Table S1. Each time a node is split into subsets, the depth of the tree increases up until the maximum allowable depth specified (max depth parameter in Table S1). The max features parameter is the maximum number of features used to determine each split, the Min samples/split parameter represents the minimum number of dB/dt samples required in order to make another split and min samples/leaf is the minimum number of dB/dt samples required to be in each group (leaf) after splitting.

A random forest is an ensemble of decision tree predictors whose majority vote determines the output prediction. Each tree is built with a random subset of the training data (with the same distribution as the overall training set) as well as a random combination of the features (the number of which is less than the total number of features and designated by the 'maximum features' hyper-parameter) and the final prediction is the popular vote of these trees (Breiman, 1996).

Each tree in a random forest is built with the same hyper-parameters, but not necessarily the same training data. In addition to the decision tree's hyper-parameters, a random forest has a setting to determine how many trees will be built in the ensemble. The random forest also has the bootstrap hyper-parameter: if True, each tree is built with a random but equally sized subset of the training data with replacement (i.e. each random subset is taken from the full data set so some trees have the same samples) and if False each tree is built from the entire data set (Breiman, 2001).

The general tuning process is as follows: the training data were split into separate but equally proportional sets (referred to as a "fold" of the data) and then a model with a combination of of hyper-parameters was trained and tested on each of the partitioned

data sets and the model that scored the highest is selected for those hyper-parameters. The number of combinations of hyper-parameters to be trained and tested on was based on each models required hyper-parameters and the computation time to fit each model with each combination of parameters. The number of separate folds each model was split into, as well as the number of combinations of hyper-parameters and the overall number of fits of the model are listed in Table S1. The specific hyper-parameters that resulted in the best model for each type of model are also listed in Table S1.

Each of the four models were tuned and cross-validated with the specific number of folds and fits shown in Table S1. The optimal hyper-parameters determined are listed. Each model is then fit to the training data with the optimal hyper-parameters, then the model is tested on the test data set that consists of data from each of the eight stations for the year of 2016, the test scores are listed at the bottom of Table S1. The results in Table S1 show that the SVM model had the highest accuracy and Heidke skill score on the test set. While the POD score was slightly higher for the random forest, the SVM classifier was chosen because the average of the three test scores for SVM was higher than tht of the random forest. Thus, the SVM was chosen as the best classifier for high-frequency dB/dt signatures in magnetic field data.

Table S1.

References

Breiman, L. (1996). Bagging predictors. *Machine Learning*(24), 123–140. doi: 10.3390/risks8030083

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. doi: 10.1007/978-3-030-62008-0_35

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning.* MIT Press.

Song, Y.-y., & Lu, Y. (2015). *Decision tree methods: applications for classification and prediction.* Shanghai Arch Psychiatry. doi: 10.11919/j.issn.1002-0829.215044

Model		SVM	GPC	Decision Tree	Random Forest
Tuning	Folds	10	5	10	5
	Candidates	49	4	2700	2880
	Fits	490	20	27000	14400
Optimal hyper-parameters		Kernel = RBF $\gamma = 1$ C = 10	Kernel = Matern length scale = 1 $\nu = 1.5$	Criterion = entropy Max depth = 8 Max features = 3 Min samples/leaf = 3 Min samples/split = 9	Criterion = gini Max depth = 10 Max features = 5 Min samples/leaf = 2 Min samples/split = 3 Number of trees = 200 Bootstrap = True
Accuracy		0.9825	0.9784	0.9677	0.9627
POD		0.9112	0.9043	0.8497	0.9590
HSS		0.8688	0.8422	0.7671	0.7607

Table S1. Table detailing the tuning, training and testing information for four machine learning algorithms