Training the SOFM Efficiently: An Example from Intrusion Detection

Leigh Wetmore, A. Nur Zincir-Heywood, Malcolm I. Heywood
Dalhousie University,
Faculty of Computer Science,
6050 University Avenue, Halifax, Nova Scotia. B3H 1W5

***Abstract*—The Dynamic Subset Selection (DSS) Active Learning algorithm is generalized to include the case of unsupervised learning. To do so, training set partitioning, exemplar difficulty and age, and early stopping criteria are introduced into the Self Organizing Feature Map algorithm. The resulting model is able to build a Hierarchical SOFM on a large (≈500 000 pattern) dataset in 3 hours. In comparison, the same architecture without active learning requires 33 hours to construct. No reduction in accuracy is recorded for the DSS SOFM model.**

***Index terms*—Self-Organizing Feature Map, Active Learning, Intrusion Detection Systems.**

I. Introduction

The Self Organizing Feature Map has received a lot of interest as a tool for analyzing network log data with the goal of detecting abnormal behaviors [1], [2], [3]. One major drawback of the method, however, lies in the computational overhead in conducting training over datasets consisting of hundreds of thousands of patterns – as is the norm for Intrusion Detection Systems (IDS). One approach would be to use hardware speedups. Although providing a solution of sorts, we believe that it should be possible to address the problem without recourse to specialist hardware. Specifically, considerable progress has been made with respect to the design of active learning algorithms [4], [5], [6]. These algorithms recognize that not all training patterns are created equal. Thus, rather than iterating over the entire training dataset, current performance is used to rank the difficulty of training patterns. This ranking is then used to filter the set of patterns over which training is actually performed, resulting in a significant speedup and potentially better models. To date, the active learning approach has only been demonstrated within the context of supervised learning algorithms. In this work we extend the approach to the case of unsupervised learning, and the Self Organizing Feature Map (SOFM) in particular.

Performance of the proposed learning algorithm is demonstrated on the 1999 Knowledge Discovery and Data Mining Tools Competition dataset (KDD-99) [7], where this describes an intrusion detection problem originally developed for a DARPA IDS initiative [8]. The dataset consists of approximately 500 000 training patterns of 41 features. The resulting hierarchical SOFM model is trained within matter of hours, as opposed to days when the SOFM is trained without active learning. Moreover, the accuracy of the resulting model is unaffected by the addition of active learning.

The remainder of the paper is organized as follows. Section II provides a brief summary of the SOFM algorithm, thus establishing the most costly components of the training algorithm. Section III reviews previous approaches to making the SOFM algorithm more efficient (as opposed to hardware acceleration). The proposed active learning algorithm is detailed in Section IV. Results are reported in Section V for the KDD-99 competition dataset, where this represents an intrusion detection problem described by a dataset of half a million patterns. Finally, conclusions and future work are identified in Section VI.

II. Basic SOFM Learning Algorithm

In order to establish the terminology and context for the following discussion of speedups to the SOFM algorithm, the basic SOFM algorithm is summarized as follows [9].

1. Initialize free parameters, w_{ij};
2. Present training pattern, $\boldsymbol{x}$;
3. Calculate the distance between exemplar, $\boldsymbol{x}$, and each neural weight vector, w_i. Return the argument of the closest neuron, or Best Matching Unit (BMU), j^*,
$$j^* = \arg\left(\min_j\left(\left\|x - w_j\right\|\right)\right)$$
where $\|\cdot\|$is the Euclidean norm;
4. Update all weights in the neighborhood of the winning neuron,
$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)K(j^*,t)\left\{x_i(t) - w_{ij}(t)\right\}$$
where $\eta(t)$ is the learning rate at instance t; and $K(j^*, t)$ is a suitable neighborhood function, centered on BMU j^*, in this case the following Gaussian function is employed,
$$K(j,t) = \exp\left(-\frac{\left\|l_k - l_j\right\|^2}{2\sigma^2(t)}\right)$$
where both the neighborhood radius function, $\sigma(t)$, and learning rate function, $\eta(t)$, decay with epoch (= mod(t, #training exemplars)), and l indexes neural location with respect to BMU;
5. Repeat steps (2) - (4) until all exemplars are presented;
6. If stop criteria is not satisfied return to step (2), else end.

Neighborhood radii and learning rate are modified in conjunction with an annealing schedule that monotonically decreases their values with each weight update. The typical form for such a schedule is usually to divide training into

This work was supported in part by Discovery grants from the Natural Sciences and Engineering Research Council of Canada and the CFI New Opportunities program. All research was conducted at the NIMS Laboratory.

two phases, ordering and convergence. Ordering is used to establish the basic topological ordering of the map and might take 1 000 updates to complete [10]. During this phase the learning rate will typically begin at around unity and decay to a value around 0.1. Likewise the neighborhood function will also begin at a suitably large value (dependent on initial size of the network), say 5, and decay to a neighborhood of one or two neurons. Phase two, or the convergence phase, establishes the fine-tuning of the network. As such the best accuracy is established by decreasing the learning rate slowly between values of 0.1 and 0.001. As a rule of thumb, Kohonen states that the number of 'training steps' should be "...at least 500 times the number of network units" [9], page 112. Moreover, in the case of large maps, the selection of an efficient annealing schedule is considered critical [9], page 112. In short, the selection of training stop criteria is a function of SOFM size and the underlying complexity of the data. A dataset with an easily identified data distribution requires less training time than data described by a more complex distribution. Thus, when comparing the proposed active learning methodology against that of the standard SOFM, we will consider the case of different epoch limits for the standard SOFM algorithm, Section V.

III. Related Work

Kohonen originally identified three speedups: Smoothing, Estimation, and Shortcut Search [9]. Specifically, smoothing refers to an early stopping heuristic in which the data driven fine tuning process is replaced by an algorithmic 'smoothing' of the neural locations i.e. applicable to the fine-tuning stage alone. Estimation refers to the specific scenario of training a large SOFM. To do so, a much smaller SOFM is first trained and used to initialize the prototype vectors of a much larger SOFM. Finally, shortcut search refers to another speedup specific to the fine tuning process. In this case, the BMU (step (3) of the generic SOFM algorithm, Section II) from the previous iteration is used to 'suggest' the BMU for the current iteration. In short, the three speedups originally identified by Kohonen are either specific to the fine tuning phase of adaption or only of use when training especially large maps. However, given that the majority of epochs are spent in the fine tuning phase, there is still the potential to receive reduction in computational requirements [11].

One final method of note provides a more general approach to seeding the SOFM by way of clustering [12]. In this case the k-means algorithm is used to establish the initial values for the SOFM prototype vectors. Unfortunately, only limited benchmarking was conducted (no computational comparison between the original SOFM algorithm and the proposed method) over data sets consisting of a few hundred exemplars, and several constraints result. Firstly, the topological map defining SOFM neighbors must be square. Secondly, the total number of rows and columns must be equal. Finally, even more emphasis is given to the regions of data with most density than in the case of the original SOFM algorithm, where this might result in problems on unbalanced datasets typical to data mining problems.

IV. SOFM and Active Learning

The active learning algorithm developed here is based on the Dynamic Subset Selection (DSS) algorithm of Gathercole and Ross [5]. This algorithm used the concepts of exemplar age and difficulty as the basis for exemplar ranking in a supervised learning context, Subsection IV.A. We then introduce the basis for the difficulty measure in an unsupervised learning context, Subsection IV.B. Given that the SOFM is typically trained for an *a priori* defined number of epochs against which learning parameters are annealed, our SOFM speedup should also consider how to identify when to stop training. This is described in Subsection IV.B. The ensuing hierarchical DSS SOFM algorithm is presented in subsection IV.C.

A. Dynamic Subset Selection for Supervised Learning

The original Dynamic Subset Selection algorithm introduced the concepts of exemplar age and difficulty as a method for focusing a supervised learning algorithm on a (fixed) subset of exemplars utilized over a single training epoch [5]. At each epoch the weight of each exemplar is calculated as a function of age and difficulty, thus exemplar i at epoch t is defined by,

$$W_i(t) = D_i(t)^d + A_i(t)^a;\ D_i(0) = A_i(0) = 0$$

where D and A are the difficulty and age of pattern i; and, d and a are user defined exponents.

Having computed the weight of each exemplar, the corresponding probability of selecting exemplar i for inclusion in the subset at epoch t is,

$$P_i(t) = \frac{W_i(t) \times T}{\sum_{j=1}^{T} W_j(t)}$$

where T is the desired size of the resulting subset. Patterns selected into the subset have their difficulties and ages reset, while those not selected have their ages incremented by 1. The difficulty of a pattern is incremented by 1 each time it is misclassified. The selection of T, d, and a is arbitrary and depends on the properties of the training data set. Optimal values can only be determined through experimentation.

DSS requires access to the entire data set to which it is applied. This is increasingly inefficient in the case of data sets too large to be stored in cache alone. A possible solution to this problem involves breaking a large data set up into subsets of exemplars [13]. There, the subsets of patterns are called blocks and are of equal size. DSS is applied to individual blocks read into memory; these blocks are selected at random with replacement. In addition two changes are made to the DSS algorithm itself. The first eliminates selection of exemplars by weight, and replaces it with selection by difficulty or age alone, where there is a probability p $(1 - p)$ that the next pattern will be selected by difficulty (age). The second fixes subset size; patterns are selected until the subset is full. This version of the algorithm does not have a, d and T as parameters,

eliminating the need to determine these parameters. Such a scheme will also form the basis for the work here.

Naturally, if exemplars have an associated exemplar difficulty and age, then we may now select blocks on the basis of their relative age and difficulty [6]. In this work we denote block difficulty as the difference between mean and median difficulty as calculated across the exemplars of a block. Moreover, block age is also calculated, with block selection performed in the same way as exemplar selection.

B. Dynamic Subset Selection for Unsupervised Learning

In order to support the concept of exemplar difficulty in an unsupervised learning context we recognize that the distance calculation already provides a measure of exemplar 'difficulty'. Thus, easy (difficult) exemplars are those that are currently 'close' ('distant') to one or more SOFM neuron(s). In addition, the difficulty of any exemplar should reflect performance of the SOFM over several iterations. Thus a gradual change in exemplar difficulty is established, as follows,

$$d(v) = \alpha \parallel x - w_i \parallel + (1 - \alpha)\, d(v - 1) \quad (1)$$

That is to say, the current difficulty is dependent upon the distance from exemplar, x, to BMU, w_i, and the previous difficulty, $d(v-1)$, with more weight going to the latter as training progresses. Exemplar age is merely defined by a count. Thus, for each epoch an exemplar is not selected the corresponding age is incremented.

From Section II it is apparent that the selection of generic stopping conditions for the SOFM is determined a priori independent of progress made during training. In this work we assume that block difficulty provides a good indicator of learning algorithm process. Thus, once all block difficulties have plateaued, training is complete. Redefining stop criteria in this way now means we cannot a priori define an annealing schedule for controlling neighborhood radii or learning rate. Instead both learning rate and radii are linearly mapped to block difficulty. This means that it is now possible for learning rate and radii to increase if the SOFM representation deteriorates during training (as may occur if block content changes significantly). As in the case of the generic SOFM algorithm both learning rate and radii have a predefined range of values. Once, the learning rate reaches the minimum limit during the ordering phase, SOFM fine-tuning takes place for twice as many epochs as were spent during the ordering phase. In this context an epoch is conducted over the subset of exemplars selected by the DSS algorithm.

C. Hierarchical SOFM with Dynamic Subset Selection

A training data set P is divided into blocks, B, of equal size b. For each block selection, n subset selections of size s are performed. The current SOFM learning rate is lr, and the fine-tuning learning rate is lr_{ft}. The current neighborhood radius is r. There is a probability p_b of a block being selected in proportion to the block difficulty (or in the case of age, $1 - p_b$). Likewise exemplars are selected for inclusion within the current subset in proportion to the corresponding exemplar difficulty p_b (or in the case of age, $1 - p_b$). The linear mapping from the average block difficulty to the learning rate and radius is m. Figure 1 details the complete system. Step 3 is the main loop in which block selections are made (point (3.a)), subsets constructed (point (3.b.i) and SOFM training performed (point (3.b.ii). Exemplar age and difficulty is updated at each exemplar presentation (point (3.d.iv.B). Block age and difficulty are updated following each block selection (point 3.c) and global learning parameters updated accordingly (point 3.d and 3.e).

The speedup provided by the algorithm is as a result of fewer exemplars being presented to the SOFM during training. This reduction in the number of exemplars actually seen during training is due to two mechanisms: epochs are conducted over a small fraction of the original data set i.e., exemplar filtering; and, a redefinition of the stop criteria in terms of block difficulty convergence.

1. Initialize pattern difficulty in P following a pass through P. All pattern ages are unity;
2. Using the pattern difficulty, initialize block difficulty;
3. While ($lr > lr_{ft}$)
   a) IF (rb > pb) THEN (Select block B with respect to block Difficulty) ELSE (Select block B with respect to block Age)
   b) For (n epochs)
      i) Perform DSS on B.
      ii) For (S iterations)
         A) IF ($r_p \le p_p$) THEN (select pattern 'p' from block B in proportion to pattern difficulty) ELSE select pattern 'p' from block B in proportion to pattern age)
         B) Add pattern 'p' to subset S without replacement.
      iii) Increment ages of all patterns in block B not a member of subset S.
      iv) For each pattern of subset S
         A) Train SOFM on pattern 'p'
         B) Update difficulty of pattern 'p', equ. (1) and reset age to unity.
   c) Re-compute block difficulty, given the updated pattern difficulties. Increment ages of all other blocks;
   d) If average block difficulty reaches a new maximum, re-compute m.
   e) Use m and average block difficulty to update lr and r.

Fig. 1. Algorithm for Hierarchical DSS-SOFM.

V. EVALUATION

Previous work has demonstrated that the SOFM has the potential to provide the basis for a detector within the context of IDS [1], [2], [3]. For comparative purposes we focus on the KDD-99 variant of the original DARPA – Lincoln Laboratory benchmark. In the following the characteristics of the KDD-99 dataset are summarized, the parameterization of the DSS-SOFM discussed, results presented between SOFMs trained with and without the DSS algorithm, and comparisons made against previous solutions on the KDD-99 dataset.

Performance of the detector is evaluated in terms of the false positive and detection rates, estimated as follows,

$$\text{False Positive Rate} = \frac{\text{Number of False Positives}}{\text{Total number of Normal Connections}}$$

$$\text{Detection Rate} = 1 - \frac{\text{Number of False Negatives}}{\text{Total number of Attack Connections}}$$

Here, False Positive (Negative) Rate is the number of Normal (Attack) connections labeled as Attack (Normal). All experiments were run on an Xserver running Mac OS X, (Jaguar) with two 1.33 GHz G4 processors and 2 GB of RAM. The data sets in use have all been partitioned (by their creators) into training and test partitions, as described below.

A. KDD-99 dataset

The KDD-99 dataset is based on the 1998 DARPA initiative to provide designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies [7]. To do so, a simulation is made of a factitious military network consisting of three 'target' machines running various operating systems and services. Additional three machines are then used to spoof different IP addresses, thus generating traffic between different IP addresses. Finally, there is a sniffer that records all network traffic using the TCP dump format. The total simulated period is seven weeks. Normal connections are created to profile that expected in a military network and attacks fall into one of five categories: User to Root; Remote to Local; Denial of Service; Data; and Probe. Note that User to Root and Remote to Local can represent content-based attacks, and may therefore only be detected indirectly by the type of system developed in this work (e.g. guessing passwords often manifests itself as multiple attempted login's between the same source destination pair).

In 1999 the original TCP dump files were preprocessed for utilization in the Intrusion Detection System benchmark of the International Knowledge Discovery and Data Mining Tools Competition [2]. To do so, packet information in the TCP dump files are summarized into connections. Specifically, "a connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol" [8]. This process is completed using the Bro IDS [2], resulting in nine "Basic features of an Individual TCP connection" [8]; hereafter referred to as 'basic features',

- Duration of the connection;
- Protocol type, such as TCP, UDP or ICMP;
- Service type, such as FTP, HTTP, Telnet;
- Status flag, derived by Bro to describe a connection;
- Total bytes sent to destination host;
- Total bytes sent to source host;
- Whether source and destination addresses are the same or not;
- Number of wrong fragments;
- Number of urgent packets;

Note that only Protocol and Service features are not derived i.e. they are estimated immediately as opposed to after a connection has completed. Moreover, the above 'status flag' should not be confused with the TCP/IP suit flags. Finally, last three features are specific to certain attack types (no variation is observed across the normal data in the training set), hence these terms were ignored in this work.

In addition to the above nine 'basic features,' each connection is also described in terms of an additional 32 *derived* features, falling into three categories,

- *Content Features*: Domain knowledge is used to assess the payload of the original TCP packets. This includes features such as the number of failed login attempts;
- *Time-based Traffic Features*: These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval;
- *Host-based Traffic Features*: Utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

The KDD-99 data consists of several components, Table I. As in the case of the International Knowledge Discovery and Data Mining Tools Competition, only the '10% KDD' data is employed for the purposes of training [2]. Moreover, the attack types are not represented equally, with Denial-of-Service attack types – by the very nature of the attack type – accounting for the majority of the attack instances. The so-called 'Corrected (Test)' dataset provides a dataset with a significantly different statistical distribution than '10% KDD' and an additional 14 (unseen) attacks.

TABLE I
BASIC CHARACTERISTICS OF THE KDD DATASET

Dataset label	Total Attack	Total Normal
10% KDD	396,744	97,277
Corrected (Test)	250,436	60,593
Category Composition		
Category	10%	Test
Normal	97 278	60 593
DoS	391 458	229 853
Probe	4 107	4 166
R2L	1 126	16 347
U2R	52	70

B. Parameterization

Parameters remain constant across all experiments. These parameters, and their respective values, are listed in Table II. The acronym ABD refers to the Average Block Difficulty. It should be noted that when the neighborhood radius is reduced after the average block difficulty plateaus, the learning rate is reduced correspondingly. In the case of the DSS SOFM model 20 different initializations are considered (verification of independence from initial conditions). Best-case training data was used to select the architecture employed for test purposes, thus facilitating comparison with solutions found by other researchers.

In the following sections, the network dimension of a first-level SOFM or DSS SOFM shall be referred to simply as its size. Testing partition classification accuracy and

training partition classification accuracy shall be referred to simply as testing accuracy and training accuracy, respectively. Experiments with single layer maps are performed against three map sizes: 10×10, 8×8 and 6×6. In this way we are able to identify whether there are any 'break even' points at which the DSS SOFM is the most appropriate.

TABLE II
SOFM TRAINING PARAMETERS

Parameter	Value
Initial Learning Rate	0.3
Fine-Tuning Learning Rate	0.01
Fraction of Total Time Spent Fine Tuning	0.67
Minimum Training Time	100 block selections
Probability of block Selection by Difficulty	0.9
Probability of exemplar Selection by Difficulty	0.7
Size of ABD History	100
Case of ABD Plateau	Reduce SOFM radii by '1'
Second Level Network Dimension	10 × 10
Trial performed per parameterization	20

C. *Performance*

Previous work with the standard SOFM architecture on the KDD data set has demonstrated that training times in the order of 33 hours are required for a first layer map of 6x6 (4 000 epochs) and six second layer maps of size 10x10, [2]. Given such an overhead, the KDD-99 data set represents the type of problem for which the DSS SOFM algorithm was explicitly designed. Performance is quoted for accuracies at each of the five exemplar categories, false positive and detection rates and computational requirements over the three map sizes. Tables III and IV detail performance under training and test partitions respectively for two layer maps.

It is apparent that the larger classes of Normal and DoS are easily recognized. Performance on probe is also reasonable at about 70% detection on test. The low training set exemplar counts for R2L and U2R, however, clearly penalized the generalization of these classes under the test set.

TABLE III
CHARACTERIZATION OF DSS-SOFM HIERARCHY ON 10% KDD

SOFM	10×10	8×8	6×6
Normal	0.999	0.933	0.993
DoS	0.999	0.999	0.999
Probe	0.885	0.876	0.825
R2L	0.885	0.876	0.825
U2R	0.192	0.250	0.173
FP	0.001	0.007	0.007
Detection	0.996	0.995	0.996
Time (hrs)	10 544	12 697	12 161

TABLE IV
CHARACTERIZATION OF DSS-SOFM HIERARCHY ON KDD TEST

SOFM	10×10	8×8	6×6
Normal	0.995	0.980	0.995
DoS	0.972	0.970	0.970
Probe	0.704	0.692	0.709
R2L	0.008	0.008	0.008
U2R	0.0	0.043	0.029
FP	0.005	0.020	0.005
Detection	0.904	0.902	0.903

Figure 2 illustrates the rate at which patterns are selected during training. It is apparent that the two largest classes (DoS and Normal) are selected least frequently, whereas the last frequently occurring pattern classes appear most frequently. The lower frequency in selection of R2L relative to Probe (where the latter has a higher training pattern count) is reflected in the equivalent training classification accuracies. U2R however was never able to improve beyond about 20% (training) accuracy in spite of the higher rate of sampling.

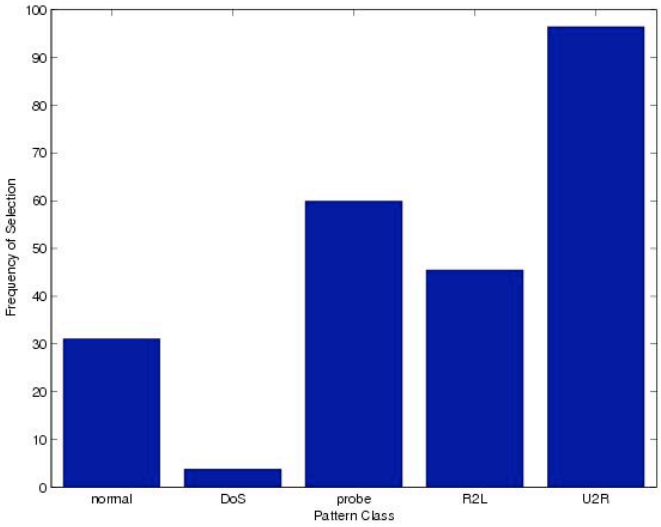

Fig. 2. DSS-SOFM pattern selection frequency.

In comparison with previous results on this data set, Table V, the DSS SOFM algorithm appears to produce very competitive results, even against the original KDD-99 competition winners (first 2 rows). Computationally, the DSS SOFM algorithm is an order of magnitude faster than the standard SOFM algorithm (3 hours versus 33 hours) whilst matching the false positive and detection rates. Thus, as the data-mining problem becomes larger the potential speedup increases relative to the original SOFM algorithm.

TABLE V
RECENT RESULTS ON THE KDD-99 BENCHMARK

Algorithm	Type of Learning	Detection Rate	FP Rate
C5 Ensemble [14]	Supervised	0.918	0.005
Kernel Miner [15]	Supervised	0.918	0.006
SOFM [2]	Unsupervised	0.906	0.016
GP [13]	Supervised	0.900	0.009
SVM [16]	Unsupervised	0.980	0.100
Estimation Clustering [16]	Unsupervised	0.930	0.100
Nearest Neighbor [16]	Supervised	0.910	0.080

VI. CONCLUSION

The Dynamic Subset Selection Active Learning Algorithm from supervised learning has been used as the basis for speeding up the SOFM model of unsupervised learning. To do so we introduce alternative definitions for pattern difficulty, as well as stop criteria appropriate to the large dataset context with which the DSS SOFM algorithm is proposed. This results in several orders of magnitude speedup in training time, relative to hierarchical SOFM architectures built without the active learning algorithm. No reductions in accuracy are observed. Future work will naturally consider the application of the algorithm to additional datasets of a similar size and the utility of the active learning approach to a wider cross-section of clustering algorithms.

REFERENCES

[1] P. Lichodzijewski, A.N. Zincir-Heywood, M.I. Heywood, "Host-Based intrusion detection using Self-Organizing Maps," *IEEE-INNS International Joint Conference on Neural Networks*, pp. 1714-1719, May 12-17, 2002.

[2] H.G. Kayacik, A.N. Zincir-Heywood, M.I. Heywood, "On the Capability of an SOM based Intrusion Detection System," *IEEE-INNS International Joint Conference on Neural Networks*, pp. 1808-1813, 2003.

[3] M. Ramadas, S. Ostermann, B. Tjaden, "detecting Anomalous Network Traffic with Self-Organizing Maps," 6th International Symposium on Recent Advances in Intrusion Detection Systems, RAID'03, Lecture Notes in Computer Science, Springer-Verlag. Vol. 2820, pp 36-54, 2003

[4] D. Cohn D., L. Atlas, R. Ladner, "Training connectionist networks with queries and selective sampling," Advances in Neural Information Processing Systems 2. Touretzky D. (ed), Morgan Kaufmann. Pp 566-573, 1990.

[5] C. Gathercole and P. Ross, "Dynamic Training Subset Selection for Supervised Learning in Genetic Programming." In Parallel Problem Solving from Nature III. Lecture Notes in Computer Science, Vol. 866. Springer-Verlag, pp 312-321, 1994.

[6] R. Curry, M.I. Heywood, "Towards Efficient Training on Large Datasets for Genetic Programming," Advances in Artificial Intelligence, Proceedings of the 17th Canadian Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence, 3060. Springer-Verlag. Pp 161-174, 2004

[7] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation." In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), Vol. 2, pp 12-26, 2000.

[8] L. Wenke, S.J. Stolfo, and K.W. Mok, "A data mining framework for building intrusion detection models." In Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp 120 –132, 1999.

[9] T. Kohonen, *Self-Organizing Maps*. 3rd Ed., Springer-Verlag, ISBN 3-540-67921-9, 2000.

[10] S. Haykin, *Neural Networks – A Comprehensive Foundation* (2nd Edition). Prentice-Hall, New Jersey, 1999.

[11] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Hokela, V. Paatero, A. Saarela, Self Organization of a Massive Document Collection. *IEEE Transactions on Neural Networks*. 11(3) 574-585, 2000.

[12] M.-C. Su, H.-T. Chang, "Fast Self-Organizing Feature Map Algorithm. *IEEE Transactions on Neural Networks*. 11(3) 721-733, 2000.

[13] D. Song, M.I. Heywood, A.N. Zincir-Heywood, "A Linear Genetic Programming Approach to Intrusion Detection'" Proceedings of the Genetic and Evolutionary Computation Conference. Lecture Notes in Computer Science. Vol. 2724. Springer-Verlag, Berlin. pp 2325-2336, 2003.

[14] B. Pfahringer, "Winning the KDD99 Classification Cup: Bagged Boosting," ACM SIGKDD Explorations, 1(2): 65-66, 2000.

[15] I. Levin, "KDD-99 Classifier Learning Context: Llsoft's results overview," ACM SIGKDD Explorations, 1(2): 67-75, 2000.

[16] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, Chapter 4, D. Barbara and S. Jajodia (editors), Kluwer, 2002.