In Proceedings of the 10th European Conference on Machine Learning, April 21-24 1998, Chemnitz, Germany.

Error-Correcting Output Codes for Local Learners

Francesco Ricci[1] and David W. Aha[2]

[1] Istituto per la Ricerca Scientifica e Tecnologica
38050 Povo (TN), Italy
[2] Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Code 5510
Washington, DC 20375 USA

Abstract. Error-correcting output codes (ECOCs) represent classes with a set of output bits, where each bit encodes a binary classification task corresponding to a unique partition of the classes. Algorithms that use ECOCs learn the function corresponding to each bit, and combine them to generate class predictions. ECOCs can reduce both variance and bias errors for multiclass classification tasks when the errors made at the output bits are not correlated. They work well with algorithms that eagerly induce *global* classifiers (e.g., C4.5) but do not assist simple *local* classifiers (e.g., nearest neighbor), which yield correlated predictions across the output bits. We show that the output bit predictions of local learners can be decorrelated by selecting different features for each bit. We present promising empirical results for this combination of ECOCs, nearest neighbor, and feature selection.

1 Introduction

Error-correcting output codes (ECOCs) can help distinguish classes in classification tasks with $m > 2$ classes by encoding error-correcting capabilities in their output representation. This can increase the classification accuracy of *global* learning algorithms [Dietterich and Bakiri, 1995] (e.g., C4.5 [Quinlan, 1993], backpropagation [Rumelhart *et al.*, 1986]).

However, ECOCs do not benefit *local* learning algorithms [Kong and Dietterich, 1995; Bottou and Vapnik, 1992], which predict classifications for a query q based only on information from examples local (i.e., nearby) to q. In Section 1.2, we explain that this limitation occurs because a local learner's predictions are correlated across the output bits.

This paper presents a method that allows ECOCs to work well with local learning algorithms; it uses a feature selection algorithm to reduce the correlation of a local learner's decision boundaries across output bits.

1.1 Error-correcting output codes

Table 1 exemplifies how ECOC encodings differ from two popular output encoding strategies. In each of these output representations, each class $c_i \in C$ is

Table 1. Example Output Representations

Class Name	Output Representation		
	Atomic	Distributed	
		One-Per-Class	ECOC
Earthling	1	100	00000
Martian	2	010	11100
Italian	3	001	00111

assigned a unique *codeword* $s_i = (s_{i1}, \ldots, s_{il})$ of l codeletters (e.g., [Dietterich and Bakiri, 1995]).

The most popular *atomic* strategy sets $l = 1$; it uses a single codeletter to represent class labels. Learning algorithms that use this approach (e.g., C4.5 [Quinlan, 1993]) induce a single concept description that distinguishes all class boundaries.

Conversely, distributed output code strategies set $l > 1$, where each codeletter s_{ij} typically has a binary value (i.e., they are bit strings). This strategy defines l binary functions f_j on the training set, where $f_j(x) = 1$ if x is in class c_i and $s_{ij} = 1$, and $f_j(x) = 0$ otherwise. f_j is also called the j-th *output bit* function; it is defined only by the j-th column of s_{ij} and provides a binary partition of the classes. For a given class c_i and output bit f_j, either f_j maps all examples in c_i to 1 or maps all of them to 0. Each output bit function corresponds to a different learning task. Given a query q, classifiers using distributed output representations generate predictions $\hat{f}_j(q)$ $(1 \leq j \leq l)$ for each output bit, and predict the class c_i whose codeword $(s_{i1}, \ldots, s_{il})$ has minimal (e.g., Hamming) distance from the predicted codeword $(\hat{f}_1(q), \ldots, \hat{f}_l(q))$.

Two types of distributed output representations are *one-per-class* and *error-correcting*. In one-per-class each bit function separates the examples in one class from the remaining examples (i.e., exactly one output bit in a one-per-class codeword has value 1; all others have value 0). Learning algorithms that use one-per-class encodings induce a separate concept description per class [Quinlan, 1993; Aha, 1992], where positive instances of a class c_i are negative instances for all other classes $c_j (i \neq j)$.

The Hamming distance between all one-per-class codewords is 2, which means that even one incorrectly predicted output bit can cause a misclassification. In contrast, ECOCs encode error-correcting capabilities: they are not restricted to having exactly as many classes as output bits, their codewords can have multiple output bits with value 1, and each class's codeword differs, in Hamming distance, from all other class codewords by (at least) a pre-determined amount h. This gives ECOCs an *error-correcting* capability of $\lfloor \frac{h-1}{2} \rfloor$. For example, $h = 3$ for the three ECOC codewords shown in Table 1. Thus, even if one bit's value is incorrectly predicted for a query q the correct class still has the minimum Hamming distance to q, and will thus be predicted.

Dietterich and Bakiri [1995] compared these three output representations on several multiclass tasks. They designed ECOC codewords, which typically require a large number of output bits, to maximize both *row separation* of the matrix s_{ij} (i.e., Hamming distance between codewords) and *column separation* (i.e., Hamming distance between two columns of s_{ij}, and between any one column and the complement of another). This ensures that the *errors* of the output bit predictions are uncorrelated. They reported that ECOCs often significantly increased the classification accuracies for C4.5 [Quinlan, 1993] and networks trained by backpropagation [Rumelhart *et al.*, 1986], although training ECOCs is slow because they must learn l concepts (i.e., one per output bit).

Kong and Dietterich [1995] showed that ECOCs work well with global learning algorithms on multiclass classification tasks because they can reduce both the *variance* and *bias* components of the output bit errors. Variance results from random variation and noise in the training set and from any random behaviors of the learning algorithm itself. ECOCs reduce variance through a voting process [Perrone and Cooper, 1993]: because Hamming distance determines the "winning" prediction (i.e., closest codeword), each output bit prediction corresponds to a vote for classes whose codewords match the predicted value.

Bias instead refers to an algorithm's systematic errors. These can also be reduced by voting, but only when the individual predictions are uncorrelated, such as by averaging the contributions of *different* prediction algorithms (e.g., [Zhang *et al.*, 1992]). Alternatively, the same algorithm can be used multiple times, but it must vote on different subproblems (i.e., using different class decision boundaries) that cause the algorithm to generate different bias errors. When output bits have good column separation, global algorithms like C4.5 can induce different class boundary hypotheses for different output bits.

1.2 Problem and proposed solution

Suppose that a query q is in class i but a classifier CL (without using ECOCs) misclassifies it as k. Suppose also that CL_{ecoc} uses error-correcting output codes. Assume further that s_{ij} and s_{kj} $(j = 1, \ldots, l)$ are the two encodings of classes i and k, respectively. CL_{ecoc} can "correct" CL prediction errors for q if and only if it assigns to q the codeword $(\hat{f}_1(q), \ldots, \hat{f}_l(q))$ and there are more j such that $\hat{f}_j(q) = s_{ij}$ than $\hat{f}_j(q) = s_{kj}$ (i.e., for any class k in $1 \leq k \leq m$).

When CL is a local classifier (e.g., nearest neighbor) q's predicted bit value $\hat{f}_j(q)$ will always be the same as the bit value of its nearest neighbor q'. Therefore if the nearest neighbor misclassifies q then its ECOC variant will also misclassify q. Thus *extending nearest neighbor with ECOCs will not modify its class predictions*. More generally, this holds for other distributed output representations, including one-per-class. Kong and Dietterich have shown that a global classifier (e.g., C4.5) can avoid this problem. That is, C4.5_{ecoc} can predict $\hat{f}_j(q) = s_{ij}$ for some j and $\hat{f}_j(q) = s_{kj}$ for some other j.

Figure 1 exemplifies how ECOCs assist global but not local classifiers. It shows two output bit partitions for three classes c_i $(1 \leq i \leq 3)$ in a two-dimensional

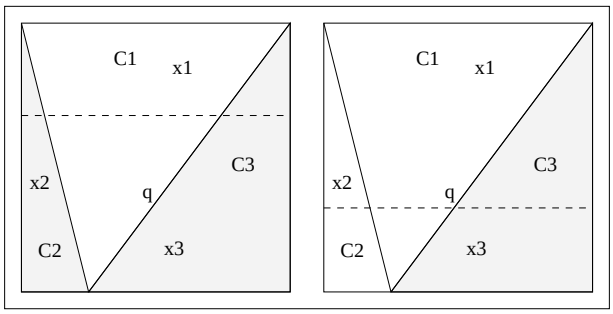

Fig. 1. Class Boundaries (solid lines), Class Groupings (shadings), and a Global Learner's Partition Boundary Hypotheses (dashed lines) for Two Output Bits

continuous space, where each class has a single training instance x_i. The solid lines define the class boundaries, the shadings define each bit's class partition,[3] and the dotted lines show the partition boundary hypotheses that might be induced by a global learner, where the predicted class of each rectangle is the class of its enclosed instances. Thus, the first hypothesis is wrong on q while the second is correct. Therefore, the second bit function can help *correct* the first error, where the underlying assumption is that the majority of hypotheses (bit functions) will correctly predict a query's partition. In contrast, nearest neighbor, when making each bit function hypothesis, will always identify x_3 as q's closest neighbor, and predict x_3's partition.

In this paper, we present empirical evidence that ECOCs can substantially improve the accuracies of the nearest neighbor classifier by incorporating an appropriate feature selection algorithm. Bit-specific decision boundaries can be obtained for nearest neighbor by applying feature selection independently for each bit, which yields a different distance function for each bit. This in turn allows a different nearest neighbor to be selected for each bit's prediction, which decorrelates the output bit errors. The bias/variance decomposition of the error (Section 4) reveals that this accuracy improvement is obtained by drastically reducing bias at the cost of moderately increasing variance.

2 Local Learning with ECOCs

To work well, ECOCs require that the errors for each of the output bits be uncorrelated. Therefore, we extended IB1 [Aha *et al.*, 1991], an implementation of nearest neighbor, to use different features when computing distances for each output bit. Figure 2 summarizes the training algorithm scheme for this extension, named IB1$_{cdwd}$. Using different codeword generation procedures (e.g., atomic, one-per-class, or ecoc), IB1$_{cdwd}$ yields different classifiers. Section 2.1 defines its algorithms for creating ECOC codewords and Section 2.2 describes how it selects features.

[3] The grey area denotes the set of examples that the output bit function maps to 1.

IB1$_{cdwd}(X, C, l, n)$

X: Training set
C: Set of classes
l: Number of bit functions (i.e., codeword length)
n: Max iterations allowed in schemata search
S: Codewords (f_j is the j-th output bit function from S)
F: Feature subsets (one subset per output bit)

1. $S :=$ **create_codewords**(C, l)
2. FOR $j = 1$ *to* l DO:
3. $\quad F_j :=$ **race-schemata**(X, f_j, n)
4. RETURN $\{S, F\}$

Fig. 2. IB1$_{cdwd}$: An Abstraction of IB1 for Alternative Output Representations

2.1 Generating ECOC codewords

The first step in IB1$_{cdwd}$ inputs C, the set of classes, and l, the number of output bits per codeword. The function *create_codewords* generates the set of codewords S, one for each of the m classes $c_i \in C$. We describe in the following how to generate ECOC codewords. Atomic and one-per-class codewords are generated as explained in Section 1.1.

Algorithms for generating ECOC codewords should maximize both row and column separation. For classification tasks where $m \leq 7$, *create_codewords* uses the *exhaustive codes* technique [Dietterich and Bakiri, 1995]. It creates all $2^{m-1} - 1$ possible codewords that are both column and row separated. The resulting codewords have Hamming distance $h = 2^{m-2}$.

When $m > 7$, *create_codewords* employs a variant of Dietterich and Bakiri's [1995] randomized hill climbing algorithm. Our variant extends the original algorithm to search more efficiently: it searches in directions that either increase row or column separation, or assist in escaping from local maxima. It initially generates m boolean codewords of length l, drawn randomly according to a uniform distribution. It then iteratively modifies these codewords, yielding the set of codewords with maximal row and column separation. During each iteration, it selects the two most similar codewords (rows) and the two bit partitions (columns) that are most similar or most dissimilar. However, the columns are selected *only* if the intersection of these rows and columns (a 2×2 boolean matrix a_{ij}) has either of the following properties:

1. If the columns have small Hamming distance, then either the two rows or the two columns of a_{ij} must be equal (i.e., $a_{11} = a_{12}$ and $a_{21} = a_{22}$ or $a_{11} = a_{21}$ and $a_{12} = a_{22}$).
2. If the columns have large Hamming distance, then the two columns of a_{ij} must differ (i.e., $a_{11} \neq a_{12}$ and $a_{21} \neq a_{22}$).

Some of these four values (a_{ij}) are then changed as follows. If all four are equal, then the values on one of the two diagonals are inverted, which increases the

race-schemata(X, f_j, n):

———————————————————

f_j: An output bit function
M: Error statistics matrix $(l \times 2)$
p: Schemata bit string of length d (initially all $\star$'s)
q: Schemata bit string of length d (instantiates p)
e: Prediction error (boolean)

———————————————————

1. FOR $i \leftarrow 1$ TO l DO:
2. $M :=$ initialize(p)
3. DO $(k \leftarrow 1$ TO $n)$:
4. $x :=$ random-select(X)
5. $q :=$ select-constrained-string(p)
6. $e :=$ compute-error(IB1,$x, f_j, q)$
7. $M :=$ update-matching-statistics(M, q, e)
8. UNTIL winner(M) or $(k = n)$
9. $p :=$ update-winning-feature(p, M)
10. RETURN features selected in p

Fig. 3. The Race-Schemata Algorithm for Feature Selection (see Figure 2 for additional documentation)

separation of these rows and columns by two. Otherwise, one of these four values, randomly selected, is inverted. This does not always increase total separation, but it helps to escape from local maximum by exchanging row with column separation (or vice versa). The search process is stopped after a pre-determined maximum number of codeword changes has occurred.

2.2 Selecting features

After creating the codewords, IB1$_{cdwd}$ calls *race-schemata* (Figure 3), which inputs the training set X, a bit function f_j, and the max iteration number n. It returns the subset of features selected by a variant of the schemata racing algorithm [Maron and Moore, 1997] (see [Ricci and Aha, 1997b]). *Race-schemata* searches over the space of schemata strings $p \in P$ of length d, the number of features, whose characters are 0's (feature is not selected), 1's (selected), or $\star$'s (selected with probability 50%). Step 2 begins one race for each (remaining) $\star$ in the schemata p, resetting each feature's error statistics for both 0 and 1. For instance, the first time d races start, and in each race schemata of type "$\star \cdots \star 0 \star \cdots \star$" compete against "$\star \cdots \star 1 \star \cdots \star$", with 0 and 1 in the same position. The interior loop (steps 3–8) iterates until the next race winner is found or n iterations are reached. Each iteration randomly selects a training instance x, selects a binary string q (without $\star$'s) that matches the 0's and 1's currently in p (i.e., it matches at least one of the schemata being raced), computes IB1's error on x for bit function f_j using the features selected in q, and updates the error statistics for all the schemata being raced that match q. The error of a schemata

is computed by averaging the error of matching samples. If a "winning" feature is found (i.e. it is highly unlikely that the error of the other schemata is significantly less) then the interior loop terminates and p is updated by fixing the winner's bit value, changing it from $\star$ to either 0 or 1 (see [Ricci and Aha, 1997b] for more details). *Race-schemata* returns the features whose values in p are 1 (i.e., a selected subset of features).

3 Evaluation

We empirically evaluated our hypothesis that ECOCs can increase IB1's accuracies, when coupled with a feature selection algorithm, in situations where feature selection yields different features for each output bit. Thus, we focused on two independent variables in our experiments: the output representation for IB1 and whether it employed feature selection.

We compared three instances of IB1_{cdwd} using the three output encodings described in Section 1.1. Their names are $\text{IB1}_{\text{atomic}}$, IB1_{opc}, and IB1_{ecoc} for the atomic, one-per-class, and ECOC output representations, respectively. Figure 2 applies to all three algorithms except that create_codewords does not modify the single-codeletter codewords for $\text{IB1}_{\text{atomic}}$ and generates only simple one-per-class codewords for IB1_{opc} (e.g., $10\ldots0, 01\ldots0, \ldots, 00\ldots1$). However, as explained in Section 1.2, distributed output representations will not modify IB1's classification behavior when feature selection is not performed. Thus, rather than reporting results for all six combinations of independent values (i.e., three output representations, either with or without feature selection), we report the results for only four of these combinations: the atomic output representation without feature selection plus all three output representations when using feature selection.

3.1 Data Sets

We selected seven data sets (Table 2) from the UCI Repository [Merz and Murphy, 1996] that have only numeric or boolean features, no missing values, and at least four classes. Even if ECOCs are applicable to general data sets, we avoided those with symbolic features because they often require distinct weighting metrics (e.g., [Stanfill and Waltz, 1986]), which complicates isolating the effects of feature selection. Data sets with fewer than four classes do not greatly benefit from ECOCs. We also used three additional proprietary data sets concerning cloud classification. We will use abbreviations for the data set names.

We conducted a ten-fold cross validation on the data sets with the following exceptions: we used only the usual single training and test set for IS due to its large size, we used only five folds for VO due to its structure, and we inverted the training and test sets for SE due to its ease.

Table 2. Selected Data Sets (C=Classes, d=dimensionality)

	Glass	Clouds98	Clouds99	Clouds204	Isolet	Letter	Satellite	Vowel	Segmentation	Zoo
Code	GL	CL98	CL99	CL204	IS	LE	SA	VO	SE	ZO
Size	214	69	321	500	7797	20000	6435	990	2310	101
|C|	6	4	7	10	26	26	6	11	7	7
d	9	98	99	204	617	16	4	10	19	16

Table 3. Average Percent Accuracies and Standard Deviations

Data Set	No Feature Selection	With Feature Selection		
	$IB1_{atomic}$	$IB1_{atomic}$	$IB1_{opc}$	$IB1_{ecoc}$
GL	68.3±11.9	69.1±11.5	**67.2**±8.3	73.8±7.8
CL98	64.0±15.1	65.7±18.1	65.5±13.4	65.5±11.6
CL99	**53.9**±9.3	58.5±9.3	**48.6**±8.3	59.8±8.4
CL204	**60.6**±7.9	**57.8**±6.7	*46.0*±5.1	73.8±4.6
IS	84.6	84.0	24.4	90.4
LE	**83.4**±2.3	83.5±2.2	**83.5**±2.1	84.9±3.2
SA	**80.7**±1.3	**80.5**±2.7	*79.7*±1.7	82.2±1.6
VO	57.7±8.0	57.7±8.0	57.7±8.0	57.8±8.1
SE	96.8±1.1	96.8±1.1	96.8±1.1	96.9±1.1
ZO	94.5±6.4	92.7±8.4	94.5±6.4	94.5±6.4

boldface: significantly lower 1-tailed t-test accuracies than $IB1_{ecoc}$ ($p < 0.05$)

italics: significantly lower 1-tailed t-test accuracies than $IB1_{atomic}$ with no feature selection ($p < 0.05$)

3.2 Empirical comparison

Table 3 summarizes the results of our experiment. Our primary finding is that, *when feature selection is useful and when different features are selected for different bit functions, ECOCs can significantly improve the accuracy of the local learner* IB1.

Feature selection was not always appropriate; it increased accuracy on only four data sets, and never significantly increased accuracy. $IB1_{ecoc}$ performed well, "repairing" $IB1_{atomic}$'s accuracy whenever it was reduced by feature selection. $IB1_{ecoc}$'s accuracies were always significantly higher or not significantly different than the other algorithms' accuracies.

We hypothesized that ECOC representations increase accuracy for high-dimensional data sets where different feature subsets are useful for learning different output bits (CL98, CL99, CL204, IS, SA). Thus, we examined whether the modified racing schemata algorithm selected different features for different bit functions. We define *Selected* as the average number of features selected per output bit function, and *ComInPairs* as the average number of features in common selected for a given pair of bit functions. The ratio of *Selected* to *ComInPairs* is a rough measure of the variability of the features selected among different bit functions.

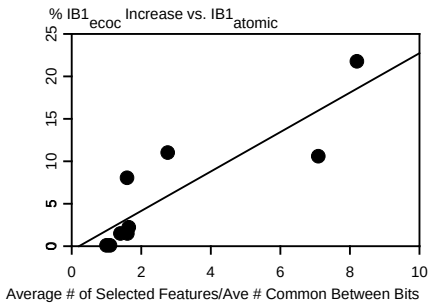

Fig. 4. Correlation Between Variances in the Feature Selection Per Bit and Average ECOC Accuracy Increases Versus IB1$_{atomic}$ Without Feature Selection

Figure 4 plots, for each data set, the ratio of Selected to ComInPairs against the percentage accuracy improvement obtained by IB1$_{ecoc}$ when compared with IB1$_{atomic}$ (without feature selection). The correlation of these variables is fairly high (0.87), suggesting a linear relationship: ECOC accuracy gains tend to increase with increasing diversity among the features selected per output bit. The line in Figure 4 is the linear regression equation computed from these values.

4 Bias, Variance, and Error Decomposition

This section first reviews Breiman's [1996] definitions for bias and variance (i.e., for classification tasks) and then presents the error decomposition for some of the data sets.

A classification problem is completely described by k real deterministic functions $P(Y = i|X = x)$, where X describes the input parameters and Y the output classes. The minimum misclassification rate is obtained using the Bayes optimal classifier:

$$Y_B(x) = \arg\max_i P(Y = i|X = x) \tag{1}$$

Given a finite training set $T = \{(x_i, y_i) : i = 1, \ldots, m\}$ the classifier induced by a supervised learner depends on T, which we denote as $\hat{Y}(x|T)$. The *aggregate* classifier $Y_A(x)$ can be defined as:

$$Y_A(x) = \arg\max_i P(\hat{Y}(x|T) = i) \tag{2}$$

The predictions of the Bayes optimal classifier and aggregate classifier differ when $P(\hat{Y}(x|T) = i) \neq P(Y = i|X = x)$. Breiman defines a classifier to be unbiased at x if $Y_A(x) = Y_B(x)$. Let $\mathcal{U}$ be the set of instances at which $\hat{Y}$ is unbiased and $\mathcal{B}$ its complement. The bias, variance, and error decomposition are defined as follows:

$$Bias(\hat{Y}) = P(\hat{Y}(x) \neq Y(x), x \in \mathcal{B}) - P(Y_B(x) \neq Y(x), x \in \mathcal{B}) \tag{3}$$

$$Var(\hat{Y}) = P(\hat{Y}(x) \neq Y(x), x \in \mathcal{U}) - P(Y_B(x) \neq Y(x), x \in \mathcal{U}) \tag{4}$$

Table 4. Bias and Variance Decomposition of the Error for Archived Data

algo	Glass	Clouds98	Clouds99	Clouds204	Zoo
IB1	0.318	0.390	0.432	0.385	0.034
Bias	0.302	0.374	0.408	0.371	0.027
Variance	0.016	0.016	0.024	0.013	0.007
$IB1_{atomic}$	0.287	0.349	0.422	0.392	0.038
Bias	0.178	0.286	0.335	0.297	0.031
Variance	0.109	0.063	0.087	0.096	0.007
$IB1_{opc}$	0.313	0.403	0.481	0.566	0.047
Bias	0.232	0.347	0.390	0.418	0.039
Variance	0.080	0.056	0.091	0.148	0.008
$IB1_{ecoc}$	0.234	0.364	0.374	0.268	0.028
Bias	0.177	0.294	0.333	0.209	0.022
Variance	0.057	0.070	0.041	0.058	0.006

$$Er(\hat{Y}) = Er(Y_B) + Bias(\hat{Y}) + Var(\hat{Y}) \tag{5}$$

To estimate bias and variance, we randomly split the data into 100 partitions, with 90% of the data used for training and 10% for testing. We used the relative frequency that each instance x was classified as i to estimate $P(\hat{Y}(x|T) = i)$. The bias set $\mathcal{B}$ and its complement $\mathcal{U}$ can be obtained based on these estimates. Given that each instance is unique in each of our data sets, we assumed, as did Kohavi and Wolpert [1996], that the Bayes optimal error rate is zero for each data set tested (see also [Ricci and Aha, 1997a]). The results of the experiments conducted on some of the data sets are shown in Table 4.

From these results we conclude that *ECOCs drastically reduce the bias component of the error at the cost of moderately increasing the variance.* Bias is always reduced, from a minimum of 18% (Clouds99) to a maximum of 44% (Clouds204). Conversely the decrease in the total error obtained by $IB1_{ecoc}$ is moderated by an increase in variance (i.e., $IB1_{ecoc}$'s variance was at least twice $IB1_{atomic}$'s variance for each data set).

5 Discussion

In addition to the research reported by Dietterich and his colleagues, this research was inspired by Aha and Bankert [1997], who reported promising ECOC results for one data set using a k-nearest neighbor variant coupled with a forward sequential feature selection algorithm. This paper extends their work, exploring whether feature selection can decorrelate output bit errors sufficiently for nearest neighbor to work well with ECOCs. We found similar behavior with $k > 1$ in other experiments (not reported here).

Previous research on ECOCs did not stress feature selection, which is crucial for some tasks. Due to their low training costs, nearest neighbor classifiers are

excellent for use with expensive feature selection approaches [Aha and Bankert, 1997]. Perhaps the most effective feature selectors are those that guide search using feedback from the classifier itself. This is expensive because it requires evaluating the classifier on many feature subsets, which is a good motivation for using an inexpensive classifier such as nearest neighbor. Thus, our contributions are useful for multiclass classification tasks where (1) feature selection is needed and (2) obtaining high predictive accuracy is a priority.

Although they can increase accuracy on some tasks, ECOCs have limitations. For example, the arbitrarily-generated class partitions corresponding to each output bit have no meaningful interpretation. Additional research could explore whether ECOCs can work well with meaningful (e.g., pre-determined) class partitions. Also, training ECOCs is slow because they must induce one classifier per output bit, and they typically require using many more output bits than classes to perform well. Feature selection compounds this problem, which is why we selected a fast feature selector. However, because feature selection algorithms that incorporate classifier feedback are computationally expensive, alternative methods should be considered when speed is important. Finally, IB1$_{ecoc}$ tended to work best with larger training sets, higher dimensional spaces, and when feature selection is appropriate.

6 Summary and Future Work

We investigated the hypothesis that ECOCs can increase the classification accuracies of local learning algorithms (e.g., nearest neighbor) when used in conjunction with a feature selection algorithm. If feature selection yields different features for each output bit, then their errors will be decorrelated because different class partition hypotheses will be generated for each output bit. This is the same reason why global algorithms increase accuracy when integrated with ECOCs.

Our empirical results provide evidence for our hypothesis. We also hypothesized conditions under which this combination of algorithms will perform well, and presented evidence suggesting that it works well for tasks that require feature selection and where different features are selected for each output bit. In some cases, the accuracy improvements were dramatic.

Future research topics include using different distance metrics for different bit functions, using alternative feature selection algorithms, and investigating whether similar benefits can be obtained using feature weighting rather than feature selection algorithms.

Acknowledgements

This work was supported by IRST and the Office of Naval Research. Thanks to Diana Gordon and Ken De Jong for their feedback on earlier versions of this paper, and to Paul Tag and Rich Bankert for the cloud data sets.

References

[Aha and Bankert, 1997] D. W. Aha and R. L. Bankert. Cloud classification using error-correcting output codes. *Artificial Intelligence Applications: Natural Science, Agriculture, and Environmental Science*, 11:13–28, 1997.

[Aha *et al.*, 1991] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[Aha, 1992] D. W. Aha. Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.

[Bottou and Vapnik, 1992] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.

[Breiman, 1996] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, University of California, Berkeley, April 1996.

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

[Kohavi and Wolpert, 1996] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss function. In *Proceeding of the Thirteenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, 1996. Morgan Kaufmann.

[Kong and Dietterich, 1995] E. B. Kong and T. G Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, Tahoe City, CA, 1995. Morgan Kaufmann.

[Maron and Moore, 1997] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, pages 193–225, 1997.

[Merz and Murphy, 1996] C. Merz and P. M. Murphy. UCI repository of machine learning databases. [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1996.

[Perrone and Cooper, 1993] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing.* Chapman and Hall, Philadelphia, PA, 1993.

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

[Ricci and Aha, 1997a] F. Ricci and D. W. Aha. Bias, variance, and error correcting output codes for local learners. Technical Report 9711-10, IRST, 1997.

[Ricci and Aha, 1997b] F. Ricci and D. W. Aha. Extending local learners with error-correcting output codes. Technical Report 9701-08, IRST, 1997.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition.* MIT Press, Cambridge, MA, 1986.

[Stanfill and Waltz, 1986] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communication of ACM*, 29:1213–1229, 1986.

[Zhang *et al.*, 1992] X. Zhang, J. Mesirov, and D. Waltz. Hybrid system for protein structure prediction. *Journal of Molecular Biology*, 225:1049–1063, 1992.

This article was processed using the LaTeX macro package with LLNCS style