Experiments with Adaboost and linear programming *

Maria Dolores Valverde, Francesc J. Ferri
Dept.d'Informàtica
Universitat de València
46100 Burjassot (València). Spain
{M.Dolores.Valverde, Francesc.J.Ferri}@uv.es

Abstract

In the context of classification problems, boosting refers to methods that create an ensemble of 'weak' classifiers in order to get a combined classifier that improves the performance of any of the weak classifiers combined and eventually leads to competitive results. It has been reported that the performance of boosting is related to the diversity of the combined classifiers. This paper presents a preliminary approach to obtain a sparse combination of classifiers from a previously learned ensemble that can improve boosting results in such cases.

1 Introduction

The basic idea of boosting methods is to create a combination of the outputs of many weak classifiers in order to get a better one. This weak classifiers, called also base classifiers could be at first any given classifier slightly better than random. Then the motivation of this combination is to gently improve the performance of any of the single classifiers that are combined.
Adaboost [2] [3] is one of the most well-known boosting algorithms that sequentially creates an ensemble of classifiers trained on different versions of the training set chosen deterministically in such a way that in each step more emphasis is given on examples that are misclassified in previous steps. Adaboost has created so much interest both because its simplicity and because of the fact that very little prior knowledge seems to be needed in principle.
On the other hand, it depends basically on the dataset and the selected base classifier. Many researchers [1] [7] show that Adaboost could be interpreted as an approximation of a solution of a linear programming that tries to maximize the minimum margin of the ensemble. The margin for an example, could be understood as a measure of confidence in the prediction. Then Adaboost maximizes the margin of all the examples [3] concentrating in each step on the examples misclassified, i.e., examples that have the minimum margins.
Dietterich showed in [8] that Adaboost creates very diverse classifiers in the absence of noise but in the presence of noise the combined classifier deteriorates because the individual classifiers are less diverse and have high individual error rates. This behavior can be also understood as a consequence of the strategy of the margin. The added noisy prototypes have the high weights and then Adaboost concentrates too much on them creating classifiers with poor performance.

*This work has been partially supported by grants TIC2003-08496-C04-01, TIC2003-08496-C04-01, TIC2002-12744-E and TIN2004-21343-E

Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005, pp.153-158
ISBN: 84-9732-449-8 © 2005 Los autores, Thomson

2 Adaboost

The Adaboost algorithm described below, takes as input a training set S=$\{(x_i, y_i)\}$ where each x_i lies in some space $\Re^d$ and y_i are labels in the set $Y = \{-1, 1\}$. Other inputs are the base learner H, i.e., the classification rule that will give a classifier h_t at each step, and T that it is the number of maximum iterations.
The idea of Adaboost is to create sequentially T classifiers over different distributions of the training set.

```
Input:  (S={(x_i, y_i)}  i = 1, ..., N,
         H base learner, T max iter)
Initialize D_1(x_i) = 1/N
  for t = 1, ..., T
     Train h_t over subset S_t sampled
        from S with distribution D_t

     Compute the weighted error for h_t:
        ε_t = Σ_{i=1}^{N} w_i I(h_t(x_i) = y_i)

        if h_t is worse than random or
           have zero training error,
           STOP

     Compute alpha, coefficient of h_t
        α_t = log((1 − ε_t)/ε_t)

     Update the distribution D_t
     D_{t+1}(x_i) = D_t(x_i) exp (−α_t y_i h_t(x_i))

  end

Output final classifier,
F(x) = sign(Σ_{t=1}^{T} α_t h_t(x))
```

Algoritmo 1: Adaboost

The selection of the training set in each step could be done in two ways, one is boosting by reweighting and the other one is boosting by resampling [3]. In order to consider a wider family of classifiers including those that cannot explicitly deal with weights, only the resampling strategy has been considerer in this preliminary experimentation.
In the first step, every point in the training set has the same relevance and Adaboost initializes D_1 as a uniform distribution. This distribution is updated at each iteration and the weight of an example increases when it is misclassified and decreases when it is correctly classified. The weight of a training object roughly reflects at each moment the importance of this object for the classification task. The higher the weight the more priority for being well classified in the next step. In this way the algorithm tries to compensate the classifiers because in the end we want the combined classifier to be more robust. At the same time, each classifier will have a different performance over the weighted training set, and this information is used for create the coefficients α_t of each classifier in the combined classifier. It is in some way a measure of the relevance of each classifier in the combined rule, because it depends on the weighted error of the classifier. The smaller the error ϵ_t the higher the α_t and then the classifier associated h_t will have more relevance in the combined classifier.

3 Improving the results of Adaboost

One of the most important components of Adaboost is the base classifier used. In [9], experiments with different linear discriminants as base classifiers in Adaboost are carried out to solve problems with small sample size. When the datasets are large the question of what to use as base classifier still remains an open problem. In this report, the first part of the experiments consist of using four different classification rules as base classifiers in order to see whether Adaboost could improve the performance of any of this classification rules. The classification rules considered are:

1. Decision Trees, used frequently in data mining because not need for prior knowledge of the data.

2. Linear Discriminant classifier. This classifier is optimal for the case of two normals with the same covariance matrix. If this is not the case, it tries to minimize the mean square error.

3. Nearest Neighbor rules, that classify each example assigning the label of the nearest object in the training set.

4. Nearest Mean classifier, a very simple linear classifier that estimates the means of the classes and generate the bisection of them.

The second part of this work focuses in the study on the classifiers created by the boosting process. As is explained by Dietterich [8], the good performance of Adaboost has a direct relation with the diversity of the classifiers in the ensemble. When the classifiers are very similar, selecting some of them could increase the relative diversity and then lead to better generalization error. Adaboost creates a linear classifier in the space of labels, whose coefficients are $(\alpha_1, ..., \alpha_t)$. The aim of our approach is to find a linear classifier in this space, that only uses few of the classifiers. Selecting classifiers can be accomplished by making some of the α coefficients vanish. One way to do this is by solving a linear program that tries to create a linear classifier in this space. Recent research has put forward the properties of the boosting process and linear programming formulations has been proposed in order to study what are the characteristics that make Adaboost work so well as in [1], or to improve the performance of Adaboost as in [7]. In this report, a linear programming formulation [4] is also used but with the intention of selecting classifiers and creating a new combination of the classifiers. Fung and Mangasarian describe a linear programming formulation that use a fast Newton method which founds an exact least 2-norm formulation to the SVM classifier. This approach is based on a 1-norm formulation of (soft margin) SVM that gives as a result a sparse solution.

$$\texttt{min} \quad \|w\|_1 + \nu \sum_{i=1}^{N} \xi_i$$

$$\texttt{s.t.} \quad y_i(A_i w) + \xi_i \ \texttt{>= 0}$$

$$\xi_i \texttt{>=0} \quad \texttt{for } i = 1, ..., N.$$

Algoritmo 2: Linear Programming (LP)

This means that the SVM classifier only depends on a few features. In the same way, we want our ensemble to rely only in few, diverse enough classifiers. Consequently, the same formulation is used but on the outputs of the classifiers as new feature space. The Algorithm 3 shows the formulation of this linear program. A, is a N x T matrix with elements given by $A_i^t = h_t(x_i)$. Note that the elements of A are 1 or -1. The ξ_i for $i = 1, .., N$ represent the misclassification for each example that can be understood as the quantity necessary to satisfy the margin constraint. Minimizing the sum of this variables is then a strategy to minimize the number of examples misclassified. Applying the linear programming to this matrix, will give a linear classifier constructed in the T-dimensional space of labels, as Adaboost do, but with the target now of getting a sparse solution in this space that only works with a few classifiers. The sparse vector of coefficients is represented by w leading to the new combined classifier : $\sum_{t=1}^{T} w_j h_j$.The best possible solution is searched changing the value of ν that it is the parameter that represents the trade off between the importance of minimizing the 1-norm of the vector w and the sum of the misclassifications $\sum_{i=1}^{N} \xi_i$.

4 Experimental study

4.1 Setup

The datasets used are Phoneme from ELENA project[6] and Satimage from the UCI Machine Learning Repository [5]. Phoneme consist of five dimensional data that tries to distinguish between nasal and oral vowels. Satimage has six classes originally. Some classes were collapsed and here is presented as a skewed two class dataset.The table 1 describes the datasets:

Dataset	Features	Ex. per class		Total
Phoneme	5	3818	1586	5404
Satimage	17	5106	1329	6435

Table 1: Datasets

The two datasets have only two classes and the outputs of the classifiers will be $+1, -1$.

	TstEr	Cl	AB	LP
Pho	ldc	**0.2419**	0.3501	0.2517 (447±74)
	nmc	0.2609	0.3806	**0.2311** (404±38)
	tree	0.3329	**0.1860**	0.2167 (500)
	1-nn	**0.0951**	0.1214	0.1710 (386±50)
Sat	ldc	**0.1091**	0.1905	0.1125 404±60
	nmc	0.1436	0.3584	**0.1382** (290±25)
	tree	0.1155	**0.0827**	0.1209 (484±5)
	1-nn	**0.0625**	0.0817	0.0803 (257±32)

Table 2: Results

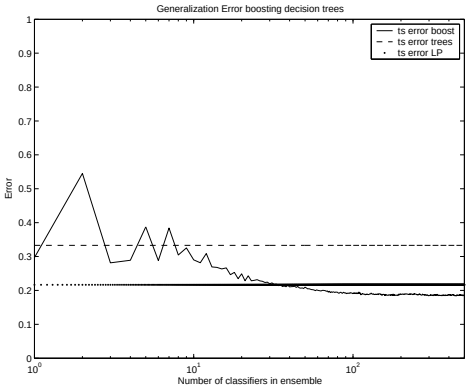

Figure 1: Dataset Phoneme. Adaboost with decision trees

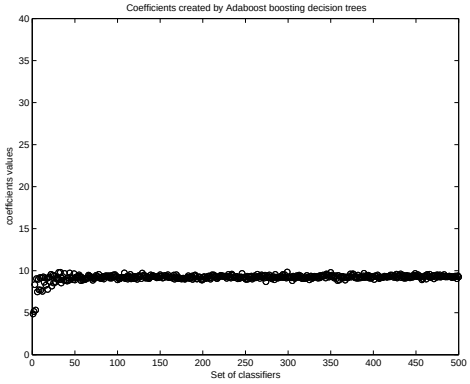

Figure 2: Dataset Phoneme. Alphas for the combination of decision trees created by Adaboost

The data is divided into five folds and for each of them in a cross validation way, it is constructed a combined classifier of 500 classifiers using Adaboost for each of the four classification rules. The resulting matrix of labels is used for the linear programming with different values of the parameter ν that controls the trade off in the LP formulation. The final value of ν is close to 0.01 in most of the cases.

4.2 Results

In table 2, the averages over the five results for each classifier, for Adaboost at last step and for the best LP result are shown. The number of selected classifiers is also shown in brackets. The best result in each case is in bold.

The figures 1, 3 show the test error for some of the results in table 1. The curves represent the test error of the combined classifier on each step of Adaboost. Horizontal lines show test error for a single classifier, and for the best LP result.
Figures 2 and 4 show the alphas, i.e., the coefficients created by Adaboost for each classifier to be combined in the ensemble. The figures 5, 6 give a representation of the distribution of the different normalized coefficients both for Adaboost and for LP.

5 Discussion and Concluding Remarks

The first thing observed is that in both cases the nearest neighbor rule gives the best result. In three of the four classification rules investigated, work with Adaboost is useless, and in the cases of linear discriminant and the nearest mean rule are even much worse than for the single rule. In [9] nearest mean was used with Adaboost with success. The reason could be the choice of the reweighted technique instead of resampling used here in each step for training the classifiers. Also the properties of

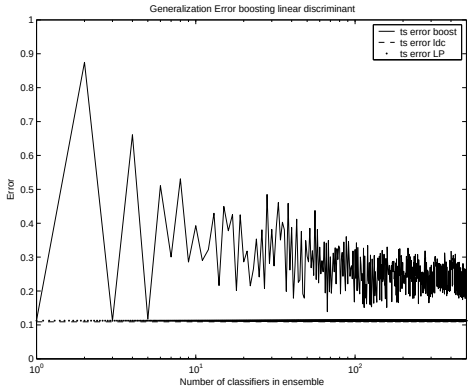

Figure 3: Dataset Satimage. Adaboost with linear discriminat

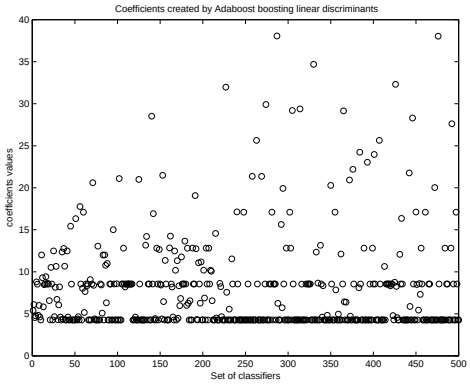

Figure 4: Dataset Satimage. Alphas for the combination of linear discriminant created by Adaboost

these datasets could be important in their performance, due to they have overlapping area that makes hard the description of their border by linear classifiers. In the figure 2 it is showed the behavior in the case of decision trees or nearest neighbor, where it can be observed that the value of the alpha does not decrease as Adaboost creates more classifiers. The figure 4 shows the behavior in the case of linear discriminant and nearest mean, where the value of alpha flips between two ranges. This probably means that Adaboost creates overall two types of classifiers and then classifiers very similar that does not converge to a combined classifier more robust.

With respect to the second question referring to linear programming, experiments indicate that the combined classifier created by this linear programming usually increases the generalization error in the cases where Adaboost improves the single classification rule and decrease in the cases where Adaboost does not improve it. In general the reduction in the number of classifiers combined as solution of linear programming is not very significant in relation with their performance. The figures 5 and 6 show the normalized alphas created by Adaboost and by LP arranged in increasing order. The curve that represents the alphas of linear programming, shows in both figures that most of the coefficients are almost zero or have a small value in comparison with the highest ones. This could mean that most of the classifiers combined by linear programming are irrelevant for the classification task and the sparsity is in fact more significant than the results show.

6 Future Work

In the future, it could be interesting to study if these results are the same using the continues confidences for the linear programming instead of crisp labels of the classifiers. Other techniques to select features could be also applicable to this problems and other LP formulation could be considered.

References

[1] Grove, A. J. & Schuurmans D.*Boosting in the limit: maximizing the margin of learned ensembles, In Proceedings of the Fifteenth National Conference on Artificial Intelligence. AAAI Press* , Meno Park, NJ, 1998.

[2] Freund Y. & Schapire, R. E., *Experiments with a new boosting algorithm,In Machine Learning: Proceedings of the Thirteenth Intenational Conference*,148-156.Morgan Kauffman, San Francisco, 1996.

[3] Freund Y. & Schapire, R. E., *A Short Introduction to Boosting, Journal of Japanese Society for Artificial Intelligence*,14(5): 771-780, September, 1999.

[4] Fung M. G. & Mangasarian, O. L., *A Feature Selection Newton Method for Support Vector Machine Classification, Computacional Optimization and Aplications*,28, 185-202, 2004.

[5] Merz C. J. & Murphy P.M., *UCI Machine Learning Repository*, www.ics.uci.edu/ mlearn/MLRepository.html.

[6] UCL Neural Network Group *ELENA project, ftp.dice.ucl.ac.be at directory, pub/neural_nets/ELENA/databases.*

[7] Rätsch, G. & Schölkopf, B. & Smola A.J. & Mika S. & Onoda T. & Müller K.-R., *Robust Ensemble Learning, In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schurmans editors, Advances in Large Margin Classifiers*, 207-219. MIT Press, Cambrigde, MA, 2000a.

[8] Dietterich, T. G. *An Experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization, Machine Learning*,1-22,1998.

[9] Skurichina, M.& Duin, R.P.W. *Boosting in Linear Discriminant Analysis, Multiple Classifier Systems* (Proc. First International Workshop, MCS 2000, Cagliari, Italy, June 2000), J. Kittler, F. Roli (eds.), Lecture Notes in Computer Science, vol. 1857, Springer-Verlag, Berlin, pp. 190-199, 2000

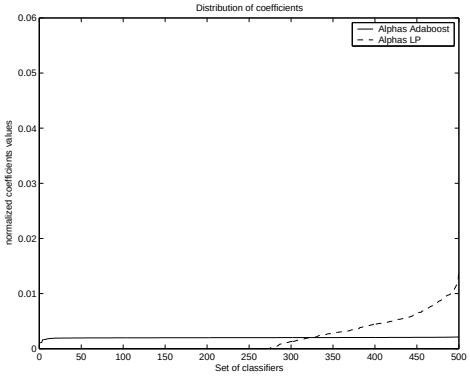

Figure 5: Dataset Phoneme. Alphas created by Adaboost boosting decision trees and LP normalized and arranged in increasing order

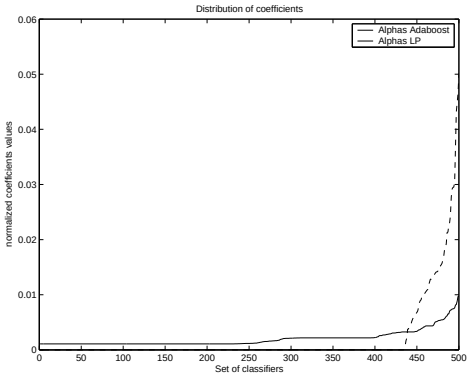

Figure 6: Dataset Satimage. Alphas created by Adaboost boosting linear discriminant and LP normalized and arranged in increasing order