CS 6490: Final Paper
A Survey of ML Algorithms for Spam Filtering

Nels Ballard, Ahmet Oguz, Michael Quigley

School of Computing, University of Utah

December 9, 2015

1 Introduction

2 Related Work

3 Adversary Model

4 Methodology

5 Design And Implementation

The project was implemented entirely in Python and can be found in a public Github [2]. For machine learning, the sklearn library [3] was used. This library supports numerous machine learning algorithms. We chose to use the Enron Email Dataset [1] for both our training and test set. We selected 12,000 spam emails and 12,000 ham emails from this set for training and testing.

The first step to using almost any supervised learning algorithm is feature selection. We decided early on to use the "bag of words" approach to feature selection. In other words, we created a text file of features (words). The occurrence of each feature was counted in each email and recorded into a feature vector for that email. The feature vectors along with their labels (spam or ham) were then all stored in files for later use so that they wouldn't have to be recomputed. Our feature selection was primarily manual in the sense that we added words into the features list that we though might occur in a spam email. We also made the observation that in some spam emails, the word "free" might be spelled correctly, or it could have one of the following spellings: "fr33", "fre3", or "fr3e". To avoid overfitting (where there are too many features to get good results), we decided to make these different situations all the same feature. We achieved this by allowing the user to specify a regular expression that corresponds to a feature. For the situation listed above, the feature entry looks as follows:

$$fr[3e]\{2\}$$

We have not seen this approach in related work and feel it is useful in avoiding overfitting although we have not performed an extensive analysis to determine if this is the case. Once the features were extracted, we would shuffle them up before passing them to any learning algorithm. We also constructed the data so that 50% was training data and 50% was test data.

Once the features were selected, we decided to run a linear support vector machine (SVM) algorithm on the list of features. To save space, we will spare many of the details except to say that we performed cross validation over the following two parameters in SVM: γ and C. The parameter γ defines the margin which is the distance between the classifying line that SVM uses to separate the data, and the points nearest the line. The parameter C defines penalty for making a mistake during the training process.

Another algorithm we tried was the decision tree. This algorithm represents each feature as a node, and a count of that feature as the edge of a node. The leaves of the tree represent the predictions of the algorithm.

The last algorithm we tried was AdaBoost. In particular, we used an ensemble of decision trees in this algorithm to make predictions. This means that each element in a particular email's feature vector was the prediction of a decision tree. The decision trees were each slightly different so that they could produce a diverse set of results. We ran AdaBoost over fifty iterations to generate our results.

6 Results

The results from SVM cross validation are shown in Table 1. The highest accuracy was obtained when C was 1000 and γ was 0.1.

Table 1: SVM Cross-Validation Accuracies

	C=1	C=10	C=100	C=1000
γ=0.0001	75%	83%	85%	85%
γ=0.001	83%	85%	86%	86%
γ=0.01	85%	86%	86%	87%
γ=0.1	86%	87%	87%	87%

The results from a decision tree classifier with a maximum depth of 10 is shown in Table 2.

Table 2: Decision Tree Prediction Accuracies

Overall	Spam	Ham
85%	94%	76%

The results form the AdaBoost classifier are shown in table 3.

Table 3: Adaboost Prediction Accuracies

Overall	Spam	Ham
89%	92%	86%

7 Conclusion

References

[1] Enron email dataset. `https://www.cs.cmu.edu/~./enron/`. Accessed: 2015-12-9.

[2] Project code. `https://github.com/aoguz/CS6490-FinalProject-SpamFilter`. Accessed: 2015-12-9.

[3] scikit-learn. `http://scikit-learn.org/stable/`. Accessed: 2015-12-9.