Email Spam Filtering

Ananda Kishore Sirivella, UFID: 9951-5080
Department of Computer Science, University of Florida,
Gainesville, USA, Email: asirivella@ufl.edu

ABSTRACT

In recent times, Electronic mail are being used a frequent to communicate by millions of people on the daily basis. Over the decade, the exploration of email advertising and increasing in unsolicited bulk e-mail has created a need for a reliable anti-spam filter. A Junk email could be an unsolicited advertisement, phishing scam, email spoofing, commercial advertisement, .etc. With day to day upgrade in email spam patterns, we look for a machine learning algorithm for the anti-spam filtering. In the project, we shall explore, implement and compare some of the most popular machine learning methods (Naïve Bayes classifier, K nearest neighbor classifier & SVM). In the project, I would implement these algorithms and learn the concepts involved in it. By the end of the project, I aim to achieve at a statistical conclusion over the effectiveness of each classifiers over different email spam patterns, knowing which classifiers fits best for a what type pattern of spam.

INTRODUCTION

Email is the most common method of conversation of the modern times. Email is termed a method of digital message exchanging with the development of email, most our communication could be personal or professional or commercial are done by email. Historically, email has started at 1970s and initial system was more functional like a fax, where sender and receiver both had to be online. Over time, email system has become more complex and flexible to the humans. Present email system are mostly based on store and forward model. The present email servers store, forward and delivery messages. There is no necessity of User to be online at the time of communication.

An email composes of three components namely message envelope, message header and the message body. The message headers holds the information regarding the transfer protocol information, sender information, the message format information, information to decode the message and various other traits of the email for the proper communication between the email servers.. The message header just holds the title of the email and the message body holds the content to be shown to the user. The configuration format of the emails is enforced by the Simple Mail Transfer Protocol (SMTP) RFC 821. There are many email platform that provide the service with the enforcement of the standard. Some of the popular email platforms are gmail, outlook, Hotmail, yahoo and many more.

With every advancement in technology, there comes its merits and demerits. By the heavy usage of the email system, there opened a wide varieties of possibilities of communication, marketing, theft, entertainment and much more. With commercialization of the internet and combination of email system as common means of communication, there came the issue of unwanted mails to the user which can be termed as spam mail or a junk mail. The first time junk mail was send in 1978 to all the users of ARPANET (the initial email system). These kind of mails weren't termed as junk until the end of 1993.

Until 1994, spamming was termed as a taboo and penalized the senders as a spammer. In 1994, a group of lawyers hired few freelance programmer, got some email Ids from the surveys and sent the first ever advertising email to a huge list of recipients. Even though they were termed as spammer, they did make a huge lot of money from it. This marks the start of the email marketing era. With the marketing and usage there came rouge programmer to manipulate users to their own profit.

In the year 2004, a programmer was first sentenced to a huge fine and a 7 year imprisonment for sending over 800 million mails to the stolen identities. Later many followed their path and initially they were identified & law enforced. With the huge spread of internet, it became obsolete to track the sender and increased the need for filtering the spam at the recipient end.

Email Spam as termed as an unwanted message that's deliver to the recipient user. Ham is termed as the legitimate message that the recipient user are anticipated to receive. Before we proceed with the traits of a spam message, it is important to understand the types of spam messages and the problem posed by these variety of spam message. Following are the types of spams:

- *Commercial Advertisements:*
  Commercial advertisements are the huge junk of regular advertisements that we find in our junk folder. These bulk emails are mostly sent from a large company regarding their advertisements. These advertisements are needed to be considered as ham

but due to their huge volume they are to be considered a spam rather than classifying as a ham. Some of the example would be a mail about black Friday day deal by Walmart or publix or homedepot.

- *Unsolicited Advertisements:*
  Unsolicited advertisements are the unwanted / unsubscribed advertisements. Typically, few of the smaller companies acquire the user email list from various online surveys and send out the advertisements to these user. These mail are a sure spam category. Some of the example would be an advertisement about weight loss, male sex organ enhancement, online education programs,etc.

- *Phishing scams:*
  These mails are a start of a scam, where in which the sender disguises the email to look exactly like a well-known financial institution or a welfare organization. These email offer equally convincing websites that are hosted in order to capture user credentials and in term compromising the original accounts.

- *Nigerian 419 scams:*
  This is a well-known and every day seen spam, where in which user gets a message regarding abandoned wealth in a faraway continent or country. And message requesting for recipient the right to claim the prize. These mails pull the users into disclosing their personal information and accumulates to identity theft of the recipient.

- *Trojan Horse Email:*
  These email are the dirtiest. These emails come with a fancy attachments and cheating user into installing a malicious software. Which in turn destroys the user's computer and opens up the user's system to internet to huge variety of possibilities.

- *Political or Terrorist Spam:*
  These are the email that spoof the government agencies and threaten the user committing into un-reasonable crimes. These types of spams threaten user to believe to hold a crime record or a future attack information or a threat call and these spams are mostly identity theft oriented.

- *Email spoofing:*
  These emails are disguised to look exactly like a well-known bigger organization. These email offer equally convincing websites and email. These given links are hosted in order to capture user credentials, in term compromising the original accounts and a huge identity loss.

Since the last decade, there has been huge rise in the spam activity over the internet, as of now every day 100 billion spams are spread of the internet. And day by day the spammers are tending to design smarter and harder to crack. We need more effective ways to protect the user and the internet from these huge spam mail. These spam email not only make us lose our important data but also eats up a lot of storage.

PROPOSED SOLUTION

Machine learning is the science of algorithm in which the algorithm are design to learn with the data and compensate the variation in the data. Machine learning algorithms can be divided into two sets supervised learning and unsupervised learning. Supervise learning is where in which we have a labelled training dataset on which we would build a model (like a black box) that would process the input mail and categorizing into its respective class (in our case spam or a ham). Unsupervised learning involves building the model (or a black box) base directly on the real time unlabeled data set, these algorithm tend to be slower and less accurate than the supervised learning algorithms but are very effective in the long run. We shall be using the machine learning algorithm for building classifiers for email filtering to spam or ham.

Some of the basic supervised learning algorithms are Naïve Bayes, K-Nearest Neighbors and SVM. We had try implementing these algorithm for email spam filtering on UCI spambase and evaluate the effectiveness of each algorithm. We shall implement these algorithm using python over UCI spambase.

The following are the traits of UCI spambase:

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Integer, Real
Number of Instances:	4601
Number of Attributes:	57
Area:	Computer
Date Donated	1999-07-01

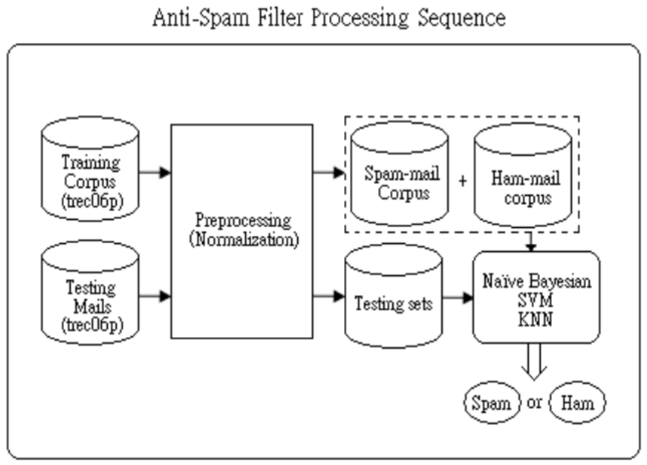


Figure 1: Proposed Solution

NAIVE BAYES CLASSIFIER

Naïve Bayes Classifier is of the basic techniques of building a classifier. Naïve Bayes classifier is a supervised learning type of algorithm. Supervise learning is one of the concepts of machine learning where in which we build our classifier model on the basis of training data set, then test it on a test data set and generate its parameters for apply it on the real-life data set. Naïve Bayes Classifier is one of the efficient methods of classification, with a small training set we could arrive at the corresponding parameters for solving necessary equations.

Naïve Bayes algorithm considers probabilistic approach to a machine learning problem, it is a conditional probability model. We represent each sample with its features as a vector and classify them as using its instant probabilities. We would represent our basic Bayes classifier probability equation as:

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

Where,
P(C_k) = Posterior probability
P(C_k|x) = Posterior probability
P(x| C_k) = Class prior probability
P(x) = Predictor Prior probability

With the email spam filtering problem, we would have to modify the equation to differentiate if an email is a spam or ham based on the word probabilities. In the email spam filtering problem, we have two classes to differentiate (spam or ham) based on the frequency of words contained in the message body. We modify the Bayes classifier and formulate new equation:

$$p(S|W) = p(W|S) * p(S) / (p(W|H)*p(S) + p(W|H)*p(H))$$

Where,
p(S|W) = probability that a message is a spam based on a given word.
p(W|S) = probability that a word exists in a spam message
p(S) = probability that a given message is a spam
p(W|H) = probability that a word exists in a ham message
p(H) = probability that a given message is a ham

For implementation of the Nave Bayes algorithm, we have various even model to populate with various distribution models. Some of them are Gaussian, multinomial,.etc. When we have a continuous data we generally tend to use the Gaussian Byes for the implementation.

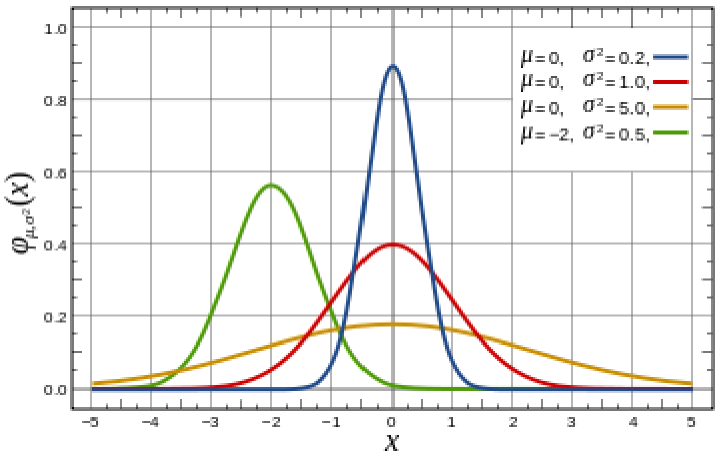


Figure 2: Gaussian Distribution

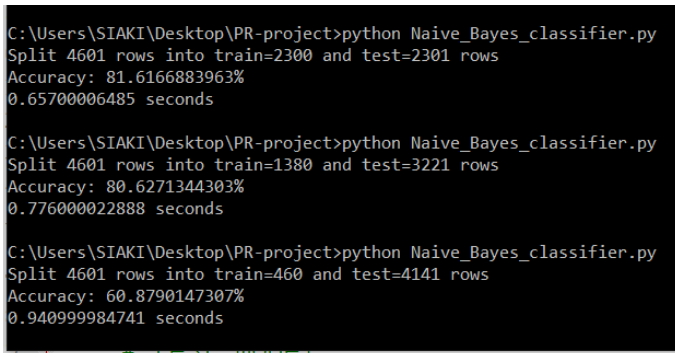
The figure explains the possible distribution of data between the mean (0,-2) and their variance (square of standard deviation).

As the figure signifies central maxima is oriented at mean with distribution area increase & the smaller peaks with increase in the variance. Applying the equation for Gaussian naïve Bayes classifier with mean μ_c and variance as σ^2_c , we get $p(x = v|c)$ as the probability distribution for a value v being assigned to a class c.

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma^2_c}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma^2_c}}$$

With Gaussian and the naïve Bayes concept well explained, we shall proceed to the specific of the experimentation. Considering the UCI spambase dataset, we had divided the dataset in to training and testing programmatically using user-defined functions. In general cases, we would achieve greater efficiency with 50% training set. We observe a 80% accuracy for 30% and the 50% training set, but with a 10% data set there seems to be steep drop in accuracy and rise in execution time. Hence, we can term 50% training set as one of the optimal solution to naïve Bayes classification.

The result of the runs for 50%, 30% & 10% training set are as follows:



```
C:\Users\SIAKI\Desktop\PR-project>python Naive_Bayes_classifier.py
Split 4601 rows into train=2300 and test=2301 rows
Accuracy: 81.6166883963%
0.65700006485 seconds

C:\Users\SIAKI\Desktop\PR-project>python Naive_Bayes_classifier.py
Split 4601 rows into train=1380 and test=3221 rows
Accuracy: 80.6271344303%
0.776000022888 seconds

C:\Users\SIAKI\Desktop\PR-project>python Naive_Bayes_classifier.py
Split 4601 rows into train=460 and test=4141 rows
Accuracy: 60.8790147307%
0.940999984741 seconds
```

K-Nearest Neighbors

K-Nearest Neighbors is of the basic and laziest technique of building a classifier. The K-Nearest neighbor as the name goes, it is a basic instance based classifier which computed the distance between k nearest neighbors (taken from the training dataset) and by analysis of the k neighbors class, we access the situation and likelihood of the object being into class that of k nearest neighbors. K-Nearest Neighbors is also one of the supervised learning algorithms, which has a training phase and a testing phase in terms of developing a classifier.

As the algorithm goes, during the training phase we just get the training set data vectors and store them to be used later. When, we feed any new data into the system, we just compute the Euclidean distances between the input data vector & the stored training data set and compute the k nearest neighbors to the new data vector. With the k neighbor's traits and class distribution data, we simply take a simple probabilistic permutation and assign the class to the new data vector. The algorithm would make itself clearer and better sense with an example.

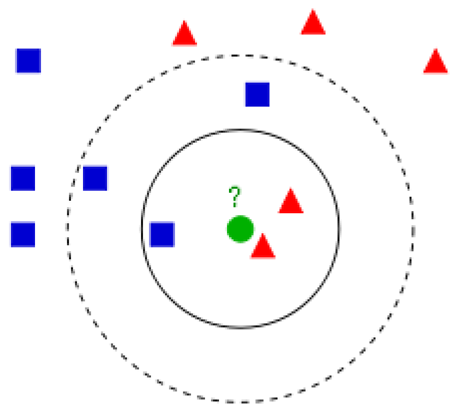


Figure 3: K-Nearest Neighbor illustration

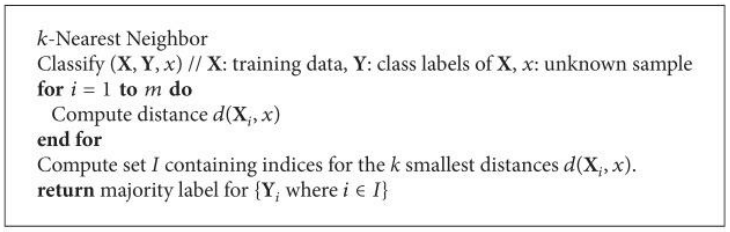
In the above figure, the blue squares correspond to class Blue and red triangles correspond to class Red. And the green circle is the new input data vector to the system. Now we consider different cases of K

For K = 1,
Then we consider the most nearest training data point and classify the same this point to that class. In this case the nearest neighbor is a red triangle and corresponds to class Red. Hence, we classify the new data point to class Red.

For K = 3,
Then we consider three nearest training data points and check for the highest occurring class and assign the same to the new data vector. In this case, the 3 nearest neighbors are 2 red triangles and one blue square. With Red triangles being 2/3 more than the blue squares, we classify the new data vector similar to red triangles .i.e. class Red.

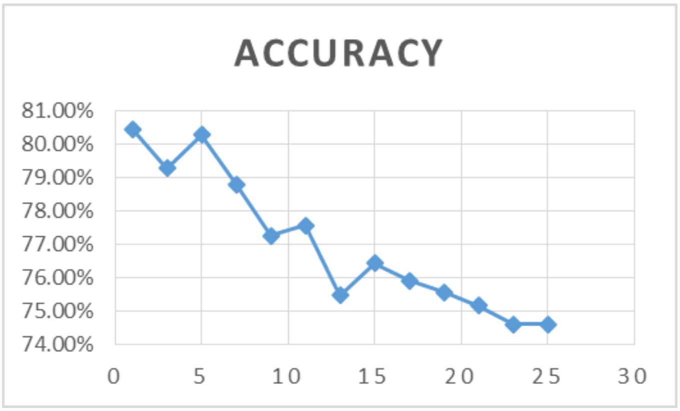
For K = 5,
We consider the dotted line circle and accumulate the five nearest neighbors and tag the corresponding class to the new data point. In the above case, the dotted line circle has three blue squares and two red triangles, hence we assign the new data vector to be class Blue. The classification simply involve the counting of training set vectors.

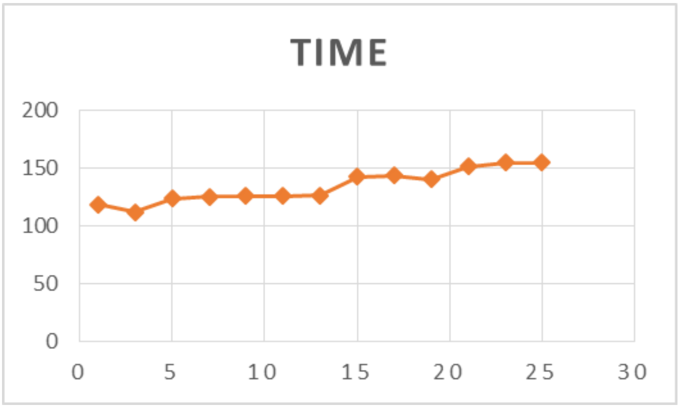
Below is the pseudo code for the K-Nearest Neighbor algorithm.



```
k-Nearest Neighbor
Classify ($\mathbf{X}, \mathbf{Y}, x$) // $\mathbf{X}$: training data, $\mathbf{Y}$: class labels of $\mathbf{X}$, $x$: unknown sample
for $i = 1$ to $m$ do
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set $I$ containing indices for the $k$ smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$
```

For the spam base problem, we consider a 50% training set and rest as test set. We store the training set and compute the nearest neighbors to each data vector in the test set and loop for different values to K to arrive at an optimized and effective value of K. As far as our observation goes the smaller the training the lower the execution time, but the low sparsity of a smaller training set could lead to a huge mis-classification error.





The above figure signifies the accuracy vs K chart for K ranging from 1 to 25. We had fixated the training set to be 50% of the mail dataset and executed the code for each K. On observation of the graph, we had seen the steep rise in the accuracy at K = 5 and drop afterwards. So, K = 5 seems to be a good optimal value for the K-Nearest Neighbor implementation. The time taken for the execution doesn't vary much with the K value but it does keep increasing with K.

The best optimal K-Nearest Neighbor model would be for K = 5 and training Set of 50%. The statistics for the set of K values would be

K	Accuracy	Time (in seconds)
1	80.44%	118.656
3	79.27%	111.852
5	80.27%	123.587
13	75.49%	126.307

Support Vector Machine

Support Vector Machine is a supervised learning model of algorithm where we analyze the data, classify it and make regressive analysis over it. With the training set marked as one of the two categories, SVM build a model to assign the new incoming data to their respective categories. A model built by SVM maps the data set as a point in vector space and maps a differentiating line or a plane between the two categories data. When it becomes hard to classify (non-linear dataset) in a 2-D vector space, SVM has a kernel trick which helps in expanding the vector space to n-dimensions to make the classification linear.

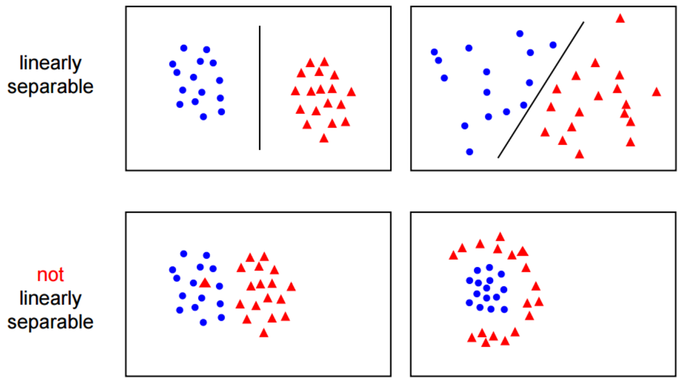
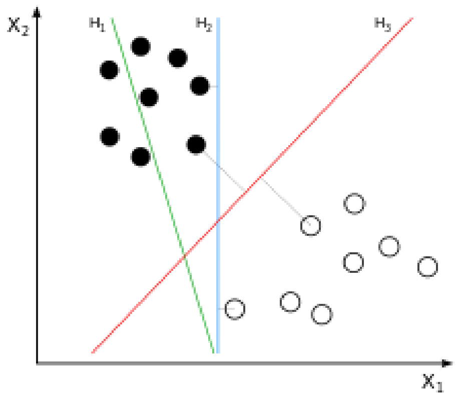


Figure 4: SVM dataset illustration

The first version of SVM ever was suggested in 1963 by Vladimir N. Vapnik and Alexey Ya. Chervonenkis. By 1992, there came a development to the SVM and a solution to handling non-linear classification (using kernel trick). The current version of SVM include stand incarnation (soft margin) and emphasizes more on the marginal error Reductions.
Coming to the basic definition of SVM built classifier, the classifier is nothing but a just a hyperplane or a set to hyperplanes (in case of higher dimensions). We had intuitively achieve a hyperplane that classifies the categories correctly and reduces the margin between the two categories.



Consider the above figure, H1, H2 & H3 all three are classifiers for the class 1 (black color filler circles) and class 2 (hollow circles). H1 is considered a separator but can't be used as a classifier as it doesn't separate class 1 and class 2 rather it bifurcates class 1. H2 can be accepted a classifier but it isn't an optimal one as it has a small margin with the data set. H3 would be considered as an optimal classifier with good margin distance.

For a given training data set $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where assume yi can be either 1 or -1. We would define a hyperplane by the below equation:

$$\vec{w} \cdot \vec{x} - b = 0,$$

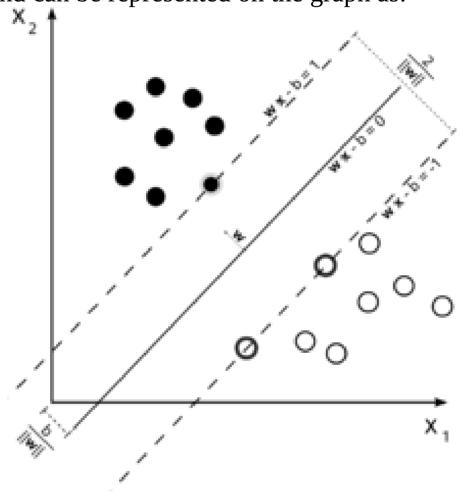
When we consider a linearly separable data set, we could have two hyper planes to categorize each class separately and bound a margin of $\frac{2}{\|\vec{w}\|}$. These hyperplanes can be described by the equations

$$\vec{w} \cdot \vec{x} - b = 1$$

and

$$\vec{w} \cdot \vec{x} - b = -1.$$

And can be represented on the graph as:



In order to find an optimal classifier, we need to find an optimal place with an objective of maximizing the margin. The distance between the planes can be termed as margin and it is given by $\frac{2}{\|\vec{w}\|}$. Now problem reduces to simple reduction to maximizing of $\|\vec{w}\|$. We had achieve that by adding further constraints to the equation .ie.

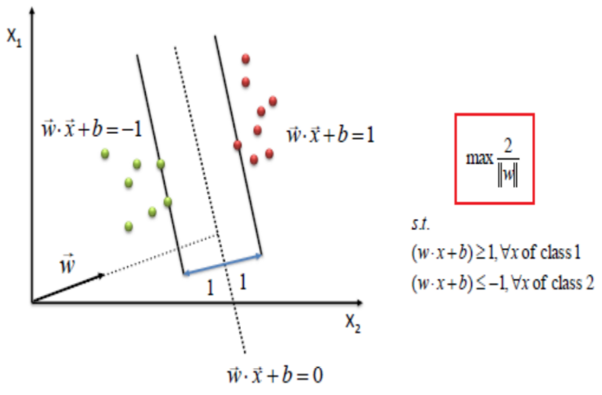
$$\vec{w} \cdot \vec{x}_i - b \geq 1,$$

Or

$$\vec{w} \cdot \vec{x}_i - b \leq -1,$$

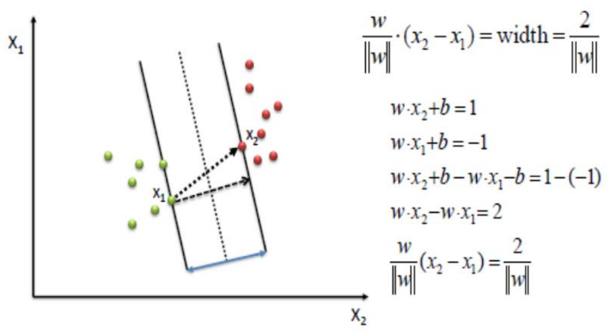
These constraints help maximizing the margin and thereby increasing the effectiveness of the classifier.

Let us compose the SVM information into a single image and deduce an optimal value to the margin maximization problem.


X_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X_2
$\vec{w}\cdot\vec{x}+b=0$
$\max \frac{2}{\|w\|}$
s.t.
$(w\cdot x+b)\geq 1, \forall x$ of class 1
$(w\cdot x+b)\leq -1, \forall x$ of class 2

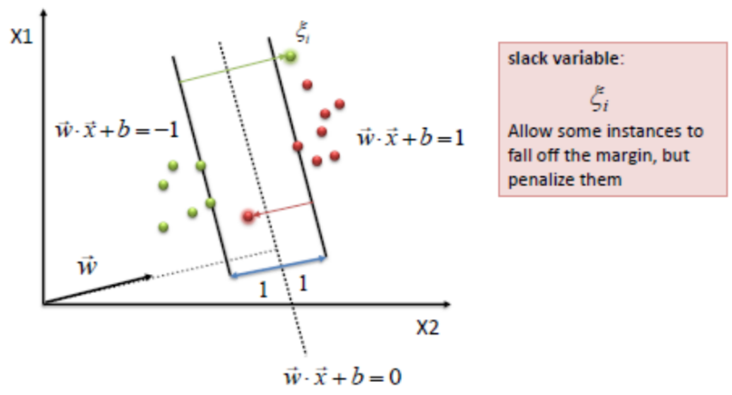
In the above figure, we constraint the class 1 by red points and class 2 by green points. We shall assume the width of margin to be 2/||w|| (normalized value) and deduce the equations for class 1 and class 2.

Consider two points x1 & x2 that fall one each of its own class margin lines. We could calculate the distance between then and with the help of pythagorean theorem we can arrive at a value for the margin. The derivations are well depicted in the image below.

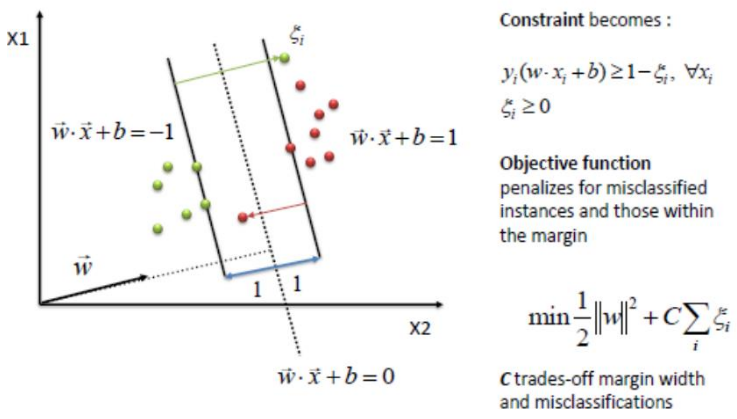

X_1
X_2
x_1
x_2
$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$
$w\cdot x_2+b=1$
$w\cdot x_1+b=-1$
$w\cdot x_2+b-w\cdot x_1-b=1-(-1)$
$w\cdot x_2-w\cdot x_1=2$
$\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}$

We shall continue with the maximization of the margin and once we reach a conclusion we shall announce an objective function that's best suits for out classifier.

We introduce a slack variable to compute the misclassification error. The data vector with its distance less than the slack variable is termed as margin violation and is considered as a misclassification. Not all cases of data can be linearly separable always.


X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$
slack variable:
ξ_i
Allow some instances to fall off the margin, but penalize them

With the introduction of slack variable the equations and the constrains deduce to

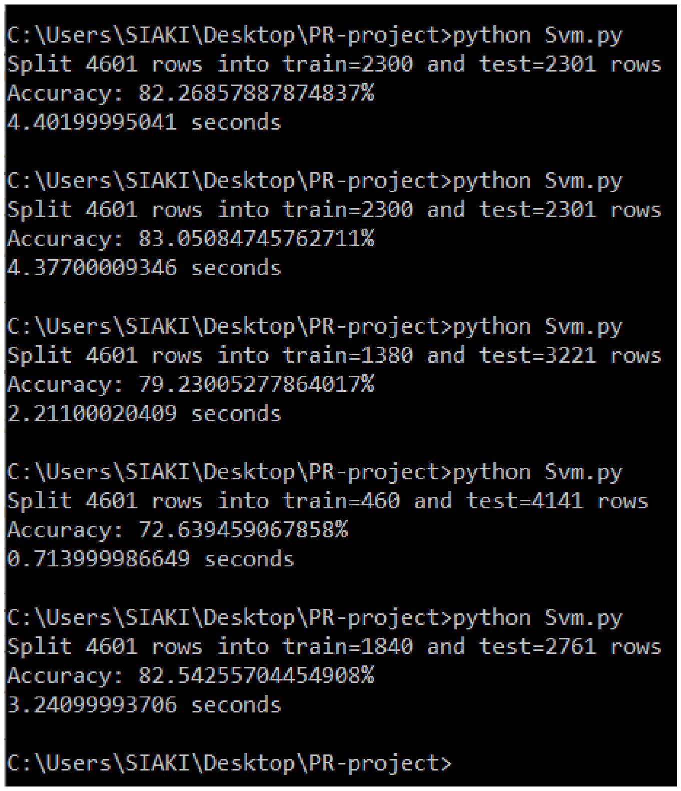

X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$
Constraint becomes :
$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$
$\xi_i \geq 0$
Objective function penalizes for misclassified instances and those within the margin
$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$
C trades-off margin width and misclassifications

With the above figure we conclude the required derivation for linear SVM. Coming to the non-linear data, we would have to user kernel trick to classify the data. With kernel trick, we would be projecting an existing 2-D data vector a higher dimensional vector (3D or more) and we deduce the equations for the set of hyperplanes as a classifier. Until now, the SVM function would always deal with the vector product of the objective function and the data vector. With kernel trick we eliminate all the vector products and introduce the scalar computation in the equations.

For a xi, the new transformed point shall be $\varphi(\vec{x}_i)$.And we introduce a kernel function to be $k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$ to convert the SVM equation to

$$b = \vec{w} \cdot \varphi(\vec{x}_i) - y_i = \left[\sum_{k=1}^{n} c_k y_k \varphi(\vec{x}_k) \cdot \varphi(\vec{x}_i)\right] - y_i$$
$$= \left[\sum_{k=1}^{n} c_k y_k k(\vec{x}_k, \vec{x}_i)\right] - y_i.$$

Coming to the Implementation of the SVM. We have numerous tools for the implementation, most commonly used tool are matlab, python scikit-learn, octava,etc. For this project we had chosen python for the implementation of the SVM classifier for UCI spambase dataset. Below are few execution trials with varied training set. It is observed to increase execution time with increase in the training set.



```
C:\Users\SIAKI\Desktop\PR-project>python Svm.py
Split 4601 rows into train=2300 and test=2301 rows
Accuracy: 82.26857887874837%
4.40199995041 seconds

C:\Users\SIAKI\Desktop\PR-project>python Svm.py
Split 4601 rows into train=2300 and test=2301 rows
Accuracy: 83.05084745762711%
4.37700009346 seconds

C:\Users\SIAKI\Desktop\PR-project>python Svm.py
Split 4601 rows into train=1380 and test=3221 rows
Accuracy: 79.23005277864017%
2.21100020409 seconds

C:\Users\SIAKI\Desktop\PR-project>python Svm.py
Split 4601 rows into train=460 and test=4141 rows
Accuracy: 72.639459067858%
0.713999986649 seconds

C:\Users\SIAKI\Desktop\PR-project>python Svm.py
Split 4601 rows into train=1840 and test=2761 rows
Accuracy: 82.54255704454908%
3.24099993706 seconds

C:\Users\SIAKI\Desktop\PR-project>
```

We have used the basic verison of svm.SVC() for the implementation. We had varied the training set percentage and verified the speed and accuracy against the percentage change. We had observed that the accuracy of the output increase on increasing the Training set but we lose the same in time. We stats refer as :

Training Set	Accuracy	Time (in seconds)
10%	74.02%	0.742
20%	76.80%	1.427
30%	78.98%	2.173
40%	80.66%	3.256
50%	82.96%	4.384

CONCLUSION

We had successfully implemented and executed all the three algorithms for spam filtering, namely Naïve Bayes, K- Nearest neighbor and Support Vector Machine. On observation of results, we can conclude Naïve Bayes approach works as a better classifier given a good training set with high sparsity.

Approach	Training Set/K	Accuracy	Execution Time
Naïve Bayes	30%	82.52%	0.816
	40%	81.89%	0.67
	50%	82.53%	0.623
	66.70%	83.08%	0.514
K-Nearest Neighbor with 50% training set	1%	80.44%	118.656
	3	79.27%	111.852
	5	80.27%	123.587
	9	77.27%	125.641
	15	76.45%	142.432
	19	75.58%	140.038
	25	74.62%	154.324
SVM	10%	74.02%	0.742
	20%	76.80%	1.427
	30%	78.98%	2.173
	40%	80.66%	3.256
	50%	82.96%	4.384

REFERENCES

[1] Sergios Theodoridis , *Machine Learning: A Bayesian and Optimization Perspective*, 1st edition, Academic Press, 2015.
[2] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*, 2nd Edition, Wiley-Interscience, October 2000.
[3] David G. Stork, Elad Yom-Tov, *Computer Manual in MATLAB to accompany Pattern Classification*, 2nd Edition, Wiley-Interscience, April 2004.
[4] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, 1st Edition, Springer, October 1, 2007.
[5] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Second Edition, Springer, February 9, 2009.
[6] Drucker, H., Wu, D., Vapnik, V.N, *Support Vector Machines for Spam Categorization*. IEEE Transactions on Neural Networks 10(5), 1048 – 1054 (September 1999)
[7] Hinneburg, C.C.A.A., Keim, D.A.: *What is the nearest neighbor in high dimensional spaces?* In: Proc. of the International Conference on Database Theory (ICDT), pp. 506 –515. Morgan Kaufmann, Cairo, Egypt (September 2000)
[8] Bickel, P.J., Ritov, Y., Zakai, A., *Some theory for generalized boosting algorithms.* Journal of Machine Learning Research 7, 705 –732 (2006)
[9] en.wikipedia .org
[10] scikit-learn.org/stable/
[11] paulgraham.com/spam.html
[12] machinelearningmastery.com/
[13] archive.ics.uci.edu/ml/datasets/Spambase