SMS SPAM DETECTION

Manish Kumar (2015csb1019) , Shikhar Jaiswal (2015csb1116)

Abstract

With reduction of Short Message Service (SMS) cost huge number of commercial advertisements (spams) are being send to our mobile phones. We have applied different machine learning algorithms and compared the result after 10-fold cross validation.

Introduction

Mobile phones are very popular these days. There are more mobile phones than humans on earth. Today, more than 30% of the SMS are spams. Though, there are various tools to catch Email-spams but they are not equally effective on SMS spams. In some countries, receiving SMS also contribute to the cost for receiver. So, SMS-spam filtering is an interesting problem to solve. Although Email and SMS spams look similar, but they are very different from each other. SMS has very informal language and lots of abbreviation, the cost factor limits the user to send large SMS so they are usually very short. Also, there is lack of real databases for SMS because people generally don't want to share their personal messages.

Our goal is to compare different machine learning algorithms to solve this problem of SMS spam detection. Apart from increasing the accuracy of spam detection, we also have to consider that very few (almost none) of the hams should be classified as spam, so that spam detection tool don't block any important message.

We have used a database of 5574 text messages from UCI Machine Learning repository. Data is in a large test file where each line corresponds to one message. It starts with a label (ham or spam) followed by the raw message. We have used SVM, Naive Bayes, KNN, Decision tree and forests for classification.

Feature Extraction

Since dataset contains raw data, preprocessing is required. We have tokenized each message. All the extra characters like comma, digits etc are removed and then each message is split into tokens of alphabetic characters (words) and their count in the message. Abbreviation and misspellings are ignored. A total of 9973 features were found. Not all the features are useful so we removed those features which are too common or too rare (count is less than 10 or greater than 500 in the whole dataset). We ran ID3 algorithm on priliminary basis and found that still there are many redundant features (deciding feature are very few). So we further reduced the features (count less than 10 and more than 500 were removed). After that, three more features were added to each message like word count, number of '@' and number of greater than 1 digit numbers. Intuition behind this is as follows: Since exceeding a limit costs extra charge so people try to limit the size of message. Generally '@' implies presence of a website or email address which is usually there in most of the spams. Greater than 1 digit numbers will denote the price money which spammers generally use to fool people, we didn't use 1 digit number for this because 1 digit numbers are generally abbreviations used by people in SMS. Also we didn't use dollar because currency may be different for different countries. We have used certain matrices to compare the algorithms like: SC represents percentage of spams caught and BH represents percentage of Blocked hams.

SC = False negative cases / Number of spams * 100

BH = False positive cases / Number of hams * 100

Support Vector Machines

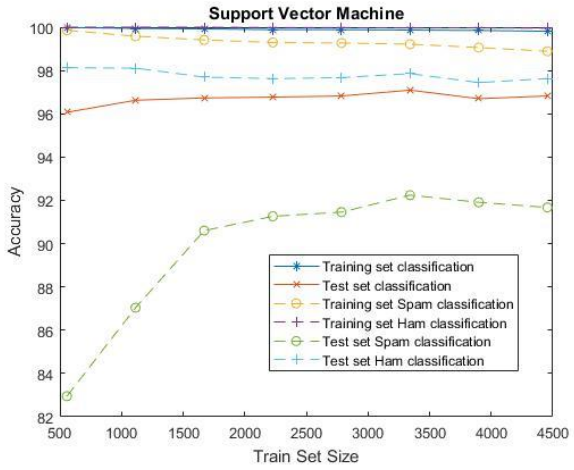
After running ID3 on preliminary basis we got an idea that data is linearly separable. So we used linear Kernel to the dataset.

Learning curve for SVM using linear Kernel is shown in figure 1.

From figure, we can see overall accuracy on train set is initially 100% which decreases a bit as we increase train data size due to overfitting of the model. Validation accuracy increases from 96 to 97%. Accuracy for train set ham classification is constant near 100%, while train set spam classification curve starts from 100% and then overfits later. Accuracy for test set ham classification is near about 98% while test set spam classification increases from 82 to 92%.

For 10 fold cross validation, Training SVM with linear Kernel gives 2.81% overall error, 92.78% SC and 2.11% HB.

Figure 1.(Learning Curve for SVM)



Naive Bayes

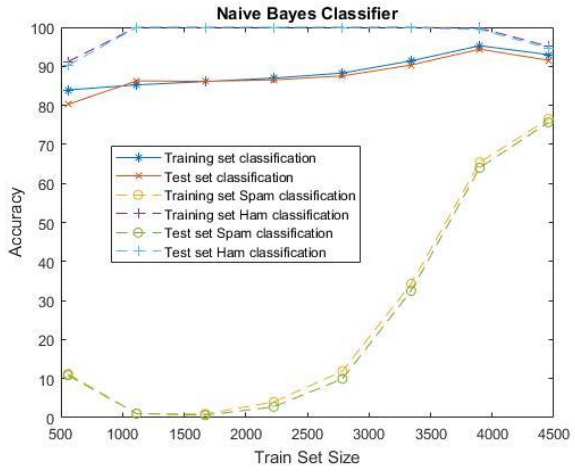
Learning curve after applying Naive Bayes (NB) algorithm to the final extracted features gave the learning curve shown in figure 2.

Overall accuracy on test set is similar to train set. Which increases from 80% to 90%. In the end model started to overfit as test accuracy becomes less than train accuracy.

Ham classification accuracy is very high for both train and test set. Spam classification accuracy for train and test set increased upto 80% from 10% with train set accuracy a bit higher.

For 10 fold cross validation, Training SVM with linear Kernel gives 5.61% overall error, 75.64% SC and 5.34% HB

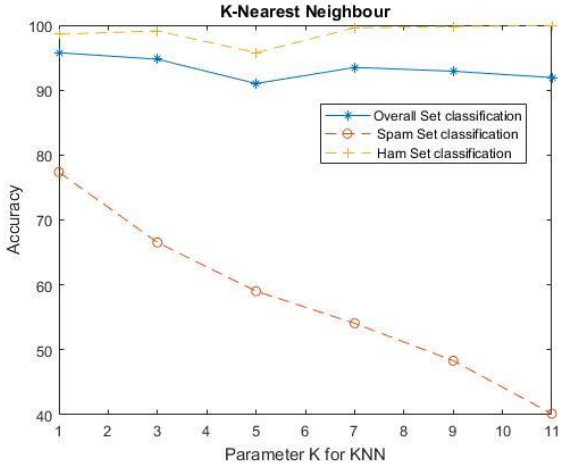
Figure 2.(Learning Curve for NBC)



K-nearest neighbor

This is a simple instance based learning algorithm where we find the euclidian distance of a test instance from all the train instances. Then assign label to test instance based on the majority vote of K-nearest neighbors. Table 1. shows results for 10 fold cross validation for different K. From the results it is evident that overall error increases if we increases the value of k. This is because of the fact that K-nearest neighbor don't consider different density clusters. We have very less examples of spams and are huge number of hams located almost everywhere on the instance space(few of which are also noise).

Figure 3.(Learning curve for KNN)



K	% Error	% Spam Caught (SC)	% Blocked Hams (BH)
1	4.23	77.37	1.3
3	5.20	66.53	0.87
5	5.88	59.03	0.45
7	6.49	54.08	0.39
9	7.08	48.32	0.18

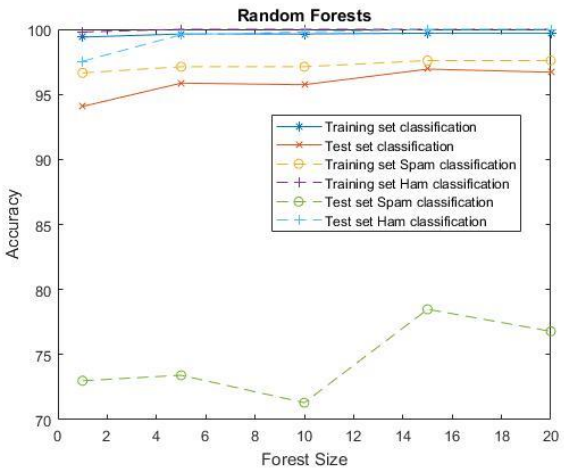
Table 1

Random Forest

This is a simple decision based algorithm where we learn number of decision trees corresponding to the training set by randomly selecting a subset of features and using ID3 algorithm. After that we decide the label of test instance based on the decision described on each node of the tree. We used 500 random features at a time and calculated accuracy by varying number of trees in the forest (figure 3).

Table 2 shows performance of decision tree for different number of trees in the forest.

Figure 4. (Learning Curve for Random Forest)



No. Trees	% Error	% Spam Caught (SC)	% Blocked Hams (BH)
1	5.92	72.99	2.45
5	4.14	73.41	0.42
10	4.26	71.30	0.21
15	3.06	78.48	0.00
20	3.26	76.79	0.00

Conclusion

Table 4. compares Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbor (KNN), Random Forests on different matrices like Overall Accuracy, Spams Caught (SC), Blocked Hams (BH) %.

Model	% Accuracy	% Spam Caught (SC)	% Blocked Hams (BH)
SVM	97.17	92.78	2.11
NB	94.39	75.64	5.34
KNN	95.77	77.37	1.30
Random Forest	97.94	78.48	0.00

The best model we found was Random Forests with 15 trees and 500 random features at a time. This gave an overall accuracy of 97.94%. The second best model was SVM with linear Kernel which gave an overall accuracy of 97.17%. Spams caught in SVM(92.78%) was better than Random Forests(78.48). But it blocked more hams than Random Forests. So if we have to be more careful that no hams should be blocked then Random Forests is best amongst these(0% BH) but if we can allow little bi of hams to be blocked then SVM is no doubt the best classifier(92.78% SC).

References

[1] SMS Spam Collection Data Set from UCI Machine Learning Repository, "http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection"

[2] "http://en.wikipedia.org/wiki/Mobile phone spam"

[3] matlab inbuilt svm implementation