Artificial Immune System Inspired Algorithm for Flow-Based Internet Traffic Classification

Brian Schmidt, Dionysios Kountanis, Ala Al-Fuqaha
Computer Science Department
College of Engineering and Applied Sciences
Western Michigan University
Kalamazoo, Michigan, USA
{brian.h.schmidt, dionysios.kountanis, ala.al-fuqaha}@wmich.edu

Abstract—**Internet traffic classification has been researched extensively in the last 10 years, with a few different algorithms applied to it. Internet traffic classification has also become more relevant because of its potential applications in the business world. Having information about network traffic has many benefits in network design, security, management, and accounting. The classification of network traffic is most easily achieved by Machine Learning algorithms, which can automatically build a model from training data, without much input from humans. Artificial Immune System classification algorithms have been used previously to classify network connections in network security systems [1]. They have proven to be very versatile, as well as having low sensitivity to input parameters. Because of this we are encouraged to explore the value of AIS algorithms to the Internet traffic classification problem. In this research, we propose an AIS-inspired algorithm for flow-based traffic classification, where each network flow is classified into an application class. We measure the algorithm's performance with and without the use of kernel functions, using a publicly available data set. We also compare the algorithm's performance with SVM and Naive Bayes classifiers. The algorithm generalizes well and gives high accuracy even with a small training set when compared to other algorithms, although the training and classification times were higher. The algorithm is also insensitive to the use of kernels, which makes it attractive for embedded and IoT applications.**

Keywords—artificial immune systems; internet traffic classification; multi-class classification; machine learning

I. Introduction

Because of issues with the reasonable utilization of the Internet, it has become more important to classify the data flowing over networks into application classes. An ISP can utilize traffic classification to prioritize the traffic of an application that might require it, as well as stop or slow down a user that is using illegal applications.

There are a few techniques with which network traffic has been classified. First and most simply, by detecting the port number that a flow is using, the application that generated it can be identified. Another way is to examine the contents of packets and compare them to classification rules, this approach is very accurate but does not scale well. When encryption is used, the data is impossible to classify directly in this manner. Another way of overcoming this weakness is by examining the interactions between the server and client, and comparing them to well-known applications.

Lastly, by examining the statistical features of network data, it is possible to identify the application that is exchanging the data. This is the approach taken in this paper and is attractive to ISPs because it does not require packet contents to be inspected, avoiding legal and ethical quandaries.

The focus of this paper will be to utilize the statistical features of network flows to identify the generating application. We will accomplish this by using a multi-class Artificial Immune System inspired classification algorithm. The rest of the paper is composed like this: in Section II, we overview some of the machine learning approaches that have been used to perform flow classification. In Section III, Artificial Immune System classifiers are presented, along with some improvements. Section IV presents our AIS-inspired classifier. Sections V and VI show our experiments and results, and lastly, in Section VII we draw conclusions and show a few ways in which we could further the research.

II. The Traffic Classification Problem in Machine Learning

As mentioned in the previous section, the flow classification problem can be solved by using the statistical features of the data travelling over the network. The features used are collated from the network data without looking at the contents of the packets, however the contents of packet headers can be examined for some features. Some examples of the information that can be used is: port numbers, inter-packet delay, packet counts, as well as the averages and medians of these values.

Using these types of features the authors of [4] applied a simple Naive Bayes classifier to the traffic classification problem. The research also applied kernel density estimation and Fact Correlation-Based Filtering (FCBF). The highest classification accuracy achieved was 96.3%.

In a review of a few different classification algorithms, Alshammari and Zincir-Heywood [5] used Naive Bayes, Support Vector Machine (SVM) RIPPER, and C4.5 classification algorithms to do flow classification. They performed tests on a publicly available data set, focusing on encrypted traffic. In another publication, Singh and Agrawal [6], also tested several of the same algorithms as [5]. The algorithms tested are Bayesian networks, multi-layer perceptron, C4.5 trees, Naive Bayes and the Radial Basis

Function Neural Network. They used feature reduction as well as full-feature data sets in their tests. The best performance is achieved by C4.5 classification trees on the reduced feature data set.

Even though Artificial Immune Systems classifiers have been used to perform traffic classification in the past, we have not found any examples of them being used to perform flow classification by application. The authors of [1] provide a good overview of relevant uses of AIS for classifying network traffic. We are motivated to apply AIS algorithms to this problem because of their insensitivity to parameters, which makes them ideal for resource-constrained network nodes, such as in IoT applications.

III. Artificial Immune systems

In this section, we introduce Artificial Immune System Algorithms, as well as the variations of AIS that allow multi-class classification.

A. Natural Immune Systems

Natural immune systems (NIS) protect organisms from outside threats. They mammalian immune system has inspired a few different algorithms, both for classification and optimization, however this research only focuses on classification. As part of its duties, the NIS classifies every cell in an organism as either "self" or "non-self", and then destroys any non-self cells.

B. Training Methods

To train itself to recognize cells, the NIS follows a simple procedure. The bone marrow produces cells known as B-cells, which in turn produce molecules known as "antibodies". An antibody is able to recognize cells by attaching to proteins found on their surface, acting as a simple classifier that recognizes a pattern. The population of B-cells is "trained" by the thymus, an organ found behind the sternum. This organ tests all B-cells before they mature and removes all B-cells from the population that recognize self tissues as non-self. In this way, only B-cells that recognize non-self tissues are kept in the body.

The equivalent computer algorithm is known as the Negative Selection algorithm. In this publication we will work with a similar algorithm, based on the process of positive selection, which is essentially the inverse of negative selection.

C. Multi-class Classification Using AIS

The first attempt at implementing multi-class classification with AIS was done by Goodman, Boggess, and Watkins [7] with their Artificial Immune System Recognition System (AIRS). This algorithm is distinct because of its insensitivity to its input parameters. Timmis and Neal introduce a resource-limited artificial immune system algorithms in [8]. Watkins and Bogess further refine the AIRS algorithm in [9] and [10]. A similar algorithm to AIRS is developed in [11]. Carted shows an AIS multi-class classifier algorithm in [12]. In [13], White and Garrett develop another AIS algorithm called CLONCLAS. Brownlee also develops a multi-class AIS classifier, called CSCA, in [14]. The negative selection algorithm is modified to perform multi-class classification in [15] and [16]. The algorithm trains a subpopulation of antibodies for each class present in the data set and then uses the antibodies to perform classification.

IV. Algorithm Description

Our algorithm classifies packet flows into application classes. Packet flows are sets of packets that have a source IP, destination IP, source port, destination port, and transport protocol in common. The data set we use was made available by Moore et al. [4]. The data set contains 249 features, of which we will use 11. The 11 features were chosen through a feature reduction process. The authors of [4] use a Fast Correlation-Based Filter (FCBF) to perform feature reduction on the data set, we follow the work done by them, using the same features. The features are listed in Table I.

There are 12 application classes present in the data set, which are listed in Table II along with examples of the applications. The FTP flows are further divided into three sub classes, with control, passive, and data flows each placed into their own class. Before the classification algorithm can be applied to a flow, the relevant features must be calculated. In this section, we describe how the algorithm works, assuming that the features are already calculated.

The algorithm works with a population of antibodies, which, like their biological counterparts, are simple classifiers. The antibodies are implemented as hyper spheres within an 11-dimensional space, with each antibody being defined by a vector, a scalar, and a class label. The vector is the center of the hyper sphere, in 11 dimensions, and the scalar is the radius. The antibody performs classification very simply, if an example vector falls within the radius of the hyper sphere, then the example is classified as being of the same class as the antibody. All data is normalized to the range of [0, 1].

TABLE I. Class Labels and Applications[4]

Feature	Description
Port, *server*	Port Number at server
Number of pushed data packets, *server->client*	# of packets with the PUSH bit set in the TCP header
Initial window bytes, *client->server*	# of bytes in the initial window
Initial window bytes, *server->client*	# of bytes in the initial window
Average segment size, *server->client*	The average segment size
IP data bytes median, *client->server*	Median of total bytes in IP packets
Actual data packets, *client->server*	# of packets with at least a byte of TCP data payload
Data bytes in the wire variance, *server->client*	Variance of # of bytes in Ethernet packet
Minimum segment size, *client->server*	The minimum segment size
RTT samples, *client->server*	The total number of Round Trip Time (RTT) samples.
Pushed data packets, *client->server*	# of packets with the PUSH bit set in the TCP header

TABLE II. CLASS LABELS AND APPLICATIONS[4]

Class Label	Applications
FTP-CONTROL, FTP PASV, FTP-DATA	FTP
DATABASE	Postgres, Sqlnet, Oracle
INTERACTIVE	SSH, klogin, rlogin, telnet
MAIL	IMAP, POP2/3, SMTP
SERVICES	X11, DNS, ident, LDAP, NTP
WWW	WWW
P2P	KaZaA, BitTorrent
ATTACK	Worm and virus attacks
GAMES	Half-Life
MULTIMEDIA	Windows Media Player

To initialize the population of antibodies, each class is allocated an equal portion of the antibody population. To create an antibody, the data in the training set is sampled with replacement, and an antibody is created centered on the sample and with the same class as the sample. The radius is initialized to zero.

To train the population of antibodies, each antibody's radius is expanded until is misclassifies a member of the training set. Each antibody is expanded by a step size, which is given as a parameter. Once the antibody reaches a non-self data point in the training set, it decreases its radius by the same step size, in order to prevent a misclassification.

Classification is performed by comparing the test example to each antibody in the population. If an antibody matches the test example, then the antibody's class is returned as the classification. If no antibody matches the example, then the class of the closest antibody is returned. If more than one antibody is at the same distance, then one of them is chosen at random.

V. MODEL VALIDATION AND EXPERIMENTAL SETUP

The algorithm was tested using stratified 10-fold cross validation. The testing, validation, and training sets were split according to 10%/10%/80%, respectively. The original data set contains 370,000 flows but we chose to limit the maximum data set size used in our tests to 1000, since making it any bigger would make the classes very unbalanced and affect the results.

Using the 10 sub data sets created by the 10-fold cross validation, each test was performed 10 times, and the results averaged. The algorithm is also implemented without kernel functions, and with polynomial kernel, a linear kernel, and a Gaussian kernel. The parameters for the kernels were chosen using a grid-search with the validation set. Each tests was also done with SVM and Naive Bayes classifiers.

Every test was done with the step size of the algorithm set to 0.01. We work exclusively with Euclidian distance to define the radius of each antibody. The algorithm is coded in Python. All tests were performed on an Intel Core i5 running at 1.8 GHz with 4 GB of memory.

VI. EXPERIMENTAL RESULTS

To show the classification accuracy achieved by our algorithm, we graph the antibody population size against the accuracy in Figure 2. The data set size is 1000 flows. The algorithm achieved a maximum accuracy of 92.3% with the linear kernel. The figure also includes the accuracy the SVM and Naive Bayes algorithms, tested with a data set of 1000 flows.

The accuracy of our algorithm is graphed against the data set size in Figure 3. The size of the antibody population used is 1000. The highest accuracy is 93.6%, achieved by the polynomial kernel. The SVM and Naive Bayes algorithms are also graphed. This figure also displays our algorithm's ability to generalize well from small training sets, as compared to the other algorithms. In Figures 2 and 3, it can be seen that no kernel improved the performance of our algorithm significantly.

Figure 4 shows the time required by the classification algorithm to classify 100 examples as the number of antibodies in the population increases. The time is displayed in seconds, and it can be seen that the time required increases linearly. The classification time of Naive Bayes and SVM classifiers are also displayed to serve as a baseline, their lines run along the bottom of the figure.

Figure 5 shows the time required by the initialization and training steps of the algorithm as the data set size grows. The time is displayed in seconds. The antibody population is 1000. The training time grows linearly with the data set size. The SVM and Naive Bayes algorithms are also displayed, the lines can be seen at the bottom of the figure.

As mentioned, our algorithm is able to achieve high accuracy with limited data. When comparing our results with the results in [2], it can be seen that our algorithm achieved equal or higher accuracy as all other classifiers with 1/3 of the training data, although our classifier does not exceed the accuracy of the best classifiers tested.

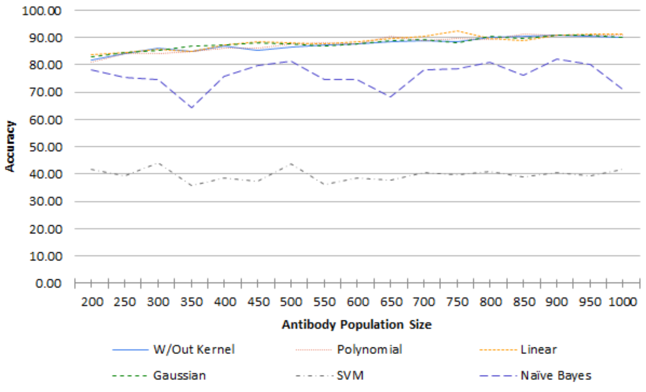

Fig 2. Classification accuracy and antibody population

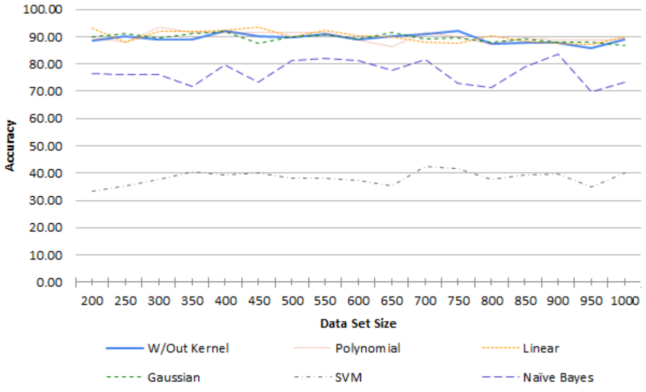

Fig 3. Classification accuracy and data set size

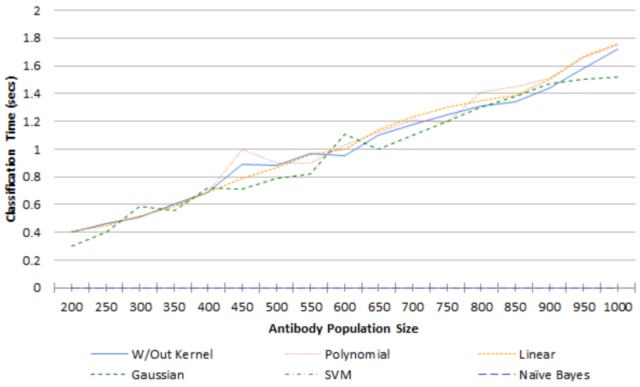

Fig 4. Classification time and antibody population size

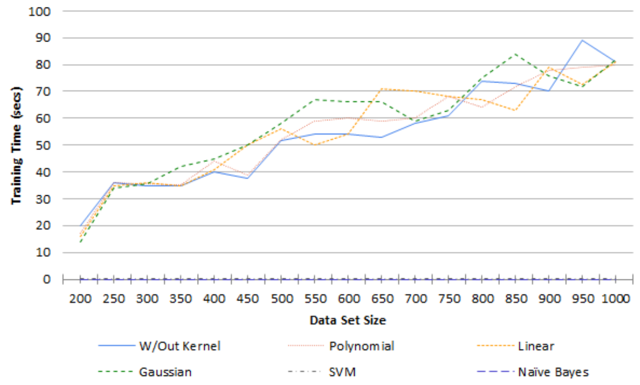

Fig 5. Training time and data set size

VI. Conclusions and Future Work

In this paper we showed the application of an AIS-inspired classification algorithm to the classification of network traffic according to classification. We outlined the classification performance of the algorithm, classification time, and training time. We tried to improve the classification accuracy of the algorithm with kernel functions, and although the highest accuracy was achieved with a kernel function, we believe that kernel functions do not significantly improve the algorithm. We also directly compare our algorithm with SVM and Naive Bayes classifiers.

Our algorithm's accuracy is similar to the accuracy of other algorithms tested on this data set [4]. Even though the algorithm is useful in any situation where network traffic classification is performed, we have found certain features of the algorithm make it especially useful in resource-limited systems such as IoT applications. Specifically, the algorithm's ability to generalize well from small training sets, as well as its insensitivity to kernel functions. In short, we were able to improve on the accuracy of Naïve Bayes and SVM classifiers when used with small training sets, but the training and classification steps of our AIS algorithm is slower than these algorithms.

The algorithm's training and classification times could be improved with the use of k-d tree or Bloom filter data structures. Furthermore, the algorithm could be easily modified to work in parallel processors such as GPUs, greatly increasing its performance.

References

[1] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco and J. Twycross, J. "Immune system approaches to intrusion detection–a review," in *Natural Computing*, pp. 413-466, 2007

[2] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*, pp. 11, December 2008

[3] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, pp. 41-54, 2005

[4] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33, No. 1, pp. 50-60, June 2005

[5] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA2009)*, pp. 1-8, 2009

[6] K. Singh and S. Agrawal, "Comparative analysis of five machine learning algorithms for IP traffic classification," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pp. 33-38, April 2011

[7] D. E. Goodman, L. Boggess, and A. Watkins, "Artificial immune system classification of multiple-class problems," in *Proceedings of the artificial neural networks in engineering*, vol. 2, pp. 179-183, 2002

[8] J. Timmis and M. Neal, "Investigating the evolution and stability of a resource limited artificial immune system," in *Proceedings of the genetic and evolutionary computation conference (GECCO)*, pp. 40-41, 2000

[9] A. Watkins and L. Boggess, "A New Classifier Based on Resource Limited Artificial Immune Systems," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, IEEE Press, 2002

[10] A. Watkins and L. Boggess, "A Resource Limited Artificial Immune Classifier," in *Proceedings of the 2002 Congress on Evolutionary Computation, Special Session on Artificial Immune Systems*. IEEE Press, 2002

[11] H. P. Cheng, and C. S. Cheng, "A hybrid multiclass classifier based on artificial immune algorithm and support vector machine," in *3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA)*, pp. 46-50, 2011

[12] J. H. Carter, "The immune system as a model for pattern recognition and classification," in *Journal of the American Medical Informatics Association*, 7(1), pp. 28-41, 2000

[13] J. A. White and S. M. Garrett, "Improved pattern recognition with artificial clonal selection?," in *Artificial Immune Systems*, pp. 181-193, 2003

[14] Brownlee, "Clonal Selection Theory & CLONALG–The Clonal Selection Classification Algorithm (CSCA)," Swinburne University of Technology, 2005

[15] U. Markowska-Kaczmar and B. Kordas, "Multi-class iteratively refined negative selection classifier," in *Applied Soft Computing*, pp. 972-984, 2008

[16] U. Markowska-Kaczmar and B. Kordas, "Negative Selection based method for Multi-Class problem classification," in *Sixth International Conference on Intelligent Systems Design and Applications*, Vol. 2, pp. 1165-1170, October 2006

[17] T. T. Nguyen, and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in *Communications Surveys & Tutorials*, 10(4), pp. 56-76, 2008