E-mail Spam Filtering

Juthika Das, UFID 7173-5283

***Abstract*—E-mail spam filtering is a very widely discussed and studied topic in the field of pattern classification. E-mails can be filtered as spam or non-spam based on many features such as the frequency or occurrence of a few words in the e-mail, the length of the e-mail or the domain from which it is being sent. Based on these basic characteristics, researchers have come up with many techniques to identify a spam e-mail for a non-spam e-mail. While most of these techniques are based on strong foundations, there are subtle or wide differences in their efficiencies. By efficiency, I mean the accuracy, time required to get to the result and other factors which can give one algorithm or technique an edge over the rest. In this project, I aim to implement and evaluate three major e-mail spam filtering algorithms. They are, Naïve Bayes method, k- Nearest Neighbors and Support Vector Machines. In this report, I'll be going through the motivation of using these algorithms or the basic science behind them, working of the aforementioned algorithms, their accuracies and other such factors. Towards the end of the report, I aim to have a solid conclusion about which ones are better for which scenarios and which ones are stronger than the rest, if at all.**

***Index Terms*—Ham , spam , Naïve Bayes , k-Nearest Neighbors, Support Vector Machines , Accuracy , Prediction of spam**

I. Introduction

THE world wide web, or the internet, as we know it, is a widely used platform for sharing knowledge and resources. Going five decades back in time, one couldn't even have imagined that sharing information, be it texts ,images, ideas or even seeing one another virtually would be such an easy task, as it is now. Everything can be sent from one place to another virtually. Just the click of a button enables you to see and hear people who live miles apart, send news and information in a jiffy, share your ideas with the world. Nothing seems too far anymore. However, with this ability to exchange ideas, thoughts, messages and news at lightning speed, comes a threat of falling prey to malicious intents of people who use the World Wide Web for wrong purposes such as fraud, cyber bullying and other forms of cyber-crime. The internet, as good as it seems, has some aspects that do not serve a clean purpose. E-mails are a major way of communicating over the internet. They are free, fast and not too informal. Ever since 1990s, people, including business professionals and students alike, have been using e-mails as a very important means of communication. E-mails, however can be used for purposes other than genuine communication. In this age, we have thousands of small businesses cropping up and people are trying to advertise their companies and products. As discussed earlier, the easiest way for such companies and groups to reach a larger audience is through the internet. These groups start sending e-mails in the form of promotions, deals, discounts and offers. While the intent of these e-mails aren't necessarily malicious, they are unwanted and people wouldn't want to have their mailboxes filled up with such undesired messages, which might actually lead to the loss of important data. These e-mails can be categorized as spam. While these are the non-malicious forms of spam e-mails, the more dangerous forms of spam e-mails are linked to phishing and data theft through these links. Some e-mails are unsolicited bulk e-mails, sent to people to trick them into clicking on links that can trace their important data such as IPs and passwords. This is a threat to data security. These spam e-mails can actually be very dangerous.

Spam e-mails are a subset of Electronic spam. Electronic spam can be defined as the use of electronic messaging systems used to send unsolicited messages, especially advertising, as well as sending messages on the same site[11]. Email spam, also known as unsolicited bulk email (UBE) or unsolicited commercial email (UCE), is the activity of sending unwanted email messages with commercial content, in large quantities to an indiscriminate set of recipients. The e-mails addresses of these recipients is gathered from sources as as forms and other social networking websites and web crawlers.

There are many types of spam e-mails. Some of them are:

- Unsolicited Advertisements

These are the spam e-mails that advertise products such as miraculous weight loss treatments, knock-off merchandise and online degrees. These messages are sent in bulk and they usually have some incentives to get the user to click on them.

- Phishing Scams

These spam e-mails are one of the harder to catch types of spam e-mails. Phishing is the activity of defrauding an online account holder of financial information by posing as a legitimate company[12]. These e-mails are made to look like official e-mails from big companies like PayPal, eBay etc. so as to get users to click on the links and sign in to their accounts. These account details are then used by site owners without the users finding out that they're on fake links.

- Nigerian 419 scams

These are the types of spam e-mails that are literally too good to be true, so good that they lure people into click on them to gain benefits such as thousands of dollars, prizes and other

monetary gifts. On clicking on these e-mails, users are asked to pay a comparatively smaller amount in the name of insurance or shipping. The scammers then send them a fake check. These scams are traceable but they have, in the past, caused many monetary thefts or losses.

- E-mail spoofing

While this isn't a category of spam e-mails by itself, it is a method used by spammers to make the users believe that they are clicking on legitimate links. Spammers use more realistic domains and email addresses to send these e-mails.

- Commercial advertisements

While commercial spam e-mails by themselves aren't malicious, they are definitely very irritating and undesirable. These can be sent from big of small companies to promote their products or business plans.

- Antivirus spam

These e-mails contain messages that say that the system that the user is using, is under threat. They come with malicious links that when clicked, can damage parts of the computer in terms of efficiency or space. These are very alluring since they look like they can solve an imminent problem.

Since the last decade and a half, a lot of importance is being given to e-mail spam filtering. Companies are coming up of ways with categorizing spam e-mails so that users so not fall into the traps mentioned above. E-mail systems such as Outlook, Gmail, Yahoo mail etc. have junk folders which store spam e-mails. Most companies also categorize e-mails as promotions or advertisements so that users need not be distracted by them while using their work, university of personal e-mails.

The results of such efforts can be seen in the graph below. There is a considerable decline in Average Daily Volume of spam e-mails in Aug-Oct 2010 as compared to the beginning of 2010.

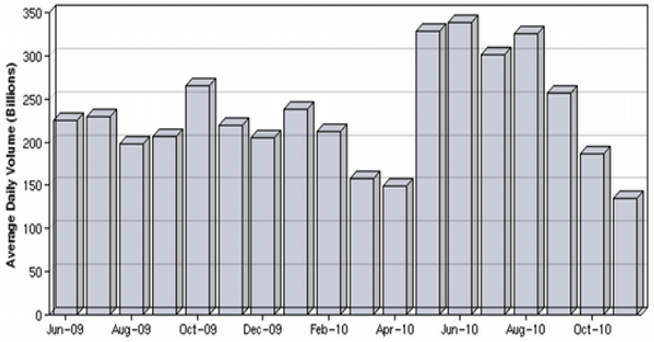


Figure 1: *Graphical representation of spam detection over the years*

In this project, I aim at implementing four spam filtering techniques that are widely used in various forms and as is. They are:

- Naïve Bayes Method for spam filtering
- K-Nearest Neighbors method for spam filtering
- Support Vector Machines

I shall be describing each of these methods in depth along with their implementations. I will also explain the methods used to implement them, the dataset used, the results in terms of accuracy and time taken as well as the drawbacks of each of them. Towards the end, I shall be discussing about the way any of these techniques have an upper hand over the others, if at all. Therefore, evaluating the techniques with respect to each other.

II. REVIEW OF LITERATURE AND INSPIRATION

PAUL Graham, in the year 2002, came up with a publication titled 'A Plan for Spam'. This was his attempt to familiarize users with the concept of spam and propose a solution for the same. He demonstrated a method termed as statistical filtering[13]. He describes the method of statistical spam filtering in the following steps.

1. Use one corpus of spam e-mails and one of non-spam e-mails. Scan through all parts of the spam and non-spam emails.
2. Consider alphanumeric characters,dashes ,apostrophes and dollar signs to be part of tokens, and everything else to be a token separator[13].
3. Count the number of times each token appears in the corpus. Create two large hash tables for doing this[13].
4. Create a third table, containing the probability that the e-mail the word I present in, is a spam-email[13].
5. Choose a bias to avoid false positives.
6. Calculate the accuracies.

This approach is called the statistical approach since it is based on the statistics of a word appearing in a spam e-mail or a non-spam e-mail.

Ever since, many statistical and non-statistical techniques have been used to filter out spam e-mails. Paul Graham, later in a conference in 2003 proposed spam fileting using Bayesian model. In the following versions of spam filtering algorithms, he made changes such as preserving the cases of words in e-mails, considering exclamations and other recurring punctuations, preserving tokens that appear in the 'To', 'From' and 'Subject' fields.

The 'spamcity' of a word is defined as the probability that a word is a part of a spam message. We set a threshold for the spamcity and if a word has a spamcity more than the threshold, it can be easily categorized as spam.

Moving forward with these basic methods, I shall be implementing a little more complex algorithms to detect spam e-mails.

III. IMPLEMENTATION OF THE PROJECT

THE project consists of 3 parts. They are:

- Data gathering and preprocessing
- Writing and implementing python programs to run the four algorithms mentioned above.
- Evaluation and analysis of results.

The first phase consisted of gathering possible spam e-mail corpuses to work the algorithms on. While there were simple text datasets of e-mails, I chose the UCI SpamBase due to the high volume of data that it provides. The SpamBase consists of 4601 instances. Each instance has 57 attributes. These attributes are further divided into continuous and non-continuous attributes. The main attributes are the 48 continuous attributes that are the word frequencies of words found in the instances. Other attributes are the total run length, longest run length etc. The general description of the dataset is as follows.

Dataset characteristics	Multivariate	**Number of instances**	4601
Attribute characteristics	Integer, Real	**Number of attributes**	57
Associated task	Classification	**Missing values?**	Yes

Table 1: Description of the Spam Base Dataset of UCI Machine Learning

In the following sections, I will describe all the methods that I have used for classification

IV. NAÏVE BAYES ALGORITHM

THE Bayesian Classification model, proposed by Thomas Bayes (1702 - 1761), is based on statistical and probabilistic method of learning. It is a type of supervised learning algorithm. Supervised learning is defined as the task of inferring a function from supervised training data[3]. A supervised training algorithm analyzes the training data and comes up with the classifier function and the regression function. We are going to deal with classification here. The inferred classification function performs the task of predicting the correct output value for a given input value. The Bayesian classifier assumes a probabilistic underlying model and determines probabilities of the outcomes. It can solve diagnostic as well as predictive problems. Naïve Bayes is based on conditional probabilities. Conditional probability is the probability of the occurrence of an event, given that another event has occurred.

For implementing Bayesian Classification for e-mail spam filtering, I used the Bayes theorem. The Bayes theorem in terms of spam and ham e-mails can be expressed as:

$$Pr(S|W) = Pr(W|S)\ Pr(S) / Pr(W|s)\ Pr(S) + Pr(W|H)\ Pr(H)$$

Where Pr(S|W) is the probability that a message is spam knowing the presence of a given word in it.
Pr(W|S) is the probability that a certain word appear in spam messages
Pr(S) is the overall probability of a message being a spam message.
Pr(W|H) is the probability that a certain word appears in ham messages.
Pr(H) is the overall probability that a message is a ham message[12].

As described in the implementation section, we have continuous attributes of categorical data. For classifying data with such attributes, I have used Gaussian Bayesian Filters. It is assumed that the continuous values are distributed along the Gaussian curve. The graph of a Gaussian is a symmetric bell shaped curve. To deal with such continuous attributes, let us say, training data x, We first segment the data by class. Then we have to compute the mean and variance of each class.

The mean is denoted by μ, standard deviation is denoted by σ and the variance is denoted by σ^2.

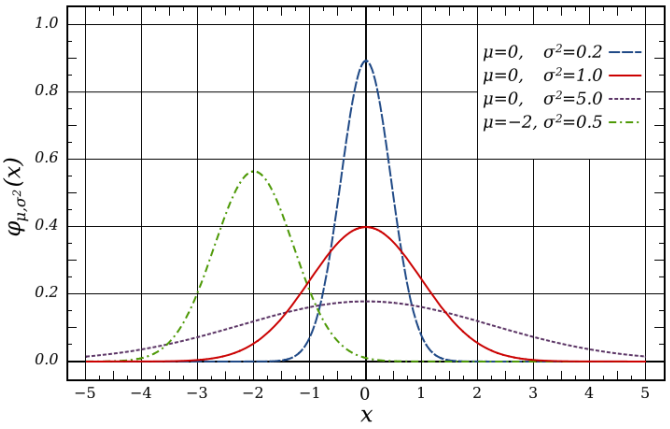


Figure 2: Plot of different values of mean, standard deviation and variance

The figure above, displays 4 Gaussian curves with means 0, 0, 0 and -2 respectively. Their standard deviations and variance are displayed in the figure.

μ is defined as the central tendency for the probability distribution. 'σ^2' is a measure of how far the set of data points are spread out in the distribution. 'σ' is the square root of the 'σ^2' which describes how close the data points are to the mean.

Then, the probability *distribution* of some value given a class, $p(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2[12]. That is,

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

Since we have to work with the same dataset as a test set and a training set, I first divide my dataset into train and test sets. My train/test ratio is 0.67.
I find the probabilities of the test set based on the training set. This gives me all the predictions for the testing data. Now that I have the predicted classification of an instance as spam or ham, and I also have their original classification, I can check how accurate my classification is. To calculate the accuracy, I check if the result of prediction is the same as the original dataset. If the values match, the classification is correct. I calculated the accuracy rate by dividing this by the total number of test data sets.

The results of my accuracies are as follows:

Time taken to run the algorithm on the SpamBase is "1.8 seconds"

The observed accuracy rate is 82.62%.

V. K- NEAREST NEIGHBORS

THE k Nearest Neighbors algorithm, also known as kNN, is a non-parametric method used for pattern classification. It is an instance-based learning algorithm. The function is simply approximated locally. All computation is deferred until classification. The neighbors for the data points are taken as a set of objects whose values are known. This is basically the training phase of the algorithm. In the classification phase of the algorithm, for each point in the testing set, we consider k neighbors where k is predefined by the programmer. We then find the most frequent class values in the training set. We can find the neighbors using many methods such as linear search, space partitioning, locality sensitive hashing etc[7].

For implementing the algorithm, I have used the Euclidean distance between the attributes of the instances. The squared Euclidean distance between the point $p = (p_1, p_2, ..., p_n)$ and the point $q = (q_1, q_2, ..., q_n)$ is the sum of the squares of the differences between the components: $Dist^2(p,q) = \Sigma i\ (pi - qi)^2$. The Euclidean distance is then the square root of $Dist^2(p,q)$[10].

While Euclidean distance is a good measure for finding all the neighbors of a given instance, it has a major drawback that it cannot work for very large data sets as the computational time is very high. I am going to demonstrate this by implementing it on the 4091 instances that I have in the dataset.

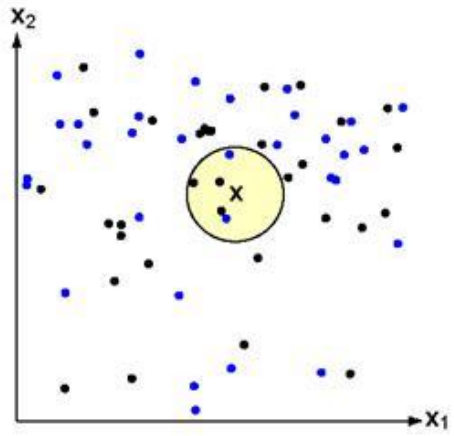


Figure 3: Plot depicting 5 nearest neighbors of x

The above figure depicts the general idea of the k Nearest Neighbors algorithms. There are two classes, blue and black. We have a data point x that needs to be classified. Considering that the value of k is taken as 5, we first find the distance of x from all of the given points. All the distances are then sorted in ascending order and the first 5 distances are considered. We are thus considering the 5 points that are nearest to point x. We then check the maximum frequency of points within the neighborhood of x. As we can see, there are 3 black points and 2 blue points. So the maximum frequency is 3 and it belongs to black. X is therefore classified as a black point.

To implement the kNN algorithm, my program divides the SpamBase into a testing dataset and a training dataset. There is no preprocessing on the training dataset. We directly calculate the k nearest neighbors for each data point in the testing dataset. Then among these k values, I calculate whether the e-mail is mostly classified as ham or spam. If majority of the k neighbors are classified as spam, the data point or the instance is classified as spam. Otherwise it is classified as ham. Then we move on to calculate the accuracy by checking the value of the ham/spam to the original class value.

Dependency of the value of k on the accuracy of the algorithm

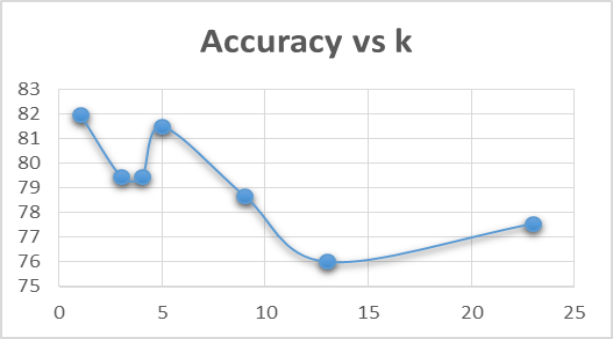
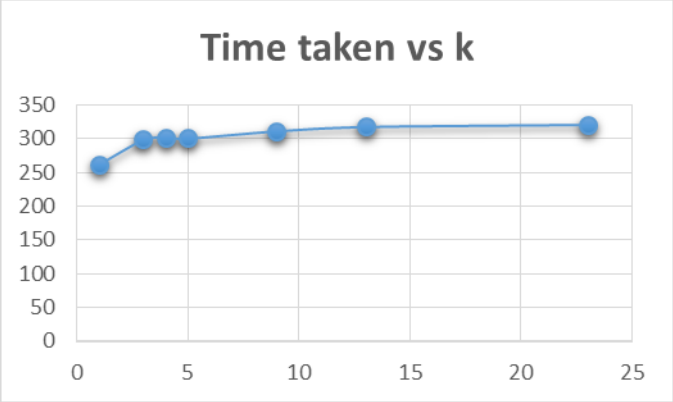


***FIGURE 4**: PLOT SHOWING THE ACCURACY OF KNN FOR DIFFERENT VALUES OF K*

During implementation, I used 5 values of k to see how the accuracy differed on changing the values. The 5 values of k are {1,3,4,5,9,13,23}. The accuracies of the values are {81.98%,79.44%,79.47,81.48%,78.67%,76.01%,77.55% }

Based on the observations above, I infer that the value of k does change the accuracy of the algorithm to an extent. However, the change in accuracy also depends on the dataset distribution. If the dataset is evenly distributed or if it is dense or sparse, can change the effect of k on the efficiency of the algorithm.

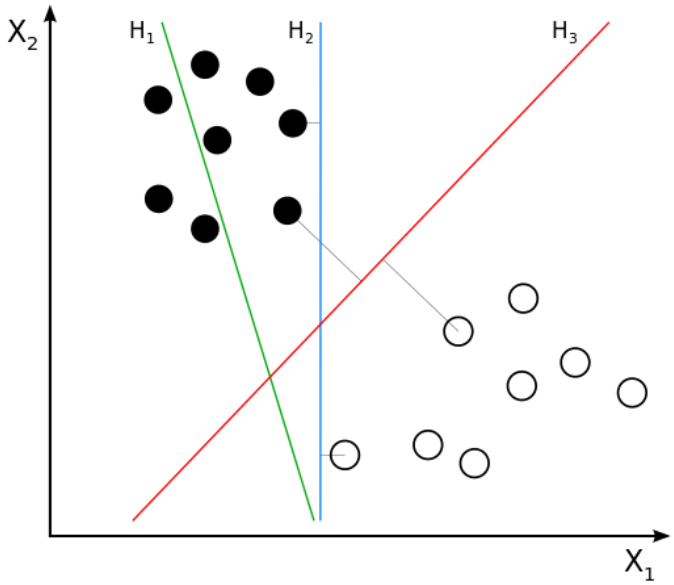
Dependency of the value of k on the time taken for the algorithm to classify data



The value of k is directly proportional to the time taken by the algorithm. On running the algorithm with the same values of k, i.e. {1,3,4,5,9,13,23} the execution times, in terms of seconds were observed to be: {261.988,298.674,301.727,300.654,311.453,317.893,321.07} As we can see, the time taken continually increases with the value of k. This can be linked to the fact that it takes more time to find more nearest neighbors.

VI. SUPPORT VECTOR MACHINES

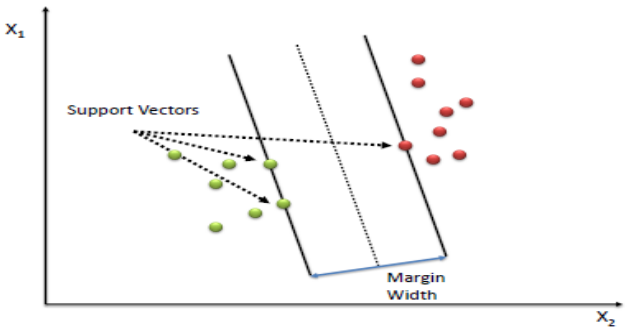
SUPPORT Vector Machines, also called as SVM are supervised learning models that analyze data and recognize patterns. If we have a set of training data points, each marked with a class value such as [0,1] , [1,-1] , [TRUE,FALSE] , on being fed with new data points, a support vector machine can categorize them into one of the two classes. Therefore, a support vector machine is a non-probabilistic binary linear classifier. How a support vector machine works is, it produces a hyper plane on a high dimensional space or a set of hyper planes on an infinite dimension space. Classification of data points is done around these hyper planes. For two classes with linearly separable data, there may be many such separators that can separate the two classes of data. Intuitively, it seems like a decision boundary that is drawn in the middle of the void between the two classes seems to be the best positioning of the decision boundary.



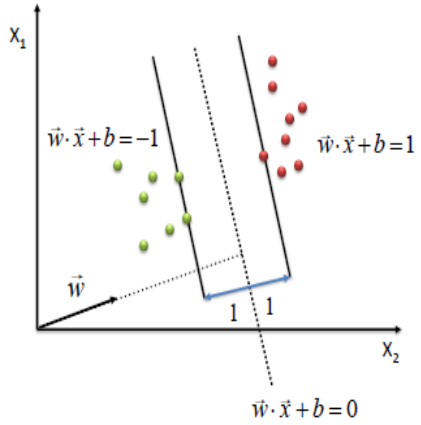
Consider the figure above, the line H1 is a bad choice for a decision boundary because it does not separate the black points and the white points into two different classes. Therefore, H1 is eliminated as an option. Now consider the line H2. H2 is a valid decision boundary, but not a good one. The reason being, the margin

While some learning methods such as the perceptron algorithm find just any linear separator, others, like Naive Bayes, search for the best linear separator according to some criterion. The SVM defines the criterion to be looking for a decision surface that is maximally far away from data points in either of the two linearly separable classes. The margin is this distance which is the distance between the nearest point to either of the classes and the hyper plane that linearly separates them. This method of construction necessarily means that the decision function for an SVM is fully specified by a (usually small) subset of the data which defines the position of the separator[14]. The points, that lie on the hyper plane are called as support vectors. Other data points play no part in determining the decision surface that is chosen[6].

Margin maximization in support vector machines
The figure given below explains and depicts the term support vectors and margin.



To find an optimal hyper plane, the main objective of the algorithm is to maximize the margin. The margin is the distance between points belonging to either of the classes with the hyper plane. By support vectors, we mean the vectors that define the hyper plane. To maximize the margin, the following steps are followed.



$$\max \frac{2}{\|w\|}$$

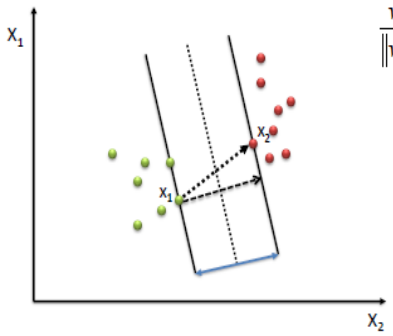
s.t.

$(w\cdot x+b)\geq 1, \forall x$ of class 1

$(w\cdot x+b)\leq -1, \forall x$ of class 2

As we can see, we have to maximize the value highlighted in the red box. Here *b* is the bias. The dot product of the margin with the vector x is -1 for the support vectors marked with the color green. The same value is 1 for the support vectors marked with red color. This gives rise to two classes that are Class 1 and Class 2. Class 1 are all the red points and their property is that the value of the dot product of the two vectors w and x with an added bias is greater than or equal to 1. On the other hand, we have the dot product of the width vector with the vector x added with the biases, if the value is less than -1, the data point belongs to Class 2.

We can define width as unit width multiplied by the distance i.e (x2-x1). This is further described in the next figure



$$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$$

$$w\cdot x_2+b=1$$

$$w\cdot x_1+b=-1$$

$$w\cdot x_2+b-w\cdot x_1-b=1-(-1)$$

$$w\cdot x_2-w\cdot x_1=2$$

$$\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}$$

We define the width in terms of the unit vector w and use the equations for the support vector to derive that the width can be written as 2/||w||.

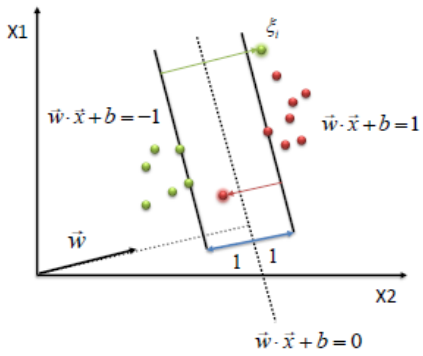
This would be the same as minimizing the inverse. That is.

$$\min \frac{1}{2}\|w\|^2$$

$$s.t.\ y_i(w\cdot x_i+b)\geq 1,\ \forall x_i$$

The thing about SVM is that if there is linear separability, then the value of global minimum is a unique value.

However, it might be very difficult, in some cases to come up with data points that are purely linearly separable. So we introduce the concept of slack variables. Slack variables may be defines as the variables that may break the rule of linear separability, but they will be penalized.

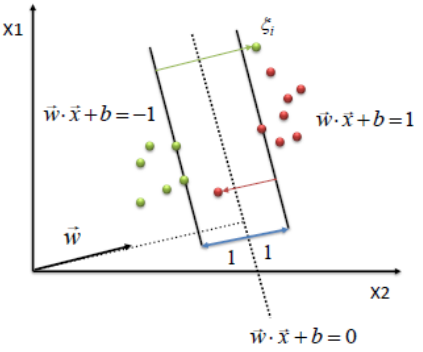


slack variable:

ξ_i

Allow some instances to fall off the margin, but penalize them

As seen above, we introduce the slack variables denoted by ς. So now the value of the equations mentioned above changes by the effect of the slack variables.



Constraint becomes :

$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$

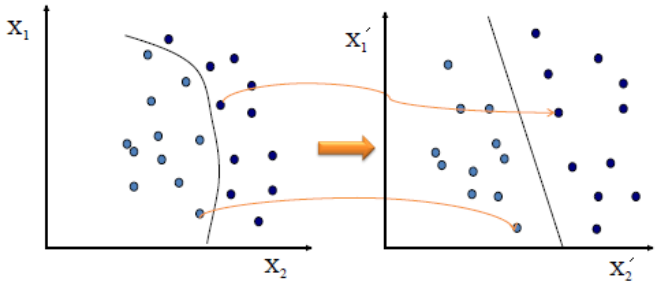
$$\xi_i\geq 0$$

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

The above descriptions have been about hyper planes that are linear. There may, however be cases such that a non linear hyper plane may be more efficient in separating data points than a linear hyper plane. The SVM handles these situation with a trick known as the 'kernel trick'[7]. The kernel trick may be defined as the use of a function that maps data into a different space where the hyper plane cannot separate the data. This function is called as the kernel function. The kernel trick therefore is using the kernel function to transform the data into a higher dimensional space so as to separate them with a hyper plane of a higher dimension. So the data is mapped into a new space and the dot product is then performed in the new space.



This figure defines how the data points are mapped in a new space and how they can be linearly separated in the new space.

To implement support vector machines on my dataset, I use two sets of my Spam Base. They're the training set and the testing set. The training set consists of all 57 columns of the dataset including the class column. The testing set includes all the columns except the class. I am using the inbuilt Python functions for SVM that is the SVC function and the NuSVC function. These are used by importing the SKLEARN package in Python.

I store the results in two class files. I am also storing the probabilities in a file. The results look as follows.

KNN.py × | result_probs.txt × | result.txt × | *Untitled Document 1 ×

```
[ 0.34933096  0.65066904]                1.0
[ 0.07007541  0.92992459]                1.0
[ 0.50984614  0.49015386]                0.0
[ 0.32249783  0.67750217]                0.0
[ 0.32251346  0.67748654]                1.0
[ 0.86211873  0.13788127]                0.0
[ 0.86346792  0.13653208]                0.0
[ 0.83763537  0.16236463]                0.0
[ 0.50984614  0.49015386]                0.0
[ 0.17134721  0.82865279]                1.0
[ 0.88809916  0.11190084]                0.0
```

The first column shows the probability of the data point being a ham message and the second column gives a probability of the dataset being a spam message. The result, that is actually stored in the result.txt file, but is being here for representation purposes, considers the maximum of the two probabilities and assigns ham or spam class to the data point at hand. The result '1.0' indicates a spam message and '0.0' indicates a spam message.

I have observed a running time of 12-15 seconds on running this algorithm on my datasets.

I have tested two versions of the algoritms by using two different functions, namely svm.SVC() and svm.NuSVC(). The are basically the same performance with different function parameters. The range of SVC is between 0 and infinity while the range of NuSVC is always between [0,1]. I observed that the function NuSVC works faster than the function SVC.

The following table depicts the difference between the execution time of SVC and NuSVC with the Spam Base dataset.

Execution time for SVC	Execution time for NuSVC
15.4674352	12.1236781
14.9845072	11.2356093
17.3467981	11.2546890

Therefore on an average, the function svm.NuSVC() seems to be 20% faster than svm.SVC().

VII. CONCLUSION

WE have seen the running times and accuracy rates of all the three algorithms. I am tabulating all of the results so that I can present the conclusion to this project.

Algorithm/Approach	**Accuracy Rate**	**Execution Time**
Naïve Bayes (First implementation)	82.62%	1.8 seconds
Naïve Bayes(Second implementation)	84.58%	1.4 seconds
K Nearest Neighbors (k = 1)	81.98%	261.988
K Nearest Neighbors (k = 3)	79.44%	298.674
K Nearest Neighbors (k = 4)	79.47%	301.727
K Nearest Neighbors (k = 5)	81.48%,	300.654
K Nearest Neighbors (k = 9)	78.67%	311.453
K Nearest Neighbors (k = 13)	76.01%	317.893
K Nearest Neighbors (k = 23)	77.55%	321.07
SVM (First Implementation using svm.SVC())	80.67%	15.4674352
SVM (second implementation using svm.SVC())	79.45%	14.9845072
SVM(First implementation using svm.NuSVC())	79.93%	12.1236781
SVM(Second implementation using svm.NuSVC())	81.06%	11.2356093

The best results would be the algorithm where the accuracy as well as the execution time are optimized. As we can see above, the Naïve Bayes algorithm gives the least execution time. It also surpasses the other two algorithms in terms of the accuracy. Support Vector machines also have high accuracies and low run times. The K Nearest Neighbors algorithm gives

the worst performance in terms of both the accuracy and the run time. The run time of KNN is approximately 255 times that of Naïve Bayes and 197 times that of Support Vector Machines. Therefore,

Naïve Bayes classifier is the best algorithm among Naïve Bayes, SVM and K Nearest Neighbors for classifying e-mails as spam while taking into consideration, their accuracies and execution times.

While there is a stark difference in the performance of the three algorithms, these could vary with different data sets. The ratio of ham/spam messages, the number of data points in the training set and the number of data points in the testing set can have an effect on the performance of the algorithms.

REFERENCES

[1] Vangelis Metsis, *Spam Filtering with Naive Bayes -- Which Naive Bayes?,* Vangelis Metsis Telecommunications (2006)

[2] Freund, Y.: *Boosting a weak learning algorithm by majority*. Information and Computation 121(2), 256–285 (1995) .

[3] Bartlett, P.L., Traskin, M., *AdaBoost is consistent*. Journal of Machine Learning Research 8, 2347–2368 (2007)

[4] Drucker, H., Wu, D., Vapnik, V.N, *Support Vector Machines for Spam Categorization.* IEEE Transactions on Neural Networks 10(5), 1048–1054 (September 1999)

[5] 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)

[6] Bickel, P.J., Ritov, Y., Zakai, A., *Some theory for generalized boosting algorithms.* Journal of Machine Learning Research 7, 705–732 (2006)

[7] Ratsch, G., Onoda, T., M ¨ uller, K.R.: *Soft margins for AdaBoost*. Machine Learning ¨ 42(3), 287–320 (2001)

[8] Ting Fan Wu, Chih-Jen Lin, Ruby C. Weng, *Probability estimates of multiclass classification by pairwise coupling, Journal of Machine Learning research* 5, 975-1005 (2004)

[9] La Bouli, Yu Shiven, Lu Qin , *An improved k Nearest Neighbors algorithm for text Categorization* (2002)

[10] Manning C. D. and Schutze H., 1999. *Foundations of Statistical Natural Language Processing* [M]. Cambridge: MIT Press.

[11] www.en.wikipedia.org

[12] www.alum.cs.sunyb.edu

[13] www.paulgraham.com/spam.html

[14] www.nlp.stanford.edu