

A Comparison of Ensemble Methods for Microarray Data Analysis

- Vishakh (vv2131@columbia.edu)

Abstract: Machine Learning tools are increasingly being applied to analyze data from microarray experiments. These include ensemble methods where weighted votes of constructed base classifiers are used to classify data. We compare the performance of AdaBoost, bagging and BagBoost on gene expression data from the yeast cell cycle. AdaBoost was found to be more effective for the data than bagging. BagBoost offered an advantage over AdaBoost due to the combination of the benefits of both bagging and boosting.

1. Introduction

The analysis of microarray data is becoming increasingly important since it has tremendous potential for aiding the understanding of genetic expression. Applications of this analysis include functional genome analysis, disease diagnosis, pharmacogenomics and toxicogenomics.

Several machine learning algorithms have been applied to the analysis of microarray data. Gene expression profiles generated by microarray experiments can be quite complex since these experiments can involve hypotheses involving entire genomes. It is only natural, then, to apply tried and tested machine learning algorithms to analyze this data in a timely, automated and cost-effective way.

The goal of the project was to study the applicability of two ensemble methods, i.e. AdaBoost and boosting, and their combination, BagBoost, to microarray data analysis. The dataset used for this was derived from microarray experiments measuring the expression levels of genes at various points in the cell cycle of yeast. Each instance represented a gene at a given time point. The features were combinations of motifs

associated with the genes and a list of known regulators. A feature for a given instance was labeled as present if the motif in the motif-regulator pair it represents was associated with the gene and the regulator was up-regulated. The label the resultant classifiers had to predict was whether the given genes were up or down regulated at the time point.

2. Algorithms

Ensemble techniques for classification use a weighted combination of many classifiers instead of using just one classifier. We now describe the ones studied for this project.

2.1 AdaBoost

Boosting is a meta-algorithm for improving on the accuracy of a weak learner while performing supervised machine learning. A weak learner is a machine learning algorithm that classifies data with accuracy greater than that of chance. Boosting runs the weak learner iteratively on the training data, rearranging the probability distribution of the given examples so that subsequent iterations of the weak learner focus on the ones that have not been accurately learnt yet. The algorithm then combines the hypotheses generated at each iteration and uses them to construct a classifier that has greater accuracy than the weak learner.

AdaBoost (short for Adaptive Boosting) was the particular variant of boosting under study in this project. AdaBoost, like other boosting algorithms, repeatedly calls a weak learner to construct several base hypotheses. It then combines these hypotheses using a weighted majority vote to construct a classifier. The pseudocode for AdaBoost is:

Inputs:

- A training set, X , consisting of labeled examples: $\{(x_1, y_1), \dots, (x_n, y_n)\}$
- A weak learner L .

Algorithm:

Create a probability distribution over the training set, initially setting the probability weight of each x_i , $D_1(x_i)$, to $1/m$.

Iterate T times and for each t from 1 to n

Call L with parameters D_t and X and get a hypothesis h_t .

Calculate ϵ_t , the weighted error of h_t using the formula:

$$\epsilon_t = \sum_{i=1}^m D_i \{y_i \neq h_t(x_i)\}$$

Let $\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$

Reweight the distribution such that

$D_{t+1}(x_i) = (D_t(x_i) * \exp(-\alpha_t * y_i * h_t(x_i))) / Z_t$
where Z_t is a normalization factor such that D_{t+1} will sum to 1.

Output:

$H(x) = \text{sign} \left(\sum_{i=1}^T \alpha_i * h_i(x) \right)$, the overall hypothesis.

2.2. Bagging

The bagging algorithm also creates a classifier using a combination of base classifiers. However, instead of iteratively reweighing the instances before each call of the base learner, it creates a replicate dataset of the same size as the original. It does this by sampling from the original dataset with replacement. It then calls the base learner on the replicate to get a classifier C_t . After doing this for the set number of iterations, it creates the overall classifier, C^* , by combining the base classifiers with a majority vote. For a given instance x :

Loop over base classifiers C_t

Loop over classes k

$V_k = V_k + 1$ if $C_t(x) = k$

$C^* = k$ such that V_k is the maximum.

2.3 Decision Stumps

The weak learners used with AdaBoost and bagging for the project were decision stumps. These very simple classifiers are elegant and easy to implement. Generally, a decision stump represents the presence or absence of a feature of the training set. If a feature i is picked as a stump, then for all instances in which i is present, the stump predicts a positive label, negative otherwise. In case binary attributes were not being used, a stump would simply be a feature and an associated threshold value for it. However, for this project, a variation of the above concept was used. A decision stump here represented whether the presence or absence of a feature was associated with the up or down regulation of a gene. While the up-regulation of a gene in the absence of a chosen feature is not biologically significant, the corresponding stumps can be ignored and the remaining stumps would certainly be biologically meaningful. At each iteration of AdaBoost, a stump was chosen such that the weighted training loss was minimized.

2.4 BagBoost

BagBoost is a variant of boosting which uses bagging. The boosting procedure, instead of using a weighted combination of individual base learners, uses a weighted combination of bagged base learners. The motivation behind using BagBoost on the microarray data was to check whether it would combine the benefits of boosting and bagging. While boosting reduces the bias of the base learner, the resultant classifier shows slightly higher variance. Bagging, on the other hand, lowers variance but does not reduce the bias substantially. It was hoped that by combining the two ensemble methods, classifiers with both lowered bias and variance could be constructed. For this project, AdaBoost was used on bagged

decision stumps.

3. Results

The results of running AdaBoost for different number of iterations are show below:

<i>Iterations</i>	<i>Training Error</i>	<i>Test Error</i>
5	41.16	42.82
10	39.49	41.14
20	37.45	39.59
30	36.87	39.05
50	35.12	37.84

It can be seen that AdaBoost iteratively improves both the training and test performances. AdaBoost effectively calls the base classifier, correctly readjusting the instance weights to focus on the instances not yet learned accurately. It can be safely posited that running the procedure for more iterations would lead to a further increase in accuracy.

The results of running bagging with different numbers of replicates are shown below:

<i>Replicates</i>	<i>Training Error</i>	<i>Test Error</i>
5	44.13	44.75
10	44.13	44.75
20	44.13	44.75
30	44.13	44.75
50	44.13	44.75

There is no improvement in results even when a much larger number of replicates is used. In fact, it seems that bagging is not doing a better job than the base classifier. However, we must keep in mind that bagging is not supposed to reduce the bias by much, but decrease the variance. Any benefits of bagging for this data might be seen

when it is combined with boosting.

The results of running BagBoost with different numbers of iterations and five replicates in each iteration are show below:

<i>Iterations</i>	<i>Training Error</i>	<i>Test Error</i>
5	41.22	42.43
10	39.11	40.62
20	38.44	40.38
30	37.27	39.57

We see here that BagBoost does better than AdaBoost initially, but then becomes worse after that. In the later iterations, low-accuracy instances have much higher weights and tend to be sampled more often and overrepresented in the bagging replicates to the point where the obtained base learners concentrate too heavily on them and do not generalize well. Initially, bagging helps the boosting procedure due to the decreased variance when the base classifier is obtained. It should be noted that BagBoost's execution time is substantially more than that for Adaboost.

4. Conclusions

Three ensemble methods for classification-AdaBoost, bagging and BagBoosting- were used on microarray data derived from yeast to test their effectiveness. It was hoped that combining boosting's decrease of bias and bagging's decrease of variance would lead to a better classifier than either constituent. AdaBoost performed better than bagging on the data and was found to be an effective classifier which iteratively decreased both training and test error. BagBoost, while also classifying effectively, did not perform as well as AdaBoost. Even the improvement in the initial iterations was marginal and given the additional recourses consumed by BagBoost, the author would recommend Adaboost for classification on this and similar datasets.

References

M Middendorf, A Kundaje et al. Predicting Genetic Regulatory Response Using Classification. 2003.

M Dettling. BagBoosting for Tumor Classification with Gene Expression Data. 2004.

J.R. Quinlan. Bagging, Boosting and C4.5. 2006.

R Schapire. The Boosting Approach to Machine Learning. 2001.

P Long & V Vega. Boosting and Microarray Data. 2003.

T Nicholas. Using AdaBoost with Decision Stumps to Identify Spam E-mail. 2003.

J Rennie. Boosting With Decision Stumps and Binary Features. 2003.

T Paul & H Iba. Extraction of Informative Genes from Microarray Data. 2005.

M Dettling & P Bühlmann. Boosting for Tumor Classification with Gene Expression Data. 2002.