AN EMPIRICAL EVALUATION OF ADABOOST IN NEAT AND rtNEAT

Robert Schukei[1], **and Dr. Blayne E. Mayfield**[2]
[1]Computer Science Department, Baker University, PO BOX 65, Baldwin City, KS, 66006
[2]Computer Science Department, Oklahoma State University, 219 MSCS, Stillwater, OK, 74078

Abstract - *While the level of spam (unsolicited, unwanted emails) has dropped over the past couple of years, it still accounts for more than 60% of email traffic.[1] Not only is this an inconvenience to email users, but in areas of the world with limited Internet bandwidth, spam can choke much of that capacity. The research described in this paper attempts to decrease the runtime of spam filter training by employing the machine learning techniques of Adaptive Boosting (AdaBoost) in conjunction with NEAT (NeuroEvolution of Augmenting Technologies) or rtNEAT (real-time NEAT).*

Keywords: NEAT, rtNEAT, AdaBoost, Neuroevolution, Spam Filtering

1 Introduction

While the level of spam (unsolicited, unwanted emails) has dropped over the past couple of years, it still accounts for more than 60% of email traffic.[1] Not only is this an inconvenience to email users, but in areas of the world with limited Internet bandwidth, spam can choke much of that capacity. Further, spam often is used as a vehicle for attempting to introduce malware onto computers that receive it.

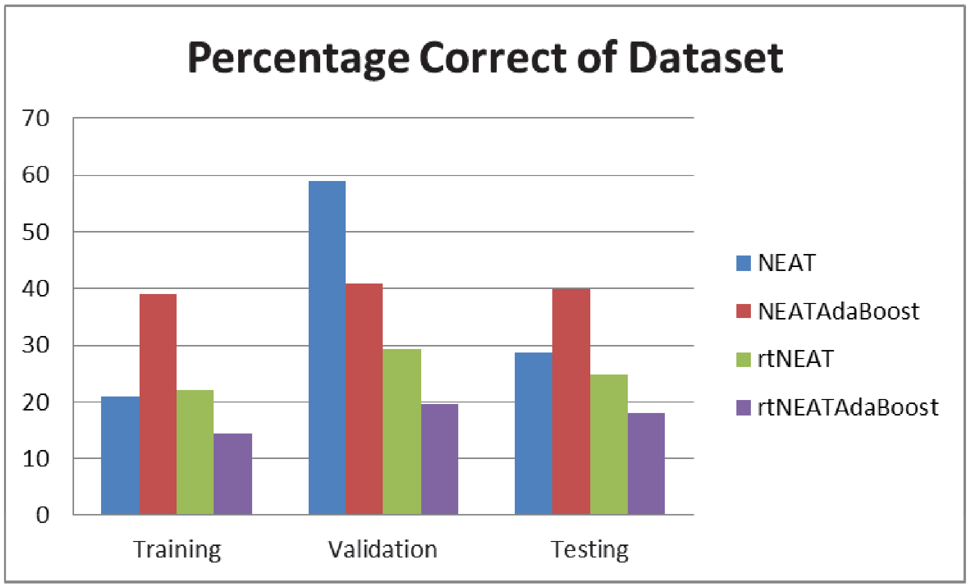
Most Internet Service Providers (ISPs), as well as email server and client software, deploy spam filters in the battle to prevent spam from getting to the end user. These spam filters require time-consuming training and retraining to do their jobs. The research described in this paper attempts to decrease the runtime of spam filter training by employing the machine learning techniques of Adaptive Boosting (AdaBoost) in conjunction with NEAT (NeuroEvolution of Augmenting Technologies) or rtNEAT (real-time NEAT). While it is understood that spam filters utilizing these techniques might not be as effective as current filters, the experiment will indicate which technique might be the best candidate for further investigation. The training will be using the UCI SPAM dataset.[2]

2 Background

NEAT is a neuroevolution technique that constructs and trains Artificial Neural Networks (ANNs) over a data set. It was developed by the Neural Networks Research Group at the University of Texas, and they state that, "neuroevolution is a method for optimizing neural network weights and topologies using evolutionary computation."[3] NEAT is an algorithm that evolves a neural network instead of having human specialists spend their time trying to work out a network topology that can solve a specific type of problem. In each generation of its run, NEAT examines and can change the weights and topologies of a network to learn better how the inputs and outputs are related to each other.[4, 5, 6] Unlike many evolutionary algorithms, NEAT uses speciation (evolution of species) to guarantee that favorable attributes in the population will persist for some number of generations, thus giving the species that exhibit those attributes a greater chance of passing them along to organisms in future generations.[4, 5, 6]

One variation of NEAT that has proven to be more effective in some cases is rtNEAT. Unlike NEAT, rtNEAT does not evolve the entire population in a single generation.[7] The result is that only small changes occur in each generation, decreasing the generation-to-generation runtime. Thus, rtNEAT is a better choice than NEAT in applications where evolution must take place in real-time. The algorithm tracks the age of each organisms and, when an organism reaches some specific age, its fitness is tested. If that fitness is among the worst in the population, then the organism dies; two, more fit organisms from the fittest species then mate to produce a replacement organism.

While NEAT and rtNeat is about evolving artificial neural networks an ensemble technique that has been used to improve performance of ANN is Adaptive Boosting. AdaBoost is a common ensemble methodology for machine learning; i.e., it is a secondary technique (sometimes referred to as a meta-algorithm) that can be used in conjunction with machine learning algorithms to improve their performance.[8, 9] AdaBoost has proven to be valid with decision trees, neural networks, and support vector machines.[10, 11, 8, 9] When used with decision trees, AdaBoost allows one or more weak hypotheses to be joined together on a decision to form a strong hypothesis. As long as the weak hypotheses are better than random, then AdaBoost had been proven to be an effective ensemble technique that can provide more accurate results. The version of AdaBoost used in this research is based on the AdaBoost.m2 algorithm, which trains each new hypothesis on the hard-to classify input.[8, 9] This allows easy-to-classify input to be handled quickly, thus permitting more time to be spent on the hard-to-classify input. One question posed in this research is whether AdaBoost can be used decrease the training time of NEAT or rtNEAT using the UCI SPAM data



3 The Experiment

The implementations of NEAT, and rtNEAT downloaded from the Neural Networks Research Group at the University of Texas was used to perform the experiments.[12, 13] The implementation of the AdaBoost algorithm was programmed based of the pseudocode provided in Boosting Neural Networks using method R of that paper.[8] Each of the four techniques (NEAT and rtNEAT, both with and without AdaBoost) was run numerous times, and data was collected. Each run was permitted to train for 100,000 organism evolutions. For NEAT with a population of 100, this equates to 1000 generations, since every organism evolves in every generation. In rtNEAT, this means 100,000 generations will pass, since rtNEAT evolves only one organism per generation. Using NEAT with AdaBoost, where we assume 10 weak hypotheses in AdaBoost, the NEAT algorithm has 100 generations for each hypothesis. While for rtNEAT with AdaBoost, again expecting 10 hypotheses, rtNEAT will have 10,000 generations. It is hoped that this will allow NEAT and rtNEAT enough time to figure out some of the basic connections for each of the hard hypotheses. For all experiments, training is done using parameters from two of the default parameter files provided as part of the NEAT and rtNEAT downloads from their respective websites. These files provide adequate starting parameters, with the understanding that they do not necessarily represent perfect choices for the parameter values. However, the assumption is that any inadequacies in the parameter values will affect NEAT, rtNEAT, and implementations with AdaBoost in the same ways, so the relative runtimes of the different techniques still can be used to determine which one (or ones) perform best.

As stated earlier, the spam dataset from the University of California, Irvine Center for Machine Learning Repository will be used. This dataset provides 56 different attributes calculated for different sample emails. These attributes include such features as maximum run-length of uppercase letters, average run-length of uppercase letters, percentage of words that match the word “order”, and percentage of words that match the word “address”.[2] (Many current spam filters look for words or sentences that are exclusively uppercase.) The dataset has over 6,000 samples that are broken into 3 different sets for this research: one set for training, another for validation, and the third for testing of the final, trained networks. In addition, the data in all three sets has been normalized and standardized. The normalization has been shown to help standard neural networks converge faster.[14]

4 Early Results

Early results of the experiment are inconclusive at this time. However, as can be seen in Figure 1, the results show that NEAT looks promising in the validation part of the dataset, but NEAT with AdaBoost seems to be better in the training and testing datasets. It also shows that NEAT with AdaBoost seems to be more consistent between the three different parts of the dataset. What this figure does not show is that the NEAT algorithm completed more iterations in the same amount of time than the other three algorithms. Until such time as more tests are completed with a greater number of generations, it will be difficult to determine whether adding AdaBoost to NEAT or rtNEAT results in faster convergence.

5 Conclusions and Future Works

The preliminary results do not show whether or not the use of AdaBoost with NEAT or rtNEAT is beneficial. As such, more experiments will be needed. Some of these might be to test different ways the NEAT generations can be used with the AdaBoost algorithm, including what might occur if the number of generations of NEAT is limited. It also might be interesting to see how other ensemble algorithms would work with NEAT instead of the AdaBoost algorithm. As these experiments have focused in on the AdaBoost.m2 algorithm, there are several other versions of AdaBoost that may work better. All of these are interesting topics that could be researched in future experiments.

6 References

[1] M. Vergelis, T. Shcherbakova, and N. Demidova. (2015, May) Kaspersky security bulletin. Spam in 2014. [Online]. Available: https://securelist.com/analysis/kaspersky-security

[2] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[3] (2015, May) Neural network research group. [Online]. Available: http://nn.cs.utexas.edu/

[4] K. O. Stanley and R. Miikkulainen, “Efficient reinforcement learning through evolving neural network topologies,” in Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO ’02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 569–577. [Online]. Available: http://dl.acm.org/citation.cfm?id=646205.758739

[5] K. O. Stanley, “Evolving neural networks,” in Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, ser. GECCO ’12. New York, NY, USA: ACM, 2012, pp. 805–826. [Online]. Available: http://doi.acm.org/10.1145/2330784.2330917

[6] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” Evol. Comput., vol. 10, no. 2, pp. 99–127, June 2002. [Online]. Available: http://dx.doi.org/10.1162/106365602320169811

[7] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, “Real-time neuroevolution in the nero video game,” Trans. Evol. Comp, vol. 9, no. 6, pp. 653–668, Dec. 2005. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2005.856210

[8] H. Schwenk and Y. Bengio, “Boosting neural networks,” Neural Comp., vol. 12, no. 8, pp. 1869–1887, Aug. 2000.

[9] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, Aug. 1997. [Online]. Available: http://dx.doi.org/10.1006/jcss.1997.1504

[10]S. J. Russell and P. Norvig, Artificial intelligence: a modern approach (3rd edition). Prentice Hall, 2009.

[11]X. Li, L. Wang, and E. Sung, “Adaboost with svm-based component classifiers,” Eng. Appl. Artif. Intell., vol. 21, no. 5, pp. 785–795, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2007.07.001

[12](2015, May) Nnrg software - neat c++. [Online]. Available: http://nn.cs.utexas.edu/?neat-c

[13](2015, May) Nnrg software - rtneat c++. [Online]. Available: http://nn.cs.utexas.edu/?rtNEAT

[14] Y. Lecun, L. Bottou, G. B. Orr, and K.-R. MÃijller, “Efficient backprop,” 1998.