A MapReduce-Based Distributed SVM for Scalable Data Type Classification

Chong Jiang[1], Ting Wu[1], Jian Xu[1], Ning Zheng[1], Ming Xu[1], and Tao Yang[2(✉)]

[1] Internet and Network Security Laboratory, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
{141050032,jian.xu,nzheng,mxu}@hdu.edu.cn, peterwuting@hotmail.com

[2] The Third Research Institute of Ministry of Public Security, Hangzhou, China
yangtao@stars.org.cn

Abstract. Data type classification is a significant problem in digital forensics and information security field. Methods based on support vector machine have proven the most successful across varying classification approaches in the previous work. However, the training process of SVM is notably computationally intensive with the number of training vectors increased rapidly. In this study, we proposed parallel distributed SVM (PDSVM) based on Hadoop MapReduce for scalable data type classification. First the map phase determines support vectors (SVs) in the splits of dataset by running the sequential minimal optimization. Then the reduce phase merges SVs and computes the degree of global convergence. Finally, PDSVM utilizes the global convergence SVs to get SVM model. The experimental results demonstrate that PDSVM can not only process large scale training dataset, but also perform well in the term of classification accuracy.

Keywords: Data type classification · Digital forensics · Support vector machine · Distributed · MapReduce

1 Introduction

The capacity of digital equipment storage is increased rapidly with the development of the computer technology. The quantity of electronic data that stored in the hard drive is in exponential growth from KB, MB to GB, TB, PB, simultaneously. Data deluge has become a severe problem should be solved in digital forensics domain. The classification of data or file fragments is an important problem in digital forensics, particularly for the purpose of carving fragmented files [1]. Unfortunately, the substantial research efforts focused on the accuracy of classification over a decade, seldom investigators take the efficiency of classification into account.

The classification of data or file fragment means there are two work to be resolved: data type classification and file type classification [2, 3]. For each concept, Erbacher and Mulholland's [4] made a clear definition:

"Data type: Indicative of the type of data embedded in a file." (p. 56)

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 115–126, 2017.
DOI: 10.1007/978-3-319-59288-6_11

"File type: The overall type of file. This is often indicated by the application used to create or access the file." (p. 56)

From the above statements, we can find a problem as Vassil et al. [1] said file types can have very loose, ambiguous, and extensible set of rules, especially for complex file types. A compound file ".docx" can have different data type content, which may embedded JPG images, OLE objects, Excel spreadsheet, etc. Besides, fragments which come from varieties of file types may belong to one data types. The result of file type classification may produce effectively meaningless classification rates and confusion matrices. Therefore Vassil et al. introduced data encoding classification instead of file type classification. Motivation by the Vassil et al's work, we use data type classification instead of data and file fragment classification. We use data type to indicate both file type and data type. For convenience, we put the data or file fragment referred to as data fragment in the remainder of this article.

Support vector machine (SVM) as one of machine learning techniques have gained popularity in terms of their application for data type classification problem. Numerous researchers have engaged in the development of sequences-based SVM for data type classification [1–3, 7]. Their evaluation results show sequences-based SVM perform better than other machine learning techniques in term of accuracy. However the training dataset (MB) of evaluation is far less than storage capacity (GB, TB) of digital devices in the real world [6]. Among them, one of the excellent works is a multithread-based SVM developed by Beebe et al. [3]. Modern digital drives involve millions or billions of data fragments, classifying such huge amount of data by one computer requires much more time. SVM's compute and storage requirements increase rapidly with the number of training vectors, putting big data classification task out of its reach [7]. Efficient parallel distributed algorithms and implementation techniques are essential to meeting the scalability and performance requirements for classification task.

MapReduce is a well-known parallel distributed frame work for processing big data first introduced by Google [29]. It has currently become a major enabling technology in support of large-scale data intensive applications [10]. In this article, we present a novel MapReduce-based parallel distributed SVM (PDSVM) model for large scale classifying data fragment using the information of byte frequency distribution and statistical measurements. From Google's Hadoop MapReduce perspective [28], training data be split over a cloud computing system's data nodes. Each Map task will process the associated data chunk to obtain a respective set of Support Vectors (SVs). The Reducer will aggregate the SVs to find out global support vectors. The global support vectors of the final SVM are used to evaluate a global convergence is whether or not reached. If not, then the whole process will be repeated until the global optimum is reached. The basic idea behind this approach is to filter out non-support vectors, and then merge local support vectors to save as global support vectors.

The remainder of the paper is structured as follows. In Sect. 2, we briefly describe related work to support our hypothesis. In Sect. 3, we fully represent the design and implementation of the parallelized distributed SVM algorithm for data type classification. In Sect. 4, we introduce our experimental setup to facilitate replication and comparative evaluation. In Sect. 5, we discuss the results of experiments. In Sect. 6, we conclude the paper and point out some limitations and contributions of this study.

2 Related Work

Data fragment type classification problem has been conducted in the past many years, especially in the digital forensics domain [1–3, 6, 14–18]. Previous work that explores the application of a combination of machine learning techniques and statistical analysis to solve the problem of data fragment classification. And SVM achieved the best data type classification accuracy on the most machine learning techniques in related literature.

Li et al. (2010) [16] made use of the histogram of the byte values as feature vectors to classify high entropy file fragments. They used a private dataset of 14080 KB and file type including jpg, mp3, pdf and dll. For the classification prediction accuracy, they achieved 81.5%. However, the number of file type is too small and other significant high entropy file type (specially, rar or zip) don't be take into account. Fitzgerald et al. (2011) [6] used unigrams, bigrams, and several complexity measures as feature vectors and 4800 KB as the maximum experimental setting. They achieved 48% classification accuracy across 24 file types. Beebe et al. used several statistical measures similar as Fitzgerald, moreover, a combination of unigram and bigram frequencies and other byte frequency-based measures as feature vectors. They achieve 73.4% classification accuracy across 38 files and data types.

As we can see, the size of dataset used to build SVM classifier is quite small in previous work. All of them used a single computer with SVM for file type classification, except Beebe et al., none of them consider scalable problem. SVM training is a computationally intensive process when the size of training dataset is very large. Besides, SVM classifier's compute and storage requirements increase rapidly with the number of training vectors, putting classification problem of digital forensics research out of their reach. Although Beebe et al. produces the Sceadan tool, they achieved sector-level data processing capacity by multithread and optimized code. However, the key to meeting the scalability and performance requirements is efficient parallel and distributed techniques.

Distributed SVM techniques [6–10] partition the large training dataset into small data chunks and process each chunk in parallel by utilizing the resources of a cluster of computers [17–21]. The approaches based on message passing interface (MPI) [18] have been proposed for distributed implementation of SVM based on a decomposition technique that splits the problem into smaller quadratic programming sub-problems and combine the outcomes of each sub problems. However, MPI is realized in homogeneous computing environments and has limited support for fault tolerance. Graf et al. [22] have been proposed a SVM algorithm that the training data is partitioned and a SVM is solved for each partition. The support vectors from each pair of classifiers are subsequently combined into a new training dataset for which an SVM is solved. The process carries on until a single final classifier is left, namely Cascade SVM.

It is to be noted that several distributed SVM algorithms are implemented on MapReduce frameworks [7, 8, 10]. Mapreduce is a distributed framework and programming model for processing large scale data in a parallel and distributed manner. Mapreduce manages data by its Distributed File System and organizes data process into three phases: map, shuffle and reduce. All data is treated as a key-value pair from a distributed file

system, produces a set of intermediate key-value pairs by the application of a map function, then group all intermediate value by key, finally reduce function process each group with different keys. A typical MapReduce workflow is described pictorially in Fig. 1.

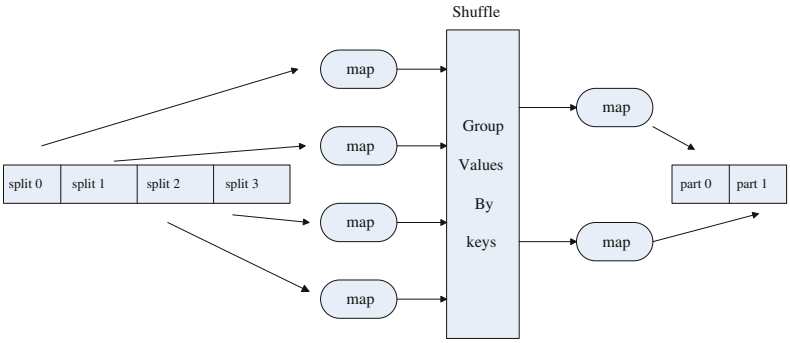

Fig. 1. Illustration of the MapReduce workflow

Sun and Fox [23] implemented a parallel SVM based Twister MapReduce framework. In this model, the training dataset is split into smaller subsets. Each subset is trained, in a distributed note, with a SVM model. The support vectors of each note are taken as the input of next layer SVMs. The global SVM model is obtained through such iteration. Catak and Balaban [24] implemented a parallel SVM based on the Hadoop MapReduce framework early. To the practical application of distributed SVM, Ku et al. [7] took advantage of a distributed SVM for email classification. Alham et al. [9, 21] implemented a parallel SVM based hadoop MapReduce for large scale image classifications and annotation. However, to the best of our knowledge, there is no parallel distributed SVM solution available for data type classification.

3 The Proposed Solution: PDSVM

In this section, we describe how to parallelize the SVM algorithm based on MapReduce in detail. The basic principle of the design of our parallel implementation is similar to Kun et al. [27]. The parallel distributed SVM (PDSVM) architecture is illustrated in Fig. 2 and the program is presented as follows.

At the first, we partition the entire training dataset into several smaller subsets (TD1 … TD4) by balanced random sampling and each of the subset is allocated to a Map worker machine. Each map task is then run Sequential Minimal Optimization (SMO) algorithm to compute the associated training subset in corresponding Map worker machine. SMO was developed by Platt [25] as one of the fastest quadratic programming (QP) optimization algorithm. To solving this QP problem, SMO breaks this problem into series of smallest possible QP problem, using Osuna's theorem to ensure convergence. Every time SMO selects two Lagrange multipliers α_i and α_j parameters to optimize by a heuristic method of choice. So the output of each Map task is support vectors (SV1…SV4) which corresponds to Lagrange multipliers $\alpha > 0$ in local

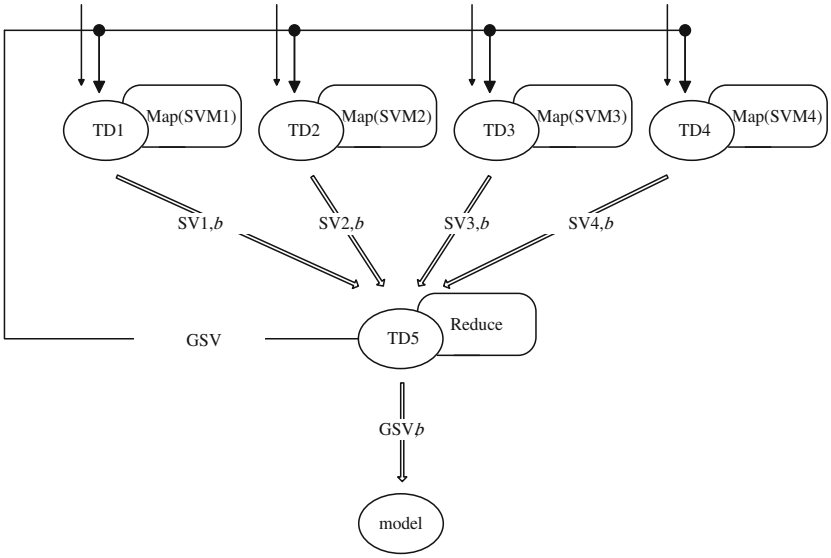

Fig. 2. The architecture of PDSVM.

partition and bias threshold b. Which then combined as input is forwarded to the reduce task, reducer joins the partial support vectors to produce the global support vectors (GSV) which will be tested for global convergence restriction. Here we use parameter λ (λ = the current iteration's number of GSV/the previous iteration's number of GSV) to determine the level of convergence, which used to observe the change of the total number of GSV. If $\lambda > 0.95$, means the number of GSV has changed little, global convergence is achieved, and vice versa. Owing to in normal conditions, the number of GSV has little change and the convergence for global optimum is very near after two times of iteration. Our experiment also proved this. So the time of iteration is rarely.

In addition, the reducer has to deal with the bias threshold b, as this value is different for each partition. If the global convergence is not achieved, then return the GSV of reducer into Map worker machines and make the next iteration computation. Otherwise the reducer calculates the weighted average of b for all partitions to obtain global b. Then the output of reduce is GSV and bias threshold b, which are used to training SVM model will be used in the classification.

Hence PDSVM algorithm can guarantee that each iteration takes full advantage of Hadoop cluster's parallel computation ability providing excellent scalability. Furthermore, the parameter λ can reduce the number of unnecessary iterative calculation, ensure the ultimate SVM in term of accuracy can be guaranteed at the same time. The simple description of PDSVM algorithm is represented in the form of pseudo code showing Algorithm 1.

Algorithm 1. Implement of PDSVM

Input: training dataset x;
Output: global support vectors (GSV) and weighted average of bias value b;
Preparation: data partition and allocate to the computation node;
1 Map_i,(i is equal to the number of sub-dataset)
2 input: training sub-dataset x_i;
3 running SMO algorithm to train allocated sub-dataset x_i;
4 output: support vectors and bias threshold b;
5 Reduce
6 input: the result (SVs and b) from each map node;
7 calculate parameter λ. if $\lambda \leq 0.95$, broadcast support vectors to each map node and go to step 2, otherwise proceed to the next step;
8 calculate GSV and weighted average of b;
9 use global SVs and b to get SVM classification model

4 Experimental Setup

4.1 Data Collection and Preprocess

In our experiments, the data is derived from open available corpus of dataset—Govdocs dataset and synthesized dataset. Fourteen well-known kinds of data types are chose for our experiment evaluation. The smallest available sector size on currently magnetic media is 512-bytes, so we select 512-bytes as the fragment size. We used a random sampling without replacement technique to collect files for each file type from Govdocs dataset. Besides, we downloaded some not copyright files and create some synthesized files to augment dataset for the ground truth.

For each of the fourteen data types, we would have at least 10000 files composed of at least 9000000 512-bytes fragments (see Table 1). And we removed the header and tail segments of each file to avoid potential data signature bias in the SVM training phase. Since the header segments frequently contains an information of data type identifier and the tail segments might not be 512-bytes in length. The aforementioned processing enables us to generate a 120 GB dataset of file fragments.

The classification problem of data type classification usually involve more than two classes. Researchers have proposed various approaches to solve multiclass problems with SVMs such as One Against Rest (OAR), One Against One. In this study, we used LIBLINEAR [26] which implement multiclass classification with SVMs by OAR approach.

Table 1. Specification of the datasets.

Data type	Number of 512-bytes fragments
CSV	18244684
BASE64	16312485
TXT	15364415
XML	16644902
LOG	18947086
JAVA	16368603
MP3	17144629
GIF	18349146
BMP	20511396
AVI	18774634
WMV	21929567
PDF	20166485
FLV	16957853
DOC	19052705

4.2 Feature Vectors Selection

In order to apply SVMs to data type classification, we need vectors of features to represent each data fragment and to discriminate different data types from each other. N-gram bytes have been found highly classification performance in previous work hence we follow the same choice of features. We selected 256 and 256^2 features which are the histogram of the byte values (i.e. the unigram and bigram) for the file fragment.

We also used several statistical measurements of data fragments as feature vectors as outlined in the here. The Shannon entropy of the unigram and bigram, as the entropy is a measure of randomness of the data. Low, medium and high ASCII frequency with range 0x00-0x1F, 0x20-0x7F and 0x80-0xFF in the block respectively. The Hamming weight, mean byte value and standard deviation of byte values are taken into account.

4.3 Environment Configuration

We have implemented PDSVM for our distributed data type classification which is developed using the Weka package. The Hadoop cluster for the set of experiments consists of ten machines each having a 4-core 2.4 GHz Xeon CPU and 12GBs of RAM. Ubuntu Linux12.04.5 and Apache Hadoop 1.2.1 were installed on the all machines.

The main goal of our experiment was to compare the accuracy and scalability of data type classification. For this purpose, the set of data fragments was partitioned into a training set and a testing set in a, roughly 5-to-1 ratio. The experiment made decisions on each data fragment in dataset belongs to which group with respect to a specific type, can be divided into four groups: True Position (TP), True Negative (TN), False Position (FP) and False Negative (FN). The scalability and accuracy measurements are defined as follows:

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

$$\text{Speedup} = \frac{\text{computing time on one machine}}{\text{computer time on cluster}} \tag{2}$$

5 Experiment Results

Experimental results showed that our approach produced very good performance. The performance evaluation of the proposed PDSVM with respect to accuracy and scalability is presented in Fig. 3, 4 and 5. Our classifier not only achieved the similar classification accuracy, but also spent much less time comparing with the previous work. The promoted method's average classified accuracy achieved 80.92% that is significantly better than random chance (1/14).

	CSV	BASE64	TXT	XML	LOG	JAVA	MP3	JPG	BMP	AVI	WMV	PNG	FLV	DOC
CSV	100%													
BASE64		100%												
TXT			99%											
XML			2%	98%										
LOG			1%		98%	1%								
JAVA			1%			97%							1%	
MP3							94%	3%						
JPG							8%	84%				3%	4%	
BMP									82%	10%	2%			5%
AVI								7%		77%	6%	3%	2%	3%
WMV							6%	1%		1%	74%	5%	3%	
PNG							2%	8%	4%	6%		61%	5%	7%
FLV								7%		6%	10%	7%	54%	7%
DOC							9%	9%	6%	3%		14%	3%	53%

Fig. 3. Confusion matrix of experiment result

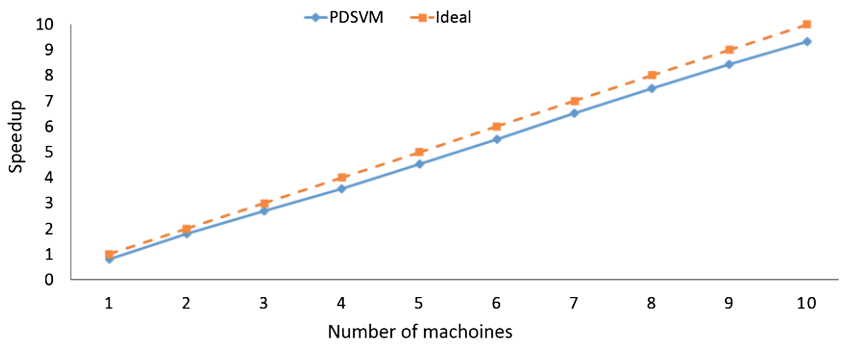

Fig. 4. The performance of speedup

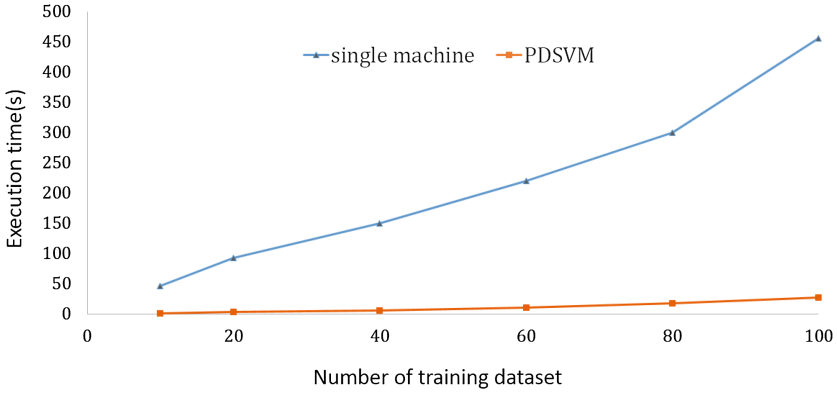

Fig. 5. The computation efficiency of PDSVM

Table 2 shows the accuracy of the experimental result achieved for 120 G data fragment processed by PDSVM with ten machine Hadoop cluster. Figure 3 shows the confusion matrix for the experimental result in detail. The classification on low entropy data types (e.g. CSV, BASE64, TXT, etc.) achieved the best performance and the worst on the high entropy data types (e.g. PDF, FLV, etc.). Therefore one of the future investigative work in data type classification is to improve the classification performance on the high entropy data types.

Table 2. Data fragment classification accuracy of experiment

Type	Accuracy
CSV	100%
BASE64	100%
TXT	99%
XML	98%
LOG	98%
JAVA	97%
MP3	94%
GIF	84%
BMP	82%
AVI	77%
WMV	74%
PDF	61%
FLV	54%
DOC	53%

To determine the scalability performance, we maintained the number of training dataset constant and increased the number of machines in the Hadoop cluster. We used formula 2 to calculate the speed of PDSVM with work machines increased. We executed PDSVM using only one machine and then we added additional machine. Figure 4 shows

the speedup values along with the number of machines increases. PDSVM shows nearly linear speedup as the number of machines increases and the slope approximating the ideal slope.

We also maintained the number of machines constant and enhanced the number of training dataset. We set the number of machines to one and ten respectively, then we used LIBLINEAR running SVM on single machine and PDSVM running on ten machines. With the number of training dataset increased, the running time overhead is described in the Fig. 5. The experimental shows that the running time of distributed SVM outperformance the sequential SVM with an increasing number of training dataset.

6 Conclusion

Data fragment classification is a well-known critical problem in digital forensics and network security, such as data recovery, reverse engineering and intrusion detection and so on. For this paper, we explored the application of distributed frame work to this problem. We generated a large dataset of data fragment for 14 kinds of file type, which is difficult for classical SVM model to process large scale dataset in a short time. We proposed a parallel distributed SVM (PDSVM) algorithm based on iterative Hadoop MapReduce, which can improve the computation speed greatly. We performed several experiments to evaluate the accuracy and scalability of PDSVM for the data type classification. Our experiments show that PDSVM not only can tackle large scale training dataset, but also performs well in the term of classification accuracy.

As part of the future work, we are planning to deeply analyze more data types and to infer the relationship of each data type, for more clearly discriminate data fragment with different data type. We also are planning to consider load balancing schemes to optimize the performance of PDSVM in a dynamic heterogeneous environment.

Acknowledgments. This work is support by Natural Science Foundation of China under Grant No. 61070212 and 61572165, the State Key Program of Zhejiang Province Natural Science Foundation of China under Grant No. LZ15F020003 and Key Lab of Information Network Security of Ministry of Public Security.

References

1. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: an open gridservices architecture for distributed systems integration. Technical report, Global Grid
2. Zheng, N., Wang, J., Wu, T., et al.: A fragment classification method depending on data type. In: IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. IEEE (2015)
3. Beebe, N.L., Maddox, L.A., Liu, L., et al.: Sceadan: using concatenated n-gram vectors for improved file and data type classification. IEEE Trans. Inf. Forensics Secur. **8**(9), 1519–1530 (2013)

4. Erbacher, R.F., Mulholland J.: Identification and localization of data types within large-scale file systems. In: International Workshop on Systematic Approaches to Digital Forensic Engineering, pp. 55–70. IEEE Computer Society (2007)
5. Beek, H.M.A.V., Eijk, E.J.V., Baar, R.B.V., et al.: Digital forensics as a service: game on. Digital Invest. **15**, 20–38 (2015)
6. Fitzgerald, S., Mathews, G., Morris, C., et al.: Using NLP techniques for file fragment classification. Digital Invest. **9**(15), S44–S49 (2012)
7. Xu, K., Wen, C., Yuan, Q., et al.: A MapReduce based parallel SVM for email classification. J. Networks, **9**(6) (2014)
8. Ke, X., Jin, H., Xie, X., et al.: A distributed SVM method based on the iterative MapReduce. In: IEEE International Conference on Semantic Computing (ICSC), pp. 116–119. IEEE Computer Society (2015)
9. Çatak, F.Ö.: Polarization measurement of high dimensional social media messages with support vector machine algorithm using MapReduce (2015)
10. Guo, W., Alham, N.K., Liu, Y., et al.: A resource aware MapReduce based parallel SVM for large scale image classifications. Neural Process. Lett., 1–24 (2015)
11. Na, G., Shim, K., Moon, K., Kong, S., Kim, E., Lee, J.: Frame-based recovery of corrupted video files using codec specifications. IEEE Trans. Image Process. **23**(2), 517–526 (2014)
12. Moody, S.J., Erbacher, R.F.: SÁDI - statistical analysis for data type identification. In: International Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE 2008, Berkeley, California, USA, May, pp. 41–54 (2008)
13. Zhang, L., White, G.B.: An approach to detect executable content for anomaly based network intrusion detection. In: 21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26–30 March 2007, Long Beach, California, USA, pp. 1–8 (2007)
14. Amirani, M.C., Toorani, M., Mihandoost, S.: Feature-based type identification of file fragments. Secur. Commun. Networks **6**(1), 115–128 (2013)
15. Amirani, M.C, Toorani, M., Beheshti, A.: A new approach to content-based file type detection. In: Computer Science, pp. 1103–1108 (2008)
16. Li, Q., Ong, A., Suganthan, P., et al.: A novel support vector machine approach to high entropy data fragment classification (2010)
17. Hazan, T., Man, A., Shashua, A.: A parallel decomposition solver for SVM: distributed dual ascend using fenchel duality, pp. 1–8 (2008)
18. Do, T.N., Poulet, F.: Classifying one billion data with a new distributed SVM algorithm. In: International Conference on Research, Innovation and Vision for the Future, pp. 59–66 (2006)
19. Chang, E.Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z.: PSVM: parallelizing support vectormachines on distributed computers. In: Proceedings of Advances in Neural Information Processing Systems, pp. 257–264 (2007)
20. Zhu-Hong, Y., Jian-Zhong, Y., Lin, Z., Shuai, L., Zhen-Kun, W.: A MapReduce based parallel SVM for large-scale predicting protein-protein interactions. Neurocomputing **145**, 37–43 (2014)
21. Guo, W., Alham, N.K., Liu, Y., et al.: A resource aware MapReduce based parallel SVM for large scale image classifications. Neural Process. Lett., 1–24 (2005)
22. Graf, H., Cosatto, E., Bottou, L., Durdanovic, I., Vapnik, V.: Parallel support vectormachines: the cascade SVM. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) (2004)
23. Sun, Z., Fox, G.: Study on Parallel SVM Based on MapReduce (2013)
24. Çatak, F.O., Balaban, M.E.: CloudSVM: training an SVM classifier in cloud computing systems. In: Proceedings of the Pervasive Computing and the Networked World—Joint International Conference (ICPCA/SWS), pp. 57–68 (2012)

25. Platt, J.: Sequential minimal optimization: a fast algorithm for training support vector machines. Technical report, MSR-TR-98-14, Microsoft Research (1998)
26. Fan, R.E., Chang, K.W., Hsieh, C.J., et al.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**(9), 1871–1874 (2008)
27. Kun, D., Yih, L., Perera, A.: Parallel SMO for training support vector machines, SMA 5505, project final report (2003)
28. Apache Hadoop. http://hadoop.apache.org
29. Ghemawat, S., Gobioff, H., Leung, S.: The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), pp. 29–43 (2003)