PROJECT REPORT ON SPAM CLASSIFIER USING SVM

SUBMITTED BY:

NISHTHA GOEL 2K12/CO/080

PALAK JAIN 2K12/CO/083

VRINDA BHATIA 2K12/CO/145

1. INTRODUCTION

1.1 MACHINE LEARNING

Arthur Samuel defined machine computers the ability to learn without being explicitly programmed. Samuel's claim to fame was that back in the 1950's, he wrote a checkers playing program. And the amazing thing about this checkers playing program, was that Arthur Samuel himself, wasn't a very good checkers player. But what he did was, he had to program for it to play 10's of 1000's of games against itself. And by watching what sorts of board positions tended to lead to wins, and what sort of board positions tended to lead to losses. The checkers playing program learns over time what are good board positions and what are bad board positions. And eventually learn to play checkers better than Arthur Samuel himself was able to. This was a remarkable result. Although Samuel himself turned out not to be a very good checkers player. But because the computer has the patience to play tens of thousands of games itself, No human, has the patience to play that many games. By doing this the computer was able to get so much checkers-playing experience that it eventually became a better player.

Tom Mitchell, defines machine learning by saying that, a well posed learning problem is defined as follows. He says,

"A computer program is said to learn from experience E, with respect to some task T, and some performance measure P, if its performance on T as measured by P improves with experience E. "

For the checkers playing example the experience e, will be the experience of having the program play 10's of 1000's of games against itself. The task t, will be the task of playing checkers. And the performance measure p, will be the probability that it wins the next game of checkers against some new opponent.

There are several different types of learning algorithms. The main two types

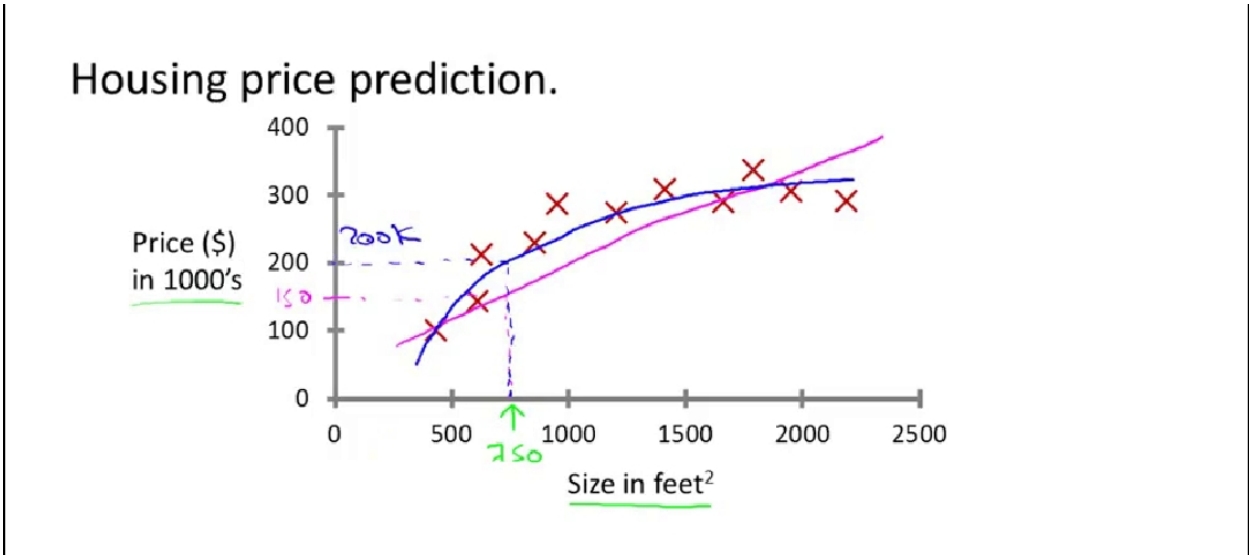
1.Supervised learning and

2.Unsupervised learning.

In supervised learning, the idea is that we're going to teach the computer how to do something, whereas in unsupervised learning we're going let it learn by itself. You will also hear other buzz terms such as reinforcement learning and recommender systems. These are other types of machine learning algorithms, but the two most used types of learning algorithms are probably supervised learning and unsupervised learning.

1.2 SUPERVISED LEARNING ALGORITHM

The term supervised learning refers to the fact that we gave the algorithm a data set in which the "right answers" were given.



For example, let's say you want predict housing prices. A while back, a student collected data sets from the Institute of Portland Oregon. And let's say you plot a data set and it looks like this. Here on the horizontal axis, the size of different houses in square feet, and on the vertical axis, the price of different houses in thousands of dollars. So, given this data, let's say you have a friend who owns a house that is, say 750 square feet and hoping to sell the house and they want to know how much they can get for the house. So how can the learning algorithm help you? One thing a learning algorithm might be able to do is put a straight line through the data or to fit a straight line to the data and, based on that, it looks like maybe the house can be sold for maybe about $150,000. But maybe this isn't the only learning algorithm you can use. There might be a better one. For example,

instead of sending a straight line to the data, we might decide that it's better to fit a quadratic function or a second-order polynomial to this data. And if you do that, and make a prediction here, then it looks like, well, maybe we can sell the house for closer to $200,000. So this is an example of a supervised learning algorithm. And the term supervised learning refers to the fact that we gave the algorithm a data set in which the "right answers" were given. That is, we gave it a data set of houses in which for every example in this data set, we told it what is the right price so what is the actual price that, that house sold for and the toss of the algorithm was to just produce more of these right answers such as for this new house, you know, that your friend may be trying to sell. To define with a bit more terminology this is also called a **regression problem** and by regression we mean we're trying to predict a continuous value output.

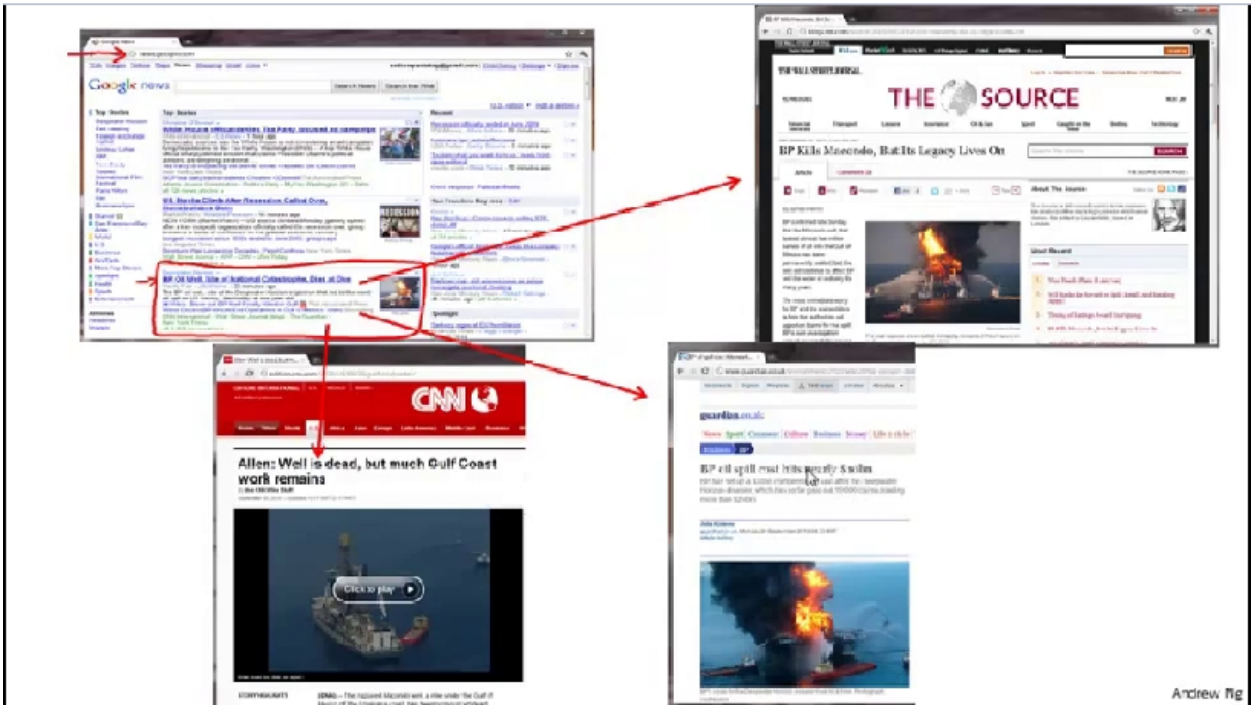
But it turns out that for some learning problems, what you really want is not to use, like, three or five features. But instead, you want to use an infinite number of features, an infinite number of attributes, so that the learning algorithm has lots of attributes or features or cues with which to make those predictions. So how do you deal with an infinite number of features? How do you even store an infinite number of things on the computer when your computer is going to run out of memory. It turns out that when we talk about an algorithm called the **Support Vector Machine**, there will be a neat mathematical trick that will allow a computer to deal with an infinite number of features. Turns out, we'll be able to come up with an algorithm that can deal with infinite number of features. In supervised learning, our data set, we are told what is the "correct answer" that we would have quite liked the algorithms have predicted on that example. By regression problem, we mean that our goal is to predict a continuous valued output. The **classification problem** is where the goal is to predict a discrete value output.

1.3 UNSUPERVISED LEARNING

In Unsupervised Learning, we're given data that looks different than data that looks like this that doesn't have any labels or that all has the same label or really no labels. So we're given the data set and we're not told what to do with it and we're not told what each data point is. Instead we're just told, here is a data set. Can you find some structure in the data? Given this data set, an Unsupervised Learning algorithm might decide that the data lives in two different clusters. And so there's

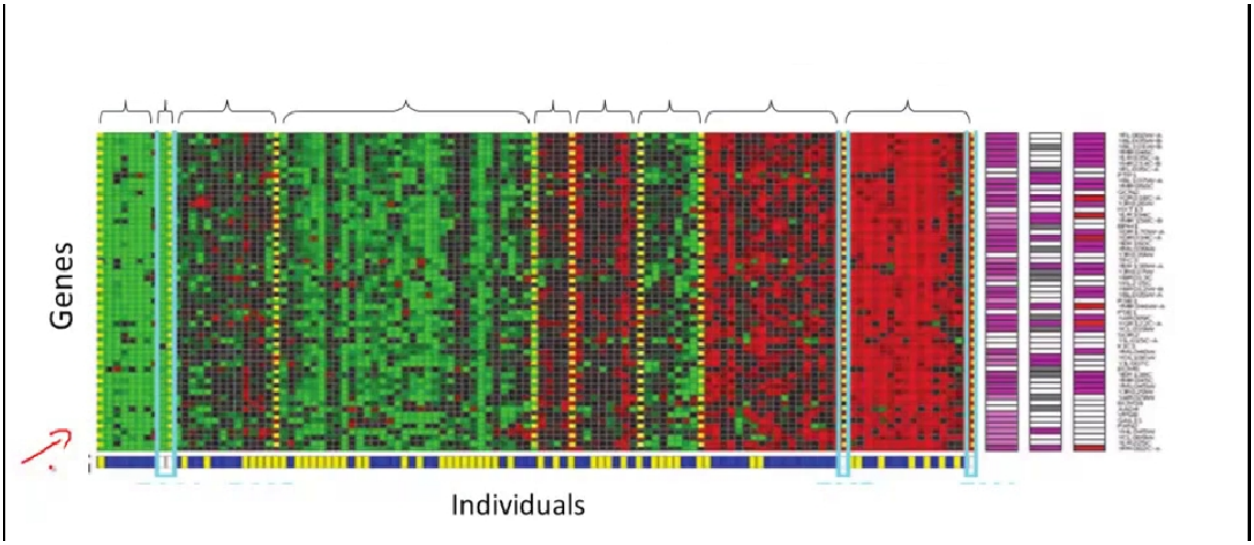
one cluster and there's a different cluster. And yes, Supervised Learning algorithm may break these data into these two separate clusters. So this is called a **clustering algorithm**. And this turns out to be used in many places.

One example where clustering is used is in **Google News** and if you have not seen this before, you can actually go to this URL news.google.com to take a look. What Google News does is everyday it goes and looks at tens of thousands or hundreds of thousands of new stories on the web and it groups them into cohesive news stories. For example, let's look here.



The URLs here link to different news stories about the BP Oil Well story. So, let's click on one of these URL's and we'll click on one of these URL's. What I'll get to is a web page like this. Here's a Wall Street Journal article about, you know, the BP Oil Well Spill stories of "BP Kills Macondo", which is a name of the spill and if you click on a different URL from that group then you might get the different story. Here's the CNN story about game, the BP Oil Spill, and if you click on yet a third link, then you might get a different story. Here's the UK Guardian story about the BP Oil Spill. So what Google News has done is look for tens of thousands of news stories and automatically cluster them together. So, the news stories that are all about the same topic get displayed together. It turns out that clustering algorithms and Unsupervised Learning algorithms are used in many other problems as well.

Here's one on understanding genomics. Here's an example of DNA microarray data.



The idea is put a group of different individuals and for each of them, you measure how much they do or do not have a certain gene. Technically you measure how much certain genes are expressed. So these colors, red, green, gray and so on, they show the degree to which different individuals do or do not have a specific gene. And what you can do is then run a clustering algorithm to group individuals into different categories or into different types of people. So this is Unsupervised Learning because we're not telling the algorithm in advance that these are type 1 people, those are type 2 persons, those are type 3 persons and so on and instead what we are saying is yeah here's a bunch of data. I don't know what's in this data. I don't know who's and what type. I don't even know what the different types of people are, but can you automatically find structure in the data. Because we're not giving the algorithm the right answer for the examples in my data set, this is Unsupervised Learning. Unsupervised Learning or clustering is used for a bunch of other applications.

2. LITERATURE REVIEW

Machine learning techniques now days used to automatically filter the spam e-mail in a very successful rate. We review some of the most popular machine learning methods such as Bayesian classification, k-NN, ANNs, SVMs, Artificial immune system and Rough sets.

2.1 Naïve Bayes classifier method

In 1998 the Naïve Bayes classifier was proposed for spam recognition. Bayesian classifier is working on the dependent events and the probability of an event occurring in the future that can be detected from the previous occurring of the same event . This technique can be used to classify spam e-mails; words probabilities play the main rule here. If some words occur often in spam but not in ham, then this incoming e-mail is probably spam. The statistic we are mostly interested for a token T is its spamminess (spam rating) [10], calculated as follows:

$$S[T] = \frac{C_{Spam}(T)}{C_{Spam}(T) + C_{Ham}(T)}$$

Where CSpam(T) and CHam(T)are the number of spam or ham messages containing token T, respectively. To calculate the possibility for a message M with tokens {T1,.....,TN}, one needs to combine the individual token's spamminess to evaluate the overall message spamminess. A simple way to make classifications is to calculate the product of individual token's spamminess and compare it with the product of individual token's hamminess The message is considered spam if the overall spamminess product S[M]is larger than the hamminess product H[M]. The above description is used in the following algorithm [10]:

Stage1. Training Parse each email into its constituent tokens Generate a probability for each token W

S[W] = Cspam(W) / (Cham(W) + Cspam(W))

store spamminess values to a database

Stage2. Filtering

For each message M while (M not end) do scan message for the next token Tiquery the database for spamminess S(Ti) calculate accumulated message probabilities S[M] and H[M] Calculate the overall message filtering indication by: I[M] = f(S[M] , H[M]) f is a filter dependent function, such as

$$I[M] = \frac{1+S[M]-H[M]}{2}$$

if I[M] > threshold msg is marked as spam else msg is marked as non-spam.

2.2 K-nearest neighbour classifier method

The k-nearest neighbour (K-NN) classifier is considered an example-based classifier, that means that the training documents are used for comparison rather than an explicit category representation, such as the category profiles used by other classifiers. As such, there is no real training phase. When a new document needs to be categorized, the k most similar documents (neighbours) are found and if a large enough proportion of them have been assigned to a certain category, the new document is also assigned to this category, otherwise not . Additionally, finding the nearest neighbours can be quickened using traditional indexing methods. To decide whether a message is spam or ham, we look at the class of the messages that are closest to it. The comparison between the vectors is a real time process. This is the idea of the k nearest neighbor algorithm:

Stage1. Training Store the training messages.

Stage2. Filtering Given a message x, determine its k nearest neighbours among the messages in the training set. If there are more spam's among these neighbours, classify given message as spam. Otherwise classify it as ham.

The use here of an indexing method in order to reduce the time of comparisons which leads to an update of the sample with a complexity O(m), where m is the sample size. As all of the training examples are stored in memory, this technique is also referred to as a memory-based classifier [6]. Another problem of the presented algorithm is that there seems to be no parameter that we could tune to reduce the number of false positives. This problem is easily solved by changing the classification rule to the following l/k-rule:

If l or more messages among the k nearest neighbours of x are spam, classify x as spam, otherwise classify it as legitimate mail.

2.3 Artificial Neural Networks classifier method

An artificial neural network (ANN), also called simply a "Neural Network" (NN), is a computational model based on biological neural networks. It consists of an interconnected collection of artificial neurons. An artificial neural network is an adaptive system that changes its structure based on information that flows through the artificial network during a learning phase. The ANN is based on the principle of learning by example. There are, however the two classical kind of the neural networks, perceptron and the multilayer perceptron. Here we will focus on the perceptron algorithm. The idea of the perceptron is to find a linear function of the feature vector $f(x) = wTx + b$ such that $f(x)> 0$ for vectors of one class [2], and $f(x)< 0$ for vectors of other class. Here w = (w1w2,…wm) is the vector of coefficients (weights) of the function, and b is the so-called bias. If we denotethe classes by numbers +1 and -1, we can state that we search for a decision function d(x)= sign (wTx + b). The perceptron learning is done with an iterative algorithm. It starts with arbitrarily chosen parameters (w0,b0) of the decision and updates them iteratively. On the n-th iteration of the algorithm a training sample (x,c) is chosen such that the current decision function does not classify it correctly (i.e. sign (wnx + bn) $\neq$ c). The parameters (wn,bn) are then updated using the rule:

$wn+1= wn+ cx$

$bn+1 = bn+ c$

The algorithm stops when a decision function is found that correctly classifies all the training samples.

2.4 Support Vector Machines classifier method

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships, the SVM modeling algorithm finds an optimal hyperplane with the maximal margin to separate two classes, which requires solving the following optimization problem.

Maximize

$$\sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i , x_j)$$

Subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

where $0 \le \alpha i \le b, i = 1, 2, \ldots n$

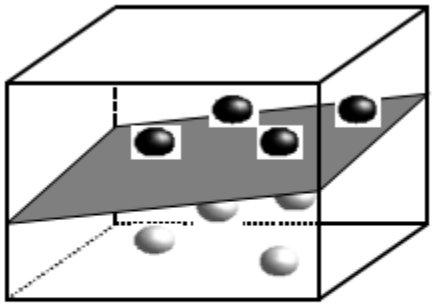


Figure 1 An SVM separating black and white points in 3 dimensions

Where αi is the weight of training samplex1. If αi > 0, x1is called a support vector bis a regulation parameter used to trade-off the training accuracy and the model complexity so that a superior generalization capability can be achieved. Kis a kernel function, which is used to measure the similarity between two samples. A popular radial basis function (RBF)kernel functions.

After the weights are determined [9], a test sample x is classified by

$$y = Sign \left(\sum_{i=1}^{n} \alpha_i y_i K(x_i , x_j) \right),$$

$$Sign(a) = \begin{cases} +1, & if\ a > 0 \\ -1, & otherwise \end{cases}$$

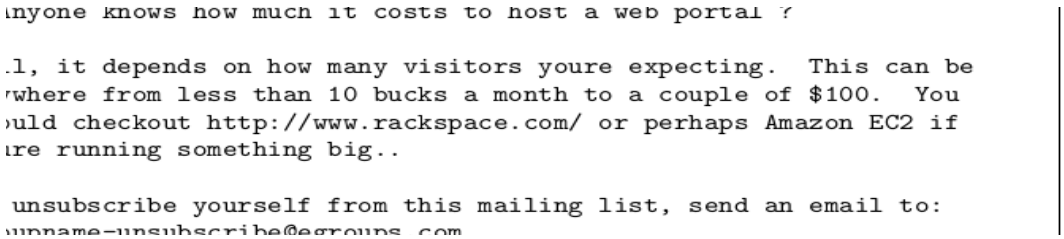
To determine the values of < γ, b >, a cross validation process is usually conducted on the training dataset . Cross validation is also used to estimate the generalization capability on new samples that are not in the training dataset. A k-fold cross validation randomly splits the training dataset into k approximately equal-sized subsets, leaves out one subset, builds a classifier on the remaining samples, and then evaluates classification performance on the unused subset. This process is repeated k times for each subset to obtain the cross validation performance over the whole training dataset. If the training dataset is large, a small subset can be used for cross validation to decrease computing costs.

3.THE PROBLEM

Here,the problem is to obtain a decision function f,that can tell us whether a given email is a spam or non-spam.

3.1 WHAT ARE SPAM EMAILS?

Spam emails are the nearly identical emails sent to a bulk of people. These emails sometimes contain some links which may direct the recipient to some phishing website or to the sites that host malware. Spam emails also include attachments that may be harmful to the computer. Spammers collect the email addresses various chatrooms, customer lists, websites .These may also include some offers at various commercial websites which are sent in bulk to all the users who have signed up on that website. Most of the times these spam messages includes images in their mails so that they don't get blocked by the word filters.In fact ,when a viewer views the image,it is directly downloaded from the spammer's website so now the spammer will know exactly which users have viewed the image so that they can spam those email addresses more.


anyone knows how much it costs to host a web portal ?

well, it depends on how many visitors youre expecting. This can be
anywhere from less than 10 bucks a month to a couple of $100. You
should checkout http://www.rackspace.com/ or perhaps Amazon EC2 if
youre running something big..

To unsubscribe yourself from this mailing list, send an email to:
groupname-unsubscribe@egroups.com

Fig 3.1 A SAMPLE SPAM EMAIL

3.2 HOW DO WE CLASSIFY EMAILS AS SPAM OR NON SPAM?

Since these unwanted emails can cause harm to our computers and even more than that,so we need to classify the inbox emails as spam or non-spam so that we don't open the spam email by mistake. We train our machine to classify the emails by inputting lot of emails.The machine processes those emails by creating a decision boundary and learn from those emails. Then we test it for some dataset.

Now the machine the same decision boundary to classify the emails.If we get good results for the dataset then we can test the model for our own emails.

4. PROPOSED MODEL

The spam classification algorithm we will be using is Support Vector Machine, popularly known as SVM.

4.1 INTRODUCTION TO SVM

SVM is one of the most popular algorithm used widely in industry as well as academia. It gives a cleaner and more powerful picture for solving nonlinear functions as compared to neural networks or logistic regression.

SVMs are based on the concept of decision boundaries, which separates a set of objects having different class memberships. The SVM algorithm finds an optimal boundary having maximum margin to separate the two classes, which requires solving the optimization objective given in next section.

4.2 OPTIMIZATION OBJECTIVE

The optimization objective is to find a solution for the following equation such that it minimizes 'Θ' :

where, $$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

C=Regularization parameter,

$cost_1\ (\Theta^T x^{(i)}) = -\log(h_\Theta x^{(i)})$,

$cost_0\ (\Theta^T x^{(i)}) = -\log(1-h_\Theta x^{(i)})$,

Θ=Parameter vector,

$h_\Theta x$=Hypothesis and

$h_\Theta x = 1$, if $\Theta^T x >= 0$ and
$= 0$, otherwise

We have used **SMO** or the Sequential Minimal Optimizatin algorithm, which is an approximate of SVM software package-*libsvm* to solve for the parameter Θ.

4.3 DECISION BOUNDARY

Fig 4.1 : $cost_1 (\Theta^T x^{(i)})$

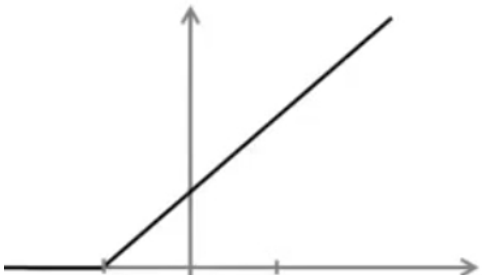


Fig 4.2 : $cost_0 (\Theta^T x^{(i)})$

As we can see from Fig 4.1 and Fig 4.2 ,

Whenever we want $y^{(i)} = 1$, $(\Theta^T x^{(i)}) > 1$
and if we want $y^{(i)} = 0$, $(\Theta^T x^{(i)}) < -1$

So, by above calculations, we get a decision boundary which is a large margin classifier between the different classes of datasets. The decision boundary can be linear as well as non-linear.

4.4 SPAM CLASSIFICATION ALGORITHM

KERNEL

We have used linear kernel , or no kernel in our implementation of SVM.

REGULARIZATION PARAMETER

We have chosen C=0.1 after experimenting with many other C values.

VOCABULARY LIST

We have created our vocabulary list by choosing all words which occur at least a 100 times in the spam corpus, resulting in a list of 1899 words. This list of words will be used to judge whether an email contains spam words or no.

LANGUAGE/SOFTWARE USED

We have used Octave to code our spam classification algorithm.

THE ALGORITHM :

1. Preprocess or normalize each email

While many emails would contain similar types of entities (e.g.,numbers, other URLs, or other email addresses), the specific entities (e.g.,the specific URL or specific dollar amount) will be different in almost every email. Therefore, one method often employed in processing emails is to “normalize” these values, so that all URLs are treated the same, all numbers are treated the same, etc.

1.1 Convert entire email to lowercase
1.2 Strip all HTML tags
1.3 Normalize URLs
1.4 Normalize email addresses
1.5 Normalize numbers
1.6 Remove non-alphanumeric characters

2. Create word index for each email

Given the vocabulary list, we now map each word in the preprocessed emails into a list of word indices that contains the index of the word in the vocabulary list.

3. Construct feature vector

We now implement the feature extraction that converts each email into a vector in $R^{n,}$ where n=number of words in vocabulary list. Specifically, the feature $x^i \in \{0, 1\}$ for an email corresponds to whether the i-th word in the dictionary occurs in the email. That is, $x^i = 1$ if the i-th word is in the email and $x^i = 0$ if the i-th word is not present in the email.

4. Train SVM for spam classification algorithm

We have divided our dataset into training set and test set. The training set contains 4000 examples of emails while the test set contains 1000 examples.Each original email was pre-processed using the processEmail and emailFeatures functions and converted into a vector

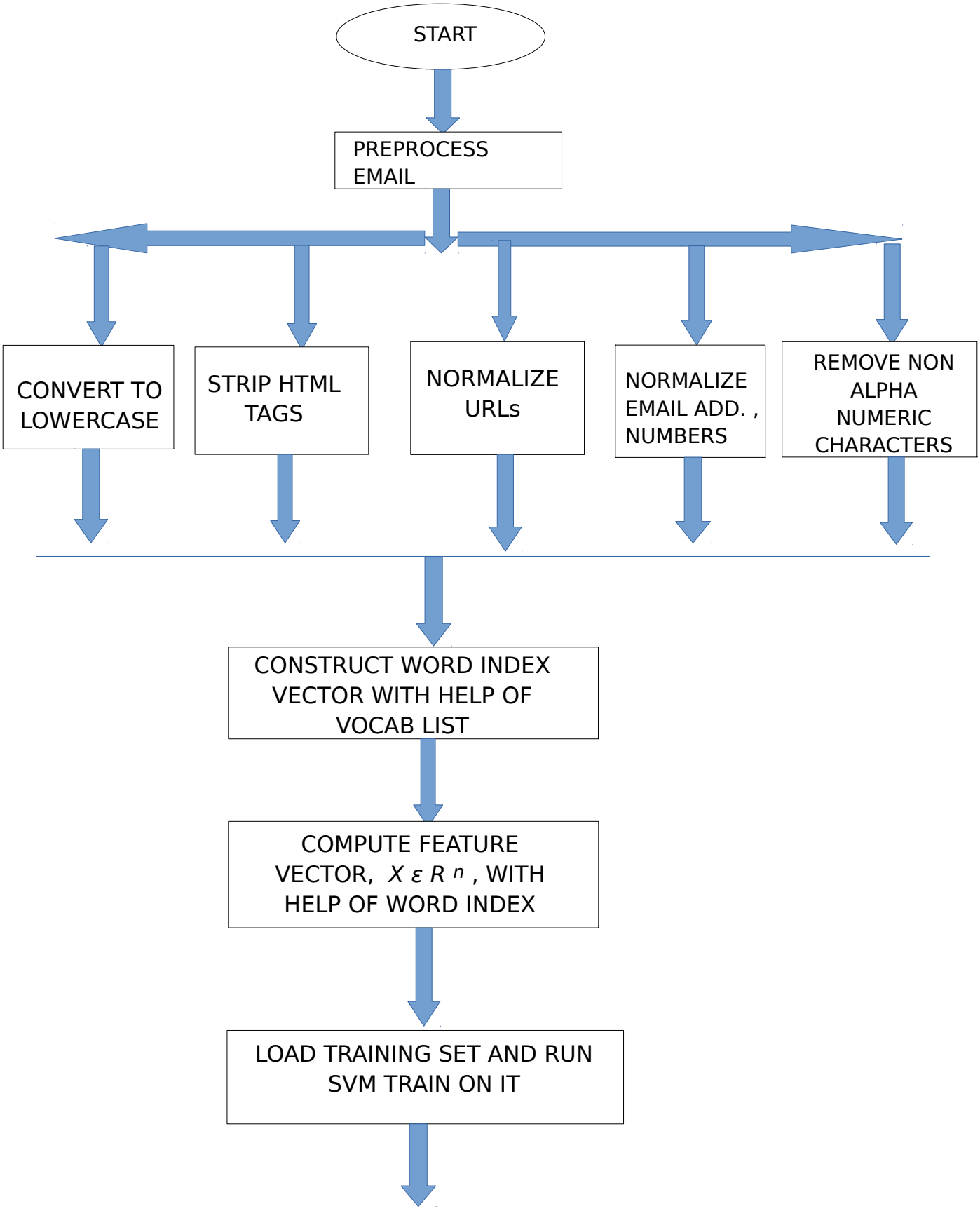
$x^{(i)} \in R^{1899}$.

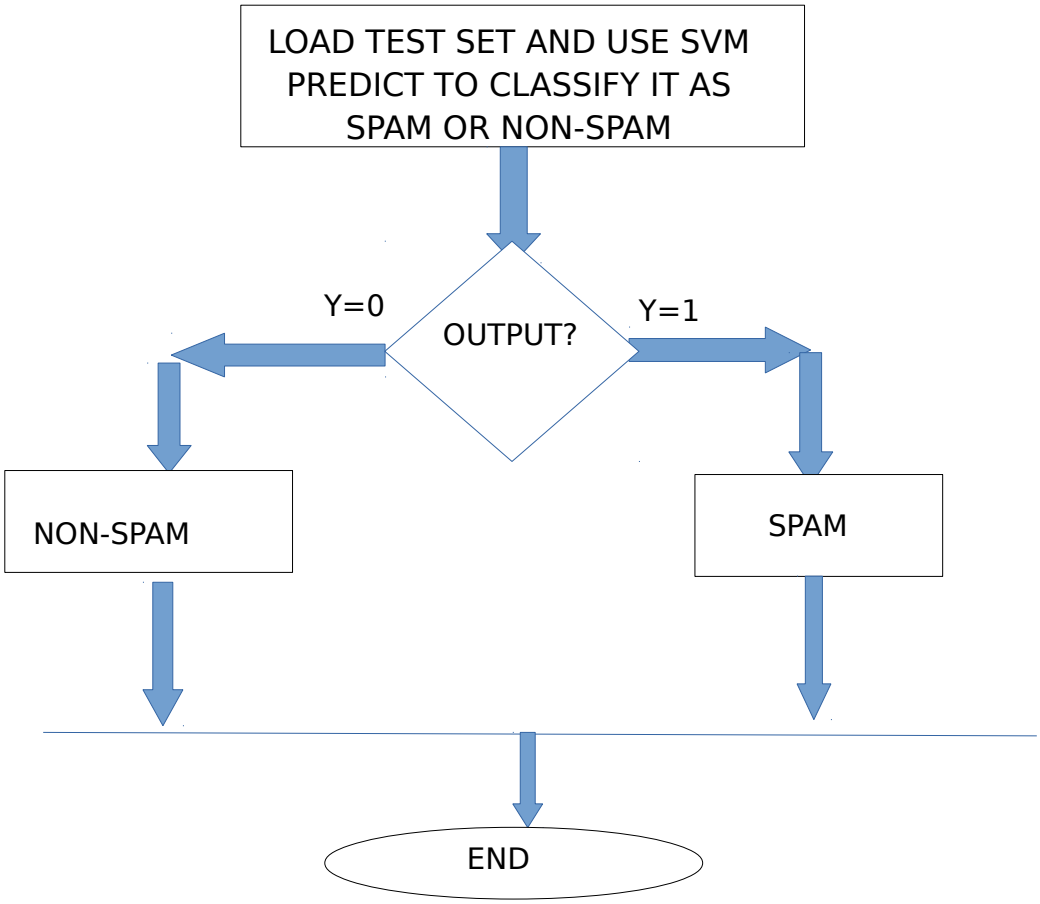
We run our SMO algorithm on our training set. Then we make observations on our test set. We calculate mean (accuracy) for both.

4.5 RESULT

Training accuracy/mean= 99.87%
Test accuracy/ mean = 98.5%

4.6 FLOWCHART




LOAD TEST SET AND USE SVM PREDICT TO CLASSIFY IT AS SPAM OR NON-SPAM
OUTPUT?
Y=0
Y=1
NON-SPAM
SPAM
END

5.CONCLUSION AND FUTURE SCOPE

The main purpose of this study was to show how can we construct a simple,yet efficient spm classifier using the supervised algorithm in Machine Learning- Support Vector Machine , which can classify a given email as spam (y=1) or non-spam (y=0) .

We have used a small dataset and a vocabulary list of 1899 words. In practice the vocabulary list is of atleast 4000-5000 words.
We have obtained an accuracy of 98.5%, which means 98 out of 100 emails can be correctly classified. This accuracy can be further improved using better SVM training algorithms and software packages.

In general, no algorithm can be implemented to obtain 100% accuracy. Due to small spam corpus, there is unknown word problem and many spam emails cannot be correctly classified. In future, a bigger spam corpus

can be used.

Also, in future we can use a combination of spam filtering algorithms like the Bayesian spam filter and the SVM filter, which can further increase the accuracy.

6.REFERENCES

1.http://spamassassin.apache.org/publiccorpus/

2.https://class.coursera.org/ml-006

3.http://www1.coe.neu.edu/~rwhelan/Pam/Uluski/algorithms_avspam.pdf

4.http://airccse.org/journal/jcsit/0211ijcsit12.pdf

5. en.wikipedia.org/wiki/Machine_learning