# Machine Learning for Intelligent Systems

SEM6420

Assignment One

Daniel McGuckin

[dam44@aber.ac.uk](mailto:dam44@aber.ac.uk)

Paper: “F Benevenuto, Gl Magno, T Rodrigues, and V Almeida. Detecting Spammers on Twitter. In Proceedings of the Seventh annual Collaboration, Electronic messaging, AntiAbuse and Spam Conference (CEAS 2010), Redmond, Washington, US, 2010”

The paper details an attempt to train a non-linear Support Vector Machine (SVM) using a Radial Basis Function Kernel. They train the algorithm to classify spam versus legitimate data, firstly concerning Twitter user accounts and then individual tweets.

# The Problems

The paper identifies that Twitter is a target for spam due to the number of users who use the service and how they communicate with each other. Elements of the service like hashtags can be used by spammers to try to fool users into believing they are posting about relevant, popular topics when they are really trying to get users to buy products or even leading them to malicious or inappropriate websites. This is aided by their ability to obfuscate the URL’s in their tweets.

To stop this spam it must first be found, this is the problem that the paper attempts to solve. The authors declare that a supervised machine learning algorithm could be used to find either the spammer accounts themselves or the spam that they tweet.

# The Solutions

The authors first used data crawling software to gather information about users. Over 58 servers white-listed for them by Twitter they found almost 55 million users and 1.8 billion tweets. They got volunteers to label 8,207 of these Twitter accounts of which 355 were spam. They then randomly selected 710 legitimate accounts to even out the numbers. This resulted in 1,065 accounts used in their training process. The paper does not detail how many tweets were used in their experiments.

Next they decided on 62 attributes describing behaviour differentiating spam from legitimate accounts and tweets. These attributes fit into two different types:

* Content – attributes detailing information about user tweets such as the number of URL’s per words.
* User behaviour – attributes detailing information about the user account such as the number of followers.

Their experiments first looked into user account classification, this was their primary focus in the paper. A 5-fold cross-validation strategy was used, they split their data into 5 samples, 4 of these samples were used to train the SVM and the other one used to test it. They performed this training and testing 5 times, each time rotating their samples so that a different one was used to test. They then repeated this entire process 5 times resulting in 25 separate trails.

They declared that on average “Approximately, 70% of spammers and 96% of non-spammers were correctly classified.” The authors state that this algorithm can effectively identify spammer accounts, thus achieving to some degree the problem this paper was attempting to solve. However, they admit that although their SVM only misclassifies a small percentage of legitimate accounts, this “could not be suitable for detection mechanisms that apply some sort of automatic punishment”.

In an attempt to rectify this they used a cost-mechanism to give priority to one class over the other. They managed to tune their cost-mechanism so that they would only misclassify 0.3% of legitimate accounts, this was a 3.3% increase from 96.4%. However, this was at the cost of a reduction in spam classification, down from 70.1% to 43.7%, a reduction of 26.4%.

The paper then discussed attempting to reduce the number of attributes used. They attempted to find which attributes were most useful by applying the information gain and Chi Squared feature selection methods. Their findings showed that using either the top 10 or 20 attributes gave results of a similar quality as using all 62 attributes. They also show that content and user behaviour attributes have a similar weighting and as content attributes are much easier to disguise by the spammers, behaviour attributes could allow intelligent spammers to be detected more easily.

The second problem explored by the paper - classification of spam tweets - was then briefly delved into. They used a selection of the content attributes in the SVM. Their results showed that on average their SVM could detect around 79% of spam tweets and 93% of legitimate ones. Although the SVM performed better on detecting spam tweets than it did spam accounts, it performed almost twice as badly when classifying legitimate ones. The authors did not explore using their cost-mechanism for this problem but one could imagine that it would produce similar results to what was achieved in the first problem.

In summary, the paper explores the problem of detecting Twitter spammers. It divides this problem into the detection of user accounts and user tweets. A non-linear SVM is trained on labelled data and then tested. The SVM is proven to be able to classify 70% of spam and 96% of legitimate users with similar results for tweets, a valid attempt was therefore made to solve the problem at hand but it is arguable whether they have actually ‘solved’ the problem due to their error margin classifying legitimate accounts. They did managed to reduce this error percentage but with considerable loss to their ability to classify spam.

# The Issues

A big issue with using machine learning solutions for detecting spam is that these systems have a certain degree of error. Whilst classifying some spam as legitimate data is generally not a big deal, classifying legitimate data as spam is. For instance if an automatic punishment system is imposed – which ultimately will always be the aim, to allow automation to handle all aspects of dealing with spam – then it is extremely bad to punish legitimate users. This creates negativity towards the website and destroys part of the user base. It also defeats the point of using automated punishment as staff will either be required to check every punished account to ensure they are not legitimate users or deal with many complaints. A 4% error when classifying users as spam does not seem high but over 55 million users this would add up to a huge amount of misclassification.

Another issue is that SVM’s can overfit the data when using a non-linear kernel to map to a potentially infinite dimensional space. When using an RBF kernel, how much the data is fit to the training data is reliant on the ϒ parameter within the kernel equation, setting it too high can cause extreme overfitting[1]. How this parameter is set is up to the developer to more or less guess. The paper does not discuss what they set this parameter to and they do not mention overfitting. Whether this means that they did not have overfitting problems or were simply avoiding the discussion is hard to say. Ultimately their results when using their SVM on their testing data seemed good, but possibly a look into how much their algorithm was overfitting on the training data could have further optimised their results.

# The Alternative Approach

My alternative approach to the problem is to use the Naïve Bayes supervised classification method with the non-linear RBF SVM used in the paper. I will first discuss Naïve Bayes and then how the two could be used together to try to reduce the misclassification rate of legitimate data whilst maintaining an acceptable spam classification rate. Using the two together is therefore my proposed solution to the primary issue the researchers had in their experiments.

## How Naïve Bayes Works [2], [3]

Naïve Bayes uses conditional probability to classify new data based on prior evidence. Conditional probability is ‘the probability that something will happen, given that something else has already happened.’[2] Conditional probability can be used to find the statistical probability of an outcome based on prior evidence. When using this for classification the prior evidence is data concerning the training set and the outcome is the class we wish to assign our new data to.

To do this the Bayes rule can be used and adapted. The Bayes rule is stated mathematically as follows:

To explain this I will discuss how it could be applied to the Twitter spam user account classification problem.

We want to know whether to class our new data as spam or legitimate -These classes are represented by A in the equation- based on our prior knowledge B: our training set. Using the Bayes theorem we can compute, for instance, if an account is spam based on… one feature in our training set.

Let’s use an example: the feature could be the probability that a user will use a URL in their tweet. We want to know whether, based on previous users URL rate, this implies that the new user is a spam account.

We must first calculate the ‘prior’ probabilities, this is the likelihood that a user will be spam or legitimate based on the numbers of these classes in the training data, regardless of anything else. If there are 100 users in the training set and 40 of them are spam then the prior probability of a user being a spammer is 0.4.

Most features concerning the Twitter accounts are continuous but we need to calculate them as discrete variables. We can either use ‘Discretization’ or ‘Distribution modeling’[4]. I will focus on using Discretization, this is to group sections of the data. For instance we can compute how likely an account is to be spam given that its URL’s per tweet are > 0.5, this would be one of our groups. We can now count all the users in our training set that have a URL rate of > 0.5. If we say that 0.2 of our users conform to this then:

Finally we need to work out the probability that a user has a URL rate of > 0.5 given that they are a spam account. We already know that 40 of our 100 users are spam accounts and we can count up which of those spam accounts has this conforming URL rate. Assuming that the URL rate distribution is biased towards spam accounts having an increased number of users with > 0.5 URL rate. Then let’s say we have 12 of these users.

The Bayes theorem therefore tells us that there is a 0.6 likelihood that the new user is a spam account given that their URL rate is > 0.5:

If we repeat this process for each class. In our case repeat it for whether an account is legitimate given the URL rate > 0.5 we can compare the resulting probabilities and classify the new account based on which has the greatest probability.

This is… somewhat lacking in practicality. In reality we rarely deal with one-dimensional data, our Twitter users have lots of features that can help us determine whether or not they are spammers and we’d like to use all these features. A single feature is rarely going to give us reliable results.

Enter Naïve Bayes. We treat each feature (or evidence) as independent of all other features. Each feature is ignorant of the other features, hence ‘Naïve’ Bayes. The standard Bayes formula is changed to incorporate n features, rather than 1 feature:

We don’t care about the actual probabilities as figures because we are purely interested in the classification. As everything will be divided by the same value we can disregard the denominator in the equation as the resulting values will be equally reliable when comparing for classification as the denominators only function is to scale the outcomes. This simplifies the equation and on a computer would decrease calculation time.

Adding in another feature C. If we compute that C has a 0.2 probability of occurring and 0.6 probability of occurring given that it is spam. If we couple this with our previous URL rate > 0.5 feature B, using multi-dimensional data with Naïve Bayes we can calculate the likelihood (unscaled by evidence) of the user being a spammer as follows:

If we go through a similar process to find the likelihood of the user being legitimate we can compare the values to find which is more likely and thus classify it. If the legitimate user equation results in 0.063 then we can classify this user as spam because 0.072 > 0.063.

## Combining SVM with Naïve Bayes

# The Limitations and Future Development

# Bibliography

[1] -<http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html>

[2] - <http://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification>

[3] - <http://ats.cs.ut.ee/u/kt/hw/spam/spam.pdf>

[4] - <https://www.quora.com/What-is-the-best-way-to-use-continuous-variables-for-a-naive-bayes-classifier-Do-we-need-to-cluster-them-or-leave-for-self-learning-Pls-help>