Neurocomputing 149 (2015) 415–425

Contents lists available at ScienceDirect

Neurocomputing

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

Anomaly detection in traffic using L1-norm minimization extreme learning machine

Yibing Wang [a], Dong Li [a], Yi Du [b], Zhisong Pan [a,*]

[a] College of Command Information System, PLA University of Science and Technology, Nanjing 210007, China
[b] Telecommunication Network Technology Management Center, Beijing 100840, China

A R T I C L E I N F O

Article history:
Received 5 August 2013
Received in revised form
3 April 2014
Accepted 7 April 2014
Available online 10 September 2014

Keywords:
Traffic classification
Anomaly detection
Extreme learning machine
Support vector machine
L1-norm minimization

A B S T R A C T

Machine learning algorithms are widely used for traffic classification and anomaly detection nowadays, however, how to fast and accurately classify the flows remains extremely challengeable. In this paper, we propose an extreme learning machine (ELM) based algorithm called L1-Norm Minimization ELM, which fully inherits the merits of ELM, and meanwhile, exhibits the sparsity-induced characteristics which could reduce the complexity of learning model. At the evaluation stage, we preprocessed the raw data trace from trans-Pacific backbone link between Japan and the United States, and generated 248 features datasets. The empirical study shows that L1-ELM can achieve good generalization performance on the evaluation datasets, while preserving the fast learning and little human intervened advantages that ELM has.

1. Introduction

Continued development of the Internet has spawned a large number of web-based applications, which have brought tremendous network traffic and anomalous activities. Fast and accurate classification of type of traffic and abnormal behaviors on the backbone network becomes more and more important for the Internet Service Providers (ISPs) and network administrators. For example, ISPs need to monitor the constitutions of different applications so as to prioritize traffic of the QoS-sensitive application, prevent and locate harmful activities and take additional steps for other reasons, say, politics [1].

To date there mainly exists four effective fashions on traffic classification and anomaly detection: (1) transport layer port number based method, which is easy to implement. However, it becomes increasingly unreliable for the emergence of all kinds of new applications types, e.g. P2P application, which uses randomly assigned ports number that are not registered with IANA [2,3]; (2) Deep packet inspection (DPI), a more effective technique, which inspect the packet payload with existing patterns [4,2,5]. While it has higher detection accuracy, this method demands much more resources and bandwidth than ISPs and network administrators could afford and could be blocked from encryption easily. It should be deployed to the border routers instead of backbone network; (3) host behavior based method, which lies on the host system to contrast the audit records and security logs against archived host profiles. It would raise an alert once they were found unmatched [4,6]; (4) traffic flow features based methods, which prevails recently and have been proven promising for the following perspectives [7,8]. Firstly, flow features could be easily derived from packets header statistics alone, which avoid privacy issues, legal constraints and resource-intensive requirements. Secondly, in the face of obstacles of encryptions and so on, flow based features could almost fully characterize the different applications by using the machine learning methodologies in the pattern recognition field. Thirdly, many efficient machine learning algorithms [9–19] have already been applied to dealing with traffic classification problems, which enriched the theoretical foundations while providing comprehensive applications and analysis.

There are many state-of-the-art methodologies applied in the traffic classification field. Erman et al. [11] used two unsupervised clustering algorithms, K-Means and DBSCAN, to demonstrate how cluster analysis can be used to effectively identify groups of traffic that are similar using only transport layer statistics. Kim et al. [1] conducted an evaluation of three traffic classification approaches: port based, host behavior based and flow features based. After comparing seven commonly used machine learning algorithms with the other two kinds of traffic classification methods, they found that Support Vector Machine (SVM) algorithm achieved the highest accuracy on every trace and application. Williams et al. [17] evaluated Naiive Bayes, C4.5, Bayesian Network and Naiive Bayes Tree algorithms to show that, although classification accuracy between the algorithms is similar, computational performance can differ significantly.

* Corresponding author.
E-mail address: hotpzs@hotmail.com (Z. Pan).

http://dx.doi.org/10.1016/j.neucom.2014.04.073
0925-2312/© 2014 Elsevier B.V. All rights reserved.

Machine learning methodologies are the main solutions to the flow based traffic classification. However, different from classical data mining scenarios, traffic classification needs more attentions in terms of its underlying characteristics. A conventional definition of flow is 5-tuple (source IP, source port, destination IP, destination port, protocol), which uniquely identifies a data stream that two hosts communicate with each other in a certain time period.

As depicted above, the state-of-the-art machine learning algorithms may achieve good accuracy, e.g. SVM, on flow based traffic classification. However, these approaches are so computationally expensive that they can be hardly put into practical use unless we are content with sacrificing accuracy. Extreme learning Machine (ELM) [20,21] is a rapidly developing learning theory proposed for generalized single-hidden layer feed-forward networks (SLFNs) with distinguishing characteristics of (1) fast learning speed compared to traditional gradient-based algorithms, (2) good generalization performance on predicting multi-class labels or regression values, (3) free of human-intervened tuning parameters with randomly generated hidden node parameters (e.g. random input weights and hidden biases). In recent years, ELM theory has made considerable progress, which inspired us a lot when in the face of traffic classification, since all of these characteristics of ELM could meet the need of the tremendous growth of the infrastructure of modern Internet. Therefore, we propose an algorithm which extends the original ELM theory and framework with L1-norm minimization of the output weights vector. In this paper, our main contributions can be summarized below:

- Proposing an L1-norm minimization extreme learning machine algorithm to exploit the intrinsic data patterns of network.
- Employing an effective preprocessing including flow features extraction from raw network data trace and labeling the flows.
- Generating the labeled anomaly detection data sets with ground truth from WIDE project.

The remainder of this paper is organized as follows. After reviewing the network traffic flows in Section 2, we describe the algorithm L1-ELM in Section 3. Section 4 describes the preprocessing details, and, we evaluate our methods in Section 5. Section 6 concludes this paper.

2. Sparse representation and its application on ELM

The hidden layer neurons have powerful generalization abilities. However, due to the randomness of the input weights, many neurons may be closely correlated. Regularization is very necessary to prevent the model from over-fitting and improve the generalization capability.

L1-norm regularization has been extensively applied for its sparsity-induced capability, when training samples have high dimensionality. Nevertheless, it draws great attentions in the optimization research field. As the L1-norm regularization will lead the problem to a non-smooth and non-differentiable constrained optimization one, the problem will become much more challengeable to solve. The state-of-the-art methodologies for solving the L1-norm optimization problem can be categorized into four categories [22]. The first methodology solves the problem as a non-smooth optimization problem through sub-gradient based algorithms. The second methodology approximates the L1-norm term with a smooth formulation, so the smooth optimization algorithms can solve the problem directly. The third methodology reformulates the problem into a smooth constraint smooth optimization problem by introducing extra variables. The fourth methodology casts the problem as a smooth objective function optimization problem with a L1-ball constraint, which is applied in this paper, and it can be formulated as

$$\min_{\mathbf{x}:\|\mathbf{x}\|_1 \le z} : F(\mathbf{x}) \tag{1}$$

where $F(\cdot)$ is a smooth loss function and $z \in \mathbb{R}^+$. The building block of this methodology is to apply Euclidean projection onto the L1-ball [23,24]. In the optimization process of the L1-ELM proposed in this paper, through casting Euclidean projections as root finding problems associated with specific auxiliary function, this problem can be solved in linear time via bisection [25].

As of the popularity of ELM theories, many works have been done fully utilizing the advantages of ELM. Parallel computing in a distributed environment is effective when in the face of computation-intensive scenarios. Li et al. [26] use MapReduce model [27] to parallelize the ELM computation across large-scale clusters of machines. Benoît et al. [28] proposed a feature selection method for nonlinear models with ELM. There also exists some works that have some common ideas with ours. Decherchi et al. [29] address the implementation of the powerful ELM model on reconfigurable digital hardware. To obtain the sparse hidden neurons, they introduce a new optimization problem with hinge loss function and L1-norm regularization. The optimization problem is

$$\min_{\mathbf{w}} : \sum_{i=1}^{N} L(y_i, h(\mathbf{x}_i)\cdot\mathbf{w}) + \lambda\|\mathbf{w}\|_1 \tag{2}$$

where $h(\mathbf{x}_i)$ is the ith row of matrix H, which will be explained in the next section. $(\mathbf{x}_i, y_i)$ is the ith training sample and its corresponding target, and $L(\cdot)$ is the hinge loss function. After solving this problem, sparse hidden neurons are selected in accordance with nonzero terms in the optimal **w**. When the selections are done, the original complete ELM process has to be finished afterwards. The main differences between their work and L1-ELM lie in: firstly, their contribution does not change the ELM theory itself, yet L1-ELM can obtain the sparse neurons and output weights vector in the meantime of training process, i.e. the forms and meanings of the two optimization problems are totally different. Secondly, the optimization process in [28] applies a conventional convex optimization algorithms, say, simplex method or interior point methods [30], while L1-ELM could achieve the convergence rate of $O(1/k^2)$, w.r.t. L1-norm regularization.

Miche et al. [31] proposed a methodology named optimally pruned extreme learning machine (OP-ELM), which is based on the original ELM algorithm. They firstly construct a SLFN using the ELM algorithm; then, OP-ELM apply the MRSR [32] algorithm to obtain a ranking of the hidden neurons. MRSR is an extension of the least angle regression (LARS) algorithm [33], and it is a variable ranking method, rather than directly selecting variables with a LASSO [34] solution. At the third step, OP-ELM prunes less useful neurons through leave-one-out validation. As we discussed above, the intrinsic mechanisms between OP-ELM and L1-ELM are different, although they both construct a SLFN with sparse hidden neurons. OP-ELM prunes less useful neurons by leave-one-out validation after ranking the neurons through modified LARS algorithm, while L1-ELM can obtain the sparse neurons after the objective function is optimized immediately. Thereafter, we compare these two methodologies with the convergence time and accuracy in the experiment.

Since Nesterovs method [35] is the one of the optimal first-order black-box methods for smooth convex optimization, its convergence rate can achieve $O(1/k^2)$, where k is the number of iterations. Efficient Euclidean Projection [25], which plays a building block role in the L1-ELM, makes L1-ELM achieve the convergence rate of $O(1/k^2)$, although the objective function is non-smooth other than smooth.

3. L1-minimization ELM (L1-ELM)

In this section, we propose L1-ELM algorithm, which takes the most advantages of ELM theory like fast training speed and good generalization performance, and at the same time, offers us with additional characteristics. We introduce this algorithm right after reviewing the elementary theories of ELM.

3.1. Extreme learning machine theory

Extreme learning machine [36,20] was originally proposed for single hidden layer feedforward neural networks and was then extended to the generalized single hidden layer feedforward networks where the hidden layer need not be neuron alike [37]. Fig. 1 shows the structure of three layer SLFN. The most distinguishable feature of ELM is that its hidden layer parameters are assigned randomly, and for one output node SLFNs with m hidden layer nodes, its output function can be easily formulated as

$$f_{output}(\mathbf{x}) = \sum_{i=1}^{m} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the input variable, $\boldsymbol{\beta} = [\beta_1, \ldots \beta_m]^T$ is the vector of output weights of hidden layer, and $h_i(\mathbf{x})$ is the ith activation function with respect to its input weights vector $\mathbf{w}_i$ and bias b_i. $\mathbf{h}(\mathbf{x})$ can also be considered as feature mapping function, as it maps the input samples from n-dimensional space to the m-dimensional hidden layer feature space.

As a learning algorithm, ELM tends to reach the smallest training error. For N training samples, $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_N]$, the learning process is to

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 \tag{4}$$

where $\mathbf{H}$ is hidden layer output matrix,

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{pmatrix} = \begin{pmatrix} h_1(\mathbf{x}_1) & \ldots & h_m(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \ldots & h_m(\mathbf{x}_N) \end{pmatrix} \tag{5}$$

And $\mathbf{Y} = [y_1, \ldots y_N]^T$ is the target vector.

According to the solution to least-squares problem, the optimal solution $\boldsymbol{\beta}^*$ to (4) can be analytically resolved:

$$\boldsymbol{\beta}^* = \mathbf{H}^{\dagger}\mathbf{Y} \tag{6}$$

Where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$.

3.2. L1-norm minimization ELM

Although least square estimate of the weights vector $\boldsymbol{\beta}$ has the smallest variance among all linear unbiased estimates, it still has large variance against biased estimates such as ridge regression. Hence, ELM algorithm has better generalization performance than that of the algorithm reducing the training error solely. Inspired by ELM, we consider reaching the smallest training error and L1-norm minimization of output weights vector through the following minimization:

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 \quad \text{and} \quad \|\boldsymbol{\beta}\|_1 \tag{7}$$

Additionally, through L1-norm minimization of output weights vector $\boldsymbol{\beta}$, we can obtain sparse hidden layer nodes which would reduce the complexity of the model. For the sake of convenient analysis, we reformulate (7) as following:

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{8}$$

which is equivalent to the constraint optimization problem:

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 \quad \text{s.t.} \ \|\boldsymbol{\beta}\|_1 \le z \tag{9}$$

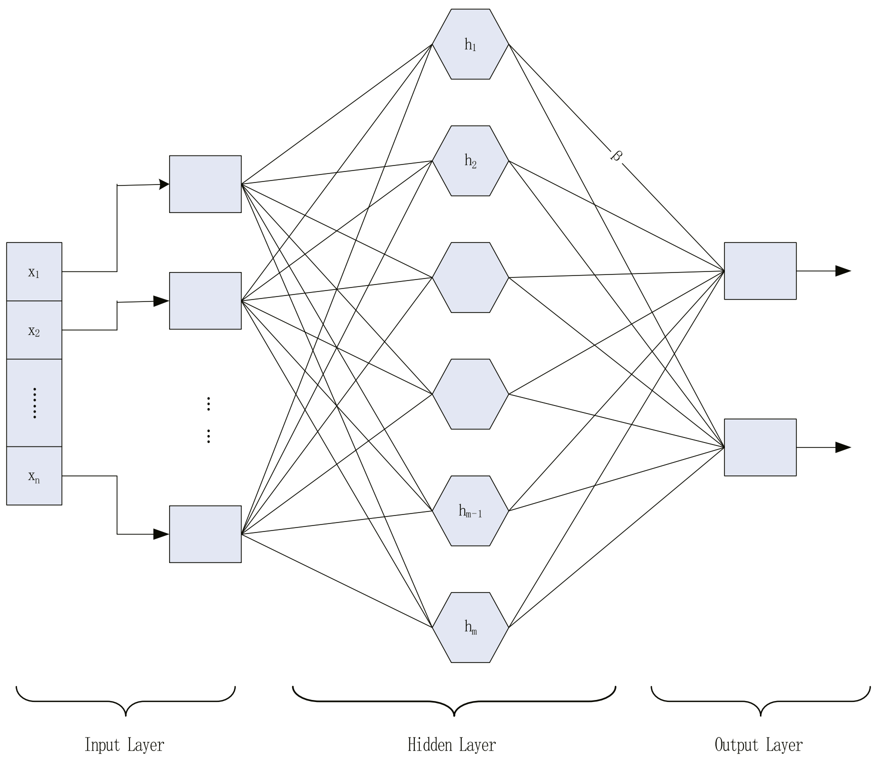

Fig. 1. Single hidden layer feedforward neural network.

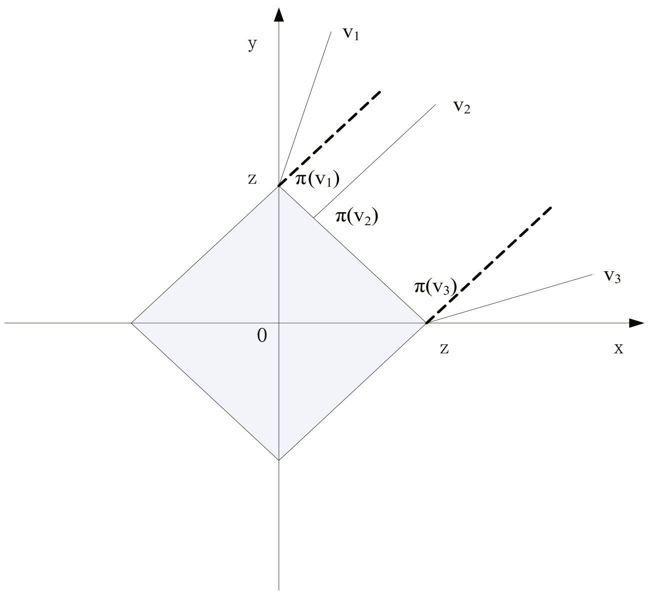

Fig. 2. Euclidean projection.

When applying first-order method (e.g. gradient descent and Nesterov's optimal method) to solve (9), one key building block is Euclidean projection onto the L1-ball [25]. Fig. 2 is an illustration of the Euclidean projection onto the L1-ball. The Euclidean projection problem can be defined as

$$\pi_G(\boldsymbol{v}) = \arg\min_{\boldsymbol{\beta}\in G} \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{v}\|_2^2 \tag{10}$$

Where $G=\{\boldsymbol{\beta}\in\mathbb{R}^m \mid \|\boldsymbol{\beta}\|_1 \le z\}$ is the L1-ball, the Lagrangian function of the Euclidean projection problem is

$$L(\boldsymbol{\beta},\alpha)=\frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{v}\|_2^2+\alpha(\|\boldsymbol{\beta}\|_1-z) \tag{11}$$

where α is the Lagrangian variable for the constraint. Let $\boldsymbol{\beta}^*$ be the primal optimal point, and α^* be the dual optimal point. The primal and dual optimal point should satisfy $\|\mathbf{x}^*\|_1 \le z$ and $\alpha^* \ge 0$. Since L1-ball constraint satisfies Slater's condition, strong duality holds, and the primal and dual optimal values are equal. Assuming that the dual optimal point α^* is known, the primal optimal point can be computed by the following problem:

$$\boldsymbol{\beta}^* = \arg\min_x L(\boldsymbol{\beta},\alpha^*) \tag{12}$$

Since the equation above has a unique solution, and variables, i.e. $\mathbf{x}^*$ and α^* are decoupled, we have

$$\beta_i^* = \arg\min_{\beta_i} \frac{1}{2}(\beta_i - v_i)^2 + \alpha^*(|\beta_i| - z) \tag{13}$$

which leads to

$$\beta_i^* = \mathrm{sgn}(v_i)\max(|v_i|-\alpha^*, 0) \tag{14}$$

According to (14), since the primal optimal point $\boldsymbol{\beta}^*$ can be obtained after the dual optimal point α^* is resolved, Liu et al. [25] proposed an improved bisection for the efficient computation of the dual optimal point, which was solved as a root finding problem of the auxiliary function $g(\alpha)$ formulated below:

$$g(\alpha) = \sum_{i=1}^{m} \max(|v_i| - \alpha, 0) - z \tag{15}$$

Through casting Euclidean projections as root finding problems associated with specific auxiliary function, this problem can be solved in linear time via bisection [25]. L1-norm minimization ELM not only maximally inherits the key features of ELM, which include fast learning speed, good generalization performance and little human intervened tuning of parameters, but also extends the theory of sparse hidden nodes selection, that simplifies the complexity of built model, which is critical in the face of online traffic classification and anomaly detection. Since the benchmark datasets for the field research are scarce, we introduce the data preprocessing stage in the next section.

4. Data preprocessing and labeling

As we mentioned before, in the traffic classification and anomaly detection research field, owing to the privacy and legal concerns, we are short of the benchmark datasets which could be used to verify our method.

When we mention the data preprocessing step of the flow based traffic classification, we are focusing on deriving the flow features through packet header statistics and labeling the traffic flows through the discovered network anomaly community. Features thoroughly characterizing the flows could lend researchers the most discriminative power to harness the data trace for classifying the unseen flows. Hence, the more detailed the features could be, the more reliable model would be built to have good generalization performance. Moore et al. [38] provided comprehensive discriminators to fully characterize the flow classes. These features are all calculated through complete TCP flows, and there are 248 features derived from packet header by taking the advantage of the tools like tcptrace, tcpdemuxerand etc. Fig. 3 demonstrates the preprocessing progress of the raw network packets. Firstly, a coarse-grained aggregation is adopted to form flows in terms of the 5-tuple elements found in every packet header. Secondly, Moore's method is applied for the features abstracting of raw packets flows. Thirdly, anomalous community based labeling algorithm is proposed to label the flows feature vectors generated in the second step.

After the discriminators of flows have been derived, we should label the objects to make the classifiers fully functional. The corresponding community structure is depicted in Fig. 4. We could benefit a lot from the anomalous communities, when we label the flows. Closely focusing on the anomalous community, the algorithm can discover the anomaly connections, i.e. anomalous flows, after the core nodes are found in the community.

5. Experiments

In this section, we first evaluate the proposed method L1-ELM by comparing it with the algorithms of original ELM and OP-ELM on synthetic datasets for regression and three UCI datasets for classification. After that, we will introduce the datasets and metrics that used to test the anomaly detection effects. Finally, thorough experiments are performed to test the effectiveness of L1-ELM.

5.1. Comparisons between ELM, OP-ELM and L1-ELM

First of all, we perform numeric experiments on synthetic data for regression test. Assume $\mathbf{X}\in\mathbb{R}^{N\times 100}$ from a Gaussian distribution $N(0,1)$, and the noise vector $\mathbf{r}\in\mathbb{R}^N$ with $N(0,0.01)$. Let model parameters vector $\boldsymbol{\theta}\in\mathbb{R}^{100}$ of $N(1,5)$, then the target values are generated by

$$y_n = \langle \mathbf{x}_n, \boldsymbol{\theta}\rangle + r_n \tag{16}$$

where $\{\mathbf{x}_n, y_n\}_{n=1}^N$ ($N = 1000$–10000) will be used and r_n is the nth entry of noise $\mathbf{r}$. In each test, $N/2$ samples randomly chosen from the whole set are taken as training samples and rest as testing

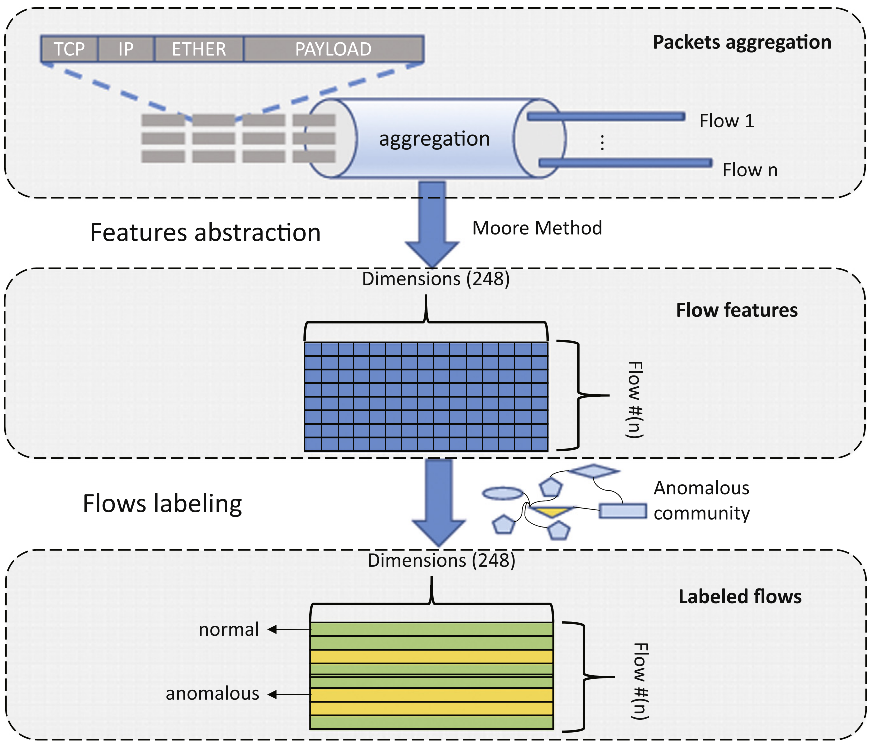

Fig. 3. Preprocessing procedure.

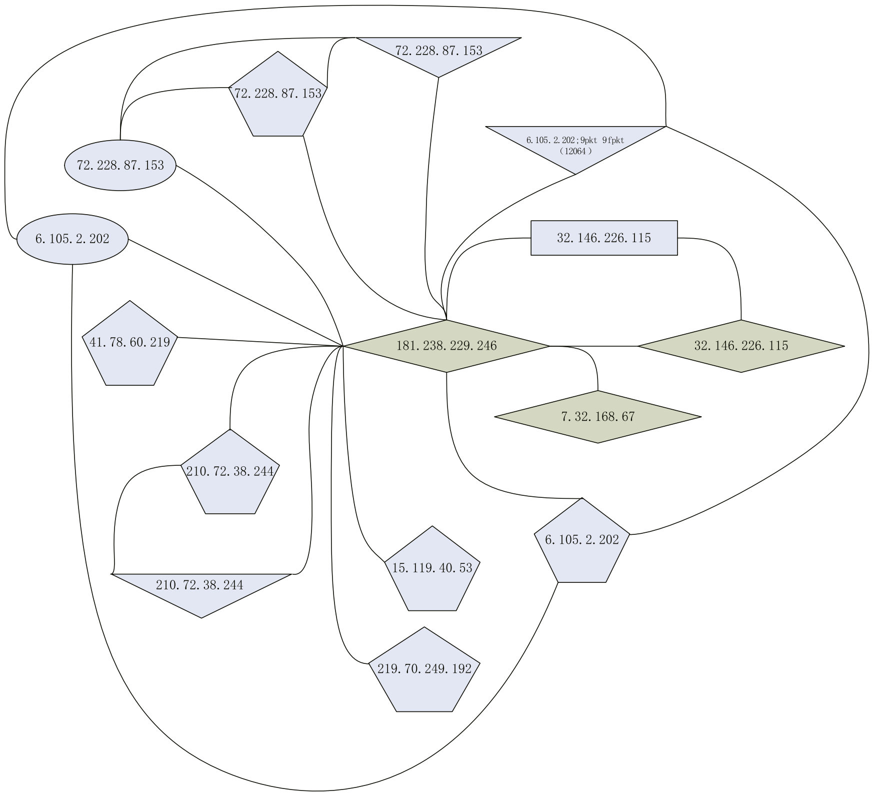

Fig. 4. Anomalous flows community structure.

samples, and all results are averaged over 10 times. As OP-ELM has no parameter to tune, the parameter lambda is tuned to make the L1-ELM have roughly the same number of hidden neurons as OP-ELM does.

From Table 1, we can see that the training time, i.e. the optimization time of all the three methodologies becomes slower with the increase of training samples. Although the training of L1-ELM is still a little slower than ELM, L1-ELM can keep up with ELM at the same order of magnitude.

The testing time reveals that, for L1-ELM, the less hidden neurons are selected, the less time is consumed for the testing compared with ELM. The strange thing is, for OP-ELM, no matter

Table 1
Regression test over 10 synthetic data sets.

N	RMSE #nonzero of β			Training time Testing time		
	L1-ELM	OP-ELM	ELM	L1-ELM	OP-ELM	ELM
1000	3.0537 39	2.9414 60	**2.7355** 100	0.0768 1.31E−04	0.1655 1.69E−03	**0.0263** **6.52E−05**
2000	2.8523 57	3.0608 70	**2.7078** 100	0.0149 **7.58E−05**	0.2822 6.62E−04	**0.0096** 1.01E−04
3000	2.7054 73	2.7889 65	**2.5240** 100	0.0233 **4.47E−05**	0.5456 5.50E−04	**0.0142** 9.83E−05
4000	2.5493 77	3.0417 65	**2.4933** 100	0.0410 **1.06E−04**	0.8641 9.31E−04	**0.0176** 1.62E−04
5000	2.8368 83	3.2115 85	**2.7926** 100	0.0448 **1.10E−04**	1.08733 1.70E−03	**0.0216** 1.99E−04
6000	2.8595 77	3.3321 85	**2.8186** 100	0.0618 **1.13E−04**	1.4120 1.97E−03	**0.0263** 2.13E−04
7000	2.9069 86	3.0616 85	**2.9027** 100	0.0611 2.18E−04	1.8072 2.46E−03	**0.0273** **1.96E−04**
8000	3.0145 89	3.5865 85	**3.0112** 100	0.1126 **1.84E−04**	2.5479 2.62E−03	**0.0322** 2.04E−04
9000	2.1663 91	2.6634 90	**2.1527** 100	0.0601 **2.70E−04**	2.7469 3.25E−03	**0.0381** 2.76E−04
10000	3.1450 81	3.6677 80	**3.0349** 100	0.1042 3.89E−04	3.5565 4.48E−03	**0.0443** **3.03E−04**

Table 2
Basic information about the UCI datasets and classification accuracy comparison.

Data set	# samples/# features	Accuracy		
		L1-ELM	OP-ELM	ELM
Balance	625/4	**0.9250** ± 0.0016	0.9236 ± 0.0017	0.9091 ± 0.0018
pid	768/8	**0.7652** ± 0.0058	0.7023 ± 0.0109	0.7121 ± 0.0085
bupa	345/6	**0.6530** ± 0.0243	0.6087 ± 0.0121	0.5739 ± 0.0348

how many hidden neurons are dropped, the testing time is still much slower than both ELM and L1-ELM.

The testing accuracy is measured through RMSE, which means that the less the RMSE is, the better the result is. Over the datasets, ELM achieves the best testing accuracies among all the datasets among the three methodologies. While N varies from 1000 to 10,000, except one exception when N is 1000, L1-ELM has better RMSE value than OP-ELM.

We also conduct some experiments on the classification effects between these methods. Three datasets from UCI repository [39] are chosen, the basic information is shown in Table 2. Table 2 displays the classification accuracies of the three chosen datasets. In every dataset, two thirds samples are randomly chosen for training, and one third are used for testing. Accuracy is calculated by averaging the 10-fold cross validations results.

From the point view of classification accuracies, L1-ELM has the best results on all of the datasets, while OP-ELM are better than ELM for 2 datasets.

Compared to ELM and L1-ELM, after sacrificing the training time, L1-ELM can not only get the good generalization ability, but also reduce the time for testing. The good performance would be mainly attributed to sparse neurons selection and model optimization which may prevent the model from over-fitting.

5.2. Datasets

The WIDE project [40] maintains a data trace repository, from which, traffic traces were captured from a trans-Pacific backbone link between Japan and the United States. This data trace repository provides raw data traces as well as XML files depicting the anomalous communities.

As anomalies are usually rare compared with normal events in the network. For the purpose of investigating the effectiveness of the proposed algorithm L1-ELM, the time frame selected should be typical, and various anomalies should also be included. We choose one day traffic trace (January 9th, 2011) as our base evaluation dataset, which started capturing at 14:00 pm and ended at 14:15 pm. Table 3 introduces the statistics of this traffic trace.

For the Internet Service Providers and network administrators, it is important to disclose the abnormal behaviors at first moment on the backbone network. In addition, new kind of anomalies emerge in an endless stream, which leaves the administrators an impossible mission to category all the anomalies. Hence, immediately discovering the anomaly is much more meaningful than making it clear what kind of the anomaly actually is. This reason makes the anomaly detection a binary classification problem.

After the preprocessing of the raw traffic trace, we formed a dataset consisted of flow discriminators, in which, every row fully characterizes a flow. We randomly selected approximately 20 K flows, then, uniformly split it into 10 datasets, which are shown in Table 4. The target of each sample is 1 or −1, which stands for normal flow and anomaly flow respectively.

5.3. Metrics

To measure the performance of L1-ELM, we use seven metrics: TP (true positive) rate (i.e. recall), FP (false positive) rate, precision, F-measure, and overall accuracy.

TP rate of each class, which is also called recall is defined as follows:

$$TPR = \frac{TP}{TP+FN} \tag{17}$$

It is the percentage of flows of that class classified correctly as that class, where TP is true positive and FN is false negative.

FP rate of each class is defined as follows:

$$FPR = \frac{FP}{FP+TN} \tag{18}$$

It is the percentage of flows misclassified as that class, where FP is the false positive and TN is true negative.

Table 3
Traffic trace statistics.

# flows	731,362
Seconds of Duration	899.20
# packets	28,944,127
# addresses	420,401

Table 4
10 evaluation datasets.

Data set	# Flows	#Anomaly flows	# Ordinary flows
Mawi01	23,503	21,632	1871
Mawi02	23,504	19,935	3569
Mawi03	23,503	20,143	3360
Mawi04	23,503	20,431	3072
Mawi05	23,503	20,442	3061
Mawi06	23,503	20,219	3284
Mawi07	23,503	20,267	3236
Mawi08	23,503	19,687	3816
Mawi09	23,503	20,221	3282
Mawi10	23,503	20,163	3340

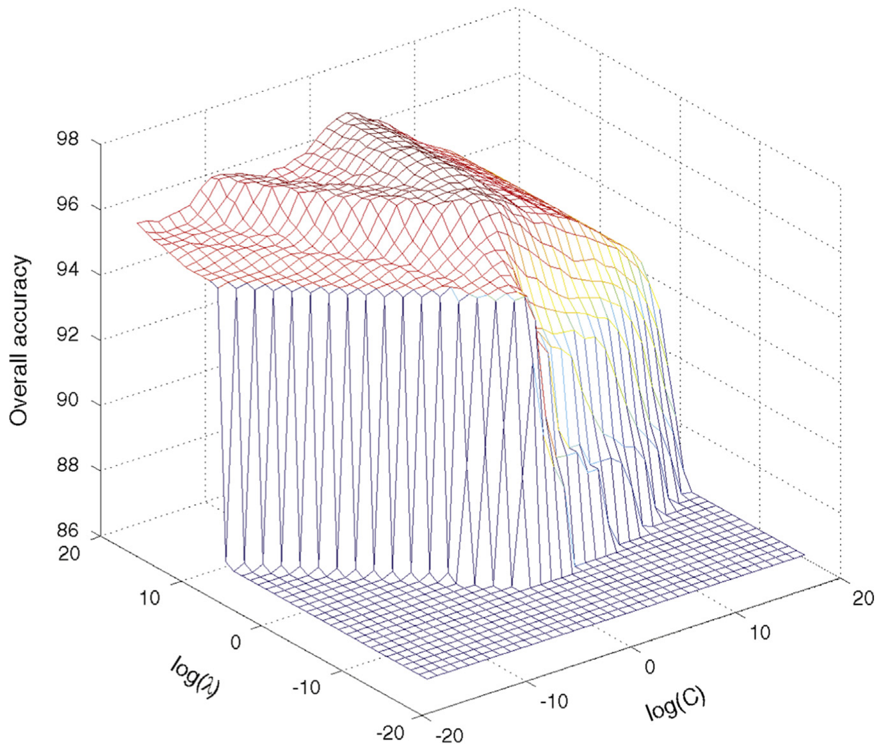

Fig. 5. Parameter selection for SVM.

Precision of each class is defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{19}$$

It is the percentage of flows classified correctly as that class.

F-measure of each class is defined as follows:

$$F\text{−}measure = \frac{2 \times precision \times recall}{precision + recall} \tag{20}$$

It considers both precision and recall in a single metric by taking their harmonic mean.

Overall accuracy is defined as follows:

$$Overall\ accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{21}$$

It is the percentage of all the flows that are classified correctly.

5.4. Experiments on anomaly detection

In this section, in order to show the effectiveness of the L1-ELM algorithm, we compare it with ELM and SVM. For SVM, We use Matlab version of libsvm [41] as the tool. The goals of our experiments are mainly twofold, firstly, to testify the good generalization of the L1-ELM algorithm; secondly, to propose the feasibility of fast flow-based traffic classification and anomaly detection. As a binary classification problem, we primarily focus on the performance of anomaly flows detection, since it is critical for operators and administrators to be aware of the situations of the network and take necessary actions before harmful event would take place.

In all the experiments we conduct, the parameters are set as follows. For SVM, C stands for the cost parameter and γ is the kernel parameter. We use the radial basis function $\exp(-\gamma * |u-v|^2)$ in libsvm. C and γ are both in range of $[2^{-20}, 2^{-9}, \ldots 2^{20}]$

For ELM and L1-ELM, m stands for the number of hidden-layer nodes which varies from 100 to 2000. For L1-ELM λ is the L1 penalty parameter, which is in range of $[2^{-10}, 2^{-9}, \ldots 2^{10}]$. When the penalty factor is given large enough, the output weights $\boldsymbol{\beta}$ is all set to zero, as a consequence, the testing output becomes the same.

Each dataset is split into 10 parts equally. All the metrics in our following experiments are calculated by averaging the 10-fold cross validations results. In every round, 9 parts are used to train the models, and rest one part to test.

Parameter selection has always been a complicated problem. It always scales with the number of parameters and the objective function of the parameters is not generally monotone. In the experiments, we tried a range of combination of two parameters in both SVM and L1-ELM.

As we can see from Figs. 5 and 6, compared with SVM, L1-ELM is less sensitive to parameters without losing much accuracy, which inherited this excellent property of original ELM.

Sparsity is an important metric for evaluating the complexity of learning model. To demonstrate the relationship of sparsity and testing accuracy of L1-ELM, we tried the penalty λ in the range of $[2^{-10}, 2^{-9}, \ldots 2^{10}]$ while holding hidden nodes to 500 on the 5-th data set. As Fig. 7 shows L1-ELM can remain the sparsity of output weights $\boldsymbol{\beta}$ without losing much accuracy, e.g. only 15 nonzero elements can achieve almost as well performance as 500 does. According to [42], testing time is more valuable than training time in practice, so a sparse architecture can significantly reduce the test time. This point has been described in Section 5.1.

As shown in Fig. 8, we compare the TP rate of the anomaly flows between ten datasets using L1-ELM, ELM and SVM. The corresponding FP rate comparison is shown in Fig. 9. From the figures, we can tell that L1-ELM achieved the highest TP rate in seven datasets, and SVM has the worst performance compared with other two, which means that L1-ELM have better generalization performance than ELM and SVM. At the same time, L1-ELM achieved the highest FP rate in 8 datasets, where SVM and ELM have relatively equal FP rate. High FP rate of anomaly flows is not ideal, because higher FP rate means more anomaly flows are classified as ordinary ones, which is sometimes harmful to the network security and operations. Fig. 10 exhibits

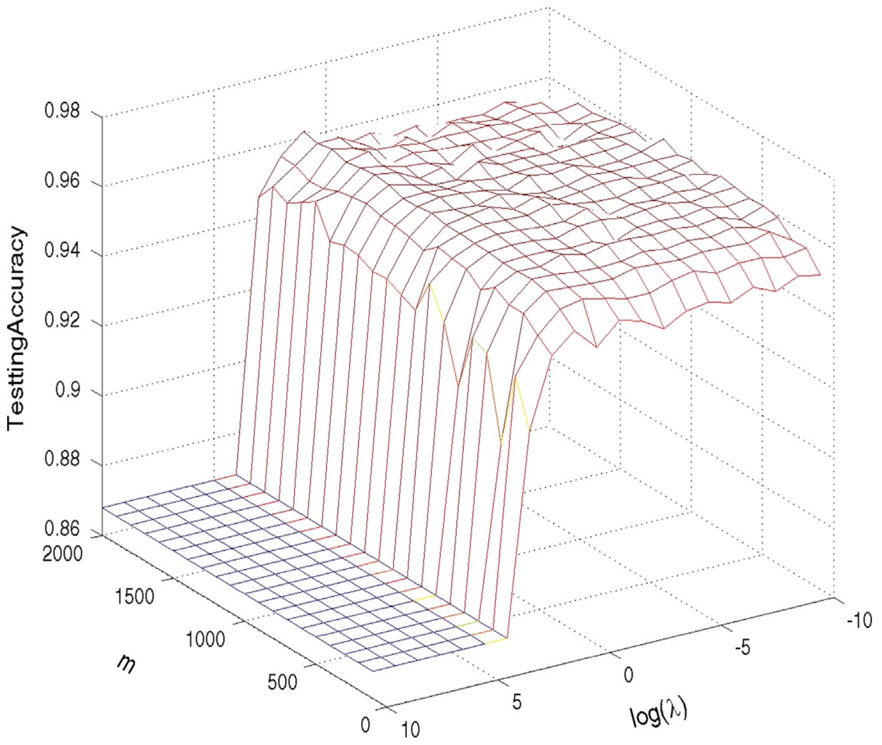

Fig. 6. Parameter selection for L1-ELM.

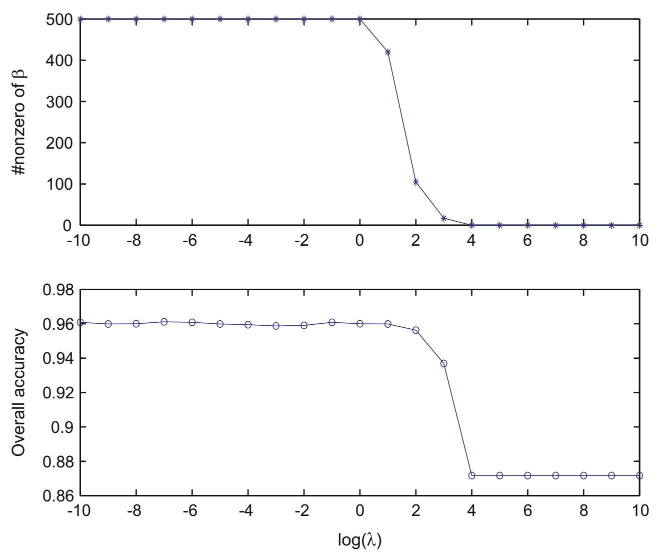

Fig. 7. Sparsity and accuracy of L1-ELM.

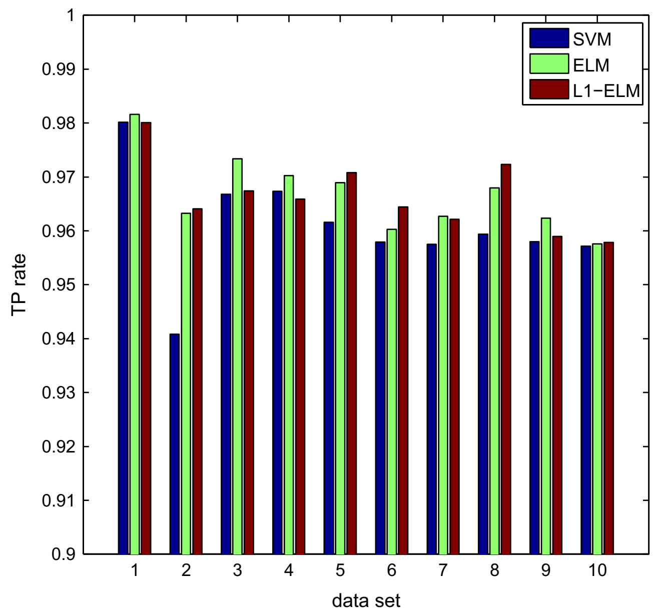

Fig. 8. TP rate of anomaly flows.

the ROC curve of TPR vs. FPR, at various threshold settings for the fifth data set. It can be directly inferred from the figure that, the AUCs for L1-ELM and ELM are comparatively equal and are both larger than that for SVM, which makes SVM a suboptimal model compared to the others.

From the perspective of anomaly detection, another two metrics: precision and recall are very important for the analysis. We usually use these two metrics together to confirm the performance of the classification process. Table 5 depicts the recall value of anomaly flows in ten datasets, whereas Fig. 11 pictures the corresponding precision value.

F-measure depicted in Fig. 12 shows the combinations of recall and precision, which is the weighted harmonic mean of precision and recall. From the demonstrations, we can see that ELM and L1-ELM both outperform the SVM on nearly all of the datasets, and L1-ELM can achieve better performance than ELM with the help of sparsity inducing ability. From the point of overall accuracy, as shown in Table 6, ELM prevails over the other two algorithms in 8 datasets. L1-ELM outperforms SVM and ELM in the rest 2 datasets. The reason may lie at that the model learned by ELM contains more hidden nodes than L1-ELM does, thus retains more representative capability. What is more, L1-ELM can be regarded as a simplified version of ELM which may subject to a bit worse performance.

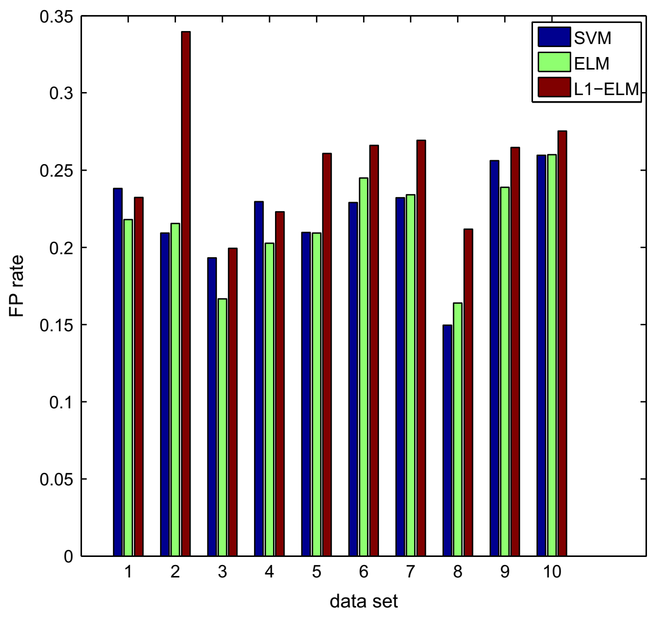

Fig. 9. FP rate of anomaly flows.

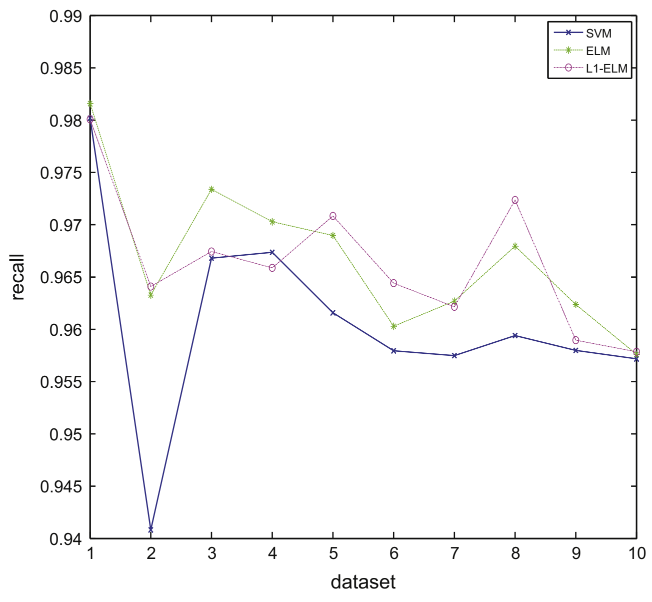

Fig. 11. Precision comparison.

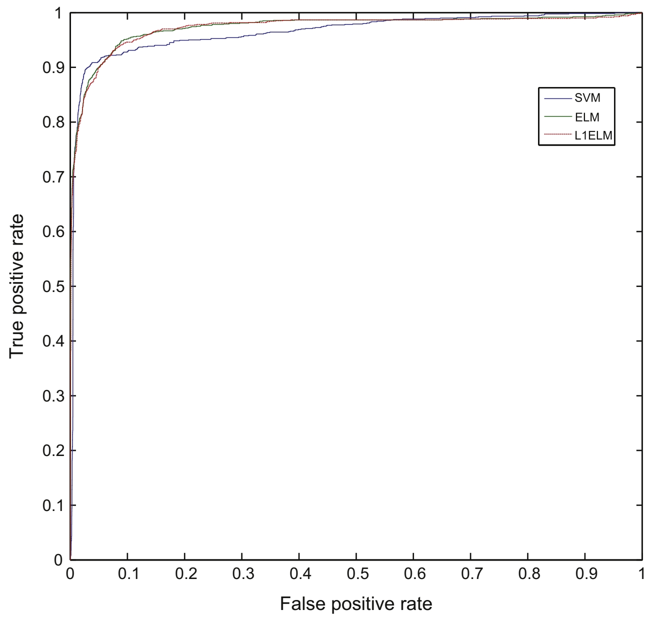

Fig. 10. ROC for classification by three algorithms.

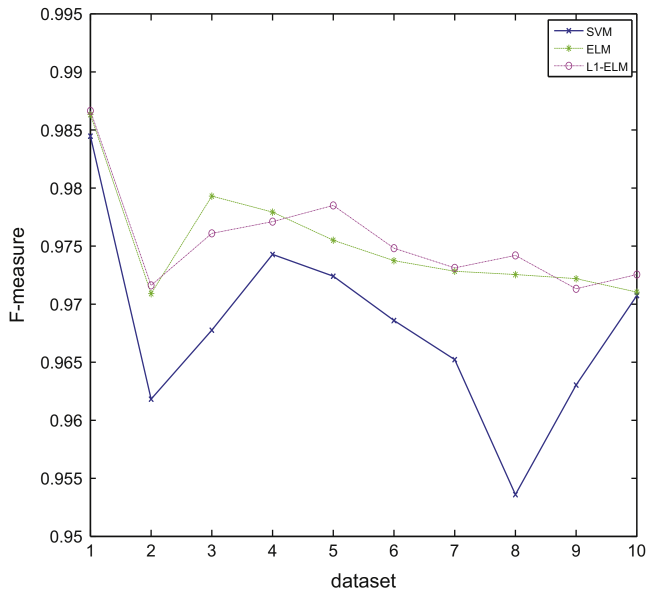

Fig. 12. F-measure comparison.

Table 5
Recall value of anomaly flows over 10 datasets.

Data set	SVM	ELM	L1-ELM
Mawi1	0.9892 ± 0.0024	0.9938 ± 0.0014	**0.9939** ± 0.0019
Mawi2	**0.9832** ± 0.0022	0.9780 ± 0.0032	0.9790 ± 0.0035
Mawi3	0.9796 ± 0.0036	0.9852 ± 0.0035	**0.9864** ± 0.0022
Mawi4	0.9813 ± 0.0026	0.9845 ± 0.0029	**0.9863** ± 0.0021
Mawi5	0.9857 ± 0.0032	0.9871 ± 0.0017	**0.9887** ± 0.0032
Mawi6	0.9809 ± 0.0027	0.9881 ± 0.0020	**0.9891** ± 0.0025
Mawi7	0.9732 ± 0.0023	0.9855 ± 0.0023	**0.9858** ± 0.0022
Mawi8	0.9686 ± 0.0044	0.9759 ± 0.0046	**0.9787** ± 0.0036
Mawi9	0.9720 ± 0.0035	0.9800 ± 0.0042	**0.9819** ± 0.0032
Mawi10	0.9857 ± 0.0020	0.9888 ± 0.0035	**0.9889** ± 0.0029

Table 6
Overall accuracy over 10 datasets.

Data set	SVM	ELM	L1-ELM
Mawi1	0.9706 ± 0.0034	**0.9759** ± 0.0034	0.9751 ± 0.0033
Mawi2	0.9335 ± 0.0081	**0.9503** ± 0.0029	0.9486 ± 0.0031
Mawi3	0.9482 ± 0.0029	**0.9612** ± 0.0058	0.9597 ± 0.0027
Mawi4	0.9531 ± 0.0038	**0.9620** ± 0.0029	0.9601 ± 0.0026
Mawi5	0.9510 ± 0.0040	**0.9598** ± 0.0035	0.9580 ± 0.0039
Mawi6	0.9471 ± 0.0031	**0.9588** ± 0.0038	0.9585 ± 0.0053
Mawi7	0.9396 ± 0.0036	**0.9548** ± 0.0036	0.9523 ± 0.0057
Mawi8	0.9015 ± 1.0067	0.9509 ± 0.0030	**0.9513** ± 0.0056
Mawi9	0.9355 ± 0.0051	0.9480 ± 0.0058	**0.9480** ± 0.0048
Mawi10	0.9452 ± 0.0037	**0.9538** ± 0.0060	0.9522 ± 0.0034
Mawi10	0.9452 ± 0.3662	**0.9538** ± 0.0060	0.9522 ± 0.0034

From Tables 5 and 6, we can see that some of results do not look significantly different. This may be caused by two reasons. Firstly, these three methods are all very effective to achieve considerable accuracy which make it hard to tell the differences significantly. Secondly, the most important goal for L1-ELM is not accuracy improvement, but reducing the testing time while maintaining reasonable level of accuracy.

These results described above keep consistence with our analysis that L1-ELM could achieve approximate accuracy as ELM does while improving the testing speed.

6. Conclusion

Network traffic classification and anomaly detection present much challenges to administrators for the privacy and legal concerns as well as high demand for computational resources. In this paper, we propose a L1-norm minimization ELM algorithm, which provides sparse hidden layer output weights vector that helps reducing the model complexity. We extract sufficient features that can fully characterize the network flows from raw data trace captured at backbone network. The datasets generated with those features helped to evaluate the L1-ELM, which retains sparsity and decreases testing time while keeping accuracy on a reasonable level as ELM and in certain situations improves generalization performance.

In the field of traffic classification and anomaly detection research, the classification is very resource-intensive, thus, we plan to make effort on selecting sparse and robust features before classifications to get rid of the most resource requirements.

Acknowledgments

This research was supported by the National Technology Research and Development Program of China (863 Program) with grant no. 2012AA01A510, and the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20140073). The authors would also want to express the sincere appreciation to the anonymous reviewers for their patient work and suggestions.

References

[1] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Lee, Internet traffic classification demystified: myths, caveats, and the best practices, in: Proceedings of the 2008 ACM CoNEXT Conference, ACM, 2008, p. 11.

[2] A.W. Moore, K. Papagiannaki, Toward the accurate identification of network applications, in: Passive and Active Network Measurement, Springer, 2005, pp. 41–54.

[3] S. Sen, O. Spatscheck, D. Wang, Accurate, scalable in-network identification of p2p traffic using application signatures, in: Proceedings of the 13th International Conference on World Wide Web, ACM, 2004, pp. 512–521.

[4] T. Karagiannis, K. Papagiannaki, M. Faloutsos, Blinc: multilevel traffic classification in the dark, in: ACM SIGCOMM Computer Communication Review, vol. 35, ACM, 2005, pp. 229–240.

[5] P. Haffner, S. Sen, O. Spatscheck, D. Wang, Acas: automated construction of application signatures, in: Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data, ACM, 2005, pp. 197–202.

[6] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, G. Varghese, Network monitoring using traffic dispersion graphs (tdgs), in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ACM, 2007, pp. 315–320.

[7] A. Dainotti, A. Pescapè, K.C. Claffy, Issues and future directions in traffic classification, IEEE Netw. 26 (1) (2012) 35–40.

[8] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, M. Mellia, Reviewing traffic classification, in: Data Traffic Monitoring and Analysis, Springer, 2013, pp. 123–147.

[9] T. Auld, A.W. Moore, S.F. Gull, Bayesian neural networks for internet traffic classification, IEEE Trans. Neural Netw. 18 (1) (2007) 223–239.

[10] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, Traffic classification through simple statistical fingerprinting, ACM SIGCOMM Comput. Commun. Rev. 37 (1) (2007) 5–16.

[11] J. Erman, M. Arlitt, A. Mahanti, Traffic classification using clustering algorithms, in: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, ACM, 2006, pp. 281–286.

[12] J. Erman, A. Mahanti, M. Arlitt, C. Williamson, Identifying and discriminating between web and peer-to-peer traffic in the network core, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 883–892.

[13] Z. Li, R. Yuan, X. Guan, Accurate classification of the internet traffic based on the SVM method, in: IEEE International Conference on Communications, 2007, ICC'07, IEEE, 2007, pp. 1373–1378.

[14] A. McGregor, M. Hall, P. Lorier, J. Brunskill, Flow clustering using machine learning techniques, in: Passive and Active Network Measurement, Springer, 2004, pp. 205–214.

[15] A.W. Moore, D. Zuev, Internet traffic classification using Bayesian analysis techniques, in: ACM SIGMETRICS Performance Evaluation Review, vol. 33, ACM, 2005, pp. 50–60.

[16] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, Class-of-service mapping for QOS: a statistical signature-based approach to ip traffic classification, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, ACM, 2004, pp. 135–148.

[17] N. Williams, S. Zander, G. Armitage, A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification, ACM SIGCOMM Comput. Commun. Rev. 36 (5) (2006) 5–16.

[18] V. Carela-Español, P. Barlet-Ros, M. Solé-Simó, A. Dainotti, W. de Donato, A. Pescapé, K-dimensional trees for continuous traffic classification, in: Traffic Monitoring and Analysis, Springer, 2010, pp. 141–154.

[19] A. Dainotti, A. Pescapé, H.-c. Kim, Traffic classification through joint distributions of packet-level statistics, in: Global Telecommunications Conference (GLOBECOM 2011), IEEE, 2011, pp. 1–6.

[20] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.

[21] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (2) (2012) 513–529.

[22] J. Liu, J. Chen, J. Ye, Large-scale sparse logistic regression, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 547–556.

[23] A.Y. Ng, Feature selection, l 1 vs. l 2 regularization, and rotational invariance, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 78.

[24] J. Duchi, S. Shalev-Shwartz, Y. Singer, T. Chandra, Efficient projections onto the l 1-ball for learning in high dimensions, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 272–279.

[25] J. Liu, J. Ye, Efficient euclidean projections in linear time, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 657–664.

[26] D. Li, Z. Pan, Z. Deng, Y. Zhang, Large scale extreme learning machine using mapreduce, Int. J. Digit. Content Technol. Appl. 6 (20) (2012) 62–65.

[27] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[28] F. Benoît, M. Van Heeswijk, Y. Miche, M. Verleysen, A. Lendasse, Feature selection for nonlinear models with extreme learning machines, Neurocomputing 102 (2013) 111–124.

[29] S. Decherchi, P. Gastaldo, A. Leoncini, R. Zunino, Efficient digital implementation of extreme learning machines for classification, IEEE Trans. Circuits Syst. II: Express Briefs 59 (8) (2012) 496–500.

[30] D.G. Luenberger, Linear and Nonlinear Programming, Springer, US, 2003.

[31] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, Op-elm: optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (1) (2010) 158–162.

[32] T. Similä, J. Tikka, Multiresponse sparse regression with application to multidimensional scaling, in: Artificial Neural Networks: Formal Models and their Applications–ICANN 2005, Springer, 2005, pp. 97–102.

[33] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, Ann. Stat. 32 (2) (2004) 407–499.

[34] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B (Methodol.) (1996) 267–288.

[35] Y. Nesterov, I.E. Nesterov, Introductory Lectures on Convex Optimization: a Basic Course, vol. 87, Springer, 2004.

[36] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks (IJCNN2004), vol. 2, Budapest, Hungary, 25–29 July, 2004, pp. 985–990.

[37] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[38] A. Moore, D. Zuev, M. Crogan, Q. Mary, Discriminators for Use in Flow-based Classification, Technical report, Intel Research, Cambridge, 2005.

[39] A. Asuncion, D.J. Newman, Uci machine learning repository [⟨http://www.ics.uci.edu/~mlearn/mlrepository.html⟩]. irvine, ca: University of California, School of Information and Computer Science, 2007.

[40] K. Cho, K. Mitsuya, A. Kato, Traffic data repository at the wide project, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, USENIX Association, 2000, pp. 51.

[41] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (3) (2011) 27.

[42] D. Yu, L. Deng, Efficient and effective algorithms for training single-hidden-layer neural networks, Pattern Recognit. Lett. 33 (5) (2012) 554–558.

Yibing Wang received the B.S. and M.S. degrees from National University of Defense Technology, Changsha, China, in 2006 and 2010 respectively. He is now a Ph.D. student at PLA University of Science and Technology, Nanjing, China. His main research interests include extreme learning machine theories, multi-kernel learning, feature selection, social media networks, and network securities.

Dong Li received the B.S. and M.S. degrees from China University of Geosciences and PLA University of Science and Technology respectively in 2010 and 2013. He is now a Ph.D. student in PLA University of Science and Technology, Nanjing. His main research interests include large scale data processing and machine learning.

Yi Du received the B.S. degree in computer science and M.S. degree in computer science and application from Institute of Communications Engineering, Nanjing, China, in 1993 and 1997, respectively, and Ph.D. degree in computer science at PLA university of Science and Technology, Nanjing, China in 2000. At present, he is a senior engineer in network management center of CESEC in Beijing. His current research interests mainly include data mining, pattern recognition and network management.

Zhisong Pan received the B.S. degree in computer science and M.S. degree in computer science and application from the PLA Information Engineering University, Zhenzhou, China, in 1991 and 1994 respectively, and Ph.D. degree in Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2003. From July 2006 to present, he led several key projects of intelligent data processing for the network management. From July 2011, he has working as a full professor in the PLA University of Science and Technology, China. His current research interests mainly include pattern recognition, machine learning and neural networks.