Using SVM and Clustering Algorithms in IDS Systems

Peter Scherer, Martin Vicher, Pavla Dráždilová, Jan Martinovič, Jiří Dvorský,, and Václav Snášel

Department of Computer Science, FEI, VSB – Technical University of Ostrava, 17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{peter.scherer, martin.vicher, pavla.drazdilova, jan.martinovic, jiri.dvorsky, vaclav.snasel}@vsb.cz

Abstract. Intrusion Detection System (IDS) is a system, that monitors network traffic and tries to detect suspicious activity. In this paper we discuss the possibilities of application of clustering algorithms and Support Vector Machines (SVM) for use in the IDS. There we used K-means, FarthestFirst and COBWEB algorithms as clustering algorithms and SVM as classification SVM of type 1, known too as C-SVM. By appropriate choosing of kernel and SVM parameters we achieved improvements in detection of intrusion to system. Finally, we experimentally verified the efficiency of applied algorithms in IDS.

Key words: Intrusion Detection System, K-means, Farthest First Traversal, COBWEB/CLASSIT, SVM, clustering

1 Introduction

Three criteria are important for computer systems security: confidentiality, integrity and availability. Computer security is defined as a protection against threads for these criteria. The major manners of computer security are techniques like user authentication, data encryption, avoiding programming errors and firewalls. They are known as first line of defense. The last line of defense is used *Intrusion Detection System* (IDS). An Intrusion Detection System is software application (device respectively) that monitors network and system activities for malicious attempts, threads or policy violations and produces reports and statistics. Several machine-learning paradigms including soft computing approach [2], neural networks and fuzzy inference system [11], genetic algorithms [14], Bayesian network, matrix factorization approach [16], multivariate adaptive regression splines etc. have been investigated for the design of IDS. In this paper we investigate and evaluate the performance of Farthest First Traversal, K-means, COBWEB/CLASSIT clustering algorithms and classification via Support Vector Machines. The motivation for using the clustering algorithms and SVM is to improve the accuracy of the Intrusion Detection System.

2 Clustering Algorithms and Their Classification

Cluster analysis is the process of grouping the objects (usually represented as a vector of measurements, or a point in a multidimensional space) so that the objects of one cluster are similar to each other whereas objects of different clusters are dissimilar.

Clustering is the unsupervised classification of objects (observations, data items, instances, cases, patterns, or feature vectors) into groups, *clusters*. In [4] author cite that from a machine learning perspective, clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. Therefore, clustering is unsupervised learning of a hidden data concept.

The applications of clustering often deal with large datasets and data with many attributes. Clustering is related to many other fields. The classic introduction to clustering in pattern recognition is given in [7]. Machine learning clustering algorithms were applied to image segmentation and computer vision [12].

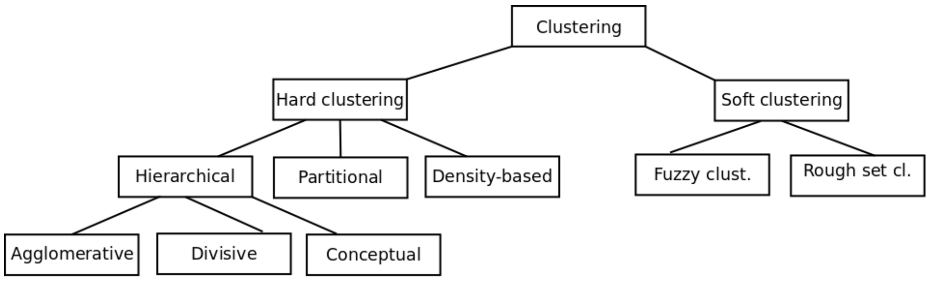

Fig. 1. A taxonomy of clustering approaches.

2.1 Classification of Clustering Algorithms

The various clustering algorithms can be classified according to how they create clusters of objects. Such division of clustering algorithms is shown in Fig. 1.

For our intention of using the clustering algorithms in an IDS, we need algorithms that can determine the jurisdiction of the object X to cluster, even if the object X was not included in the set of objects, from which we generate clusters. For this purpose we chose the algorithms K-means, Farthest First Traversal (they are partitional algorithms) and Cobweb/CLASSIT (this is a conceptual clustering algorithm).

Partitional Algorithms Partitional algorithms divide the objects into several disjoint sets and creates a one level of non-overlapping clusters. But the problem is to determine how many clusters has algorithm detect.

Algorithms of Conceptual Clustering Algorithms of conceptual clustering create by incremental way, the structure of the data by division of observed objects into subclasses. The result of these algorithms is a classification tree. Each node of the tree contains the objects of its child nodes, so root of this tree contains a all objects. According to the above classification are a these algorithms hierarchical, incremental algorithms that combine both – aggregation and division approach.

2.2 Farthest First Traversal

Farthest first traversal (FFT) algorithm is partitional clustering algorithm. This algorithm first select K objects as the centers of clusters and then assign other objects into the cluster (according to measure of dissimilarity to centers of the clusters). The first center of cluster is chosen randomly, the second center of cluster as most dissimilar to first center of cluster and every other center of cluster is chosen as the one whose value of measure of dissimilarity [9] to the previously selected centers of the clusters is greatest.

2.3 K-means

Algorithm K-means, according to the classification above is partitional clustering algorithm. The main idea of the algorithm is to find K centers (one for each cluster) of clusters. The question is, how choose these centers of clusters, because this choice will significantly affect the resulting clusters. The best would be to pick center of cluster least similar to each other. The next step is assign each object from data set to the center of cluster, to which is most similar. Once this occurs, the next step in the classification is to determine the new center of each cluster (centers are derived from clusters of objects). Again, is performed the classification of objects into different clusters according to their dissimilarity [9] with new centers of clusters. These steps are repeated until we find out that centers of clusters no longer change or until is achieved maximum number of repetitions.

2.4 COBWEB/CLASSIT

This incremental clustering algorithm creates a hierarchical structure of clusters by using four operators (operator for creating a new cluster, inserting an object into an existing cluster, union of two clusters into one cluster and splitting cluster into two clusters) [8] and the categorization utility [15]. When processing object into the cluster is always used one of the operators, but always are tested all four operators and categorization utility evaluate distribution of clusters after applying one of the operator. Finally, as the resulting distribution is chosen distribution that was evaluated (by using a categorization utility) as the best.

3 Classification SVM of type 1 (C-SVM) and their parameters

3.1 Support Vector Machines Classifier

Support Vector Machine (SVM) is a preferably technique for linear binary data classification. In [10] authors state that a classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value (i.e. the class labels) and several attributes (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

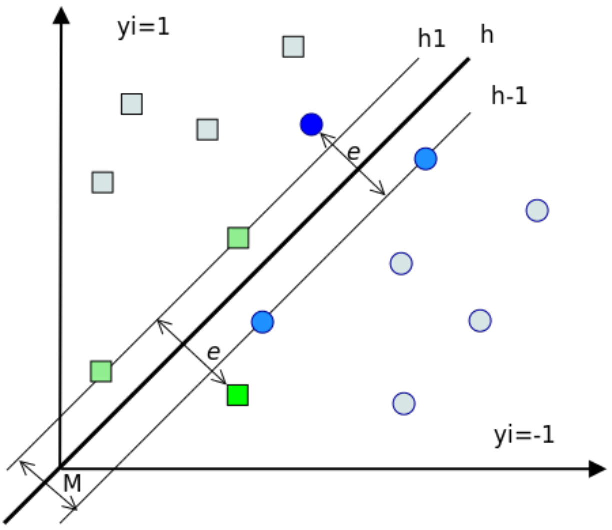

Fig. 2. General linear binary classification case.

Given a binary training set $(\boldsymbol{x}_i, y_i)$, $\boldsymbol{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, $i = 1, \ldots, m$, the basic variant of the SVM algorithm attempts to generate a separating hyper-plane in the original space of n coordinates (x_i parameters in vector $\boldsymbol{x}$) between two distinct classes, Fig. 2. During the training phase the algorithm seeks for a hyper-plane which best separates the samples of binary classes (classes 1 and -1). Let $h_1 : \boldsymbol{wx} + b = 1$ and $h_{-1} : \boldsymbol{wx} + b = 1$ ($\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^n, b \in \mathbb{R}$) be possible hyper-planes such that majority of class 1 instances lie above h_1 and majority of class -1 fall below h_{-1}, whereas the elements coinciding with h_1, h_{-1} are hold for Support Vectors. Finding another hyper-plane $h : \boldsymbol{wx} + b = 0$ as the best separating (lying in the middle of h_1, h_{-1}), assumes calculating $\boldsymbol{w}$ and b, i.e. solving the nonlinear convex programming problem. The notion of the best separation can be formulated as finding the maximum margin M that separates the data from both classes. Since $M = 2\left\|\boldsymbol{w}\right\|^{-1}$, maximizing the margin cuts down to minimizing $\|\boldsymbol{w}\|$ Eq.(1).

$$\min_{\boldsymbol{w},b} \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 + C\sum_i \varepsilon_i \tag{1}$$

with respect to: $1 - \varepsilon_i - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \leq 0$, $-\varepsilon_i \leq 0$, $i = 1, 2\ldots, m$

Regardless of having some elements misclassified (Fig. 2) it is possible to balance between the incorrectly classified instances and the width of the separating margin. In this context, the positive slack variables ε_i and the penalty parameter C are introduced. Slacks represents the distances of misclassified points to the initial hyper-plane, while parameter C models the penalty for misclassified training points, that trades-off the margin size for the number of erroneous classifications (bigger the C smaller the number of misclassifications and smaller the margin). The goal is to find a hyper-plane that minimizes misclassification errors while maximizing the margin between classes. This optimization problem is usually solved in its dual form (dual space of Lagrange multipliers):

$$\boldsymbol{w}^* = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{2}$$

where $C \geq \alpha_i \geq 0$, $i = 1, \ldots, m$, and where $\boldsymbol{w}^*$ is a linear combination of training examples for an optimal hyper-plane. However, it can be shown that $\boldsymbol{w}^*$ represents a linear combination of Support Vectors $\boldsymbol{x}_i$ for which the corresponding α_i Langrangian multipliers are non-zero values. Support Vectors for which $C > \alpha_i > 0$ condition holds, belong either to h_1 or h_{-1}. Let x_a and x_b be two such Support Vectors ($C > \alpha_a, \alpha_b > 0$) for which $y_a = 1$ and $y_b = -1$. Now b could be calculated from $b^* = 0.5\boldsymbol{w}^*(\boldsymbol{x}_a + \boldsymbol{x}_b)$, so that classification (decision) function finally becomes:

$$f(\boldsymbol{x}) = \operatorname{sgn}\sum_{i=1}^{m} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^* \tag{3}$$

To solve non-linear classification , one can propose the mapping of instances to a so-called feature space of very high dimension: $\varphi : \mathbb{R}^n \to \mathbb{R}^d$, $n \ll d$ i.e. $\boldsymbol{x} \to \varphi(\boldsymbol{x})$. The basic idea of this mapping into a high dimensional space is to transform the non-linear case into linear and then use the general algorithm already explained above Eqs. (1), (2), and (3). In such space, dot-product from Eq. (3) transforms into $\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x})$. A certain class of functions called *kernels* [6] for which $k(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})\cdot\varphi(\boldsymbol{y})$ holds, are called kernels. They represent dot-products in some high dimensional dot-product spaces (feature spaces), and yet could be easily recomputed into the original space. As example was chosen a Radial Basis Function Eq. (4), also known as Gaussian kernel [1], and was one of implemented kernels in the experimenting procedure.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma\left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2) \tag{4}$$

Now Eq. (3) becomes:

$$f(\boldsymbol{x}) = \text{sgn} \sum_{i=1}^{m} \alpha_i y_i k(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^* \tag{5}$$

After removing all training data that are not Support Vectors and retraining the classifier, the same result would be obtained [6] by applying the function above. Thus, one depicted, Support Vectors could replace the entire training set, which is the central idea of SVM implementation.

4 Experiments

The data used for training and testing was prepared by the Agency DARPA intrusion detection evaluation program in 1998 at MIT Lincoln Labs [13]. Experiments were performed on a collection containing five pairs of data sets: the learning set (5092 vectors of 42 attributes) and testing set (6890 vectors of 42 attributes). Each pair represents a learning and testing data for one type of five classes of network attacks. Individual vectors describing the network traffic are described by 41 attributes (range $0 - 1$, is therefore not necessary to normalization). The 42^{nd} attribute was used in learning process. The attribute determines type of network attack in the question. In the case of testing, the existence of the attribute was neglected. We measure only classification accuracy of the vector, that describes the network attack.

4.1 Classification Using SVM type 1 (C-SVM)

It is necessary to determine the appropriate combination of parameters C and γ for better efficiency. In our experiment, the parameter C is in the range of 2^{-5} and 2^{15} in increments of powers of 2 and a parameter γ is in the range of 2^{-15} and 2^{3} in increments of powers of 2. We used 110 combinations of parameters C γ in total. In the case of same results of prediction with different parameters C and γ, the combination of parameters with the lowest time-intensive calculation model was chosen. In Tables 1,2, 3, and 4 is possible to see the best result combination.

The four most utilized kernel functions (linear, polynomial, RBF and sigmoid) was used for process of learning. As technology, we used library LibSVM [5].

4.2 Classification with Algorithm Farthest First Traversal

During experiments with the algorithm Farthest First Traversal we tried to reveal the effect of number of generated clusters on success rate of the classification of network traffic, and on training time. The measure used by this algorithm was cosine measure. Tables 5 and 6 shows results of each experiments with algorithm FFT. Of these it is possible to deduce that the time of training increases with the number of generated clusters. We tried to optimize this algorithm by using data structure KD-tree. Training time of this algorithm with and without using

Table 1. Classification using linear kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.71	2^{-1}	2^{-1}	99.55
Probe	0.25	2^{3}	2^{-1}	99.81
DOS	0.35	2^{7}	2^{-3}	99.81
U2R	0.17	2^{3}	2^{-3}	99.80
R2L	0.35	2^{5}	2^{-5}	99.64

Table 2. Classification using polynomial kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.78	2^{13}	2^{-7}	99.83
Probe	0.24	2^{-3}	2^{-1}	99.81
DOS	0.47	2^{9}	2^{-5}	97.18
U2R	0.16	2^{15}	2^{-5}	99.80
R2L	0.24	2^{15}	2^{-5}	99.71

Table 3. Classification using RBF kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.88	2^{1}	2^{-3}	99.87
Probe	0.26	2^{5}	2^{-5}	99.90
DOS	0.29	2^{15}	2^{-7}	99.88
U2R	0.18	2^{9}	2^{-3}	99.83
R2L	0.37	2^{13}	2^{-7}	99.75

Table 4. Classification using sigmoid kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.95	2^{5}	2^{-5}	99.58
Probe	0.38	2^{7}	2^{-5}	99.88
DOS	0.43	2^{15}	2^{-9}	99.83
U2R	0.20	2^{5}	2^{-3}	99.83
R2L	0.42	2^{11}	2^{-7}	99.65

of KD-tree is shown in Tables 5 and 6. As you can see in the Tables 5 and 6 training time of this algorithm with using KD-tree was reduced by almost half. Table 7 presents the results of the algorithm FFT with using a KD-tree for each class of attack.

Table 5. Results of algorithm FFT for class of attack Normal without using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	2.99	74.82
20	6.89	74.73
30	8.42	81.86
40	12.72	77.90
50	15.21	77.29
100	25.24	82.03

Table 6. Results of algorithm FFT for class of attack Normal with using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	1.64	74.82
20	5.38	74.73
30	4.54	81.86
40	5.96	77.90
50	7.51	77.29
100	16.31	82.03

4.3 Classification with Algorithm K-means

During experiments with algorithm K-means we tried to reveal the influence of the number of generated clusters on training time and success rate of the network traffic classification. The measure that was used by this algorithm was cosine measure. In Tables 8, 9 and 10 are shown results for each experiment. Of these it is possible to deduce that the time of training is increasing with the number of generated clusters. We tried to optimize this algorithm by using data structure KD-tree. Training time of this algorithm with and without using of KD-tree is shown in Tables 8 and 9. As you can see in the Tables 8 and 9, training time of this algorithm with using KD-tree not declined as significantly as at algorithm FFT. For certain number of generated clusters was training time even worse than at algorithm without using KD-tree. This is due overhead of

Table 7. Results of algorithm FFT for each class of attack with using KD-Tree.

Attack type	Training time (s)	Accuracy (%)
Normal	5.96	84.92
Probe	5.94	98.77
DOS	6.18	82.64
U2R	5.85	95.04
R2L	5.99	99.27

creating KD-tree in each iteration of the algorithm and for a small number of generated clusters is more effective search cluster, where object fall, sequentially than by using KD-tree. Table 10 presents the results of algorithm K-means using a KD-tree for each class of attack.

Table 8. Results of algorithm K-means for class of attack Normal without using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	29.53	94.71
20	46.69	99.93
30	60.89	98.64
40	74.88	99.62
50	82.24	99.46
100	147.70	98.27

4.4 Classification with Algorithm COBWEB/CLASSIT

To achieve the best success rate is necessary to determine values of parameters Acuity and Cutoff. These parameters must be selected manually and is not known method how select the best combination. Based on experiments with the values of these parameters, when the values for the parameter Acuity were changed in the interval 0.225 to 0.01 with step 0.025 with the constant value of parameter Cutoff 0.1 and experiments when parameter Acuity had constant value 0.1 and values of parameter Cutoff were changed in the interval $0.1 - 1$ with step 0.1. We have chosen values for parameter Acuity 0.1 and for parameter Cutoff 0.6. Table 11 shown the results of the algorithm COBWEB/CLASSIT for each class of attack.

Table 9. Results of algorithm K-means for class of attack Normal with using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	36.21	94.71
20	49.83	99.93
30	56.92	98.64
40	67.88	99.62
50	71.20	99.46
100	107.68	98.27

Table 10. Results of algorithm K-means for each class of attack with using KD-Tree.

Attack type	Training time (s)	Accuracy (%)
Normal	71.80	99.46
Probe	79.14	98.19
DOS	98.59	99.91
U2R	95.04	99.97
R2L	101.11	97.46

Table 11. Results of algorithm COBWEB/CLASSIT for each class of attack.

Attack type	Training time (s)	Accuracy (%)
Normal	284.72	83.73
Probe	356.98	97.79
DOS	260.07	83.12
U2R	265.33	93.58
R2l	216.78	97.92

Table 12. Classification using SVM.

Attack type	SVM kernel			
	linear	polynomial	RBF	sigmoid
Normal	99.550	99.830	99.870	99.580
Probe	99.810	99.810	99.900	99.880
DOS	99.810	97.180	99.880	99.830
U2R	99.800	99.800	99.830	99.830
R2L	99.640	99.710	99.750	99.650
Average	99.722	99.266	99.846	99.754

Table 13. Classification using clustering algorithm.

Attack type	FFT	K-means	COBWEB/CLASSIT
Normal	84.92	99.46	83.73
Probe	98.77	98.19	97.79
DOS	82.64	99.91	83.12
U2R	95.04	99.97	93.58
R2L	99.27	97.46	97.92
Average	92.128	98.998	91.228

5 Conclusion

In this paper we have described the method for the illustrated prediction accuracy by using clustering algorithms and SVM in the IDS. In Table 13 for each used algorithm is shown success rate for each class of attack. The best average success rate has SVM algorithm, more than 99% (best of all is algorithm SVM that is using the RBF kernel, it has a success rate 99.722%). The average success rate of other algorithms was between 91.228% and 98.998%. It will be useful to compare these two methods on other document collections. In our future work we will investigate other kernel functions to search for better attacks prediction in the IDS, SVM paralelization and optimalization clustering algorithms.

Acknowledgment

This work is partially supported by Grant of Grant Agency of Czech Republic No. 205/09/1079, and SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2011/172.

References

1. S. Abe. Support Vector Machines for pattern classification. London, Springer, 2005.
2. A. Abraham and R. Jain. Soft Computing Models for Network Intrusion Detection Systems. Classification and Clustering for Knowledge Discovery Studies in Computational Intelligence, p. 191–207, 2005.
3. B. Al-Shboul and S.-H. Myaeng. Initializing k-means using genetic algorithms, 2009.
4. P. Berkhin. A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data, p. 25–71, 2002.
5. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001 `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
6. N. Cristiani and J. Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge, Cambridge University Press, 2000.
7. R. Duda and P. Hart. Pattern Classification and Scene Analysis. John Wiley & Sons, New York, 1973.

8. D. H. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. Kluwer Academic Publisher, 1987.
9. G. Gan, C. Ma, and J. Wu. Data Clustering Theory. Algorithms and Applications. ASASIAM, 2007.
10. C. Hsu, C. Chang and C. Lin. A Practical Guide to Support Vector Classification, journal Bioinformatics, 2003.
11. S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham and S. Sanyal. Adaptive Neuro-Fuzzy Intrusion Detection Systems, International Conference on Information Technology: Coding and Computing (ITCC'04), 2004.
12. A. K. Jain and P.J. Flynn. Image segmentation using clustering. In Advances in Image Understanding: A Festschrift for Azriel Rosenfeld, IEEE Press, 65–83, 1996.
13. MIT Lincoln Laboratory `http://www.ll.mit.edu/IST/ideval/`
14. S. Owais, V. Snasel, P. Kromer and A. Abraham. Survey: Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques, p. 300–307, Computer Information Systems and Industrial Management Applications, 2008.
15. N. Sahoo. Incremental hierarchical clustering of text documents. adviser: Jamie Callan, 2006.
16. V. Snasel, J. Platos, P. Kromer, A. Abraham. Matrix Factorization Approach for Feature Deduction and Design of Intrusion Detection Systems, p. 172–179, The Fourth International Conference on Information Assurance and Security, 2008.