Tree Pruning for Output Coded Ensembles

T. Windeatt and G. Ardeshir
Centre for Vision, Speech and Signal Processing,
School of Electronics, Computing and Mathematics,
University of Surrey, Guildford, Surrey, GU2 7XH, UK
t.windeatt@surrey.ac.uk

Abstract

Output Coding is a method of converting a multiclass problem into several binary subproblems and gives an ensemble of binary classifiers. Like other ensemble methods, its performance depends on the accuracy and diversity of base classifiers. If a decision tree is chosen as base classifier, the issue of tree pruning needs to be addressed. In this paper we investigate the effect of six methods of pruning on ensembles of trees generated by Error-Correcting Output Code (ECOC). Our results show that Error-Based Pruning outperforms on most datasets but it is better not to prune than to select a single pruning strategy for all datasets.

1. Introduction

In the output coding method of learning, several binary classifiers are trained on two class sub-problems and their outputs are combined to solve the original multiclass problem. To decompose a k class problem these methods use a $k \times B$ binary (typically 0-1) code matrix where B is the number of binary classifiers. One-Per-Class and Distributed Output Code [5] are examples of this kind of decomposition technique. In One-Per-Class a k-dimensional binary target vector represents each one of k classes using a single binary value at the corresponding position, for example $[0, ...0, 1, 0, ...0]$. This leads to a code matrix that has 1's along the diagonal and 0 elsewhere. Distributed Output Code is different in that the k rows are binary strings that are assigned on the basis of meaningful features corresponding to each bit position. For this to provide a suitable decomposition some domain knowledge is required so that each classifier output can be interpreted as a binary feature. The strings are treated as code words, one for each class, and a test pattern is assigned to the class corresponding to the closest code word.

Error-Correcting Output Code (ECOC) uses a method of assigning similar to Distributed Output Code but the code matrix is problem-independent. The motivation for using an error-correcting code comes from the assumption that the learning task can be modelled as a communication problem, in which class information is transmitted over a channel [5]. In this model, errors introduced into the process arise from various sources including the learning algorithm, features and finite training sample. From the transmission channel viewpoint, it is expected that One-Per-Class and Distributed Output Coding matrices would not perform as well as the ECOC matrix, because of inferior error-correcting capability. The output coding concept has been successfully applied to problems in several domains such as cloud classification, text classification, text to speech and face recognition [9]. It has also been shown to improve performance with different kinds of base classifier including decision tree, multi-layer perceptron, SVM and k-nearest-neighbour.

If a decision tree is chosen as base classifier, the issue of tree pruning needs to be addressed. In the past, effort has been directed toward developing effective tree pruning methods (for a review see [6]) in the context of a single tree. The goal of single tree pruning in an ensemble is to produce a simpler tree base classifier which gives improved ensemble performance. The methods used in this study are Error-based Pruning (EBP), Minimum Error Pruning (MEP), Reduced-Error Pruning (REP), Critical Value Pruning (CVP) and Cost-Complexity Pruning (CCP). It is worth noting that tree pruning is an art rather than a science. For example consider EBP, which is the default pruning strategy for C4.5, in which the author emphasises its heuristic nature by noting that the reasoning behind it (Section 3.1) should be taken with a ”grain of salt” [11].

In [5] ECOC with C4.5 showed no significant difference on pruned versus unpruned on six out of eight datasets. There have also been some recent studies of pruning with ensembles using C4.5 for Bagging, Boosting [13] and randomisation [4]. In [4] over thirty-three datasets, significant difference due to pruning (EBP confidence level 10%) was observed in ten datasets for both C4.5 and randomised C4.5, in four datasets for Bagged C4.5 and in none for Boosted

C4.5. In [13], five pruning methods were compared and EBP performed best for both Bagging and Boosting. The empirical comparison of ensemble classifiers in [2] did not use C4.5, but did compare size of bagged pruned and unpruned trees, and concluded that pruning reduces bias while increasing variance.

2. Output Coded Ensembles

The binary $k \times B$ output coding matrix C defines the problem decomposition. A training pattern with target class w_i ($i = 1...k$) is re-labelled either as class Ω_1 or as class Ω_2 depending on the value of C_{ij}. One way of looking at this re-labelling is to consider that for each column the k classes are arranged into two super-classes Ω_1 and Ω_2. A pattern is presented to the B trained binary classifiers and thereby mapped into vector $\mathbf{y} = [y_1, y_2, ...y_B]$, in which y_j is the real-valued output of jth base classifier. The output vector represents super-class probabilities and a pattern is assigned to the class corresponding to the closest row (code word) in C. Hamming Distance was originally chosen to measure closeness, since the code was based on error-correcting principles. However when it was shown that the method produced good probability estimates the decision strategy was modified to the L_1^1 norm or Minkowski distance given by $L_i^1 = \sum\limits_{j=1}^{b} |C_{ij} - y_j|$. The decision strategy assigns class w_i according to $ArgMin_i(L_i^1)$.

When the ECOC technique was developed it was believed that the code matrix should be chosen to satisfy certain properties, including high Hamming Distance between pairs of rows and between pairs of columns [5]. Various output coded ensembles have been proposed, but most code matrices that have been investigated previously are binary and problem-independent, that is pre-designed. Exhaustive codes were proposed when number of classes is between 3 and 7, column selection from exhaustive codes when number is between 8 and 11 and randomised hill climbing and Bose, Ray-Chaudhuri & Hocquenhem (BCH Code) when there are more than 11 classes [5]. Random codes were investigated in [12] for combining Boosting with ECOC, and it was shown that a random code with a near equal column split of labels was theoretically better. Random codes were also shown in [8] to give Bayesian performance if pairs of code words are equidistant. In [14] a random assignment of class to codeword was suggested in order to reduce sensitivity to code word selection.

Overall there is little evidence that any particular binary problem-independent code is superior, and random codes appear to perform as well as others providing B is large enough. Recent developments include proposal of a three-valued code [1] which allows specified classes to be omitted from consideration and [3] in which problem-dependent discrete and continuous codes are investigated.

Attempts have been made to develop theories for ensemble classifiers. The margin concept is useful for finding bounds on generalisation error [1]. In [10] a modified definition for bias and variance from regression theory was proposed and it was shown that ECOC reduces both variance and bias according to their definition. However, different definitions for bias and variance with 0-1 loss function have been suggested, and there are recognised shortcomings with the approach [7].

3. Experiments

3.1. Pruning Techniques

Decision tree pruning is a process in which one or more subtrees of a decision tree are removed to reduce its complexity and make it more comprehensible. There are different types of pruning but post-pruning is more usual. The disadvantage of pre-pruning is that tree growth can be prematurely stopped, since it is based on a difficult estimation of when to stop constructing the tree. A way around this problem is to use a post-pruning method since then the full tree is grown before being retro-spectively pruned. Post-pruning methods remove one or more subtrees and replace them by a leaf or a branch. One class of these algorithms divides the training set into a growing set and a pruning set, which is used to select the best tree. In the case of shortage of training set cross-validation can be used. Another class of post-pruning algorithm uses all the training set for both growing and pruning. However it is then necessary to define an estimate of the true error rate using the training set alone. In this study, the comparison is limited to post-pruning methods and they are briefly discussed in this section, more details and equations for pruning criteria available in [6, 13]. The distinction between the work described here and that in [6] is that pruning methods are compared in the context of tree ensembles rather than single trees.

EBP was developed for use with C4.5 and uses prediction of error rate (earlier version known as Pessimistic Error Pruning) [11]. The assumption is that the error in the set of patterns covered by a leaf of a tree follows a binomial distribution. The upper limit of confidence of the probability of mis-classification can then be calculated from an assumed confidence level (default 25%). The predicted error rate comes from multiplying the upper limit of confidence by the number of patterns covered by the leaf. If the number of predicted errors is less than that for the subtree containing the leaf, then the subtree is replaced with the leaf.

MEP was first introduced using Laplace probability estimates, and later modified to what was referred to as *m-probability estimation*. The parameter m is varied in an attempt to match degree of tree pruning to properties of the learning domain such as noise. To prune a tree at a node, the expected error rates of its children are first determined, and this is called *static error*. *Dynamic error*, defined as the weighted sum of the static error of its children, is then calculated and if static error is greater than dynamic error the node is replaced by the leaf.

REP requires a separate pruning set, and simply replaces each non-leaf node by the best possible branch with respect to error rate over the pruning set. Branch pruning is repeated until there is an increase in the pruning set error rate. The procedure is guaranteed to find the smallest, most accurate subtree with respect to the pruning set.

CVP operates with a variety of node selection measures. The idea is to set a threshold, the *critical value*, which defines the level at which pruning takes place. A non-leaf node is only pruned if the associated selection measures for the node and all its children do not exceed the *critical value*. The full tree is pruned for increasing critical values giving a sequence of trees, and then the best tree is selected on the basis of predictive ability. A number of suggestions were made for finding the best tree in the sequence, the obvious one being to use a separate pruning set as in REP.

CCP produces a sequence of trees by pruning those branches that give lowest increase in error rate per leaf over the training set. The error rate calculation is based on the number of training patterns, the number of leaves in the subtree and the additional number of misclassified examples when the leaf is removed. In order to select the best tree in the sequence, either cross-validation on the training set or a separate pruning set is employed. The selected tree is either the smallest tree with error rate less than minimum observed error rate (CCP0) or less than minimum observed error rate plus one standard error (CCP1).

3.2. Results

Datasets from UCI are used in these experiments and have been randomly split into training set (70%) and test set (30%). The numbers of patterns, classes and features are shown in table 1. We have used C4.5 as base classifier with six methods of pruning given in Section 3.1, plus the unpruned (UNP) case. To generate the ECOC code matrix with 31 columns ($B = 31$) we have used exhaustive and BCH codes as explained in Section 2.

For datasets described in table 1, table 2 shows the mean test error over ten independent runs normalised with respect to the minimum test error over all pruning methods for the specified dataset. Each error rate has been divided by the respective minimum error rate so that the mean effect over all datasets can be assessed. The minimum test errors (%) for the datasets are as follows Anneal/7.5, Audiology/31.3, Car/13.0, Dermatology/3.8, Glass/26.9, Iris/5.1, Letter/9.9, Segmentation/2.1, Soybean-large/11.3, Vehicle/28.4. Also shown in table 2, after the normalised error for each pruning method, is the number of times that the pruning method is significantly different (McNemar's test 5%) from UNP.

In table 2, the mean normalised error (last row) over the ten datasets shows that MEP on average has higher error and UNP has lower error compared with all other pruning methods. However for MEP we used the default value of $m = 2$ and it is likely that the error would improve if m was allowed to vary. There is little difference in the mean normalised error and mean number of significant differences for EBP, CVP, REP, CCP0, CCP1. We can see that UNP performs best for five datasets, EBP for 3 datasets and CVP and CCP0 for one dataset. Ignoring UNP, EBP outperforms other pruning methods for seven data sets (Glass, Iris, Vehicle, Car, Letter, Dermatology, Segmentation), CCP0 for two data sets (Soybean-large, audiology) and CVP for one (audiology).

Overall it appears that if a single problem-independent strategy is to be selected then UNP is likely to give lowest error rate. If pruning is required to simplify the ensemble, any one of EBP, CVP, REP, CCP0, CCP1 should give similar error rate. However, since EBP performs better on seven out of ten datasets it may be that problem-dependent pruning should be considered. From table 2 and from specification of datasets table 1 we can see that EBP performs well with those datasets that have only continuous features. For example, if EBP was selected for datasets with no discrete features (glass, iris, letter, segmentation, vehicle) and UNP was selected for the other datasets the mean normalised error over ten datasets would be 1.025. This suggests that a larger study should be carried out to test the hypothesis that problem-dependent pruning is worthwhile.

4. Conclusion

In this paper we have compared six pruning strategies for an output coded ensemble method that uses error-correcting codes. For the ten datasets tested, it is shown that it is better not to prune rather than select a single problem-independent pruning strategy for all datasets. However there was was some evidence that problem-dependent pruning might be beneficial.

Acknowledgement Gholamreza Ardeshir is grateful to the Ministry of Culture and Higher Education of Iran for its financial support during his Ph.D studies.

References

[1] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multi-class to binary: A unifying approach for margin classifiers. *Machine learning research*, 1:113–141, 2000.

Dataset	EBP	MEP	CVP	REP	CCP0	CCP1	UNP
Anneal	2.55/10	3.19/10	1.00/2	1.19/1	1.09/1	1.26/1	1.04
Audiology	1.31/1	1.31/1	1.02/2	1.10/3	1.00/0	1.01/0	1.17
Car	1.19/5	2.32/10	1.34/10	1.26/6	1.25/7	1.28/7	1.00
Dermatology	1.00/0	1.68/1	1.52/2	1.33/0	1.23/0	1.24/2	1.17
Glass	1.00/0	1.34/2	1.16/3	1.19/3	1.17/4	1.19/2	1.02
Iris	1.00/0	1.13/0	1.52/0	1.52/0	1.48/0	1.48/0	1.22
Letter	1.01/1	5.78/10	1.11/8	1.09/7	1.09/7	1.10/8	1.00
Segmentation	1.01/0	6.88/10	1.70/3	1.59/3	1.62/1	1.58/2	1.00
Soybean-large	1.40/8	3.22/10	1.23/0	1.31/2	1.19/2	1.22/2	1.00
Vehicle	1.02/0	1.22/5	1.07/1	1.05/0	1.06/1	1.04/0	1.00
(Mean)	1.25/2.5	2.80/5.9	1.27/3.1	1.26/2.5	1.23/2.3	1.24/2.4	1.06

Table 2. Mean test error normalised with respect to minimum error over all pruning methods followed by number of significant differences from unpruned (UNP)

Table 1. Specification of Datasets

Name	#patterns	#Class	#Features	
			Cont.	Disc.
Anneal	898	6	9	29
Audiology	200	24		69
Car	1728	4		6
Dermatology	366	6	1	33
Glass	214	6	9	
Iris	150	3	4	
Letter	20000	26	16	
Segmentation	2310	7	19	
Soybean-Large	683	19		35
Vehicle	846	4	18	

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Maching Learning*, 36(1):105–142, 1999.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, to appear.

[4] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[5] T.G. Dietterich and G Bakiri. Solving multi-class learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[6] F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, May 1997.

[7] G. M. James. Variance and bias for general loss functions. *Machine Learning*, to appear.

[8] G. M. James and T. Hastie. The error coding method and PICT's. *Computational and Graphical Statistics*, 7:377–387, 1998.

[9] J. Kittler, R. Ghaderi, T. Windeatt, and G. Matas. Face verification using error correcting output codes. In *Computer Vision and Pattern Recognition CVPR01*, Hawaii, December 2001. IEEE Press.

[10] E.B. Kong and T.G. Diettrich. Error-correcting output coding corrects bias and variance. In *12th Int. Conf. of Machine Learning*, pages 313–321, San Fransisco, 1995. Morgan Kaufmann.

[11] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,California, 1993.

[12] R.E. Schapire. Using output codes to boost multiclass learning problems. In *14th International Conf. on Machine Learning*, pages 313–321. Morgan Kaufman, 1997.

[13] T. Windeatt and G. Ardeshir. An empirical comparison of pruning methods for ensemble classifiers. In *IDA 2001*. Springer-Verlag, Lecture notes in computer science, 2001.

[14] T. Windeatt and R. Ghaderi. Binary labelling and decision level fusion. *Information Fusion*, 2(2):103–112, 2001.