Use of Ensemble & Hybrid Classifiers for Intrusion Detection Systems

[1]Preeti Narooka, [2]Randeep Kaur Kahlon , [3]Shaveta Malik, [4]Sonali Dhamele, [5]Dnyaneshwar Bavkar

[1]Associate Professor, [2,3,4,5]Assistant Professor
Computer Engineering Department,
Terna Engineering College, Mumbai, India

***Abstract*: There is an increase in frequency of malicious network activities and network policy violations. To combat the unauthorized use of a network's resources, intrusion detection systems (IDSs) have emerged. A wide variety of machine learning methods which can be integrated into an IDS have been produced by recent advances in information technology. An overview of intrusion classification algorithms, based on popular methods in the field of machine learning has been presented in this study. Specifically, various ensemble and hybrid techniques were examined, considering both homogeneous and heterogeneous types of ensemble methods. Ensemble methods which are the simplest to implement and generally produce favourable results typically which are based on voting techniques were given special attention. A survey of recent literature shows that hybrid methods, where feature selection or a feature reduction component is combined with a single-stage classifier, have become commonplace. Therefore, the scope of this study has been expanded to encompass hybrid classifiers.**

***Index Terms*: Ensemble classifiers, Hybrid classifiers, Intrusion detection**

I. INTRODUCTION (HEADING 1)

One of the major tasks in machine learning (ML) is constructing a good model from a given data set. Strong classifiers are desirable, but are difficult to find. Training many classifiers at the same time to solve the same problem, and then combining their output to improve accuracy, is known as an ensemble method. When an ensemble, also known as a multi-classifier system, is based on learners of the same type, it is called a homogeneous ensemble. When it is based on learners of different types, it is called a heterogeneous ensemble. Usually, the ensemble's generalization ability is better than a single classifier's, as it can boost weak classifiers to produce better results than a single strong classifier. New door for creating strong classifiers using ensemble methods was opened by two results published in 1990s. The empirical study by L. K. Hansen & P. Salamon found that a combination of multiple classifiers produces more accurate results than the best single one, and the theoretical study by R. E. Schapire showed that weaker classifiers can be boosted to produce stronger classifiers. There are two essential elements involved, in the design of systems that integrate multiple classifiers. First, it is necessary to follow a plan of action to set up an ensemble of classifiers with characteristics that are sufficiently diverse. Second, there is the need for a policy for combining the decisions, or outputs, of particular classifiers in a manner that strengthens accurate decisions and weakens erroneous classifications. Some of the most-used methods, regarding the first element, namely: bagging and its variations, boosting and its generalized version AdaBoost, stacking, and, finally, a mixture of competing experts are covered in section 2. In section 3, strategies for achieving the second element are described. In section 4, an exploration of results obtained from machine learning techniques that use different ensemble approaches is given. Finally, concluding remarks and a critical analysis are expressed in section 5.

II. METHODS OF CREATING ENSEMBLE CLASSIFIERS

An abundance of ensemble-based classifiers has been produced and improved in recent years. Nonetheless, a number of these classifiers are variations on just a few well-established algorithms with capabilities that have been comprehensively validated and broadly published. An overview of the most commonly used ensemble algorithms is presented in this section.

Bagging:-

Breiman's bootstrap aggregating method, or "bagging" for short, was one of the first ensemble-based algorithms, and it is one of the most natural and straightforward ways of achieving a high efficiency. Variety of results are produced in bagging, using bootstrapped copies of the training data; that is, numerous subsets of data are randomly drawn with replacement from the complete training data. A distinct classifier of the same category is modelled, using a subset of the training data. Fusing of particular classifiers is achieved by the use of a majority vote on their selections. Thus, for any example input, the ensemble's decision is the class selected by the greatest number of classifiers. Algorithm 1 contains a pseudocode for the bagging method.

```
Algorithm 1 Bagging
Input: $I$ (a classifier inducer), $T$ (# of iterations), $S$ (Data set for training), $N$ (subset size).
Output: $Ct$; = 1, 2, .., t T
1: $t \leftarrow 1$
2: repeat
3: $S_t \leftarrow$ Subset of $N$ instances taken, with replacement, from $S$.
4: Create Classifier $Ct$ by using $I$ on $St$.
5: $t++$
```

```
6: **until** $t > T$.
```

An approach that is derived from bagging is called the “random forests” classifier. It received its name because it builds a model from number of decision trees. A means of creating this kind of classifier is by training different decision trees, and randomly varying parameters related to training. As in bagging, those parameters can be bootstrapped copies of the training data; however, in contrast with bagging, they also can be particular feature subsets, which is the practice in the random subspace method.
Another approach that is derived from bagging is called “pasting of small votes.” Unlike bagging, pasting small votes was an approach devised to operate on large data sets. Data sets of a large size are partitioned into subsets of a smaller size, which are called “bites,” and those bites are used to train different classifiers. Pasting small votes has led to the creation of two variations: the first one, known as Rvotes, generates the data subsets at random; the other, called Ivotes, builds successive data sets, considering the relevance of the instances. Of the two, Ivotes has been shown to yield better outcomes, similar to the idea present in the boosting-based methods, by which each classifier directs the most relevant instances for the ensemble part that is in use.

Boosting:-
It was shown by Schapire, in 1990, that a weak learner, namely an algorithm that produces classifiers that can slightly out-perform random guessing, can be transformed into a strong learner, namely an algorithm that constructs classifiers capable of correctly classifying all the instances except for an arbitrarily small fraction. Boosting generates an ensemble of classifiers, as does bagging, by carrying out re-sampling of the data and combining decisions using a majority vote. However, that is the extent of the similarities with bagging. Re-sampling in boosting is carefully devised to supply consecutive classifiers with the most informative training data. Essentially, boosting generates three classifiers as follows: A random subset of the available training data is used for constructing the first classifier. The most informative subset given for the first classifier is used for training the second classifier, where the most informative subset consists of training data instances, such that half of them were correctly classified by the first classifier and the other half were misclassified. Finally, training data for the third classifier is made of instances on which the first and second classifiers were in disagreement. A three-way majority vote is then used, to combine the decisions of the three classifiers.
In 1997, Freund and Schapire presented a generalized version of the original boosting algorithm called “adaptive boosting” or “AdaBoost” for short. The method received that name from to its ability to adapt to errors related to weak hypotheses, which are obtained from WeakLearn. AdaBoost.M1 and AdaBoost.R are two of the most frequently used variations of this category of algorithms, because they are suitable for dealing with multi-class and regression problems, respectively. AdaBoost produces a set of hypotheses, and then uses weighted majority voting of the classes determined by the hypotheses in order to combine decisions. A weak classifier is trained to generate the hypotheses, by drawing instances from a successively refreshed distribution of the training data. The updating of the distribution guarantees that it will be more likely to include in the data set for training the subsequent classifier examples that were wrongly classified by the preceding classifier. Thus, the training data of successive classifiers tend to advance towards increasingly hard-to-classify instances. Pseudocodes for AdaBoost, and the similar AdaBoost.M1, are shown in algorithm 2 and algorithm 3, respectively.

```
**Algorithm 2** AdaBoost
**Input:** I (a weak classifier inducer), T (# of iterations), S (Data set for training), N (subset size).
**Output:** $Ct, \alpha t$; = 1, 2, .., T
1: $t \leftarrow 1$
2: $D_1(i) \leftarrow 1/m$ ; $I = 1, 2, .., m$
3: **repeat**
4:     Create Classifier Ct by using I and the distribution Dt.
5:     $\varepsilon_t \leftarrow \sum_{i : C_t(x_i) \neq y_i} D(i)$
6:     **if** $\varepsilon t > 0.5$ **then**
7:         $T \leftarrow t - 1$
8:         exit Loop
9:     **end if**
10:    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$
11:    $D_{t+1}(i) \leftarrow D_t(i) . e^{-\alpha_t y_t C_t(x_i)}$
12:    Normalize so that it becomes a distribution
13:    $t++$
14: **until** $t > T$.
```

```
**Algorithm 3** AdaBoost.M1
**Input:** I (a weak classifier inducer), T (# of iterations), S (Data set for training), N (subset size).
**Output:** $Ct, \beta t$; = 1, 2, .., t T
1: $t \leftarrow 1$
2: $D_1(i) \leftarrow 1/m$ ;  $i = 1, 2, .., i\ m$
3: **repeat**
4:     Create Classifier Ct by using I and the distribution Dt.
5:     $\varepsilon_t \leftarrow \sum_{i\ Ct(xi) \neq yi} D\ t(i)$
```

```
6: **if** $\varepsilon_t > 0.5$ **then**
7: $T \leftarrow t - 1$
8: exit Loop
9: **end if**
10: $\beta_t \leftarrow \frac{\varepsilon_t}{1-\varepsilon_t}$
11: $D_{t+1}(i) \leftarrow D_t(i) \cdot \begin{cases} \beta_t & C_t(i) = y \\ 1 & otherwise \end{cases}$
12: Normalize so that it becomes a distribution
13: $t++$
14: **until** $t > T$.
```

Stacking

Some instances are very likely to be misclassified, because it can happen that they are in the close neighbourhood of the decision boundary, and, therefore, usually are placed on the wrong side of the boundary determined by the classifier. On the other hand, there can be instances that are likely to be classified well, as a result of being on the correct side and far away from the corresponding decision boundaries. The idea behind Wolpert's stacking generalization is that the outputs of an ensemble of classifiers serve as the inputs to another, second-level meta-classifier, which has the purpose of learning the mapping that relates the ensemble outputs with the real true classes

Mixtures of competing experts

Mixtures of competing experts is a technique that approaches the problem in a way similar to stacking. In this method, the ensemble is created using a set of classifiers, followed by a second-level classifier, which has the purpose of assigning the weights that a subsequent combiner requires for fusing decisions. An important characteristic is that the combiner is not generally a classifier, but is a plain combination rule, as is, for instance, random selection (from a weight distribution), weighted majority, or winner-takes-all. Although the combiner might not be a classifier, the set of weights that the combiner uses is selected by a second-level classifier, commonly a neural network that is called a gating network. The training method for the gating network is either a standard back-propagation based on gradient descent, or, more frequently, the expectation maximization (EM) algorithm. The mixture of competing experts technique can be, accordingly, categorized as a classifier selection algorithm. Particular classifiers specialize in a region of the feature space, and the purpose of the combination rule is to select the most suitable classifier. Or, alternatively, classifiers can be balanced according to their expertise, with respect to the instance *x*. The weights may be used by the pooling or combining system in various ways: a single classifier may be selected, if it exhibits the highest weight; or a weighted sum of classifier outputs may be computed for each class, and the class with the highest weighted sum may be chosen. That last strategy is applicable, if the classifier outputs are continuous-valued for each class.

III. METHODS THAT COMBINE CLASSIFIERS

The practice of combining classifiers is the second fundamental element present in ensemble schemes. This approach uses combination rules that are usually categorized according to the following criteria: (*i*) combination rules that are trainable vs. those that are non-trainable; or, alternatively, (*ii*) class labels vs. class-specific applicable combination rules. An independent algorithm establishes the parameters required by the combiner, which are commonly called "weights," in the case of trainable combination rules. An example of this category of methods is the EM algorithm used in the mixture of competing experts model. In the trainable combination rules, the parameters are generally instance-specific, and, for this reason, are known as dynamic combination rules. In contrast, in the case of non-trainable combination rules, the training is not independent; instead, it is incorporated to the training of the ensembles. Weighted majority voting falls into this category of non-trainable rules, as discussed below, given that the weights are directly obtained when the classifiers are created. According to the other taxonomy, class labels having applicable rules that solely require the classification decision (i.e., one of are opposed to those having inputs consisting of continuous-valued outputs produced by particular classifiers. Generally, what these values represent is to what extent the classifiers support each class, and, consequently, they can be used to estimate class-conditional posterior probabilities $P(\omega_j|x)$. Two conditions are required for that last statement: (*i*) the values have to be properly normalized, so that they add up to 1 considering all classes; and (*ii*) the training data used by the classifiers are required to be sufficiently dense.

There exist many models that correspond to this category: MLP and RBF networks are typical examples. Those two models produce continuous-valued outputs that are commonly used as posterior probabilities, although the second required condition concerning sufficiently dense training data often is not met. This paper focuses on the second taxonomy: first, combination rules applicable to class labels are analyzed, and subsequently the methods that fuse class-specific continuous outputs are considered.

Methods that combine class labels

For the ideas presented in this section, the assumption is made that the classifier outputs consist of only the class labels. The decision that is produced by the tth classifier is designated as $d_{t,j} \varepsilon (0,1)$, t =1 ….. T. , where j = 1, … C, T is the number of classifiers and *C* is the number of classes. The combined decision will produce $d_{t,j} = 1$, if the classifier decides for class ω_j, and $d_{t,j} = 0$ otherwise.

Variants on majority voting

Majority voting ensemble methods can be categorized into three versions, with different strategies of choosing the class. In the different strategies, the decisions are taken as follows: (*i*) the class is assigned with the agreement of all the classifiers, and this approach is known as "unanimous voting;" (*ii*) the decision is made, if the number of classifiers agreeing in one class is at least one more than half of the total number of classifiers, which is commonly known as "simple majority;" and, finally, (*iii*) the class assigned is that which receives the majority of the votes, without the condition that the sum of votes is greater than any percentage of the models, and this way of deciding is known as "plurality voting" or "majority voting" without any other adjective. The output of the ensemble, in the last category of plurality voting, can be outlined with the following proposition: select class w_J, whenever the following is true:

$$\sum_{t=1}^{T} d_{t,J} = \max_{j=1}^{c} \sum_{t=1}^{T} d_{t,j} \quad (1)$$

Weighted majority rule

The plurality voting mechanism can be surpassed, in terms of overall performance, if a strategy is devised with the knowledge that some of the experts are better than others at making decisions. The decisions of those better qualified experts can be taken into account, using a larger weight than the others. "let us designate the decision that is produced by the t^{th} classifier upon class ω_j as $d_{t,j}$, and establish that, if the t^{th} classifier selects ω_j, then , otherwise $d_{t,j} = 0$." Then, accordingly, class ω_J is chosen by this method of weighted majority rule, if the combination of the decisions made by the classifiers satisfies the following

$$\sum_{t=1}^{T} w_t d_{t,J} = \max_{j=1}^{c} \sum_{t=1}^{T} w_t d_{t,j} \quad (2)$$

Other schemes that combine class labels, and that are worth mentioning here, are the behaviour knowledge space (BKS) and Borda count.

Methods that combine continuous outputs

There are also classifiers that provide a continuous output for each class. In those schemes, that output represents how much that class is endorsed by the classifier. That value is, in some cases, taken as a predicted value for the respective class posterior or revised probability. The requirements for accepting this type of continuous output value as an estimate of the posterior probability are that the sum of the values corresponding to all classes, once normalized, must add up to 1; and that the classifier deals with sufficiently dense accessible data for training. The normalization usually selected for this purpose is the softmax function.

IV. OVERVIEW OF ENSEMBLE TECHNIQUES

In the literature, one can see the gradual development and implementation of a wide variety of anomaly detection systems based on various machine learning techniques. Many studies have implemented single-stage learning algorithms, such as artificial neural networks (ANN), genetic algorithms (GA) and support vector machines (SVM). However, systems based on a combination of several methods, such as hybrid or ensemble systems, have been common as well. This section presents an overview of such approaches for intrusion detection systems. The overview is accompanied by an analysis of voting-based ensemble techniques in other fields of research.

Homogeneous ensembles for IDS

In general, homogeneous ensembles can be viewed as a simple and effective way of extending the classification hypotheses of a single classification algorithm by creating several variations of that classifier. Although there are numerous ensemble methods by which this can be achieved, the core principles are the same: the aggregation of several relatively simple decision rules should lead to a more sophisticated and reliable final decision. Usually, the selected classifier is trained with different training subsets, at various stages of ensemble development. As a result, the classifier analyses the problem from different perspectives, and, each time, aggregates the knowledge gained towards the definition of an ensemble classification hypothesis.

In the boosting group of algorithms, Folino, Pizzuti, and Spezzano proposed a method for a distributed intrusion detection that used genetic programming to generate decision-tree classifiers. These classifiers were then combined into an ensemble using AdaBoost.M2, a variant of AdaBoost. The KDD 99 data set was used to evaluate the proposed system. Experimental results showed that the proposed approach was comparable to the top two entries to the KDD Cup 99. Additionally, this technique was shown to be suitable for distributed intrusion detection.

Among other methods that used boosting together with another technique, Bahri, Harbi, and Huu introduced a hybrid approach, based on an ensemble method called Greedy-Boost. In their experiments, they compared the precision and recall of AdaBoost, C4.5, and Greedy-Boost, for classification of the KDD 99 data set. Reported results indicated that Greedy-Boost out-performed the other algorithms in terms of the precision, even for probe, U2R, and R2L attacks. This method was good at detecting rare attacks, and lowered average cost, but was not tested on unseen attacks.

To conclude discussion of homogeneous ensembles, Masarat and Taheri implemented a fuzzy combiner, as a method of obtaining an ensemble decision from multiple decision tree classifiers. The experimental procedure was conducted on the full KDD 99 data set, and a decision tree classifier (J48) was used as a base algorithm. The pre-process involved the roulette wheel algorithm, based on the gain ratios for selecting features, where each decision tree was generated with a unique subset of features. Finally, the decisions of all the trained classifiers were weighted and combined in a fuzzy ensemble classifier. The authors reported the accuracy to be nearly 93%, based on 15 selected features. This method has the advantage of solving the computing time limitation, but cannot be used in real-time.

Heterogeneous ensembles for IDS

The defining characteristic of heterogeneous ensembles is that the final decision is based on the classification rules of diverse base classifiers. The chief obstacle to creating such ensembles is that each expert in the ensemble employs a method to construct its classification hypothesis. To generate heterogeneous ensembles, the output of each base classifier must be interpretable in the same way. There are various strategies for aggregating the classification results into a final decision, and the voting procedure is one of the simplest and easiest methods to implement.

First will be considered a group of heterogeneous methods that used majority voting to combine decisions. An early contribution was presented by Mukkamala, Sung, and Abraham, which combined decisions made by classifiers of five different kinds, namely: SVM, MARS, ANN (RP), ANN (SCG) and ANN (OSS). The data set used in this work was a subset of DARPA 1998 and the ensemble performance showed better accuracy than individual classifiers. Time cost, mainly due to ANNs, was a shortcoming.

More recently, Govindarajan and Chandrasekaran proposed a hybrid ensemble method that combined the decisions of diverse classifiers. They implemented a generalized version of bagging and boosting algorithms. Adaptive re-sampling and combining, also called "Arcing," was used to generate different training sets, for two classifiers: radial basis function (RBF) neural network and SVM. In addition, the authors implemented a best-first search (BFS), for feature selection. The final decision was reached by majority voting. An experimental procedure, conducted on the NSL-KDD data set, demonstrated that a hybrid approach was more effective than a single classifier. The reported classification accuracy of the RBF-SVM ensemble was 85.17%.

Heterogeneous ensembles based on voting applied to other domains

To ensure that all available options have been included, we have included heterogeneous ensemble techniques from other research fields.

Tsoumakas, Katakis, and Vlahavas examined three different approaches for combining the decision rules of heterogeneous classifiers. A set of 10 base classifiers was deployed: decision tables (DTab), JRip, PART, J48, IBK, K*, NB, SMO, RBF, and MLP. The performance of each base classifier was evaluated with cross-validation, and weights were derived based on classification accuracy. Base classifiers were evaluated with paired *t*-tests, where each classifier was rated by comparing its performance against other classifiers in the ensemble. The significance score was computed based on paired *t*-test results. Three strategies for base classifier selection were suggested: (*i*) one or more classifiers with the highest significance score were used in making the final decision, and, if there was more than one classifier, then weighted majority voting was used to combine them; (*ii*) several classifiers with similar significance scores were selected and combined with weighted majority voting; and (*iii*) three classifiers with the highest significance score were used to create a majority voting ensemble. The authors, however, did not obtain the expected results, for experiments conducted on 40 data sets selected from the UCI repository. They established that under-performance of the proposed selection methods, due to an inability to adequately secure the efficiency of base classifiers with cross-validation, was the cause.

Kausar, Ishtiaq, Jaffar, and Mirza presented a weighted majority voting ensemble of binary classifiers, based on PSO-generated weights. The developed ensemble was created with four base classifiers: linear discriminant classifier (LDC), quadratic discriminant classifier (QDC), *k*NN, and back-propagation neural network (BP), the outputs of which were defined in a binary domain (0 or 1). PSO was used to generate weights, and the final decision was reached with weighted majority voting. A meta-heuristic approach was, therefore, used to find a near-optimal set of weights for which the classification error of the ensemble was minimized. The performance of the defined method was examined, with respect to four UCI repository data sets: Heart, Diabetes, Iris, and Transfusion.

V. CONCLUSION

Although there are many approaches to knowledge extraction, multiple-expert systems remain one of the most active research areas. Pattern classification problems are often solved through the implementation of ensemble-based techniques. An overview of related studies suggested that many such approaches have been successfully employed, in various fields of research. In general, there are many approaches to deploying multiple classifiers.

This overview has highlighted two main categories of multiple-classifier systems:

- Homogeneous ensembles, or systems based on a single classification approach
- Heterogeneous ensembles, or systems based on two or more different classification approaches

Utilization of homogeneous ensembles in IDS construction has been a fruitful ground for research in the past several years. However, parallel analysis of related studies for both approaches, presented in sections 4.1 and 4.2, reveal that the implementation of heterogeneous ensembles in IDSs is somewhat less complete.

REFERENCES

[1] L.K. Hansen, P. Salamon, Neutral network ensembles, IEEE transactions on pattern analysis and machine intelligence 12 (1990) 993 – 1001.
[2] R. E. Schapire, The strength of weak learnability, Machine learning 5 (2) (1990) 197–227.
[3] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.
[4] S. Freund, A desicion-theoretic generalization of on-line learning and an application to boosting, In Proceedings Euro COLT 94.
[5] N. V. Chawla, L. O. Hall, K. W. Bowyer, T. Moore Jr, W. P. Kegelmeyer, Distributed pasting of small votes, in: International Workshop on Multiple Classifier Systems, Springer, 2002, pp. 52–61.
[6] M. Van Erp, L. Schomaker, Variants of the borda count method for combining ranked classifier hypotheses, in: IN THE SEVENTH INTERNATIONAL WORKSHOP ON FRONTIERS IN HANDWRITING RECOGNITION. 2000. AMSTERDAM

LEARNING METHODOLOGY INSPIRED BY HUMAN'S INTELLIGENCE BO ZHANG, DAYONG DING, AND LING ZHANG, Citeseer, 2000, pp. 443–452

[7] D. H. Wolpert, Stacked generalization, Neural networks 5 (2) (1992) 241–259.

[8] G. Folino, C. Pizzuti, G. Spezzano, An ensemble-based evolutionary framework for coping with distributed intrusion detection, Genetic Programming and Evolvable Machines 11 (2) (2010) 131–146.

[9] G. Folino, C. Pizzuti, G. Spezzano, An ensemble-based evolutionary framework for coping with distributed intrusion detection, Genetic Programming and Evolvable Machines 11 (2) (2010) 131–146

[10] S. Mukkamala, A. H. Sung, A. Abraham, Intrusion detection using an ensemble of intelligent paradigms, Journal of network and computer applications 28 (2) (2005) 167–182.

[11] M. Govindarajan, R. Chandrasekaran, Intrusion detection using an ensemble of classification methods, in: World Congress on Engineering and Computer Science, Vol. 1, 2012, pp. 1–6.

[12] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining multi-label data, in: Data mining and knowledge discovery handbook, Springer, 2009, pp. 667–685.

[13] A. Kausar, M. Ishtiaq, M. A. Jaffar, A. M. Mirza, Optimization of ensemble based decision using pso, in: Proceedings of the World Congress on Engineering, WCE, Vol. 10, 2010, pp. 1–6.