An Empirical Comparison of Hierarchical vs. Two-level Approaches to Multiclass Problems

Suju Rajan and Joydeep Ghosh

Laboratory of Artificial Neural Systems,
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin TX-78712, USA.
{rsuju, ghosh}@lans.ece.utexas.edu

Abstract. The ECOC framework provides a powerful and popular method for solving multiclass problems using a multitude of binary classifiers. We had recently introduced the Binary Hierarchical Classifier (BHC) architecture that addresses multiclass classification problems using a set of binary classifiers arranged as a tree. Unlike ECOCs, the BHC groups classes according to their natural affinities in order to make each binary problem easier. However it cannot exploit the powerful error correcting properties of an ECOC ensemble that can provide good results even when individual classifiers are weak. Using well-tuned SVMs as the base classifiers, we provide a comparison of these two diverse approaches using a variety of datasets. The results show that while there is no clear advantage to either technique in terms of classification accuracy, BHCs typically achieve this performance using fewer classifiers, and have the added advantage of automatically generating a hierarchy of classes. Such hierarchies often provide a valuable tool for extracting domain knowledge, and achieve better results when coarser granularity of the output space is acceptable.

1 Introduction

Classifiers such as k-nearest neighbors and multi-layered perceptrons can directly deal with multi-class problems. In difficult pattern recognition problems involving a large number of classes however, it has been often observed that obtaining a classifier that discriminates between two (groups of) classes is much easier than one that simultaneously distinguishes among all classes. This has prompted substantial research in using a collection of binary classifiers to address single multi-class problems. Further interest in this area has arisen from the popularity of classifiers such as SVMs [1] whose native formulation is for binary classification problems. While several extensions to multi-class SVMs have been proposed, a careful study in [2] showed that none of these approaches were superior to using a set of binary SVMs in an "All Pairs" framework.

The One-Vs-All method [3], the Round Robin Method [4] or the All Pairs method [5] (also known as pairwise method), and the Error Correcting approaches [6] [7] [8] are some of the techniques proposed for solving multiclass problem by decomposing the output space. All these different methods can be unified into a common framework in which the output space is represented by a binary code matrix which, differs

according to the technique used [7] [9]. All of these can be considered as two-level approaches with the second level providing some relatively simple mechanism to output the final class label based on the decisions obtained from a set of binary "base" classifiers at the first level. Another common characteristic of these methods is that they do not take into consideration the affinities among the individual classes. As a result, some of the groupings might result in complex decision boundaries. In particular, this suggests the need to use powerful base classifiers in the ECOC framework (decision stumps will not do!). The performance of ECOC methods also hinges heavily on the choice of the code matrix, and though multiple solutions have been proposed, none of them are guaranteed to produce the optimal matrix for a problem at hand.

The BHC [10] is a multiclassifier system that was primarily developed as a solution to the problem of dealing with multiclass hyperspectral data. Like the multiclass systems mentioned above, the BHC decomposes a C class problem into C-1 binary meta-class problems. However the grouping of the individual classes into meta-classes is determined by the class distributions. A recent work, that compared the performance of BHC against that of an ECOC system showed the superior performance of the BHC method on real-world hyperspectral data. [11]. However, simple Bayesian classifiers were used for the binary decisions in both architectures. Given that the BHC besides yielding valuable domain knowledge also resolves the problem of having to come up with an optimal code matrix, we decided to evaluate the performance of the BHC against ECOC methods on a wider range of standard datasets. SVM being a popular and powerful binary classifier was used as a constituent classifier for both the systems. Our experiments show that the performance of the BHC is comparable to that of ECOC classifiers while remaining robust for small training sets. We also show that besides using far lesser number of classifiers in most cases, the binary trees produced by the BHC are consistent with those a human expert might have constructed when given just the class labels.

2 Background

2.1 Binary Hierarchical Classifier

The Binary Hierarchical Classifier (BHC) [10] is a method of recursively decomposing a multiclass (C-classes) problem into C-1 two meta-class problems, resulting in C-1 classifiers arranged as a binary tree. The given set of classes is first partitioned into two disjoint meta-classes and each meta-class thus obtained is partitioned recursively until it contains only one of the original classes. The tree thus has a number of leaf nodes equal to the number of classes in the output space. The partitioning of a parent set of classes into two meta-classes is not arbitrary, but is obtained through a deterministic annealing process that encourages similar classes to remain in the same partition [12]. Thus as a direct consequence of the BHC algorithm, classes that are similar to each other in the input feature space are lumped into the same meta-class higher up in the tree. Interested readers are referred to [10] for details of the algorithm. Also, note that the BHC is an example of a coarse-to-fine strategy, which has seen several

application-specific successes and for which solid theoretical underpinnings are beginning to emerge [17].

The BHC algorithm offers a lot of flexibility in designing the classifier system. For instance, one can replace the Bayesian classifier at the internal nodes of the generated BHC tree with stronger binary classifiers such as SVM [1]. Moreover, different feature selection methods can be used at each node that is specific to the domain of the input data. It has to be kept in mind that the maximum number of classifiers required in a BHC system is C-1, which is an advantage against other methods such as All-Pairs in which the number of classifiers is quadratic in the cardinality of the output space. Further since the BHC essentially clusters the classes, it yields additional information about the inherent relationships among the different classes, such information can be leveraged in creating classifiers with varying levels of granularity or in forming simple rules or for identifying those features that help distinguish between groups of classes

2.2 ECOC Classifiers

The ECOC method of combining binary classifiers is based on the framework used in [13] for the NETalk system. In this method the training data of each class was associated with a unique binary string of length n. The different bit positions in the binary string represent a particular independent feature, the presence or absence of which helps differentiate one class from the other. During training n binary functions are learnt, one for each bit position. New data samples were then classified by each of the n classifiers, the combined output of which gives rise to a n-bit binary string. The bit string thus obtained is then compared with the representative bit strings of each class, and the class with the closest distance measure to the bit string of the data sample is assumed to have generated it.

This idea was extended in [6] where instead of generating bit strings that in some sense represent the input space; a matrix of code vectors having good error correcting properties was used. Thus if d is the Hamming distance between any two code vectors in the matrix, then the error correcting properties of the code can correct up to $|(d-1)/2|$ errors. Hence if we train a set of classifiers, one for each bit position then we can obtain the right class label even when there are $|(d-1)/2|$ errors. These error-correcting codes will be successful only when the errors made by the individual classifiers are independent [14]. The ECOC matrices used in [6] were predefined and did not take into account any other factor inherent in the data other than the number of classes. An error–correcting matrix is considered to be a good one if

- The codewords are well-separated in Hamming distance
- The columns of the code matrix should be uncorrelated which translates into having a large Hamming distance between pairs of columns, and also between each column and the complements of the rest. Binary classifiers do not differentiate between learning a particular bit string or its complement. Hence even though complementary columns have the maximum Hamming distance care should be taken to not include either identical columns or complementary columns in the code matrix

Dietterich et al. [6], then use a simple set of rules to form the code matrices depending on the cardinality of the output space *k*.

- Exhaustive codes of length 2^{k-1}-1 when $k<=7$
- Column selection from exhaustive codes when $8<=k<=11$
- For $k>11$ a method based on randomized hill climbing or using BCH [15] codes are used to form the code matrices

The above methods however don't guarantee the best possible code matrix for a given problem and involve some heuristics in choosing a good matrix among the available choices. Finding the optimal code matrix for is still an open research question though many alternatives have been suggested. Allwein et al. [7], use the confidence of predictions and perform decoding based on a loss function instead of the Hamming distance. The performances of different choices of code matrices were also evaluated. Some of the code matrices were

- A complete code matrix which is a framework that unifies One-Vs.-All, Error Correcting codes and All-pairs classification
- A dense random code with $|10\log2(k)|$ columns for a *k*-class problem. The code matrix was selected by generating 10,000 matrices in which each column had a uniform distribution of {+1 and –1} and then choosing that matrix which had the largest inter-row Hamming distance
- A sparse random code with $|15\log2(k)|$ columns in which the elements were 0 with probability 1 /2 and 1 or –1 with probability 1 /4 each. The code matrix with the largest hamming distance was again chosen as the best one.

Crammer et al. [8] attempt to find a code matrix that is problem dependent by trying to learn a matrix with real-valued entries that has a good performance on the training data. The One-Vs-All [3] and the Pairwise classification techniques [4] [5] can be considered as special cases of representing the code matrix. More sophisticated methods such as recursive ECOC have also been proposed [18].

Though there are numerous methods of combining the different binary classifiers in the ECOC framework, no one method has till date claimed superior performance over the rest in terms of the error rates on the test data. The code matrices that are suggested are based on a trial and error method and the ensemble of classifiers finally obtained are not easily interpretable and are not guaranteed to be the best for a given problem. In fact, a careful evaluation of the different multiclassifier methods [9] shows comparable performance of the different classifiers and hence suggests the use of the One-Vs.-All method (with very well tuned and powerful base classifiers) as being the most intuitive and simplest in terms of complexity.

3 Experiments

Table 1 summarizes the datasets used in our experimental evaluation. All the datasets are publicly available at the UCI Machine Learning Repository. These datasets were chosen as they all had numeric attributes and have also been used by others to evalu-

ate ECOC systems. We tested the performance of BHC with base Bayesian Classifiers, BHC with base SVM classifiers and ECOC with base SVM classifiers.

Table 1. Datasets.

No.	Name	# Train	# Test	# Classes	# Attributes
1.	Glass	214	5C.V	7	9
2.	Yeast	1484	5 C.V	11	9
3.	Satimage	4435	2000	6	36
4.	Segment	2310	5C.V	7	16
5.	Page-Blocks	5473	5C.V	5	10
6.	Pendigits	7494	3498	10	16
7.	Optdigits	3823	1797	10	64
8.	Letter	16000	4000	26	16
9.	Vowel	528	462	11	12

For those datasets with individual test sets we used the original test/train split. On those datasets with no independent test set, we performed five-fold cross validation. None of the datasets had any missing values. The training data was always normalized to have a zero mean and unit variance and the test set was also normalized with the mean and variance of the training data. For the BHC with constituent Bayesian Classifiers the in-built feature extraction method based on the fisher discriminant was applied at each of the internal nodes. This BHC method did not require the setting of any parameters and hence the training set was used as is.

The BHC algorithm was then modified to accommodate the SVM classifiers. The SVM classifiers were implemented in Matlab using the toolbox provided by [16]. Each training set was split into a 90% training set and a 10% validation set which was then used to tune the parameters of the SVM. SVMs with polynomial and RBF kernels were experimented with and in all cases RBF kernels yielded better performance on the validation data. To tune the kernel width, it was first set at 1 and then was increased or decreased until there was no improvement on the validation set. The kernel width was then set at the best possible value within the observed range. For both the BHC-SVM and the ECOC-SVM methods parameter tuning was done individually for each validation set. BHC-SVM also used a simple Forward Feature Selection algorithm at each node, as the nodes towards the leaves tend to have far less training data than the nodes near the root. Feature selection was performed only when the ratio of the number of training samples to the input dimensionality was less than 5 [11]. For the ECOC-SVM method, the kernel parameter was tuned as before. The code matrices were generated by

- The Exhaustive method in [6] when the number of classes $k<7$
- The Dense random code method of [7] when $7<k<11$
- Using the BCH code matrix when $k>11$

The BCH-like code matrix for the Letter dataset was also obtained from [16].

Table 2 shows the absolute error rates of the three different classifier schemes. The number in the parentheses under an accuracy rate is the standard deviation of the percentage accuracy over the five cross-validation runs, and an asterisk indicates that the results are statistically significantly different from that of the BHC-SVM at the 90% confidence level

Table 2. Percentage Accuracy across the different datasets.

No.	Name	BHC	BHC-SVM	ECOC-SVM	# Of classifiers BHC / ECOC
1.	Glass	59.2 (7.79)	66.59 (6.2)	63.7 (6.14)	6 / 31
2.	Yeast	43.56* (3.25)	56.91 (3.28)	53.81 (3.7)	10 / 34
3.	Satimage	83.85*	91.45	91.4	5 / 15
4.	Segment	43.9* (10.1)	89.94 (1.34)	89.17 (0.90)	6 / 31
5.	Page-Blocks	82.64* (5.66)	94.5 (2.05)	94.8 (1.9)	4 / 7
6.	Pendigits	89.30*	97.9	98.45*	9 / 33
7.	Optdigits	93.9*	96.6	96.99	9 / 33
8.	Letter	73.7* (1.13)	96.23 (0.60)	96.83 (0.32)	25 / 15
9.	Vowel	47.4*	61.03	51.29	10 / 34

From the above table it can be seen that the use of SVM as the internal classifier improves the performance of the BHC significantly. The BHC-SVM also has percentage accuracies that are comparable if not better than that of ECOC-SVM in most of the cases. Moreover, the BHC involved using fewer classifiers in almost all cases except that of the Letter dataset for which a BCH-like code matrix of length 15 was used. It has been remarked in [6] that the BCH codes have a few practical drawbacks and human intervention is necessary to generate good shortened BCH codes. All other methods of binary classifier generation except for the One-Vs-All method use far more classifiers than the BHC while showing similar performance on the UCI datasets [4] [6] [7] [9].

As mentioned earlier, since the BHC performs a clustering of the output classes based on their separability, the algorithm tends to generate hierarchies of classes that appeal intuitively. A few sample trees for the datasets are included in the Appendix.

The numbers at the internal nodes of the trees are just a way of representing the classes at these nodes. It can be seen that the BHC does indeed generate meaningful class hierarchies. For instance in the Satimage tree, the Vegetation/Foliage classes form one meta-class while the different soil classes form another. In some cases it might be sufficient to just know whether an incoming data sample is one of vegetation or soil. A single classifier at the root node is capable of performing this task whereas in, say, the One-Vs.-All method one would have to evaluate six different binary classifiers to obtain the same result. Feature selection techniques can also be used to generate simple classifiers that can distinguish between the meta-classes and one can identify those features that help distinguish classes at a coarser level of granularity. Such knowledge extraction is not readily available in the other multiclassifier systems.

A strength of ECOC based classifiers is that it can compensate for weak base classifiers. So one should also examine results based on much less amounts of training data to see if ECOCs have superior properties in such situations. Fig 1. and Fig 2. show typical BHC-SVM and ECOC-SVM training curves obtained for two datasets. The fact that the BHC curves are still comparable to the ECOC-SVM based ones even at very low percentages of training data is noteworthy.

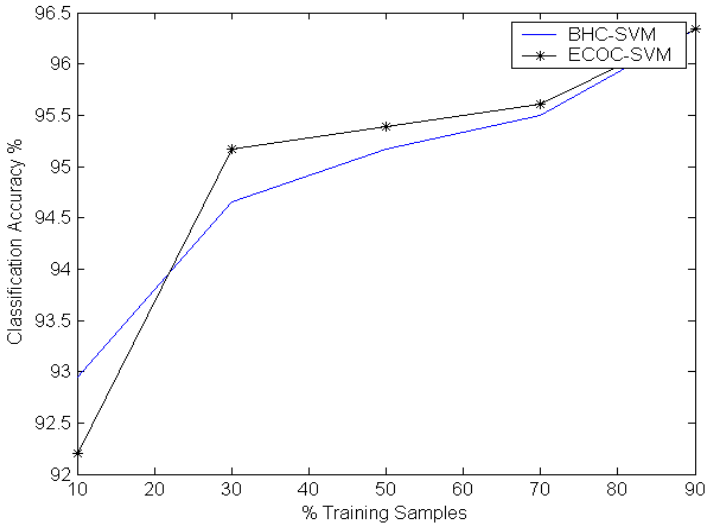


Fig. 1. Training curves for the Page-Blocks Dataset

4 Conclusions

The empirical studies presented in this paper show that the Binary Hierarchical Classifier is a very viable approach to multiclass problems. As compared to ECOC, the

BHC offers the added advantages of not requiring an optimal code matrix, while using far fewer classifiers and automatically generating class hierarchies. While independence of the (meta)-classes is desired for success of the ECOC schemes, the BHC exploits the very dependence to generate a robust and scalable classifier with several beneficial properties

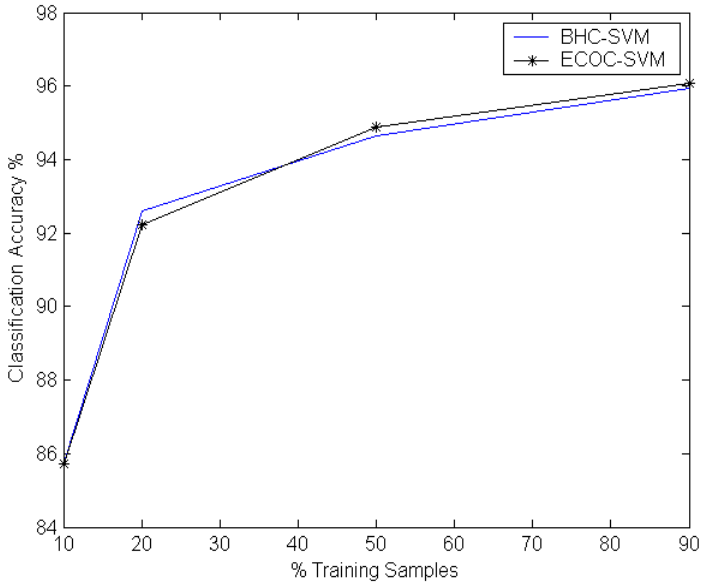


Fig. 2. Training curves for the Letters Dataset

References

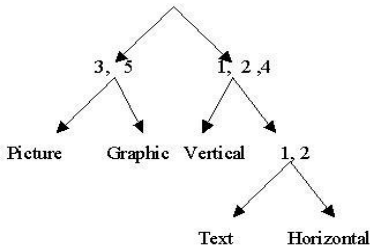
1. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, USA, 1995
2. Hsu, C., Lin, C.: A Comparison of Methods for Multiclass Support Vector Machines, *IEEE Transactions on Neural Networks*, 13 (2002), 415-425
3. Nilsson, N. J.: *Learning machines*, McGraw-Hill, New York, 1965
4. Fürnkranz, J.: Round Robin Classification, *Journal of Machine Learning Research*, 2(Mar), (2002), 721-747
5. Hastie, T., Tibshirani, R.: Classification by Pairwise Coupling, *Advances in Neural Information Processing Systems*, Vol.10, The MIT Press, (1998)
6. Dietterich, T. G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes, *Journal of Artificial Intelligence Research*, Vol.2, (1995), 263-286
7. Allwein, E. L., Schapire, R. E., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, (2000), 9-16
8. Crammer, K., Singer, Y.: On the Learnability and Design of Output Codes for Multiclass Problems, *Computational Learning Theory*, (2000), 35-46

9. Rifkin, R., Klautau, A.: In Defense of One-Vs-All Classification, *Journal of Machine Learning Research*, 5(Jan), (2004), 101-141
10. Kumar, S., Ghosh, J., Crawford, M. M.: Hierarchical Fusion of Multiple Classifiers for Hyperspectral Data Analysis, *Pattern Analysis and Applications*, spl. Issue on Fusion of Multiple Classifiers Vol. 5, No. 2, (2002), 210-220
11. Morgan, T. J., Henneguelle, A., Ham, J., Ghosh, J., Crawford, M.M.: Adaptive Feature Spaces for Land Cover Classification with Limited Ground Truth Data, *International. Journal. of Pattern Recognition and Artificial Intelligence*, J. Kittler and F. Roli (Eds.),(To appear 2004)
12. Kumar, S., Ghosh, J.: GAMLS: A Generalized framework for Associative Modular Learning Systems, *Application and Science of Computational Intelligence II*, SPIE Vol. 3722, (1999), 24-35
13. Sejnowski, T. J., Rosenberg, C. R.: Parallel Networks that learn to pronounce English text, *Complex Systems*, 1, (1987), 145-168
14. Kong, E. B., Dietterich, T. G.: Error-Correcting Output Coding Corrects Bias and Variance, *International Conference on Machine Learning*, (1995), 313-321
15. Bose, R. C., Ray-Chauduri, D. K.: On a Class of Error Correcting Binary Group Codes, *Information and Control*, (3), (1960), 68-79
16. http://www.cis.tugraz.at/igi/aschwaig/software.html
17. Kittler, J, Ahmadyfard, A, and Windridge, D: Serial Multiple Classifier Systems Exploiting a Coarse to Fine Output Coding, in *LNCS, Vol 2709, (Proc. MCS 2003 Workshop),* T. Windeatt and F. Roli (Eds), Springer, (2003), 96-104.
18. Tapia, E, Gonzalez, J. C., Garcia-Villalba, J: Good Error Correcting Output Codes for Adaptive Multiclass Learning, in *LNCS, Vol 2709, (Proc. MCS 2003 Workshop),* T. Windeatt and F. Roli (Eds), Springer, (2003), 156-165.

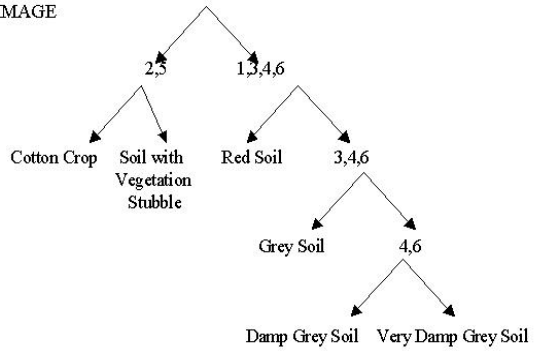
Appendix

A few of the sample trees obtained using the BHC:

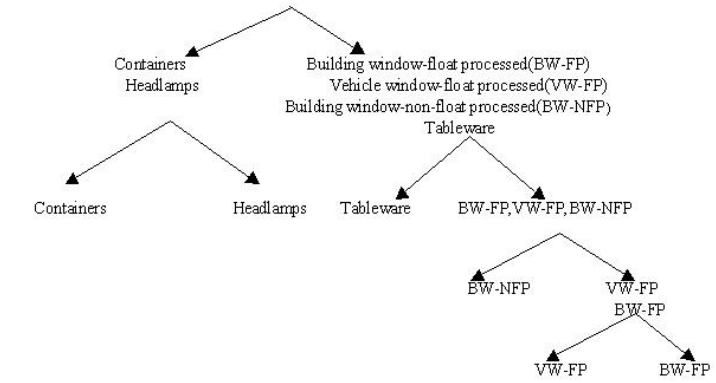
PAGE-BLOCKS



SATIMAGE



GLASS



VOWEL