Sequential Intrusion Detection System Using Distance Learning (Metric Learning) and Genetic Algorithm

Ezat soleiman, abdelhamid fetanat

Abstract— since current intrusion detection systems are signature based, they still unable to detect new forms of attacks even these attacks are slightly derived from knowen ones.this paper introduces a new idea of sequential intrusion detection system using genetic algorithms and metric learning.

Index Terms— genetic algorithm, intrusion detection system, Large margin nearest neighbor, LMNN ,metric learning.

1. INTRODUCTION

THe insufficiency of traditional security tools like antivirus in facing current attacks conducts to the development of intrusion detection systems (IDS). IDSs search in the network traffic for malicious signature and then send an alarm to the user. Since current IDSs are signature based they still unable to detect new forms of attacks even if these attacks are slightly derived from known ones. So, recent researches concentrate on developing new techniques, algorithms and IDSs that use intelligent methods like neural networks, data mining, fuzzy logicm and genetic algorithms.

2. GENETIC ALGORITHM

Genetic algorithm is an evolutionary optimization technique based on the principles of natural selection and genetics. The working of genetic algorithm was explained by John Holland [1]. It has been successfully employed to solve wide range of complex optimization problems where search space is too large. GA evolves a population of initial individuals to a population of high quality individuals, where each individual represents a solution of the problem to be solved. Each individual is called a chromosome, and is composed of a predetermined number of genes. The quality of fitness of each chromosome is measured by a fitness function that is based on the problem being solved. The algorithm starts with an initial population of randomly generated individuals. Then the population is evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality. During each generation, three basic genetic operators are applied to each individual with certain probabilities, i.e. selection (rank selection, roulette wheel selection and tournament based selection), crossover and mutation.

- *Ph.D. Candidate, Dept of ICT Eng, Maleke Ashtar Univ. of Tech., Islamic Republic of Iran. Email: ezut_sol73@yahoo.com*
- *Assistant Professor, Maleke Ashtar Univ. of Tech, Islamic Republic of Iran. Email: abfetanat@gmail.com.*

2.1 Parameters in Genetic Algorithm

There are many parameters to consider for the application of GA. Each of these parameters heavily influences the effectiveness of the genetic algorithm [2].

The fitness function is one of the most important parameters in genetic algorithm. To evolve input features and network structure simultaneously, the fitness, which is based on the detection accuracy rate, includes a penalty factor for the number of operative input nodes and a penalty factor for the number of operative hidden nodes.

3. METRIC LEARNING:

3.1 Introduction

In mathematics, there are multiple feature spaces that are useful in enormous problems. For example, Euclidian space or Euclidian distance is the most popular metric that we use in our real world or Mahalanobis distance that we use to demonstrate the distance between two places in the city.

In classification and clustering problems, the majority of methods, assumes the Euclidian feature space to solve their problems and they evaluate their result in this space. One question may appear in mind is what is the best feature space for my problem? In other words, is this distance metric the best metric for my problem? In the next section we discuss about it with some example.

3.2 Metric learning with example

Based on figure 1, suppose Ali, Mohammad and Hasan were Arab and Bryan, David and Michel were American. If we want to classify them based on Euclidian distance between their houses, Ali should be American because Ali's house is next to Bryan's house and this scenario is the same with Mohammad-David and Hasan-Michel. So in this classification problem, Euclidian distance is not useful. Note that Euclidian distance is meaningful but is not useful.

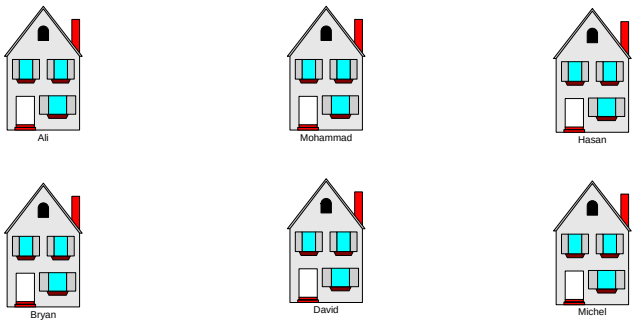

Fig. 1.

Is there any transform so that if we transform their house location, in transformed space, the Euclidian distance demonstrates their class?

Suppose we have found a transform and transformed locations were figure 2.

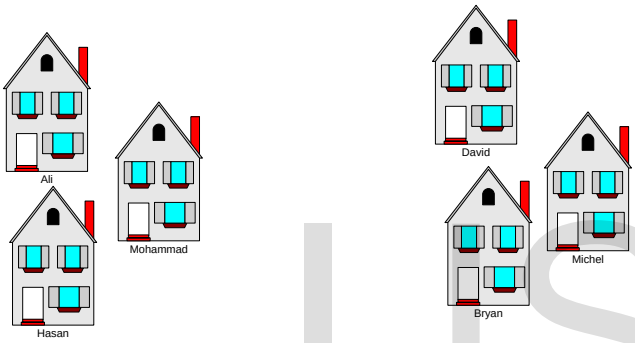

Fig. 2.

In this space, Euclidian distance is useful and it properly helps us to classify them based on Euclidian distance.

Another example is in intrusion detection systems. Suppose each connection has two features (x_1, x_2), for example (dst_host_srv_serror_rate,src_byte). In figure 3, suppose blue squares were normal connection and red circles were attack connections.

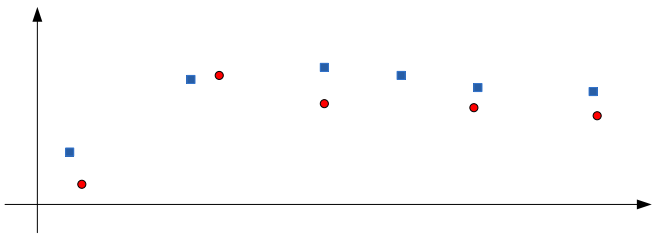

Fig. 3.

If the classification method was KNN, this input samples will produce high error rate because the intra class distances are high and inter class distances are low. If we find a transform T so that it reduces the intra class distance and increases the inter class distance, we can use KNN with low error rate.

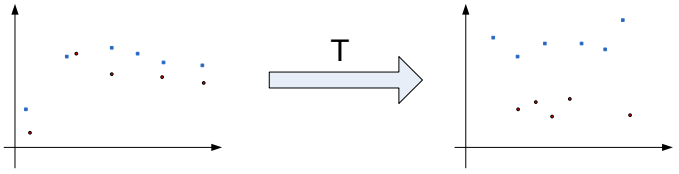

Fig. 4.

We will use LMNN method for metric learning. This method has shown its strength in classification problems. In the next section we will introduce it.

3.3 LMNN:

Large margin nearest neighbor (LMNN) classification is a statistical machine learning algorithm. It learns a Pseudo metric designed for k-nearest neighbor classification. The algorithm is based on semi definite programming, a sub-class of convex optimization [3].

The goal of supervised learning (more specifically classification) is to learn a decision rule that can categorize data instances into pre-defined classes. The k-nearest neighbor rule assumes a training data set of labeled instances (i.e. the classes are known). It classifies a new data instance with the class obtained from the majority vote of the k closest (labeled) training instances. Closeness is measured with a pre-defined metric. Large Margin Nearest Neighbors is an algorithm that learns this global (pseudo-)metric in a supervised fashion to improve the classification accuracy of the k-nearest neighbor rule [4].

The main intuition behind LMNN is to learn a pseudo metric under which all data instances in the training set are surrounded by at least k instances that share the same class label. If this is achieved, the leave-one-out error (a special case of cross validation) is minimized. Let the training data consist of a data set

$$D = \{(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)\} \subset R^d \times C \quad (1)$$

, where the set of possible class categories is

$$C = \{1, ..., c\}$$

The algorithm learns a pseudo metric of the type

$$d(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^T M (\vec{x}_i - \vec{x}_j). \quad (2)$$

For $d(.,.)$ to be well defined, the matrix M needs to be positive semi-definite. The Euclidean metric is a special case, where M is the identity matrix. This generalization is often (falsely) referred to as Mahalanobis metric.

Figure 5 illustrates the effect of the metric under varying M. The two circles show the set of points with equal distance to the center $\vec{x}_i$. In the Euclidean case this set is a circle, whereas under the modified (Mahalanobis) metric it becomes an ellipsoid.

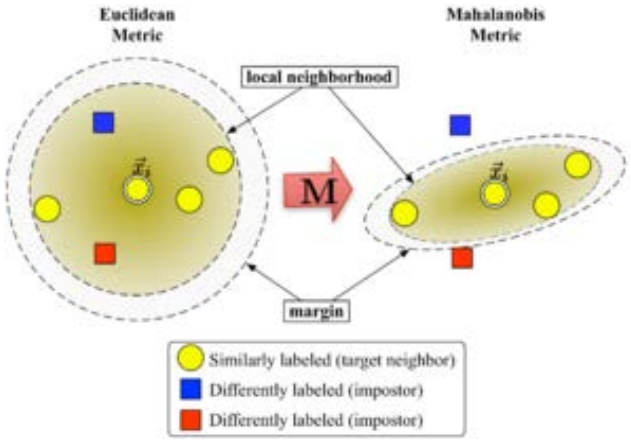

Fig. 5. Schematic illustration of LMNN.

The algorithm distinguishes between two types of special data points: target neighbors and impostors.

3.4 Target Neighbors

Target neighbors are selected before learning. Each instance $\vec{x}_i$ has exactly k different target neighbors within D, which all shares the same class label y_i. The target neighbors are the data points that should become nearest neighbors under the learned metric. Let us denote the set of target neighbors for a data point $\vec{x}_i$ as N_i.

3.5 Impostors

An impostor of a data point $\vec{x}_i$ is another data point $\vec{x}_i$ with a different class label (i.e. $y_i \neq y_j$) which is one of the k nearest neighbors of $\vec{x}_i$. During learning the algorithm tries to minimize the number of impostors for all data instances in the training set.

Large Margin Nearest Neighbors should optimize the matrix M. The objective is twofold: For every data point $\vec{x}_i$, the target neighbors should be close and the impostors should be far away. Figure 3 shows the effect of such an optimization on an illustrative example. The learned metric causes the input vector $\vec{x}_i$ to be surrounded by training instances of the same class. If it was a test point, it would be classified correctly under the $k = 3$ nearest neighbor rule.

The first optimization goal is achieved by minimizing the average distance between instances and their target neighbors

$$\sum_{i,j \in N_i} d(\vec{x}_i, \vec{x}_j) \tag{3}$$

The second goal is achieved by constraining impostors $\vec{x}_i$ to be one unit further away than target neighbors $\vec{x}_i$ (and therefore pushing them out of the local neighborhood of $\vec{x}_i$). The resulting inequality constraint can be stated as:

$$\forall i, j \in N_i, l, y_l \neq y_i$$
$$d(\vec{x}_i, \vec{x}_j) + 1 \leq d(\vec{x}_i, \vec{x}_l) + \xi_{ijl} \tag{4}$$

The margin of exactly one unit fixes the scale of the matrix M. Any alternative choice $c > 0$ would result in a rescaling of M by a factor of $\frac{1}{c}$.

The final optimization problem becomes:

$$\min_M \sum_{i,j \in N_i} d(\vec{x}_i, \vec{x}_j) + \sum_{i,j,l} \xi_{ijl} \tag{5}$$

$$\forall i, j \in N_i, l, y_l \neq y_i$$
$$d(\vec{x}_i, \vec{x}_j) + 1 \leq d(\vec{x}_i, \vec{x}_l) + \xi_{ijl} \tag{6}$$

$$\xi_{ijl} \geq 0, \; M \geq 0$$

Here the slack variables ξ_{ijl} absorb the amount of violations of the impostor constraints. Their overall sum is minimized. The last constraint ensures that M is positive semi-definite. The optimization problem is an instance of semi definite programming (SDP). Although SDPs tend to suffer from high computational complexity, this particular SDP instance can be solved very efficiently due to the underlying geometric properties of the problem. In particular, most impostor constraints are naturally satisfied and do not need to be enforced during runtime. A particularly well suited solver technique is the working set method, which keeps a small set of constraints that are actively enforced and monitors the remaining (likely satisfied) constraints only occasionally to ensure correctness[5].

4. PROPOSED METHOD:

Our proposed algorithm has overall structure in figure 6.

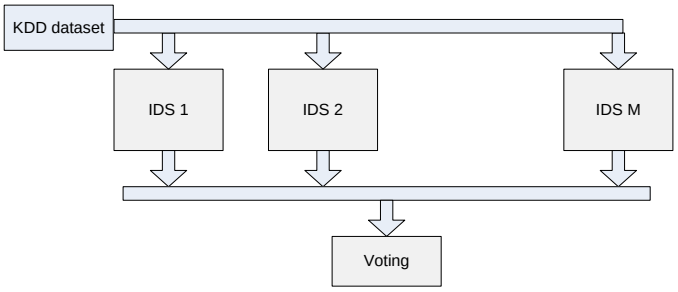

Fig. 6.

The KDD data set are passed to each intrusion detection system and voting algorithm is used to make decision based on each IDS system result. Each IDS has same structure that we will explain in next section.

IDS has two phases: The train phase and test phase. In train phase, we sample from train data frequently and in each sampling, we change the selection probability of samples. Figure 7 illustrates the training phase.

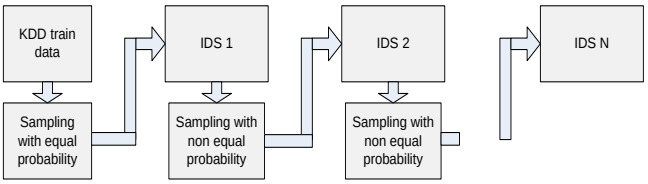

Fig. 7.

If $D=(d_1,d_2,...,d_n)$ and $S=(s_1,s_2,...,s_k)$ were the training set and samples set respectively, at the first step, each $\frac{1}{n}$ sample has equal probability. We sample k samples from train data and pass them to metric learner 1. Then we learn the learner and at the next step we decrement the probability of correctly classified samples and increase incorrectly classified samples. If the learner has C correctly classified samples and I error samples, the increment and decrement of probabilities are as follow:

If C samples were classified correctly, its probability reduced by multiplying it by factor α. So:

$$P_c^{\,t} = \alpha P_c^{\,t-1} \quad 0<\alpha,\beta<1 \qquad (7)$$

Where $P_c^{\,n}$ is the probability of correctly classified sample in time t. So, total probability of correctly classified samples will be $C\alpha P_c^{\,t-1}$.

β percent of remaining probability will be assigned to incorrectly classified samples. The amount of incorrectly classified samples will be $(k-C)$. So,

$$P_i^{\,t} = \frac{\beta}{(k-C)}(1-C\alpha P_c^{\,t-1}) \qquad (8)$$

Where $P_i^{\,t}$ is the probability of each error samples.

The remaining samples ($n-k$) will have the probability:

$$P^{t} = \frac{(1-\beta)}{(n-k)}(1-C\alpha P_c^{\,t-1}) \qquad (9)$$

Where P^t is the probability of deselected samples.

This procedure will be continued until Nth learner is trained.

Each IDS will predict the label of each sample as attack connection or normal connection with some certainty. In the voting part of figure 6, we use genetic algorithm to make final decision. As you see in figure 8, we assign a weight to each IDS result.

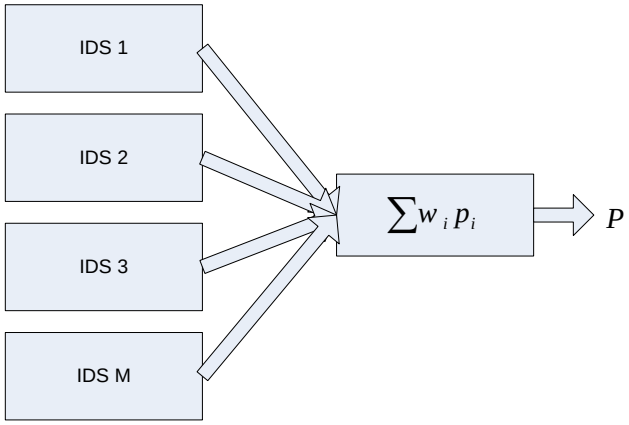

Fig. 8.

We use genetic algorithm to find optimum weights for voting. The details of genetic algorithm are as follow:

4.1 Population size:

We can start the GA with G=1000 population size.

4.2 Chromosome:

A chromosome is a binary sequence of weights like figure 9.

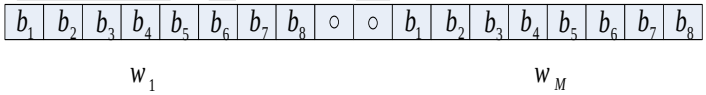

Fig. 9.

Each w_i is a signed float number with two bits for decimal part.

4.3 Fitness function:

Fitness function is the classification accuracy of train samples.

$$Fitness = \frac{n-e}{n} \qquad (10)$$

e is the error samples count.

4.4 New population generation:

The selection of parent chromosome is based on fitness function and roulette wheel selection algorithm. The cross over algorithm is two points crossover algorithm so that cross over points are selected randomly on chromosomes. The mutation is done by toggling a random bit of chromosome.

4.5 Termination condition:

The algorithm will end when the error rate reaches under the user defined threshold T or the epochs of iteration reaches to predefined value of epochs.

The test phase is simple. We pass the new test data to the system of figure 6 and based on weighted voting we can distinguish that the connection is attack or not.

5. ADVANTAGE AND DISADVANTAGES:

5.1 Advantages

- Sequential manner with unequal subsampling method will help the classifier to scatter each IDS on feature space so that each IDS be expert in own partition of feature space.
- Boosted property of proposed method will increase the accuracy of classifier because if one classifier fails to classify some data, the next classifiers can be subsidiary of previous classifiers.
- This method tries to find a suitable distance metric instead of Euclidian distance metric.
- The aggregation of each classifier results is done by GA because GA finds suitable near optimal solution for that.
- It can adaptively add an IDS when some abnormal behavior exceed some predefined threshold in the partition of abnormal behavior. So it can periodically extend itself by retraining in a period of time.

5.2 Disadvantages:

- The algorithm has some constants (α, β, k) so that we cannot set the optimal value of them. The optimal value of these parameters can change in various train data sets.
- We cannot tell the best number of IDSs.
- It is not real time in train phase.

REFERENCES

[1] H. Holland, "Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", The MIT Press, 1992.

[2] Li, W. "A Genetic Algorithm Approach to Network Intrusion Detection". http://www.giac.org/practical/GSEC/Wei_Li_GSEC.pdf. Accessed January 2005

[3] Weinberger, K. Q.; Blitzer J. C., Saul L. K. (2006). "Distance Metric Learning for Large Margin Nearest Neighbor Classification". Advances in Neural Information Processing Systems 18 (NIPS): 1473–1480.

[4] Weinberger, K. Q.; Saul L. K. (2009). "Distance Metric Learning for Large Margin Classification". Journal of Machine Learning Research 10: 207–244.

[5] Kumar, M.P.; Torr P.H.S., Zisserman A. (2007). "An invariant large margin nearest neighbour classifier". IEEE 11th International Conference on Computer Vision (ICCV), 2007: 1–8.