Towards better accuracy for Spam predictions

Chengyan Zhao
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
M5S 2E4
czhao@cs.toronto.edu

Abstract

Spam identification is crucial in implementing an effective email filtering system, while spam recognization has different properties comparing with normal text recognization. In this paper, we present three different classifiers with detailed analysis on various training data set of the given spam database. We then combine these classifiers into a mixture of expert system which yields overall better performance than any of individual contributors. We also give instructions for further improvements on classifiers as well as its requirement on spam databases.

1. Introduction

Email spam appears with the introduction of email system. Its volume and popularity increases with the adoption of email as a major communication mechanism. Despite its un-wanted nature, spam, or unsolicited email message, does not only waste precious working time, but also generates considerable amount of network traffic. Recently added natures include virus attachments and spyware agents, which make the situation worse. Commercial companies and research groups have started working on spam identification and removal long time ago, but with only limited success. Most of them use methods such as authentication, authorization, explicit blacklist, etc. In most cases, they will tentatively mark an email message as a possible spam candidate and it would be up the users to justify.

Spam identification could be considered as a special case of text identification and recognization. In this paper, we are more interested in the statistical aspects of spam. We list its comparative properties as following.

a. The definition of spam is vague. People have different opinions on whether a particular email is spam. Thus, even for the same email, it could be spam in one case and non-spam in another. Thus, we need some clearly pre-defined spam and non-spam data rather than only relying on a general classifier as the filer.

b. Email mis-labeling has different side-effects. False-Negative (FN) is generally ok, as a spam marked as non-spam will cause the user to read through it and delete it. However, false-Positive (FP) will mark a non-spam email as spam, which usually follow the sequence of automatic deletion, causing user to lose important information.

K-Nearest Neighbors (KNN) and Classical Gaussian are both well-known algorithms in Machine Learning. However, their effects on progressively changing training set' size is not studied previously. Boosting is a powerful method to supercharge a mediocre algorithm into an excellent learning algorithm, but the effectiveness of comparative results of such supercharged algorithm with classically well-known algorithms are not clear neither. Further, we implement a Mixture of Experts (MOE) system which takes inputs from KNN, Gaussian and Boosting, gives a predictive result based on each expert's past behaviors.

The goals of this paper are three fold. (1) Analyzing classical algorithms, under the condition of varying training set size. (2) Comparing prediction accuracy of best classical algorithm with boosting a mediocre algorithm. (3) Analyzing prediction accuracy for a MOE system with classical algorithms and boosting algorithm as inputs.

The remainder of the paper is organized as the following. Section 2 describes dataset's representation and experimental setup. Section 3 describes the three methods (KNN, Gaussian, Boosting), and then combines them as inputs into the MOE. Changes on training set size and predication accuracy are analyzed. Section 4 gives some suggestions for further improvements, and section 6 concludes the paper.

2. Data representation and Experimental Setup

We use the spam email database from UCI[1]'s machine learning data repository[2]. It was generated in Jul. 1999 and donated from HP Labs. There are totally 4601 emails in the Spambase, among which 1813 (39.4%) are spam. Each email in the Spambase is represented as a vector of 57 real values, called feature space. 47 of such features are word frequencies, which is the occurring frequency of some specific words. 6 of the remaining features are character frequencies, such as the occurrence of frequency of some specific characters in the email. The remaining 3 are statistics of capital letters, as the longest, average and sum of the length of continuous capital letters.

We initially partition the spam space into 2 parts: 3000 cases for training data (among which the 1st 1000 are fixed training data and 1000-3000 are dynamic training data) and the remainings are test data. For each algorithm, we perform the experiments on static partition, as well as with various sizes of training data (1000 - 3000, with increment of 200 each). For each algorithm, we also use 4-fold cross validation on the training set only.

False Positive (FP) and False Negative (FN) are described in section 1. We give out their definitions in the following.

$$FP = \frac{nonspam_misclassified}{total_spam_samples} \qquad FN = \frac{spam_misclassified}{total_spam_samples}$$

[1] University of California at Irvine. http://www.uci.edu
[2] Spambase is located at: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/,

3. Existing Algorithms and Evaluations

We evaluate three algorithms individually this section. K-Nearest Neighbor (KNN) Classical Gaussian (CG) and Boosting with Multi-Layer Perceptron (MLP). Further, we build a MOE based on the three individual algorithms to achieve higher prediction accuracy.

3.1 K-Nearest Neighbor (KNN)

KNN is a relatively simple but very efficient algorithm. It can achieve very good performance on relatively small training set. The idea of KNN is that in order to classify a test point, the algorithm chooses the most common classes among its K-nearest neighbors in training set. This K (usually an odd number) neighbors will then vote to decide the test point's label on their majority. Below is KNN's representation.

$$y(d_i) = \arg\max_k \sum_{xj \in KNN}^{n} y(X_j, C_k)$$

d_i is the test point, X_j belongs to the class of C_k on all the comparisons with training data. $y(d_i)$ is the class label with largest number of votes among X_j.

Training size	1000	1200	1400	1600	1800	2000
FP	0.208	0.192	0.188	0.190	0.182	0.180
FN	0.251	0.233	0.237	0.226	0.209	0.201
Training size	2200	2400	2600	2800	3000	
FP	0.177	0.181	0.174	0.171	0.169	
FN	0.199	0.193	0.188	0.184	0.188	

Table-1 Error Rate for KNN (K = 3)

	1	2	3	4
CV error rate	0.192	0.187	0.181	0.198

Table-2 Error rate of up to 4-fold cross-validation (K = 3) for KNN

It is clear from the table that with training set size increases, the KNN error rate decreases gradually. FP is usually smaller than FN. For cross-validations, error rate gradually reduces. The open parameter is K which value is assigned statically[3] in our testing. This value of K is also optimal when considering efficiency and performance.

[3] K = 3 is the optimal KNN size (yields lowest error rate) when evaluating the static partition with tracing K between 2 and 25.

3.2 Conditional Gaussian

Conditional Gaussian (CG) is one of the well-studied classifiers in Machine Learning. Despite its computational expense, it usually generates very good result (if not the best) given relatively large training set. The idea behind Gaussian is that assuming inputs are uncorrelated and distributions of different classes differ only in mean values, training the model towards maximum average log likelihood. Below is Gaussian's representation.

$$p(y=k)=\alpha_k$$

$$p(x \mid y=k)=(2\pi)^{-D/2}\,|\sum\nolimits_K|^{1/2}\,\exp\{-\frac{1}{2}(x-u_k)^T\sum\nolimits_K^{-1}(x-u_k)\}$$

D is the number of dimensionality of the matrix, u_k is the mean of the matrix and $\sum_K$ is the covariance of the matrix.

Training size	1000	1200	1400	1600	1800	2000
Error rate	0.133	0.112	0.113	0.110	0.104	0.098
Training size	2200	2400	2600	2800	3000	
Error Rate	0.081	0.073	0.055	0.051	0.044	

Table-3 Error Rate for Conditional Gaussian

	1	2	3	4
CV error rate	0.046	0.044	0.045	0.043

Table-2 Error rate of up to 4-fold cross-validation for Gaussian

We notice that with the increased training set size, Gaussian performs exceptionally well, with error rate well below the best rate produced by KNN. We also notice that 4-fold cross validation does not work effectively for Gaussian. The reason is that when training set size reaches over 3000, Gaussian cannot find any further independently distributed data. Data reusing in cross validation won't improve Gaussian's accuracy any further.

3.3 Boosting

The goal of boosting is to find a highly accurate classifier based on one or a few "weak" classifiers. Boosting works by repetitively applying the "weak" algorithms. "At each iteration, it increases the weight on those samples that the last classifier got wrong. Overtime, it focuses on the examples that are consistently difficult and forgets the ones that are consistently easy."[4] After a certain number of iterations, boosting needs to combine many "weak" rules into a single "strong" rule, and hoping this "strong" rule

[4] Prof. Sam Roweis's Machine Learning class notes, Lecture 12.

would yield more accurate results. In our experiments, we use multilayer perceptron (MLP) neural network as the "weak" classifier. Testing results are given in the following.

Training size	1000	1200	1400	1600	1800	2000
FP	0.068	0.061	0.057	0.049	0.051	0.044
FN	0.144	0.141	0.137	0.133	0.130	0.129
Training size	2200	2400	2600	2800	3000	
FP	0.040	0.042	0.044	0.040	0.038	
FN	0.122	0.124	0.117	0.109	0.102	

Table-5 Error Rate for Boosting (MLP)

	1	2	3	4
CV error rate	0.061	0.067	0.064	0.060

Table-6 Error rate of up to 4-fold cross-validation for Boosting (MLP)

Size of the training set is affecting the accuracy of boosting. As shown in Table-5, increased training set size will improve accuracy. Cross validation error rate seems to be in-sensitive, it might be due to boosting's dynamic adjustments. Also, number of hidden variables directly affects boosting in terms of accuracy and performance. We find numbers between 7 and 13 are good candidates.

3.4 Mixture of Experts

Any expert system needs to have a number of experts and a gate. Each of the experts can produce independent and sensible outputs based on the given data, and the gate needs to decide which expert to be called upon for a particular set of inputs. For our MOE, each of the previous three individual algorithms is an expert. The gate keeps a record for each expert's past behavior. It penalizes the experts who make higher number of wrong predictions by reducing its weight. This will result in a dynamic adjustment of the weights at the gate. This adjustment makes the system focus more on the experts who has a history of high prediction accuracy. The algorithm used by the gate is illustrated as following.

$$y = \sum_{j=1}^{n} g_j \sum_{k=1}^{K} w_{jk}(y_k)$$

y is the output of the expert system. g_j is the independent output expert j produced. w_{jk} is the weight of expert j based on its past k behavior(s).

Testing on MOE is shown in Table-7.

Training size	1000	1200	1400	1600	1800	2000
FP	0.068	0.066	0.062	0.058	0.059	0.051
FN	0.172	0.170	0.166	0.149	0.133	0.122
Training size	2200	2400	2600	2800	3000	
FP	0.044	0.047	0.041	0.043	0.039	
FN	0.120	0.113	0.106	0.100	0.092	

Table-7 Error Rate for Mixture of experts

3.5 Comparison and Analysis

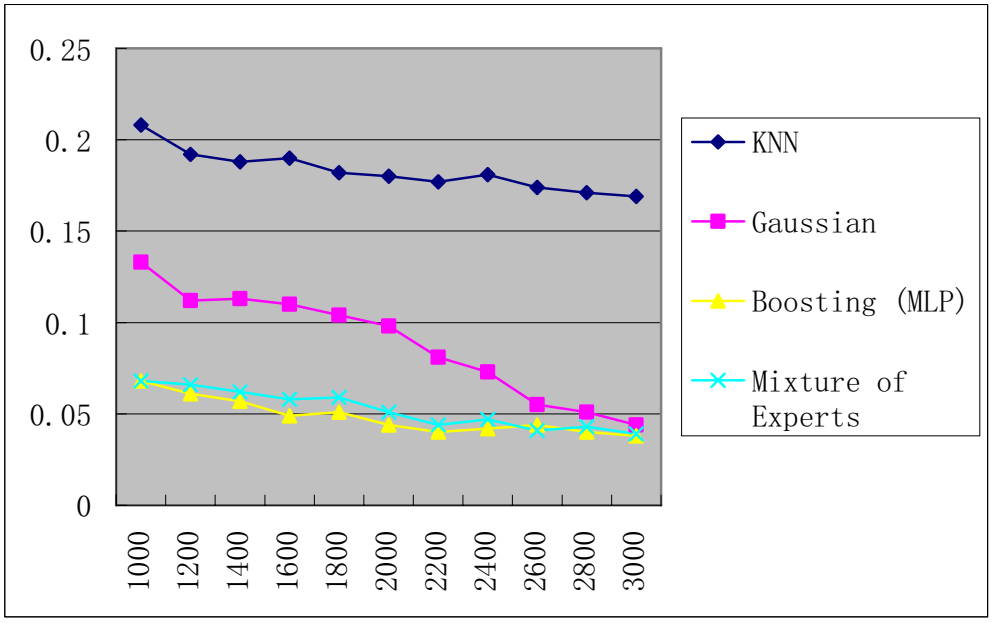

Fig-1 Comparison and Analysis of all models (number of y-axis is error rate)

Fig-1 shows a synthetic comparison, with each of the individual classifier and the mixture of expert system.

First, the overall trend is that the error rate will decrease when the size of training data set increases. Three of the classifiers (Gaussian, Boosting with MLP and MOE) converge almost to the same lowest error rate. Also notice that Boosting and Mixture of experts are relatively in-sensible to the size changes of training data set.

Second, Boosting performs exceptionally well. In many cases, its performance is comparable to MOE. In other cases, it performs even slightly better than MOE. The reason behind could be that mixture of experts will suffer from current mis-predictions as it started with optimistic assumptions on each individual expert. This penalization will

show up when training set size increases. On the other hand, boosting does not have this delaying behavior. It can fully explore the given dataset and tune the number of its hidden variables
.
Third, the lower the curve (error rate), the higher the computational expense. Different classifiers have trade-offs in terms of accuracy vs. efficiency (performance). KNN finishes quickly, but has highest error rate. Boosting and MOE take much longer to run, but yields better error rates. Gaussian is in between.

4. Conclusion and Future Work

In this paper, we discussed the accuracy and computational expenses of three individual classifiers to filer spam emails. Based on these 3 classifiers, a mixture of expert system is introduced. We notice that Boosting is surprisingly effective, it could even out perform MOE in some given ranges of training data. We next give the directions for its related future work.

4.1 Extending Models

The current analysis is based on the spam data, assuming a normal distribution. However, we notice that that last few values[5] in each email vector usually have a very large number. These numbers, apparently, won't sum up to one. It would be interesting to see the result if normalization is performed before any testing is conducted.
Due time and resource constrains, we limit the cross validation to 4 fold. It would be more convincing if we can increase this number. E.g. to make it 10 or 20 fold. The analysis would be more representative when validating these results of cross validation.

4.2 Extending Spambase information

In addition to the four classifiers introduced in section 3, we also tried Support Vector Machine (SVC). Unfortunately, SVC does not give us better error rate on top of Boosting or MOE[6]. We feel that most of the independence and variance in the given dataset have been explored, it would be hard for any further algorithm to give better performance on the same given data, no matter how complex or computational expensive this algorithm would be. More information, e.g., email arrival patterns, word' correlation and dependencies, etc., in addition to only the word frequency counts, is needed. Once available, we can apply novel classifiers, e.g. Hidden Markov Model (KMM) on this enhanced spam data representations. We anticipate better predictions for this new spam data.

References

[1] X. Carreras and L. Marquez, Boosting Trees for Anti-Spam Email Filtering, in proc. Euro Conference Recent Advances in NLP, Sept. 2001, Pages 3-4

[5] Recall these are the statistical synthesis of longest, average and sum of un-interruptible Capital letters in the email sources.
[6] Due to space constrains, details of SVM is not presented in this paper.

[2] Corinna Cortes and Vladimir Vapnik, Support Vector Networks, Machine Learning 20(3): 273-297

[3] Stan Z. Li, Kap Luk Chan, Changliang Wang. Performance evaluation of the Nearest Feature Line Method in Image Classification and Retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence, Page 7-10

[4] R. E. Schapire, The boosting approach to machine learning: An overview. MSRI Workshop on Non-linear estimation and Classification, 2002, Page 3-4.

[5] H. Drucker, D. Wu, and V. N. Vapnik, Support Vector Machines for Spam Categorization, IEEE Transactions on Neural Networks, Vol. 20, No. 5. Sept 1999, Page 2-4

[6] Rajani R. Joshi, R. Deepalakshmi, ANN Algorithm for Incremental Machine Learning: Heuristics in a Bayesian Technique, Aug. 1998

[7] Sandeep Mane, Jaideep Srivastava, San-Yih Hwang, Jamshid Vayghan, Estimation of False Negatives in Classification, Nov. 2004