Computer Networks 144 (2018) 111–119

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection

Tamer Aldwairi[a,*], Dilina Perera[a,b], Mark A. Novotny[a,b]

[a] Distributed Analytics and Security Institute, High Performance Computing Collaboratory, Mississippi State University, Starkville, MS 39762, United States
[b] Department of Physics and Astronomy, Mississippi State University, MS 39762, United States

ARTICLE INFO

Article history:
Received 28 December 2017
Revised 3 July 2018
Accepted 25 July 2018
Available online 3 August 2018

Keywords:
Restricted Boltzmann Machine
Anomaly Network Intrusion Detection Systems
NetFlow traffic
Cybersecurity
Machine learning
ISCX dataset

ABSTRACT

The continuous increase in the number of attacks on computer networks has raised serious concerns regarding the importance of establishing a methodology that can learn and adapt to new and novel attacks, such a model should be able to act or react to such threats within a timely manner, so that measures are undertaken to counter any potential breaches within the network. Training a model to distinguish between normal and anomalous network behavior is a difficult task due to the high dimensionality of the network traffic data. One of the key requirements of a successful Anomaly Network Intrusion Detection Systems (A-NIDS) is the ability to recognize new patterns of attacks that it has never before seen. This objective can be achieved through incorporating machine leaning techniques in the learning model of the A-NIDS. In this study, we demonstrate the use of a powerful machine learning technique called the Restricted Boltzmann Machine (RBM) to distinguish between normal and anomalous NetFlow traffic. We evaluate our approach through testing it on the newly renowned Information Security Center of Excellence (ISCX) dataset. Our results indicate that RBMs can be trained successfully to classify normal and anomalous NetFlow traffic. Unlike previous studies, we employ measures of true positives and negatives along with the accuracy to test the effectiveness of RBM as a classifier for A-NIDS. We also utilize the usage of a balanced set to reduce any biases that appear during the RBM training.

1. Introduction

Recent advancement in technologies governing the internet caused a rise in its usage in nearly every aspect of our daily life. As a consequence, the amount of data exchanged between parties over different channels has grown exponentially and caused an escalation in the volumes of traffic flowing through networks all over the world. One of the direct consequences of this growth is the increase in the quantity, magnitude, intensity, and variety of attacks across such networks. This has caused many institutions and organizations to incur great losses that could have been avoided if proper protection procedures had been taken. For this reason, securing networks from any attack or anomalous activity is a critical matter that has to be dealt with using immediate measures. Among the different measures and approaches used to secure networks are A-NIDS, which if implemented correctly, possess the ability to learn and adapt to continuous changes within the network environment. Therefore, building a reliable A-NIDS is considered a major goal for many organizations and institutions to secure their network.

A Network Intrusion Detection System can be defined as a network security software application designed to detect any potential breach within the network, through monitoring and analyzing the data gathered from different resources, for signs of any possible malicious activity based on incidents that are occurring within the network. This is done using a variety of measures that facilitate the detection of such incidents. An incident does not mandate that there is a malicious activity, in fact, lots of incidents could result through some inaccurate actions performed by authorized users of the network [1].

There are two known types of IDS, one being misuse-based IDS where the attack or abnormal behavior is previously defined. For example, the system compares the attack to a signature stored in its database where the database is continuously updated so it can deal with new attacks. The problem with misuse-based IDS is that they can only be as good as their database profiles since the IDS cannot detect novel attacks if the signature for that attack is not included in the database [2].

The other type of IDS is anomaly-based, where the system builds a profile through monitoring the system's normal activ-

* Corresponding author.
E-mail address: taldwairi@gmail.com (T. Aldwairi).

https://doi.org/10.1016/j.comnet.2018.07.025

ity, and then creates its own definition of what is considered normal behavior and anomalous behavior. Anomaly-based IDS (A-NIDS) usually are capable of detecting new unknown attacks that are not usually detected using misuse-based IDS [1,2]. A-NIDS can be further classified into three main categories: statistical-based, knowledge-based and machine learning-based [3]. In this paper, we are interested in exploring A-NIDS from the perspective of machine learning-based techniques.

The need for a model that effectively learns and adapts to new attacks is an important requirement in an intrusion detection system. Yet, such a model is faced with challenges pertaining to the unique characteristics of network security data that makes it different from a lot of other data types. Those challenges can be summed in two main points [4].

- The complexity of the network security data represented by different factors, including the high dimensionality of features.
- The continuous change of the data represented by the new and evolving patterns that need to be learned. That is why in order for the A-NIDS to be effective, it should not only be able to classify current network traffic correctly but should also have the ability to change and adapt to new unseen patterns of traffic and classify them correctly.

In this paper, we examine the suitability of RBM as a classifier for A-NIDS. We use NetFlow network traffic information to train a RBM on a balanced set extracted from the ISCX dataset. The rest of this paper is organized as follows. In Section 2, we discuss the previous work that utilized RBMs within the context of cyber security and IDS. In Section 3, we describe the internal mechanisms governing the RBM and the algorithms used to train it. In Section 4, we explain our choice for the dataset used, and the preprocessing steps used to prepare the data for RBM usage. In Section 5, we describe the procedure used for training and tuning the hyperparameters of the RBM. In Section 6, we evaluate the performance of the RBM using a new testing set. In Section 7, we present our conclusions.

2. Previous work and motivation

Many recent studies have shown that RBMs can be effectively trained using Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD) algorithms, and are able to recognize and differentiate between a large variety of patterns. Several of those studies led by Hinton [5–8], Bengio [9,10], Andrew Ng [11,12] and many others revealed that RBM was successful – along with its various forms (Discriminative RBM, Convolutional RBM, etc.) in distinguishing different patterns of visual images and speech sound waves. Nevertheless, only a few studies addressed the potential of using such a powerful energy-based model for cyber security applications and especially in securing networks from attacks through integrating its learning ability within an A-NIDS.

A recent study by Fiore et al. [4] demonstrated using Discriminative RBM (D-RBM) for the purpose of assessing its performance as a classifier for A-NIDS. They tested whether the D-RBM learning ability can be generalized by testing it on a different network than the one it was trained on. Their results indicate that the performance of D-RBM drops when used on a new network not previously seen. They recommend additional investigation of the basic differences between the nature of normal and anomalies traffic.

Gao et al. [13] focused on Big Data classification in the IDS context through analyzing its performance on Deep Belief Networks (DBN), using RBM in the unsupervised layers of DBN. Their experimental results on the KDD Cup99 dataset showed that the performance of DBN outperforms both SVM and ANN performance.

Alom et al. [14] explored the capabilities of DBN through training a sequence of RBMs and evaluating its performance against DBN-SVM on the NSL-KDD dataset. Salama et al. [15] investigated the use of DBN based on RBM for feature dimensionality reduction using the NSL-KDD dataset. After using RBM for feature reduction they used SVM for classification of the data and achieved a high classification accuracy.

Many of the studies mentioned above dealt with some important aspects and presented some promising results concerning the usage of a RBM. With that being said, some of those studies suffer from problems that can affect the reliability of the conclusions drawn from their results.

The first and foremost problem is the usage of the KDD99 dataset [16] for evaluation of the A-NIDS. The dataset is well known to have fundamental flaws in the way it was constructed and is not reflective of the real world network traffic. Even though some studies got around this problem through using the NSL-KDD [17], which is basically a new dataset derived from the KDD99 dataset that solves many of the essential problems found in the KDD99 dataset, the NSL-KDD dataset still lacks some of the properties found in a real non-simulated dataset [17].

The second problem is the usage of a single measure to evaluate the RBM performance. Many of the studies mentioned above evaluate the performance of the RBM solely on the basis of the accuracy achieved during the classification procedure. This evaluation approach cannot be considered sufficient to prove the effectiveness of RBM since it does not take into account the percentage of true positives and true negatives when measuring RBM's classification accuracy. Neglecting those two measures would eventually lead to incorrect conclusions concerning the validity of the RBM to act as a classifier for A-NIDS. A high accuracy can still be achieved even if the learning model used for the classification task was performing poorly. For example if the ratio of normal to attacks was 10:1, a high classification accuracy could be the result of having a very high number of true negatives (i.e., correctly identifying normal traffic most of the time) while having a very low number true positives (i.e., incorrectly identifying anomalous traffic most of the time).

To solve the aforementioned problems we did the following: we used the ISCX dataset [18] which we will discuss in detail in Section 4.1. We also incorporated in our analysis the true positive rate and the true negative rate as additional measures, along with the classification accuracy in order to get a more accurate assessment of RBMs true performance. Furthermore, to ensure proper training of the RBM, we extracted a balanced set from the ISCX dataset. Throughout this study when referring to a balanced dataset, we are referring to a dataset with an equal number of normal/anomalous records (i.e., a 1:1 ratio). The balanced dataset aids in the reduction of the bias that appears in some cases, when the training results indicate that the classifier appears to have learned how to distinguish between different normal/anomalous records, but in reality the classifier learned how to distinguish only one type of record (i.e., normal or anomalous); the one with the most number of records.

Using a balanced dataset along with a high variety of normal/anomalous records ensures a more effective training of the learning model, and can have a direct effect on the levels of accuracy, true positives and true negatives achieved. In addition, the wider variety of normal/anomalous records can make it easier to generalize the model, so it can be tested on other datasets without affecting the reliability of the results.

3. Background

3.1. Boltzmann Machine

A Boltzmann Machine (BM) [19] is a bidirectionally-connected network of stochastic processing units. Each node of a BM can

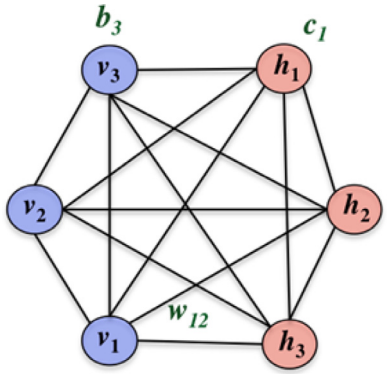


Fig. 3.1. A graphical representation of a general, fully-connected Boltzmann machine, with three hidden and three visible nodes.

be categorized as either *visible* or *hidden*. Visible nodes represent components of an observation. For instance, in an image classification system, each visible node may represent a single pixel of a digital image. Hidden nodes capture dependencies between visible nodes that cannot be modeled via direct pairwise interactions between visible nodes (Fig. 3.1).

For a BM consisting of m visible nodes $v = (v_1, v_2, \ldots, v_m)$ and n hidden nodes $h = (h_1, h_2, \ldots, h_n)$, with each $v_i \in \{0, 1\}$ and $h_j \in \{0, 1\}$, the energy function is given by [2]

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{v}^T\hat{\boldsymbol{W}}\boldsymbol{v} - \frac{1}{2}h^T\bar{W}h - v^T\boldsymbol{W}\boldsymbol{h} - b^T\boldsymbol{v} - \boldsymbol{c}^T\boldsymbol{h}, \tag{3.1}$$

where $\boldsymbol{\theta} = \{\widehat{W}, \bar{W}, W, b, c\}$ are the model parameters, with $\widehat{W}, \bar{W}, W$, respectively, representing visible-to-visible, hidden-to-hidden, and visible-to-hidden weights, while b, c, respectively, represent the bias fields acting on visible and hidden nodes. The probability that the model assigns to a particular visible vector v takes the form

$$p(\boldsymbol{v}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\sum_{h} e^{-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})}, \tag{3.2}$$

where $Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})}$ is the *partition function* [20]. The conditional probabilities of hidden and visible nodes are given by

$$p(h_j = 1 | \boldsymbol{v}, \boldsymbol{h}_{-j}) = \sigma\left(\sum_{i=1}^{m} W_{ij}v_i + \sum_{k=1\setminus j}^{n} \bar{W}_{jk}h_k + c_j\right), \tag{3.3}$$

and

$$p(v_i = 1 | \boldsymbol{h}, v_{-i}) = \sigma\left(\sum_{j=1}^{n} W_{ij}h_j + \sum_{k=1\setminus i}^{m} \hat{W}_{ik}v_k + b_i\right), \tag{3.4}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

3.2. Training Boltzmann Machines

Training is the process in which the weights and biases of a Boltzmann Machine are iteratively adjusted such that its marginal probability distribution $p(v; \theta)$ fits the training data as well as possible. A standard way of achieving this is to maximize the *likelihood* or the probability of the training data set S given the model parameters θ [21–25].

Since it is impracticable to maximize the log-likelihood analytically, it is typically achieved via the method of *gradient descent* on the log-likelihood.

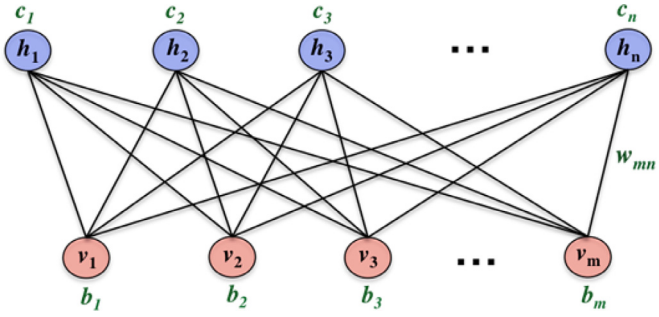


Fig. 3.2. A graphical representation of a restricted Boltzmann machine (RBM) with m visible and n hidden nodes.

3.3. Gradient of the log-likelihood

The most computationally intensive step of the gradient descent method is the calculation of the derivative of the log-likelihood during each iteration of the algorithm. The log-likelihood for a single training vector v is given by

$$\begin{aligned}\log \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v}) &= \log p(\boldsymbol{v}|\boldsymbol{\theta}) = \log\frac{1}{Z(\boldsymbol{\theta})}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})} \\ &= \log\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})} - \log\sum_{v,h} e^{-E(v,h;\theta)}\end{aligned} \tag{3.8}$$

One can show that the gradient of the log-likelihood then takes the form

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{heta}}\left(\log \mathcal{L}(\boldsymbol{\theta}|v)\right) &= -\sum_{h} p(h|v)\frac{\partial E(v, h; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \sum_{v,h} p(v, h)\frac{\partial E(v, h; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\ &= -\left\langle \frac{\partial E(v, h; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle_{\text{data}} + \left\langle \frac{\partial E(v, h; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle_{\text{model}}\end{aligned} \tag{3.9}$$

Exact computation of data dependent expectation and model's expectation takes a time that is exponential in the number of visible and hidden nodes. Therefore, in practice, these expectations are estimated by drawing samples from the corresponding probability distributions using Markov Chain Monte Carlo methods [26].

Gibbs sampling can be readily applied for estimating the data-dependent and model-dependent expectations in BM. However, depending on the complexity of the free energy landscape, the Markov chain may not necessarily converge to the true joint probability distribution of the BM (i.e., the Boltzmann distribution), which would lead to poor estimates of the data- and model-dependent expectations.

3.4. Restricted Boltzmann Machines (RBM)

The learning process for a general BM with unconstrained connectivity is too slow to be utilized in real world applications. However, the learning procedure can be made efficient enough for practical applications if certain limitations are imposed on the node connectivity. A class of limited-connectivity-BMs that has recently gained wide spread attention are Restricted Boltzmann Machines (RBM) [21,22,24]. In an RBM, there are no visible-to-visible or hidden-to-hidden connections. That is, each hidden node is *only* connected to visible nodes and vice versa (see Fig. 3.2).

Due to the limited connectivity of nodes in an RBM, the energy function given in Eq. (3.1) reduces to

$$\begin{aligned}E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) &= -\boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h} - \boldsymbol{b}^T v - c^T h = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}v_i h_j \\ &\quad - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j,\end{aligned} \tag{3.10}$$

and the conditional probabilities $\boldsymbol{p(h_j = 1|v)}$ and $\boldsymbol{p(v_i = 1|h)}$, respectively, reduce to

$$p\left(h_j = 1|\boldsymbol{v}\right) = \sigma\left(\sum_{i=1}^{m} W_{ij}v_i + c_j\right), \tag{3.11}$$

and

$$p(v_i = 1|\boldsymbol{h}) = \sigma\left(\sum_{j=1}^{n} W_{ij}h_j + b_i\right). \tag{3.12}$$

Moreover, since the hidden nodes are independent given the state of visible nodes and vice versa, the conditional probability of $\boldsymbol{h}$ given $\boldsymbol{v}$ becomes

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j=1}^{n} p(h_j|v), \tag{3.13}$$

and the conditional probability of v given h takes the form

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{m} p(v_i|\boldsymbol{h}). \tag{3.14}$$

The conditional independence of the nodes of the same type simplifies Gibbs sampling in an RBM. Instead of having to sample all the nodes sequentially, nodes of the same type can be sampled simultaneously. Thus, each iteration of the Gibbs sampling algorithm consists of two steps:

(1) Sample a new hidden vector $h^{(t)}$ based on the conditional probability $p(h|\boldsymbol{v}^{(t-1)})$.
(2) Sample a new visible vector $v^{(t)}$ based on the conditional probability $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$.

3.5. Gradient of the log-likelihood for an RBM

Factorization of the conditional probabilities, as presented in Eqs. (3.13) and (3.14), simplifies the data-dependent and model-dependent expectations in an RBM. For instance, it can be shown that the data-dependent expectation with respect to the visible-to-hidden weight $\boldsymbol{w_{ij}}$ reduces to

$$\left\langle \frac{\partial E\left(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}\right)}{\partial w_{ij}} \right\rangle_{\text{data}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E\left(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}\right)}{\partial w_{ij}} = -\, p(h_i = 1|\boldsymbol{v})v_j, \tag{3.15}$$

and the model's expectation becomes

$$\begin{aligned}\left\langle \frac{\partial E\left(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}\right)}{\partial w_{ij}} \right\rangle_{\text{model}} &= \sum_{v,h} p(v, h) \frac{\partial E(v, \boldsymbol{h}; \theta)}{\partial w_{ij}} \\ &= -\sum_{v} p(v)p(h_i = 1|v)v_j. \end{aligned} \tag{3.16}$$

Thus, the derivative of the log-likelihood with respect to $\boldsymbol{w_{ij}}$ takes the form,

$$\frac{\partial}{\partial w_{ij}} \log \mathcal{L}\left(\boldsymbol{\theta}|\boldsymbol{v}\right) = p(h_i = 1|\boldsymbol{v})v_j - \sum_{v} p(v)p(h_i = 1|\boldsymbol{v})v_j. \tag{3.17}$$

Similarly, the log-likelihood derivatives for the bias b_i of the ith visible node and the bias c_j of the jth hidden node are, respectively, given by

$$\frac{\partial}{\partial b_i} \log \mathcal{L}\left(\boldsymbol{\theta}|\boldsymbol{v}\right) = v_i - \sum_{v} p(\boldsymbol{v})v_i, \tag{3.18}$$

and

$$\frac{\partial}{\partial c_j} \log \mathcal{L}\left(\boldsymbol{\theta}|\boldsymbol{v}\right) = p\left(h_j = 1|\boldsymbol{v}\right) - \sum_{v} p(\boldsymbol{v})p\left(h_j = 1|\boldsymbol{v}\right). \tag{3.19}$$

Although the data-dependent expectations in Eqs. (3.17), (3.18) and (3.19) can be calculated exactly, exact computation of the model's expectation is still intractable as it requires summing over all possible visible vectors. Therefore, one has to again rely on Gibbs sampling [27] to obtain an estimate of the model's expectation.

3.6. Contrastive divergence (CD) algorithm for estimating log-likelihood gradient

Contrastive divergence (CD) [28] is an efficient algorithm for estimating the log-likelihood derivatives of an RBM using Gibbs sampling. In CD learning, a Markov chain is run only for k steps (typically, $k = 1, 2, \ldots, 10$). The Gibbs chain is initialized with one of the training vectors, $\boldsymbol{v}^{(0)}$. During each Gibbs sampling step t, $\boldsymbol{h}^{(t)}$ is sampled from the conditional probability $p(\boldsymbol{h}|\boldsymbol{v}^{(t-1)})$, followed by the sampling of $\boldsymbol{v}^{(t)}$ based on $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$. k steps of Gibbs sampling yield $\boldsymbol{v}^{(k)}$ and $h^{(k)}$, which are subsequently used for estimating the log-likelihood derivatives.

3.7. Persistent contrastive divergence (PCD) algorithm

Persistent contrastive divergence (PCD) [29] is a direct derivative of the CD algorithm. The PCD algorithm is built upon the assumption that if the learning rate is sufficiently small, model parameters change only slightly, and hence the Markov chains remain close to the stationary Boltzmann distribution between subsequent updates. Therefore, one could keep "persistent" Markov chains, without reinitializing them with a training vector $v^{(0)}$ after each parameter update, as we do in the CD algorithm. This would allow the Markov chains to approach thermal equilibrium faster, which in turn, would make the training process more efficient.

4. The ISCX dataset

One of the challenges faced by researchers during the evaluation of A-NIDS is finding a suitable dataset. Acquiring a real world dataset that represents the traffic flowing through the network without any sort of anonymization or modification is a problem that has been continuously encountered by the cybersecurity research community [17,18,30–33].

This challenge stems from the fact that most companies and institutions who are using or governing the internet will not allow data to be recorded, monitored or shared. This is mainly due to the laws and regulations related to privacy and confidentiality. Even in the cases where the data is allowed to be released or shared for public use, it will be heavily anonymized or severely altered leaving the researcher to deal with data that have significant portions removed, restricted, or changed. This will cause a lot of the essential data components that are considered critical to the researchers to be lost, and any statistical inferences based on the data may no longer be reliable or of real value.

For this reason, many researchers have decided to use simulated data sets like the well-known KDD99 dataset [16], or one of its contemporaries the NSL-KDD [17] or the gureKDDcup [34]. Recently there has been a significant effort to try and develop data sets that are reflective of real world data. One of those known successful attempts to create a dataset was the ISCX dataset [18].

4.1. ISCX dataset description

The ISCX dataset was generated using real network settings by capturing packets in real time throughout a period of seven days. The data contains approximately 85 GB of network traffic data along with profiles that describe the flow of the data and the attacks that occurred during that week. The ISCX dataset is characterized by the fact that it was gathered in real time and that the

Table 1
Ratio of attack to normal traffic for different days of the week.

Day	Description	Size (GB)	Flows	Normal	Attack	Ratio attack/normal
Friday	Normal ActivityNo malicious activity	16.1	378,667	378,667	0	0.0000
Saturday	Normal Activity some malicious activity	4.22	133,193	133,111	2082	0.0156
Sunday	Infiltrating the network from inside + Normal Activity	3.95	275,528	275,170	20,358	0.0740
Monday	HTTP Denial of Service + Normal Activity	6.85	171,380	167,609	3771	0.0225
Tuesday	Distributed Denial of Service using an IRC Botnet	23.4	571,698	534,320	37,378	0.0700
Wednesday	Normal Activity No malicious activity	17.6	522,263	522,263	0	0.0000
Thursday	Brute Force SSH + Normal Activity	12.3	397,595	392,392	5203	0.0133

attacks were not simulated, but instead the attacks were initiated during the process of capturing and recording the packets flowing within the network. We choose the ISCX dataset in our analysis for the following reasons [18].

- It represents a realistic dataset without any post-capture trace insertions on the data, thereby providing the researcher with a more realistic measure of the effects of certain attacks on the network.
- The dataset has profiles that describe the attacks along with some supplementary information that describes the time and approach used to initiate the attack. With some preprocessing of the profiles, a researcher can extract labeled flows that can be used as the basis or ground truth for evaluating the performance of any model without worrying about the accuracy of the labeling approach. This can assist the researcher in avoiding mistakes that may take place when using manual or automated labeling methods.
- The dataset recorded all the network interactions, which provides the researcher with a more accurate means to evaluate and interpret the results.
- Other benefits of the dataset include a complete capture of all network interactions within the network without removing or anonymizing any parts of the payload or the data.
- In addition, the ISCX dataset provides diverse intrusion scenarios for different days.

4.2. The ISCX dataset prepossessing

Using the ISCX profiles, we extracted 16 features that describe various characteristics of the NetFlow traffic along with normal/attack labels for each record. It is important to pre-process and convert those features into a binary form that can be used as a suitable input for the RBM during the different stages of training, validation, and testing.

Due to the probabilistic nature of the RBM, we need to make sure that the RBM is trained in a way that maximizes the learning of the different network traffic patterns. To achieve this, we need a dataset that incorporates a balanced ratio of normal and anomalous records. Training with an unbalanced data set (i.e., high percentage of normal traffic and a low percentage of attacks) can result in the RBM learning normal patterns of traffic and achieving a high detection accuracy for normal traffic, but a low detection accuracy for anomalous traffic. Another issue that we need to be aware of is that a high percentage of accuracy does not necessarily mean that the algorithm has learned to correctly distinguish between normal and anomalous behavior. Careful examination of true positives and negatives is necessary to make sure that both types of patterns have been learned correctly.

4.2.1. Creating a balanced data set

The ratio of anomalous to normal recordes is a major issue that can signifigintly affect the RBM training and learning process. Table 1. below shows the ratio between attacks and normal traffic.We can clearly see that the number of attack records is quite low compared to normal records. This makes sense since attacks do not usually occur as frequent as normal traffic. This also explains why on some days no attack occurs at all. But this does not help the training process because for our training and learning to be effective the ratio between attacks and normal traffic has to be adquatly presented and somewhat evenly distributed across the dataset.

Table 2
An example of balanced 1 to 1 ratio attack/normal dataset.

Set type	No of normal records	No of anomalous records	Total
Training set	41,278	41,278	82,556
Validation set	13,757	13,757	27,514
Testing set	13,757	13,757	27,514

To create a balanced dataset of normal/anomalous records from the ISCX dataset. We need to include an equal number of normal and anomalous records from each day, this is espically true since there are differernt attack types for each day. This ensures variety in the records of normal/attack within each dataset, which guarantees proper training and leads to a better accuracy during the validation and testing phases. Then we divide the set into three separate datasets, 60% for training, 20% for validation, and 20% for testing. Table 2 below shows a dataset with a 1 to 1 ratio between normal and anomalous records. In a balanced dataset the number of records for both sets does not have to be equivalent, we only need to make sure that the ratio of normal to attack records including the differient variations of attacks are proberly presented to gauarentee that the training phase is effective. Such a requirement is not nessacerasry during the validation and testing phases.

4.2.2. Discretization

Since the data contains a mix of nominal and numeric attributes, it is important to discretize the continuous numeric attributes into separate bins using discretization before we convert the data into binary form. For this dataset, we choose to apply the supervised discretization filter implemented in Weka [35]. We used supervised discretization to make sure that the class label information is taken into consideration during the discretization process.

After discretization, the continuous numeric attributes were placed into separate bins. The ranges (i.e., width) of the bins are not necessarily equal or contain the same number of instances. The bins are numbered in ascending order starting from zero so that we can convert them in the next step into their binary equivalent. The nominal attributes were also subjected to a similar procedure, but without the need for discretization since they are innately divided into separate bins. Once converted into binary form, each record contained 109 bits, with the last bit representing the class label (normal/anomaly).

5. Training and evaluation

For the proper training of the RBM, we are mainly concerned about two important matters. The first is the type and the char-

acteristics of the data that is fed to the RBM. This was handled and explained in detail in the previous section. The second is the tuning of the RBM hyperparameters, which we will explain in the following subsection.

5.1. Hyperparameter tuning

There are different hyperparameter settings, and based on the nature of the application used and the characteristics of the data set, some hyperparameters might have a greater impact on learning than others. For this specific application, and when using RBM as our model, we are primarily concerned with the learning rate, the number of iterations, the batch size, and the k-value in the CD/PCD algorithm. To train the RBM, we used either the contrastive divergence (CD) or the persistent contrastive divergence (PCD) algorithm.

Finding the right values for the hyperparameters is not an easy task; it requires continuous experimentation on how changing those parameters affect the training of the RBM. This is done by evaluating the RBM performance on a validation set after each training phase, and continuously tuning the hyperparameters to achieve the best results. It is important to point out that if a model performs well during the training and validation phases, that does not necessarily imply that it will perform as well during the testing phase, or when using a new external dataset.

This issue can be attributed mainly to the problem of overfitting where the model adapts very closely to the training data, but it loses much of its predictive capability and performs poorly when it is tested on another set that it has not seen previously. To be able to generalize the results of the RBM training, and avoid the problem of overfitting, an additional testing phase should be performed on a new or a hold-out dataset other than the one used for hyperparameter tuning (i.e., the validation set).

To evaluate the performance of the RBM, we used three main measures: the accuracy, the true positive rate (TPR) known as sensitivity, and the true negative rate (TNR) known as specificity. We calculated each of those measures using the following formulas:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{5.1}$$

$$\text{TPR} = \frac{TP}{TP + FN} \tag{5.2}$$

$$\text{TNR} = \frac{TN}{TN + FP} \tag{5.3}$$

Here, the letters T and F stand for true and false, respectively, P and N stand for positive and negative, respectively, and R stands for rate. We point out that we have also calculated measures such as precision, negative predictive value, fall-out rate, fall-discovery rate, miss rate, F1 score, and the Matthews correlation coefficient, but their significance was not regarded essential to our evaluation process.

Before we started the tuning process, we performed some preliminary tests that allowed us to narrow our search space of possible hyperparameter values. By doing so, we thereby limit our search space to values that yield only practical results. This is a necessary step, since testing all the possible values would require an exhaustive search that is computationally expensive, and most likely intractable due to its exponentially large search space [36–38].

After we found some initial settings where the learning works well, we started the tuning process by modifying one hyperparameter at a time while keeping all the other settings constant, until we found the set of values for that specific hyperparameter that returned a satisfactory result.

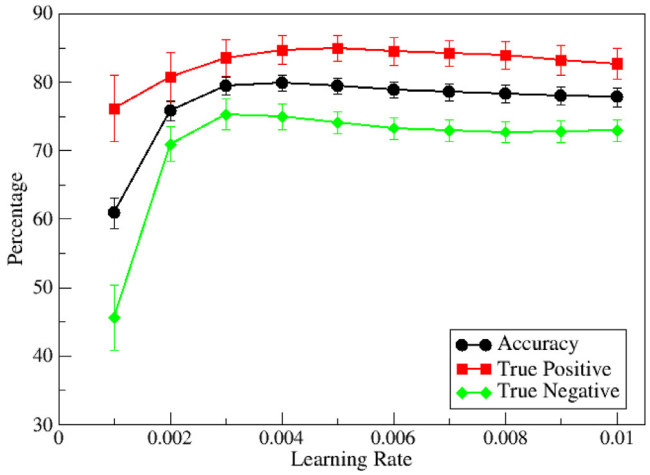


Fig. 5.1. Tuning the leaning rate hyperparameter for the training of RBM using CD-1, with 3500 iterations and a mini-batch size of 200.

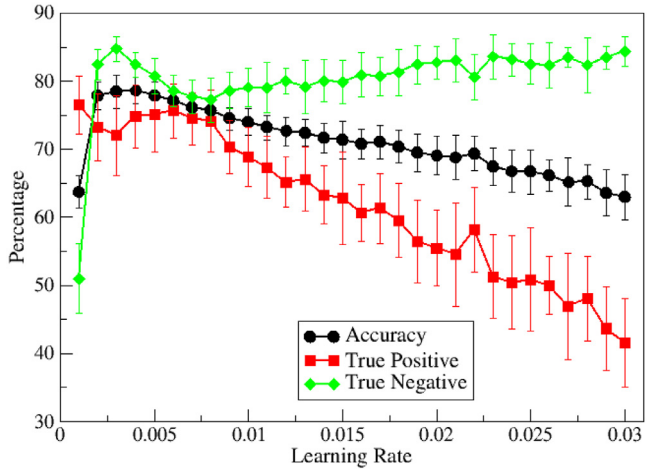


Fig. 5.2. Tuning the leaning rate hyperparameter for the training of RBM using PCD-1, with 4000 iterations and a mini-batch size of 200.

We started the tuning process by running the CD algorithm at different learning rates ranging from 0.001 to 0.01 using 3500 iterations, a mini-batch size of 200, mini-batch is defined as the number over which the weight changes are accumulated before fully updating the weights [39], and a k value of 1. Fig. 5.1 shows the accuracy, TPR, and TNR as functions of the learning rate. The error bars were estimated using 10 independent runs initiated from different random number seeds. We notice that the accuracy keeps increasing until it reaches its highest value of 79.8 ± 1.2%, at the learning of 0.004. The TPR and TNR on the other hand increase up to 0.005 and 0.003, respectively, and then stabilize with a small decrease and minor fluctuations up to 0.01.

We also ran the PCD algorithm at different learning rates ranging from 0.001 to 0.03, using 4000 iterations, a mini-batch size of 200, and a k value of 1. Fig. 5.2 shows the accuracy, TPR, and TNR as functions of the learning rate. The highest accuracy achieved was 78.7 ± 1.9%, when the learning rate was set to 0.004. The TPR and TNR at this learning rate are also significantly high; 74.9 ± 4.6% and 82.4 ± 1.8%, respectively.

After finding a learning rate value for both CD and PCD algorithms that produces a high accuracy, the next step is to determine the optimal number of iterations that provides the best performance. To do that, we ran the CD algorithm for a different number of iterations ranging from 500 to 5000, with a fixed learning

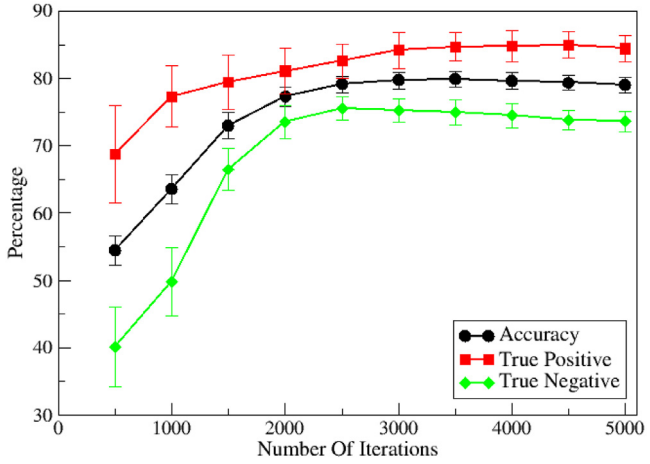


Fig. 5.3. Tuning the number of iterations hyperparameter for the training of RBM using CD-1, with the learning rate and mini-batch size held constant at of 0.004 and 200, respectively.

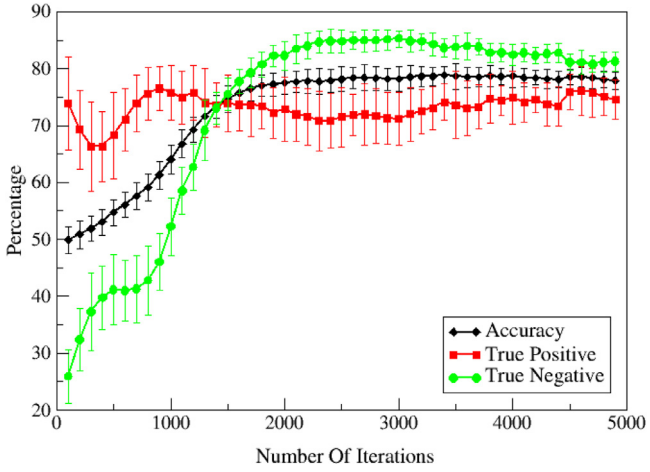


Fig. 5.4. Tuning the number of iterations hyperparameter for the training of RBM using PCD-1, with the learning rate and mini-batch size set to 0.004 and 200, respectively.

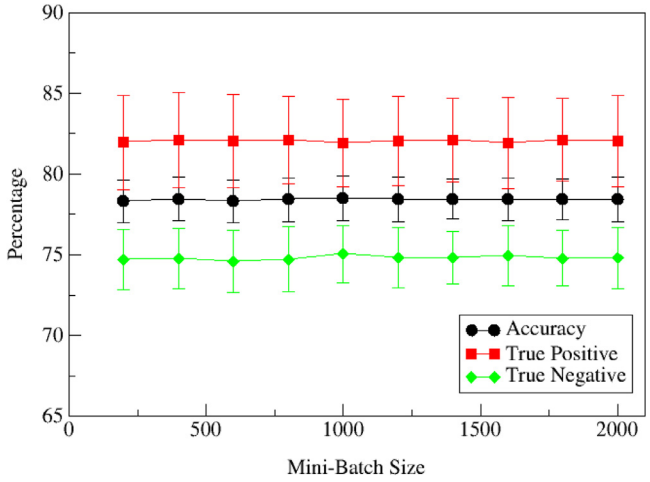


Fig. 5.5. The accuracy, TPR, and TNR as functions of the mini-batch for the CD algorithm at a learning rate of 0.003, 3000 iterations, and a k value of 1.

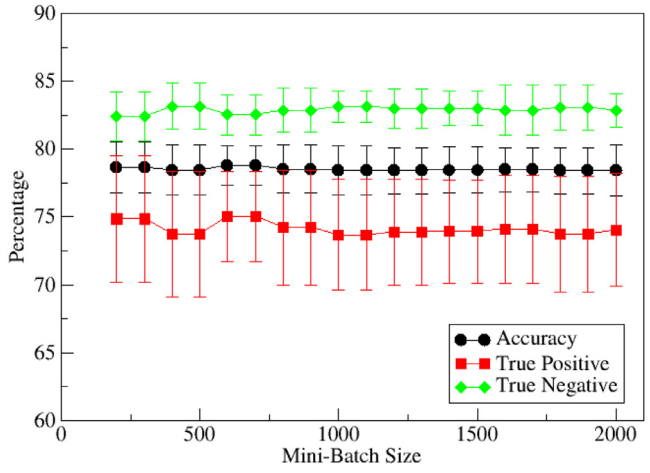


Fig. 5.6. The accuracy, TPR, and TNR as functions of the mini-batch size for the PCD algorithm at a learning rate of 0.004, 4000 iterations, and a k value of 1.

rate of 0.004, a mini-batch size of 200, and a k value of 1. Fig. 5.3 shows the accuracy, TPR and TNR as functions of the number of iterations used.

We notice that around 2500 iterations, the accuracy, sensitivity, and specificity start to stabilize, which can be associated with the convergence of the log-likelihood to its maximum value. The best value achieved for the accuracy was 79.8 ± 1.2% and it occurred at 3500 iterations. The TPR and TNR also reported significantly high values of 84.7 ± 2.1% and 75 ± 1.8%, respectively.

We also ran the PCD algorithm for a different number of iterations ranging from 100 to 5000, with a learning rate of 0.004, a mini-batch size of 200, and a k value of 1. Fig. 5.4 shows the accuracy, TPR, and TNR as functions of the number of iterations used. We notice that around 2000 iterations, the accuracy, sensitivity, and specificity start to stabilize. The best value achieved for the accuracy was 78.9 ± 1.5% and it occurred at 3400 iterations. The TPR and TNR at that point are also significantly high; 74.1 ± 3.3% and 83.6 ± 1.6%, respectively.

In the next step of our tuning process, we examined the effect of mini-batch size on the learning of the RBM. We ran the CD algorithm using a learning rate of 0.003 with 3000 iterations and a k value of 1 for different mini-batch sizes ranging from 20 records up to 2000 records (see Fig. 5.5). We also ran the PCD algorithm using a learning rate of 0.04 with 4000 iterations and a k value of 1 for different mini-batch sizes ranging from 200 to 2000 records (see Fig. 5.6). We found that changing the mini-batch size does not have a drastic effect on learning compared to changes in the learning rate or the number of iterations. Figs. 5.5 and 5.6 illustrate the fact that mini-batch size has a minor effect on the learning of the RBM for the ISCX dataset.

We also investigated the effect of increasing the value of k up to 10 by repeating the hyperparameter tuning process used previously. There was no obvious increase in the accuracy compared to what we obtained earlier, and most of the results were comparable to those we obtained using a k value of 1.

We continued the turning process for each hyperparameter by changing one hyperparameter at a time and observing the effect of that change on the other hyperparameters until we found the set of hyperparameters that provides us with the highest accuracy while maintaining high TPR and TNR values.

Eventually, we found that the highest accuracy of 79.8 ± 1.2% was achieved for CD at a learning rate of 0.003, 5000 iterations, a mini-batch size of 200, and a k-value of 1, the TPR and TNR also reported high values of 84.5 ± 2.3% and 75.2 ± 1.4%, respectively. For PCD, the highest accuracy of 78.9 ± 1.5% was achieved at a learning rate of 0.004, 3400 iterations, a mini-batch size of 200, and a

k value of 1, with the TPR and TNR reporting values of 74.1 ± 3.3% and 83.6 ± 1.6%, respectively.

There is no guarantee that the accuracies, TPR and TNR reached using the previous approach are the highest possible. To reach such a conclusion, an exhaustive search of all the possible parameter settings would be required, which takes a computationally long time to complete. Nevertheless, the high values achieved through the fine tuning process demonstrates that even a RBM with only one layer of hidden units is able to learn how to distinguish normal from anomalous traffic flow when properly trained using a balanced dataset of normal and anomalous records.

Following the process of hyperparameter tuning on the balanced dataset, we found the following:

- The learning rate, followed by the number of iterations has the most prevalent effect on learning.
- The mini-batch size has a very little to no effect on the RBM learning.
- Increasing the k value does not appear to affect the RBM learning in a drastic way.
- After reaching a certain number of iterations, the changes or fluctuations in the accuracy, true positive rate, and true negative rate become minor.

6. RBM performance on the ISCX test set

After training the RBM on a balanced dataset of normal and anomalous records, we tuned the RBM's hyperparameters using a separate validation set until we got to a point where only minor or no fluctuations in the accuracy were detected. We then evaluate our model on a separate new test set, not used before for either training or validation. The following hyperparameter values were chosen during the testing phase to evaluate the RBM's performance using CD and PCD algorithms. A learning rate of 0.003 was chosen for CD, and 0.004 was chosen for PCD. The number of iterations was set to 5000 for CD, and 3400 for PCD. The mini-batch size was 200 and a k-value of 1 was used in both cases. The results for CD and PCD were, respectively, as follows: 88.6 ± 0.6% and 89 ± 0.7% for the accuracy, 88.4 ± 0.8% and 84.2 ± 1.4% for the true positive rate, and 88.8 ± 1.1% and 93.8 ± 1% for the true negative rate. It is worth pointing out that it is usually harder to get a high rate of true positives when building a model for NIDS. The reason stems from the fact that there are so many variations of attacks, and novel ones appear every now and then. Our results on the testing set as well as our high TPR and TNR values during both validation and testing support our proposition that a RBM, when trained properly, can be used as a classifier for A-NIDS.

7. Conclusion

In this paper, we evaluated the suitability of a RBM as a machine learning model for A-NIDS. To test this, we used CD and PCD as the training algorithms. We also performed hyperparameter tuning of the RBM by changing one hyperparameter at a time and observing the effect of that change on the accuracy, true positive rate, and true negative rate. This was done throughout the turning process until we found optimal or semi-optimal settings for training the RBM. We used a balanced dataset extracted from the ISCX dataset to perform training, validation, and testing. A balanced dataset allows sufficient training for both normal and anomalous records. From our analysis, we found that the RBM is suitable for training A-NIDS, and can correctly distinguish between normal and anomalous behavior within a network, and identify novel patterns of attacks.

It is important to note that we have only explored the potential of two layer RBMs. A multilayer RBM, known as Deep Restricted Boltzmann Machine (DRBM), might even yield better results with higher accuracy, TPR, and TNR. However, the computational complexity associated with DRBM might discourage many researchers as well as some institutions from exploring its potential, especially if they are interested in real-time detection of cyber security threats. Nevertheless, recent advancements in high-performance computing along with the introduction of Adiabatic Quantum Computers (AQC) [40] may allow us to explore the true potential of DRBMs with real-time data in the near future.

Conflict of interest

None declared.

Acknowledgments

Funding for this work was partially provided by the Pacific Northwest National Laboratory (PNNL), under U.S. Department of Energy Contract DE-AC05-76RL01830. MAN was partially supported by the Air Force Research Laboratory (AFRL) under agreement number FA8750-18-1-0096. We would also like to thank the High Performance Computing Collaboratory (HPC^2) at Mississippi State University for providing access to the Shadow-II super computer. MAN is supported in part by a Fulbright Distinguished Chair grant. The views and conclusions herein are those of the authors, and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of AFRL, PNNL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

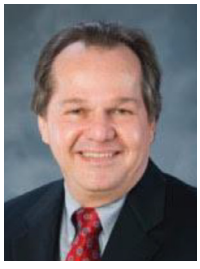
References

[1] K. Scarfone, P. Mell, Guide to intrusion detection and prevention systems (IDPS) recommendations of the national institute of standards and technology, Nist Spec. Publ. 800–94 (2007) 127.
[2] M. Albayati, B. Issac, Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system, Int. J. Comput. Intell. Syst. 8 (2015) 841–853, doi:10.1080/18756891.2015.1084705.
[3] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: techniques, systems and challenges, Comput. Secur. 28 (2009) 18–28, doi:10.1016/j.cose.2008.08.003.
[4] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, Network anomaly detection with the restricted Boltzmann machine, Neurocomputing 122 (2013) 13–23, doi:10.1016/j.neucom.2012.11.050.
[5] A. Krizhevsky, I. Sutskever, E.H. Geoffrey, Imagenet. Adv. Neural Inf. Process. Syst. 25 (2012) 1–9, doi:10.1109/5.726791.
[6] N. Jaitly, G.E. Hinton, Learning a better representation of speech sound waves using restricted Boltzmann machines, Acoust. Speech, Signal Process. 1 (2011) 1–4, doi:10.1109/ICASSP.2011.5947700.
[7] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the Twenty-seventh International Conference on Machine Learning, 2010, pp. 807–814. doi:10.1.1.165.6419.
[8] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proceedings of the Twenty-fourth International Conference on Machine Learning, 2007, pp. 791–798, doi:10.1145/1273496.1273596.
[9] H. Larochelle, Y. Bengio, Classification using discriminative restricted Boltzmann machines, Icml (2008) 536–543, doi:10.1145/1390156.1390224.
[10] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127, doi:10.1561/2200000006.
[11] A. Coates, A. Arbor, A.Y. Ng, An analysis of single-layer networks in unsupervised feature learning, Aistats 2011 (2011) 215–223, doi:10.1109/ICDAR.2011.95.
[12] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the Twenty-sixth Annual International Conference on Machine Learning. ICML 09. 2008, 2009, pp. 1–8, doi:10.1145/1553374.1553453.
[13] N. Gao, L. Gao, Q. Gao, H. Wang, An intrusion detection model based on deep belief networks, in: Proceedings of the 2014 Second International Conference on Advanced Cloud Big Data, 2014, pp. 247–252, doi:10.1109/CBD.2014.41.
[14] Alom, M.Z., Bontupalli, V., Taha, T.M.: Intrusion Detection using Deep Belief Networks. 339–344 (2016). doi:10.1109/NAECON.2015.7443094
[15] M.A. Salama, H.F. Eid, R.A. Ramadan, A. Darwish, in: Hybrid Intelligent Intrusion Detection Scheme, Springer, Berlin Heidelberg, 2011, pp. 293–303, doi:10.1007/978-3-642-20505-7_26.
[16] KDD Cup 1999 Data. (1999)

[17] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, NJ, USA, IEEE Press, Piscataway, 2009, pp. 53–58.
[18] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (2012) 357–374, doi:10.1016/j.cose.2011.12.012.
[19] D. Ackley, G. Hinton, T. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. 9 (1985) 147–169, doi:10.1016/S0364-0213(85)80012-4.
[20] L.D. Landau, E.M. Lifshitz, Statistical Physics, Part 1, Butterworth-Heinemann, 1980.
[21] A. Fischer, C. Igel, An introduction to restricted Boltzmann machines, Prog. Pattern Recognit. Image Anal. Comput. Vis. Appl. 7441 (2012) 14–36 Lecture Notes Computer Science, doi:10.1007/978-3-642-33275-3_2.
[22] A. Fischer, C. Igel, Training restricted Boltzmann machines: an introduction, Pattern Recognit. 47 (2014) 25–39, doi:10.1016/j.patcog.2013.05.025.
[23] R. Salakhutdinov, G. Hinton, Deep Boltzmann machines, Aistats 1 (2009) 448–455, doi:10.1109/CVPRW.2009.5206577.
[24] G. Hinton, A practical guide to training restricted Boltzmann machines, Computer 9 (2010) 1, doi:10.1007/978-3-642-35289-8_32.
[25] Brakel, P., Dieleman, S., Schrauwen, B.: Training restricted Boltzmann machines with multi-tempering: harnessing parallelization. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 92–99 (2012)
[26] P. Brémaud, Markov Chains, Springer New York, New York, NY, 1999.
[27] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1984) 721–741, doi:10.1109/TPAMI.1984.4767596.
[28] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (2002) 1771–1800, doi:10.1162/089976602760128018.
[29] T. Tieleman, Training restricted Boltzmann machines using approximations to the likelihood gradient, in: Proceeding of the Twenty-fifth International Conference on Machine Learning, 307, 2008, pp. 1064–1071, doi:10.1145/1390156.1390290.
[30] G. Creech, J. Hu, Generation of a new IDS test dataset: time to retire the KDD collection, in: Proceeding of the IEEE Wireless Communications and Networking Conference, WCNC, 2013, pp. 4487–4492.
[31] C. Kolias, G. Kambourakis, A. Stavrou, S. Gritzalis, Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset, IEEE Commun. Surv. Tutorials. 18 (2016) 184–208, doi:10.1109/COMST.2015.2402161.
[32] Sperotto, A., Sadre, R., Van Vliet, F., Pras, A.: A labeled data set for flow-based intrusion detection. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 39–50. Springer-Verlag, Berlin, Heidelberg (2009)
[33] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: Proceedings of the 2015 Military Communications and Information Systems Conference, 2015, pp. 1–6, doi:10.1109/MilCIS.2015.7348942.
[34] I. Perona, I. Gurrutxaga, O. Arbelaitz, J.I. Martin, J. Muguerza, J.M. Perez, Service-independent payload analysis to improve intrusion detection in network traffic, in: Proceedings of the Seventh Australasian Data Mining Conference, Australian Computer Society, Inc, 2008, pp. 171–178.
[35] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software, SIGKDD Explor. Newsl. 11 (2009) 10–18, doi:10.1145/1656274.1656278.
[36] L. Yu, H. Liu, Feature selection for high-dimensional data: a fast correlation-based filter solution, in: Proceedings of the International Conference on Machine Learning, 2003, pp. 1–8. doi:citeulike-article-id:3398512.
[37] M.A. Hall, L.A. Smith, Feature subset selection: a correlation based filter approach, in: Proceedings of Fourth International Conference on Neural Information Processing and Intelligent Information Systems, 1997, pp. 855–858.
[38] Y. Guo, D. Schuurmans, Discriminative batch mode active learning, Adv. Neural Inf. Process. Syst. 20 (2008) 593–600.
[39] D.R. Wilson, T.R. Martinez, The general inefficiency of batch training for gradient descent learning, Neural Netw. 16 (2003) 1429–1451, doi:10.1016/S0893-6080(03)00138-2.
[40] Y. Koshka, D. Perera, S. Hall, M.A. Novotny, Empirical investigation of the low temperature energy function of the restricted Boltzmann machine using a 1000 qubit D-wave 2X, in: Proceedings of the International Joint Conference on Neural Networks, 2016, pp. 1948–1954.

Tamer Aldwairi received his M.S. in computer science and Ph.D. in computational engineering from Mississippi State University, USA. He Worked as a Postdoc research associate at the Distributed Analytics and Security Institute in the High Performance Computing Collaboratory (HPC^2) at Mississippi State University. He is currently a visiting assistant professor at Ursinus College.

Dilina Perera is a visiting assistant professor at the Department of Physics and Astronomy, Mississippi State University. He received his Ph.D. in Physics from the University of Georgia in 2015. His research interests span a wide range of topics in Computational Condensed Matter and Statistical Physics, and Machine Learning.



Mark A. Novotny received his Ph.D. in Physics from Stanford University. He is currently Professor and Head of the Department of Physics and Astronomy at Mississippi State University, where he is a Giles Distinguished Professor. He is a Fellow of both the American Physical Society (APS) and AAAS.