World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 1, No. 3, 105-109, 2011.

A Hybrid Classifier using Boosting, Clustering, and Naïve Bayesian Classifier

A. J. M. Abu Afza, Dewan Md. Farid, and Chowdhury Mofizur Rahman
Department of Computer Science and Engineering
United International University
Dhaka-1209, Bangladesh
afza22bd@gmail.com, dewanfarid@gmail.com, cmruiu@gmail.com

Abstract—a new classifier based on boosting, clustering, and naïve Bayesian classifier is introduced in this paper, which considers the misclassification error produced by each training example and update the weights of training examples in training dataset associated to the probability of each attribute of that example. The proposed classifier clusters the training examples based on the similarity of attribute values and then generates the probability set for each cluster using naïve Bayesian classifier. Boosting trains a series of classifiers for a number of rounds that emphasis to the misclassification rate in each round. The proposed classifier addresses the problem of classifying the large data set and it has been successfully tested on a number of benchmark problems from the UCI repository, which achieved high classification rate.

Keywords- clustering; naïve Bayesian classifier; boosting; hybrid classifier.

I. INTRODUCTION

Boosting is the process of combining many classifiers to generate a single strong classifier with very low error. The concept of adaptive boosting, which is known as AdaBoost boosting algorithm was first introduced by Freund and Schapire in 1997 [1] that classify an example by voting the weighted predictions of a set of base classifiers, which are generated in a series of rounds. The major drawback of boosting is overfitting; that is, with many rounds of boosting, the test error increase as the final classifier becomes overly complex [2]-[4]. The test error of AdaBoost algorithm often tends to decrease well after the training error is zero, and does not increase even after a very large number of rounds. However, the test error has been observed to increase slightly after an extremely large number of rounds [5]. Boosting have become one of the alternative framework for classifier design, together with the more established classifiers, like Bayesian classifier, decision tree, neural network, and support vector machine [6]-[8].

The naïve Bayesian (NB) classifier is an efficient technique for performing classification in data mining, which is widely used in many real world applications, like stock prediction, medical diagnosis, weather forecasting etc [9],[10]. Using Bayesian principal, it provides an optimal way to predict the class value of an unknown example whose attribute values are known. In naïve Bayesian classifier the prior and conditional probabilities are calculated from the training dataset and then it classifies the test dataset with the posterior probabilities. If there were sufficient randomly sampled examples of every possible combination of attribute values, such estimation would be straightforward, acceptably reliable and accurate. However, most combinations are not represented in the training data. Hence, conditional probabilities cannot be calculated accurately while applying Bayesian learning. Naïve Bayesian learning circumvents this problem by assuming that all attributes are independent. This assumption makes naïve Bayesian algorithm simple and time efficient, gives it a linear time complexity dependent only of training data. However, when the attribute independence assumption of NB is violated, which appears to be very common and realistic, the performance of naïve Bayesian classifier becomes very poor and classification accuracy degrades. A number of approaches have been sought to alleviate the independence assumption problem of naïve Bayesian classifier. It includes NBTree [11], LBR [12], [13], RBC [14], BSEJ [15] etc.

In this paper, we present a new hybrid classifier using boosting, clustering, and naïve Bayesian classifier, which first initializes the weight of training examples to $1/n$, where n is the total number of examples in training data set. After that it creates a new data set from training data set using selection with replacement technique and then clustering the new data set into k clusters. Then calculates the prior and conditional probabilities for each cluster, and classify the training examples with each cluster's probabilities values. The weights of the training examples updated according to how they were classified. If an example is misclassified then its weight is increased, or if an example is correctly classified then its weight is decreased. Then it creates another new data set with the misclassification error produced by each training example from training dataset, and continues the process until all the training examples are correctly classified. To classify a new

example use all the prior and conditional probabilities of each cluster in each round and consider the class of new example with highest classifier's vote. The proposed classifier has been successfully tested on a number of benchmark problems from UCI machine learning repository, which achieved high classification rates with very low error rate.

The remainder of the paper is organized as follows. Section II describes the basic concept of boosting, clustering, and naïve Bayesian classifier. Section III presents the proposed hybrid classifier using boosting, clustering, and naïve Bayesian classifier. Section IV describes the experimental results based on a number of widely used benchmark datasets from UCI machine learning repository. Finally, Section V presents our conclusions with future works.

II. Classifier Construction Techniques

A. Boosting

Boosting is an iterative procedure used to adaptively change the distribution of training examples so that the base classifiers will focus on examples that are hard to classify. Boosting assigns a weight to each training example and adaptively changes the weight at the end of each boosting round. A sample is drawn according to the sampling distribution of the training examples to obtain a new training dataset. Next, a classifier is induced from the training dataset and used to classify all the examples in the original dataset. The weights of the training examples are updated at the end of each boosting round. Examples that are incorrectly classified will have their weights increased, while those that are correctly classified will have their weight decreased. This forces the classifier to focus on examples that are difficult to classify in subsequent iterations.

B. Clustering

Clustering analyzes data examples without consulting a known class label. In general, for clustering the class labels are not present in the training dataset. Clustering can be used to generate such class labels. The examples from dataset are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. That is, clusters of examples are formed so that examples within a cluster have high similarity in comparison to one another, but are very dissimilar to examples in other clusters. Each cluster that is formed can be viewed as a class of examples, from which rules can be derived. Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together.

C. Naïve Bayesian Classifier

Naïve Bayesian classifier is a simple classification scheme, which estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label *c*. The conditional independence assumption can be formally stated as follows:

$$P(A \mid C = c) = \prod_{i=1}^{n} P(A_i \mid C = c) \qquad (1)$$

Where each attribute set $A = \{A_1, A_2, \ldots, A_n\}$ consists of n attribute values. With the conditional independence assumption, instead of computing the class-conditional probability for every combination of A, only estimate the conditional probability of each A_i, given C. The latter approach is more practical because it does not require a very large training set to obtain a good estimate of the probability. To classify a test example, the naïve Bayesian classifier computes the posterior probability for each class C.

$$P(C \mid A) = \frac{P(C)\prod_{i=1}^{n} P(A_i \mid C)}{P(A)} \qquad (2)$$

Since $P(A)$ is fixed for every A, it is sufficient to choose the class that maximizes the numerator term,

$$P(C)\prod_{i=1}^{n} P(A_i \mid C) \qquad (3)$$

III. Proposed Algorithm

Given a training data $D = \{t_1, \ldots, t_n\}$ where $t_i = \{t_{i1}, \ldots, t_{ih}\}$ and the training data D contains the following attributes $\{A_1, A_2, \ldots, A_n\}$ and each attribute A_i contains the following attribute values $\{A_{i1}, A_{i2}, \ldots, A_{ih}\}$. Also the training data D contains a set of classes $C = \{C_1, C_2, \ldots, C_m\}$. Each training example has a particular class C_j. The algorithm first initializes the weights of training examples an equal value of $w_i = 1/n$, where n is the total number of the examples in training data D. Then the algorithm generates a new data set D_i with equal number of examples from training data D using selection with replacement technique and cluster the D_i into k number of clusters. For clustering the algorithm arbitrarily choose n examples from D_i as the initial cluster centers and assign each example t_i from D_i to the cluster to which the example is the most similar based on the attribute values, or the mean value of the examples in the cluster. Next the algorithm calculates the prior $P(C_j)$ and conditional $P(A_{ij}|C_j)$ probabilities for each cluster k_i.

The prior probability $P(C_j)$ for each class is estimated by counting how often each class occurs in the cluster. For each attribute A_i the number of occurrences of each attribute value A_{ij} can be counted to determine $P(A_i)$. Similarly, the conditional probability $P(A_{ij}|C_j)$ for each attribute values A_{ij} can be estimated by counting how often each attribute value occurs in the class in the cluster. Then the algorithm classifies all the training examples in training data D with these prior $P(C_j)$ and conditional $P(A_{ij}|C_j)$ probabilities from each cluster. For classifying the examples, the prior and conditional probabilities are used to make the prediction. This is done by combining the effects of the different attribute values from that example. Suppose the example e_i has independent attribute values $\{A_{i1}, A_{i2}, \ldots, A_{ip}\}$, we know $P(A_{ik} \mid C_j)$, for each class C_j and attribute A_{ik}. We then estimate $P(e_i \mid C_j)$ by

$$P(e_i \mid C_j) = P(C_j) \prod_{k=1 \to p} P(A_{ij} \mid C_j) \qquad (4)$$

To classify the example, the probability that e_i is in a class is the product of the conditional probabilities for each attribute value with prior probability for that class. The posterior probability $P(C_j \mid e_i)$ is then found for each class and the example classifies with the highest posterior probability for that example. The algorithm classifies each example $t_i \in D$ with maximum posterior probabilities from each clusters probabilities values using majority voting technique. After that the weights of the training examples t_i in training data D are adjusted/ updated according to how they were classified. If an example was misclassified then its weight is increased, or if an example was correctly classified then its weight is decreased.

To updates the weights of training data D, the algorithm computes the misclassification rate, the sum of the weights of each of the training example $t_i \in D$ that *were* misclassified. That is,

$$error(M_i) = \sum_{i}^{d} w_i * err(t_i); \quad (5)$$

Where $err(t_i)$ is the misclassification error of example t_i. If the example t_i was misclassified, then is $err(t_i)$ 1. Otherwise, it is 0. The misclassification rate affects how the weights of the training examples are updated. If a training example was correctly classified, its weight is multiplied by $error(M_i)/(1-error(M_i))$. Once the weights of all of the correctly classified examples are updated, the weights for all examples including the misclassified examples are normalized so that their sum remains the same as it was before. To normalize a weight, the algorithm multiplies the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified examples are increased and the weights of correctly classified examples are decreased. Now the algorithm generates another new data set D_i from training data D with maximum weight values and continues the process until all the training examples are correctly classified. Or, we can set the number of rounds that the algorithm will iterate the process. To classify a new/unseen example use all the prior and conditional probabilities of each cluster in each round and consider the class of new example with highest classifier's vote. The main procedure of proposed algorithm is described as follows:

Algorithm: A hybrid classifier algorithm – create an ensemble of classifiers using boosting, clustering, and naïve Bayesian classifier.

Input: D, Training data D of labeled examples t_i.

Output: A hybrid classifier model.

Procedure:

1. Initialize the weight $w_i=1/n$ of each example $t_i \in D$, where n is the total number of the examples.
2. Generate a new data set D_i with equal number of examples from D using selection with replacement technique. The same example form D may be selected more than once in D_i.
3. Arbitrarily choose k examples from data set D_i as the initial cluster centers.
4. Assign each example t_i from D_i to the cluster to which the example is the most similar, based on the attribute values of the examples in the cluster. Or, based on the mean value of the examples in the cluster.
5. Calculate the prior probability $P(C_j)$ for each class C_j in each cluster: $P(C_j) = \frac{\sum t_{i \to C_j}}{\sum_{i=1}^{n} t_i}$;
6. Calculate the conditional probabilities $P(A_{ij}|C_j)$ for each attribute values in each cluster:
$$P(A_{ij}|C_j) = \frac{\sum_{i=1}^{n} A_{i \to C_j}}{\sum t_{i \to C_j}};$$
7. Classify each training example t_i in training data D with maximum posterior probabilities from each clusters probabilities values using majority voting technique.
$$P(e_i \mid C_j) = P(C_j) \prod_{k=1 \to p} P(A_{ij} \mid C_j)$$
8. Updates the weights of each training examples $t_i \in D$, according to how they were classified. If an example was misclassified then its weight is increased, or if an example was correctly classified then its weight is decreased. To updates the weights of training examples the misclassification rate is calculated, the sum of the weights of each of the training example $t_i \in D$ that *were* misclassified: $error(M_i) = \sum_{i}^{d} w_i * err(t_i)$; Where $err(t_i)$ is the misclassification error of example t_i. If the example t_i was misclassified, then is $err(t_i)$ 1. Otherwise, it is 0. If a training example was correctly classified, its weight is multiplied by $error(M_i)/(1-error(M_i))$. Once the weights of all of the correctly classified examples are updated, the weights for all examples including the misclassified examples are normalized so that their sum remains the same as it was before. To normalize a weight, the algorithm multiplies the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified examples are increased and the weights of correctly classified examples are decreased.
9. Repeat steps 2 to 8 until all the weights of training examples $t_i \in D$ are correctly classified. Or we can define the number of rounds that the algorithm will iterate.
10. To classify a new/unseen example use all the probability set of each cluster in each round (each round is considered as a classifier) and consider the class of new example with highest classifier's vote.

IV. Experimental Result

The proposed classifier was tested on a number of widely used benchmark problems from UCI machine learning repository [16]. The datasets from UCI repository are relatively small size and often lacking time tag. The main reason of using datasets from UCI repository is that a number of solutions exist in the literature for classification.

1. Tic-Tac-Toe: It encodes the complete set of possible board configurations at the end of Tic-Tac-Toe games, where "x" is assumed to play first. The target concept is "win of x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). In Tic-Tac-Toe dataset, there are total 958 instances/examples (626 positive examples and 332 negative examples), number of classes: 2 (positive and negative), and the number of attributes: 9 (each attribute corresponding to one tic-tac-toe square and has 3 attribute values x, o, and b)
2. Soybean: There are 19 classes, only the first 15 of which have been used in prior work. The folklore seems to be that the last four classes are unjustified by the data since they have so few examples. There are 35 categorical attributes.
3. Iris: This data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.
4. Zoo: A simple database containing 17 Boolean-valued attributes. The "type" attribute appears to be the class attribute. Here is a breakdown of which animals are in which type: (I find it unusual that there are 2 instances of "frog" and one of "girl"!).
5. Diabetes: This data set prepared for the use of participants for the 1994 AAAI Spring Symposium on Artificial Intelligence in Medicine. Diabetes patient records were obtained from two sources: an automatic electronic recording device and paper records. The automatic device had an internal clock to timestamp events, whereas the paper records only provided "logical time" slots (breakfast, lunch, dinner, bedtime).

The experimental results in table 1 illustrate that proposed algorithm achieved better classification rates than naïve Bayesian classifier.

TABLE I. Classification rate on different domains

Dataset	No. of Class	No. of Attributes	No. of Cases	NB classifier	Proposed classifier
Tic-Tac-Toe	2	9	958	98.43	100
Soybean	19	35	683	97.65	100
Iris	3	4	151	92.43	99.25
Zoo	7	16	101	98.69	99.33
Diabetes	2	8	768	96.70	100
Vehicle	2	18	946	94.68	99.15
Monk 3	2	6	554	93.11	99.45

V. Conclusion

In this paper, we introduce a new hybrid classifier by using boosting, clustering, and naïve Bayesian classifier to develop the classification rates with very low error. Clustering splits the dataset into sub-datasets, which improves the values of probabilities. On the other side, the ensemble method of boosting improves the classification rate of classifiers. The naïve Bayesian classifier has several advantages such as it is easy to use and only one scan of training data is required. The naïve Bayesian classifier can easily handle the missing values by simply omitting the probability when calculating the likelihoods of membership in each class. The performance of our proposed classifier tested on seven benchmark datasets from UCI machine learning repository, and the experimental result proves that the proposed new classifier achieves high classification rates with very low misclassification error. The future research issue will be applying this classifier in classification problems of real world problem domains.

Acknowledgment

Support for this research received from Department of Computer Science and Engineering, United International University, Bangladesh.

References

[1] Y. Freund, and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, Vol. 55, 1997, pp. 119-139.

[2] L. Breiman, "Arcing classifier," *The Annals of Statistics*, 26, 1998, pp. 801-849.

[3] H. Drucker, and C. Cortes, "Boosting decision trees," *Advances in Neural Information Processing Systems*, 8, 1996, pp. 479-485.

[4] J. R. Quinlan, "Bagging, boosting, and C4.5," *Proc. of the 13 National Conference on Artificial Intelligence*, 1996, pp. 725-730.

[5] A. J. Grove, and D. Schuurmans, "Boosting in the limit: Maximizing the margin of learned ensembles," *Proc. of the 15 National Conference on Artificial Intelligence*, 1998.

[6] R. Jin, and G. Agrawal, "Efficient decision tree construction on streaming data," *In Proc. of ACM SIGKDD*, 2003, pp. 571-576.

[7] T. Hastie, R. Tibshirani, and J. Rriedman, "The element of statistical learning," *Data Mining, Inference and Prediction*, Heidelberg, Germany: Springer-Verlag, 2001.

[8] K. M. A. Chai, H. T. Ng, and H. L. Chieu, "Bayesian online classifiers for text classification and filtering," *In Proc. of SIGIR 2002*, Tampere, Finland, August 11-15, pp. 97-104.

[9] I. Kononenko, "Comparison of inductive and naïve Bayesian learning approaches to automatic knowledge acquisition," Wieling, B, (Ed), Current trend in knowledge acquisition, Amsterdam, IOS press, 1990.

[10] P. Langely, W. Iba, and K. Thomas, "An analysis of Bayesian classifier," *In Proc. of the 10th National Conference on Artificial Intelligence*, San Mateo, CA: AAAI Press, 1992, pp. 223-228.

[11] R. Kohavi, "Scaling up the accuracy of naïve Bayesian classifier: A decision tree hybrid," In Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, Menlo Park, CA: AAAI Press, pp. 202-207.

[12] Z. Zheng, and I. G. Webb, "Lazy learning of Bayesian roles," *Machine Learning-1*, Kluwer Academic Publishers, Boston, 2000, pp. 1-35.

[13] Z. Zheng, and G. I. Webb, "A heuristic lazy Bayesian rule algorithm," *Australian Data Mining Workshop in Conjunction with the 15th Australian Joint Conference on Artificial Intelligence*, Canberra, December, 2002, pp. 57-64.

[14] P. Langley, "Induction of recursive Bayesian classifier," *In Proc. of the European Conference on Machine Learning*, 1993, pp. 153-164.

[15] M. J. Pazzani, "Constructive induction of Cartesian product attributes," In Proc. of the Conference Information, Statistic and Induction in Science, ISIS, Singapore, 1996, pp. 66-67.

[16] The Archive UCI Machine Learning Datasets. http://archive.ics.uci.edu/ml/datasets/

AUTHORS PROFILE

A J. M. Abu Afza is currently completing Master of Science in Computer Science and Engineering from United International University, Bangladesh. He obtained B.Sc. Engineering in Computer Engineering from American International University of Bangladesh in 2004. He is an Assistant Programmer in Bangladesh Technical Education Board. He is also consultant of UNICEF as WIS monitoring project. He is a member of IEB and BCS in Bangladesh. He is trained from TATA Consultancy Service (TCS) Ltd., New Delhi, India from 20th March to 29th April, 2006 sponsored by BSM Division, Ministry of External Affairs, Government of India.

Dewan Md. Farid is a doctoral candidate in the Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh. He obtained B.Sc. Engineering in Computer Science and Engineering from Asian University of Bangladesh in 2003 and Master of Science in Computer Science and Engineering from United International University, Bangladesh in 2004. He is a part-time faculty member in the Department of Computer Science and Engineering, United International University, Bangladesh. He is a member of IEEE and IEEE Computer Society. He has published 3 international journals and 9 international conference papers in the field of data mining, machine learning, and intrusion detection. He has participated and presented his papers in international conferences at France, Italy, Portugal, and Malaysia. He worked as a visiting researcher at ERIC Laboratory, University Lumière Lyon 2 – France from 01-09-2009 to 30-06-2010.

Professor Dr. Chowdhury Mofizur Rahman had his B.Sc. (EEE) and M.Sc. (CSE) from Bangladesh University of Engineering and Technology (BUET) in 1989 and 1992 respectively. He earned his Ph.D from Tokyo Institute of Technology in 1996 under the auspices of Japanese Government scholarship. Prof Chowdhury is presently working as the Pro Vice Chancellor and acting treasurer of United International University (UIU), Dhaka, Bangladesh. He is also one of the founder trustees of UIU. Before joining UIU he worked as the head of Computer Science & Engineering department of Bangladesh University of Engineering & Technology which is the number one technical public university in Bangladesh. His research area covers Data Mining, Machine Learning, AI and Pattern Recognition. He is active in research activities and published around 100 technical papers in international journals and conferences. He was the Editor of IEB journal and worked as the moderator of NCC accredited centers in Bangladesh. He worked as the organizing chair and program committee member of a number of international conferences held in Bangladesh and abroad. At present he is acting as the coordinator from Bangladesh for EU sponsored eLINK project. Prof Chowdhury has been working as the external expert member for Computer Science departments of a number of renowned public and private universities in Bangladesh. He is actively contributing towards the national goal of converting the country towards Digital Bangladesh.