Packet Classification using Support Tensor Machines

Antonio Augusto Teixeira Peixoto, Laise Santos e Carlos Alexandre Rolim Fernandes

***Abstract*— A wide variety of packet classification algorithms exist in the research literature and commercial market. The existing solutions exploit various design tradeoffs, providing high search rates, power and space efficiency and the ability to scale to large numbers of filters. However, still remains a need for techniques that achieve a favorable balance among these tradeoffs and scale to support classification. Based on this motivations, this paper presents a tensor approach for the classification of TCP and UDP packets. By using a multidimensional structure, more specifically a 4-th order tensor, to store the packet data, a tensorial algorithm known as Support Tensor Machines (STM) is used to perform classification. Results showed good performance of the approach in comparison to other classifiers such as the Support Vector Machines and Naive-Bayes.**

***Keywords*— packet classification, tensor, support tensor machines.**

I. Introduction

The process of classifying information is directly related to categorization, where ideas, objects or data are recognized, differentiated, understood and then, separated in different tags [1]. Moreover, data classification can be achieved with methods aimed to determine whether or not the data contains some specific information, feature, or behavior, then, identifying the correct class of the data [2]. This method is an important branch of computer vision, machine learning and computational intelligence, being used in many fields such as geophysics (seismic recognition, seismic swarms) [3], recognition of fingerprint images [4], face [5], handwritten digits [6], gait [7], electrocardiogram signals [8], and even identification of specific vehicles [9].

The most common methods of classification are based on supervised learning, which is the task of learning a function that maps an input to an output based on example input-output pairs [10]. In supervised learning, each example is a pair consisting of an input object, typically a vector, and a desired output value, also called the supervisory signal. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. A common learning algorithm that has been reported used on various classification applications such as the mentioned earlier, is the the support vector machine (SVM) [11], mostly employed to solve two-class problems, but also used on multi-class solutions [12].

Augusto Peixoto, Redes de Computadores, IFCE, Jaguaribe-CE, e-mail: antonio.peixoto@ifce.edu.br; Laise Santos, Redes de Computadores, IFCE, Jaguaribe-CE, e-mail: layse0877@gmail.com; Carlos Alexandre R. Fernandes, Engenharia da Computação, UFC, Sobral-CE, e-mail: alexandrefernandes@ufc.br

More specifically, a standard SVM model is based on vector inputs and cannot directly deal with matrices or higher dimensional data structures, namely, tensors, which are very common in real-life applications [15]. The SVM realization on such high dimensional inputs is by reshaping each sample into a vector. However, when the training data sample size is relatively small compared to the feature vector dimension, it may easily result in poor classification performance, known as curse of dimensionality [15], [16].

In order to avoid the data destruction by converting tensors into vectors, the supervised tensor learning method is proposed [17]. This technique has been extensively studied in recent years, and, proposes a supervised tensor learning framework, which extends the standard linear SVM framework to tensor patterns by constructing multilinear models, hence, called Support Tensor Machines (STM). Also, the technique utilizes a rank-one tensor to capture the data structure, thereby alleviating the overfitting and curse of dimensionality problems in the conventional SVM [17], [18]. Moreover, in the context of supervised tensor learning, preserving the structural information and exploiting the discriminating nonlinear relationships of tensor data are crucial for improving the performance of learning tasks [19].

On the other hand, classification of data packets in computer networks is a very demanding task [20]. Packet classification is important for applications such as firewalls, intrusion detection, and differentiated services. Existing algorithms for packet classification reported in the literature scale poorly in either time or space as filter databases grow in size [21]. Also, existing solutions may require high computational cost. In the work of [22], an overview of packet classification algorithms is presented. As explained [22], researchers have proposed a variety of algorithms which, broadly speaking, can be categorized as basic search algorithms, geometric algorithms, heuristic algorithms, or hardware-specific search algorithms.

Moreover, the use of tensors and tensor-based classifiers in packet classification is not common in the literature, with few works addressing the topic. We may cite [23], which utilizes a multidimensional approach for multi-scale feature attention approach to network traffic classification, by using convolutional neural networks (CNN) as the building block of the deep packet analysis model. With so few tensor-based works in covering this topic, it is desirable the development of multilinear algorithms that could be used in packet classification.

In this paper, a tensor-based approach for the classification of TCP and UDP packets is presented. By using a multidimensional structure, to store the packet data, a tensorial algorithm

known as STM is used to perform classification. The first step is a creation of a 4-th order tensor from two packet databases, with information from TCP and UDP segments obtained through a packet capture software. After obtaining the tensor, the data is classified using the STM algorithm, achieving interesting results. For comparison purposes, both linear SVM and Naive-Bayes were tested.

A. Organization

The rest of this paper is organized as follows. Section II presents methods used, while Section III describes the Support Tensor Machine formulation. Section IV introduces the STM algorithm, Section V presents the results and, finally, Section VI ends this paper with the conclusions.

B. Notation

Scalars are denoted by Roman lower-case letters (a,b,...), vectors as lower-case boldface letters (**a**,**b**,...), matrices as upper-case boldface letters (**A**,**B**,...) and tensors as calligraphic letters ($\mathcal{A}$,$\mathcal{B}$,...). To retrieve the element (i,j) of an arbitrary matrix $\mathbf{A} \in \mathbb{C}^{I\times R}$, we use $a_{i,j}$ (the same for tensors). $\mathbf{A}^T$ stands for the transpose. Also, $\otimes$ denotes the Kronecker product between $\mathbf{A} \in \mathbb{R}^{I\times K}$ and $\mathbf{B} \in \mathbb{R}^{J\times L}$, resulting in $\mathbf{A}\otimes\mathbf{B} \in \mathbb{C}^{IJ\times KL}$. A matrix unfold of a 4-th order tensor $\mathcal{X} \in \mathbb{R}^{K\times M\times N\times P}$ is given by $\mathbf{X}^{[n]}$, i.e. $\mathbf{X}^{[1]} \in \mathbb{R}^{KMN\times N}$.

II. Methods

The methodology of this work was developed as follows. Initially, TCP and UDP segment data were collected using the Wireshark software. This software is a network protocol analyzer, allowing the visualization of data from a given network, allowing the user to interact by browsing through the captured data and seeing the details of each packet [24]. Two networks were analyzed for packets, one wired and the other one wireless, where packets were obtained during conventional web browsing, video calls and streaming services.

The data collected from the TCP and UDP packets were as follows:

- Source port;
- Destination port;
- Header length;
- Window size;
- Payload (in bytes);
- Flag;

The attributes described above were chosen because they facilitate the distinction between TCP and UDP segments, thus facilitating the classification of these later on. After collecting 100 samples of TCP and UDP segments from the wired network, another 100 samples were obtained from the wireless network. Once the samples were properly tabulated, a 3rd order tensor containing the database was assembled, with the tensor dimensions being $2 \times 3 \times 6 \times 100$, with a total of 600 packets. The first mode of the tensor denotes the two types of network: wired and wireless, the second mode refers to how the packets were obtained (web browsing, video call or streaming), the third mode refers to the number of attributes whereas the fourth mode indicates the number of samples. The class tags are two: a packet of the dataset is either TCP or UDP. In Figure 1 we can see the flowchart of the methodology.

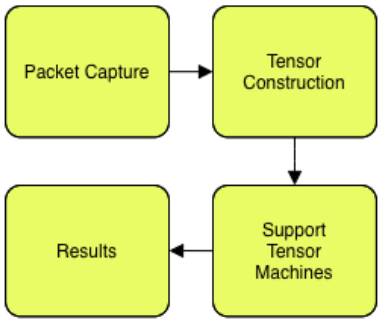

Fig. 1. Flowchart of the presented classification method.

TABLE I
DATABASE DESCRIPTION.

Size: 2 × 3 × 6 × 100	**Number of Packets**
Wired Network	100 (60 TCP, 40 UDP)
Wireless Network	100 (60 TCP, 40 UDP)

Table I shows the dataset description, with tensor dimensions, number of TCP and UDP packets per type of network. In the following, the adopted classifier is presented.

III. Support Tensor Machines (STM) Formulation

Considered an extension to the conventional SVM, the STM works in the following way. Let $\mathcal{X} \in \mathbb{R}^{K\times M\times N\times P}$ be the training data set tensor split into P third order tensors and a vector **y** consisting of the class tags associated to each of the P tensors, with **y** $\in$ {-1,1}, thus, we need to find a tensor classifier such the two classes can be separated with maximum margin as the decision function:

$$f(\mathcal{X}_p) = \mathcal{X}_p(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}) + b, \tag{1}$$

where $\mathcal{X}_p \in \mathbb{R}^{K\times M\times N}$, $\mathbf{v}^{(1)} \in \mathbb{R}^{1\times K}$, $\mathbf{v}^{(2)} \in \mathbb{R}^{1\times M}$ and $\mathbf{v}^{(3)} \in \mathbb{R}^{1\times N}$ are vectors orthogonal to the hyperplane. We can also define:

$$\mathcal{X}_p(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}) = \sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N} x_{k,m,n}\mathbf{v}_k^{(1)}\mathbf{v}_m^{(2)}\mathbf{v}_n^{(3)}. \tag{2}$$

Then, (1) can be rewritten as follows:

$$f(\mathcal{X}_p) = \mathcal{X}_p \times_1 \mathbf{v}^{(1)} \times_2 \mathbf{v}^{(2)} \times_3 \mathbf{v}^{(3)} + b. \tag{3}$$

In matricial notation, we have:

$$f(\mathbf{X}_p) = (\mathbf{v}^{(2)} \otimes \mathbf{v}^{(3)})\mathbf{X}_p^{(1)}(\mathbf{v}^{(1)})^T + b, \tag{4}$$

where $\mathbf{X}^{[1]} \in \mathbb{R}^{MN\times K}$ is an unfolding of the tensor $\mathcal{X}_p$.

As stated earlier, the LR-STM method is an generalization of the SVM for higher order arrays. In order to use (3), it

is necessary to compute $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$. The algorithm is described as follows:

- Initialize $\mathbf{v}^{(2)}$ = {1,...,1} $\in \mathbb{R}^{1\times M}$ and $\mathbf{v}^{(3)}$ = {1,...,1} $\in \mathbb{R}^{1\times N}$;
- Let $\mathbf{x}_p = (\mathbf{X}_p^{[1]})^T(\mathbf{v}^{(2)} \otimes \mathbf{v}^{(3)})^T$;
- Compute $\mathbf{v}^{(1)}$ by solving the following optimization problem:

$$\min_{\mathbf{v}^{(1)}} \frac{1}{2}\langle \mathbf{v}^{(1)}, \mathbf{v}^{(1)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{5}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(1)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{6}$$

where

$$\xi_p \geq 0.$$

- With $\mathbf{v}^{(1)}$ obtained, let $\mathbf{x}_p = (\mathbf{X}_p^{[2]})^T(\mathbf{v}^{(3)} \otimes \mathbf{v}^{(1)})^T$, then compute $\mathbf{v}^{(2)}$ with:

$$\min_{\mathbf{v}^{(2)}} \frac{1}{2}\langle \mathbf{v}^{(2)}, \mathbf{v}^{(2)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{7}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(2)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{8}$$

where

$$\xi_p \geq 0.$$

- With $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$ obtained, let $\mathbf{x}_p = (\mathbf{X}_p^{[3]})^T(\mathbf{v}^{(1)} \otimes \mathbf{v}^{(2)})^T$, then compute $\mathbf{v}^{(3)}$ with:

$$\min_{\mathbf{v}^{(3)}} \frac{1}{2}\langle \mathbf{v}^{(3)}, \mathbf{v}^{(3)}\rangle + C\sum_{p=1}^{P}\xi_p \tag{9}$$

subject to

$$\mathbf{y}_p(\langle \mathbf{v}^{(3)}, \mathbf{x}_p\rangle + b) \geq 1 - \xi_p, \tag{10}$$

where

$$\xi_p \geq 0.$$

- After the steps above, we can iteratively compute $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ until they converge.

Now, with $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ obtained, we can classify the test data $\mathcal{Z}_s \in \mathbb{R}^{K\times M\times N}$, with s = 1,...,S and $J = P + S$, where P and S denotes the training and test samples, respectively. Thus we have:

$$g(\mathcal{Z}_s) = sign(\mathcal{Z}_s \times_1 \mathbf{v}^{(1)} \times_2 \mathbf{v}^{(2)} \times_3 \mathbf{v}^{(3)} + b). \tag{11}$$

IV. STM Estimation Algorithm

The algorithm of the STM method, in pseudo-code format, is shown in Algorithm 1. The basic process of classification is illustrated in Figure 2, where, with the training samples, we build the LR-STM model by estimating $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$. Then, we use the model to classify the test samples with (11).

Convergence is achieved as follows. As we shown, the optimizations problems in (5,7,9) are the same as in the standard SVM algorithm, then we can use the computational

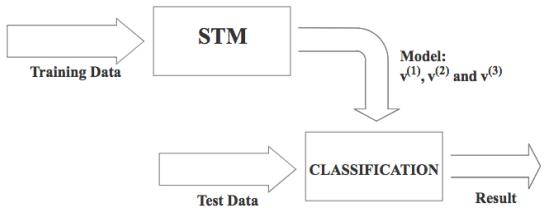

Fig. 2. STM classification steps.

Algorithm 1 STM Algorithm

For p = 1,...,P:
1)Initialization: Set i = 0; Initialize $\mathbf{v}_i^{(2)}$ and $\mathbf{v}_i^{(3)}$;
2)With $\mathbf{x}_p = (\mathbf{X}_p^{[1]})^T(\mathbf{v}_i^{(2)} \otimes \mathbf{v}_i^{(3)})^T$, estimate $\mathbf{v}_i^{(1)}$ using (5);
3)i = i + 1;
4)With $\mathbf{x}_p = (\mathbf{X}_p^{[2]})^T(\mathbf{v}_{i-1}^{(3)} \otimes \mathbf{v}_{i-1}^{(1)})^T$, estimate $\mathbf{v}_i^{(2)}$ using (7);
5)With $\mathbf{x}_p = (\mathbf{X}_p^{[3]})^T(\mathbf{v}_{i-1}^{(1)} \otimes \mathbf{v}_i^{(2)})^T$, estimate $\mathbf{v}_i^{(3)}$ using (9);
6)Repeat steps 2-5 until convergence;

methods for SVM to solve (5,7,9). As for the convergence of the algorithm, the iterative procedure to solve the optimization problems (5,7,9) will monotonically decreases the objective function values in (6,8,10), and hence the STM algorithm converges.

Let $\mathbf{v}_0^{(2)}$ and $\mathbf{v}_0^{(3)}$ be the initial values. Fixing $\mathbf{v}_0^{(2)}$ and $\mathbf{v}_0^{(3)}$, we get $\mathbf{v}_0^{(1)}$ by solving the optimization problem (5). Likewise, fixing $\mathbf{v}_0^{(1)}$ and $\mathbf{v}_0^{(3)}$, we get $\mathbf{v}_1^{(2)}$ by solving the optimization problem (7) and so on. Notice that the optimization problem of SVM is convex, so the solution of SVM is globally optimum [25]. Thus, we have:

$$f(\mathbf{v}_0^{(1)}, \mathbf{v}_0^{(2)}, \mathbf{v}_0^{(3)}) \geq f(\mathbf{v}_0^{(1)}, \mathbf{v}_1^{(2)}, \mathbf{v}_0^{(3)}). \tag{12}$$

And finally we get:

$$f(\mathbf{v}_0^{(1)}, \mathbf{v}_0^{(2)}, \mathbf{v}_0^{(3)}) \geq \ldots \geq f(\mathbf{v}_1^{(1)}, \mathbf{v}_1^{(2)}, \mathbf{v}_1^{(3)}) \geq \ldots. \tag{13}$$

V. Results

In this section, the obtained classification results are presented. The STM algorithm was implemented using MATLAB 2017b, in a 9-th generation core i3 processor. For comparison purposes, the conventional SVM with linear kernel and the Naive-bayes classifiers were tested, using a vectorized version of the data, in order to better evaluate the STM performance. Also, an Artificial Neural Network (ANN) implementing forward propagation with two fully connected layers, Rectified linear unit (ReLU) activation functions and a softmax function to the final fully connected layer, using vectorized data, was tested against the STM.

The data was classified using K-fold cross validation, with K = 10 and the relaxing constant was set C = 100. The results are presented in the form of accuracy, which is defined as the number of correctly classified samples over the total number of samples, execution time, in seconds, of the feature technique plus classifier, and, confusion tables. The results were averaged 100 times to eliminate fluctuation.

The first result, presented in Table II shows the accuracy and execution times of the tested methods. As can be seen, the STM showed the highest accuracy, thus showing its good performance in packet classification. The second best classification rate was achieved by the ANN, however, at a cost of more execution time. The worst accuracy was obtained by the Naive-Bayes classifies. Moreover, although the STM demanded more running time than the SVM and Naive-Bayes techniques, it achieved higher accuracy, showing a trade-off between performance of classification and computational cost.

TABLE II

ACCURACY AND EXECUTION TIME RESULTS FOR THE TESTED TECHNIQUES

Classifier	Accuracy	Time (s)
STM	**88.3**%	201.11
SVM	80.3%	171.2
Naive-Bayes	75.4%	126.8
ANN	83.2%	244.67

Furthermore, in Table III, the obtained confusion matrix is shown, illustrating the performance of STM in the TCP and UDP packet classification process. As we can see, an accuracy of 88.3% was achieved in the classification of the data.

TABLE III

CONFUSION TABLE - ACCURACY OF THE STM.

88.3% Accuracy	TCP	UDP
TCP	330	40
UDP	30	200

VI. CONCLUSIONS

In this work, a tensor-based approach for packet classification was presented. By using a multidimensional structure, to store the TCP and UDP packets, a tensorial algorithm known as STM is used to perform classification. Results showed good performance of the proposed approach in comparison to other classifiers such as the SVM and Naive-Bayes techniques.

REFERENCES

[1] M. Pardo and G. Sberveglieri, “Learning from data: A tutorial with emphasis on modern pattern recognition methods," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 203-217, 2009.

[2] S. R. Kulkarni, G. Lugosi and S. S. Venkatesh, “Learning pattern classification-a survey," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2178-2206, 1998.

[3] Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., & Johnson, P. A. (2017). Machine learning predicts laboratory earthquakes. Geophysical Research Letters, 44(18), 9276-9282.

[4] Jia, J., Cai, L., Lu, P., & Liu, X. (2007). Fingerprint matching based on weighting method and the SVM. Neurocomputing, 70(4-6), 849-858.

[5] M. J. Lyons, J. Budynek, and S. Akamatsu, “Automatic classification of single facial images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 12, pp. 1357-1362, 1999.

[6] B. Savas, and L. Eldén, “Handwritten digit classification using higher order singular value decomposition," *Pattern recognition*, vol. 40, no. 3, pp. 993-1003, Elsevier, 2007.

[7] R. K. Begg, M. Palaniswami and B. Owen, “Support vector machines for automated gait classificatio," *IEEE transactions on Biomedical Engineering*, vol. 52, no. 5, pp. 828-838, 2005.

[8] F. Melgani & Y. Bazi, “Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE transactions on information technology in biomedicine*, vol. 12, vol. 5, pp. 667-677, 2008.

[9] S. Gupte, O. Masoud, R. F. Martin and N. P. Papanikolopoulos, Detection and classification of vehicles. *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 1, pp. 37-47, 2002.

[10] Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 3(1), 1-130.

[11] Vapnik, V. (2013). The nature of statistical learning theory. Springer science & business media.

[12] Mathur, A., & Foody, G. M. (2008). Multiclass and binary SVM classification: Implications for training and classification users. IEEE Geoscience and remote sensing letters, 5(2), 241-245.

[13] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), 121-167.

[14] Ma, L., Hu, Y., & Zhang, Y. (2017). Support Tucker Machines Based Bubble Defect Detection of Lithium-ion Polymer Cell Sheets. Engineering Letters, 25(1).

[15] Chen, C., Batselier, K., Ko, C. Y., & Wong, N. (2019, July). A support tensor train machine. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[16] Li, J., Allinson, N., Tao, D., & Li, X. (2006). Multitraining support vector machine for image retrieval. IEEE Transactions on Image Processing, 15(11), 3597-3601.

[17] D. Tao, X. Li, X. Wu, W. Hu, and S. J. Maybank. Supervised tensor learning. Knowledge and Information Systems, 13(1):1?42, 2007.

[18] Cai, D., He, X., Wen, J. R., Han, J., & Ma, W. Y. (2006). Support tensor machines for text categorization.

[19] He, L., Lu, C. T., Ma, G., Wang, S., Shen, L., Yu, P. S., & Ragin, A. B. (2017, August). Kernelized support tensor machines. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 1442-1451). JMLR. org.

[20] Taylor, David E., and Jonathan S. Turner. "Scalable packet classification using distributed crossproducing of field labels." In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 1, pp. 269-280. IEEE, 2005.

[21] Baboescu, Florin, and George Varghese. "Scalable packet classification." IEEE/ACM transactions on networking 13, no. 1 (2005): 2-14.

[22] Gupta, Pankaj, and Nick McKeown. "Algorithms for packet classification." IEEE Network 15, no. 2 (2001): 24-32.

[23] Wang, Yipeng, Xiaochun Yun, Yongzheng Zhang, Chen Zhao, and Xin Liu. "A Multi-scale Feature Attention Approach to Network Traffic Classification and Its Model Explanation." IEEE Transactions on Network and Service Management (2022).

[24] Lamping, Ulf, and Ed Warnicke. "Wireshark user’s guide." Interface 4, no. 6 (2004): 1.

[25] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20, no. 3 (1995): 273-297.