On Dataset Biases in a Learning System with Minimum A Priori Information for Intrusion Detection

H. G. Kayacik
Dalhousie University
Faculty of Computer Science
Halifax, NS, Canada
kayacik@cs.dal.ca

A. N. Zincir-Heywood
Dalhousie University
Faculty of Computer Science
Halifax, NS, Canada
zincir@cs.dal.ca

M. I. Heywood
Dalhousie University
Faculty of Computer Science
Halifax, NS, Canada
mheywood@cs.dal.ca

Abstract

A critical design decision in the construction of intrusion detection systems is often the selection of features describing the characteristics of the data being learnt. Selecting features often requires a priori or expert knowledge and may lead to the introduction of specific attack biases – intended or otherwise. To this end, summarized network connections from the DARPA 98 Lincoln Labs dataset are employed for training and testing a data driven learning architecture. The learning architecture is composed from a hierarchy of self-organizing feature maps. Such a scheme is entirely unsupervised, thus the quality of the intrusion detection system is directly influenced by the quality of the dataset. Dataset biases are investigated through three different dataset partitions: 10% KDD (default training dataset); normal connections alone; 50/50 mix of attack and normal. The three resulting intrusion detection systems appear to be competitive with the alternative cluster based data-mining approaches.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access;
I.5.1 [**Models**]: neural nets

General Terms

Security, Design

Keywords

Intrusion detection, Self-Organizing Maps

1 Introduction

Along with benefits, the Internet also created numerous ways to compromise the stability and security of the systems connected to it. Although static defense mechanisms such as firewalls and software updates can provide a reasonable level of security, more dynamic mechanisms such as intrusion detection systems (IDS) should also be utilized. Intrusion detection systems are typically classified as host based or network based. A host based IDS will monitor resources such as system logs, file systems and disk resources; whereas a network based intrusion detection system monitors the data passing through the network. Different detection techniques can be employed to search for attack patterns in the data monitored. Misuse detection systems try to find attack signatures in the monitored resource. Anomaly detection systems typically rely on knowledge of normal behavior and flag any deviation from this. Intrusion detection systems currently in use typically require human input to create attack signatures or to determine effective models for normal behavior. Support for learning algorithms provides a potential alternative to expensive human input. The main task of such a learning algorithm is to discover directly from (training) data appropriate models for characterizing normal/ not normal behavior. The ensuing model is then used to make predictions regarding unseen data. One of the biggest challenges in network-based intrusion detection is the extensive amount of data collected from the network. Therefore, before feeding the data to learning algorithm, raw network traffic should be summarized into higher-level events such as connection records. Each higher-level event is described with a set of features. Selecting good features is a crucial activity and requires extensive domain knowledge. Unfortunately, the selection of features has the potential to introduce uncontrollable biases into the operation of the learning algorithm.

Given the significance of the intrusion detection problem, there have been various initiatives that attempt to quantify the current state of the art. In particular, MIT Lincoln Lab's DARPA intrusion detection evaluation datasets have been employed to design and test intrusion detection systems. In 1999, recorded network traffic from the DARPA 98 Lincoln Lab dataset [8] was summarized into network connections with 41-features per connection. This formed the KDD 99 intrusion detection dataset in the International Knowledge Discovery and Data Mining Tools Competition [2]. Although not without its drawbacks [3], KDD 99 provides the only *labeled* dataset for comparing IDS systems, which the authors are aware of. The most recent works based on unsupervised learning systems in this area are able to provide detection (false positive) rates

in the range of 91% (8%) to 98% (10%) whilst using all 41-connection features [4, 6].

In this work, a learning system based on a hierarchy of Self-Organizing Maps (SOM) is utilized. The scheme naturally provides a divide and conquer approach to learning and has been demonstrated in a range of real-world machine learning problems, including fingerprint recognition [5] and intrusion detection [7]. The principle interest of this work, however, lies in investigating the significance of utilizing "basic" connection features (a case of minimal *a priori* information). Specifically this implies that features may only be derived from the original packet headers without inspecting the packet payload and do not directly target the detection of specific attacks. Another feature in KDD 99 dataset is intended to detect only land attacks. The "basic" connection features take the form of the six (of the nine) "Basic features of an Individual TCP connection" [2]. These basic features can be derived from every connection regardless of the nature of the connection.

In the work reported here a two level SOM architecture is employed. At the first level, six SOMs are built, one for each input feature, where the general objective is to encode the necessary temporal features. The second level of the hierarchy integrates the information from each first level SOM. Neurons in the second level are therefore labeled using the training set, but the training process itself is entirely unsupervised. Moreover, a direct implication of the proposed scheme is that it makes very little use of any *a priori* information. This means that the composition of the training set can have significant implications on the quality of the detector. To this end, three different partitions are considered: the 10% portion of the KDD dataset as provided for training in the original competition [2]; the same data set without any attack connections (an anomaly detection system); and a dataset composed from an equal number of normal and attack connections (connections derived from the 10% KDD data alone). Of the three systems, that built on the 10% KDD data performed better than the other two cases in terms of false positive rate. However, the system trained on an equal number of attack and normal connections performed better than the other two systems on novel (unseen) attacks. The implication being that better generalization appears on a dataset biased towards the representation of attack connections. Detection (false positive) rate of this system on the test set varies between 89% (4.6%) to 91.5% (14.5%) depending on the KDD-99 test partition employed.

The remainder of the paper is organized as follows. Section 2 provides the background and methodology of the work. Details of each learning algorithm comprising the system are given in Section 3. Results are reported in Section 4 and Conclusions are drawn in Section 5.

2 Methodology

As indicated in the introduction, the basic objective of this work is to assess the significance of features used to build an IDS with minimal *a priori* domain knowledge during the design and feature selection process. To this end, an approach based on topological SOMs is employed. This assumes that given sufficient resolution in the SOMs, it is possible to separate normal from attack behavior. Moreover, all exemplars are considered to have equal risk. In a previous work, it is established that by utilizing a shift register to embed the temporal relationship between incoming connections, described in terms of session information, a simple two level SOM hierarchy was sufficient to distinguish between different behaviors [7]. However, the dataset used in that scheme was "home grown" and only consisted of seven attacks. In this work, on the other hand, we thoroughly benchmark the hierarchical SOM methodology on the KDD-99 benchmark. To this end, we first describe the characteristics of the data set and then the SOM architecture utilized.

2.1 KDD Dataset

The KDD-99 dataset is based on the 1998 DARPA initiative to provide designers of intrusion detection systems with a benchmark on which to evaluate different IDS methodologies [8]. To do so, a simulation is made of a factitious military network consisting of three 'target' machines running various operating systems and services. Additional machines are then used to spoof different IP addresses, thus generating traffic between different IP addresses. Finally, a sniffer is used to record 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 four categories: User to Root; Remote to Local; Denial of Service; and Probe.

- *Denial of Service (dos):* Attacker tries to prevent legitimate users from using a service.
- *Remote to Local (r2l):* Attacker does not have an account on the victim machine, hence tries to gain access.
- *User to Root (u2r):* Attacker has local access to the victim machine and tries to gain super user privileges.
- *Probe:* Attacker tries to gain information about the target host.

Note that "User to Root" and "Remote to Local" attacks typically 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 file 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" [9]. This process

is completed using the Bro IDS [10], resulting in nine "Basic features of an Individual TCP connection"; 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, the 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.

In this work, none of these additional derived features are employed.

2.2 Hierarchical SOM

The basic motivation for utilizing a hierarchical SOM architecture is to steadily build more abstract features as the number of SOM levels increase. That is to say, our hypothesis is that features learnt at the initial levels of a hierarchy may still be interpreted in terms of recognizable basic measured properties, whereas features at the highest level in the architecture will capture aspects synonymous with normal or attack behaviors (depending on the content of the training dataset). Specifically, two levels are employed. In the first, individual SOMs are associated with each basic TCP feature. This provides a concise summary of the interesting properties of each basic feature, as derived over a suitable temporal horizon. The second level SOM integrates the views provided by the first level SOMs into a single view of the problem. At this point, we use the training set labels associated with each connection to label the respective best matching unit (neuron) in the second level.

2.3 Preprocessing and Clustering

In order to build the hierarchical SOM architecture, several data normalization operations are necessary, where these are for the purposes of preprocessing and inter-level 'quantization' of SOMs. Preprocessing has two basic functions, to provide a suitable representation for the initial data and support the representation of time. In the case of initial data representation, three of the basic features – Protocol type, Service type and Status flag – are alphanumeric. As the first SOM level treats each feature independently, we merely map each instance of an alphanumeric character to sequential integer values. Numerical features – connection duration, total bytes set to destination/ source host – are used unchanged.

In the case of representing time, the standard SOM used here has no capacity to recall histories of patterns directly. However, sequence as opposed to time stamp, is the property of significance in this work [7]. A shift register of length 'l' is therefore employed in which a 'tap' is taken at a predetermined repeating interval 't' such that $l \,\%\, t = 0$, where $\%$ is the modulus operator. The first level SOMs only receive values from the shift register that correspond to tap locations. Thus, as each new connection is encountered (enters at the left), the content of each shift register location is transferred one location (to the right), with the previous item in the lth location being lost.

The purpose of the second level SOM is to provide an integrated view of the input feature specific SOMs developed in the first level. There is therefore the potential for each neuron in the second level SOM to have an input dimension defined by the total neuron count across all first level SOM networks. This would be a brute force solution that does not scale computationally (there are half a million training set patterns). Moreover, given the topological ordering provided by the SOM, neighboring neurons will respond to similar stimuli. We therefore quantize the topology of each first level SOM in terms of a fixed number of neurons and re-express the first level best matching units in terms of these alone. This significantly reduces the dimension seen by neurons in the second level SOM.

3 Learning Algorithms

Two learning algorithms are used to build the hierarchical SOM architecture. The first is used to train each SOM in the hierarchy [11]. The second is a clustering algorithm that is used to quantize the number of SOM neurons 'perceived' by the second level. In the latter case the Potential Function algorithm is employed [12]. These are briefly summarized as follows.

3.1 Self-Organizing Feature Map

Kohonen's Self-Organizing Feature Map (SOM) algorithm is an unsupervised learning algorithm in which an initially 'soft' competition takes place between neurons to provide a topological arrangement between neurons at convergence [11]. The learning process is summarized as follows,

1. Assign random values to the initial network weights, w_{ij};
2. Present an input pattern, $\boldsymbol{x}$, in this case a series of taps taken from the shift register providing the 'reconstruction' state space on which the SOM is to provide a suitable quantized approximation.
3. Calculate the distance between pattern, $\boldsymbol{x}$, and each neuron weight $\boldsymbol{w}_j$, therefore identify the winning neuron, or
$$d = \min_j \left\{ \left\| x - w_j \right\| \right\} \quad (1)$$
where $\|\cdot\|$ is the Euclidean norm and $\boldsymbol{w}_j$ is the weight vector of neuron j;
4. Adjust all weights in the neighborhood of the winning neuron, or
$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)K(j,t)\left\{ x_i(t) - w_{ij}(t) \right\} \quad (2)$$
where $\eta(t)$ is the learning rate at epoch t; and K(j, t) is a suitable neighborhood function;
5. Repeat steps (2) – (4) until the convergence criterion is satisfied.

Following convergence, presentation of an input vector, $\boldsymbol{x}$, results in a corresponding output vector, $\boldsymbol{d}$, the Euclidian distance between each neuron and input. The neuron with the smallest distance represents the winning or best matching unit, step (3). Once the SOMs are trained, it is this concept of a best matching unit that is used to facilitate the labeling of the second level SOM.

3.2 Potential Function Clustering

The Potential Function Clustering algorithm consists of four steps [12],

1. Identify the potential of each data point relative to all other data points. All data points represent candidate cluster centers;
2. Select the data point with largest potential and label as a cluster center;
3. Subtract the potential of the data point identified at step (2) from all others and remove this point from the list of candidate cluster centers;
4. Repeat on step (2) until the end criterion is satisfied.

In this application, the set of data points correspond to the set of neurons in each (first level) SOM, where the weights of each neuron describe a neuron position in terms of the original input space. Step 1 characterizes neurons in terms of how close they are to others. A neuron with many local neighbors should have a high 'potential' as expressed by a suitable cost function, or

$$P_t(w(j)) = \sum_{i=1}^{M} \exp\left(-\alpha \left\| w(i) - w(j) \right\|^2 \right)$$

where $w(j)$ is the 'j'th SOM neuron, $P_t(w(j))$ is the potential for such neuron at iteration t, M is the number of data points (in this case SOM neurons), and α is the cluster radii.

Step 2 identifies a candidate cluster center (SOM neuron) by choosing the point with largest potential $P_t(x^*)$. Step 3 removes the influence of the chosen neuron from the remaining (unselected) set of SOM neurons. That is, the remaining neurons have their respective potentials decreased by a factor proportional to the distance from the current cluster center, or

$$P_{t+1}(w(j)) = P_t(w(j)) - P_t(w^*)\exp\left(-\beta \left\| w(i) - w(j) \right\|^2 \right)$$

where $t + 1$ is the index of the updated potential at iteration t; w^* is the data point associated with the current cluster center, and β is the cluster radii ($\beta < \alpha$).

The result of step 3 is the labeling of a specific SOM neuron as a cluster center. Step 4 iterates the process in conjunction with some suitable stop criterion. In this case, we stop when six cluster centers are identified, where the alpha and beta values are set accordingly. That is to say, further cluster centers correspond to points with a potential value less than 10% of the first potential located. The net effect of this process is therefore that each of the six first level SOMs are characterized in terms of 6 clusters, resulting in a total of 36 inputs to the second level SOM.

Once the 6 cluster centers are identified for each SOM, representing the 'quantized' SOM output, we normalize as follows,

$$y = \frac{1}{1 + \left\| w - x \right\|}$$

where w is the cluster center and $\boldsymbol{x}$ is the original SOM input. The second level SOM now receives a vector, $\boldsymbol{y}$, of the form,

$$\boldsymbol{y} = [y_{1,1}, \ldots, y_{1,6}, y_{2,1}, \ldots, y_{i,j}]$$

where i is the SOM index and j is the cluster (neuron) index.

4 Results

In all cases the SOM Toolbox and SOM-PAK were employed for the design of each SOM comprising the SOM hierarchy [13]. In the following we describe the dataset, training procedure and evaluation of the proposed architecture.

4.1 KDD-99 Dataset

The KDD-99 data consists of several components, Table 1. 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]. This contains 24 attack types and is essentially a more concise version of the 'Whole KDD' dataset. One side effect of this is that it actually contains more examples of attacks than normal connections. 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 either '10% KDD' or 'Whole KDD (Test)' and contains an additional 14 (unseen) attacks.

Given the unsupervised nature of the learning system, the principle method for incorporating any *a priori* information is through the dataset. To this end, three different approaches are employed for preparing the training data set, Table 2. In the first approach, SOMs are trained with the 10% KDD (as it is). In the second approach, the 10% KDD dataset is filtered so that only the normal connections are used to train the SOMs. In doing so, the objective is to train the SOM hierarchy as a network anomaly detection system. Finally, in the third approach, a training dataset that consists of 50% attack and 50% normal connections is utilized. This is achieved by using all the 97,277 normal connections, and the first 97,277 attack connections from the 10% KDD.

Dataset label	Total Attack	Total Normal
10% KDD	396,744	97,277
Corrected (Test)	250,436	60,593
Whole KDD (Test)	3,925,651	972,780

Table 1. Basic Characteristics of the KDD dataset

Dataset label	Total Attack	Total Normal
10% KDD	396,744	97,277
10% KDD (Normal only)	0	97,277
50/50 Normal/Attack	97,277	97,277

Table 2. Basic Characteristics of the three training datasets Employed

4.2 Training

Learning parameters for the SOMs are summarized in Table 3, where this process is repeated for each SOM comprising the hierarchy. In each case, training is completed in two stages, the first providing for the general organization of the SOM and the second for the fine-tuning of neurons. Table 4 summarizes the additional parameters utilized by the shift register and Potential Function clustering algorithm. The resulting SOM hierarchy consists of 6 SOM networks in the first level (temporal encoding), each consisting of 6×6 grid and 20 inputs. Potential Function clustering 'quantizes' each original first level SOM to six neurons using the process described in Section 3.2, resulting in 36 inputs to the second level (integration) SOM. Once training of the second level is complete, labeling takes place. That is, for each connection in the training set, the corresponding label is given to the best matching unit in the second level. A count is kept for the number of normal and attack connections each best matching unit receives. If a second level neuron receives more counts from the attack connections in the training set, the neuron is labeled as attack. Similarly, if a second level neuron receives more counts from the normal connections in training set, that neuron is labeled as normal. A new connection is classified according to its best matching unit on the second level SOM.

4.3 Evaluation

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

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

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

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

where False Positive (Negative) Rate is the number of Normal (Attack) connections labeled as Attack (Normal). Details of the evaluation results based on the three different training data sets are given below.

Parameter	Rough Training	Fine Tuning
Initial α	0.5	0.05
α decay scheme	$f(\text{epoch}^{-1})$	
Epoch Limit	4,000	
Neighborhood Parameters		
Initial Size	2	1
Function	Gaussian	
Relation	Hexagonal	

Table 3. SOM Training Parameters

		Potential Function (For each feature respectively)
10% KDD	α	$1\text{x}10^{-6}$, $1\text{x}10^{-3}$, $2\text{x}10^{-7}$, $1\text{x}10^{-6}$, $16\text{x}10^{-7}$, 0.1599015
	β	$2\text{x}10^{-2}$, $2\text{x}10^{-2}$, $1\text{x}10^{-2}$, $4\text{x}10^{-2}$, $1\text{x}10^{-1}$, $1\text{x}10^{-2}$
10%KDD (Normal)	α	$5\text{x}10^{-7}$, $1\text{x}10^{-5}$, $1\text{x}10^{-5}$, $1\text{x}10^{-5}$, $33\text{x}10^{-7}$, $6\text{x}10^{-7}$
	β	$9\text{x}10^{-1}$, $13\text{x}10^{-2}$, $7\text{x}10^{-2}$, $13\text{x}10^{-2}$, $1\text{x}10^{-2}$, $1\text{x}10^{-2}$
50/50	α	$9\text{x}10^{-6}$, $1\text{x}10^{-5}$, $1\text{x}10^{-7}$, $1\text{x}10^{-5}$, $175\text{x}10^{-7}$, $7\text{x}10^{-7}$
	β	$2\text{x}10^{-1}$, $15\text{x}10^{-2}$, $7\text{x}10^{-2}$, $13\text{x}10^{-2}$, $1\text{x}10^{-2}$, $1\text{x}10^{-2}$

Shift Register	
Length	96
# Taps (k = 5)	20

Table 4. Potential Function and Shift Register Parameters

4.4 System 1 – Training with 10% KDD

The Hit histogram in Figure 1 summarizes the count of attack and normal connections in the second level SOM for 10% KDD, where proportionally larger counts result a greater area of the hexagon being colored. In hit histograms, neurons are numbered starting from the upper left corner, column by column (i.e. upper left corner neuron is 1 with numbering continuing sequentially top-down). It is apparent that nodes 1, 32 and 36 account for most of the attack connections and neuron 19 most of the normal connections. It is also apparent that several neurons also respond to both normal and attack connections. Table 5 details test set performance for SOM hierarchy, which is trained on the first training set, on the 'Corrected (Test)' KDD test set.

KDD Corrected (Test)	
FP rate	Detection rate
7.6 %	90.6 %

Table 5. Test Set Results for the SOM trained on the 10% KDD

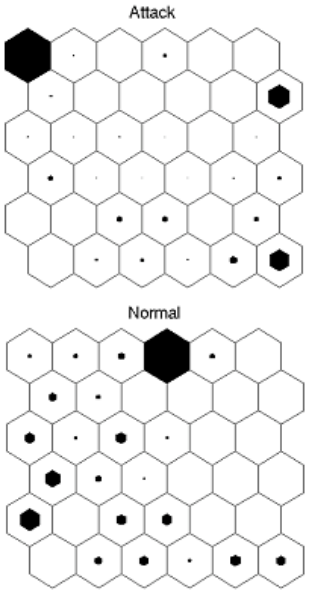

Figure 1. Hit histogram of the second level SOM for training with 10% KDD

4.5 System 2 – Training with Normal Only Connections of 10% KDD

In the second system, training is only conducted with those connections from 10% KDD labeled as Normal, resulting in an anomaly detector. Once trained, labeling of second level neurons commences using all the 10% KDD data set. Figure 2 summarizes the SOM at the second level. It is apparent that neurons 1, 5, 16, 18, 35 and 36 account for most of the normal connections although some of them (16 and 36) are also excited by attack connections. Neurons 16, 18 and 36 account for attacks. Moreover, 21 neurons account for normal connections only whereas the remaining 12 neurons do not account for any connection. Table 6 details the test set performance on the 'Corrected (Test)' set.

Corrected (Test)	
FP rate	Detection rate
14.5%	91.5%

Table 6. Test Set Results for Normal Only

On comparing performance of systems 1 and 2 (attack dominant verses no attack), detection rates are essentially unchanged, however, the FP rate is approximately 7% better in system 1. An FP rate of 14.5% is not acceptable for an intrusion detection system in practice.

4.6 System 3 – Training with 10% KDD Modified – 50% Attack / 50% Normal Connections

As indicated in section 4.1, the 10% KDD training dataset contains more attacks than normal traffic. In this last case, the training set is balanced by using all the 97,277 normal connections, and the first 97,277 attack connections from the 10% KDD. Second level SOM is labeled with the entire 10% KDD dataset. Figure 3 summarizes the labeled SOM for this data set at the second level. In this case neurons 1, 6, 14, 22, 27, 31 and 32 account for most of the attack connections, whereas 6 neurons do not account for any connection and the remaining neurons account for normal connections of the dataset. Although there seems to be a vague separation between attack and normal connections on the SOM – neurons at the upper part of the SOM account for attacks and neurons at the lower end of the SOM account for normal – the FP rate is still very high, Table 7. Thus, compared with the results for System 1 – Table 5 – balancing the dataset could not improve on the performance of the attack-based dataset. Whether this remains true if a 50/50 dataset were composed to ensure that all attack types, which appear in 10% KDD, also appear in the 50/50 dataset remains an open question.

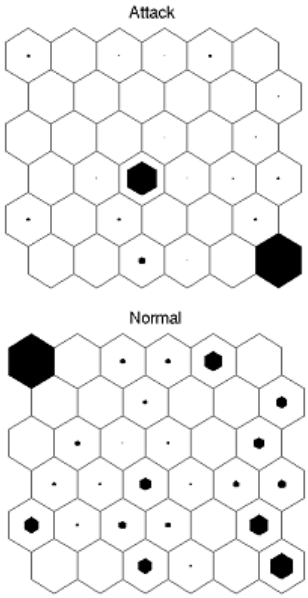

Figure 2. Hit histogram of the second level SOM for training with Normal Only

Corrected (Test)	
FP rate	Detection rate
14.3%	91.3%

Table 7. Test Set Results for training under a balanced data set

Table 8 details the test set performance results for the three systems discussed. All three systems perform well on denial of service and probe attacks. However performance of user to root and remote to local categories are low. Most of the remote to local and user to root attacks are content based which means the attack is manifested in the packet payload. Although 6 basic features employed in this work do not involve content inspection, some of the user to root and remote to local attacks could be detected indirectly. Moreover, performance on the 'Normal' connection category represented the deciding factor in the overall (false positive and detection) performance of the three systems.

	Normal	DoS	Probe	U2R	R2L
System 1	92.4%	96.5%	72.8%	22.9%	11.3%
System 2	85.5%	96.5%	91.0%	22.9%	20.5%
System 3	85.7%	96.7%	79.7%	30.0%	18.4%

Table 8. Performance of the three systems on different categories

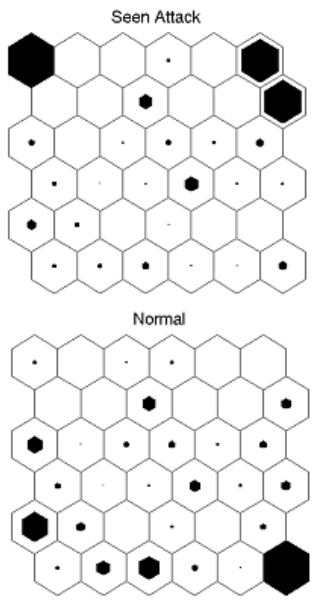

Figure 3. Hit histogram of the second level SOM for training under a balanced data set.

As indicated before, the test (Corrected) set contains an additional 14 new attacks, which are not available in the original 10% KDD training dataset. Performance of the three systems on Table 9 qualifies the ability of the three systems to detect novel attacks. Although Systems 2 and 3 produced significantly more false alarms compared with System 1 (Tables 5 and 7), their performance on unseen attacks is better than that of System 1.

Table 10 shows the performance of the three systems on test (Corrected) data, where the listed attacks are all available in the training (10% KDD) dataset. That is System 1 has seen all attacks of table 10 during training, System 2 was forbidden to use any during training, but could employ them during labeling, and System 3 received a percentage of them during training. All three systems performed well on denial of service attacks, which forms the majority of the test and training data.

Attack Name	Category	System 1	System 2	System 3
apache2.	dos	90.3%	29.2%	96.0%
mailbomb.	dos	7.8%	8.0%	8.0%
processtable.	dos	59.4%	77.2%	71.9%
Udpstorm.	dos	0.0%	0.0%	0.0%
mscan.	probe	90.2%	92.3%	91.7%
saint.	probe	79.1%	97.4%	89.1%
httptunnel.	r2l	58.9%	88.6%	71.5%
named.	r2l	23.5%	41.2%	35.3%
sendmail.	r2l	5.9%	29.4%	11.8%
snmpgetattack.	r2l	11.5%	23.0%	20.1%
xlock.	r2l	0.0%	11.1%	11.1%
xsnoop.	r2l	0.0%	0.0%	0.0%
Ps.	u2r	0.0%	0.0%	6.3%
xterm.	u2r	23.1%	30.8%	38.5%

Table 9. Detection rate of the "new attacks" that did not appear in the training set

Table 11 provides a summary of some recent results from alternative – unsupervised – data mining approaches trained on the KDD-99 dataset and tested using the 'Corrected (Test)' data [4, 5]. Detection rates are very similar to those reported for the SOM hierarchy constructed here. However, there are actually several additional factors with which these results need to be interpreted. Firstly, all the data mining approaches are based on all 41 features; the SOM hierarchy only utilizes 6. Secondly, in the case of [4], figures quoted are for a mixture of specific and multiple attack types, making it difficult to determine performance over the entire dataset. Thirdly, use was also made of Host based information for all data mining approaches, thus providing an advantage when detecting content based attacks [4].

Attack	Category	System 1	System 2	System 3
back.	dos	50.9%	7.2%	52.0%
land.	dos	100.0%	100.0%	100.0%
neptune.	dos	96.4%	97.4%	96.9%
pod.	dos	6.9%	60.9%	4.6%
smurf.	dos	99.9%	100.0%	99.8%
teardrop.	dos	16.7%	16.7%	25.0%
ipsweep.	probe	25.5%	83.0%	25.2%
nmap.	probe	40.5%	95.2%	85.7%
portsweep.	probe	52.0%	93.8%	84.7%
satan.	probe	73.7%	88.0%	76.6%
ftp_write.	r2l	0.0%	0.0%	33.3%
guess_passwd.	r2l	7.9%	16.7%	13.4%
imap.	r2l	0.0%	100.0%	100.0%
multihop.	r2l	0.0%	16.7%	0.0%
phf.	r2l	0.0%	50.0%	50.0%
snmpguess.	r2l	3.2%	5.5%	7.5%
warezmaster.	r2l	27.3%	34.1%	34.5%
worm.	r2l	50.0%	50.0%	50.0%
buffer_overflow.	u2r	13.6%	18.2%	22.7%
loadmodule.	u2r	0.0%	0.0%	0.0%
perl.	u2r	0.0%	0.0%	50.0%
rootkit.	u2r	69.2%	53.8%	61.5%
sqlattack.	u2r	50.0%	50.0%	50.0%

Table 10. Detection rate of the attacks that appeared in the training set

Technique	Detection Rate	FP Rate
Data-Mining [6]	70-90%	2%
Clustering [4]	93%	10%
K-NN [4]	91%	8%
SVM[1] [4]	98%	10%

Table 11. Recent Results on THE KDD benchmark

5 Conclusions and Future Work

A hierarchical SOM approach to the IDS problem is proposed and demonstrated on the International Knowledge Discovery and Data Mining Tools Competition intrusion detection benchmark [2] in which only the 6 basic features are utilized. Specific attention is given to the representation of connection sequence (time) and the hierarchical development of abstractions sufficient to permit direct labeling of SOM nodes with connection type. Other than these two concepts, no additional use of *a priori* information is employed. To understand the effect of

[1] The Support Vector Machine (SVM) algorithm is nominally a supervised learning algorithm. In this case however, only the unsupervised activity for forming what would normally be the hidden layer is employed.

training data set on the overall performance of the learning system for IDS, three different versions of KDD data set are employed.

In general all three systems provided similar performance on DoS attacks. However, System 2 – an anomaly detector – actually attained the best or second best classification accuracies across all four-attack types. However, given the very similar performance of all three detectors on the most frequent connection category (DoS), the 'normal' connection category came to represent the deciding factor in determining overall detector ranking. System 1 actually provided the best classification rate over 'normal' connections, an interesting result given that System 2 was never trained on anything other than 'normal' connections. Given the specific characteristics of each detector, it is conceivable that a combination of System 1 (for detection of normal category connections alone) and System 2 (for labeling of attack categories) would provide the best IDS.

In comparison to data mining approaches currently proposed, the two-layered SOM approach provides competitive performance whilst utilizing a fraction of the feature set (6 of the 9 "Basic features of an Individual TCP connection" and none of the 32 additional higher-level derived features). It is anticipated that future work will investigate the utilization of the third level SOMs for those second level neurons, which win for both attack and normal behavior. Third level SOMs can be trained with the network connections for which the corresponding second layer neuron is the best matching unit. This divide and conquer approach will increase the likelihood of separation between attack and normal connections in third level SOMs.

Acknowledgements

This work was supported in part by IRIS Emerging Opportunity, NSERC Discovery and CFI New Opportunity Grants from the Canadian Government. Further information regarding the work described in this publication is available through the project homepage of the NIMS research group http://www.cs.dal.ca/projectx/.

References

[1] T. Bass, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, 43(4), pp. 99-105, April 2000.

[2] S. Hettich, S.D. Bay, *The UCI KDD Archive*. Irvine, CA: University of California, Department of Information and Computer Science, http://kdd.ics.uci.edu, 1999.

[3] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, 3(4), pp. 262-294, 2001.

[4] 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, ISBN 1-4020-7054-3, 2002.

[5] U. Halici, G. Ongun, "Fingerprint Classification Through Self-Organizing Feature Maps Modified to Treat Uncertainties," Proceedings of the IEEE, 84(10), pp 1497-1512, 1996.

[6] W. Lee, S. Stolfo, K. Mok, "A data mining framework for building intrusion detection models," *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 120-132, 1999.

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

[8] *The 1998 intrusion detection off-line evaluation plan*. MIT Lincoln Lab., Information Systems Technology Group. http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt, 25 March 1998.

[9] *Knowledge discovery in databases DARPA archive*. Task Description. http://www.kdd.ics.uci.edu/databases/kddcup99/task.html

[10] *Bro user manual*, http://www.icir.org/vern/bro-manual/index.html

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

[12] S.L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent and Fuzzy Systems*, 2, pp. 267-278, 1994.

[13] *Software Packages from Helsinki University of Technology*, Laboratory of Computer and Information Science Neural Networks Research Centre, http://www.cis.hut.fi/research/software.shtml