Fast Regularized Least Squares and k-means Clustering Method for Intrusion Detection Systems

Parisa Movahedi, Paavo Nevalainen, Markus Viljanen and Tapio Pahikkala
Dept. of Information Tech., University of Turku, FI-20014, Turku, Finland

Keywords: Intrusion Detection, k-means Clustering, Regularized Least Squares, Kernel Approximation.

Abstract: Intrusion detection systems are intended for reliable, accurate and efficient detection of attacks in a large networked system. Machine learning methods have shown promising results in terms of accuracy but one disadvantage they share is the high computational cost of training and prediction phase when applied to intrusion detection. Recently some methods have been introduced to increase this efficiency. Kernel based methods are one of the most popular methods in the literature, and extending them with approximation techniques we describe in this paper has a huge impact on minimizing the computational time of the Intrusion Detection System (IDS).

This paper proposes using optimized Regularized Least Square (RLS) classification combined with k-means clustering. Standard techniques are used in choosing the optimal RLS predictor parameters. The optimization leads to fewer basis vectors which improves the prediction speed of the IDS. Our algorithm evaluated on the KDD99 benchmark IDS dataset demonstrates considerable improvements in the training and prediction times of the intrusion detection while maintaining the accuracy.

1 INTRODUCTION

In recent years, attack detection has gained tremendous attention. Internet and devices connected to it have resulted in more network traffic and confidential information moving around. Utilizing accurate, but also efficient, detection methods is therefore essential. Especially, when using Machine Learning classifiers, the prediction step has to be as efficient as possible. Anomaly detection is one of the most common machine learning tasks in this respect. It can be viewed as a classification problem distinguishing abnormal from normal traffic based on the feature characteristics of the traffic flow.

Techniques such as clustering and artificial neural networks (ANN) (Portnoy et al., 2001), kernel based methods (Kim and Kim, 2005b), genetic algorithms (Li, 2004), fuzzy logic (Kaur and Gill, 2013) and many others have been applied to IDS.

Support vector machine (SVM) (Vapnik, 1995) is one of the most popular kernel based machine learning techniques (Schölkopf and Smola, 2002) used in classification problems. It has been applied to intrusion detection problems (Sung and Mukkamala, 2003; Li et al., 2003) as well. Kernel based methods enable solving nonlinear problems by mapping the input space to a feature space in which the data is separated by a hyperplane. The kernel function implicitly defines the feature space by calculating an inner product of two given inputs.

In (Kim and Kim, 2005a), SVM is used for attack classification in intrusion detection systems. (Sotiris et al., 2010) have used one-class SVM technique for anomaly detection to a normal class and attack class of data. One drawback of SVM based methods are that SVM training requires solving a convex quadratic programming which has a steeply increasing computing cost as a function of the size of the training data.

One alternative to the quadratic programming based SVM is the regularized least squares algorithm (RLS or LS-SVM). In (Rifkin et al., 2003), RLS binary classifier is used to minimize the regularized square loss. The advantage over the well established SVM is that RLS training requires constructing and solving a single system of linear equations instead of solving a convex quadratic problem required by SVM. RLS and SVM both seek a Tikhonov regularized solution from the representing kernel Hilbert space, but they have different loss functions. The methods have comparable accuracy (Poggio and Girosi, 1990).

Our efficiency focused approach is closely related to two previous studies. In (Gao and Wang, 2006),

264 Movahedi P., Nevalainen P., Viljanen M. and Pahikkala T..
Fast Regularized Least Squares and k-means Clustering Method for Intrusion Detection Systems.
DOI: 10.5220/0005246802640269
In *Proceedings of the International Conference on Pattern Recognition Applications and Methods* (ICPRAM-2015), pages 264-269
ISBN: 978-989-758-077-2
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

LS-SVM based intrusion detection system utilizes kernel approximation and greedy search. We will compare our results to their method in terms of accuracy and computational complexity. We also compare to the latest work of (Kabir, 2014), which uses optimum allocation based LS-SVM. They have a two-step approach: first phase is an optimum allocation scheme to determine the sufficient amount of training data which is representative of the whole data set. The second phase then applies LS-SVM on this smaller training set.

Our approach has two steps, those steps are:

1. k-means clustering algorithm on the training dataset to get optimized base vectors suitable for classification to each attack type.
2. A kernel approximation method to minimize the kernel space computation and fast parameter selection methods.

Both steps introduce one new parameter for the prediction performance optimization search. New parameters are *k* and *M*, the k-means cluster number and the size of the approximation base. This arrangement has a huge impact on training and prediction time of our the intrusion detection algorithm.

This paper is organized as follows. The RLS algorithm, kernel approximation method and optimized cross validation method (Pahikkala et al., 2012) are outlined in Ch. 2. The implementation is built around RLSCore package[1]. We also present the intrusion detection system based on RLS and k-means clustering in Ch. 2. Ch. 3 contains experiments on the KDD99 intrusion detection data set and comparisons to other IDS systems. The last section presents the conclusion from the experiments.

2 METHOD

2.1 Regularized Least Squares

Given a training set $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, where the feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and the class labels $y_i \in \mathbb{R}$. $(\mathbf{x}_i, y_i)$ are independent identically distributed (*iid*) samples. The RLS formulation is to find c^* such that:

$$\mathbf{c}^* = \arg\min_c \frac{1}{n} \|\mathbf{y} - \mathbf{Kc}\|^2 + \lambda \mathbf{c}^T \mathbf{Kc} \tag{1}$$

where $\mathbf{c} = \{c_i\}_{i \in N}$ are the weights, $\mathbf{y} = \{y_i\}_{i \in N}$ are the labels, λ is the regularization parameter and $\mathbf{K} = \{K(x_i, x_j)\}_{i,j \in N}$ is the kernel matrix. In this paper we use a Gaussian kernel function:

$$K(x_i, x) = \exp\left(\frac{\|x_i - x\|^2}{2\sigma^2}\right) \tag{2}$$

2.2 Kernel Approximation

RLS approach requires $O(n^2)$ memory space for $\mathbf{K}$ matrix and $O(n^3)$ time complexity for solving the corresponding matrix equation. The steeply increasing computational costs can be overcome by using an approximation of the kernel space where only a subset of basis vectors are used. This avoids storing the entire kernel matrix in the memory and decreases the time complexity. We use the Nyström kernel approximation method used in Regularized Least Squares Classification (Rifkin et al., 2003), (Airola et al., 2011).

The training set $X_R = \{x_i | i \in R \subseteq N\}$ is made small enough to gain computational advantage over the non-sparse RLS, i.e. $\mid R \mid \ll \mid N \mid$, while seeking to not give up too much prediction performance. Denote the full data set and the new reduced training set by the corresponding index sets N and R. The training points X_R are explicit and omitted points $X_{N \setminus R}$ will be approximated as a linear combination of the points in X_R. Nyström approximation $\hat{\mathbf{K}} \approx \mathbf{K}$ of the kernel matrix is defined as

$$\hat{\mathbf{K}} = \mathbf{K}_{NR} (\mathbf{K}_{RR})^{-1} \mathbf{K}_{RN} \tag{3}$$

where $\mathbf{K}_{NR} = \{K_{ij}\}_{i \in N, j \in R}$. Similar set indexing scheme for matrices is employed in the following text.

Now instead of explicitly constructing the approximation kernel $\hat{\mathbf{K}}$ we are able to calculate its Cholesky decomposition matrix $\hat{\mathbf{C}}$:

$$\hat{\mathbf{C}} = \mathbf{K}_{NR} \mathbf{C}^{-T} \tag{4}$$

The cost of calculating $\mathbf{C}^{-1}$ is $O(|R|^3)$ and the memory complexity is $O(n|R|)$. The cost of the matrix multiplication equals $O(n|R|^2)$ which scales considerably better than straight-forward decomposition of the kernel matrix. (Airola et al., 2011) uses the Cholesky decomposition of $\mathbf{K}_{RR}$ having $\mathbf{K}_{RR} = \mathbf{CC}^T$. For comprehensive overview see (Airola et al., 2011). Similar advantage carries on to prediction computation, which can be formulated as:

$$\tilde{y} = \sum_{i \in R} b_i K(x, x_i) \tag{5}$$

where weights b_i are solved once for the whole computation (see the actual derivation from (Rifkin et al., 2003)) leading to $O(|R|)$ complexity.

[1]RLScore, https://github.com/aatapa/RLScore

2.3 k-means Clustering

One of the important tasks in kernel approximation is the selection of basis vectors i.e. the subset R because that selection has a direct impact on how well $\widehat{\mathbf{K}}$ approximates $\mathbf{K}$. There are various methods for choosing the basis vectors such as k-means clustering and uniform sampling.

One of the widely used and simple flat clustering algorithms is the k-means algorithm (Hartigan, 1975). The algorithm divides the input set to k different clusters in an iterative manner. The first iteration begins with k randomly selected cluster center candidates $\mathbf{m}_i \in \mathbb{R}^d$, which form a set $M = \{\mathbf{m}_i\}, i = 1...k$, which is one parameter of the minimization process. Each point gets assigned to the closest center candidate, and at each iteration the center candidates are updated to the new mean of the assigned points. These iterations are repeated until the algorithm converges.

The k-means clustering algorithm also minimizes the following within-cluster sum of squares (WCSS) with given k:

$$\mathbf{M}^* = \underset{M=\{\mathbf{m}_j\}}{\arg\min} = \sum_{j=1}^{k} \sum_{i\in N} \left\|\mathbf{x}_i - \mathbf{m}_j\right\|^2 \qquad (6)$$

We seek to extract cluster centroids $\mathbf{M}$ to represent the cluster entities. Therefore, defined by this measure, the k-means algorithm finds the best possible extracted cluster centroids. This makes the algorithm a safe way to find the reduced kernel set R. Reduced sets of different sizes can be obtained by simply varying k.

2.4 Proposed Framework

Our proposed framework consist of four main stages, see Fig. 1:

- Pre-processing of the data. This includes normalization and rejection of invalid entries.
- k-means clustering method Eq. (6) is used to partition the data to k clusters (centroids). The search of best k value is completely automated, we have used the range $20 \leq k \leq 100$.
- Predictor is constructed based on RLS formulation Eq. (1), using the Gaussian kernel function (2). The optimal parameters λ and σ (regularization and Gaussian scale) are found using fast cross-validation (Pahikkala et al., 2012).
- Testing performance with the test data set.

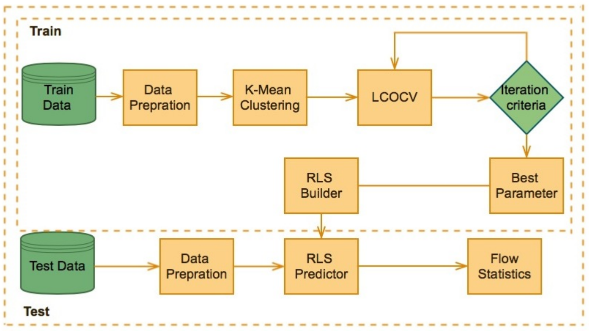

Figure 1: The Proposed IDS Framework.

The stage two provides the centroids which serve as the basis vectors for our approximated kernel function in the RLS learner. Centroids represent the average characteristics of the group of data points and the approximated kernel function is based on them. They do not co-inside with the data (they do not have labels, for example). The RLS algorithm is modified to take this into account.

The fast cross-validation method in stage three has $O(n^2)$ time complexity for the entire leave-one-out cross-validation scheme. This makes it cheaper to scan through the RLS parameter space. Basis vectors and the obtained fixed kernel parameters together form the predictor. The approximated kernel is presented in Eqs. (3) and (4).

3 EXPERIMENTS

We now present the experiments conducted to evaluate the training time and detection accuracy. The main objective of this study is to enhance the prediction time of the IDS. To be able to compare our method with other IDS methods we have chosen to use the KDD99[2]. benchmark data set which is a data set derived out of the 1998 DARPA intrusion detection dataset presented by MIT's Lincoln Lab.

KDD99 consists of five million training data instances and about two million test data instances of network flow traffic. We use the 10% of KDD labeled data, totaling 494021 training data points and 311029 test data points. Each instance of data represents a well established IP connection between two hosts, containing 41 different categorical or numerical attributes of that connection. Training data set includes 24 different attack types which can be further categorized to 4 main categories.

These main attack categories are: Denial of service attack (DoS), Prob attack, User To Root attack (U2R) and Remote to Local attack types (R2L). Including the normal state as a category (NORMAL), this sums up to 5 different categories.

[2]KDD99, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

We compare our method with the method proposed in the work of (Kabir, 2014) and (Gao and Wang, 2006). For this purpose we had to make two different subsets of KDD99 to make the environment comparable to each of their methods.

The data has attack type feature, which has categorical values: NORMAL, DOS, Probe, U2R and R2L. To correctly reflect the arrangements of (Kabir, 2014), the data was partitioned into four different data sets D1,..,D4, each containing instances of normal data and one type of attack data. For example the D1 data set consists of normal and DoS instances. To make our data compatible with RLS classifier we have converted the categorical data into numerical data and normalized all the samples to have unit norm. Since the kernel approximation parameters are different for each combination of the four classes of attacks, in this paper we have constructed four different predictors, one for each of these attack classes.

Our Environment consists of Python 2.7, 3.8 G RAM, Dual Core Dell Processor, CPU 2.26 GHz. We run the program 10 times with different train subsets and report average results of the RLS learner. We give Precision, Recall and F-value results of Fast k-means RLS algorithm, and the training and testing times of our method.

3.1 First Experiment

In this Section we will compare our Fast KM-RLS method to the other two methods of intrusion detection. First is the Layered approach using Conditional Random Fields presented by (Gupta et al., 2010) and second is the OA-LS-SVM IDS frame work presented by (Kabir, 2014). We chose the training and testing partitions similar to theirs to make the results comparable. We focus on improving the IDS prediction time therefore all comparisons are mainly based on prediction time needed for classifying each traffic instance.

3.1.1 D1: DoS Attack vs Normal Traffic

For the training phase we chose 9,000 random normal traffic instances and 9,000 DoS attack instances from the KDD99 train data set. We tested the learner on 5,000 of normal and 6,000 of DoS instances from KDD99 test data set. Optimal results for our KM-RLS were obtained by $\lambda = 4$ and $\sigma = -5$ using $k = 30$ cluster centroids. Table 1 demonstrates the results of our algorithm compared to Kabir's and Guptal's results. Test time of Fast KM-RLS for 11,000 instances is 0.12 seconds and for an instance it is 0.009 milliseconds which is much faster prediction rate than the two methods provide.

Table 1: DoS attack detection, Comparison of KM-RLS, Kabir's OA-LS-SVM and Gupta's CRF method.

Method	Precision	Recall	Fscore	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	99.84	**97.67**	**98.74**	169.23	**0.009**
OA-LS -SVM	**99.86**	97.31	98.56	**79.36**	0.4
Layered CRF	99.78	97.05	98.10	256.11	0.05

3.1.2 D2: Prob Attack Vs Normal Traffic

For the training phase we chose 9,000 random normal traffic instances and 2,000 Prob attack instances from the whole KDD99 train data set. We tested the learner on 5,000 normal and 3,000 Prob instance of the KDD99 test data set Table 2. Test time of 8,000 instance of traffic is 0.17 seconds and for one instance it is 0.01 milliseconds. Optimal detection for Prob attack was obtained by $\lambda = 1$ and $\sigma = -3$, using 80 cluster centroids. This achieved 99.73 percent accuracy on the training set.

Table 2: Prob attack detection.

Method	Precision	Recall	Fscore	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	**98.11**	**95.80**	**96.94**	517.52	**0.01**
OA-LS -SVM	97.64	90.89	94.14	**22.49**	0.2
Layered CRF	82.53	88.06	85.21	200.6	0.03

3.1.3 D3: R2L Attack Vs Normal Traffic

For the training phase we chose 1,000 random normal traffic instances and all 1,126 R2L attack instances from the whole KDD99 train data set. We tested the learner on 10,000 normal and 8,000 R2L instances of the KDD99 test data set Table 3. The test time of 18,000 instances of traffic is 0.14 seconds and for one instance it is 0.008 milliseconds. Optimal detection for the R2L attack is obtained by $\lambda = 8$ and $\sigma = -15$, using 30 cluster centroids for the basis vectors.

Table 3: R2L attack detection.

Method	Precision	Recall	Fscore	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	73.81	**97.86**	**84.15**	30.65	**0.008**
OA-LS -SVM	83.45	71.48	76.93	**3.40**	0.1
Layered CRF	**92.35**	15.10	25.94	23.40	0.09

3.1.4 D4: U2R Attack Vs Normal Traffic

For the training phase we chose 1000 random normal traffic instances and 52 U2R attack instances from the whole KDD99 train data set. We tested the learner on 5,000 normal and all the U2R instances of the KDD99 test data set Table 4. Test time of 5,078 instances is 0.09 seconds and for one instance it is 0.07 milliseconds. Optimal detection for the U2R attack is obtained by $\lambda = 2$ and $\sigma = -2$, using 70 cluster's centroids for the basis vectors.

Table 4: U2R attack detection.

Method	Precision	Recall	Fscore	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	88.83	22.45	37.09	9.40	**0.007**
OA-LS -SVM	**95.04**	38.24	**54.51**	11.68	0.2
Layered CRF	52.16	**55.02**	53.44	**8.35**	0.05

Our algorithm shows considerable improvements in the prediction time of a test point, making the Intrusion Detection system closer to real time detection. It maintained the high accuracy in detection of DoS, Prob and R2L attack, see Table 2. The U2R attack (Table 4) has also considerably better detection time but the accuracy is not as good as the other two methods.

3.2 Second Experiment

Gao presented the kernel approximation method for LS-SVM (Gao and Wang, 2006), an algorithm similar to our approach but with a different method of choosing the basis vectors and kernel parameters. We will use their choice of data set from KDD99 in order to compare our prediction time to theirs.

First we combined all the KDD99 training and test data sets and then chose 2,500 random DoS attacks and 2,500 normal traffic cases as the set D1, 2,500 Prob attack and 2,500 normal cases as the set D2, and 2500 instances of both R2L and U2R attacks with 2,500 normal data points as the set D3. We used 40% of the data for training and 60% for testing the RLS. The result of using our method and Gao's method are presented in Tables 5, 6 and 7.

Our approach is an order of magnitude faster in both training and testing times of the IDS. The accuracy stays comparable to the other IDS methods. We have come very close to a real time detection system.

Table 5: DoS attack detection, Comparison of KM-RLS, Gao's KSA-SVM and LS-SVM.

Method	Basis Vectors	Accuracy	False Alarm	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	30	**98.6**	**0.3**	**5.01**	**0.003**
KSA-SVM	46	98.5	0.4	8160	0.04
LS-SVM	600	55.02	53.44	1320	5.5

Table 6: Prob attack detection.

Method	Basis Vectors	Accuracy	False Alarm	Train Time (sec)	Test Time Per Instance (millisec)
Fast KM-RLS	30	96.9	0.4	**6.02**	**0.006**
KSA-SVM	58	**98.8**	**0.3**	13044	0.05
LS-SVM	600	82.5	3.8	2040	8.21

Table 7: U2R and R2L attack detection.

Method	Basis Vectors	Accuracy	False Alarm	Train Time (sec)	Test Time Per Instance (sec)
Fast KM-RLS	50	93.3	10.9	**8.04**	**0.006**
KSA-SVM	83	**94.6**	**10.5**	13044	0.08
LS-SVM	400	90.8	11.3	2040	2.613

4 CONCLUSIONS

To address the problem of efficiency in networked intrusion detection systems caused by the massive amount of data needed to be processed and classified, we have proposed a Fast Regularized Least Squares algorithm combined with k-means clustering. This enables us to choose optimized basis vectors to construct an approximated kernel function. We have combined this with fast cross validation techniques to select best kernel parameters in a reasonable time. Having optimized basis vectors result in a smaller comparison matrix and better detection time of each instance while maintaining accuracy.

This paper demonstrates that the detection time can be improved signicantly using existing algorithms applied already in other Machine Learning fields. Benchmarking on the KDD99 network attack data shows highly significant improvements in this regard. The problem complexity is affected by approximated kernel and by feature selection, both reduce the size of the problem independently.

In the future we will experiment with different feature selection methods to choose best traffic feature set representing each attack type. The used fast KM-RLS method allows efficient feature selection. This will keep the training time competitive while making the prediction phase even more efficient.

ACKNOWLEDGEMENTS

We like to thank for anonymous referees for the insightful comments.

REFERENCES

Airola, A., Pahikkala, T., and Salakoski, T. (2011). On learning and cross-validation with decomposed nyström approximation of kernel matrix. *Neural Processing Letters*, 33(1):17–30.

Gao, H. and Wang, X. (2006). Ls-svm based intrusion detection using kernel space approximation and kernel-target alignment. In *The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006.*, volume 1, pages 4214–4218.

Gupta, K. K., Nath, B., and Kotagiri, R. (2010). Layered approach using conditional random fields for intrusion detection. *IEEE Transactions on Dependable and Secure Computing*, 7(1):35–49.

Hartigan, J. (1975). *Clustering algorithms*. New York: John Wiley & Sons.

Kabir, M. (2014). A statistical framework for intrusion detection system. In *Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2014)*.

Kaur, H. and Gill, N. (2013). Host based anomaly detection using fuzzy genetic approach (fga). *International Journal of Computer Applications*, 74(20):5–9.

Kim, B.-J. and Kim, I. (2005a). Machine learning approach to realtime intrusion detection system. In Zhang, S. and Jarvis, R., editors, *AI 2005: Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Computer Science*, pages 153–163. Springer Berlin Heidelberg.

Kim, B.-J. and Kim, I.-K. (2005b). Kernel based intrusion detection system. In *Fourth Annual ACIS International Conference on Computer and Information Science, 2005.*, pages 13–18.

Li, H., Guan, X.-H., Zan, X., and Han, C.-Z. (2003). Network intrusion detection based on support vector machine. *Journal of Computer Research and Development*, 6:799–807.

Li, W. (2004). Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*, pages 1–8.

Pahikkala, T., Suominen, H., and Boberg, J. (2012). Efficient cross-validation for kernelized least-squares regression with sparse basis expansions. *Machine Learning*, 87(3):381–407.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*. Citeseer.

Rifkin, R., Yeo, G., and Poggio, T. (2003). Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, Cambridge, MA.

Sotiris, V. A., Tse, P. W., and Pecht, M. G. (2010). Anomaly detection through a bayesian support vector machine. *IEEE Transactions on Reliability*, 59(2):277–286.

Sung, A. and Mukkamala, S. (2003). Identifying important features for intrusion detection using support vector machines and neural networks. In *Symposium on Applications and the Internet, 2003.*, pages 209–216.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.