2nd National Conference on Information and Communication Technology (NCICT) 2011
Proceedings published in International Journal of Computer Applications® (IJCA)

Intrusion Detection through Ensemble Classification Approach

Priyanka J. Pathak
Department of Computer science and Engg.
G.H.Raisoni College of Engineering,
Nagpur, India

Snehlata S. Dongre
Department of Computer science and Engg.
G.H.Raisoni College of Engineering,
Nagpur, India

ABSTRACT
Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to bring down high profile company networks and web services. Intrusion Detection System (IDS) is an important detection that is used as a countermeasure to preserve data integrity and system availability from attacks. The main reason for using data mining classification methods for Intrusion Detection System is due to the enormous volume of existing and newly appearing network data that require processing. Data mining is the best option for handling such type of data. This paper presents various techniques used for clustering and classification used in intrusion detection systems to maximize the effectiveness in identifying attacks, thereby helping the users to construct more secure information systems.

Keywords- Data mining, Clustering, Classification, security.

1. INTRODUCTION
In increasing trends of network environment every one gets connected to the system. So there is need of securing information, because there are lots of security threats are present in network environment. A number of techniques are available for intrusion detection[6]. Data mining is the one of the efficient techniques available for intrusion detection. Data mining techniques may be supervised or unsuprevised.Various Author have applied various clustering algorithm for intrusion detection, but all of these are suffers form class dominance, force assignment and No Class problem. This paper proposes a hybrid model to overcome these problems.The most of the current research in multiple classifier systems is devoted to static environments. We assume that the classification problem is fixed and we are presented with a data set, large or small, on which to design a classifier. The solutions to the static task have marvelled over the years to such a perfection that the dominance between the classification methods is resolved by a fraction of percent of the classification accuracy. Everything that exists changes with time and so will the classification problem. The changes could be minor fluctuations of the underlying probability distributions, steady trends, random or systematic, rapid substitution of one classification task with another and so on.

A classifier (individual or an ensemble) if intended for a real application, should be equipped with a mechanism to adapt to the changes in the environment. Various solutions to this problems have been proposed over the years. Here we try to give a systematic perspective on the problem and the current solutions, and outline new research avenues.

The paper is organized as follows. Section 2 gives the details of clustering algorithms .Section 3 & Section 4 details about classification algorithm and ensemble strategies for classification for novel class detection in evolving data streams for intrusion detection.

2. CLUSTERING:
The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster[11] is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression. Although classification is an effective means for distinguishing groups or classes of objects, it requires the often costly collection and labeling of a large set of training tuples or patterns,which the classifier uses to model each group. It is often more desirable to proceed in the reverse direction: First partition the set of data into groups based on data similarity (e.g., using clustering), and then assign labels to the relatively small number of groups. Additional advantages of such a clustering-based process are that it is adaptable to changes and helps single out useful features that distinguish different groups.

Clustering [8]is an unsupervised learning technique which divides the datasets into subparts, which share common properties. For clustering data points, there should be high intra cluster similarity and low inter cluster similarity. A clustering method which results in such type of clusters is considered as good clustering algorithm.

2.1 K-means Algorithm:
K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the

previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

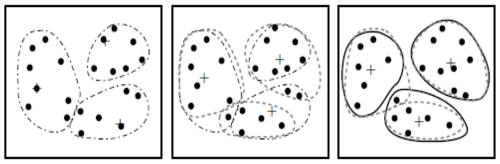


Fig: 1 Clustering of a set of objects based on the *k*-means method

The algorithm is composed of the following steps:

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:
k: the number of clusters,
D: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:
(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) repeat
(3) reassign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) until no change;

This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

2.2 The *K*-Medoids Method

The *k*-means algorithm is sensitive to outliers because an object with an extremely large value may substantially distort the distribution of data. This effect is particularly exacerbated due to the use of the *square*-error function. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used, defined as:

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

where E is the sum of the absolute error for all objects in the data set; ***p*** is the point in space representing a given object in cluster *Cj*; and ***oj*** is the representative object of *Cj*. In general, the algorithm iterates until, eventually, each representative object is actually the medoid, or most centrally located object, of its cluster. This is the basis of the *k*-medoids method for grouping *n* objects into *k* clusters.

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:
k: the number of clusters,
D: a data set containing *n* objects.

Output:
A set of *k* clusters.

Method:
(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) repeat
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, ***o***random;
(5) compute the total cost, *S*, of swapping representative object, ***oj***, with ***o***random;
(6) if *S* < 0 then swap ***oj*** with ***o***random to form the new set of *k* representative objects;
(7) until no change.

The *k*-medoids method is more robust than *k* means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the *k*-means method. Both methods require the user to specify *k*, the number of clusters.

3. CLASSIFICATION

Classification is a data mining (machine learning) technique used to predict group membership for data instances.

3.1 K-Nearest neighbors Classification Approach

A very simple classifier can be based on a nearest-neighbor approach. In this method, one simply finds in the *N*-dimensional feature space the closest object from the training set to an object being classified. Since the neighbor is nearby, it is likely to be similar to the object being classified and so is likely to be the same class as that object. Nearest neighbor methods have the advantage that they are easy to implement. They can also give quite good results if the features are chosen carefully (and if they are weighted carefully in the computation of the distance.)

The K-nearest-neighbor (KNN) algorithm measures the distance between a query scenario and a set of scenarios in the data set. Because the distance between two scenarios is dependant of the intervals, it is recommended that resulting distances be scaled such that the arithmetic mean across the dataset is 0 and the standard deviation 1. This can be accomplished by replacing the scalars with according to the following function:

$$x' = \frac{x - \bar{x}}{\sigma(x)}$$

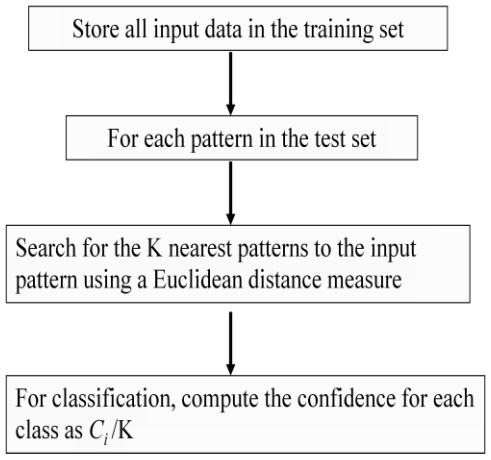
Where is the unscaled value, is the arithmetic mean of feature across the data set (see Equation 4), is its standard deviation (see Equation 5), and is the resulting scaled value. The arithmetic mean is defined as:

We can then compute the standard deviation as follows:

$$\sigma(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

The *k*-NN algorithm can also be adapted for use in estimating continuous variables. One such implementation uses an inverse distance weighted average of the *k*-nearest multivariate neighbors. This algorithm functions as follows:

1. Compute Euclidean from target plot to those that were sampled.
2. Order samples taking for account calculated distances.
3. Choose heuristically optimal *k* nearest neighbor based on RMSE done by cross validation technique.
4. Calculate an inverse distance weighted average with the *k*-nearest multivariate neighbors.


Store all input data in the training set
For each pattern in the test set
Search for the K nearest patterns to the input pattern using a Euclidean distance measure
For classification, compute the confidence for each class as C_i/K

3.2. Decision Trees

A decision tree [4] is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. Another use of decision trees is as a descriptive means for calculating conditional probabilities. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values.

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data
Partition *D*.

Input:

Data partition, *D*, which is a set of training tuples and their associated class labels;
attribute list, the set of candidate attributes;
Attribute selection method, a procedure to determine the splitting criterion that "best" partitions
the data tuples into individual classes. This criterion consists of a *splitting attribute*
and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1) create a node N;
(2) if tuples in D are all of the same class, C then
(3) return N as a leaf node labeled with the class C;
(4) if attribute list is empty then
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply Attribute selection method(D, attribute list) to find the "best" splitting criterion;
(7) label node N with splitting criterion;
(8) if splitting attribute is discrete-valued and
multiway splits allowed then // not restricted to binary trees
(9) attribute list attribute list □ splitting attribute; // remove splitting attribute
(10) for each outcome j of splitting criterion
// partition the tuples and grow subtrees for each partition
(11) let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12) if Dj is empty then
(13) attach a leaf labeled with the majority class in D to node N;
(14) else attach the node returned by Generate decision tree(Dj, attribute list) to node N;
endfor
(15) return N;
```

4. ENSEMBLE METHOD OF CLASSIFICATION

The ensemble model[10] uses combination of classifiers. Each combines a series of *k* learned models (classifiers or predictors), *M*1, *M*2, : : : , *Mk*, with the aim of creating an improved composite model, *M**.This is used to improve the accuracy of classifiers. The Bagging and boosting are the techniques used to improve the accuracy of classifiers.

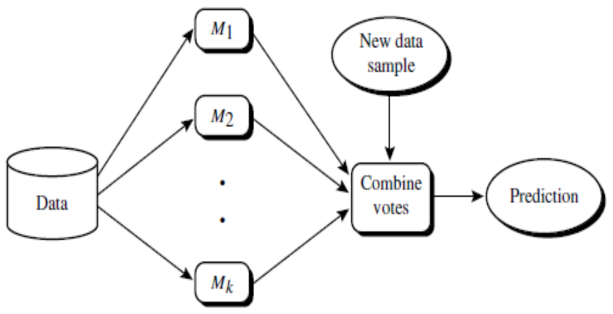


Fig. 2: Combining and predicting the accuracy of Classifiers

4.1. Bagging

For ease of explanation, it will assume at first that our model is a classifier. Suppose that you are a patient and would like to have a diagnosis made based on your symptoms. Instead of asking one doctor, you may choose to ask several. If a more than any of the others, you may choose this as the final or best diagnosis. That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote. Now replace each doctor by a classifier, and you have the basic idea behind bagging. Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

Given a set, *D*, of *d* tuples, bagging works as follows. For iteration *i* (*i* = 1, 2, : : : , *k*), a training set,*Di*, of *d* tuples is sampled with replacement from the original set of tuples,*D*. Note that the term bagging stands for *bootstrap aggregation*. Bagging can be applied to the prediction of continuous values by taking the average value. The bagged classifier (Han & Kamber 2007) often has significantly greater accuracy than a single classifier derived from *D*, the original training data. It will not be considerably worse and is more robust to the effects of noisy data. The increased accuracy occurs because the composite model reduces the variance of the individual classifiers. For prediction, it was theoretically proven that a bagged predictor will *always* have improved accuracy over a single predictor derived from *D* of each prediction for a given test tuple.

Algorithm: Bagging.

The bagging algorithm—create an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally-weighted prediction.

Input:
D, a set of *d* training tuples;
k, the number of models in the ensemble;
a learning scheme (e.g., decision tree algorithm, back propagation, etc.)
Output:
A composite model , *M_*.
Method:

```
(1) for  i = 1 to k do // create k models:
(2) create bootstrap sample, Di, by sampling D with
replacement;
(3) use Di to derive a model, Mi;
(4) endfor
```

To use the composite model on a tuple, ***X***:

```
(1) if classification then
(2) let each of the k models classify X and return the majority
vote;
(3) if prediction then
(4) let each of the k models predict a value for X and return the
average predicted value;
```

4.2. Boosting

In boosting, weights are assigned to each training tuple. A series of *k* classifiers is iteratively learned. After a classifier *Mi* is learned, the weights are updated to allow the subsequent classifier,*Mi*+1, to "pay more attention" to the training tuples that were misclassified by *Mi*. The final boosted classifier, *M_*, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy. The boosting algorithm can be extended for the prediction of continuous values. Adaboost is apopularboosting algorithm. Suppose we would like toboost the accuracy of some learning method. We are given *D*, a data set of *d* class-labeled tuples, (***X*1**, *y*1), (***X*2**, *y*2), : : :, (***Xd***, *yd*),where *yi* is the class label of tuple***Xi***. Initially, Adaboost assigns each training tuple an equal weight of 1=*d*. Generating *k* classifiers for the ensemble requires *k* rounds through the rest of the algorithm. In round *i*, the tuples from *D* are sampled to forma training set,*Di*, of size *d*. Sampling with replacement is used—the same tuple may be selected more than once. Each tuple's chance of being selected is based on its weight.

A classifier model, *Mi*, is derived from the training tuples of *Di*. Its error is then calculated Using *Di* as a test set. The weights of the training tuples are then adjusted according to how they were classified. If a tuple was incorrectly classified, its weight is increased. If a tuple was correctly classified, its weight is decreased. A tuple's weight reflects how hard it is to classify—the higher the weight, the more often it has been misclassified. These weights will be used to generate the training samples for the classifier of the next round. The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.

5. RELATED WORKS

The main challenges in novel class detection are as follows:

- Saving the training data efficiently without using much memory.
- Knowing when to classify a test instance immediately, and when to postpone the classification decision.
- Predicting the presence of a novel class quickly and correctly

The "best" classifier not necessarily the ideal choice. When solving a classification problem, many individual classifiers with different parameters will be trained. The "best" classifier will be selected according to some criteria e.g., training accuracy or complexity of the classifiers.

Drawback of Single Classifier:

1. The final decision must be wrong if the output of selected classifier is wrong.

2. The trained classifier may not be complex enough to handle the problem.

Combining a number of trained classifiers lead to a better performance than any single one Errors can be complemented by other correct classifications
Different classifiers have different knowledge regarding the problem. To decompose a complex problem into sub- problems for which the solutions obtained are simpler to understand, implement, manage and update.

6. OVERVIEW OF PROPOSED WORK

This paper introduces a method about clustering and classification and ensemble classification techniques. In this, ensemble classification method is used to increase the performance of classifiers through which novel class can be detected. Classifier ensemble consists of a set of individual classifiers a fusion/selection method to combine/select individual classifier outputs to give a final decision.

For each sample, identify a single classifier which is most likely to produce the correct classification label . As a result, only the output of the selected classifier is taken as a final decision. Usually, the input sample space is partitioned into smaller areas and each classifier learns the sample in each area. It is similar to the "Divide and Conquer" approach.

7. CONCLUSION

In this paper the major issues are investigeted in classifying large volume, high speed and dynamically changing streaming data and proposed a novel approach, of Ensemble Classifier. Comparing with other classifier, The Ensemble Classifier method achieves distinct features as; it detect correct class of attack if detected and it informs user about the attack with alert generation feature of system.

8. REFERENCES

[1] Hongbo Zhu, Yaqiang Wang, Zhonghua Yu "Clustering of Evolving Data Stream with Multiple Adaptive Sliding Window" 2010 International Conference on Data Storage and Data Engineering.

[2] Peng Zhang†, Xingquan Zhu‡, Jianlong Tan†, Li Guo† "Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams" 2010 IEEE International Conference on Data Mining

[3] T.Jyothirmayi, Suresh Reddy "An Algorithm for Better Decision Tree" T.Jyothirmayi et. al. / (IJCSE) International Journal on Computer Science and Engineering 2010

[4] Yongjin Liu, NaLi, Leina Shi, Fangping Li "An Intrusion Detection Method Based on Decision Tree" 2010 International Conference on E-Health Networking, Digital Ecosystems and Technologies

[5] LID Li-xiong, KANGJing, GUO Yun-fei, HUANGHai "A Three-Step Clustering Algorithm over an Evolving Data Stream" The National High Technology Research and Development Program("863" Program) of China, Fund 2008 AAOII002 sponsors.

[6] Mrutyunjaya Panda, Manas Ranjan Patra "A COMPARATIVE STUDY OF DATA MINING ALGORITHMS FOR NETWORK INTRUSION DETECTION" 2008 First International Conference on Emerging Trends in Engineering and Technology

[7] Shuang Wu, Chunyu Yang and Jie Zhou "Clustering-training for Data Stream Mining" Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)

[8] Sang-Hyun Oh, Jin-Suk Kang, Yung-Cheol Byun, Gyung-Leen Park3 and Sang-Yong Byun3 "Intrusion Detection based on Clustering a Data Stream" Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05).

[9] YI-HONG LU^1, YAN HUANG^2 "MINING DATA STREAMS USING CLUSTERING" Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

[10] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby and R. Gavalda, "New ensemble methods for evloving data streams", In KDD'09, ACM, Paris, 2009, pp. 139-148.

[11] G. Cormode, S. Muthukrishnan, and W. Zhuang, "Conquering the divide: Continuous clustering of distribututed data streams", In ICDE, 2007, pp. 1036-1045.