28Machine Learning Methods for Spam e-mail Classification

CARL MONTECLARO, ASHLEY ODDEN, DYLAN WAHL
Department of Computer Science, University of Lynchburg, Lynchburg, Virginia, U.S.A.

Abstract:

This is a rather brief article on the importance of addressing growing data problems and how the spam filter was instrumental in how we endeavor to handle future complications. This article was written and documented by Carl Monteclaro with Ashley Odden and Dylan Wahl serving as the primary developers of this software. Our goal is to utilize the knowledge we have gathered over the course of the last semester in order to implement a new and innovative approach to how spam filters can and should be addressed.

Introduction:

In the age of data, it's very often that one takes for granted the security mechanisms and tools given to us by the developers of the software and firmware we use today. Take for example the wonderful filters in our email inboxes. They are there to both streamline and organize the many conversations, updates and promotions brought to us from across the web. With that said, amongst these filters there is a champion that protects our inbox: the spam filter. For those unaware, the way a lot of companies and websites try to advertise is through your viewership online of certain ads, websites or videos you happen to click on. This can often result in what is referred to as the butterfly effect. Your data could and probably still can be inadvertently shared with multiple advertisers and scam artists around the web just from a single click. Then your inbox would receive a massive influx of useless email promotions or spam and would make things rather hard to organize and sort through what is needed and what isn’t.

As it stands, spam detection is often harder and harder to account for as data gets bigger and bigger in our generation. It’s very much akin to a tug-of-war where developers at Google or Bing wish to do the utmost to protect and serve you while many start-ups, large corporations or shady businesses seek to find ways to work around. The more security measures are built-in, the harder the other side of the spectrum pulls. How it was originally solved was (insert previous method here, still looking for a good relevant article I can extrapolate from). How we (insert start-up company name here) propose to solve this ever-evolving problem is through the use of Bayesian Classification and trusty Neural Networks. Bayesian Classification is a theorem of probability in which there are two types of said probability, the Posterior Probability and the Prior Probability. Provided below is the theorem transposed into an equation. *h* represents a given hypothesis and **D** is the provided data tuple in relation to the hypothesis. What Bayesian Classification does is it assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Why this is important is that it ensures that data that might make a spam email seem legitimate in nature to the preferences of what the user seeks in actuality cannot be relevant, stopping sneaky loopholes and virus-ridden links from plaguing our users.

$$P\,(D) = \frac{P(h)P(h)}{P(D)}$$

$$P(B) = \Sigma P\,(A_i)\,P(A_i)$$

$$P\left(x_{i,}x_2\ldots,x_n|c_i\right) = \Pi P(x_i \mid c_i\;)$$

Alongside this powerful implementation is Neural Networks, which are modeled after information transfer in the most powerful computer in the world: the human brain. Though science and study are not new, the many ways we are able to tackle and utilize them have only increased as time has gone on. How they are relevant can be summated in a single sentence: understand and think like the person you want to be in order to grow. In simpler terms, Neural Networks are made to simulate human decision-making and understanding. Humans interpret information through our five senses and those can be referred to as layers of information. Having a computer emulate that same operation is hard but is in fact doable with the right techniques, which will be divulged in full later on.

Our Program:

Our system is divided up into three parts: Pre-processing of our data (in this case, a csv file filled with viable data and spam data), utilizing Bayesian classification to ensure the integrity of our spam filter so that elusive spam cannot sneak through with some positive data values alone and Neural Network Classification. We have implemented two Neural Network classifications that will be covered further below.

The way pre-processing went as follows. We identify stop words, stem the leftover words, take a word count of the entirety of our data and rank them in accordance with frequency, then identify the 50 most frequent words in our data set and from this set of frequencies we point out outliers. This is done to prepare the data for transfer into our next two steps. Dylan Wahl, Ashley Odden and Carl Monteclaro all contributed to the pre-processing step.

Bayesian Classification was handled by Dylan Wahl. This part of the project involved splitting up the given data into the correctly proportioned training and testing sets. The program then uses the training set to create a dictionary for bot unique ham and spam words with unique words as the keys, and the number of occurrences of the word as the value. To evaluate whether a sentence in the testing set was ham or spam, the tokens or words were compared to the aforementioned dictionaries. For every word they had in common, a similarity score was incremented. After the entire sentence had been analyzed, the similarity scores were compared and the larger score determined the prediction. This method, while simple, was fairly accurate with an average 87% success rate. To improve the accuracy of this classifier, it would be a good idea to calculate the individual probabilities that a sentence belongs in each category based on each word, and adding up those probabilities. This second method might yield better results because it would take into account the relative weights of each word in the training sets.

Neural Networking

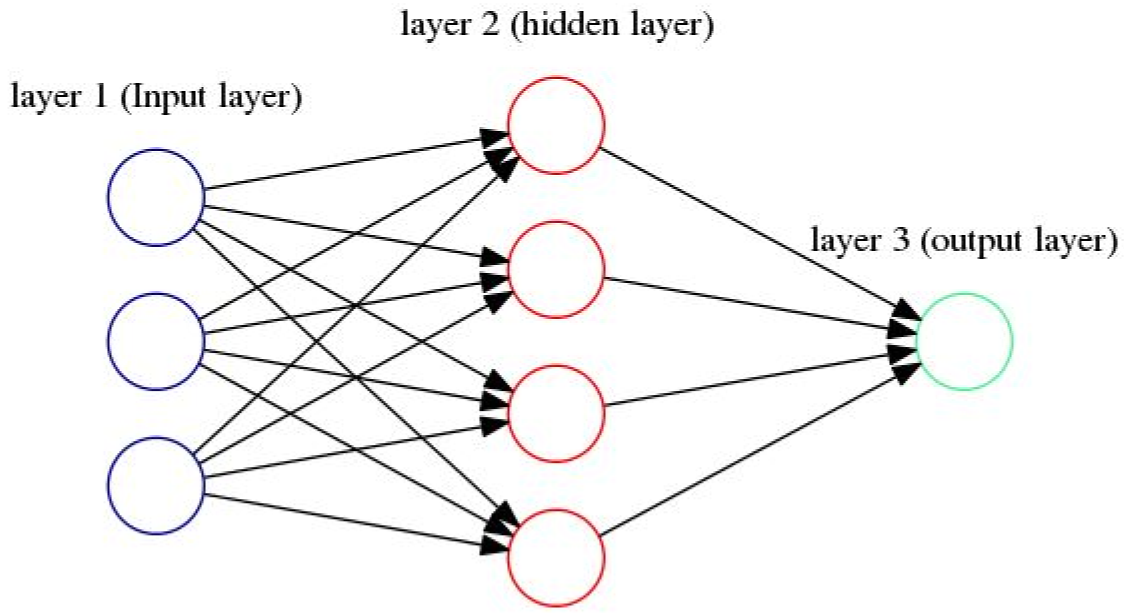
Our neural network was handled by Ashley Odden.An artificial neural network can be considered as an assembly of simple processing nodes/neurons. The processing ability of our network is stored in an interconnected network of such neurons. The interconnection strengths are known as weights. These weights are obtained through learning from an input layer, an output layer, and hidden layers. Each layer has different units. The units themselves are outputs of a weighted sum of their inputs. For this function we are trying to teach a neural network how to recognize spam email messages from ham (non spam) email messages. For this process we need to train the network by finding the most occured words in both the spam and the ham. From the most common in each we have a reference document that can help train the neural network. With these two reference documents we can compare each spam or ham message to each of their classes' most commonly used words.

The first step requires us to reduce morphological variation and to remove any arbitrary words, known as stopwords. What this does is allow for our program to understand when words denote the same meaning. For example cat and cats still refer to the object cat, just multiple instances.

The computer otherwise would not know the difference between the two, so we must split the string of words to just their words to then go to the next step.

The second step requires us to utilize a process known as bag-of-words method(BOW). This converts our string data in our email messages to numeric values. This model is a way of representing arbitrary text into fixed-length vectors, by first counting the occurences of the sampled messages against their class common words (for our case 50 most common words in ham and spam). Secondly, we vectorize the documents. Doing so is through counting the occurences of the words found in our top 50 words. If the word is not found we return a 0. If the word is represented in our most common word set, we set the value to 1.

The third step is training the model now that we have our vectors to have our network discover the weighted values that denote the connections and meaning. If the word is found in the top 50 words, this weighted value describes meaning to our function. Through processing the weighted values, through linear regression and our rectified linear unit, we have a model that we can assign to our train data set to relate our predicted outcome and use criterion through getting the mean squared error to get an estimation of the unobserved measures of the average of the differences between our estimated values and the virtual value that we are trying to predict, which is an output layer of 1.



(Used from Medium.com)

This is our criterion that we use to compute our loss function to perform gradient descent. You could picture gradient descent as a flowing river down a mountain. You naturally follow a gradient downward, the loss function acts with the gradient and finds the lowest means and this is what we use to perform our back propagation then update our parameters. Through this, our network uses methods of recursion in order to train itself in a fine working loop.

Conclusion:

This project proved to be a considerable challenge for our group. At this time in the semester, we were all spread thin with large workloads and diverse personal priorities. Initially, we had considerable trouble syncing our work together as we were attempting to use online compilers with incredibly unreliable internet access. We solved the collaboration problem when we switched to using github with git bash for windows. Of course, this had its own learning curve, but it worked better than collaborating over shared documents online. Once we solved this problem, we began to tackle issues related to our differing levels of familiarity and availability, eventually creating a revised workload distribution as listed below. Another struggle we had as a group was different styles of programming and different preferences. Eventually we solved these issues to the best of our ability to create the mostly-working spam detection program employing both the use of a bayesian classifier and neural networks.

Work breakdown:
Our roles in this project were very fluid as we were accounting for each other's schedule, availability, and familiarity with different parts of the project. below is an itemized breakdown of the work assigned to each member of the group.

- Data preprocessing
    - Stop words: Ashley, completed
    - Word Stemming: Ashley, completed
    - Word Count: Dylan, completed
    - Frequent Word Identification: Ashley and Dylan, completed
    - Outlier Identification: CJ, not completed
- Bayesian classification: Dylan, completed
- Neural Network classification: Ashley, completed
- Evaluation: Dylan, Ashley; completed
- Report
    - Introduction: CJ, Ashley; completed
    - Our System: Dylan and Ashley, completed
    - Conclusion: Dylan, Ashley; not completed
- Oral Presentation: everyone, not completed as of 5/1/2020