Spam or Not Spam
Using an Artificial Neural Network

Michael Yachanin

December 13, 2015

Abstract

Artificial neural networks (ANNs) are a set of models used in machine learning. They were contrived from human interest in biological neural networks. When trained, ANNs can be used to approximate the values of functions that depend on a large number of inputs. The classification of spam email is a relevant problem in today's society that can be reasonably approximated with a sufficient number of inputs.

1 Introduction

This project uses the Spambase Data Set [2]. The data was acquired from a postmaster at University of California, Irvine (UCI) and a donation from George Forman. There are 4601 datapoints with 57 attributes total, including the classification. It is worth noting that some of the attributes are not generic, such as the frequency of the word "George". It makes intuitive sense that this is a good attribute to have. However, that means these results are not representative of a generic email spam detector. This project evaluates the performance of an ANN against the results found in the study: "Email: Spam or Not Spam (CLASSIFICATION PROBLEM)" [1].

2 Pre-Processing

2.1 Feature Selection & Extraction

To select the most relevant features, the data is loaded into Weka, and a simple best-first search is used. Doing so narrowed the data from 56 attributes to 15, not including the classification. This was exported to a CSV file for use.

2.2 Randomization

The vanilla dataset has non-spam and spam emails clustered together in the CSV file. Therefore, it is necessary to randomize the data to ensure the testing data is representative of the

entire dataset. To do this gshuf is used: $gshuf spambase.data > spambaseRandom.data$. gshuf mixes lines in a file and redirects them to standard output. That is then redirected to a file for future use.

3 Design

To start, 5% of the dataset will be set aside for validation. The other 95% will be used for training. During the training phase, the training data is segmented into 10 folds. 10 ANNs are created, each with identical initial states. Each ANN receives one fold to use as testing data and the other 9 as training data. Then, the ANNs are trained in iterations via backpropogation [3]. Each iteration passes all of the training data through the ANN and backpropogates the errors back. Then, at the end of each iteration, the ANN is tested against the testing data to obtain a testing error. This process is repeated until the network exhibits a testing error of an arbitrary value. Over many trials, 4% error seemed to work best. However, if the network does not reach 4% error within a certain number of iterations, the training is short circuited and the results are discarded. It seems as though the initial weights play a large part in obtaining a well-trained ANN. If 4% error is not reached within, say 20 iterations, it is likely that the initial weights are poor choices. It will be much faster to move on. If any ANN reaches the required error threshold, the network is tested against the set-aside validation data to obtain its final error. In order to retain my sanity, this process is put in a $while(true)$ loop; The best ANNs are automatically saved for future consideration.

Note: In order to further my understanding of ANNs, I decided to code an ANN from scratch. The ANN is designed to be as modular as possible. However, currently only the use of 1 hidden layer is supported. It is possible that using more than 1 hidden layer could provide better results than those found below.

4 Results

The study "Email: Spam or Not Spam (CLASSIFICATION PROBLEM)" uses two different algorithms to classify spam. The first is Recursive Partitioning and Regression Trees. The model shows an accuracy of 89.5% and a confusion matrix:

	Spam	Not Spam
Spam	876	51
Not Spam	110	497

The second algorithm is Support Vector Machine. It obtained an accuracy of 92.9% and a confusion matrix:

	Spam	Not Spam
Spam	537	70
Not Spam	39	888

The artificial neural networks created exhibit errors of about 90%. The best result has an accuracy of 91.3% and a confusion matrix:

	Spam	Not Spam
Spam	88	7
Not Spam	13	122

5 Conclusion

An artificial neural network can be created to classify email spam with a relatively low error-rate. Possible improvements include using more than one hidden layer and lowering the learning rate from 10%. Furthermore, using non-generic inputs is beneficial. For instance, if a sender knows the receiver's name, the email is less likely to be spam. Therefore, a good approach may be to build a model for each individual instead of one generic model.

References

[1] "Hidalgo, J. M. G., and Garcia, G. S. Email: Spam or not spam (classification problem), may 2013. `http://rstudio-pubs-static.s3.amazonaws.com/6288_1e178feb2c144ec5961a18a288330e12.html`.

[2] "Hopkins, M., Reeber, E., Forman, G., and Suermondt, J. Spambase data set, july 1999. `http://archive.ics.uci.edu/ml/datasets/Spambase`.

[3] Russell, S. *Artificial intelligence : a modern approach*. Prentice Hall, Upper Saddle River, NJ, 2010.