Application of Machine Learning Techniques to Detecting Anomalies in Communication Networks: Classification Algorithms

Zhida Li, Qingye Ding, Soroush Haeri, and Ljiljana Trajković

Abstract In this chapter, we apply various machine learning techniques for classification of known network anomalies. The models are trained and tested on various collected datasets. With the advent of fast computing platforms, many neural network-based algorithms have proved useful in detecting BGP anomalies. Performance of classification algorithms depends on the selected features and their combinations. Various classification techniques and approaches are compared based on accuracy and F-Score.

Keywords Routing anomalies · Border Gateway Protocol · Classification algorithms · Machine learning techniques

1 Introduction

In Chapter "Application of Machine Learning Techniques to Detecting Anomalies in Communication Networks: Datasets and Feature Selection Algorithms", we have introduced Border Gateway Protocol (BGP) datasets used to detect anomalies, feature extractions, and various feature selection algorithms. We describe here machine learning classification techniques used to detect BGP anomalies. We examine the effect of feature selection on performance of BGP anomaly classifiers by evaluating performance of feature selection algorithms introduced in Chapter "Application of Machine Learning Techniques to Detecting Anomalies in Communication Networks: Datasets and Feature Selection Algorithms" in terms of classification accuracy and F-Score.

The readers are advised to first read "Chapter 3: Application of Machine Learning Techniques to Detecting Anomalies in Communication Networks: Datasets and Feature Selection Algorithms."

Z. Li · Q. Ding · S. Haeri · L. Trajković (✉)
Simon Fraser University, Vancouver, BC, Canada
e-mail: zhidal@sfu.ca; qingyed@sfu.ca; shaeri@sfu.ca; ljilja@cs.sfu.ca

A. Dehghantanha et al. (eds.), *Cyber Threat Intelligence*, Advances in Information Security 70, https://doi.org/10.1007/978-3-319-73951-9_4

We apply machine learning techniques to develop classification models for detecting BGP anomalies [6–10]. These models are trained and tested using various datasets that consist features extracted and selected in Chapter "Application of Machine Learning Techniques to Detecting Anomalies in Communication Networks: Datasets and Feature Selection Algorithms". They are used to evaluate the effectiveness of the extracted features. We report on improved classification results emanating from our previous studies (Support Vector Machine (SVM) two-way classification, Naive Bayes (NB) two-way and four-way classifications) and also implement the Long Short-Term Memory (LSTM) machine learning technique.

A survey of various methods, systems, and tools used for detecting network anomalies reviews a variety of existing approaches [12]. The authors have examined recent techniques to detect network anomalies and discussed detection strategy and employed datasets, including performance metrics for evaluating detection method and description of various datasets and their taxonomy. They also identified issues and challenges in developing new anomaly detection methods and systems.

Various machine learning techniques to detect cyber threats have been reported in the literature. One-class SVM classifier with a modified kernel function was employed [36] to detect anomalies in IP records. However, unlike the approach in our studies, the classifier is unable to indicate the specific type of anomalies. Stacked LSTM networks with several fully connected LSTM layers have been used for anomaly detection in time series [27]. The method was applied to medical electrocardiograms, a space shuttle, power demand, and multi-sensor engine data. The analyzed data contain both long-term and short-term temporal dependencies. Another example is the multi-scale LSTM that was used to detect BGP anomalies [14] using accuracy as a performance measure. In the preprocessing phase, data were compressed using various time scales. An optimal size of the sliding window was then used to determine time scale to achieve the best performance of the classifier. Multiple HMM classifiers were employed to detect Hypertext Transfer Protocol (HTTP) payload-based anomalies for various Web applications [8]. Authors first treated payload as a sequence of bytes and then extracted features using a sliding window to reduce the computational complexity. HMM classifiers were then combined to classify network intrusions. It was shown [29] that the naive Bayes classifier performs well for categorizing the Internet traffic emanating from various applications. Weighted ELM [39] deals with unbalanced data by assigning relatively larger weights to the input data arising from a minority class. Signature-based and statistics-based detection methods have been also proposed in [40].

This Chapter is organized as follows. We first introduce machine learning techniques and performance metrics in Sect. 2. Experimental procedures using various classification algorithms are described in Sects. 3–8. The advantages and shortcomings of various classification algorithms are offered in Sect. 9. We conclude with Sect. 10. The list of relevant references is provided at the end of the Chapter.

1.1 Machine Learning Techniques

Machine learning techniques have been employed to develop models for detecting and designing BGP anomaly detection systems. They are the most common approaches for classifying BGP anomalies.

Unsupervised machine learning models have been used to detect anomalies in networks with non-stationary traffic [16]. The one-class neighbor machines [31] and recursive kernel-based online anomaly detection [7] algorithms are effective methods for detecting anomalous network traffic [6].

While unsupervised learning techniques are often used for clustering, supervised learning is employed for anomaly classification when the input data are labeled based on various categories. Well-known supervised learning algorithms include Support Vector Machine (SVM) [11, 38], Long Short-Term Memory (LSTM) [20, 22], Hidden Markov Model (HMM) [11], naive Bayes [28], Decision Tree [33], and Extreme Learning Machine (ELM) [23, 24].

The SVM algorithms often achieve better performance compared to other machine learning algorithms albeit with high computational complexity. LSTM is trained using gradient-based learning algorithms implemented as a recurrent neural network. It outperforms other sequence learning algorithms because of its ability to learn from past experiences long-term dependencies.

No single learning algorithm performs the best for all given classification tasks [37]. Hence, an appropriate algorithm should be selected by evaluating its performance based on various parameters. Statistical methods, data mining, and machine learning have been employed to evaluate and compare various algorithms [17, 19].

2 Classification Algorithms

Classification aims to identify various classes in a dataset. Each element in the classification domain is called a class. A classifier labels the data points as either an anomaly or a regular event. We consider datasets of known network anomalies and test the classifiers' ability to reliably identify anomaly class, which usually contains fewer samples than the regular class in training and test datasets. Classifier models are usually trained using datasets containing limited number of anomalies and are then applied on a test dataset. Performance of a classification model depends on a model's ability to correctly predict classes. Classifiers are evaluated based on various metrics such as accuracy, F-Score, precision, and sensitivity.

Most classification algorithms minimize the number of incorrectly predicted class labels while ignoring the difference between types of misclassified labels by assuming that all misclassifications have equal costs. The assumption that all misclassification types are equally costly is inaccurate in many application domains. In the case of BGP anomaly detection, incorrectly classifying an anomalous sample may be more costly than incorrect classification of a regular sample. As a result, a

classifier that is trained using an unbalanced dataset may successfully classify the majority class with a good accuracy while it may be unable to accurately classify the minority class. A dataset is unbalanced when at least one class is represented by a smaller number of training samples compared to other classes. The Slammer and Code Red I anomaly datasets that have been used in this study are unbalanced. In our studies, out of 7,200 samples, Slammer and Code Red I contain 869 and 600 anomalous events, respectively. The Nimda dataset is more balanced containing 3,521 anomalous events. Hence, the majority of samples are regular data.

Various approaches have been proposed to achieve accurate classification results when dealing with unbalanced datasets. Examples include assigning a weight to each class or learning from one class (recognition-based) rather than two classes (discrimination-based) [13]. The weighted SVMs [38] assign distinct weights to data samples so that the training algorithm learns the decision surface according to the relative importance of data points in the training dataset. The fuzzy SVM [26], a version of weighted SVM, applies a fuzzy membership to each input sample and reformulates the SVM so that input points contribute differently to the learning decision surface. In this study, we create the balanced datasets by randomly reducing a portion of regular data points. Each balanced dataset contains the same number of regular and anomalous data samples.

2.1 Performance Metrics

The confusion matrix shown in Table 1 is used to evaluate performance of classification algorithms. True positive (TP) and False negative (FN) are the number of anomalous training data points that are classified as anomaly and regular, respectively. False positive (FP) and True negative (TN) are the number of regular data points that are classified as anomaly and regular, respectively.

Variety of performance measures are calculated to evaluate classification algorithms, such as accuracy and F-Score:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{F-Score} = 2 \times \frac{\text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}}, \tag{2}$$

Table 1 Confusion matrix

	Predicted class	
Actual class	Anomaly (positive)	Regular (negative)
Anomaly (positive)	TP	FN
Regular (negative)	FP	TN

where

$$\text{precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{sensitivity(recall)} = \frac{TP}{TP + FN}. \tag{4}$$

As a performance measure, accuracy reflects the true prediction over the entire dataset. It is commonly used in evaluating the classification performance. Accuracy assumes equal cost for misclassification and relatively uniform distributions of classes. It treats the regular data points to be as important as the anomalous points. Hence, it may be an inadequate measure when comparing performance of classifiers [32] and misleading in the case of unbalanced datasets. The F-Score, which considers the false predictions, is important for anomaly detection because it is a harmonic mean of the precision and sensitivity, which measure the discriminating ability of the classifier to identify classified and misclassified anomalies. Precision identifies true anomalies among all data points that are correctly classified as anomalies. Sensitivity measures the ability of the model to identify correctly predicted anomalies.

As an example, consider a dataset that contains 900 regular and 100 anomalous data points. If a classifier identifies these 1,000 data points as regular, its accuracy is 90%, which seems high at the first glance. However, no anomalous data point is correctly classified and, hence, the F-Score is zero. Therefore, the F-Score is often used to compare performance of classification models. It reflects the success of detecting anomalies rather than detecting either anomalies or regular data points. In this study, we use F-Score to compare various classification algorithms.

3 Support Vector Machine (SVM)

Support Vector Machine is a supervised learning algorithm used for classification and regression tasks. Given a set of labeled training samples, the SVM algorithm learns a classification hyperplane (decision boundary) by maximizing the minimum distance between data points belonging to various classes. There are two types of SVM models: hard-margin and soft-margin [15]. The hard-margin SVMs require that each data point is correctly classified while the soft-margin SVMs allow some data points to be misclassified. The hyperplane is acquired by minimizing the loss function [11]:

$$C \sum_{n=1}^{N} \zeta_n + \frac{1}{2} ||w||^2, \tag{5}$$

with constraints: $t_n y(x_n) \geqslant 1 - \zeta_n,\ n = 1, \ldots, N,$

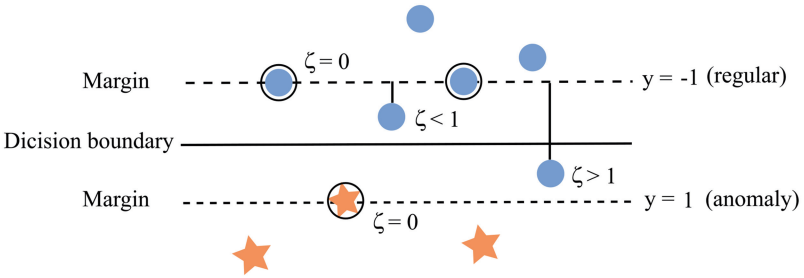

Fig. 1 Illustration of the soft margin SVM [11]. Shown are correctly and incorrectly classified data points. Regular and anomalous data points are denoted by circles and stars, respectively. The circled points are support vectors

where the regularization parameter C controls the trade-off between the slack variable ζ_n, N is the number of data points, and $\frac{1}{2}||w||^2$ is the margin. The regularization parameter $C > 0$ is used to avoid over-fitting problem. The target value is denoted by t_n while $y(x_n)$ and x_n are the training model and data points, respectively. The SVM solves a loss function as an optimization problem (5).

An illustration of the soft margin is shown in Fig. 1 [11]. The solid line indicates the decision boundary while dashed lines indicate the margins. Encircled data points are support vectors. The maximum margin is the perpendicular distance between the decision boundary and the closest support vectors. Data points for which $\zeta = 0$ are correctly classified and are either on the margin or on the correct side of the margin. Data points for which $0 \leq \zeta < 1$ are also correctly classified because they lie inside the margin and on the correct side of the decision boundary. Data points for which $\zeta > 1$ lie on the wrong side of the decision boundary and are misclassified. The outputs 1 and -1 correspond to anomaly and regular data points, respectively. The SVM solution maximizes the margin between the data points and the decision boundary. Data points that have the minimum distance to the decision boundary are called support vectors.

The SVM employs a kernel function to compute a nonlinear separable function to map the feature space into a linear space. The Radial Basis Function (RBF) was chosen because it creates a large function space and outperforms other types of SVM kernels [11]. The RBF kernel k is used to avoid the high dimension of the feature matrix:

$$k(\mathbf{x}, \mathbf{x}') = exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2). \tag{6}$$

It depends on the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ feature vectors. The datasets are trained using tenfold cross validation to select parameters $(C, 1/2\sigma^2)$ that give the best accuracy.

The SVM algorithm is applied to datasets listed in Table 2. We experiment with the MATLAB SVM functions and the SVM^{light} [4] library modules to train and

Table 2 Training and test datasets

Training dataset	Anomalies	Test dataset
1	Slammer and Nimda	Code Red I
2	Slammer and Code Red I	Nimda
3	Nimda and Code Red I	Slammer
4	Slammer	Nimda and Code Red I
5	Nimda	Slammer and Code Red I
6	Code Red I	Slammer and Nimda
7	Slammer, Nimda, and Code Red I	RIPE or BCNET

test the SVM classifiers. The SVM^{light} library, developed in C language, is an effective tool for classification, regression, and ranking when dealing with large training samples. We adjust the value of parameter C (5), which controls the trade-off between the training error and the margin as well as the cost factor [30].

Feature selection algorithms were implemented in MATLAB and were used to minimize the dimension of the dataset matrix by selecting the ten most relevant features. We used: Fisher, minimum redundancy maximum relevance (mRMR) (mutual information difference (MID), mutual information quotient (MIQ), and mutual information base (MIBASE)), odds ratio (OR), extended/weighted/multi-class odds ratio (EOR/WOR/MOR), and class discriminating measure (CDM). Hence, the dimension of feature matrices that correspond to a 5-day period of collected data is $7{,}200 \times 10$. Each matrix row corresponds to the top ten selected features within the 1-min interval. In two-way classifications, we target two classes: anomaly (positive) and regular (negative) for each training dataset. While the two-way classification only identifies whether or not a data point is anomaly, the four-way classification detects the specific type of BGP anomaly: Slammer, Nimda, Code Red I, or regular (RIPE and BCNET). SVM classifies each data point x_n, where $n = 1, \ldots, 7{,}200$, with a training target class t_n either as anomaly $y = 1$ or regular $y = -1$.

In a two-way classification, all anomalies are treated as one class. Validity of the proposed models is tested by applying two-way SVM classification on BGP traffic traces collected from RIPE and BCNET on December 20, 2011. All data points in the regular RIPE and BCNET datasets contain no anomalies and are, hence, labeled as regular traffic, as shown in Table 3. The results are generated using the MATLAB *fitcsvm* support vector classifier. The index of models reflects the dataset used for training and testing. These datasets contain no anomalies (both TP and FN values are zero) and, hence, precision is equal to zero while sensitivity is not defined. Consequently, the F-Score is also not defined and the accuracy reduces to:

$$\text{accuracy} = \frac{TN}{TN + FP}. \tag{7}$$

Table 3 Performance of the two-way SVM classification using unbalanced datasets

		Accuracy (%)			F-Score (%)
			Regular		
SVM	Feature	Test dataset	RIPE	BCNET	Test dataset
SVM_1	37 features	78.65	69.17	57.22	39.51
SVM_1	Fisher	81.93	85.67	80.49	41.16
SVM_1	MID	85.38	92.63	83.68	45.18
SVM_1	MIQ	80.86	86.89	83.75	39.56
SVM_1	MIBASE	80.86	87.10	88.47	39.51
SVM_1	OR	78.57	70.15	66.74	38.01
SVM_1	WOR	**88.03**	89.88	70.90	**47.18**
SVM_1	MOR	83.88	83.40	83.75	44.53
SVM_1	CDM	84.40	81.36	90.56	44.05
SVM_2	37 features	**55.50**	89.89	82.08	**24.29**
SVM_2	Fisher	54.22	96.28	**98.33**	16.43
SVM_2	MID	53.89	95.88	95.76	11.89
SVM_2	MIQ	55.10	96.10	97.57	20.74
SVM_2	MIBASE	55.11	92.78	95.83	19.32
SVM_2	OR	54.93	93.90	97.64	15.87
SVM_2	WOR	54.53	**97.39**	93.26	14.56
SVM_2	MOR	55.36	96.74	97.85	20.21
SVM_2	CDM	54.67	96.60	97.64	16.65
SVM_3	37 features	93.04	73.92	59.24	75.93
SVM_3	Fisher	93.31	80.63	57.71	76.51
SVM_3	MID	93.35	75.33	59.65	76.30
SVM_3	MIQ	93.28	78.99	57.50	76.27
SVM_3	MIBASE	**93.40**	79.14	57.85	**76.52**
SVM_3	OR	90.44	80.53	79.03	70.32
SVM_3	WOR	93.08	77.51	58.19	75.61
SVM_3	MOR	92.92	76.79	68.68	75.48
SVM_3	CDM	92.93	76.97	68.13	75.54

The best accuracy (93.40%) for two-way classification is achieved by using SVM_3 for the Slammer test dataset. The Nimda test data points that are correctly classified as anomalies (true positive) in the two-way classification are shown in Fig. 2 (top) while incorrectly classified anomaly and regular data points are shown in Fig. 2 (bottom).

The SVM models are used to compare results for unbalanced and balanced training datasets. Examples using SVM_3 models are shown in Table 4. For unbalanced training datasets, the features selected by the MIBASE algorithm generate the best F-Score (76.52%). The best F-Scores is achieved by SVM_b3 (66.59%) that is trained using balanced datasets and the MID algorithm. Incorrectly classified anomalies (false positive) and regular points (false negative) are shown in Fig. 3.

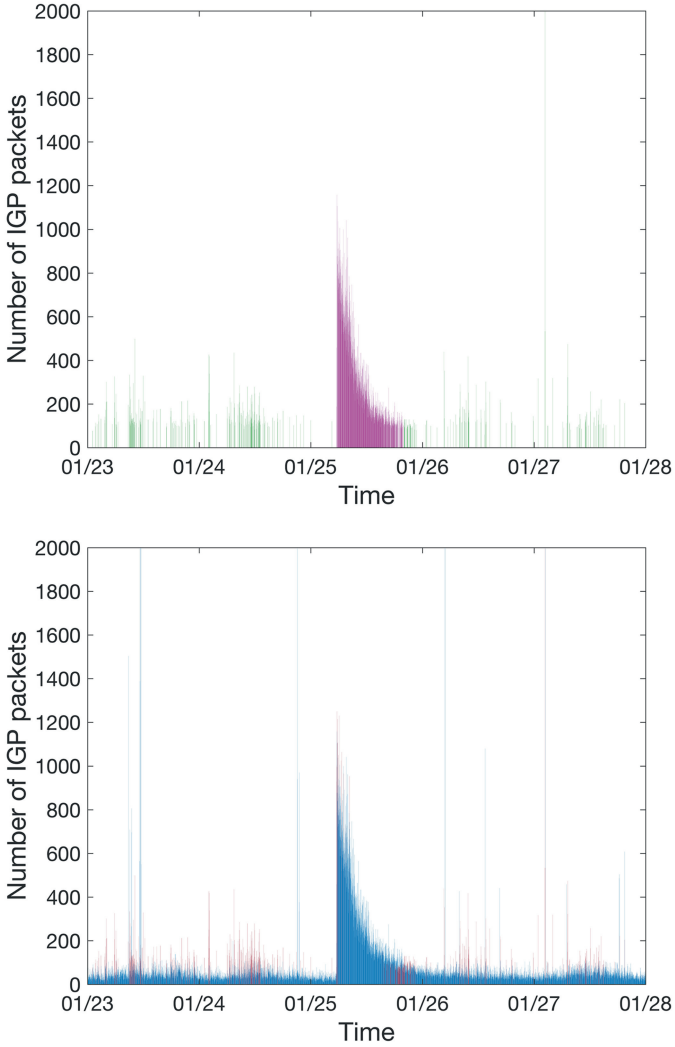

Fig. 2 SVM classifier applied to Slammer traffic collected from January 23 to 27, 2003: Shown in purple are correctly classified anomalies (true positive) while shown in green are incorrectly classified anomalies (false positive) (top). Shown in red are incorrectly classified anomalies (false positive) and regular (false negative) data points while shown in blue are correctly classified anomaly (true positive) and regular (true negative) data points (bottom)

The SVM classifier may be also used to identify multiple classes [21]. For four-way classification, we use four training datasets (Slammer, Nimda, Code Red I, and RIPE) to identify the specific type of BGP data point: Slammer, Nimda, Code Red I, or regular (RIPE or BCNET). Classification of RIPE dataset achieves 91.85% accuracy, as shown in Table 5. The results are generated using the MATLAB *fitcecoc* support vector classifier.

Table 4 Accuracy and F-Score using the SVM$_b$3 models using balanced datasets

Balanced datasets					
Accuracy (%)					F-Score (%)
Model		Test dataset	RIPE	BCNET	Test dataset
SVM$_b$3	37 Features	87.19	63.31	51.11	64.76
SVM$_b$3	Fisher	84.74	62.58	52.01	60.54
SVM$_b$3	MID	**88.33**	66.10	**60.28**	**66.59**
SVM$_b$3	MIQ	87.56	66.69	59.93	65.43
SVM$_b$3	MIBASE	87.40	**66.76**	60.14	65.18
SVM$_b$3	OR	71.94	52.92	52.78	46.16
SVM$_b$3	WOR	87.68	65.61	56.60	65.45
SVM$_b$3	MOR	86.44	64.69	57.43	63.69
SVM$_b$3	CDM	86.67	64.81	57.85	63.99

4 Long Short-Term Memory (LSTM) Neural Network

The LSTM approach employs a special form of the recurrent neural networks (RNNs). Traditional RNNs are designed to store inputs in order to predict the outputs [30]. However, they perform poorly when they need to bridge segments of information with long-time gaps. Unlike the traditional RNNs, LSTM networks are capable of connecting time intervals to form a continuous memory [22]. They were introduced to overcome long-term dependency and vanishing gradient problems [35].

The LSTM module consists of an input layer, a single hidden LSTM layer, and an output layer. The input layer consists of 37 nodes (each node corresponds to one feature) that serve as inputs to the LSTM layer, which consists of LSTM cells called the "memory blocks" [34]. An LSTM cell is composed of: (a) forget gate f_n, (b) input gate i_n, and (c) output gate o_n. The forget gate discards the irrelevant memories according to the cell state, the input gate controls the information that will be updated in the LSTM cell, and the output gate works as a filter to control the output. The logistic sigmoid and network output functions are denoted by σ and tanh, respectively. The output layer has one node that is connected to the output of the LSTM layer. The output is labeled by 1 (anomaly) or -1 (regular). An LSTM module is shown in Fig. 4 [3].

Keras [2], a modular neural network library for the Python language, is designed for deep learning and used as a framework for implementing the LSTM classifier. It uses either TensorFlow or Theano library as the back-end. In this study, we used Keras 2.0.2 with Python 2.7.13 and TensorFlow 1.0.1 [5]. We use all 37 features [18] because LSTM cells select the useful features during the learning process. Keras generates LSTM sequential models with 37-dimensional inputs, 1 hidden layer, and 1-dimensional outputs. Each hidden layer contains 256 LSTM cells. The length of the time sequence is 20, which implies that samples from the current and the 19 most recent time stamps are used to predict the future instance. We utilize the datasets

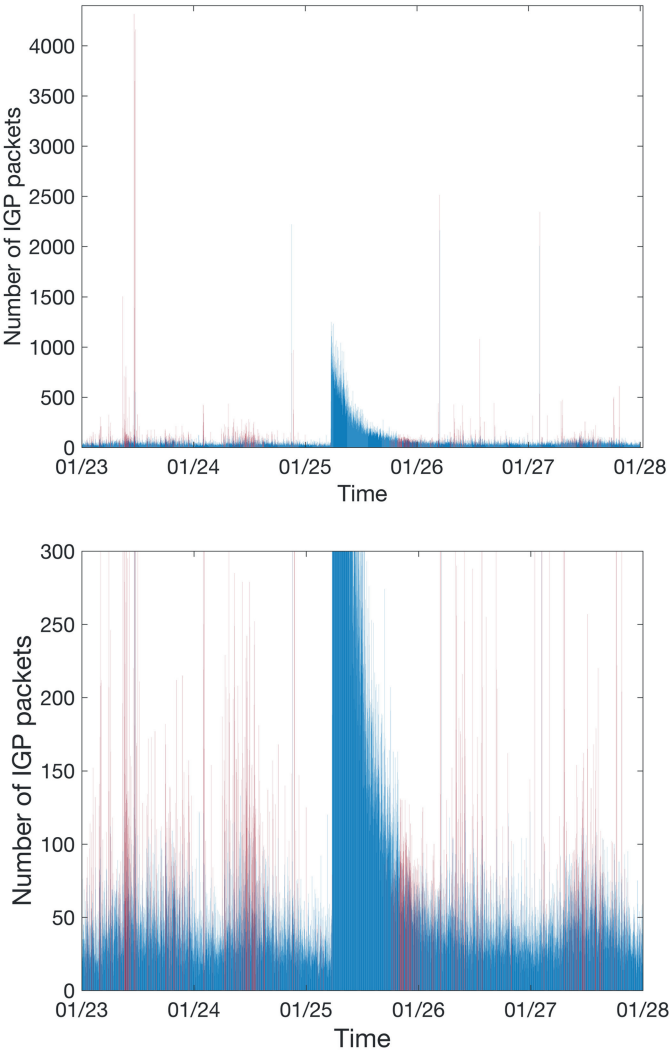

Fig. 3 SVM classifier applied to Slammer traffic collected from January 23 to 27, 2003. Shown in red is incorrectly classified traffic (top) and the detail (bottom)

shown in Table 2 to generate LSTM models. The "Adam" optimizer [25] with the learning rate "lr = 0.001" is used when compiling the LSTM model because of its superior performance when dealing with large datasets and/or high-dimensional parameter spaces.

Results of the LSTM classification are shown in Table 6. When using unbalanced datasets, the highest F-Score (84.62%) is achieved by LSTM_u3 trained by using the combined Nimda and Code Red I datasets. Among balanced datasets, the

Table 5 Accuracy of the four-way SVM classification

	Accuracy (%)	
Feature	RIPE	BCNET
37 features	84.47	76.11
Fisher	87.00	**83.13**
MID	91.11	80.07
MIQ	87.26	73.06
MIBASE	87.14	82.64
EOR	87.69	74.38
WOR	88.57	65.35
MOR	88.72	66.39
CDM	**91.85**	72.15

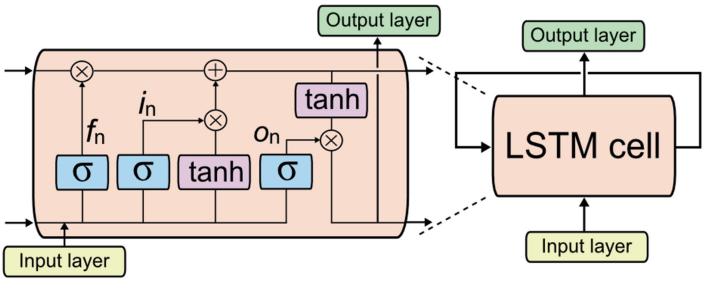

Fig. 4 Repeating module for the LSTM neural network. Shown are the input layer, LSTM layer with one LSTM cell, and output layer

Table 6 Accuracy and F-Score using LSTM models for unbalanced and balanced datasets

	Unbalanced datasets			
	Accuracy (%)			F-Score (%)
	Test dataset	RIPE	BCNET	Test dataset
$LSTM_u1$	95.22	65.49	57.30	83.17
$LSTM_u2$	53.94	51.53	50.80	11.81
$LSTM_u3$	**95.87**	56.74	58.55	**84.62**
	Balanced datasets			
	Accuracy (%)			F-Score (%)
	Test dataset	RIPE	BCNET	Test dataset
$LSTM_b1$	56.43	60.48	62.78	26.59
$LSTM_b2$	56.32	44.27	53.58	**65.96**
$LSTM_b3$	**82.98**	55.00	48.20	58.54

$LSTM_b2$ model achieves the best performance (65.96%). We suspect that the poor performance of $LSTM_u2$ may be caused by noisy data. LSTM models reported in the previous study [18] were improved by adjusting the length of time sequence and choice of the optimizer.

5 Hidden Markov Model (HMM)

HMMs are statistical tools used to model stochastic processes that consist of two embedded processes: the observable process that maps features and the unobserved hidden Markov process. We assume that the observations are independent and identically distributed (iid). In this survey, HMMs are used for non-parametric supervised classification. We implement the first order HMMs using the MATLAB statistical toolbox. Each HMM model is specified by a tuple $\lambda = (N, M, \alpha, \beta, \pi)$, where:

N: number of hidden states (cross-validated)
M: number of observations
α: transition probability distribution $N \times N$ matrix
β: emission probability distribution $N \times M$ matrix
π: initial state probability distribution matrix.

The proposed HMM classification models consist of three stages:

- *Sequence extracting and mapping*: All features are mapped to an observation vector.
- *Training*: Two HMMs for a two-way classification and four HMMs for a four-way classification are trained to identify the best probability distributions α and β for each class. The HMMs are trained and validated for various numbers of hidden states N.
- *Classification*: Maximum likelihood probability $p(x|\lambda)$ is used to classify the test observation sequences.

In the sequence extraction stage, the BGP feature matrix is mapped to a sequence of observations by selecting feature sets: FS1 (2,5,6,7,9,10,11,13) and FS2 (2,5,6,7,10,11,13). In both cases, the number of observations is chosen to be the maximum number of selected features.

HMMs are trained and validated for various numbers of hidden states. For each HMM, a tenfold cross-validation with the Baum-Welch algorithm [11] is used for training to find the best α (transition) and β (emission) matrices by calculating the largest maximum likelihood probability $p(x|\lambda_{\mathrm{HMM}_x})$. We construct seven two-way HMM models, as shown in Table 7. The name of each HMM model reflects the training dataset and the number of hidden states. Each test observation sequence is classified based on $p(x|\lambda_{\mathrm{HMM}_x})$.

Table 7 HMM models: two-way classification

	Number of hidden states						
Training dataset	2	3	4	5	6	7	8
Slammer, Nimda, and Code Red I	HMM_1	HMM_2	HMM_3	HMM_4	HMM_5	HMM_6	HMM_7

Table 8 Accuracy of the two-way HMM classification

N		Accuracy (%)	
No. of hidden states	Feature set	RIPE	BCNET
2	(2,5,6,7,9,	42.15	50.62
3	10,11,13)	45.21	**62.99**
4		16.11	36.53
5		19.31	27.15
6		16.81	21.11
7		**83.26**	52.01
8		67.50	41.04
N		Accuracy (%)	
No. of hidden states	Feature set	RIPE	BCNET
2	(2,5,6,7,	41.46	50.56
3	10,11,13)	26.60	**59.17**
4		10.14	25.76
5		4.03	28.06
6		16.67	21.11
7		**82.99**	51.94
8		66.87	40.97

We use regular RIPE and BCNET datasets to evaluate performance of various classification models. The FS1 and FS2 are used to create an observation sequence for each HMM. The accuracy (1) is calculated using the highest $p(x|\lambda_{\text{HMM}_x})$ in the numerator while sequences in the denominator share the same number of hidden states. As shown in Table 8, HMMs have higher accuracy using feature set FS1. The regular RIPE and BCNET datasets have the highest accuracy when classified using HMMs with seven and three hidden states, respectively. Similar HMM models were developed for four-way classification and tested on regular RIPE and BCNET datasets.

6 Naive Bayes

The naive Bayes classifiers are among the most efficient machine learning classification techniques. The generative Bayesian models are used as classifiers using labeled datasets. They assume conditional independence among features. Hence,

$$\Pr(\mathbf{X}_k = \boldsymbol{x}_k, \mathbf{X}_l = \boldsymbol{x}_l | c_j) = \Pr(\mathbf{X}_k = \boldsymbol{x}_k | c_j) \Pr(\mathbf{X}_l = \boldsymbol{x}_l | c_j), \tag{8}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$ are realizations of feature vectors $\mathbf{X}_k$ and $\mathbf{X}_l$, respectively. In a two-way classification, classes labeled $c_1 = 1$ and $c_2 = -1$ denote anomalous and regular data points, respectively. For a four-way classification, four classes labeled $c_1 = 1$, $c_2 = 2$, $c_3 = 3$, and $c_4 = 4$ refer to Slammer, Nimda, Code Red I, and regular data points, respectively. Even though it is naive to assume that

features are independent for a given class (8), for certain applications naive Bayes classifiers perform better compared to other classifiers. They have low complexity, may be trained effectively with smaller datasets, and may be used for online real time detection of anomalies.

The probability distributions of the priors $\Pr(c_j)$ and the likelihoods $\Pr(\mathbf{X}_i = \boldsymbol{x}_i|c_j)$ are estimated using the training datasets. Posterior of a data point represented as a row vector $\boldsymbol{x}_i$ is calculated using the Bayes rule:

$$\begin{aligned}\Pr(c_j|\mathbf{X}_i = \boldsymbol{x}_i) &= \frac{\Pr(\mathbf{X}_i = \boldsymbol{x}_i|c_j)\Pr(c_j)}{\Pr(\mathbf{X}_i = \boldsymbol{x}_i)} \\ &\approx \Pr(\mathbf{X}_i = \boldsymbol{x}_i|c_j)\Pr(c_j). \end{aligned} \tag{9}$$

The naive assumption of independence among features helps calculate the likelihood of a data point as:

$$\Pr(\mathbf{X}_i = \boldsymbol{x}_i|c_j) = \prod_{k=1}^{K} \Pr(X_{ik} = x_{ik}|c_j), \tag{10}$$

where K denotes the number of features. The probabilities on the right-hand side (10) are calculated using the Gaussian distribution $\mathcal{N}$:

$$\Pr(\mathbf{X}_{ik} = \boldsymbol{x}_{ik}|c_j, \mu_k, \sigma_k) = \mathcal{N}(X_{ik} = x_{ik}|c_j, \mu_k, \sigma_k), \tag{11}$$

where μ_k and σ_k are the mean and standard deviation of the kth feature, respectively. We assume that priors are equal to the relative frequencies of the training data points for each class c_j. Hence,

$$\Pr(c_j) = \frac{N_j}{N}, \tag{12}$$

where N_j is the number of training data points that belong to the jth class and N is the total number of data points.

The parameters of two-way and four-way classifiers are estimated and validated by a tenfold cross-validation. In a two-way classification, an arbitrary training data point $\boldsymbol{x}_i$ is classified as anomalous if the posterior $\Pr(c_1|\mathbf{X}_i = \boldsymbol{x}_i)$ is larger than $\Pr(c_2|\mathbf{X}_i = \boldsymbol{x}_i)$.

We use the MATLAB statistical toolbox to implement naive Bayes classifiers and identify anomaly or regular data points. Datasets listed in Table 2 are used to train the two-way classifiers. The test datasets are Code Red I, Nimda, and Slammer. The combination of Code Red I and Nimda training data points (NB3) achieves the accuracy (92.79%) and F-Score (66.49%), as shown in Table 9. The NB models classify the data points of regular RIPE and regular BCNET datasets with 90.28% and 88.40% accuracies, respectively. The OR and EOR algorithms generate identical results for the two-way classification and thus performance of the EOR algorithm is omitted.

Table 9 Performance of the two-way naive Bayes classification

No.	NB	Feature	Accuracy (%)			F-Score (%)
			Test dataset	RIPE	BCNET	Test dataset
1	NB1	37 features	82.03	82.99	79.03	29.52
2	**NB1**	Fisher	**90.94**	88.13	76.46	36.08
3	NB1	MID	86.75	86.04	83.61	24.64
4	NB1	MIQ	88.86	87.78	75.21	30.38
5	NB1	MIBASE	90.92	87.64	77.92	35.12
6	**NB1**	OR	90.35	83.82	72.57	**37.33**
7	NB1	WOR	86.64	86.88	78.06	18.34
8	NB1	MOR	89.28	86.04	75.56	26.05
9	NB1	CDM	89.53	87.78	75.14	27.50
10	**NB2**	37 features	**62.56**	82.85	86.25	**48.78**
11	NB2	Fisher	57.51	87.01	83.26	27.97
12	**NB2**	MID	57.64	79.58	**88.40**	31.31
13	NB2	MIQ	56.68	84.38	82.15	26.35
14	NB2	MIBASE	57.60	86.88	82.99	28.55
15	NB2	OR	60.38	84.31	84.93	37.31
16	NB2	WOR	53.76	80.69	87.36	18.23
17	NB2	MOR	56.81	88.06	84.10	26.34
18	**NB2**	CDM	56.50	**90.28**	83.26	25.50
19	NB3	37 features	83.58	84.79	81.18	51.12
20	**NB3**	Fisher	**92.79**	87.36	75.97	**66.49**
21	NB3	MID	86.71	87.08	86.32	52.08
22	NB3	MIQ	92.47	88.68	77.15	65.03
23	NB3	MIBASE	92.49	89.31	80.42	62.97
24	NB3	OR	80.63	67.29	59.79	52.47
25	NB3	WOR	88.22	87.22	81.81	50.29
26	NB3	MOR	89.89	88.06	81.39	51.01
27	NB3	CDM	90.89	88.54	77.92	55.43

The Slammer worm test data points that are correctly classified as anomalies (true positives) during the 16 h time interval are shown in Fig. 5 (top). Incorrectly classified (false positives and false negatives) using the NB 3 classifier trained based on the features selected by Fisher in the two-way classification are shown in Fig. 5 (bottom). Most anomalous data points with large number of IGP packets (*volume* feature) are correctly classified.

The four-way naive Bayes model classifies data points as Slammer, Nimda, Code Red I, or Regular. Both regular RIPE and BCNET datasets are tested. Classification results for regular datasets are shown in Table 10. Although it is more difficult to classify four distinct anomalies, the classifier trained based on the features selected by the CDM algorithm achieves 90.14% accuracy. Variants of the OR feature selection algorithm perform well when combined with the naive Bayes classifiers

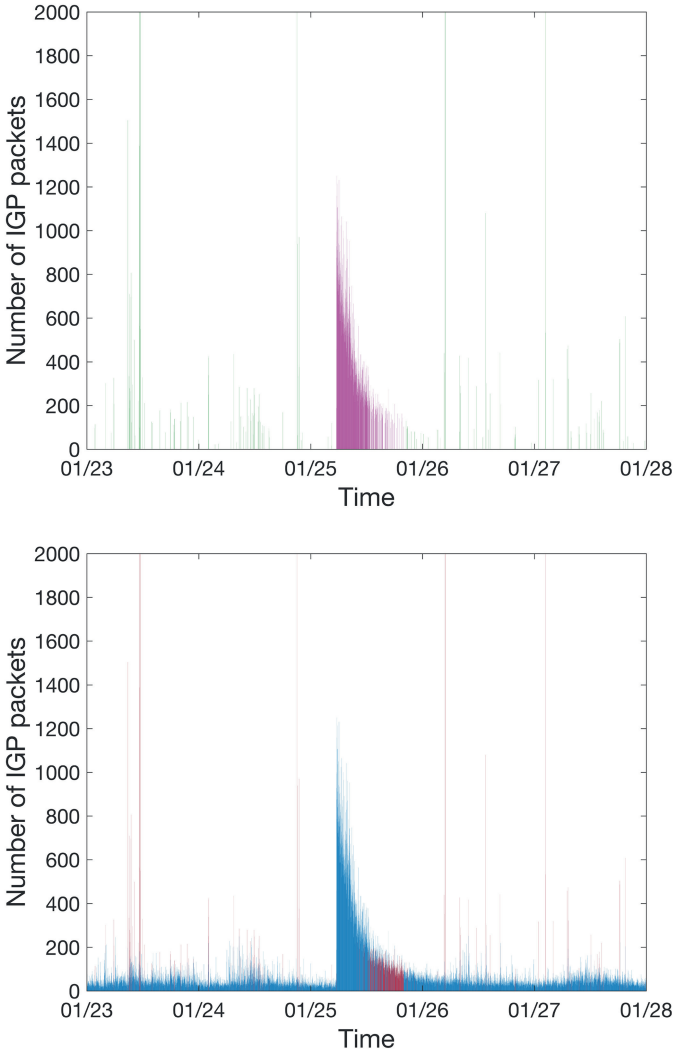

Fig. 5 Naive Bayes classifier applied to Slammer traffic collected from January 23 to 27, 2003: Shown in purple are correctly classified anomalies (true positive) while shown in green are incorrectly classified anomalies (false positive) (top). Shown in red are incorrectly classified anomalies (false positive) and regular (false negative) data points while shown in blue are correctly classified anomaly (true positive) and regular (true negative) data points (bottom)

because feature scores are calculated using the probability distribution that the naive Bayes classifiers use for posterior calculations. Hence, the features selected by the OR variants are expected to have stronger influence on the posteriors calculated by the naive Bayes classifiers [28]. Performance of the naive Bayes classifiers is often inferior to the SVM and HMM classifiers.

Table 10 Accuracy of the four-way naive Bayes classification

No.	Feature	Average accuracy (%)	
		RIPE regular	BCNET
1	37 features	85.90	85.07
2	Fisher	88.75	84.24
3	MID	86.18	82.85
4	MIQ	89.38	84.51
5	MIBASE	88.75	84.24
6	EOR	89.03	90.07
7	WOR	88.40	87.36
8	MOR	88.75	87.71
9	**CDM**	**90.14**	83.54

Table 11 Decision tree algorithm: performance

Training dataset	Test dataset	Accuracy (%)			F-Score (%)
		(Test)	RIPE	BCNET	(Test)
Dataset 1	Code Red I	85.36	89.00	77.22	47.82
Dataset 2	Nimda	58.13	94.19	81.18	26.16
Dataset 3	Slammer	**95.89**	89.42	77.78	**84.34**

7 Decision Tree Algorithm

The decision tree algorithm is one of the most successful supervised learning techniques [33]. A tree is "learned" by splitting the source set into subsets based on an attribute value. This process is repeated on each derived subset using recursive partitioning. The recursion is completed when the subset at a node contains all values of the target variable or when the splitting no longer adds value to the predictions. After a decision tree is learned, each path from the root node (source) to a leaf node is transformed into a decision rule. Therefore, a set of rules is obtained by a trained decision tree that is used for classifying unseen samples. Test accuracies and F-Scores are shown in Table 11. The results are generated using MATLAB *fitctree* from the statistics and machine learning toolbox. The lower accuracy of the training dataset 2 is due to the distributions of anomaly and regular data points in training and test datasets.

8 Extreme Learning Machine Algorithm (ELM)

The Extreme Learning Machine (ELM) [1, 23] is an efficient learning algorithm used with a single hidden layer feed-forward neural network. It randomly selects the weights of the hidden layer and analytically determines the output weights. ELM avoids the iterative tuning of the weights used in traditional neural networks and, hence, it is fast and may be used as an online algorithm.

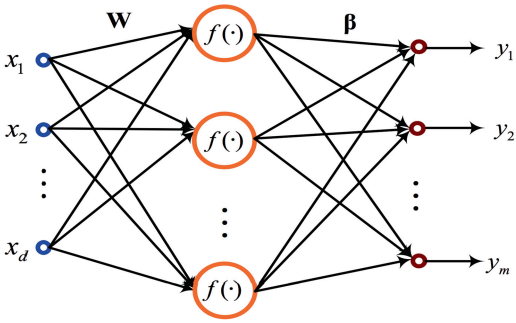

Fig. 6 Neural network architecture of the ELM algorithm

ELM employs weights connecting the input and hidden layers with the bias terms randomly initialized while the weights connecting the hidden and output layers are analytically determined. Its learning speed is higher than the traditional gradient descent-based method. Reported research results indicate that ELM may learn much faster than SVMs. Incremental and weighted extreme learning machines are variants of ELM.

A neural network architecture of the ELM algorithm is shown in Fig. 6, where $[x_1, x_2, \ldots, x_d]$ is the input vector; d is the feature dimension; $f(.)$ is the activation function; $\mathbf{W}$ is the vector of weights connecting the inputs to hidden units; $[y_1, y_2, \ldots, y_m]$ is the output vector; and β is the weigh vector connecting the hidden and the output units.

The three datasets used to verify ELM's performance are shown in Table 2. The number of hidden units is selected by a 5-fold cross validation for each training dataset. The best testing accuracy was achieved by choosing 315 hidden units for each dataset. The input vectors of the training datasets are mapped onto $[-1, 1]$ as:

$$x_i^{(p)} = 2\frac{x_i^{(p)} - x_{i_{min}}}{x_{i_{max}} - x_{i_{min}}} - 1, \tag{13}$$

where $x_i^{(p)}$ is the ith feature of the pth sample while $x_{i_{min}}$ and $x_{i_{max}}$ are the minimum and maximum values of the ith feature of the training sample, respectively.

The accuracies and F-Scores for the three ELM test datasets with 37 or 17 features are shown in Table 12. Accuracies for RIPE and BCNET datasets are also included. For each dataset, 100 trials were repeated. The binary Features 14–33 (shown in Chapter 3, Table 6) are removed to form a set of 17 features.

9 Discussion

We have introduced and examined various machine learning techniques for detecting network anomalies. Each approach has its unique advantages and limitations. Soft-margined SVMs perform well in classification tasks. However, they require

Table 12 Performance of the ELM algorithm using datasets with 37 and 17 features

No. of features	Training dataset	Test dataset	Accuracy (%)			F-Score (%)
			(Test)	RIPE	BCNET	(Test)
37	Dataset 1	Code Red I	80.92	75.81	69.03	36.27
	Dataset 2	Nimda	54.42	96.15	91.88	13.72
	Dataset 3	Slammer	**86.96**	78.57	73.47	**55.31**
17	Dataset 1	Code Red I	80.75	73.43	62.43	39.90
	Dataset 2	Nimda	55.13	94.11	83.75	15.97
	Dataset 3	Slammer	**92.57**	80.57	72.71	**72.52**

relatively long time for training models when dealing with large datasets. Although LSTM algorithms achieve high accuracy and F-Score, their performance is limited to using time sequential input data. HMM and naive Bayes algorithms compute probabilities that events occur and are suitable for detecting multiple classes of anomalies. Decision tree is commonly used in data mining due to its explicit and efficient decision making. ELM is an efficient classifier while its performance is limited due to its simple structure.

10 Conclusion

In this Chapter, we classify anomalies in BGP traffic traces using a number of classification models that employ various feature selection algorithms. We conduct experiments using a number of datasets and various features extracted from data points. We analyze performance of BGP anomaly detection models based on SVM, LSTM, HMM, naive Bayes, Decision Tree, and ELM classifiers. When the testing accuracy of the classifiers is low, feature selection is used to improve their performance. Performance of the classifiers is greatly influenced by the employed datasets. While no single classifier that we have employed performs the best across all used datasets, machine learning proved to be a feasible approach to successfully classify BGP anomalies using various classification models.

Acknowledgements We thank Yan Li, Hong-Jie Xing, Qiang Hua, and Xi-Zhao Wang from Hebei University, Marijana Ćosović from University of East Sarajevo, and Prerna Batta from Simon Fraser University for their helpful contributions in earlier publications related to this project.

References

1. (Jan. 2017) Extreme Learning Machines [Online]. Available: http://www.ntu.edu.sg/home/egbhuang/elm_codes.html.
2. (Mar. 2017) Keras: Deep Learning library for Theano and TensorFlow. [Online]. Available: https://keras.io/.

3. (Mar. 2018) Understanding LSTM Networks [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
4. (Mar. 2018) SVM^{light}: Support Vector Machine. [Online]. Available: http://svmlight.joachims.org/.
5. (Mar. 2018) TensorFlow. [Online]. Available: https://www.tensorflow.org/.
6. T. Ahmed, B. Oreshkin, and M. Coates, "Machine learning approaches to network anomaly detection," in *Proc. USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, Cambridge, MA, Apr. 2007, pp. 1–6.
7. T. Ahmed, M. Coates, and A. Lakhina, "Multivariate online anomaly detection using kernel recursive least squares," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, Anchorage, AK, USA, May 2007, pp. 625–633.
8. D. Ariu, R. Tronci, and G. Giacinto, "HMMPayl: an intrusion detection system based on Hidden Markov Models," *Comput. Security*, vol. 30, no. 4, pp. 221–241, 2011.
9. N. Al-Rousan and Lj. Trajković, "Machine learning models for classification of BGP anomalies," in *Proc. IEEE Conf. on High Performance Switching and Routing (HPSR)*, Belgrade, Serbia, June 2012, pp. 103–108.
10. N. Al-Rousan, S. Haeri, and Lj. Trajković, "Feature selection for classification of BGP anomalies using Bayesian models," in *Proc. Int. Conf. Mach. Learn. Cybern. (ICMLC)*, Xi'an, China, July 2012, pp. 140–147.
11. C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag, 2006, pp. 325–358.
12. M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Commun. Surveys Tut.*, vol. 16, no. 1, pp. 303–336, Mar. 2014.
13. N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 1–6, June 2004.
14. M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, "MS-LSTM: a multi-scale LSTM model for BGP anomaly detection," in *Proc. 2016 IEEE 24th Int. Conf. Netw. Protocols, Workshop Mach. Learn. Comput. Netw.*, Singapore, Nov. 2016, pp. 1–6.
15. C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
16. A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, Feb. 2012.
17. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Rev.*, vol. 7, pp. 1–30, Jan. 2006.
18. Q. Ding, Z. Li, P. Batta, and Lj. Trajković, "Detecting BGP anomalies using machine learning techniques," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, Budapest, Hungary, Oct. 2016, pp. 3352–3355.
19. T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1924, Oct. 1998.
20. D. N. T. How, K. S. M. Sahari, Y. Hu, and C. K. Loo, "Multiple sequence behavior recognition on humanoid robot using long short-term memory (LSTM)," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, China, Dec. 2014, pp. 109–114.
21. C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
22. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Oct. 1997.
23. G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.
24. G. B. Huang, X. J. Ding, and H. M. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, Dec. 2010.
25. D. P. Kingma and Jimmy Ba, "Adam: A method for sochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, USA, Dec. 2014.
26. C. F. Lin and S. D. Wang, "Fuzzy support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 464–471, Feb. 2002.

27. P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. Eur. Symp. Artificial Neural Netw., Comput. Intell. Mach. Learn.*, Belgium, Apr. 2015, pp. 89–94.
28. D. Mladenic and M. Grobelnik, "Feature selection for unbalanced class distribution and naive Bayes," in *Proc. Int. Conf. Machine Learning*, Bled, Slovenia, June 1999, pp. 258–267.
29. A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. Int. Conf. Measurement and Modeling of Comput. Syst.*, Banff, AB, Canada, June 2005, pp. 50–60.
30. K. Morik, P. Brockhausen, and T. Joachims, "Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring," in *Proc. Int. Conf. on Machine Learning, ICML 1999*, Bled, Slovenia, June 1999, pp. 268–277.
31. A. Munoz and J. Moguerza, "Estimation of high-density regions using one-class neighbor machines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 476–480, Mar. 2006.
32. F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proc. 15th Int. Conf. Mach. Learn.*, Madison, WI, USA, July 1998, pp. 445–453.
33. J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
34. H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. INTERSPEECH*, Singapore, Sept. 2014, pp. 338–342.
35. R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, May 1992.
36. C. Wagner, J. Francois, R. State, and T. Engel, "Machine learning approach for IP-flow record anomaly detection," in *Lecture Notes in Computer Science: Proc. 10th Int. IFIP TC 6 Netw. Conf.*, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, C. Scoglio, Eds. Springer 2011, vol. 6640, pp. 28–39.
in *Proc. 10th Int. IFIP TC 6 Conf. Netw.*, NETWORKING 2011. Lecture Notes in Computer Science, vol 6640 Apr. 2009 vol. I, pp. 28–39.
37. D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996.
38. X. Yang, Q. Song, and A. Cao, "Weighted support vector machine for data classification," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Montreal, QC, Canada, Aug. 2005, vol. 2, pp. 859–864.
39. W. Zong, G. B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.
40. K. Zhang, A. Yen, X. Zhao, D. Massey, S.F. Wu, L. Zhang, "On detection of anomalous routing dynamics in BGP," in *Lecture Notes in Computer Science: Proc. Int. Conf. Research Netw.*, N. Mitrou, K. Kontovasilis, G. N. Rouskas, I. Iliadis, L. Merakos, Eds. Springer 2004, vol. 3042, pp. 259–270.