Volume 2, Issue 10, October 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Boosting Techniques on Rarity Mining

B. Sateesh Kumar

Assistant Professor, CSE , JNTUHCEJ, JNT University

Hyderabad, INDIA

ABSTRACT: ***Many real world data mining applications involve classification of rare cases from imbalanced data sets. It is a common problem in many domains such as detecting oil spills from satellite images, predicting telecommunication equipment failures and finding associations between infrequently purchased supermarket items . Rare cases warrant special attention because they pose significant problems for data mining algorithms. Classifying data using Boosting algorithm performs supervised learning which is known as machine learning meta-algorithm. Boosting methods are commonly used to detect objects or persons in videoconference, security system, etc. This paper gives an overview of boosting based algorithms used for classification namely LPBoost, TotalBoost, BrownBoost, GentleBoost, LogitBoost, MadaBoost, RankBoost.***

Keywords: *AUC, Bootstrapping, Bagging, cost-sensitive learning, Precision, Rare cases, small disjuncts.*

I. INTRODUCTION

Rare events are events that occur very infrequently, i.e. whose frequency ranges from say 5% to less than 0.1%, depending on the application. Classification of rare events is a common problem in many domains, such as detecting fraudulent transactions, network intrusion detection, Web mining, direct marketing, and medical diagnostics. For example, in the network intrusion detection domain, the number of intrusions on the network is typically a very small fraction of the total network traffic. In medical databases, when classifying the pixels in mammogram images as cancerous or not [1], abnormal (cancerous) pixels represent only a very small fraction of the entire image. The nature of the application requires a fairly high detection rate of the minority class and allows for a small error rate in the majority class since the cost of misclassifying a cancerous patient as non-cancerous can be very high. Rare cases warrant special attention because they pose significant problems for data mining algorithms.

A. Rare Case

Informally, a *case* corresponds to a region in the instance space that is meaningful with respect to the domain under study and a *rare case* is a case that covers a small region of the instance space and covers relatively few training examples. As a concrete example, with respect to the class *bird, non-flying bird* is a rare case since very few birds (e.g., ostriches) do not fly. Figure 1 shows rare cases and common cases for unlabeled data (Figure 1a) and for labeled data (Figure 1b). In each situation the regions associated with each case are outlined. Unfortunately, except for artificial domains, the borders for rare and common cases are not known and can only be approximated.

One important data mining task associated with unsupervised learning is *clustering*, which involves the grouping of entities into categories. Based on the data in Figure 1a, a clustering algorithm might identify four clusters. In this situation we could say that the algorithm has identified one common case and three rare cases. The three rare cases will be more difficult to detect and generalize from because they contain fewer data points. A second important unsupervised learning task is *association rule mining*, which looks for associations between items (Agarwal, Imielinski & Swami, 1993).

Figure 1b shows a *classification problem* with two classes: a positive class *P* and a negative class *N*. The positive class contains one common case, *P1*, and two rare cases, *P2* and *P3*. For classification tasks the rare cases may manifest themselves as *small disjuncts*. Small disjuncts are those disjuncts in the *learned* classifier that cover few training examples (Holte, Acker & Porter, 1989). If a decision tree learner were to form a leaf node to cover case *P2*, the disjunct (i.e., leaf node) will be a small disjunct because it covers only two training examples. Because rare cases are not easily identified, most research focuses on their learned counterparts—small disjuncts.

Existing research indicates that rare cases and small disjuncts pose difficulties for data mining. Experiments using artificial domains show that rare cases have a much higher misclassification rate than common cases (Weiss, 1995; Japkowicz, 2001), a problem we refer to as the problem with rare cases. A large number of studies demonstrate a similar problem with small disjuncts. These studies show that small disjuncts consistently have a much higher error rate than large disjuncts (Ali & Pazzani, 1995; Weiss, 1995; Holte, et al., 1989; Ting, 1994; Weiss & Hirsh, 2000). Most of these studies also show that small disjuncts collectively cover a substantial fraction of all examples and cannot simply be eliminated doing so will substantially degrade the performance of a classifier. The most thorough empirical study of small disjuncts showed that, in the classifiers induced from thirty real-world data sets, most errors are contributed by the smaller disjuncts (Weiss & Hirsh, 2000).

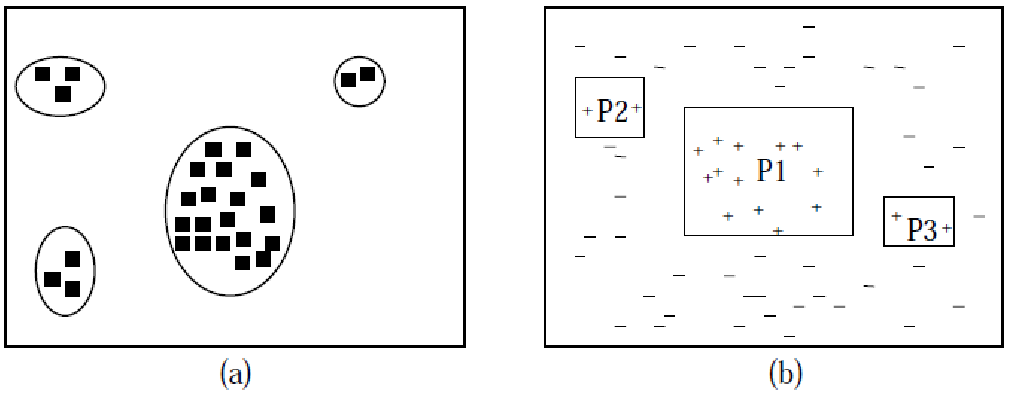

Figure 1: Rare and common cases in unlabeled (a) and labeled (b) data

One important question to consider is whether the rarity of a case should be determined with respect to some absolute threshold number of training examples ("absolute rarity") or with respect to the relative frequency of occurrence in the underlying distribution of data ("relative rarity"). If absolute rarity is used, then if a rare case covers only three examples from a training set, then it should be considered rare. However, if additional training data are obtained so that the training set increases by factor of 100, so that this case now covers 300 examples, then absolute rarity says this case is no longer a rare case. However, if the case covers only 1% of the training data in both situations, then relative rarity would say it is rare in both situations. From a practical perspective, both forms of rarity pose problems for virtually all data mining systems.

II. EVALUATION METRICS TO ADDRESS RARITY

Evaluation metrics that take rarity into account can improve data mining by better guiding the search process and better evaluating the end-result of data mining. *Accuracy* places more weight on the common classes than on rare classes, which makes it difficult for a classifier to perform well on the rare classes. Because of this, additional metrics are coming into wide-spread use. Perhaps the most common is *ROC analysis* and the associated use of the *area under the ROC curve* (AUC) to assess overall classification performance [4; 7]. AUC does not place more emphasis on one class over the other, so it is not biased against the minority class. ROC curves, like *precision-recall curves*, can also be used to assess different tradeoffs—the number of positive examples correctly classified can be increased at the expense of introducing additional false positives. ROC analysis has been used by many systems designed to deal with rarity, such as the Shrink data mining system [6]. *Precision* and *recall* are metrics from the information retrieval community that are useful for data mining. The *precision* of a classification rule, or set of rules, is the percentage of times the predictions associated with the rule(s) are correct. If these rules predict class *X* then *recall* is the percentage of all examples belonging to *X* that are covered by these rule(s).

The problem with using accuracy to label rules that cover few examples is that it produces very unreliable estimates—and is not even defined if no examples in the training set are covered. Several metrics have therefore been designed to provide better estimates of accuracy for the classification rules associated with rare cases/small disjuncts. One such metric is the *Laplace estimate*. The standard version of this metric is defined as $(p+1)/(p+n+2)$, where p positive and n negative examples are covered by the classification rule. This estimate moves the accuracy estimate toward ½ but becomes less important as the number of examples increases. A more sophisticated error-estimation metric for handling rare cases and small disjuncts was proposed by *Quinlan* [9]. This method improves the accuracy estimates of the small disjuncts by taking the *class distribution* (class priors) into account.

Metrics that support cost-sensitive learning are the subject of much research. *Cost-sensitive learning methods* [38] can exploit the fact that the value of correctly identifying the positive (rare) class outweighs the value of correctly identifying the common class. For two-class problems this is done by associating a greater cost with false negatives than with false positives. This strategy is appropriate for most medical diagnosis tasks because a false positive typically leads to more comprehensive (i.e., expensive) testing procedures that will ultimately discover the error, whereas a false negative may cause a life-threatening condition to go undiagnosed, which could lead to death. Assigning a greater cost to false negatives than to false positives will improve performance with respect to the positive (rare) class. If this misclassification cost ratio is 3:1, then a region that has ten negative examples and four positive examples will nonetheless be labeled with the positive class. Thus non-uniform costs can bias the classifier to perform well on the positive class—where in this case the bias is desirable. One problem with this approach is that specific cost information is rarely available. This is partially due to the fact that these costs often depend on multiple considerations that are not easily compared [10]. Most modern data mining systems can handle cost-sensitivity directly, in which case cost information can be passed to the datamining algorithm. In the past such systems often did not have this capability. In this case cost-sensitivity was obtained by altering the ratio of positive to negative examples in the training data, or, equivalently, by adjusting the probability thresholds used to assign class labels [16].

III. BOOSTING TECHNIQUES

A. History – Method Backgrounds

Several methods of estimation have preceded boosting approach. Common feature for all methods is that they work out by extracting samples of a set, calculating the estimate for each drawn sample group repeatedly and combining the calculated results into unique one. One of the ways, the simplest one, to manage estimation is to examine the statistics of selected available samples from the set and combine the results of calculation together by averaging them. Such approach is a *jack-knife estimation*, when one sample is left out from the whole set each time to make an estimation [12]. Obtained collection of estimates is averaged afterwards to give the final result. Another, improved method, is B*ootstrapping*. Bootstrapping repeatedly draws certain number of samples from the set and processes calculated estimations by averaging, similar to jack-knife [12]. *Bagging* is the further step towards boosting. It consists of Bootstrap aggregation which increases classifier stability and reduces variance over a collection of samples. In this, samples are drawn with replacement and each draw has a classifier C_i attached to it, so that final classifier becomes a weighted vote of C_i - s. Bootstrapping and Bagging techniques are *non-adaptive Boosting* techniques.

The Boosting procedure is similar to Bootstrap and Bagging. The first was proposed in 1989 by Schapire and is as follows.

Boosting Algorithm for Classification

Input: $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (x_i, y_i)$ as training set.

Output: H(x), a classifier suited for the training set.

1. Randomly select, *without* replacement, $L_1 < N$ samples from Z to obtain Z_1; train weak learner H_1 on it.

2. Select $L_2 < N$ samples from Z with half of the samples misclassified by H_1 to obtain Z_2; train weak learner H_2 on it.

3. Select all samples from Z that H_1 and H_2 disagree on; train weak learner H_3.

4. Produce final classifier as a vote of weak learners $H(x) = \text{sign}\left(\sum_{n=1}^{3} Hn(x) \right)$

Essential Boosting idea is combining together basic rules, creating an ensemble of rules with better overall performance than the individual performances of the ensemble components. Each rule can be treated as a hypothesis, a classifier. Moreover, each rule is weighted so that it is appreciated according to its performance and accuracy. Weighting coefficients are obtained during the boosting procedure which, therefore, involves learning.

Mathematical roots of Boosting originate from probably approximately correct learning (PAC learning) [21, 23]. Boosting concept was applied for real task of optical character recognition using neural networks as base learners [25] . Recent practical implementation focuses on diverse fields, giving answers to questions such as tumour classification [4] or assessment whether household appliances consume energy or not [25].

1) Connection Between Bootstrap, Bagging And Boosting

Figure 2 shows the connection between bootstrap, bagging and boosting. This diagram emphasizes the fact that these three techniques are built on random sampling, being that bootstrapping and bagging perform sampling with replacement while boosting does not. The bagging and boosting techniques have in common the fact that they both use a majority vote in order to perform the final decision.

B. METHODS

Boosting method uses series of training data, with weights assigned to each training set. Series of classifiers are defined so that each of them is tested sequentially comparing the result of the previous classifier and using the results of previous classification to concentrate more on misclassified data. All the classifiers used are voted according to accuracy. Final classifier, combines weight of the votes of each classifier from the test sequence[22].

Two important ideas have contributed development of Boosting algorithms' robustness. First tries to find the best possible way to modify the algorithm so that its weak classifier produces more useful and more effective prediction results. Second tries to improve the design of a weak classifier. Answers to both concepts result in a large family of boosting methods[26]. Relations between two concepts of optimization and Boosting procedures have been a basis for establishing new types of Boosting algorithms.

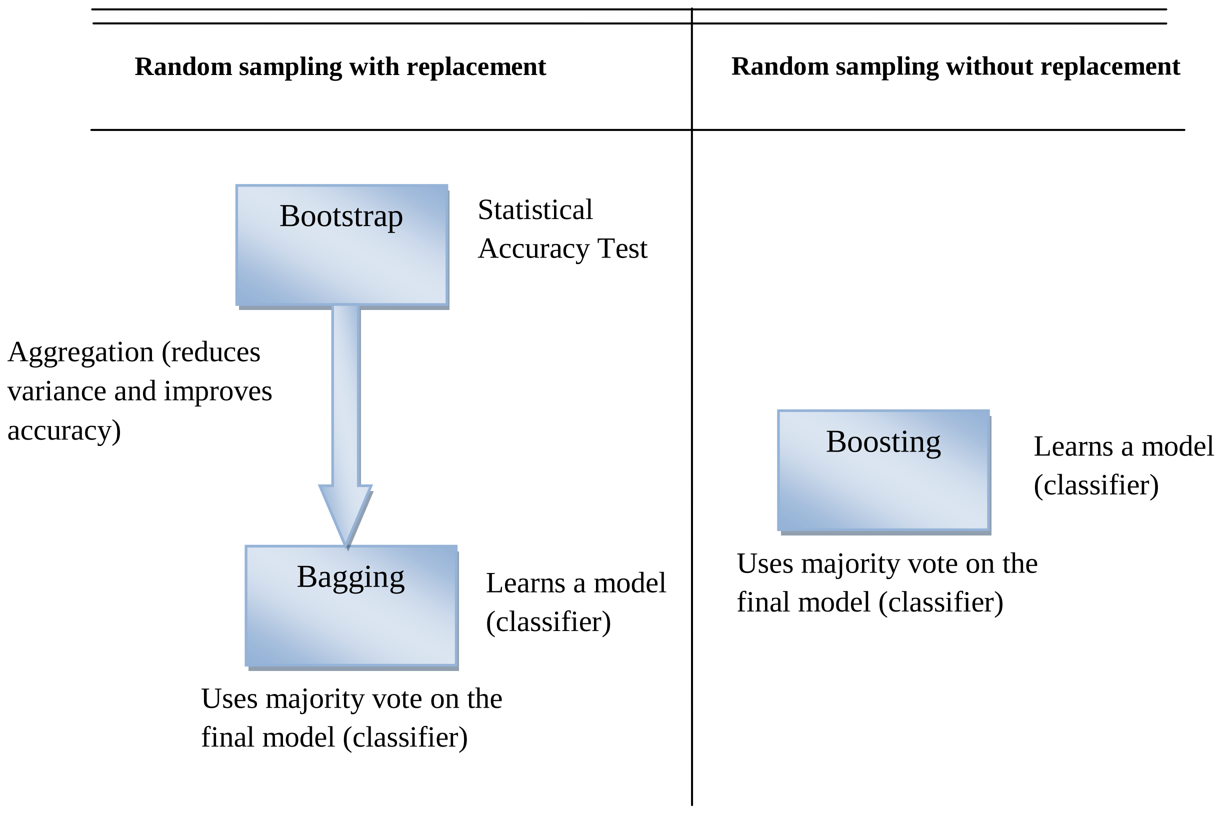
Random sampling with replacement
Random sampling without replacement
Bootstrap
Statistical Accuracy Test
Aggregation (reduces variance and improves accuracy)
Bagging
Learns a model (classifier)
Uses majority vote on the final model (classifier)
Boosting
Learns a model (classifier)
Uses majority vote on the final model (classifier)

Figure 2: Connection between bootstrap, bagging and boosting

1) Basic Methods

a) Discrete AdaBoost

The same researchers that proposed the Boosting algorithm, Freund and Schapire, also proposed in 1996, the Discrete AdaBoost (Adaptive Boosting) algorithm [5]. The idea behind adaptive boosting is to weight the data instead of (randomly) sampling it and discarding it. The AdaBoost algorithm [5, 14] is a well-known method to build ensembles of classifiers with very good performance [14]. It has been shown empirically that AdaBoost with decision trees has excellent performance [2], being considered the best off-the-shelf classification algorithm [14]. This algorithm takes training data and defines weak classifier functions for each sample of training data. Classifier function takes the sample as argument and produces value -1 or 1 in case of a binary classification task and a constant value - weight factor for each classifier. Procedure trains the classifiers by giving higher weights to those training sets that were misclassified. Every classification stage contributes with its weight coefficients, making a collection of stage classifiers whose linear combination defines the final classifier [20]. Each training pattern receives a weight that determines its probability of being selected as a training set for an individual component. Inaccurately classified patterns are likely to be used again. The idea of accumulating weak classifiers means adding them so that each time the adding is done, they get multiplied with new weighting factors, according to distribution and relating to the accuracy of classification. *Discrete AdaBoost* or just *AdaBoost* was the first one that could change weak learners [20]. Generally, AdaBoost has shown good performance at classification. Bad feature of Adaptive Boosting is its sensitivity to *noisy data* and *outliers*. Boosting has a feature of reducing variance and bias, and a major cause of boosting success is variance reduction.

The function I(c) used in steps 2.b and 2.d is an indicator function I(c) = 1, if c =“true” and I(c) = 0 if c =“false”. The algorithm stops when $m = M$ or if $err_m > 0.5$; this last condition means that it is impossible to build a better ensemble using these weak classifiers, regardless of the increase of their number. This way, M represents the maximum number of classifiers to accommodate in the learning ensemble.

Discrete AdaBoost Algorithm for Classification

Input: $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (x_i, y_i)$ as training set . M, the maximum number of classifiers.

Output: H(x), a classifier suited for the training set.

```
1.Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots, N\}$.

2.For m=1 to M
a) Fit a classifier $H_m(x)$ to the training data using weights $w_i$.
b) Let $err_m = \frac{\sum_{i=1}^{N} w_i I \ (y_i \neq H_m(x_i))_1}{\sum_{i=1}^{N} w_i}$
c) Compute $\alpha_m = 0.5 \log(\frac{1-err_m}{err_m})$

d) Set $w_i \leftarrow w_i \exp(-\alpha_m I(yi \neq H_m(x_i)))$ and renormalize to $\sum_i w_i = 1$.

3. Output $H(x) = \text{sign} \left( \sum_{m=1}^{m} \alpha_m H_m(x) \right)$.
```

TABLE I: COMPARISON OF BOOSTING AND ADABOOST ALGORITHMS

Feature	Boosting	Adaptive Boosting
Data Processing	Random Sampling without replacement	Weighting (No Sampling)
No. of Classifiers	Three	Upto M
Decision	Majority Vote	Weighted Vote

b) RealBoost

The creators of boosting concept have developed a general version of AdaBoost, which changes the way of expressing predictions. Instead of Discrete AdaBoost classifiers producing -1 or 1, a RealBoost classifiers produce real values. The sign of classifier output value defines which class the element belongs to. Those real values produced by classifier will serve as measure of how confident in prediction we are, so that classifiers implemented later can learn from their predecessors. Difference is that with real value, confidence can be measured instead of having just the discrete value that expresses classification result.

Real AdaBoost Algorithm for Classification

Input: $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (x_i, y_i)$ as training set . M, the maximum number of classifiers.

Output: H(x), a classifier suited for the training set.

```
1.Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots, N\}$.

2. For m=1 to M
a) Fit the class probability estimate $p_m(x) = P_w(y = 1|x) \in [0, 1]$, using weights $w_i$ on the training data.
b) Set $H_m = 0.5 \log(\frac{1-p_m(x)}{p_m(x)}) \in R$
c) Set $w_i \leftarrow w_i \exp(-y_i H_m(x_i))$ and renormalize to $\sum_i w_i = 1$.

3. Output $H(x) = \text{sign} \left( \sum_{m=1}^{m} H_m(x) \right)$.
```

Comparing this algorithm with Discrete AdaBoost, we see that the most important differences are in steps 2a) and 2b). On the Real AdaBoost algorithm, these steps consist on the calculation of the probability that a given pattern belongs to a class. The AdaBoost algorithm classifies the input patterns and calculates the weighted amount of error.

2) Weight Function Modification

a) GentleBoost

GentleBoost algorithm represents modified version of the Real AdaBoost algorithm. The Real AdaBoost algorithm performs exact optimization with respect to Hm. The Gentle AdaBoost [11] algorithm improves it, using Newton stepping, providing a more reliable and stable ensemble. Instead of fitting a class probability estimate, the Gentle AdaBoost algorithm uses weighted least-squares regression to minimize the function

$$E[\exp(-yH(x))] \qquad (1)$$

GentleBoost allows to increase performance of classifier and reduce computation by 10 to 50 times compared to Real AdaBoost [19]. This algorithm usually outperforms Real AdaBoost and LogitBoost at stability.

Gentle AdaBoost Algorithm for Classification

Input: $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (x_i, y_i)$ as training set . M, the maximum number of classifiers.

Output: H(x), a classifier suited for the training set.

```
1.Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots ,N\}$.
2. For m=1 to M
a) Train $H_m(x)$ by weighted least-squares of $y_i$ to $x_i$, with weights $w_i$.
b) Update $H(x) = H(x) + H_m(x)$.
c) Update $w_i \leftarrow w_i \exp(-y_i H_m(x_i))$ and renormalize to $\sum_i w_i = 1$.
3. Output $H(x) = \text{sign}\ ( \sum_{m=1}^{m} H_m(x) )$.
```

b) MadaBoost

Domingo and Wanatabe propose a new algorithm, MadaBoost, which is a modification of AdaBoost [10]. Indeed, AdaBoost introduces two main disadvantages. First, this algorithm cannot be used by filtering framework [16]. Filtering framework allows to remove several parameters in boosting methods [6]. Second, AdaBoost is very sensitive to noise [16]. MadaBoost resolves the first problem by limiting the weight of samples with their initial probability. Moreover, filtering framework allows to resolve the problem of noise sensitivity [10]. With AdaBoost, weight of misclassified samples increases until samples are correctly classified [14]. Weighting system in MadaBoost is different. Indeed, variance of sample weights is moderate [10]. MadaBoost is resistant to noise and can progress in noisy environment [10].

3) Adaptive "Boost By Majority"

a) BrownBoost

AdaBoost is a very popular method. However, several experimentations have shown that AdaBoost algorithm is sensitive to noise during the training [8]. To fix this problem, Freund introduced a new algorithm named BrownBoost [16] which makes changing of the weights smooth and still retains PAC learning principles. BrownBoost refers to Brownian motion which is a mathematical model to describe random motions [2]. The method is based on boost by majority, combining many weak learners simultaneously, hence improving the performance of simple boosting [15] [13]. Basically, AdaBoost algorithm focuses on training samples that are misclassified [18]. Hence, the weight given to the outliers is larger than the weight of the good training samples. Unlike AdaBoost, Brown-Boost allows to ignore training samples which are frequently misclassified [16]. Thus, this classifier created is trained with non-noisy training dataset [16]. BrownBoost is more performant than AdaBoost on noisy training dataset. Moreover, more training dataset becomes noisy, more BrownBoost classifier created becomes accurate compared to AdaBoost classifier.

4) Statistical Interpretation Of Adaptive Boosting

a) LogitBoost

LogitBoost is a boosting algorithm formulated by Jerome Friedman, Trevor Hastie, and Robert Tibshirani [19]. It introduces a statistical interpretation to AdaBoost algorithm by using additive logistic regression model for determining classifier in each round. Logistic regression is a way of describing the relationship between one or more factors, in this case instances from samples of training data, and an outcome, expressed as a probability. In case of two classes, outcome can take values 0 or 1. Probability of an outcome being 1 is expressed with logistic function. The LogitBoost algorithm uses Newton steps for fitting an additive symmetric logistic model by maximum likelihood [19]. Every factor has a coeficient attached, expressing its share in output probability, so that each instance is evaluated on its share in classification. LogitBoost is a method to minimize the logistic loss, AdaBoost technique driven by probabilities optimization. This method requires care to avoid numerical problems. When weight values become very small, which happens in case probabilities of outcome become close to 0 or 1, computation of the working response can become inconvenient and lead to large values. In such situations, approximations and threshold of response and weights are applied.

LogitBoost Algorithm for Classification

Input: $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (x_i, y_i)$ as training set . M, the maximum number of classifiers.

Output: H(x), a classifier suited for the training set.

```
1.Initialize the weights $w_i = 1/N$, $i \in \{1, \ldots, N\}$.

2. For m=1 to M and while $H_m \neq 0$

    a) Compute the working response $z_i = y_i - p(x_i) / p(x_i)(1 - p(x_i))$ and weights
       $w_i = p(x_i)(1 - p(x_i))$.
    b) Fit $H_m(x)$ by weighted least-squares of $z_i$ to $x_i$, with weights $w_i$.
    c) Set $H(x) = H(x) + 0.5 H_m(x)$ and $p(x) = \frac{\exp(H(x))}{\exp(H(x)) + \exp(-H(x))}$

3. Output $H(x) = \text{sign}\left( \sum_{m=1}^{m} H_m(x) \right)$.
```

5) "Totally-Corrective" Algorithms

a) LPBoost

LPBoost is based on Linear Programming [19]. The approach of this algorithm is different compared to AdaBoost algorithm. LPBoost is a supervised classifier that maximizes margin of training samples between classes. Classification function is a linear combination of weak classifiers, each weighted with value that is adjustable. The optimal set of samples is consisted of a linear combination of weak hypotheses which perform best under worst choice of misclassification costs [3]. At first, LPBoost method was disregarded due to large number of variables, however, efficient methods of solving linear programs were discovered later. Classification function is formed by sequentially adding a weak classifier at every iteration and every time a weak classifier is added, all the weights of the weak classifiers present in linear classification function are adjusted (totally-corrective property). Indeed, in this algorithm, we update the cost function after each iteration [3]. The result of this point of view is that LPBoost converge to a finite number of iterations and need less iterations than AdaBoost to converge [24]. However, computation cost of this method is more expensive than AdaBoost [24].

b) TotalBoost

General idea of Boosting algorithms, maintaining the distribution over a given set of examples, has been optimized. A way to accomplish optimization for TotalBoost is to modify the way measurement of hypothesis' goodness, (edge) is being constrained through iterations. AdaBoost constrains the edge with the respect to the last hypothesis to maximum zero. Upper bound of the edge is chosen more moderately whereas LPBoost, being a totally-corrective algorithm too always chooses the least possible value[33]. An idea that was introduced in works of Kivinen and Warmuth (1999) is to constrain the edges of all past hypotheses to be at most adapted and otherwise minimize the relative entropy to the initial distribution. Such methods are called totally-corrective. TotalBoost method is "totally corrective", constraining the edges of all previous hypotheses to maximal value that is properly adapted. It is proven that, with adaptive edge maximal value, measurement of confidence in prediction for a hypothesis weighting increases[33]. Compared with simple boost algorithm that is totally corrective, LPBoost, TotalBoost regulates entropy and moderately chooses which has led to significantly less number of iterations [33], helpful feature for proving iteration bounds.

6) RankBoost

RankBoost is an efficient boosting algorithm for combining preferences [17] solves the problem of estimating rankings or preferences. It is essentially based on pioneering AdaBoost algorithm introduced in works of Freund and Schapire (1997) and Schapire and Singer (1999). The aim is to approximate a target ranking using already available ones, considering that some of those will be weakly correlated with the target ranking. All rankings are combined into a fairly accurate single ranking, using RankBoost machine learning method. The main product is an ordering list of the available objects using preference lists that are given. Being a Boosting algorithm, defines RankBoost as a method that works in iterations, calls a weak learner that produces ranking each time, and a new distribution that will be passed to the next round. New distribution gives more importance to the pairs that were not ordered appropriately, placing emphasis on following weak learner to order them properly.

IV. CONCLUSIONS

This paper provides a good survey of the literature on mining with rare classes and rare cases using Boosting techniques that shows original approach to classification and its variants. Different evaluation metrics on rarity mining are also discussed in this paper. This article clearly specifies the connection between Bootstrap, Bagging and Boosting techniques. Currently, the milestone method, AdaBoost, has become a very popular algorithm to use in practise. It emerged to have plenty of versions, each giving different contribution to algorithm performance. It has been interpreted as a procedure based on functional gradient descent (AdaBoost), as an approximation of logistic regression (LogitBoost), or enhanced with arithmetical improvements of calculation of weight coefficients (GentleBoost and MadaBoost). It was connected with linear programming (LPBoost), Brownian motion (BrownBoost), entropy based methods for constraining hypothesis goodness (TotalBoost). Finally, boosting was used for such implementations as ranking the features (RankBoost). Boosting methods are used in different applications like Faces detection, Classification of musical genre , Real-Time vehicle tracking, Tumour classification with gene expression data, Film ranking and Meta-search problem.

ACKNOWLEDGEMENT

I would like to express sincere gratitude to my guide **Dr.A.Govardhan** who is a Professor in CSE & Director of Evaluation at JNTU Hyderabad for his valuable comments and suggestions in preparing this article. I thank all the referees authors of this paper who have helped directly or indirectly the possibility to completion of this paper. Finally I thank JNTU Hyderabad, which provided sources to complete this paper.

REFERENCES

[1] N. V. Chawla, K.W. Bowyer, L. O. Hall, W. P. Kegelmeyer, *SMOTE: Synthetic Minority Over-Sampling Technique*, Journal of Artificial Intelligence Research, vol. 16, 321-357, 2002.
[2] E. Bauer and R. Kohavi, *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants*, Machine Learning, 36:105–139, 1999.
[3] A. Bradley, *The use of the area under the ROC curve in the evaluation of machine learning algorithms*, Pattern Recognition, 30(7): 1145-1159, 1997.
[4] Marcel Dettling and Peter Buhlmann, *Finding predictive gene groups from microarray data*, J. Multivar. Anal., 90(1):106-131, 2004.
[5] Y. Freund and R. Schapire, *Experiments with a new boosting algorithm,* In Thirteenth International Conference on Machine Learning, pages 148–156, Bari, Italy, 1996.
[6] M. Kubat, R. C. Holte, and S. Matwin, *Machine learning for the detection of oil spills in satellite radar images*, Machine Learning, 30(2):195-215, 1998.
[7] F. Provost, and T. Fawcett, *Robust classification for imprecise environments,* Machine Learning, 42: 203-231, 2001.

[8] Thomas G. Dietterich, *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. In Bagging, boosting, and randomization*, Machine Learning, pages 139-157, 1998.
[9] J. R. Quinlan, *Improved estimates for the accuracy of small disjuncts*, Machine Learning 6:93-98, 1991.

[10] N. V. Chawla, *C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure,* In Workshop on Learning from Imbalanced Datasets II, International Conference on Machine Learning, 2003.
[11] J. Friedman, T. Hastie, and R. Tibshirani, *Additive logistic regression: a statistical view of boosting,* The Annals of Statistics, 28(2):337–374, April 2000.
[12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification,* Wiley-Interscience Publication, 2000.
[13] Yoav Freund, *Boosting a weak learning algorithm by majority*, Inf. Comput., 121(2):256-285, 1995.
[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2nd edition, 2001.
[15] Yoav Freund, *An adaptive version of the boost by majority algorithm*, Machine Learning, 43(3):293-318, 2001.
[16] C. Elkan, *The foundations of cost-sensitive learning,* In Proceedings of the Seventeenth International Conference on Machine Learning, pages 239-246, 2001.
[17] Yoav Freund, Raj Iyer, Robert E. Schapire, Yoram Singer, and G. Dietterich. *An efficient boosting algorithm for combining preferences,* In Journal of Machine Learning Research, pages 170 -178, 2003.
[18] Yoav Freund and Robert E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting,* Journal of computer and system sciences, 55:119-139, 1996.
[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *Additive logistic regression: a statistical view of boosting,* Annals of Statistics, 28:2000, 1998.
[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, Special invited paper, *Additive logistic regression: A statistical view of boosting,* The Annals of Statistics, 28(2):337-374, 2000.
[21] L. G. Valiant, *A theory of the learnable*, Commun. ACM, 27(11):1134-1142, 1984.
[22] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques,* Morgan Kaufmann, 2000.
[23] Michael Kearns and Leslie Valiant, *Cryptographic limitations on learning boolean formulae and finite automata,* J. ACM, 41(1):67-95, 1994.
[24] Jure Leskovec and John Shawe-Taylor, *Linear programming boosting for uneven datasets,* In ICML, pages 456-463, 2003.
[25] Ron Meir and Gunnar Ratsch, *An introduction to boosting and leveraging,* pages 118-183, 2003.
[26] Robert E. Schapire and Yoram Singer, *Improved boosting algorithms using confidence-rated predictions*, 1999.

AUTHORS BIOGRAPHY

B.Sateesh Kumar , is working as Assistant Professor of Information Technology, JNTUH college of Engineering JAGITYALA, JNT University, Hyderabad. He received B.Tech (CSE) from Kakatiya University, M.Tech(SE) from JNTU Hyderabad and pursuing Ph.D(CSE) from JNTU, Hyderabad. He received “BHARATH JYOTHI AWARD” from New Delhi in December 2011. He organized a five day workshop on “Computer Awareness for Police Department” Under TEQIP in Department of CSE at JNTU College of Engineering, Kakinada. He guided many projects for B.Tech, M.Tech and MCA students. His research interest is in the area of Data Mining. He is one of the coordinators for *UGC sponsored program for SC/ST students* conducted at JNTU Kakinada. He is nominated as MEMBER OF BOARD OF STUDIES in CSE department at P.R.Govt.College, Kakinada. He was the reviewer of International conference on intelligent systems and data processing (ICISD 2011) under Multiconf-2011 organized by Department of IT, GH Patel college of engineering at Vallabh, Vidyanagar. His research articles are accepted in international Conferences and journals.