Email Spam Filtering

By Ananda Kishore Sirivella

UFID: 9951-5080, EEL 6825 Spring 2016

University of Florida, Gainesville, FL

Introduction

- Email a common means to communications
- Every new development has its positives and its negative
- Emails help in faster & cheaper communication
- The mis-use of email turns into spam.

Spam

An unwanted message received by the recipient is termed as SPAM. Types of spam mails:-

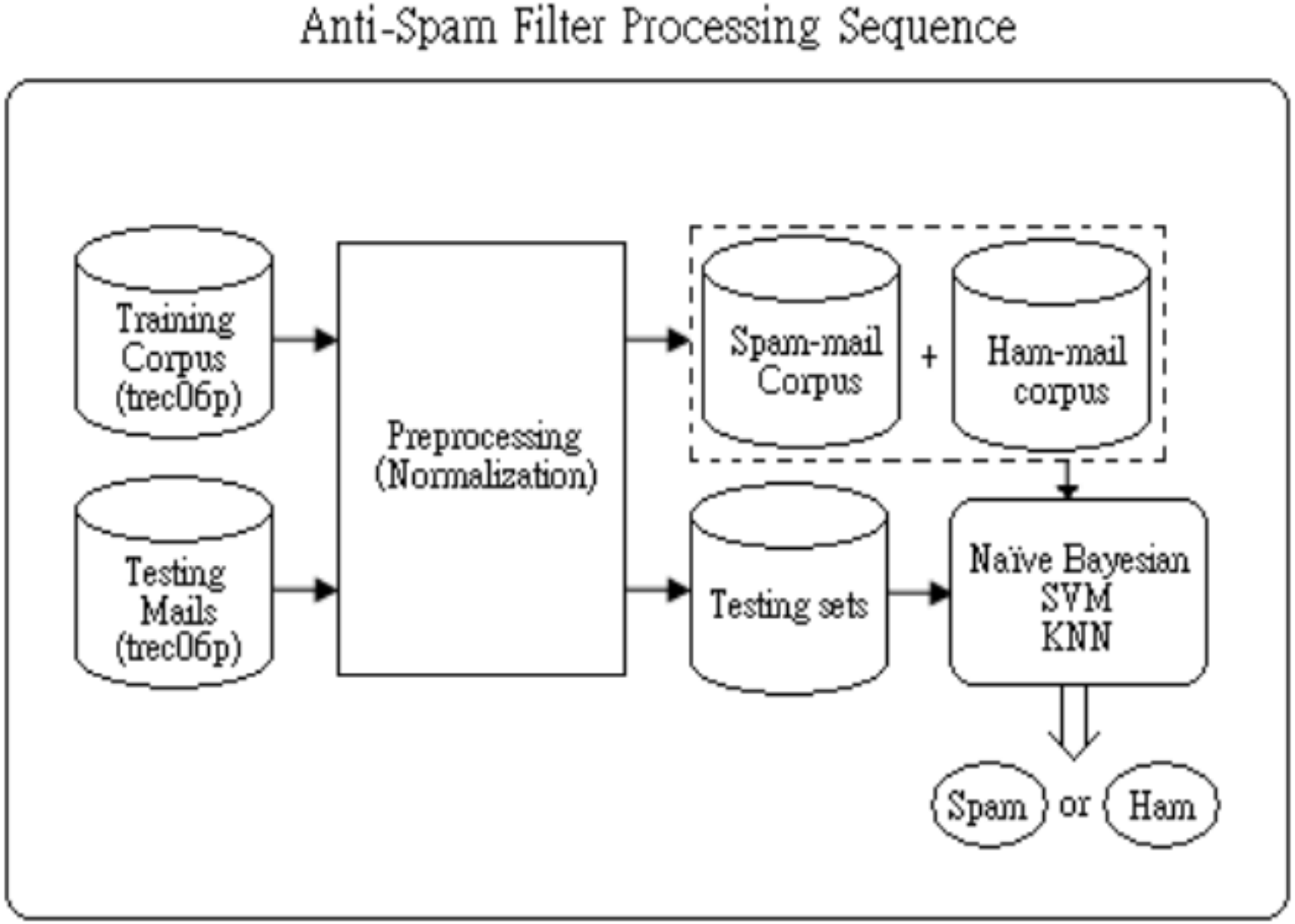
- Commercial Advertisements
- Unsolicited Advertisements
- Phishing scams
- Nigerian 419 scams
- Trojan Horse Email
- Political or Terrorist Spam
- Email spoofing
- Adult content Mails

Proposed Solution

Machine Learning algorithms, algorithms that provide computer the ability to learn without being programmed specifically. Machine learning algorithms build a model of black box which takes a input and output the processed data. In our case its classification of spam and ham.

- Supervised learning: Building a model over a pre-defined labeled training data set and using the model over other data set.
  - Naïve Bayes
  - K-Nearest neighbors
  - SVM
- Un-Supervised learning: Building a model on the real unlabeled time data rather than a training set or test set. Tends to be in accurate at times.

Proposed solution



UCI spambase

- Continuous Spam base
- Had wide varieties of instances
- Each column attributed to different words
- Had the count to spam words frequency.

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Integer, Real
Associated Tasks:	Classification
Number of Instances:	4601
Number of Attributes:	57
Missing Values?	Yes
Area:	Computer
Date Donated	1999-07-01
Number of Web Hits:	173555

Naïve Bayes Classifier

- Probabilistic approach to the problem

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

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

Naïve Bayes

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 Neighbor

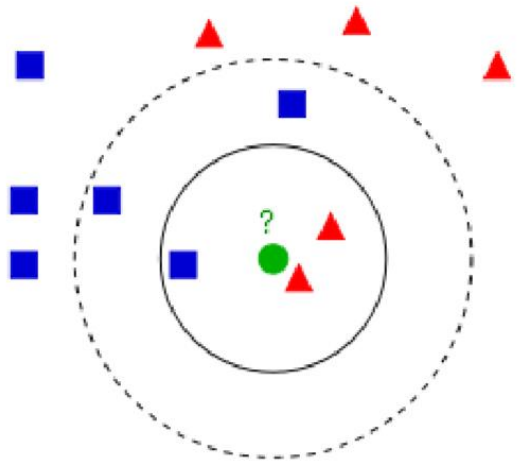
- Stored the label training set and associates the new input to poll of corresponding K neighbors
- In fig,
  - For K = 3,

    Consider the black bold line circle

    & conclude with red triangle's class (2/3)
  - For K = 5,

    Consider the dotted line circle

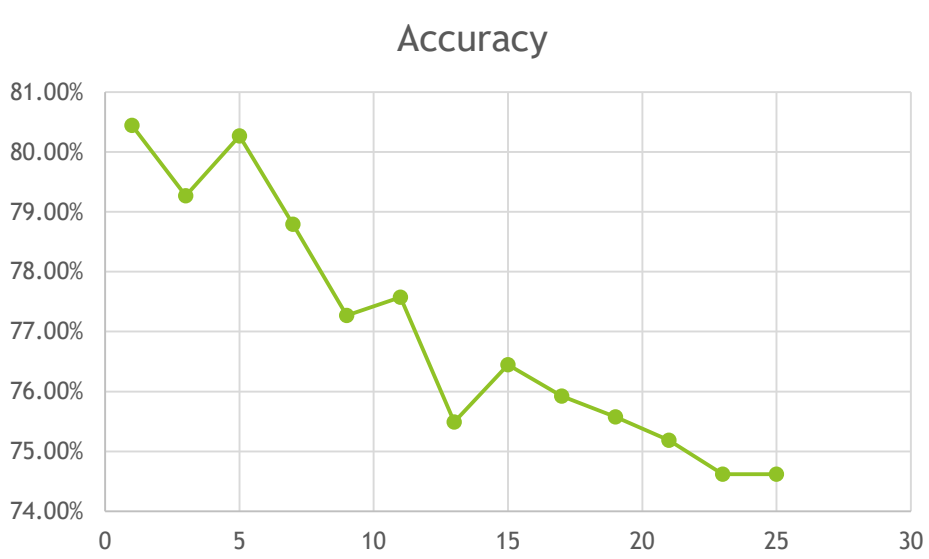
    & conclude with Blue square's class (3/5)

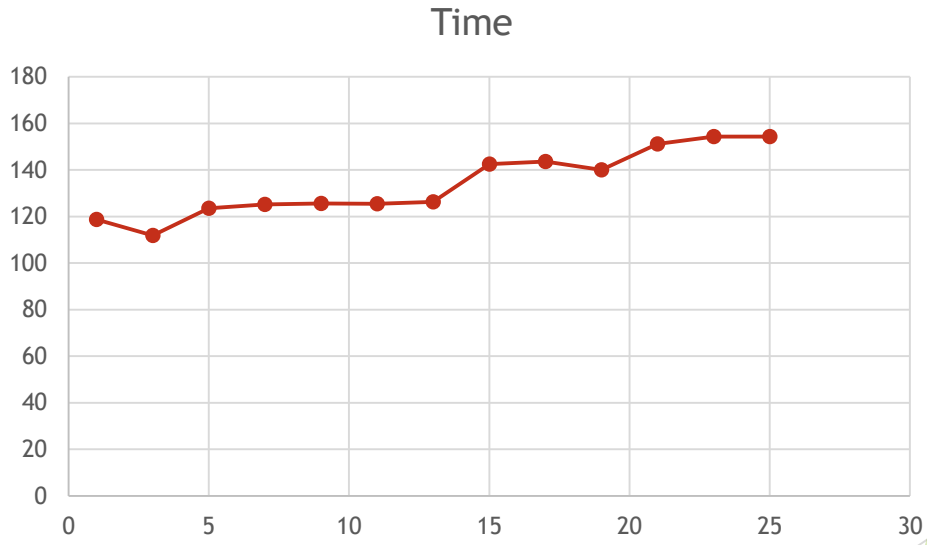


K-Nearest Neighbor illustration

K-Nearest Neighbor Run statistics

- Output with 50% training set and different values ranging from 1-25

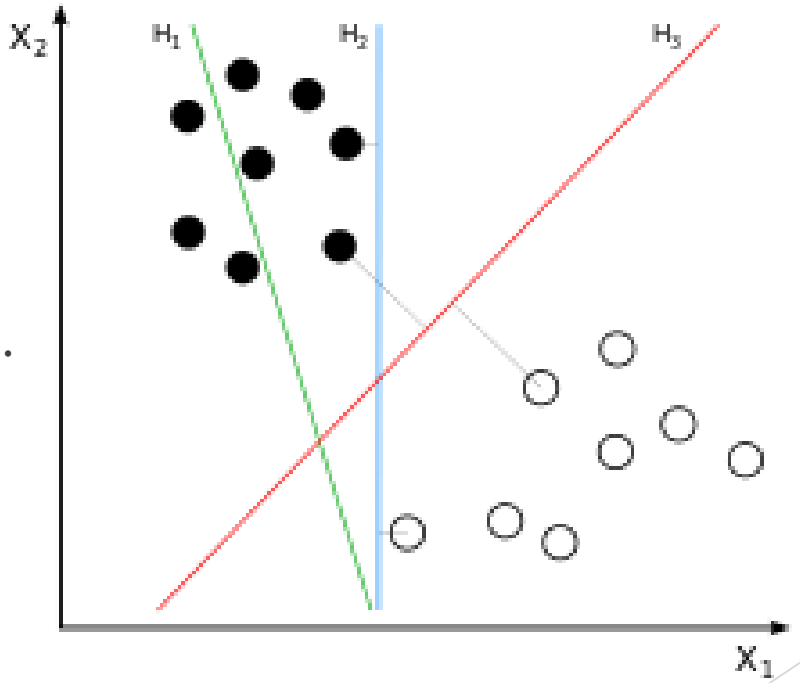




Support Vector Machine

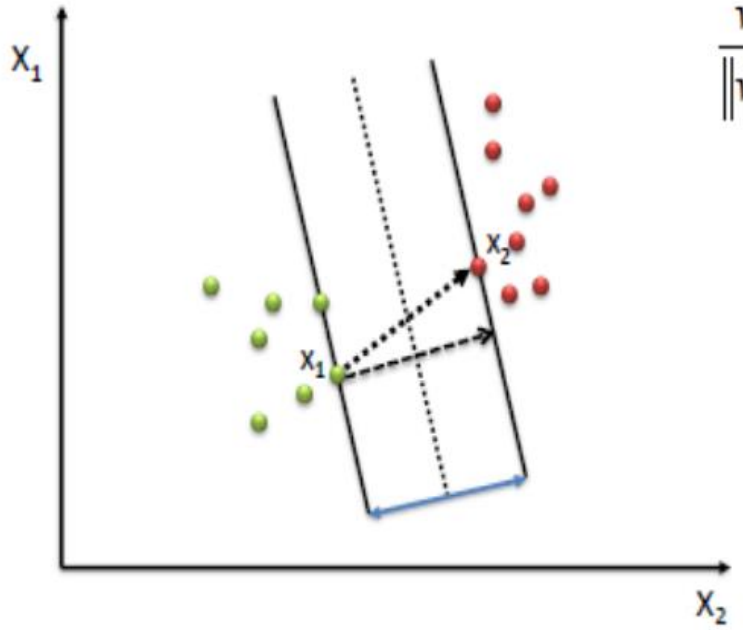
- SVM maps the data set as a point in vector space and maps a differentiating line or a plane between the two categories data.

- H_1 does not separate the classes.
- H_2 does, but only with a small margin.
- H_3 separates them with the maximum margin.



Support Vector Machine

- Draw margins to the classifier and try maximizing the margin.



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

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

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

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

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

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

Support Vector Machine run statistics

```
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>
```

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

- 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

Reference

- en.wikipedia.org
- scikit-learn.org/stable/
- paulgraham.com/spam.html
- machinelearningmastery.com/
- archive.ics.uci.edu/ml/datasets/Spambase