Iterative-Tuning Support Vector Machine For Network Traffic Classification

Yang Hong[1], Changcheng Huang[1], Biswajit Nandy[2], Nabil Seddigh[2]
1 – Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada
2 – Solana Networks, Ottawa, Ontario, Canada
E-mail: {yanghong, huang}@sce.carleton.ca[1], {bnandy, nseddigh}@solananetworks.com[2]

***Abstract*—Accurate and timely traffic classification is a key to providing Quality of Service (QoS), application-level visibility, and security monitoring for network operations and management. A class of traffic classification techniques have emerged that apply machine learning technology to predict the application class of a traffic flow based on the statistical properties of flow-features. In this paper, we propose a novel iterative-tuning scheme to increase the training speed of the classification algorithm using Support Vector Machine (SVM) learning. Meanwhile we derive the equations to obtain SVM parameters by conducting theoretical analysis of iterative-tuning SVM. Traffic classification is carried out using flow-level information extracted from NetFlow data. Performance evaluation demonstrates that the proposed iterative-tuning SVM exhibits a training speed that is two to ten times faster than eight other previously proposed SVM techniques found in the literature, while maintaining comparable classification accuracy as those eight SVM techniques. In the presence of millions of flows and Terabytes of data in the network, faster training speeds is essential to making SVM techniques a viable option for real-world deployment of traffic classification modules. In addition, network operators and cloud service providers can apply network traffic classification to address a range of issues including semi-real-time security monitoring and traffic engineering.**

Keywords—network management; traffic classification; machine learning; quality of service; support vector machine

I. INTRODUCTION

The increasing demand for cloud-based services has caused a rapid increase in Internet traffic levels and topological complexity. Service providers seek profitable means of delivering value-added, bundled, or personalized network/cloud services to a large number of subscribers. Applications running over the network/cloud are typically classified into different categories, such as bulk data transfer (e.g., file transfer protocol (FTP) and peer-to-peer (P2P) file downloads), cloud service (e.g., cloud computing and database transactions), and real-time streaming (e.g., voice and video), as illustrated by Fig. 1.

There are a number of different reasons why network/cloud operators require accurate classification of the applications or traffic. A good example relates to Quality of Service (QoS) treatment. Different types of network/cloud applications impose inherently different QoS requirements, e.g., low end-to-end delay for interactive applications, high throughput for file transfer applications. Cloud providers and enterprises expect to optimize the utilization of their existing network infrastructures under finite capacity and cost constraints, while ensuring satisfactory performance for various user applications [1, 2]. A critical requirement for treating applications differently on the network/cloud by the network/cloud operator is the ability to accurately identify or classify the applications/traffic.

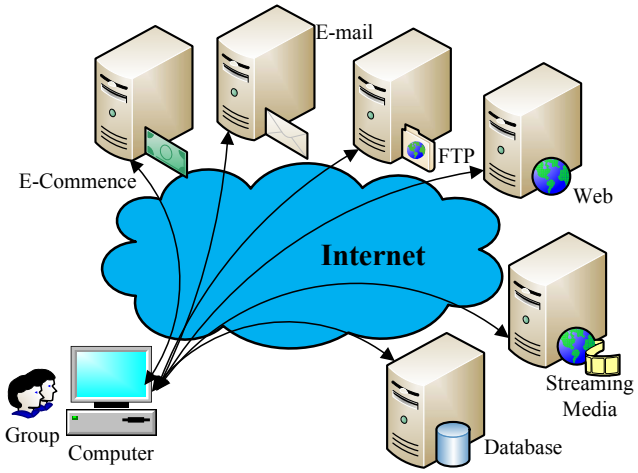

Fig. 1. Different applications running over network.

The traditional approach to network/cloud traffic classification relies on a port-based approach that performs application mapping using the Internet Assigned Numbers Authority (IANA) standardized port numbers. Port-based classification is generally quite accurate in classifying certain types of applications (e.g., E-mail, FTP, and Telnet) that are strictly delivered via static traditional ports. Such techniques have been widely implemented in a variety of traffic analysis tools, e.g., CoralReef [3]. However, the static port-based approach is insufficient for accurate classification of those applications which assign ports dynamically or share popular ports, e.g., P2P traffic or VoIP/Video [4].

The limitation of port-based classification has motivated the development of Deep Packet Inspection (DPI) techniques. DPI inspects the packet application headers and payload to match it against well-known application pattern signatures. DPI has been utilized successfully to identify applications carried by traffic via dynamic ports [5]. However, DPI has certain shortcomings related to scale, cost and reliance on the availability of the packet payload [4]. To augment port-based and DPI classification, new promising approaches have been proposed to identify applications by analyzing the statistical properties of traffic or the characteristics of particular applications based on either the flow-features or the host-behaviors [6]. As a powerful statistical tool, machine learning

technology has been applied to classify network traffic, while achieving high accuracy [6–8]

On the other hand, network traffic patterns fluctuate sharply during busy hours [9]. Building a new classification model based on a combination of a new training data set plus the historical training data set would boost the real-time classification accuracy. Increasing training speed can help cloud service providers and network operators to reach a new classification model for traffic classification more quickly.

The main contributions of this paper are:

- Proposal of an iterative-tuning scheme to increase the training speed of Support Vector Machine (SVM) learning algorithms against the multi-class classification problem;
- Theoretical analysis of proposed iterative-tuning scheme to derive the equations to obtain SVM parameters;
- Application of iterative-tuning SVM to classify network traffic, thereby achieving a better trade-off between classification accuracy and training speed in comparison with other 8 typical SVM learning approaches (as depicted in Fig. 4 later on). In particular, quite different from most existing traffic classification schemes that rely on packet-level information (and which are cost-expensive in network environments with millions of flows), the traffic classification approach proposed in this paper leverages flow-level information available from flow-based protocols such as NetFlow [10].

The rest of the paper is organized as follows. Section II reviews the related work on network traffic classification. Section III proposes the idea of iterative-tuning SVM for traffic classification. Section IV evaluates the validity of iterative-tuning SVM for traffic classification, and carries out performance comparison with other 8 typical SVM algorithms. Conclusions are presented in Section V.

II. Related Work

There is a growing body of work on developing network traffic classification approaches over the past few years, e.g., [1–8, 11–27]. Generally these traffic classification approaches can be grouped into 4 main categories: (i) port-based, (ii) payload-based, (iii) host-behavior-based, and (iv) flow-features-based.

In the introductory section we outlined the limitations faced by port-based schemes.

Payload-based classification schemes (also referred to as DPI) can improve traffic classification accuracy by identifying different services or applications (e.g., P2P file sharing) which share the same registered port number [11, 12]. Commercial tools based on payload-based classification have been developed, e.g., Ellacoya [13] and Packeteer [14]. However, there are a number of issues associated with payload-based classification: (i) DPI cannot accurately identify the traffic application if the payload is encrypted – which is not uncommon [15]; (ii) DPI technology requires powerful hardware and ASIC technology in order to scale to line rates – as a consequence, expensive cost hinders DPI from wide deployment in the cloud platform and enterprise networks.

A. Host-behavior-based Classification Approach

To overcome the limitation of the payload-based approach, a host-behavior-based approach was developed to capture social interaction among the hosts, and detect the encrypted payload sent by the hosts [15–17]. A host-behavior-based approach called "BLINC" captures the behavioral information of a host, in terms of the destinations and ports it communicates with. Then by comparing the captured profile with the historical database which stores host-behavior signatures of application servers, BLINC can identify applications the host is running [15]. Other host-behavior-based approaches were proposed by [16, 17].

Host-behavior-based approaches need to create different databases for different clients (e.g., enterprise, government, hospital, etc.) due to different types of main traffic patterns. Moreover, without reading the payload, host-behavior-based approach cannot identify specific application sub-types. For example, BLINC can identify P2P flows, but it cannot detect the specific P2P protocol (e.g., eDonkey versus Gnutella) [18].

B. Flow-features-based Classification Approach

Presently, port-based approaches are less effective when faced with the increase of applications tunneled through HTTP (e.g., chat, streaming), the constant emergence of new protocols, and the wide-spread use of P2P networking [15].

In order to achieve a better trade-off between classification accuracy and cost, a flow-features-based approach (e.g., [1, 4, 6–8, 18–27]) has been proposed to capture the flow features (e.g., flow duration, number and size of packets per flow, packet inter-arrival time) from the traffic trace. Each flow is characterized by a set of statistical features and associated feature values. Given that different applications exhibit different statistical features, flow-features-based classification can apply machine learning technique to map instances of network flows into different applications [26].

Machine learning algorithms generally fall into the categories of being supervised or unsupervised. Unsupervised approaches group traffic flows into different clusters according to the similarities in their feature values. The clusters are not pre-defined, and no training data is required. Unsupervised learning algorithms detect new applications by automatically grouping known applications into dynamically created clusters [19]. For supervised learning, a classification model is built based on the training dataset which is labeled to represent a group of classes with different feature values. The model is used to predict class membership of an unknown flow by examining its feature values [6]. Typical supervised machine learning techniques adopted by flow-features-based approaches include Naive Bayes [26], Bayesian Network [26], C4.5 Decision Tree [8, 26], Naive Bayes Tree [26], Support Vector Machine (SVM) [27], Nearest Neighbors [1], and Neural Networks [22].

By conducting a thorough experiment on a broad range of network traffic traces, the authors in [6] demonstrate that the SVM algorithm achieves the highest traffic classification accuracy, when compared with the port-based approach, host-behavior-based approach, and flow-features-based approaches based on other machine learning algorithms. Therefore, in this

paper, we propose a flow-features-based approach with SVM learning technique for traffic classification.

III. Iterative-Tuning SVM For Traffic Classification

The most popular machine learning techniques based on Bayes' theorem (e.g., Naive Bayes, Bayesian Network, C4.5 Decision Tree, Naive Bayes Tree) may be trapped into local optimization, because Bayesian estimation makes classification accuracy highly dependent on the prior probability of samples [27]. The training and testing samples in Bayesian estimation may be biased, if a certain class of traffic (e.g., the WWW traffic) constitutes the large majority of the samples [20].

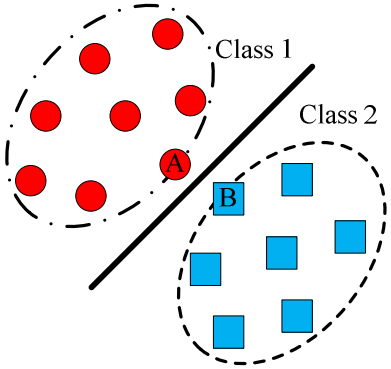

Fig. 2. Hyper-planes constructed by SVM for two different classes.

In order to achieve global optimization, the SVM-based machine learning technique provides optimal statistical classification via a decision function [27]. The general methodology of SVM is very flexible so that it can be customized to meet various application requirements [27].

To realize the goal of classifying traffic flows into different classes which represent different applications, SVM constructs a separate hyper-plane for each traffic class with multiple flow-features, and then maximizes the distance between the closest training data samples of different classes in an n-dimensional flow-feature space, where n denotes the number of flow-features for each traffic class. Fig. 2 illustrates two different hyper-planes that enclose training data samples of two different classes respectively, where a red circle represents a training sample of Class 1, and a blue square represents a training sample of Class 2.

A. SVM Multi–Class Formulation

Numerous approaches have been proposed to perform SVM classification. The basic idea of SVM classification is to produce a classification model by use of training data samples, and then use the model to predict the membership class of a testing sample according to its flow-features. Generally two main methodologies are adopted: (1) finding a training sample in each class which is closest to the hyper-planes of other classes (e.g., the red circle marked with "A" in Class 1, and the blue square marked with "B" in Class 2, as shown by Fig. 2), and then predicting the membership class of a testing sample by comparing the distance between the testing sample and the closest training sample of all classes; (2) predicting the membership class of a testing sample by comparing the distance between the testing sample and all training samples of all classes (e.g., all red circles of Class 1, and all blue squares of Class 2, as shown by Fig. 2). If all the flow-features of two classes overlap, the 2nd methodology can achieve higher accuracy because all the training samples contribute to the decision weights of the classification model [28].

Leveraging the 2nd methodology, the SVM multi-class problem can be formulated as the following primal problem [29, 30]:

$$\min_{w_i,\xi_k} \frac{1}{2}\sum_{i=1}^{m}\|w_i\|^2 + C\sum_{k=1}^{l}\xi_k \,, \tag{1}$$

$$\text{Constraints } w_{y_k}^T x_k - w_i^T x_k \geq e_{i,k} - \xi_k, \quad \forall i,k, \tag{2}$$

where i is index of a class; m is number of classes; w_i is a weight vector associated with class i; C is a general regularization parameter and $C>0$; k is index of a training sample; l is number of training samples; ξ_k is a non-negative constraint associated with the training sample (i.e., $\xi_k \geq 0$); $x_k \in \Re^n$ is a input vector (n is the number of flow-features) associated with the k-th training sample; $y_k \in \{1, \ldots, m\}$ is the corresponding membership class for x_k; $e_{i,k} = 1 - \delta_{y_k,i}$ ($\delta_{y_k,i} = 1$, if $y_k = i$; $\delta_{y_k,i} = 0$, if $y_k \neq i$). The weight w_i can be regarded as the inverse of the mean distance between an sample and all training samples of class i. Minimizing the cost function of the weights corresponds to maximizing the distance between the hyper-planes of any two classes. The decision function for predicting the membership class of a testing sample is given as

$$D(x) = \max_i D_i = \max_i w_i^T x \,. \tag{3}$$

Equation (3) is used to calculate the value of a testing sample with each weight w_i. The maximum value provided by the weight w_i predicts the membership class of a testing sample, indicating the testing sample is far away from the hyper-planes of other classes according to the constraint described by Equation (2) [29, 30].

However, solving the primal problem formulated by Equations (1) and (2) is quite CPU-intensive and time-consuming. A cost-effective and practical solution is to look at its dual problem as follows:

$$\min_{\alpha} \Omega(\alpha) = \frac{1}{2}\sum_{i=1}^{m}\|w_i(\alpha)\|^2 + \sum_{k=1}^{l}\sum_{i=1}^{m} e_{i,k}\alpha_{i,k} \,, \tag{4}$$

$$\text{Constraints } ([1]\ \alpha_{i,k} \leq C_{i,k} \quad \forall i, \quad [2]\ \sum_{i=1}^{m}\alpha_{i,k} = 0) \quad \forall k, \tag{5}$$

where a particular regularization parameter is given as

$$C_{i,k} = C \text{ if } y_k = i, \text{ and } C_{i,k} = 0 \text{ if } y_k \neq i, \tag{6}$$

and the weight vector w_i is defined as the system function of a dual parameter vector α, i.e.,

$$w_i(\alpha) = \sum_{k=1}^{l} w_{i,k}(\alpha) = \sum_{k=1}^{l} \alpha_{i,k} x_k \,. \tag{7}$$

An optimal solution for the dual parameter vector α to the following equation (8) would give a minimum cost Ω (i.e., the solution of the dual multi-class problem),

$$\frac{\partial\Omega(\alpha)}{\partial\alpha_{i,k}}=0\,, \tag{8}$$

where the gradient of $\Omega(\alpha)$ is given as,

$$g_{i,k}=\frac{\partial\Omega(\alpha)}{\partial\alpha_{i,k}}=w_i(\alpha)^T x_k+e_{i,k}\,,\quad \forall k,i. \tag{9}$$

It is non-trivial and complicated work to reach $\partial\Omega(\alpha)/\partial\alpha_{i,k}=0\ \forall i,k$ for all l training samples, because the total number l of all training samples is potentially large ($1\le k\le l$). A good solution is picking one sample k, obtaining $\partial\Omega(\alpha)/\partial\alpha_{i,k}=0$ for the sample k, i.e., optimizing $\alpha_{i,k}\ \forall i$, while keeping all the parameters for other samples fixed. In order to reach global optimization, indefinite iterations for parameter optimization need to go through a group of training samples [29, 30]. Heuristic solutions have been proposed to search the optimal α_{ik}, and the optimality of $\alpha_{i,k}$ can be checked using the quantity,

$$\upsilon_k=\max_i g_{i,k}-\min_{i:\alpha_{i,k}<C_{i,k}} g_{i,k}\,,\quad \forall k, \tag{10}$$

Equation (10) indicates that υ_k is non-negative. Dual optimality holds when $\upsilon_k=0$, $\forall k$. For practical termination, one can approximately check optimality using a tolerance parameter ε ($\varepsilon>0$), i.e.,

$$\upsilon_k<\varepsilon,\ \forall k. \tag{11}$$

In practice, we only need to require that at least one training sample of each class satisfies Equations (10) and (11) during the parameter searching [29, 30].

Remarks:

The working mechanism of building a SVM multi-class classification model for network traffic classification can be summarized as follows:

(1) each training sample x_k determines a particular parameter $\alpha_{i,k}$ for each class i;

(2) the decision weight w_i for each class is determined by the dual parameter vector of all training samples – each training sample x_k contribute more to the decision weight w_k of its corresponding class and less to the decision weight w_i of other class i ($i\ne k$);

(3) to predict the membership class of a testing sample, each decision weight w_i is used to calculate a corresponding decision value of a testing sample based on Equation (3) – the maximum decision value provided by the decision weight w_k of class k predicts that the testing sample belongs to class k.

B. Iterative-Tuning Scheme

Most existing solutions (e.g., [29–31]) adopt heuristic methods to search the optimal dual parameter vector α to solve the dual problem formulated by Equations (4) and (5). Thousands of iterations may be required to obtain an optimal classification model, which would make the parameter search quite time-consuming if a large-scale training dataset is used. We propose an iterative-tuning scheme to speed up the parameter search and increase the training speed of traffic classification. To the best of our knowledge, no any other published work utilizes an iterative approach for SVM-based classification.

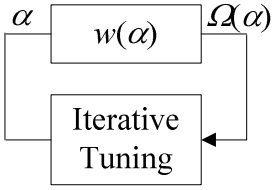

Fig. 3. System diagram of iterative-tuning scheme, where α is a dual parameter vector of SVM classification model, $w(\alpha)$ is SVM system function, and $\Omega(\alpha)$ is cost function of SVM dual problem.

Similar to the existing heuristic schemes for parameter search, the iterative-tuning scheme aims to find a parameter vector α to maximize the SVM classification accuracy. The iterative-tuning scheme can be considered as a criterion-based parameter optimization problem, where the optimization is carried directly on parameters of the classification model. Fig. 3 depicts a system diagram of iterative-tuning scheme for building a SVM classification model.

Existing heuristic schemes (e.g., [29–31]) may cause the parameter search to unnecessarily drift in the opposite direction before reaching an optimal solution. It has been noted that the Gauss-Newton algorithm can predict the parameter search direction more accurately, thus providing faster convergence speed in non-linear optimization [32, 33]. To minimize the cost $\Omega(\alpha)$ formulated Equations (4) and (5), our iterative-tuning scheme applies the Gauss-Newton algorithm to iteratively tune the dual parameter vector α'_k as follows,

$$\alpha'_{k,j+1}=\alpha'_{k,j}-\gamma_j\lambda_j^{-1}\frac{\partial\Omega(\alpha'_{k,j})}{\partial\alpha'_k}\,, \tag{12}$$

$$\alpha'_k=[\alpha_{1,k};\cdots;\alpha_{i,k};\cdots;\alpha_{m,k}]\,, \tag{13}$$

where j is number of tuning iterations, γ is a positive real scalar that determines the step size, λ_j is a positive definite matrix which can be an identity matrix. For each iteration, a gradient estimate $\partial\Omega(\alpha)/\partial\alpha$ of the cost function is recalculated.

Considering the 2nd constraint for the dual parameter belonging to the same class (as expressed by Equation (5)), for each iteration, we only need to compute the parameters for (m–1) classes, and obtain the parameter for the m-th class as

$$\alpha_{m,k}=-\sum_{i=1}^{m-1}\alpha_{i,k}\ . \tag{14}$$

The initial values of the dual parameter vector α'_k for the 1st iteration are set to 0, i.e.,

$\alpha_{i,k}=0,\ 1\le i\le m,$ (15)

Our numerous experimental results suggest that a good choice of λ_j can be a Gauss-Newton approximation of the Hessian of Ω [32, 33], i.e.,

$$\lambda_j = \sum_{i=1}^{m}\left(\frac{\partial w_i(\alpha'_{k,j})}{\partial \alpha'_k}\left[\frac{\partial w_i(\alpha'_{k,j})}{\partial \alpha'_k}\right]^T\right). \quad (16)$$

According to Equation (7), the gradient of w_i is given as

$$\frac{\partial w_i(\alpha_k)}{\partial \alpha_{i,k}} = x_k, \quad (17)$$

$$\frac{\partial w_i(\alpha_k)}{\partial \alpha_{h,k}} = 0,\ 1 \le h \le m-1\ \&\&\ h \ne i. \quad (18)$$

According to Equations (17) and (18), the gradient vector $\partial w_i(\alpha'_{k,j})/\partial \alpha'_k$ is a diagonal matrix. This makes the matrix calculation of Equation (16) CPU cost-effective.

IV. Numerical Result Of SVM Traffic Classfication Algorithm

Current traffic classification techniques primarily rely on packet traces instead of flow records [4, 6]. Packet traces consist of the detailed packet-level information which provides sufficient number of flow-features for traffic classification. However, recording packet traces for millions of flows would consume huge CPU and memory resources. The expensive cost causes cloud service providers and network operators to avoid wide-scale deployment of packet-level-based traffic classification in large data networks.

In comparison to recording packet traces, collecting flow-records is cost-effective and acceptable for real-world implementation of traffic classification modules.

Cisco® NetFlow [10] provides a per-flow traffic summary for network/cloud traffic traversing the routers in the form of flow-records. Generally each flow-record collected by a flow-based protocol (e.g., NetFlow) reports the aggregation of information from different packets of a given protocol to a flow defined by *n*-tuple within a time period. In particular, a unique flow in NetFlow-V5 is defined in terms of a 7-tuple consisting of the same source-IP, destination-IP, source-port, destination-port, type of service, interface, and protocol (Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) [34, 35]).

Unlike packet traces, a flow-record provides a limited number of flow-features for traffic classification. It is worth studying whether the flow-features extracted from flow-level information are adequate for traffic classification. This motivates us to use the NetFlow-V5 data for traffic classification in our experiment.

If the proposed iterative-tuning SVM technique or other SVM techniques based on flow-level information produces a satisfactory classification accuracy, the SVM technique would be readily deployed in all the existing routers that support NetFlow [10] or a similar flow-based protocol such as IPFIX [36], sFlow [37], and JFlow [38].

A. Experimental Setup

We use Cisco® NetFlow-V5 to collect the summarized network traffic trace from an enterprise data network with 200 hosts via a full-duplex Gigabit Ethernet link. The data is collected over a 24-hour period. As listed in Table 1, we utilize a total of 12 flow-features obtained from NetFlow-V5 flow-records as the basis for input to the classification algorithms. Numerous experimental results indicate that traffic classification achieve a better accuracy, if all 12 flow-features are selected.

TABLE 1. Flow-Features For NetFlow-Based Traffic Classification

Feature ID	Feature Name
1	source port
2	destination port
3	average packet size
4	average bytes/sec (source→destination)
5	average bytes/sec (destination→source)
6	packet count (source→destination)
7	packet count (destination→source)
8	byte count (source→destination)
9	byte count (destination→source)
10	ratio of byte count (source→destination) / byte count (destination→source)
11	SYN flag count
12	flow duration

The NetFlow data trace used for traffic classification consists of 241,223 TCP flows. TCP is a stateful protocol, and each TCP flow is a sequence of TCP packets starting and ending at some well-defined times [39, 40]. Network applications running over these TCP flows can be categorized into 7 traffic classes: Database, FTP, Mail, Multimedia, P2P, Service, and WWW. We identify and manually label application class of each TCP flow. In order to evaluate the validity of SVM learning algorithm for traffic classification, we apply SVM to classify the traffic of 3 testing datasets that consist of 130,527 flows, 55,531 flows, and 55,165 flows that belong to 3 *different time periods* respectively. Table 2 shows the number of traffic flows that belong to 7 different application classes in the 3 testing datasets.

TABLE 2. Network Traffic Classes For Different Applications

Application Class	1st Testing Dataset	2nd Testing Dataset	3rd Testing Dataset
Database	781	36	43
FTP	4,422	307	386
Mail	13,018	2,771	2,508
Multimedia	488	36	33
P2P	797	109	283
Service	1,037	293	220
WWW	109,984	51,979	51,692
Total	130,527	55,531	55,165

A traditional strategy for generating a training dataset is to select training samples of each different application class according to their actual ratio in the network, e.g., [4, 6, 18–27]. In order to allow accurate classification of traffic traversing the network at *different time periods*, we randomly select about 10% of total traffic flows, and create a training dataset with 24,298 labeled flows that belong to *the entire time period* as suggested by [6].

To measure the classification performance of different SVM learning algorithms, we use two metrics: overall accuracy and precision. (1) Overall accuracy is the ratio of the sum of all True Positives over the sum of all the flows for all classes. This metric measures the accuracy of a classifier on the whole training dataset; (2) Precision is the ratio of True Positives over the sum of the flows for a given application class. This metric evaluates the quality of classification for a particular application class. True Positives denote the number of correctly classified flows [6].

B. Classification Comparison of Different SVMs

We apply iterative-tuning SVM (SVM-IT) learning and other 8 typical SVM learning algorithms (summarized by [41]) to build a classification model using the same training dataset, and then use the classification model to classify the TCP flows in the 3 testing datasets. The SVM learning algorithms and classification application are implemented in C++ [42].

In experimental results, we use "SVM-0" to "SVM-7" to indicate other 8 typical SVM learning algorithms. Table 3 shows the training time for building a classification model and the overall classification accuracy of different SVM classification algorithms for the 1st testing dataset. Among all the 9 SVM classification algorithms, iterative-tuning SVM exhibits the fastest training speed, and its overall accuracy is comparable with other SVMs; SVM-4 proposed by [41] has the highest classification accuracy, but its training speed is much slower than the proposed iterative-tuning SVM.

TABLE 3. COMPARISON OF DIFFERENT SVM LEARNING ALGORITHMS FOR 1ST TESTING DATASET: TRAINING TIME FOR BUILDING A MODEL AND OVERALL CLASSIFICATION ACCURACY

SVM Type	Training time (ms)	Overall Accuracy
SVM-IT	187	98.66% (128776/130527)
SVM-0	1,575	98.48% (128551/130527)
SVM-1	2,698	98.68% (128804/130527)
SVM-2	530	98.62% (128724/130527)
SVM-3	1,388	98.2% (128172/130527)
SVM-4	1,528	99.1% (129356/130527)
SVM-5	7,534	98.56% (128644/130527)
SVM-6	2,932	98.16% (128127/130527)
SVM-7	5,911	98.5% (128571/130527)

The objective for a successful classification scheme is to maximize accuracy while minimizing training times [6]. We use the ratio of accuracy over training time to represent the ratio of performance/cost for a SVM classification algorithm. Fig. 4 illustrates the ratio of accuracy/training time provided by 9 different SVM classification algorithms for the 1st testing dataset. One can see that SVM-5 exhibits the lowest performance/cost ratio; iterative-tuning SVM provides the highest performance/cost ratio, thus achieving better trade-off between classification accuracy and training speed than other SVMs.

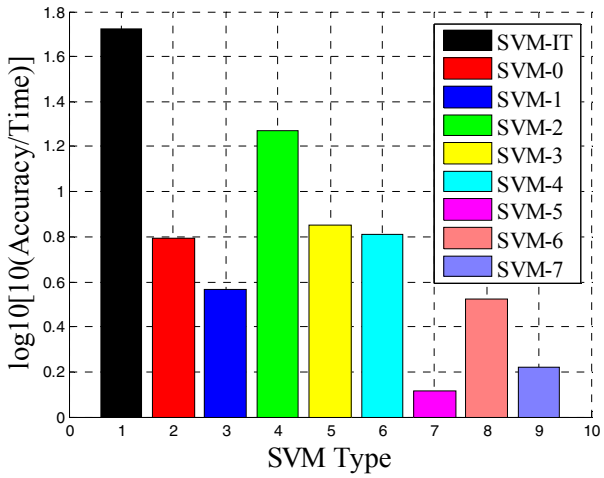

Fig. 4. Ratio of Accuracy/Training time (in logarithmic scale) provided by 9 different SVM classification algorithms for 1st testing dataset

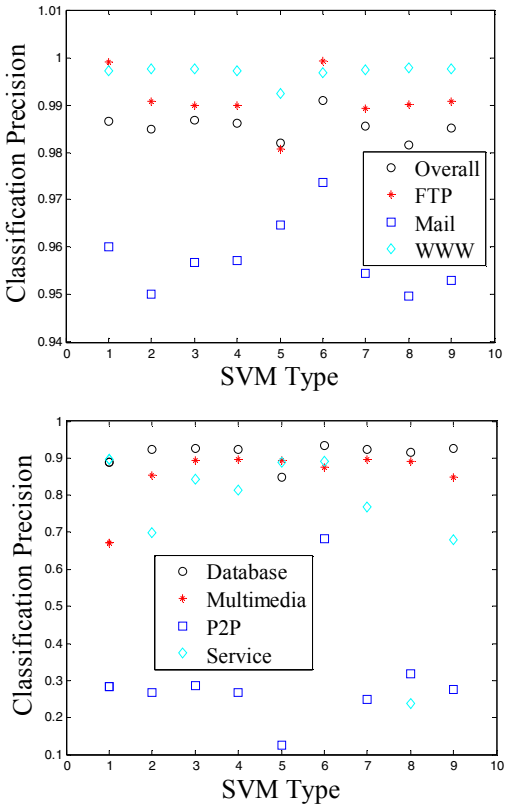

Fig. 5. Classification precision of each class and overall accuracy of all classes provided by 9 different SVM classification algorithms for 1st testing dataset

Fig. 5 depicts the classification precision of each class and the overall accuracy of all classes provided by 9 different SVMs for the 1st testing dataset. From left to right in Fig. 5, the 9 different SVM algorithms are iterative-tuning SVM, SVM-0 to SVM-7. All 9 SVMs can identify more than 99% of *WWW* traffic; SVM-4 has the highest precisions for

identifying *Database*, *FTP*, and *P2P* traffic; iterative-tuning SVM can identify 90% of *Service* traffic, more precisely than the other 8 SVMs; SVM-3 exhibits higher precision for classifying *Mail* traffic than the other 8 SVMs; SVM-5 can identify *Multimedia* traffic with greater precision than the other 8 SVMs; SVM-IT exhibits the lowest precision for predicting Multimedia traffic, because Gauss-Newton algorithm behind SVM-IT splits the overlapped region between Multimedia traffic and other traffic relatively evenly.

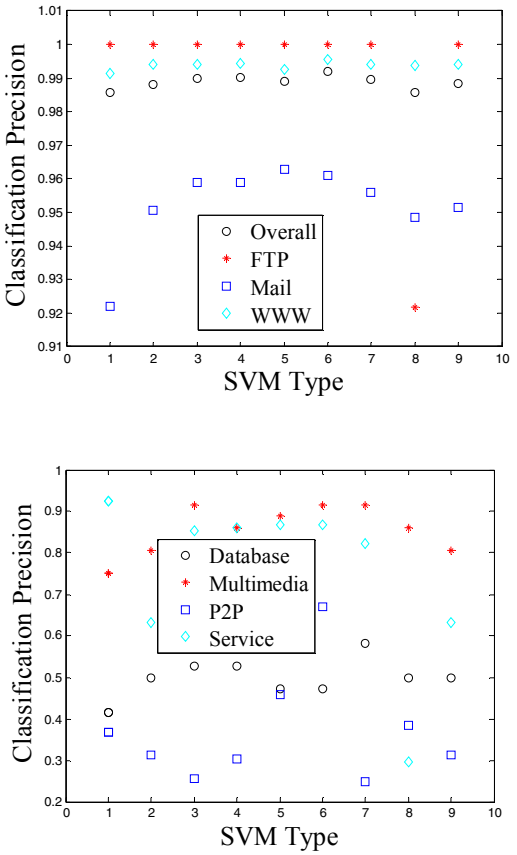

Fig. 6. Classification precision of each class and overall accuracy of all classes provided by 9 different SVM classification algorithms for 2nd testing dataset

Fig. 6 and Fig. 7 show the classification precision of each class and the overall accuracy of all classes provided by 9 different SVMs for the 2nd and 3rd testing datasets respectively. According to the classification results of the 3 testing datasets shown by Figures 5 to 7, the 9 different SVM algorithms provide comparable classification precision of each class and overall accuracy of all classes across the 3 different testing datasets. The experimental results state that all 9 SVM algorithms exhibit robustness to classification accuracy for traffic flows at different time periods. We note an interesting phenomenon whereby the classification precision of multimedia traffic in the 3rd dataset is much higher than other two testing datasets. This is because there are only 33 multimedia flows – the limited sample size introduces a bias on prediction accuracy, according to the theory of statistics, the foundation of SVM learning.

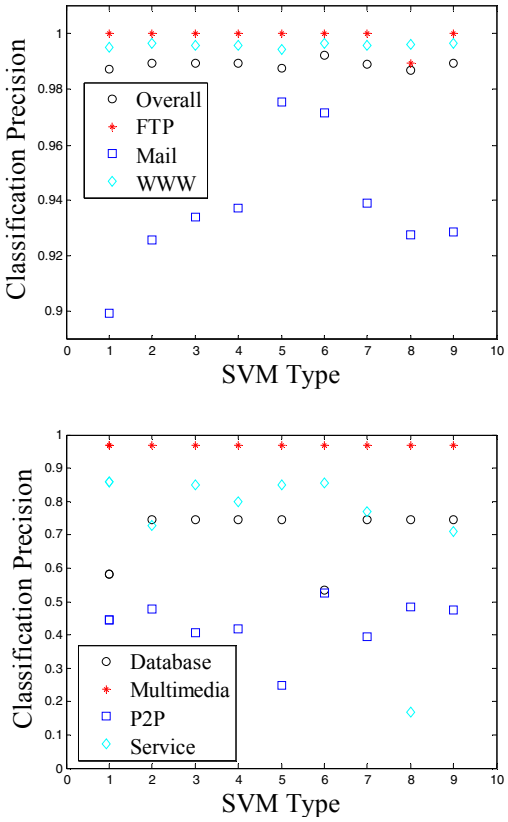

Fig. 7. Classification precision of each class and overall accuracy of all classes provided by 9 different SVM classification algorithms for 3rd testing dataset

C. Benefit of SVM Classification over Port-based Classification

If only the two port-based flow-features (i.e., source port and destination port) are used to create the classification model based on the same training dataset, the overall classification accuracy of all classes provided by any of 9 different SVMs for the 1st testing dataset is about 88%, much lower than the overall accuracy (indicated by Table 3) of classification using the 12 flow-features. Thus SVM can improve classification accuracy of traditional port-based approach significantly.

D. Advantage and Disadvantage of Unbiased Training Dataset

When training samples for each different application class match their actual ratio in the network, the corresponding training dataset reflect traffic pattern of real network precisely. However, such training dataset may result in an arbitrarily low classification precision for a particular class which contributes quite a small amount of training samples. Thus, it is necessary to study the classification accuracy under the unbiased training dataset where each different application class contributes the same number of training samples.

We create an unbiased training dataset by randomly selecting 400 flows for each application class. Fig. 8 depicts the classification precision of each class and the overall accuracy of all classes provided by 9 different SVMs for the

1st testing dataset when an unbiased training dataset is used. Obviously unbiased training dataset makes the classification precision of each different class more balanced, and there is no arbitrarily low precision for any particular class. Classification using the biased training dataset may cause less than 50% of P2P and Service flows to be identified correctly (as shown in Fig. 5). The advantage of classification using the unbiased training dataset is that no matter which SVM algorithm is applied, each different class exhibits satisfactory precision (as shown in Fig. 8): (1) more than 90% flows of *Database*, *FTP*, *Mail*, *Service* and *WWW* traffic can be identified successfully; (2) precisions of identifying Multimedia and P2P traffic are more than or approximately 70%.

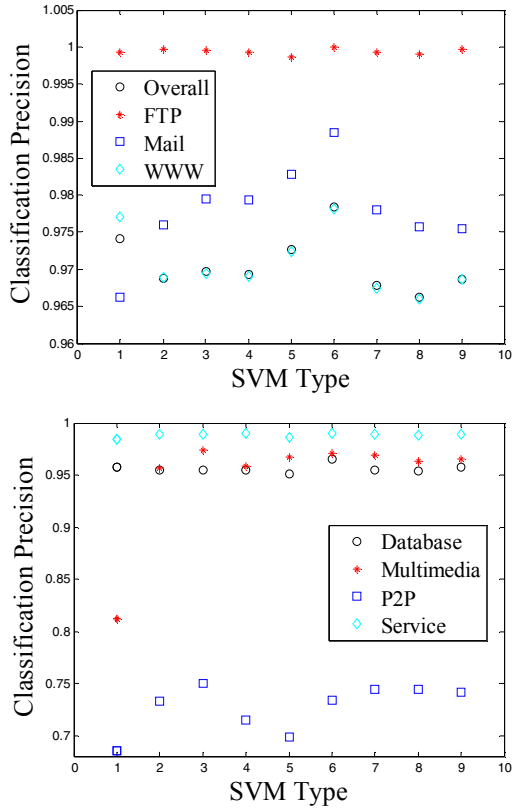

Fig. 8. Classification precision provided by 9 different SVM classification algorithms for 1st testing dataset when unbiased training dataset is used

However, the disadvantage of classification using the unbiased training dataset is that the overall accuracy decreases by nearly 2%, or from approximately 99% to 97% (as shown by the black circles in the left plots of Fig. 5 and Fig. 8).

Similar observation is found in the experimental results (omitted due to page limit) of the 2nd and 3rd testing datasets, when SVM classification leverages the unbiased training dataset.

V. Conclusions

We have sought to develop a scalable and accurate traffic classification scheme based on a flow-based protocol such as NetFlow. Two main contributions of this paper are: (i) An iterative-tuning scheme has been proposed to increase the training speed for the SVM multi-class classification dual problem; (ii) The working mechanism of iterative-tuning scheme for building a classification model has been analyzed theoretically, so that equations (i.e., Equations (12) to (18)) for obtaining the dual parameter vector for the SVM classification model have been derived.

The key idea of building a SVM multi-class classification model is that (1) each training sample x_k determines a particular parameter $\alpha_{i,k}$ for each class i; (2) the decision weight w_i of each class is determined by the dual parameter vector of all training samples – each training sample x_k contribute more to the decision weight w_k of its corresponding class k and less to the decision weight w_i of other class i ($i \neq k$). Given that the statistical property of SVM decision requires a large number of training samples, a cost-effective solution is selecting a training sample and optimizing its parameter $\alpha_{i,k}$, while keeping all parameter vectors for other samples fixed. In order to avoid the bias due to the parameter optimization order of different samples, one needs to perform indefinite iterations to optimize parameter vector of each sample repetitively until a global optimality is achieved. The Gauss-Newton algorithm behind the proposed iterative-tuning scheme can predict the parameter searching direction more accurately than existing heuristic parameter searching schemes, thus making the training speed of the iterative-tuning SVM much faster.

We have applied the iterative-tuning SVM learning algorithm to classify traffic. Performance comparison with 8 other typical SVM learning algorithms demonstrates that the proposed iterative-tuning SVM is computationally more efficient than the 8 typical SVMs, while exhibiting almost identical accuracy as those 8 SVMs. On the other hand, we discover that an unbiased training dataset drives the precisions of identifying each different class, but slightly decreases the overall classification accuracy of all classes.

A potential commercial benefit generated by this paper is: experimental results based on the flow-level information confirms that the overall classification accuracy of the proposed iterative-tuning SVM learning algorithm and other 8 typical SVM learning algorithms varies between 98% and 99%. Such high accuracy allows the network/cloud operators to apply traffic classification for a range of issues including semi-real-time security monitoring and traffic engineering.

Since NetFlow does not provide the detailed packet-level information, less than 80% of P2P flows can be identified correctly even when an unbiased training dataset is used. For future work, we propose to integrate the SVM classification approach based on flow-features with a heuristic approach based on the dynamic application characteristics to identify P2P traffic more accurately in the future work.

Acknowledgment

We appreciate financial support from NSERC grant #CRDPJ 354729-07 and OCE grant #CA-ST-150764-8. This work is also supported in part by Solana Networks.

References

[1] M. Roughanx, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification," in *Proceedings of ACM IMC*, Taormina, Italy, October 2004, pp. 135–148.

[2]]R.L. Gomes, L.F. Bittencourt, and E.R.M. Madeira, "A framework for SLA establishment of virtual networks based on QoS classes," in *Proceedings of* 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), May 2013, pp.1175–1178.

[3] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K.C. Claffy, "The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite," in *Proceedings of Passive and Active Network Measurement Workshop* (*PAM*), Amsterdam, Netherlands, 2001.

[4] J. Erman, M. Arlitt, A. Mahanti, and C. Williamson, "Identifying and Discriminating Between Web and Peer-to-Peer Traffic in the Network Core," in *Proceedings of WWW*, May 2007.

[5] B. Park1, Y. Won, J. Chung, M. Kim, and J.W. Hong, "Fine-grained traffic classification based on functional separation," International Journal of Network Management, 23(5), September/October 2013, pp. 350–381.

[6] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of ACM CoNEXT*, Madrid, Spain, December 2008.

[7] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Network Magazine*, 26(1), 2012, pp. 35–40.

[8] G. Szabo, J. Szule, B. Lins, Z. Turanyi, G. Pongracz, D. Sadok, and S. Femandes, "Capturing the real influencing factors of traffic for accurate traffic identification," in *Proceedings of IEEE ICC*, Ottawa, Canada, June 2012, pp. 2129–2134.

[9] J.L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of ISP Traffic: Trends, User Habits, and Access Technology Impact," IEEE Transactions on Network and Service Management, 9(2), June 2012, pp. 142–155.

[10] Cisco Systems Inc., "Cisco IOS Flexible NetFlow Overview," *white paper*, 2010. Available at
http://www.cisco.com/en/US/docs/ios/fnetflow/configuration/guide/fnetflow_overview.pdf

[11] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proceedings of Passive and Active Network Measurement Workshop* (*PAM*), April 2005.

[12] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proceedings of WWW*, May 2004.

[13] Ellacoya. http://www.ellacoya.com.

[14] Packeteer. http://www.packeteer.com.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," in *Proceedings of ACM SIGCOMM*, August 2005.

[16] T. Karagiannis, A. Broido, M. Faloutsos, and K.C. Claffy, "Transport layer identification of p2p traffic," in *Proceedings of ACM IMC*, October 2004.

[17] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs," in *Proceedings of ACM IMC*, October 2007.

[18] T.T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials*, 10(4), 2008, pp. 56–76.

[19] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proceedings of Passive and Active Network Measurement Workshop* (*PAM*), April 2004.

[20] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proceedings of ACM SIGMETRICS*, June 2005.

[21] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classificaton Using Clustering Algorithms," in *Proceedings of ACM SIGCOMM MineNet Workshop*, September 2006.

[22] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for Internet traffic classification," *IEEE Transactions on Neural Networks*, 18(1), January 2007, pp.223–239.

[23] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM CCR*, 37(1), January 2007, pp.7–16.

[24] D. Schatzmann, W. Muhlbauer, T. Spyropoulos, and Xenofontas Dimitropoulos, "Digging into HTTPS: Flow-Based Classification of Webmail Traffic," in *Proceedings of ACM IMC*, Melbourne, Australia, November 2010.

[25] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/Realtime Traffic Classification Using Semi-Supervised Learning," in *Proceedings of* IFIP Performance, October 2007.

[26] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM CCR*, 36(5), October 2006, pp.7–15.

[27] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate Internet traffic classification," *Information Systems Frontiers*, 12(2), 2010, pp. 149–156.

[28] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 2, June 1998, pp. 121–167.

[29] K. Crammer and Y. Singer, "On the learn-ability and design of output codes for multiclass problems," in *Proceedings of COLT*, 2000.

[30] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, 2, 2001, pp. 265–292.

[31] S.S. Keerthi, S. Sundararajan. K.W. Chang, C.J. Hsieh, and C.J. Lin, "A sequential dual method for large scale multi-class linear SVMs," in *Proceedings of ACM KDD*, 2008.

[32] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.

[33] W.K. Ho, Y. Hong, A. Hansson, H. Hjalmarsson, and J.W. Deng, "Relay Auto-Tuning of PID Controllers using Iterative Feedback Tuning," *Automatica*, 39(1), January 2003, pp. 149–157.

[34] Y. Hong, C. Huang, and J. Yan, "Analysis of SIP Retransmission Probability Using a Markov-Modulated Poisson Process Model," in *Proceedings of IEEE NOMS*, Osaka, Japan, April 2010, pp. 179–186.

[35] Y. Hong, C. Huang, and J. Yan, "Applying Control Theoretic Approach To Mitigate SIP Overload," *Telecommunication Systems*, 54(4), December 2013, pp. 387–404.

[36] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," *IETF RFC* 7011, September 2013.

[37] P. Phaal and M. Lavine, "sFlow Version 5," *white paper*, July 2004. Available at
http://www.sflow.org/sflow_version_5.txt

[38] Juniper Networks Inc., "Juniper Flow Monitoring – J-Flow on J Series Services Routers and Branch SRX Series Services Gateways," *white paper*, 2011. Available at
http://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf

[39] Y. Hong, O.W.W. Yang, and C. Huang, "Self-Tuning PI TCP Flow Controller for AQM Routers With Interval Gain and Phase Margin Assignment," in *Proceedings of IEEE Globecom*, Dallas, U.S.A, November 2004, pp. 1324–1328.

[40] Y. Hong, C. Huang, and J. Yan, "A Comparative Study of SIP Overload Control Algorithms," *Network and Traffic Engineering in Emerging Distributed Computing Applications*, Edited by J. Abawajy, M. Pathan, M. Rahman, A.K. Pathan, and M.M. Deris, IGI Global, 2012, pp. 1–20.

[41] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, 9, 2008, pp. 1871–1874.

[42] B. Stroustrup, *The C++ Programming Language*, 4th Edition, Addison-Wesley, Boston, MA, May 2013.