World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:12, 2007

Ensembling Classifiers – An Application to Image Data Classification from Cherenkov Telescope Experiment

Praveen Boinee, Alessandro De Angelis, and Gian Luca Foresti

***Abstract*—**Ensemble learning algorithms such as AdaBoost and Bagging have been in active research and shown improvements in classification results for several benchmarking data sets with mainly decision trees as their base classifiers. In this paper we experiment to apply these Meta learning techniques with classifiers such as random forests, neural networks and support vector machines. The data sets are from MAGIC, a Cherenkov telescope experiment. The task is to classify gamma signals from overwhelmingly hadron and muon signals representing a rare class classification problem. We compare the individual classifiers with their ensemble counterparts and discuss the results. WEKA a wonderful tool for machine learning has been used for making the experiments.

***Keywords*—**Ensembles, WEKA, Neural networks [NN], Support Vector Machines [SVM], Random Forests [RF].

I. Problem Domain

MAGIC [1] is the Cherenkov telescope used to detect the gamma rays from the outer universe. It is designed to provide vital information on several established gamma-ray sources, like Active Galactic Nuclei, Supernova Remnants, Gamma Ray Bursts and Pulsars.

It collects gamma ray events (in little quantities) along with many other particle events represented as images shown in the (Fig.1). The pixels making up the image can be converted to some set of image parameters also called as hillas parameters by various image processing and feature extraction techniques [2], which statistically allow a separation of events. A gamma ray signal defines an ellipse in the camera plane of the telescope where as the other showers make up an error ellipse plane (Fig.1). The data sets used in the experiment contain 10 image parameters. Due to atmospheric radiations, the ground based telescope collects overwhelming events of hadrons and muons also called as background. To understand the gamma ray sources, it is an important task for separating gammas from other particles. There is only a weak discrimination between the gamma and background events, making the data an excellent proving ground for the classification techniques [2]. Things are further complicated by added noise in the data collection by the hardware bias in the telescope.

As the sources of high-energy gammas are few and their signal comparatively weak, creating a case for the rare class classification problem. It will be a daunting task to separate the gamma signals from these overwhelming sea of background signals, both having very similar characteristics.. Classification of signals plays a vital role for making the astronomical analysis of gamma ray objects, and any small improvements in the classification accuracy will be significant in these analysis tasks.

Automated classification of vector data into existing groups is a well defined problem in machine learning. The task is to construct a classifier which associates every data vector to one of the groups such that misclassifications are minimized.

Previous classification studies performed on MAGIC data sets shown that random forest, neural networks performed better than other classifiers [2]. We used WEKA a machine learning framework for making the experiments. WEKA [8] uses powerful object oriented programming techniques for making ensembles. It provides facilities for using various machine learning algorithms as base classifiers for ensembles. In this paper we experiment with making the ensembles of random forests, neural networks and support vector machines and study their performance results.

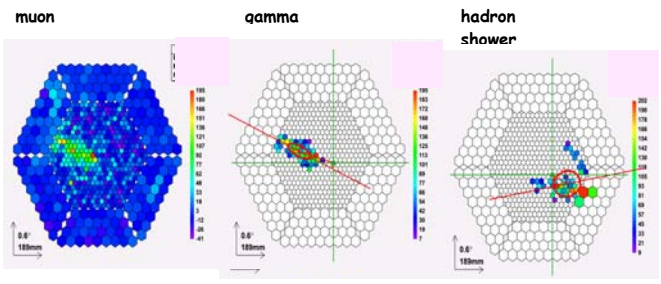

Fig. 1 Signal images from MAGIC Telescope generated after various pre-processing and image cleaning techniques

Manuscript received July 15, 2005.

Praveen Boinee is the PhD Student in Computer Science in Udine University, Udine, 33100, Italy (phone: 0039-0432-558231; e-mail: boinee@fisica.uniud.it).

Alessandro De Angelis is the Professor in Experimental and Computational Physics at Udine University, Udine, 33100, Italy (Phone: 0039-0432-558239; email: deangelis@fisica.uniud.it).

Gian Luca Foresti is the Professor in Computer Science at Udine University, Udine, Italy (e-mail: foresti@dimi.uniud.it)

II. Ensembles of Classifiers

Ensemble learning algorithms combine "base classifiers" to predict the label for the new data points. Experiments on several benchmark data sets and real world data sets showed an improved classification results from these techniques. In this paper we concentrate on 2 ensembles techniques AdaBoost and Bagging. We experiment with using neural networks [back propagation neural nets], random forests and support vector machines as base classifiers.

The training data set is a collection of the data points associated with labels. The data points, usually a vector of features (*x*), and the labels *y*, are bounded by an underlying function *f* such that *y = f(x)* for each training data point (x,y) [3]. Machine learning algorithms search for a best possible hypothesis *h* to *f* that can be applied to assign labels to new x values. Ensemble learning algorithms construct a set of hypothesis $\{h_1, h_2, ..., h_k\}$ and construct a voted classifier $H^*(x) = T(h_1(x), h_2(x), ..., h_n(x))$ to predict the label of new data points, where T a criterion to combine the hypothesis.

A. Bagging

Bagging is a statistical re-sample and combine technique [4] based on bootstrapping and aggregating techniques. The basic idea of bagging is to use bootstrap re-sampling to generate multiple versions of a predictor which, when combined, should perform better than a single predictor built to solve the same problem. Bootstrapping is based on random sampling with replacement. Therefore, taking a bootstrap i.e., (random selection with replacement) of the training set X, one can sometimes avoid or get less misleading training objects in the bootstrap training set. Consequently, a classifier constructed on such a training set may have a better performance. Aggregating actually means combining classifiers [5]. Often a combined classifier gives better results than individual classifiers, because of combining the advantages of the individual classifiers in the final solution. Therefore, bagging might be helpful to build a better classifier on training sample sets with misleaders.

On average, when taking a bootstrap sample of the training set, approximately 37% of the objects are not presented in the bootstrap sample, meaning that possible 'outliers' in the training set sometimes do not show up in the bootstrap sample. Thus, better classifiers (with a smaller apparent error – classification error on the training data set) may be obtained by the bootstrap sample than by the original training set. These classifiers will be presented 'sharper' in the apparent error than those obtained on the training sets with outliers. Therefore, they will be more decisive than other bootstrap versions in the final judgment. Thus, aggregating classifiers in bagging can sometimes give a better performance than individual classifiers.

B. AdaBoost

Boosting works by repeatedly running a learning algorithm on various distributions over the training data, and then combining the classifiers produced by the learner into the single composite classifier [6]. The boosting algorithm takes as input a training set of m examples $S = ((x_1, y_1), ..., (x_m, y_m))$ where each instance x_i is a vector of attributes drawn from the input space X and y_i belonging to finite label set Y, is the class label associated with x_i. In boosting classifiers and training sets are obtained in a strictly deterministic way. Both training sets and classifiers are obtained sequentially in the algorithm, in contrast to bagging, where training sets and classifiers are obtained randomly and independently from the previous step of the algorithm. At each step of the boosting, training data are reweighed in such a way that incorrectly classified objects get larger weights in a new modified training set [7]. AdaBoost manipulates the training examples to generate multiple hypotheses. It maintains the probability distribution $p_l(x)$ over the training examples. In each iteration l, it weights the training samples with the probability distribution $p_l(x)$. The learning algorithm is then applied to produce the classifier h_l. The error rate ε_l of this classifier on the training examples is computed and used to adjust the probability distribution on the training examples. The effect of the change in the weights is to place more weight on training examples that were misclassified by h_l and less weight on examples that were correctly classified in the last stage. In subsequent iterations, therefore, AdaBoost tend to construct progressively more difficult learning problems. The final classifier, h_{final}, is constructed by a weighted vote of the individual classifiers $h_1, h_2, ...h_n$. Each classifier is weighted according to its accuracy for the distribution p_l that it was trained on.

C. Dealing with Rare Class Problem

The MAGIC data sets provide with a rare class classification problem. In this paper we used a one night observation of telescope data which contains 90% of hadron signals [negative examples, making a majority class] and only 10 % of gamma signals [positive examples, making a minority class] . Traditional machine learning algorithms may be attracted towards the majority class, thus producing poor predictive accuracy over the minority class. Ensembling techniques have proved to be performing better in the context of the rare classes [12]. In this paper we concentrate on boosting and bagging techniques for addressing this problem.

III. Base Classifiers used in Experiment

The base classifiers are well known in machine learning literature. Here we briefly outline the algorithms.

A. Back-Propagation Neural Network [BPNN]

One fundamental weakness of neural networks is that they are very sensitive to the training data sets (i.e.) small changes in training set and/or parameter selection can cause large changes in performance. They are unstable or exhibit variance. This instability is magnified when real-world systems such as MAGIC data sets are modeled as they contain more noise and dominated by only one class events.

In boosting each t-th neural network uses a different training set at each epoch, by resampling with replacement after each training epoch. After each epoch, a new training set is obtained by sampling from the original training set with probabilities $P_t(i)$. Training continues until a fixed number of pattern presentations has been performed. The training cost that is minimized for a pattern is the i-th one from the original training set is $\frac{1}{2}\sum_j D_t(i,j)\left(z_{ij} - \hat{z}_{ij}\right)^2$ [9]. We used a 10-5-2 architecture [10 input neurons, 5 hidden neurons, 2 output neurons one for gamma and one for background] with learning rate as 0.2 and momentum factor as 0.3. 10 neural nets are used with each neural network trained for 20 iterations.

Bagging can be successively used to overcome the instability of neural networks. The individual neural networks architecture and parameters are same as that used for boosting. Bagged neural network has shown slight improvements in classification compare to that of individual neural nets.

B. Random Forests [RF]

Random forests are one of the most successful ensemble methods that are fast, robust to noise, do not over fit and offers possibilities for explanation and visualization of its output. In the random forest method, a large number of classification trees are grown and combined. Two random elements serve to obtain a random forest, bagging and random split selection. Bagging is done here by sampling multiple times with replacement from the original training data set. Thus in the resulting samples, a certain event may appear several times, and other events not at all. About 2/3rd of the data in the training sample are taken for each bootstrap sample. Random split selection is used in each trees growing process. The tree growing begins with all cases being contained in the root node. The root node is then split by a cut using one of the image parameters, into two successive nodes to achieve a classification by separation of the classes. When using a total of 10 image parameters, three (square root of total, an experimental best value) are chosen randomly (uniformly distributed) from the total 10. The image parameter yielding the smallest Gini-index [10] among these three is used for splitting. Using only one random variable for selection was not sufficient, but using two or four parameters yield a result very similar to what is obtained with three variables.

Subsequently, the same procedure is applied to each branch in turn. The tree growing stops only when all nodes contain pure data, i.e. from one class only. These nodes are then called terminal nodes and receive their class label from the training data. No pruning is performed. Using unpruned trees (in general poor classifiers) requires a reasonably large number of them to be combined. For our data, growing 20 trees turns out to be sufficient: the quality of the classification in terms ROC plot remains unchanged after a certain number of trees has been reached (Figure 2). Combining classifications from the trees is simply done by calculating the arithmetic mean from the 20 classifications of all trees (considered as 0 and 1). The results are verified with WEKA, Breiman original FORTRAN sources for the Random Forests technique. The simple arithmetic mean for combination seems to be adequate when dealing with a reasonably large number of unpruned trees. We also tried weighted combinations of trees using several different estimates of confidences of the prediction in individual leaves, but this gave better results only for a very small number of trees [less than 10]; which is too small a number to assure convergence in classification error. This fits well with the assumption that a large enough forest is less sensitive to variance than to bias.

Random Forests are themselves proved to be powerful classifiers, and given good results for MAGIC data sets. The ensembles of random forests have further improved the classification accuracy. For both bagging and boosting, the random forests of 20 trees, each constructed while considering four random features have been used as base classifiers and iterated for 10 times. Further increase in trees and iterations does not shown any improvements.

C. Support Vector Machines [SVM]

SVM's are becoming increasingly popular in the machine learning and computer vision communities. Training a Support Vector Machine (SVM) requires the solution of a very large quadratic programming (QP) optimization problem. In this study we use a variant of SVM for fast training using sequential minimal optimization [11]. SMO breaks this large QP problem into a series of smallest possible QP problems avoiding large matrix computation. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets. SMO's computation time is dominated by SVM evaluation; hence SMO is fastest for linear SVMs and sparse data sets. SVM ensembles can improve the limited classification performance of the SVM. In bagging, each individual SVM is trained independently, using randomly chosen training samples via a bootstrap technique. In boosting, each individual SVM is trained using training samples chosen according to the sample's probability distribution, which is updated in proportion to the degree of error of the sample.

TABLE I
COMPARATIVE STUDY OF 4 CLASSIFIERS AND THEIR ENSEMBLES

Model Name	Classification Accuracy[%]	ROC Area	Mean Error rate	Time taken [In seconds]
Random Forests [RF]	78.9085	0.818	0.2748	120
Bagged RF	81.2461	0.8753	0.276	495
Boosted RF	79.9527	0.8481	0.2006	518
Back propagation Neural net [BPNN]	79.28	0.8651	0.2644	158
Bagged BPNN	80.1893	0.8666	0.2888	513
Boosted BPNN	81.7508	0.8671	0.2564	547
C5.0 Classification trees	76.9716.	0.8051	0.2941	98
Bagged C5.0	80.789	0.8582	0.2794	483
Boosted C5.0	77.1707	0.8309	0.2291	503
Support vector machines [SVM]	68.5016	0.5842	0.315	189
Bagged SVM	68.99	0.6028	0.3105	543
Boosted SVM	69.3691	0.7529	0.3739	614

IV. COMPARATIVE STUDY OF CLASSIFICATION RESULTS

Table I shows the classification results for Random forests (RF), Back propagation neural networks [BPNN], Classification Trees and Support vector machines [SVM] along with their bagged and boosted ensembles. Though the ensembles take much time than single classifiers they produce improved classification results. We saw significant improvements with bagged random forests. Adaboosted random forests also showed improved classification results though they are inferior to that of bagged ones. In neural network category, Boosted neural network has shown improved accuracies with bagged MLP standing second. Support vector machines have shown inferior results compare to other classifiers, supporting the results from [2]. The ensembles of SVM shown improvements in the results but less accuracies compare to that of RF or neural networks.

The statistical measures used in the comparative study are the following

- **ROC curves**: A Receiver Operating Characteristic (ROC) curve summarizes the performance of a two-class classifier across the range of possible thresholds. It is recommended for comparing classifiers, as it does not merely summarize performance at a single arbitrarily selected decision threshold, but across all possible decision thresholds. It plots the sensitivity (class two true positives) versus one minus the specificity (class one false negatives). An ideal classifier hugs the left side and top side of the graph, and the area under the curve is 1.0.

- **Classification accuracy:** The classification accuracy for a classifier is defined as percentage of number of correctly classified samples to the total number of samples.

- **Mean absolute Error:** It is the ratio of Incorrectly Classified Instances to that of Total Number of Instances.

V. DISCUSSION

Figure 2.1, 2.2 make a comparative chart of ROC for bagged and boosted ensembles. The ensembles have shown classification improvements for all algorithms. Though they take more time for training, it is important for MAGIC experiment as analysis of gamma ray events are heavily dependent on classification accuracies.

In overall rankings Boosted neural networks, Bagged random forests have shown superior results. Bagging has shown good results with tree based classifiers, especially for random forests. Though neural networks benefited from bagging but they got better accuracies with boosting. Support vector machine also shown improvements in accuracies but at the cost of large training times. The future work will be concentrated on introducing more base classifiers and experimenting with other ensemble techniques such as multi-boost, random committee.

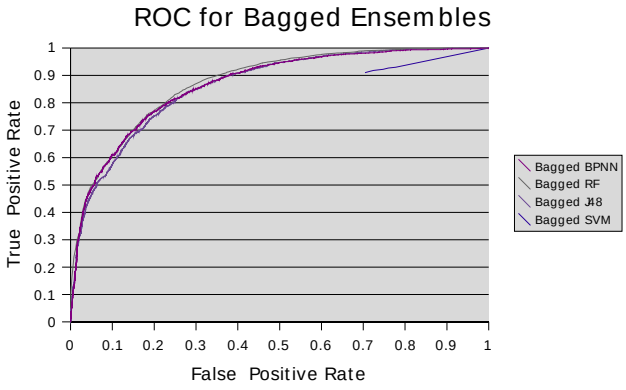

Fig. 2 ROC Comparison for Bagged Ensembles

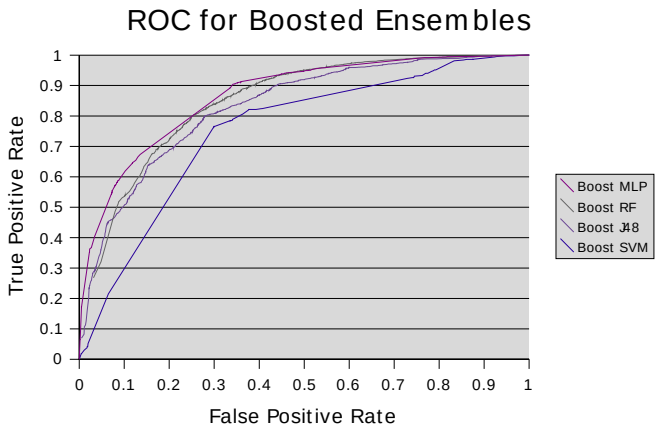

Fig. 3 ROC comparison for Boosted ensembles

REFERENCES

[1] http://wwwmagic.mppmu.mpg.de.
[2] Bock, R.K., et al.: Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope, Nucl. Instr. Methods A516 (2004) 511.
[3] Dietterich, T.G.: Machine Learning. In Nature Encyclopedia of Cognitive Science, Macmillan, London (2003).
[4] Breiman, L.: Bagging Predictors: Machine Learning (1996) 24:123-140.
[5] Carney, J., Cunningham, P. The NeuralBAG algorithm: optimizing generalization performance in Bagged Neural Networks. In: Verleysen, M. (eds.): Proceedings of the 7th European Symposium on Artificial Neural Networks (1999), pp. 3540.
[6] Freund, Y., Schapire, RE.: Experiments with a new boosting algorithm. In Proceedings 13th International Conference on Machine Learning (1996) 148–156.
[7] Skurichina, M., Duin, R.P.W.: Bagging, Boosting and the Random Subspace Method for Linear Classifiers, Vol. 5, no. 2, Pattern Analysis and Applications (2002) 121-135.
[8] Ian H. Witten and Eibe Frank: Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
[9] Schwenk, H., Bengio, Y.: Boosting Neural Networks, Vol. 12, Issue 8, Neural Computation (2000).
[10] Breiman, L.: Random Forests Technical Report, University of California, 2001.
[11] Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Scholkopf, B., Burges, C., Smola, A. (eds.): Advances in Kernel Methods - Support Vector Learning, MIT Press (1998).
[12] Rong, Y., Yan, L., Rong, J., Hauptmann, A.: On predicting rare classes with SVM ensembles in scene classification: Proceedings of Acoustics, Speech, and Signal Processing, (ICASSP'03) 2003 IEEE International Conference.