Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

New Ensemble Machine Learning Method for Classification and Prediction on Gene Expression Data

Ching Wei Wang
Vision and Artificial Intelligence Group, Department of Computing & Informatics, University of Lincoln
Brayford Pool, Lincoln LN6 7TS, United Kingdom
cweiwang@lincoln.ac.uk

Abstract –**A reliable and precise classification of tumours is essential for successful treatment of cancer. Recent researches have confirmed the utility of ensemble machine learning algorithms for gene expression data analysis. In this paper, a new ensemble machine learning algorithm is proposed for classification and prediction on gene expression data. The algorithm is tested and compared with three popular adopted ensembles, i.e. bagging, boosting and arcing. The results show that the proposed algorithm greatly outperforms existing methods, achieving high accuracy over 12 gene expression datasets.**

Index Terms – ensemble machine learning, pattern recognition, microarray

I. Introduction

One of the most active areas of researches in supervised learning has been to study methods for constructing good ensembles of classifiers. The main discovery is that the ensemble classifier constructed by ensemble machine learning algorithms, such as bagging and boosting approaches, often performs much better than single classifiers that make them up. Recent researches [1, 2] have confirmed the utility of ensemble machine learning algorithms for gene expression analysis. In this paper, a new ensemble machine learning algorithm is proposed for classification and prediction on gene expression data. The algorithm is tested and compared with other ensemble machine learning algorithms, including Bagging, Boosting and Arcing, over 12 gene expression datasets [5, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. The results show that the proposed algorithm greatly outperforms existing methods, achieving high accuracy in classification.

The outline of this paper is as follows. The ensemble machine learning approach and three popular ensembles, i.e. bagging, boosting and arcing, are introduced in section 2. The proposed method is presented in section 3. The evaluation on the proposed algorithm is displayed in section 4 with comparison on three ensembles. The paper is concluded in section 5.

II. Technical Analysis

A. Ensemble Machine Learning Methods

Ensemble methods are learning algorithms that construct a set of base classifiers and then classify new data points by taking a vote of their predictions. The spirit in ensemble machine learning is to combine a number of rough "rules-of-thumb" into a more accurate aggregate class prediction rule. The learning procedure for ensemble algorithms can be divided into the following two parts.

1*) Constructing base classifiers / base models*: The main tasks of this division are (1) Data processing: prepare the input training data for building base classifiers by perturbing the original training data, and (2) Base classifier constructions: build base classifiers on the perturbed data with a learning algorithm as the base learner. In this project, the C4.5 decision tree algorithm [4] is employed as the base learner.

2) Voting: The second stage of ensemble methods is to combine the base models built in the previous step into the final ensemble model. There are various kinds of voting systems. Two main voting systems are generally utilized, namely weighted voting and un-weighted voting. In the weighted voting system, each base classifier holds different voting power. On the other hand, in the un-weighted system, individual base classifier has equal weight, and the winner is the one with most number of votes.

B. Bagging, Boosting and Arcing

1) Bagging: The bagging algorithm, which is introduced by Breiman [11], constructs base classifiers with inputs generated by the bootstrapping technique. The construction process of every base classifier is independent to each other. It perturbs the training set repeatedly to generate multiple predictors, and combines these base classifiers by simple voting (classification) or averaging (regression) so as to obtain an aggregated predictor. The multiple input data for building base classifiers is formed by bootstrapping replicates of the original learning data.

2) Boosting: Boosting was introduced by Schapire [6] as a method to enhance the performance of a weak learning algorithm. Freund and Schapire [3] proposed an algorithm called AdaBoost. There are lots of varieties of boosting algorithms, and AdaBoostM1 is chosen as the boosting method used in this project. Boosting adaptively re-weights the training set in a way based on an error rate of the previous base classifier. The boosting algorithm improves its behaviour in reflection to the latest faults it makes. Moreover, if the error rate of a base classifier is greater than 0.5 or equal to 0, the sequential construction of base classifiers stops.

3) Arcing: The framework of arcing introduced by Breiman [7] is similar to the one employed in boosting. They

both proceed in sequential steps. The major difference between arcing and boosting is that arcing improves its behaviour based on the accumulation of its faults in history. It examines all previous base classifiers' faults for construction of a new base classifier while boosting only checks the previous one base classifier. Apart from this, arcing adopts un-weighted voting system whereas boosting uses weighted voting. In addition, unlike boosting, no checking procedure exists through the constructions of base classifiers.

III. METHOD

In this section, we first describe the disadvantage of existing ensembles and then further present the proposed approach, which defeats the weakness of current methods and utilizes the strength of them.

A. Analysis on Weakness of Existing Ensembles

The accuracy of boosting models will remain the same after specific numbers of base models are established because of the checking mechanism after each construction of base classifiers. The specific criterion in boosting stops further construction while its error rate is equal to 0 or greater than 0.5. Therefore, if the sequential construction halts after building 6 base classifiers, same result will be obtained on evaluating over boosting models with any number greater than 6, because fundamentally these models are all identical. In other words, as long as the error rate of the 1st base classifier is equal to 0 or greater than 0.5, no matter how many base models specified, the whole construction terminates, and consequently the entire ensemble model will be composed of exactly one base model. This is an extreme case in boosting, but it does happen very often in our experiments. As gene expression data consists of large amounts of genes, it helps learning methods to generate a more precise classifier that fit exactly on the training set. The checking criterion seriously influences the diversity of boosted models by forbidding further construction of base models.

On the other hand, the performance of arcing models may deteriorate while more base classifiers are constructed. Unlike boosting, without the checking condition interrupting, the other unwelcome situation may occur. The extreme undesirable result is an arced model with all identical base models. Without the checking criterion, the arcing may keep on producing same base models. For example, once arcing develops a base classifier that precisely fits the training data, there will be no misclassification value to be added in, and all misclassification values will remain the same. Hence, the instances' weights will stay the same as well. With the same data, the same instances' weights and the same base learner, the arcing algorithm will produce exactly the same base classifier. As a result, due to the low diversity issue of base models, ensembles with self-optimization learning style may suffer from over-fitting issue.

On the contrary, bagging generates its base models by chance. The constructions of individual base model are independent to each other.

B. Design

The advantage of boosting and arcing is to refine their behavior based on previous experience. However, over-fitting issue decreases the quality of these ensemble models. Hence, a data-perturbing technique is created for individual construction on base classifiers in order to produce base models in higher diversity. Particularly, boosting refines itself based on errors by the previous model whereas arcing learns based on errors by all previous models. We found that boosting structure obtains higher accuracy (See Table I). Simulating human learning behavior, because the errors have been adjusted through constructions of base models, it is not necessary to pay attention on errors by all previous models. Hence, the boosting structure is adopted for sequential self-optimization.

1) Sequential self-optimization: Refine its behavior based on its previous experiences. In other words, pay more attention on misclassified instances by previous base models.

2) Data Bootstrapping: achieve higher diversity and defeat overfit issue.

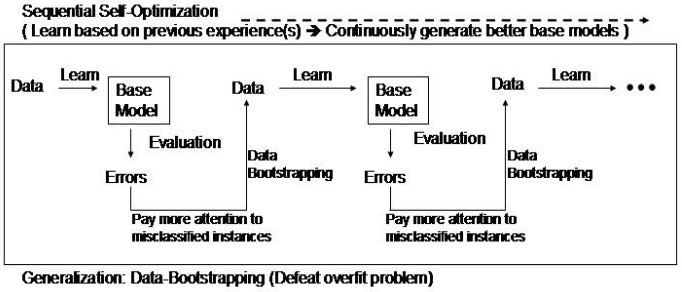

Fig. 1 Behavior of the Proposed Ensemble

C. Design: Data Bootstrapping

The bootstrapping method is to generate a new dataset by sampling from the given dataset. Therefore, some instances may be selected several times whereas others may be ignored. For every bootstrapping, a list of random floating numbers will be generated. The number of the random values is exactly the same as the size of input data set. These floating numbers are then processed into a list of accumulated probabilities and the maximum value of this distribution is adjusted to the summation of instances' weights. In brief, a random distribution is generated, and the distribution is represented by accumulated probabilities. Equation 1 presents the computation of each accumulated probability:

$$P(a) = [\sum_{i=1}^{a} \text{Random(i)} \ / \sum_{j=1}^{n} P(j)] \cdot \sum_{k=1}^{n} W(k) \quad (1)$$

Afterwards, the P(a) value is compared to the accumulated weight value – ΣW(b).

$$\sum W(b) = \sum_{i=1}^{b} Weight\ (i) \quad (2)$$

If the P(a) < ΣW(b), then add instance b into the result bootstrapped dataset, and the next comparison will be made between P(a+1) and ΣW(b). Otherwise, do not select instance b, and the next comparison will be made between P(a) and ΣW(b+1). As a result, one instance may be selected many times, and the higher its weight is the greater the accumulated weights value becomes, contributing to a bigger chance to be selected. The algorithm is devised in Fig. 2.

Input: A dataset with n instances
Output: A bootstrapped dataset with the same size of the input dataset
Assumption:
Each instance in the original dataset has a weight, which represent the relative level of importance in the dataset.
Process:
1. Generate n random numbers: R_1 to R_n.
** Compute n accumulated probabilities P_1 to P_n
2. P(i) = sum of Random number R_1 to R_i, where i =1 to n.
** Normalize probabilities' values
3. P(m) = [P(m) / (Sum of Probabilities)] * (Sum of weights), where m = 1 to n.
** Compare accumulated probabilities with accumulated weights
4. Initialize two indices for iteration through two lists.
k = the index of the first probability in probabilities' list
x = the index of the first instance's weight in weights' list
** Iterate through lists of probabilities and instances' weights
5. Check if any components left in both lists
5.1 If yes, sumW(x) = sum of weight1 to weight x
5.1.1 Check if components left in probabilities list and P(k) is smaller than sumW(x)
5.1.1.1 If yes, add instance x to the output dataset, set the weight of instance k of the output dataset as x and plus 1 to k.
5.2 Otherwise, plus 1 to x.
6. Terminate with the output dataset

Fig. 2 Algorithm: Data Bootstapping

D. Algorithm

The proposed method is composed of a sequential self-optimization structure (boosting) and data-bootstrapping technique.

Inputs:
1. A training set T < X, Y >, where X represents the instances and Y are the classes.
X: a set of instances: $\{x \mid x = <a_1, \ldots, a_q>\}$, where a_i is an attribute value and q is the number of attributes.
Y: a set of classes (with z different classes)
T: $\{<x_1, y_1>, \ldots, <x_n, y_n> \mid x \in X, y \in Y\}$, where n is the size of the training set.
2. Number of base classifiers R
3. The limit value of bootstrap times
4. Base Learner / Inducer: C4.5 decision tree is used in this paper.
Output:
The boosted model: Function C*
Steps:
** Initialise instances' weights (Normalisation)
1. For i = 1 to n, Weight: $W_1(i) = 1/n$
2. Generate a copy of the training data for constructing base classifiers: S (the training data will be used for evaluation whereas the copy - S is used for building base classifier and its instances' weights will be changed in every construction.)
3. Repeat 3.1 to 3.9 R times
3.1 Bootstrap S dataset from previous round S dataset
3.2 Build a new Classifier $C_r(X)$ using weighted S dataset (X, W_r) by base learner.
3.3 Compute the error rate by evaluate the base classifier $C_r(X)$ with training data. Error rate = Sum of the weights of the misclassified instances by the base classifier $C_r(X)$
3.4 Check if the error rate equals to 0 and the number of bootstrap is smaller than the bootstrap limit. If both true, go back to 3.1 to do the bootstrap.
3.5 If the error bigger than 0.5 or equal to 0, go to step 4.
3.6 Br = error rate / (1 - error rate)
** Set sum of instance weights for next round to 0
3.7 $SumW_{r+1} = 0$
** Update instances' weights
3.8 For i = 1 to n, check if Cr misclassifies instance i.
3.8.1 If true, $W_{r+1}(i) = W_r(i) * B_r$ Otherwise, $W_{r+1}(i) = W_r(i)$
3.8.2 $SumW_{r+1} = SumW_{r+1} + W_{r+1}(i)$
** Normalise instances' weights
3.9 For i = 1 to n, $W_{r+1}(i) = W_{r+1}(i) / SumW_{r+1}$
4. Produce the arced model ***Function C*(instance)*** by **Voting**
5. Return ***Function C****

Auxiliary algorithm: Voting

Function C*(instance)
Input: instance
Output: predicted result y_i
Steps:
** Initialise votes of classes $y_1 \sim y_z$ to 0
1. For i = 1 to z
1.1 V(i) = 0
2. For j = 1 to R
(i = the class index generated by base classifier j in classifying the input instance.)
2.1 $V(i) = V(i) + \log(1/B_j)$
3. Find class y_i with the largest vote V(i)
4. Terminate with the predicted result y_i

Fig. 3 Sequential Self-Optimization + Bootstrapping

IV. Experimental Results

Cross-validation is deemed as an objective and common used tool in model selection. Especially, cross-validation is markedly superior for datasets with small number of samples, which exactly matches the gene expression data case; this fact is demonstrated in [12]. In this paper, 10-fold cross validation is utilized for evaluation. Moreover, the experiments are conducted in a linux based cluster [23]. 12 gene expression datasets are obtained from published research works [5, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22], and the C4.5 decision tree algorithm [4] is employed as the base learner. Three existing approaches, namely arcing, boosting and bagging, and two new proposed methods, i.e. (New1) Data Bootstrapping + Arcing Skeleton and (New2) Data Bootstrapping + Boosting Skeleton, are tested.

Table 1 presents the percentage of accuracy in classification over 12 gene expression datasets by the five ensembles. The experimental results show that the proposed ensemble classifiers stably achieve high accuracy over all 12 gene expression datasets in comparison to existing ensembles. Particularly, the combination of data bootstrapping and boosting skeleton (New2) performs best in classifying gene

TABLE I

Dataset	Arcing	Boost	Bagging	New1	New2
AMLALL	88.9	91.7	94.4	100	100
Brain Cancer	84	82	84	94	94
Breast Cancer	80.41	85.57	90.72	94.85	95.88
CNS	78.33	90	88.33	93.33	95
Colon Tumor	69.35	80.65	79.03	79.03	83.87
Lung Cancer	97.24	97.24	97.79	98.34	99.45
Prostate Outcome	87.5	90.44	94.12	94.85	97.06
Prostate Tumor	66.67	76.19	61.9	95.24	100
DLBCL Outcome	84.48	93.1	98.28	100	100
DLBCL Tumor	85.71	94.81	92.21	97.4	98.7
ALL,MLL, AML Subtypes	91.67	91.67	91.67	93.06	98.61
	80.12	92.66	91.44	89.3	93.88

expression data among five models.

V. Conclusion

Accompanied with the sequential self-optimization structure, the entire ensemble model is inclined to refine its behavior in a correct direction. The experimental results on 12 gene expression datasets prove that the proposed ensemble classifiers are largely upgraded with the data-bootstrapping technique. The combination of data bootstrapping and boosting skeleton (New2) is able to obtain high accuracy stably and hence it is recommended for gene expression data analysis.

VI. Future Work

Obtaining patterns and rules with high accuracy in classification and prediction, we would like to extract information about sets of influential attributes / genes from the resulting patterns and rules, allowing advanced study in biological meaning of genes and drug discovery. With the decision tree algorithm as the base learner in this study, a further investigation can be made to discover sets of influential genes. That is to consider the relative importance of attributes within each base model. The most important criterion in classification will be placed as the first priority in consideration, i.e. the attributes in upper tier are more meaningful in the classifying procedure. Hence, to obtain a comprehensive view on the importance of attributes in the entire ensemble model, a design is created to exhibit attributes' precedence among all base models. The plan is to assign a score for each tree level, and the attributes gain points related to the score in the particular tree level. By summing up an attribute's points gained in all base models, its total score can then be obtained. The design scheme is presented in Fig. 4.

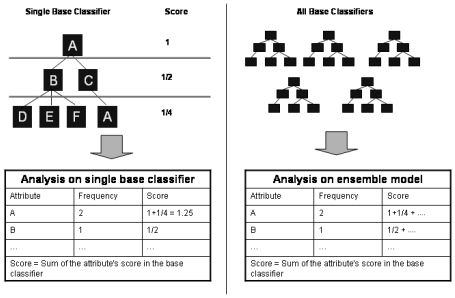

Fig. 4 Computation on Decision Power of Genes / Attributes

VII. Acknowledgments

The author is thankful to Prof. David Gilbert and Dr. Aik Choon Tan for their valuable comments where this study was conducted. Three software programs are imported, namely WEKA machine learning package [8] for boosting and bagging and C4.5 decision tree models, OAIDTB Boosting Extension [9] for a variety of boosting algorithms and JFreeChart [10] for a delicate visualization in presentation.

References

[1] Dettling M. [2004] BagBoosting for tumor classification with gene expression data, Bioinformatics 20(18):3583-3593.
[2] Tan A.C. and Gilbert D. [2003] Ensemble Machine Learning on Gene Expression Data for Cancer Classification, Applied BioInformatics, in press
[3] Freund Y. and Schapire R. [1996] Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth Internation Conference.
[4] Quinlan J. R. [1996] Bagging, boosting, and C4.5. Proceedings of the Thirteenth National Conference on Artificial Intelligence:725-730.
[5] Armstrong S. A., Staunton J. E., Silverman L. B., Pieters R., et al. [2002] MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nature Genetics 30:41-47
[6] Schapire R. [1990] The strength of weak learnability, Machine Learning 5(2):197-227
[7] Breiman L. [1998] Arcing Classifiers, the Annals of Statistics, 26(3):801-849
[8] Ian H. W. and Frank E. [2005] Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann
[9] Santiago D. V. B. [2003] http://pisuerga.inf.ubu.es/lsi/Software/oaidtb/
[10] JFreeChart, http://www.jfree.org/jfreechart/index.html
[11] Breiman, L. [1996b] Bagging predictors, Machine Learning 26(2):123-140
[12] Goutte C. [1997] Note on free lunches and cross-validation, Neural Computation 9:1245-1249
[13] Golub T. R., Slonim D. K., Tamayo P., Huard C., Gaasenbeek M., et al. [1999] Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286: 531-537
[14] Catherine L. N., Mani D. R. , Rebecca A. B., Pablo T. et al [2003] Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Research 63(7):1602-1607
[15] Zembutsu H., Ohnishi Y., Tsunoda T., Furukawa Y., Katagiri T., Ueyama Y., et al. [2002] Genome-wide cDNA microarray screening to correlate gene expression profiles with sensitivity of 85 human cancer xenografts to anticancer drugs. Cancer Res 62(2):518-27
[16] Scott L. Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M. Sturla, et al [2002] Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature 415:436-442
[17] Alon U., Barkai N., Notterman D. A., Gish K., Ybarra S., Mack D., and Levine A. J.. [1999] Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. National Academy of Science, Cell Biology 96:6745-6750
[18] Gavin J. G., Roderick V. J., Li-Li H., Steven R. G., Joshua E. B., Sridhar R., William G. R., David J. S., and Raphael B. [2002] Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma. Cancer Research 62:4963-4967
[19] Dinesh S., Phillip G. F., Kenneth R., Donald G. J., Judith M., et al. [2002] Gene expression correlates of clinical prostate cancer behavior. Cancer Cell 1:203-209
[20] Ash A. A., Michael B. E., Davis R. E. , Ma C., Izidore S. L., Andreas R., et al. [2000] Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403:503-511
[21] van 't Veer L. J., Dai H., van de Vijver M. J., He Y. D., Hart A. A., Mao M., Peterse H. L., et al. [2002] Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871):484-5
[22] Yeoh E. J., Ross M. E., Shurtleff S. A., Williams W. K., Patel D., Mahfouz R., Behm F. G., et al. [2002] Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1(2):133-143
[23] ScotGrid, http://www.scotgrid.ac.uk/