Impact of Self C Parameter on SVM-based Classification of Encrypted Multimedia Peer-to-Peer Traffic

Vanice Canuto Cunha, Damien Magoni, Pedro Inácio, Mario Freire

Impact of Self C Parameter on SVM-based Classification of Encrypted Multimedia Peer-to-Peer Traffic

Vanice Canuto Cunha[1,2], Damien Magoni[3], Pedro R. M. Inácio[2], and Mario M. Freire[2] *

Abstract Home users are increasingly acquiring, at lower prices, electronic devices such as video cameras, portable audio players, smartphones, and video game devices, which are all interconnected through the Internet. This increase in digital equipment ownership induces a massive production and sharing of multimedia content between these users. The supervised learning machine method Support Vector Machine (SVM) is vastly used in classification. It is capable of recognizing patterns of samples of predefined classes and supports multi-class classification. The purpose of this article is to explore the classification of multimedia P2P traffic using SVMs. To obtain relevant results, it is necessary to properly adjust the so-called Self C parameter. Our results show that SVM with linear kernel leads to the best classification results of P2P video with an F-Measure of 99% for C parameter ranging from 10 to 70 and to the best classification results of P2P file-sharing with an F-Measure of 98% for C parameter ranging from 30 to 70. We also compare these results with the ones obtained with Kolmogorov-Smirnov (KS) tests and Chi-square tests. It is shown that SVM with linear kernel leads to a better classification performance than KS and chi-square tests, which reached an F-Measure of 67% and 70% for P2P file-sharing and P2P video, respectively, for KS test, and reached an F-Measure of 85% for both P2P file-sharing and P2P video for chi-square test. Therefore, SVM with linear kernel and suitable values for the Self C parameter can be a good choice for identifying encrypted multimedia P2P traffic on the Internet.

[1]Universidade Federal de Mato Grosso, Cuiabá, Brasil
[2]Instituto de Telecomunicações, Universidade da Beira Interior, Covilhã, Portugal
[3]LaBRI-CNRS, Université de Bordeaux, Talence, France
e-mail: vanice@ic.ufmt.br, e-mail: damien.magoni@u-bordeaux.fr, e-mail: inacio@di.ubi.pt, e-mail: mario@di.ubi.pt

* This work was financed by CAPES (Brazilian Federal Agency for Support and Evaluation of Graduate Education) within the Ministry of Education of Brazil under a scholarship supported by the International Cooperation Program CAPES/COFECUB - Project 9090-13-4/2013 at the University of Beira Interior. This work is also funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020 and by FCT/COMPETE/FEDER under the project SECURIoTESIGN with reference number POCI-01-0145-FEDER-030657, and by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, co-funded by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT.

Key words: Chi-square test, Kolmogorov-Smirnov test, P2P Traffic, Support Vector Machine, SVM

1 Introduction

According to the 2020 report from Sandvine [1], 80% of the current Internet traffic is generated by three key application classes: video, gaming, and social sharing. Among these applications, video corresponds to the largest traffic volume. More specifically, video streaming grew its overall traffic share during lockdown, which included accelerated video releases to streaming, binge-watching multiple seasons of TV shows, search for entertainment and information on what is happening in the world and video traffic from social networks like TikTok. Among video streaming applications, we pay a particular attention in this paper to peer-to-peer (P2P) video streaming. According to the global application total traffic share in 2020 reported by Sandvine [1], BitTorrent is the fourth most used application/platform after YouTube, NetFlix and HTTP-based streaming.

For P2P media streaming, users can take advantage of their aggregated upload bandwidth capacity for efficiently distributing video content among themselves. However, P2P traffic, including BitTorrent traffic, is difficult to detect, prioritize or mitigate, namely inside organizations, specially when protocol obfuscation techniques are used.

Streaming sessions among peers can last for long periods, which can interfere with the available network bandwidth in organizations required to perform critical network-based enterprise tasks. For this reason, Internet Service Providers (ISPs) and network administrators in organizations consider the identification and classification this type of traffic as an important matter, enabling to appropriately managing resource allocation and planning future network growth [2, 3].

On the other hand, nowadays P2P traffic is often encrypted and has varying packet lengths. It is important to classify encrypted multimedia P2P traffic to properly manage the network's resources. In that context, recognizing the different types of apps that use the network's resources and classify them is a pre-requirement that contributes for an advanced management of the network, such as providing quality of service (QoS) and price, besides identifying anomalies.

P2P multimedia applications can affect the performance of servers, services or critical applications of organizations or tasks dependent on the network. In this situation, a network administrator may need to impose limitations on P2P traffic, by limiting the transmission rate, differentiating services or even blocking those connections, to ensure a good performance of the internal applications, and / or to enforce rules to regulate the use of P2P systems.

The purpose of this article is to investigate the impact of both adjusting the Self C parameter and selecting a particular SVM kernel for specifically classifying multimedia P2P traffic.

2 Related Work

Recently, many studies have been carried out to classify traffic with the help of the SVMs [4–18]. Some of them have optimized the kernel settings and SVM parameters to improve the classification results, such as in [6, 18]. Self C is one of the parameters of SVM, also denominated as C Penalty, corresponding to the degree of punishment and causing implications on the experimental results. It is important to properly adjust this parameter, as it will directly affect the network traffic classification effectiveness. This parameter is responsible for the optimization of the SVM, avoiding an incorrect classification, being thus a regularization parameter [19].

Several works addressed the classification of Internet traffic using SVM, as we show concisely in Table 1. However, to our knowledge, the current literature is lacking a study presenting the impact of the adjustment of specific SVM parameters for the classification of multimedia P2P traffic. Therefore, this article addresses this issue.

Table 1 Summary of the main points on traffic classification using SVM addressed in articles found in the literature. In the Performance column: Precision - P, Recall – R, Accuracy – A, F-Measure – FM.

Work	Method	Real-time Operation	Detection of Encrypted Traffic	Performance(%)
Mavroforakis *et al.* (2006) [7]	SVM	No	No	A: -
Yuan *et al.* (2010) [9]	SVM	No	Yes	A: 81.75 and 95.98
Aggarwal *et al.* (2017) [8]	SVM and Naïve Bayes	Yes	Yes	A: 88.88
Aamir *et al.* (2019) [16]	SVM + KNN + RF	No	No	A: 95 – 96.66
Rezvani *et al.* (2019) [4]	Fuzzy + SVM	No	No	A: 99.44
Tang *et al.* (2019) [5]	SVM + Wavelet (WL)	No	No	A:-
Akinyelu *et al.*(2019) [10]	SVM	No	No	A: -
Sankaranarayanan *et al.* (2019) [12]	SVM	No	Yes	A: -
Han *et al.* (2019) [13]	Entropy + SVM	No	No	-
Luo *et al.* (2019) [14]	SVM and Genetic Algorithm	Yes	No	A: 100; FM: 61 – 66.67
Budiman *et al.* (2019) [18]	SVM	No	No	A:-
Şentaş *et al.* (2020) [15]	SVM	Yes	No	A:-
Raikar *et al.* (2020) [17]	SVM, NB, Nearest Centroid	Yes	No	A: 91 - 96

3 Methodology

3.1 Classification Method

SVM takes ground on the static learning theory, which aims to provide requirements to pick a classifier that has a good performance. SVM is a supervised learning machine consisting of training and test phases for the available data groups.

It is capable of recognizing sample patterns of pre-defined classes, of supporting multiclass learning, and of implementing the *one-against-one* approach. In this ap-

proach, for k classes, $k(k-1)/2$ classifiers are built. Depending on the number of classes, each classifier is trained as if there were two classes only: the intended one and all others. For the implementation of this work, the *one-against-one* approach was used [20].

The choice of the Kernel function is vital in the learning process and classification with the SVM. This choice can have a meaningful role in the results. For Zhongsheng *et al.* in [21], when we use SVM, and properly choose the kernel functions, better results are reached.

As an example, in the training phase the SVM uses techniques to divide data that are not divided with the linear kernel function use. To determine the separation hyperplane, the *smooth* margin technique allows an error margin of the classification. In SVM's training phase, there is a parameter set by the user that specifies the allowed *smoothness* of this margin.

Some parameters of the SVM method for classification are defined by the user, including the Self C parameter. The C parameter is responsible for the optimization of the SVM, preventing the classification from being done incorrectly. Self C is the main parameter in the SVM, this parameter is responsible for the tolerance and the level of acceptance of error in the classification [22]. The application of SVM for traffic identification requires fine-tuning the algorithm and the adjustment of its parameters for the classification of multiclass traffic. A trade-off must found between the efficiency and the Accuracy of the detection. The proposed method is also applicable to encrypted network traffic.

One of the problems encountered in configuring the classification with the SVM method was the selection of the kernel and its parameter values.

In this work, we explored the usage of four different kernels for SVM: the linear kernel, the sigmoid kernel, the Radial Basis Function (RBF) kernel, and the polynomial (degree = 3) kernel. The higher the C value, the higher the probability to get all training points classified correctly [23]. The main settings for the SVM algorithm are the kernel employed and the error or cost penalty parameter C, which is beneficial in network traffic classification problems as shown in [24]. With respect to the cost variable, we tried several values in the interval $[0.1; \quad 70.0]$. In most implementations of an SVM technique (e.g., in Python), the Self C parameter comes with a default value of 1.0. Table 2 shows the parameters used for the classification.

Figure 1 shows the architecture of the classifier adopted to perform the classification. Raw data were pre-processed, extracting the distribution of the size of the packets by flows, forming a new database. This new base served as input for the SVM method, where 30% of the base sample was used for the training set, generating the training models and the other 70% of the sample was used for the test set. For classification, SVM uses the models generated in the training set together with the test set. After these procedures, we obtained the exit from the classification.

The experiments were executed on a desktop computer running Ubuntu 14.04.5 Operating System and equipped with a 64-bit Intel core i7, 2.93GHz, 6GiB of system memory.

Table 2 SVM parameters used for optimizing encrypted multimedia traffic detection.

Parameter	Value
Self C	[0.1; 70.0]
Kernel	'linear', 'sigmoid', 'RBF', 'poly'
Degree	3
Gamma	auto deprecated
Coef 0	0.0
shrinking	True
probability	False
tol	0.001
cache size	200
class weight	None
verbose	False
max iter	-1
decision function shape	ovr
random state	None

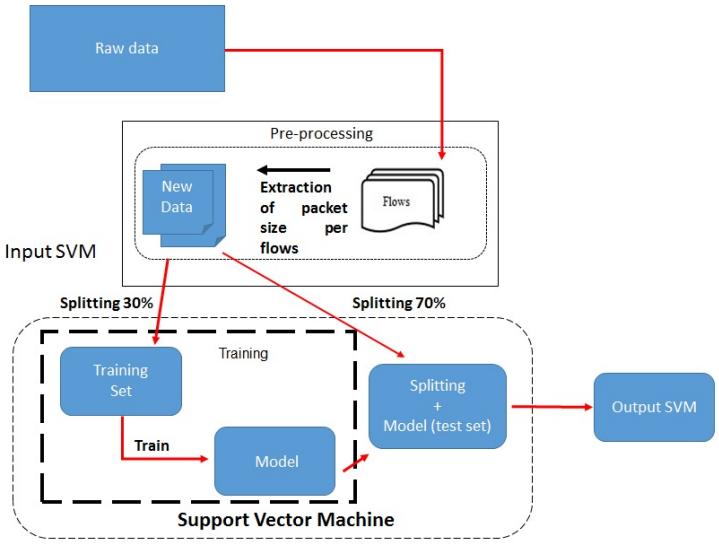

Fig. 1 Architecture of the classifier.

For classification with SVM, the sklearn module[2] provided by the scikit-learn python library [25] was used and applied to our data set. The classification was divided into 3 steps, as follows:

- Step 1 - Data treatment - Generation of the new database: For this step, a script *GeraBaseSVMNew.py* was created using the python language, whose objective

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

is to convert the raw database into a new database, which was used as input in SVM. First, we treat the flows using the tuple [source ip, destination ip, packet size] we extract from the streams the distribution of the relative frequencies of the size of the packets per stream, forming a new database.
We create buckets to calculate the distribution of the relative frequency. The features, were the buckets, where each row, has 100 columns, considered a feature. The conversion of the raw database into a new database of relative frequency was necessary to improve computational performance.
Mapping the classes - The classes were defined based on the IP of each application, for each collective file, formed a Target database with the protocols.

- Step 2 - Training and test phase - The database generated by the script in step 1, was used to generate the models (training phase) and test. To perform the tests, the models were created using the script *SVM_Multiclass.py* [26] also implemented in python, in addition, this script was used to classify and return the classification reports [27].
- Step 3 - Data validation and performance evaluation.

3.2 Dataset and Classification Features

In this research work, we use a dataset which was also described in a previously published work [28]. The data set contains approximately 25 GB of network traffic traces generated by different Internet applications and services, captured using the `tcpdump` tool and stored on disk. Since the flows were previously stored in a database, all tests carried out in this work have used offline classification only.

The data stored and generated by machines dedicated to a specific traffic, allowed us by construction to obtain the ground truth for the classes.

To accomplish step 1, it is necessary to calculate or update the cumulative probability distribution of the size of the packets per each type of flow, so that we can later obtain the values of the relative frequencies by type of flow, as shown in the table 3. With the amount of data obtained, the calculation of the distribution function was performed as follows:

- 100 buckets were defined for counting the occurrences of packet sizes.
- In each bucket, the number of observed packets having a size falling within the bounds of the bucket will be counted (Observed Frequency f_i).
- Once the observed frequencies are obtained, the relative frequencies are calculated by Eq. (1).

$$fr_i = \frac{f_i}{n}, \tag{1}$$

where n represents the total number of transmissions observed in each "Traffic Class" or "Application/Protocol"; table 3 shows the distribution of flows. The classes considered for the traffic analysis are commonly used on the Internet, and are briefly presented in Table 4.

Table 3 Definition of the buckets for the distribution of packet sizes.

Bucket	Packet size bounds	Frequency	Relative Frequency
1	0 - 15	f_1	fr_1
2	16 - 31	f_2	fr_2
3	32 - 47	f_3	fr_3
.	.	.	.
.	.	.	.
.	.	.	.
100	1584 - 1600	$f100$	$fr100$

Table 4 Analyzed traffic flows.

Application / Protocol	Traffic Class	Number of flows
Bittorrent	P2P file-sharing	961
Edonkey	P2P file-sharing	961
Gaming Runscape	P2P Video	418
Gaming War of legends	P2P Video	418
Ppstream	P2P Video	419
Sopcast	P2P Video	419
Tvu	P2P Video	418
Http, web browsing, telnet	Others [a]	179

[a]The other classes are those that are not mapped.

4 Evaluation

4.1 Classification Results

After obtaining the results (output) provided by the classifier, the results were validated through the ground truth and evaluated using the confusion matrix, the Recall, Precision and F-measure metrics as defined in [29].

The features used as entrance to our classification were relative frequencies and accumulated frequencies. The results obtained through SVM were compared to the Kolmogorov-Smirnov(KS) and Chi-squared tests [30]. KS was used with the aim to select the distribution that best represents the applications (flows). On the other hand, Chi-squared test [31] was used to compare the relative frequency distribution to the relative frequency of a distribution previously selected that represents a traffic or application class. KS is defined by [30]:

$$D = MAX_x \mid F_{1,n}(x) - F_{2,n'}(x) \mid, \tag{2}$$

where $F_{1,n}$ and $F_{2,n'}$ are the accumulated distributions that were compared and for each variable n, n' were determined, that represents the observation numbers.

Chi-squared is defined by [31]:

$$X^2 = \sum_{i=1}^{k} \frac{(x_i - E_i)^2}{E_i}, \tag{3}$$

where x_i and E_i $(0 \leq i \leq k)$ are respectively the observed and expected frequencies, and $k \in \mathbb{N}$ represents the number of buckets.

The resulting classifications using the SVM classifier with the linear, RBF, sigmoid and polynomial kernels are shown in figures 2, 3 and 4. The amount of *support* was 2091 for the P2P video class and 1922 for the P2P file-sharing. The support is the number of occurrences of the class specified in the data set. In the case of our article, it corresponds to the number of items in the class (flows).

We observe that SVM can classify multimedia traffic and that we can optimize the results by adjusting the C parameter, specifically for P2P multimedia traffic. The results demonstrate that there is an impact of the parameter self C on the classification. The factor of that impact for the values self $C =$ $[0.1;$ $70.0]$, are shown in figures 2, 3 and 4 for each SVM kernel.

The results obtained in the linear kernel with C = (0.1, 0.5) were below the values obtained with the default parameter, corresponding to 91% of Precision for the P2P video class with the self C = (0.1) and 89% of Precision for the P2P class file-sharing. For both the P2P video and P2P file-Sharing classes, we obtained the best results with the linear kernel from self C = (30.0), when the classifier reached its highest classification level for both classes, reaching 99% of Precision, 100% Recall, and 98% F-Measure, as shown in 2, 3 and 4.

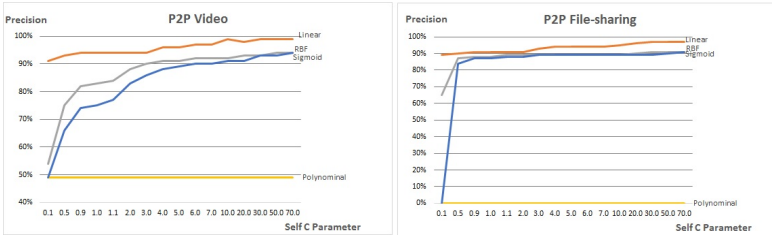

Fig. 2 Precision, as a function of Self C parameter, of SVM-based classification for P2P video and P2P file sharing traffic.

The results obtained with the RBF kernel showed a significant impact when compared to values of self C lower than the default and values greater than the default, mainly for the P2P file-sharing class. For this class, the impact was a 74% improvement in the performance of the F-measure with self C = (30.0), as shown in 4.

The calculation of F-Measure was important to evaluate the efficiency of the classification, since it represents the value of the harmonic mean between the values found for Recall and Precision. For the P2P video and P2P file-sharing classes, Precision was higher than Recall, indicating that the methodology has greater ability to reduce false positive samples (type II error), than false negative samples (type I error).

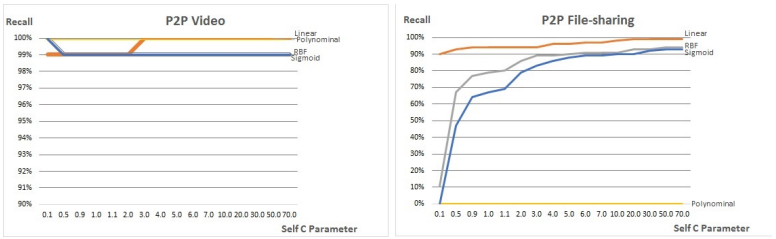

Fig. 3 Recall, as a function of Self C parameter, of SVM-based classification for P2P video and P2P file sharing traffic.

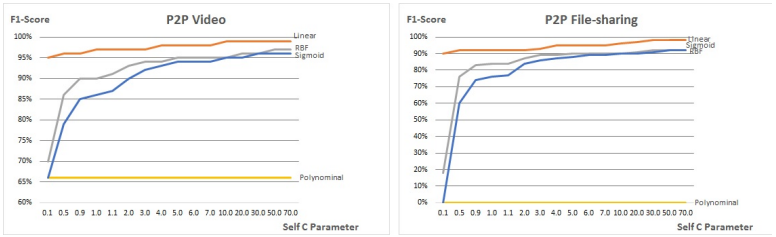

Fig. 4 F1-Score, as a function of Self C parameter, of SVM-based classification for P2P video and P2P file sharing traffic.

Analyzing the impact of self C, in the classification with the sigmoid kernel, we can see that the biggest impact was on the performance of the P2P file-sharing class. With the self C = (0.1) we have a performance so low that it reached 0% of Precision, Recall and F-measure. For self C = (70.0), we reached the highest performance point for the P2P file-sharing class where we obtained 91% of Precision, 93% of Recall and 92% of F-measure.

For the classification with the polynomial kernel, for both the P2P video class and the P2P file-sharing class, there was no impact. The performance for both classes remained the same for all tested self C values. We can conclude that given the analysis of figures 2, 3 and 4 and for our test scenario, the self C in the polynomial kernel did not have any impact on the classification performance.

For results with C = 50, it can be seen by the analysis that the P2P video and P2P file-sharing classes achieved 99% and 97% of Precision with the linear kernel and 94% and 91% of Precision, respectively, with the RBF, showing an excellent performance to discriminate how many instances are correctly classified in these classes. However, the linear kernel exhibited higher Precision results for P2P video and a slightly better one for P2P file-sharing. The P2P video classes obtained 100% of Precision with the linear and RBF kernels, and 99% of Recall, which means that both are able with high performance to identify how many of this class are encounters across the number of elements of that class. For the P2P file-sharing class, the linear kernel presented a better result for the Recall, although the result obtained for the RBF kernel is also considered. The method using the polynomial kernel obtained 49% of Precision for the P2P video class. It could not classify the

data set with the relative frequencies used in this article. The results achieved for the P2P file sharing class were very low or close to 0, for all values of C in [0.1; 70.0].

Table 5 presents a comparison among classification results obtained with SVM with linear and RBF kernels, KS, and Chi-square tests. In the classification with the KS statistical method, we obtained a Precision of 84% for P2P file-sharing and 100% for P2P Video. For P2P file-sharing and P2P video, we obtained a Recall of 56%. The F-Measure values were 67% for P2P file-sharing and 70% for P2P video. This means that the classification with the statistical method KS had a lower average performance when compared to the classification with the linear kernel associated with a C parameter in the range of 30 to 70, and with the RBF kernel associated with a C parameter in the range of 50 to 70.

Table 5 Summary of results - comparative table of the results obtained with SVM-Linear and RBF in the best range of C parameter, KS and Chi-Square.

Performance	Methods							
	Linear kernel (C=[30 -70])		RBF kernel (C=[50-70])		KS		Chi-Square	
	P2P file-sharing	P2P Video	P2P file-sharing	P2P Video	P2P file-sharing	P2P Video	P2P file-sharing	P2P Video
Precision	97%	99%	91%	94%	84%	100%	91%	100%
Recall	99%	100%	94%	99%	56%	56%	80%	74%
F-Measure	98%	99%	92%	97%	67%	70%	85%	85%

In the classification with the Chi-square statistical method, we obtained a Precision of 91% and a Recall of 80% for P2P file-sharing, and a Precision of 100%, and a Recall of 74% for the P2P video. The F-Measure values achieved 85% for P2P file-sharing and P2P video. This means that the Chi-square achieved performance average better than KS, with 15% higher for P2P video and 18% higher for P2P file-sharing. Although these values are better than compared to KS, the statistical method chi-square was low to the mean performance when compared to linear kernel and RBF kernel with the adjusted C parameter. In linear kernel with the parameter C in the range of 30 to 70, we obtained 15% more than the performance average when compared to chi-square. On RBF kernel with C parameter in the range of 50-70, we obtained 7% more than Chi-square for P2P file-sharing and 12% for P2P video.

Our results have shown that the linear kernel leads to the best classification results of P2P video with an F-Measure of 99%, which is achieved for C parameter ranging from 10 to 70. The linear kernel also leads to the best classification results of P2P file-sharing with an F-Measure of 98%, which is achieved for values of C parameter between 30 to 70.

4.2 Computational Performance

We evaluate the computational performance by measuring CPU consumption (in %) and memory consumption (in MB) during the execution time needed to classify the

database using psrecord [3]. Figures 5 and 6 show the computational performance of the linear, RBF, sigmoid, and polynomial kernels which presented the most significant results in the classification. During our tests, we have seen that the memory consumption was more significant when compared to the CPU consumption.

Analyzing the results, it can be seen that the memory is released by the process at the end of the execution of the linear kernel.

Note that the shortest execution time among the four kernels was obtained for SVM with the linear kernel, with an execution time of 1.75 seconds.

This does not happen in the RBF kernel at the end of the execution, as we can see in the graph that the process does not release the memory. The execution time of the linear kernel is relatively shorter when compared to the RBF kernel. The CPU usage (in %) is almost the same for both cases.

The computational performance of the sigmoid kernel is lower when compared to the polynomial kernel. The sigmoid kernel has an execution time which is 2 seconds shorter than the polynomial kernel. However, it has a longer execution time when compared to the linear and RBF kernels. As with the linear kernel, the memory is released as soon as the process is released. CPU consumption is about the same for both kernels. These results show that the linear kernel, in addition to showing better classification results, correctly identifies flows that belong to the class and correctly identifies flows that do not belong to the class and has a lower computational cost.

The computational costs of the classification using the KS and Chi-square methods were higher compared to the linear, RBF, poly, and sigmoid kernels. Memory consumption exceeded 600MB for both, and execution time achieved 3000 seconds for the KS method and almost 400 seconds for the Chi-square method. These execution times were considered high when compared to the ones of the SVM kernels.

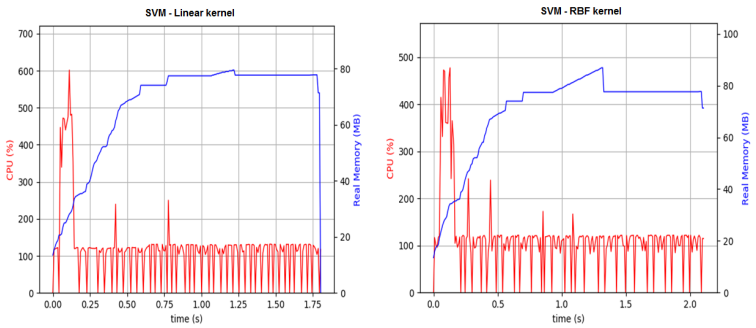

Fig. 5 Computational resource usage in terms of CPU (%) and memory (MB) of linear and RBF kernels.

[3] https://pypi.org/project/psrecord

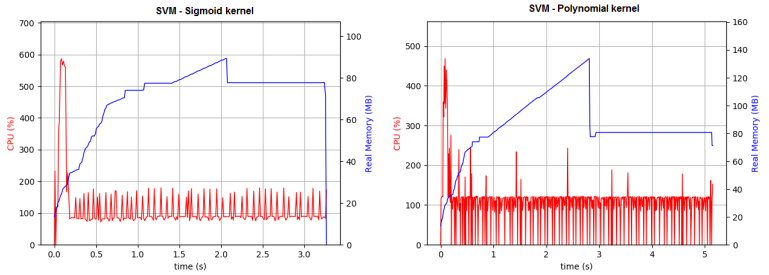

Fig. 6 Computational resource usage in terms of CPU (%) and memory (MB) of sigmoid and polynomial kernels.

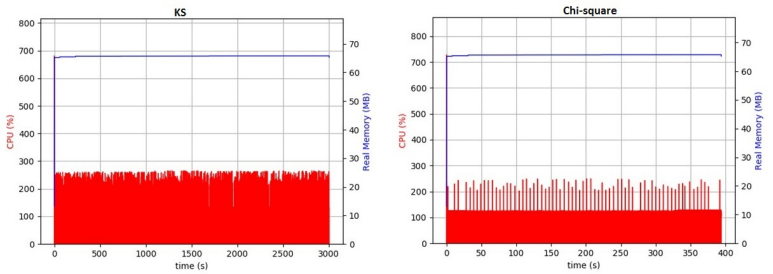

Fig. 7 Computational resource usage in terms of CPU (%) and memory (MB) of KS and Chi-square.

5 Conclusion

SVM classification has shown significantly better results for the linear kernel, RBF, and Sigmoid, when compared to the Polynomial kernel for the data set presented in this paper. These results can be attributed to the fact that SVM considers properties of the multimedia P2P traffic flow, such as the distribution of packets per flow, an important characteristic to differentiate it from the other protocols and classes found in internet traffic. With the adjustment of the self C parameter, SVM has demonstrated a high discrimination capacity for P2P protocols. The more data for the training are entered, the better the classification will be. When we increase the value of self C, we notice that the Precision and Recall values also increase. We can conclude that increasing the values in parameter C reduces type I and II errors and improves the ability to identify flows. The computational cost for the execution of the SVM method was presented taking into account the use of both CPU and memory during the classification. We have observed that over time, the CPU usage remained the same, while the memory usage increased. Our results show that SVM can indeed be a good choice for identifying multimedia P2P traffic on the internet. In comparison with the statistical methods KS and Chi-square, the linear kernel has shown the best F-measure performance for both P2P file-sharing and P2P video re-

sults. For future work, we intend to implement new classifiers for the internet traffic based on statistical methods such as distances and divergences and compare them with the ones investigated in this article.

References

1. Sandvine, "The global internet phenomena report covid-19 spotlight." https://www.sandvine.com/covid-internet-spotlight-report?hsCtaTracking=69c3275d-0a47-4def-b46d-506266477a50\%7Cac52173f-34c1-42df-8469-a091e7219e7a, May 7, 2020.
2. J. Yang, L. Yuan, C. Dong, G. Cheng, N. Ansari, and N. Kato, "On characterizing peer-to-peer streaming traffic," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 175–188, 2013.
3. K. Pal, M. C. Govil, M. Ahmed, and T. Chawla, "A survey on adaptive multimedia streaming," in *Recent Trends in Communication Networks*, pp. 185–202, IntechOpen, 2019.
4. S. Rezvani, X. Wang, and F. Pourpanah, "Intuitionistic fuzzy twin support vector machines," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 11, pp. 2140–2151, 2019.
5. J. Tang, X. Chen, Z. Hu, F. Zong, C. Han, and L. Li, "Traffic flow prediction based on combination of support vector machine and data denoising schemes," *Physica A: Statistical Mechanics and its Applications*, vol. 534, p. 120642, 2019.
6. I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *Telkomnika*, vol. 14, no. 4, p. 1502, 2016.
7. M. E. Mavroforakis and S. Theodoridis, "A geometric approach to support vector machine (svm) classification," *IEEE transactions on neural networks*, vol. 17, no. 3, pp. 671–682, 2006.
8. R. Aggarwal and N. Singh, "A new hybrid approach for network traffic classification using svm and naïve bayes algorithm," *Int. J. Comput. Sci. Mobile Comput*, vol. 6, pp. 168–174, 2017.
9. R. Yuan, Z. Li, X. Guan, and L. Xu, "An svm-based machine learning method for accurate internet traffic classification," *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, 2010.
10. A. A. Akinyelu and A. E. Ezugwu, "Nature inspired instance selection techniques for support vector machine speed optimization," *IEEE Access*, vol. 7, pp. 154581–154599, 2019.
11. J. Xiao, "Svm and knn ensemble learning for traffic incident detection," *Physica A: Statistical Mechanics and its Applications*, vol. 517, pp. 29–35, 2019.
12. S. Sankaranarayanan and S. Mookherji, "Svm-based traffic data classification for secured iot-based road signaling system," *International Journal of Intelligent Information Technologies (IJIIT)*, vol. 15, no. 1, pp. 22–50, 2019.
13. W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Information Security*, vol. 13, no. 2, pp. 109–116, 2019.
14. C. Luo, C. Huang, J. Cao, J. Lu, W. Huang, J. Guo, and Y. Wei, "Short-term traffic flow prediction based on least square support vector machine with hybrid optimization algorithm," *Neural processing letters*, vol. 50, no. 3, pp. 2305–2322, 2019.
15. A. Şentaş, İ. Tashiev, F. Küçükayvaz, S. Kul, S. Eken, A. Sayar, and Y. Becerikli, "Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type and color classification," *Evolutionary Intelligence*, vol. 13, no. 1, pp. 83–91, 2020.
16. M. Aamir and S. M. A. Zaidi, "Clustering based semi-supervised machine learning for ddos attack classification," *Journal of King Saud University-Computer and Information Sciences*, 2019.

17. M. M. Raikar, S. Meena, M. M. Mulla, N. S. Shetti, and M. Karanandi, "Data traffic classification in software defined networks (sdn) using supervised-learning," *Procedia Computer Science*, vol. 171, pp. 2750–2759, 2020.
18. F. Budiman, "Svm-rbf parameters testing optimization using cross validation and grid search to improve multiclass classification," *Scientific Visualization*, vol. 11, no. 1, pp. 80–90, 2019.
19. M. Singla and K. Shukla, "Robust statistics-based support vector machine and its variants: a survey," *Neural Computing and Applications*, pp. 1–22, 2019.
20. T. Marwala, "Support vector machines," *Handbook of Machine Learning, Wold Scientific*, pp. 97–112, 2018.
21. W. Zhongsheng, W. Jianguo, Y. Sen, and G. Jiaqiong, "Traffic identification and traffic analysis based on support vector machine," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 2, p. e5292, 2020.
22. Z. Fan and R. Liu, "Investigation of machine learning based network traffic classification," in *2017 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 1–6, IEEE, 2017.
23. K.-B. Duan and S. S. Keerthi, "Which is the best multiclass svm method? an empirical study," in *International workshop on multiple classifier systems*, pp. 278–285, Springer, 2005.
24. J. Velasco-Mata, E. Fidalgo, V. González-Castro, E. Alegre, and P. Blanco-Medina, "Botnet detection on tcp traffic using supervised machine learning," in *International Conference on Hybrid Artificial Intelligence Systems*, pp. 444–455, Springer, 2019.
25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
26. Vanice-ufmt. https://github.com/Vanice-ufmt/Codigo, October 30, 2020.
27. C. Reports". `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`, April 28, 2020.
28. V. C. Cunha, A. A. Zavala, P. R. Inácio, D. Magoni, and M. M. Freire, "Classification of encrypted internet traffic using kullback-leibler divergence and euclidean distance," in *International Conference on Advanced Information Networking and Applications*, pp. 883–897, Springer, 2020.
29. J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.
30. M. Neto, J. V. Gomes, M. M. Freire, and P. R. Inácio, "Real-time traffic classification based on statistical tests for matching signatures with packet length distributions," in *2013 19th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, pp. 1–6, IEEE, 2013.
31. N. Pandis, "The chi-square test," *American journal of orthodontics and dentofacial orthopedics*, vol. 150, no. 5, pp. 898–899, 2016.