PATTERN RECOGNITION AND STRING MATCHING
D. Chen and X. Cheng (Eds.) pp. – - –

Terry Windeatt, Reza Ghaderi
School of Electronics, Computing and Maths
University of Surrey, Guildford, Surrey, United Kingdom GU2 7XH
E-mail: t.windeatt@surrey.ac.uk

Contents

1 Abstract **2**

2 Introduction **2**

3 Output coding **3**
3.1 Error-Correcting Output Coding (ECOC) 4
3.2 ECOC algorithm 5

4 String Design **5**
4.1 Maximum Distance Code 6
4.2 Equidistant code and Bayes rule 7
4.3 Equidistant code and unbiased distance 7
4.4 Summary of Constraints 9
4.5 Code Sensitivity 9

5 Alternate Combining Strategies **10**
5.1 Least squares (LS-OC) 10
5.2 Inverse Hamming Distance (IHD-OC) 10
5.3 Linear estimation (Lin-OC) 11
5.4 Dempster-Shafer (DS-OC) 11

6 Examples **12**
6.1 Synthetic Data 12
6.2 Popular Benchmark Data 13
6.3 Face Verification 14

7 Conclusion **16**

References

1 Abstract

The Output Coding technique for solving multi-class learning problems was originally proposed with rows of an error-corecting code matrix acting as code words to represent the classes. We summarise the requirements on design of the binary strings in the code matrix and consider alternate combining strategies. For shorter codes, it is shown that both code design and combining strategy can affect generalisation. Examples are presented on synthetic data, on natural benchmark data and on an application in face verification.

2 Introduction

Traditionally, the approach that has been used in the design of pattern classification systems is to experimentally assess the performance of several classifiers with the idea that the best one will be chosen. However, the theory of ensemble classifiers represents a departure from the traditional strategy and has been developed in the past decade to address the problem of designing a system with improved accuracy. Recognising that each classifier may make different and perhaps complementary errors, the aim is to pool together the results from all classifiers to find a composite system that outperforms any individual classifier. In this way a single complex classifier may be replaced by a set of relatively simple classifiers. In the context of a system of multiple classifiers these constituent classifiers are sometimes referred to as base classifiers.

It is known that certain conditions need to be satisfied to realise the improvement from combining multiple classifiers, in particular that the base classifiers be not too highly correlated [23]. If each classifier solves a slightly different problem, then composite performance may improve. Various learning techniques have been devised with the aim of creating diverse sub-problems, thereby reducing correlation between classifiers before combining. These correlation reduction techniques are of various types and include: (i) reducing dimension of training set to give different feature sets, (ii) incorporating different types of base classifier, (iii) designing base classifiers with different parameters for same type of classifier, (iv) re-sampling training set so each classifier is specialised on different subset, and (v) coding multi-class outputs to create complementary two-class problems. The well-known techniques of Bagging [3] and Boosting [7] are in category (iv), while Output Coding, described in Section 3, is in category (v).

Bagging and the original form of Boosting are both voting algorithms and achieve impressive results for some problems by combining hard-level classifications. By hard-level we mean that a single-hypothesis decision is taken for each

base classifier, in contrast with soft-level which implies a measure of confidence associated with the decision. Bagging (from Bootstrap Aggregating) forms replicate training sets by sampling with replacement, and combines the resultant classifications with a majority vote. Boosting, which combines with a weighted vote is more complex than Bagging in that the distribution of the training set is adaptively changed based upon the performance of sequentially constructed classifiers. Each new classifier is used to adaptively filter and re-weight the training set, so that the next classifier in the sequence has increased probability of selecting patterns that have been previously mis-classified.

The Output Coding method does not re-sample the training set as with Bagging and Boosting, but rather creates diverse two-class sub-problems by relabelling the training patterns according to a set of binary strings. In this way, each classifier is solving a different sub-problem of the original problem. A useful way of trying to understand correlation reduction by class re-labelling is to interpret the implied partitioning of the input space, as shown in the example in Section 6 using synthetic data. There are several motivations for decomposing a multi-class problem into separate and complementary two-class problems. Firstly some accurate and efficient two-class classifiers do not naturally scale up to multi-class. Attention can then be focused on developing an effective technique for the two-class case, without having to consider explicitly the design and automation of the multi-class classifier. Secondly, it is hoped that the parameters of a simple parallel machine run several times are easier to set than a complex machine run once and may facilitate more efficient solutions. Finally, solving different 2-class sub-problems, perhaps repeatedly with random perturbation, may help to reduce error evident in the original problem.

3 Output coding

First consider the need for a suitable Output Coding with reference to the output representation used with a Multi-layer Perceptron (MLP) network, which is a popular base classifier. A single multiple output MLP can handle a multiclass problem directly. The standard technique is to use a k-dimensional binary target string that represents each one of k classes using a single binary value at the corresponding position, for example $[0, ...0, 1, 0, ...0]$ which is sometimes referred to as one-per-class (OPC) encoding. The reason that a single multiclass MLP is not a suitable candidate for use with Output Coding is that all nodes share in the same training, so errors are far from independent and there is not much benefit to be gained from combining [16]. However if a 2-class MLP is used as base classifier, independence among classifiers can be achieved by the problem decomposition defined by

the string, as well as by injection of randomness through the starting weights. Of course, no guarantee can be given that a single MLP with superior performance will never be found, but the assumption is that even if one exists its parameters would be more difficult to determine.

An alternative to OPC is distributed output coding [21], in which k binary strings are assigned to k classes 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 which indicates the presence or otherwise of a useful feature of the problem at hand. The strings are treated as code words and a test pattern is assigned to the class that is closest to the corresponding code word. It is this method of string matching, which is analogous to the assignment stage of error-correcting coding, that provides the motivation for employing error-correcting codes in classification.

3.1 Error-Correcting Output Coding (ECOC)

The Error-Correcting Output Coding (ECOC) method [6], described in Section 3.2, is a two stage method. The second stage is the combining step, based on error-correcting principles under 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. For a two-class problem, classification errors can be one of two types, either predicted class Ω_1 for target class Ω_2 or predicted class Ω_2 for target class Ω_1. From the transmission channel viewpoint, we would expect that the one-per-class (OPC) and distributed output coding matrices would not perform as well as the ECOC matrix, because of inferior error-correcting capability.

The first stage of the ECOC method provides the decomposition strategy, which is the re-labelling of patterns defined by the Output Coding matrix. Each re-labelled class contains patterns from a number of the classes from the original problem. There exists another category of methods for decomposing a complex multiclass problem into simpler two-class sub-problems. In this other category, each base classifier is capable of distinguishing between a pair of classes, which means that the final decision boundary is made up of decision boundaries that each originate from just two classes. Methods like *binary decision clustering* [24] and *pairwise coupling* [9] are included in this category. The main advantage is that sub-problems are normally simpler, which means that the decision boundary of each sub-problem is less complex. However, each sub-problem is based on only a subset of patterns and this may cause some difficulties in training. Furthermore all boundaries between pairs of classes are needed to reconstruct the original bound-

ary, which can lead to a large number of base classifiers ($b = \frac{k!}{2!.(k-2)!}$ for a k class problem). In contrast the ECOC approach requires fewer but more powerful classifiers with the ability to repeat parts of the decision boundary, which can reduce variability as shown in [13].

3.2 ECOC algorithm

In the ECOC method, a $k \times b$ code word matrix Z has one string (code word) for each of k classes, with each column defining one of b sub-problems that use a different labelling. Specifically, for the jth sub-problem, 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 Z_{ij} (typically 0 or 1). One way of looking at the re-labelling is to consider that for each column the k classes are arranged into two super-classes Ω_1 and Ω_2. The jth ($j = 1...b$) classifier is trained using re-labelled patterns from the jth sub-problem.

For example, in a five-class problem if the jth column of Z is given by $[10110]^T$ the relabelling associated with the jth classifier means that training patterns with target classes represented by the first, third and fourth rows of Z comprise superclass Ω_1 and the remaining training patterns comprise superclass Ω_2.

A test pattern is applied to the b trained classifiers giving

$$\mathbf{y} = [y_1, y_2, ...y_b]^T \tag{1}$$

in which y_j is the output (usually real-valued) of jth base classifier. The distance between $\mathbf{y}$ and code word for each class is given by

$$L_i^1 = \sum_{j=1}^{b} |Z_{ij} - y_j| \tag{2}$$

and the pattern is assigned to the class w_i corresponding to closest code word $ArgMin_i(L_i^1)$.

Originally the matching was based on Hamming Distance, but when it could be shown that ECOC produced good probability estimates [14] the combining scheme was changed to soft-level, equation (2) which represents the L^1 norm.

4 String Design

When the Output Coding technique was first developed it was believed that the code matrix should be designed using error-correcting principles to enable it to

generalise well [5, 6]. Various codes have since been proposed, but most code matrices that have been investigated previously are binary and problem-independent, that is pre-designed. Random codes have received much attention, and were first mentioned in [5] as performing well in comparison with error-correcting codes. In [6] random, exhaustive, hill-climbing search and BCH coding methods were used to produce code matrices for different column lengths. Random codes were investigated in [20] for combining Boosting with Output Coding, 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 [11] to give Bayesian performance if pairs of code words were equidistant, and it was claimed that a long enough random code would not be outperformed by a pre-defined code. Although various heuristics have been employed to produce better binary problem-independent codes there appears to be little evidence to suggest that performance significantly improves by a clever choice of code, except that OPC is usually inferior [1, 5, 6]. Recently a three-valued code [1] was suggested which allows specified classes to be omitted from consideration (don't care for third value), thereby permitting integrated representation of methods such as *pairwise coupling* [9]. Another recent development is described in [4] in which problem-dependent codes are investigated and it is claimed that designed continuous codes show more promise than designed discrete codes.

In the Output Coding framework the choice of code matrix may affect performance. Before looking at properties of the code matrix that reduce generalisation error, we first note that there are three kinds of errors that can be distinguished in this framework. First there is the *Bayesian error*, which is due to overlapped classes and cannot be removed by combining, error-correcting or any other method. Secondly there is *base classifier added error*, which is related to the training of base classifiers, and which we hope to remove by the error-correcting capability of code words. Finally there is the *combining error*, which is introduced by the way in which we choose to combine the outputs of base classifiers, and is present even if all classifiers are perfect (behave like Bayesian classifier).

4.1 Maximum Distance Code

It seems reasonable that the greater the Hamming Distance between code words the more likely that error will be reduced, since the code matrix maps the problem into a different space in which classes are more easily separated. In addition, sub-problems are more independent and likely to benefit from combining if Hamming Distance between columns is maximised, remembering that a column and its complement represent identical classification problems [6]. Therefore the first two desirable properties of the code matrix are maximum Hamming distance between

rows and between columns.

4.2 Equidistant code and Bayes rule

Consider that the decomposition of a multiclass classification problem into binary sub-problems in Output Coding can be interpreted as a transformation between spaces from the original output $\mathbf{q}$ to $\mathbf{y}$, given in matrix form by

$$\mathbf{y} = Z^T \mathbf{q} \tag{3}$$

where $\mathbf{q}$ are individual class probabilities.
From the distance-based decision rule in (2) and using (3)

$$L_i^1 = \sum_{j=1}^{b} |(\sum_{l=1}^{k} q_l Z_{lj}) - Z_{ij}| \tag{4}$$

and knowing that $\sum_{l=1}^{k} q_l = 1$ we have

$$L_i^1 = \sum_{j=1}^{b} |\sum_{l=1}^{i-1} q_l Z_{lj} + q_i Z_{ij} + \sum_{l=i+1}^{k} q_l Z_{lj} - Z_{ij}| \quad l = 1, 2...k \tag{5}$$

and therefore

$$L_i^1 = (1 - q_i) \sum_{j=1}^{b} \sum_{l=1}^{k} |Z_{ij} - Z_{lj}| \tag{6}$$

Consider the situation that the Hamming Distance between ith and jth code word is equal for any i and j, $i \neq j$. Then from (6), we see that when all pairs of code words are equidistant, minimising L_i^1 implies maximising posterior probability which is equivalent to Bayes rule (as also derived in [10, 27])

$$ArgMax_i(q_i) = ArgMin_i(L_i^1) \tag{7}$$

Therefore, any variation in Hamming Distance between pairs of code words will reduce the effectiveness of the combining strategy. The third desirable property of the code matrix is equal Hamming Distance between all pairs of rows.

4.3 Equidistant code and unbiased distance

Consider the case that there is a single output for each classifier (y_j) so that we can assume that a threshold exists which separates the members of the two *super groups* of classes. If the threshold for all classifiers is half way between the binary

limit values (0.5 for 0-1 binary), there will be a fair comparison of distance to each code word. However if there exists some base classifier bias that causes the threshold to move away from the mid-position, then the comparison will not be fair unless there are an equal number of 1's in each code word. Therefore the third desired property of the code matrix is equal number of 1's in all code words.

Now we show that an equidistant code automatically satisfies this property. Let Z be an equidistant code with d bits error-correcting capability, $\sum_{l=1}^{b} |Z_{il} - Z_{jl}| = 2d + 1$ for any pair i, j. Since Hamming Distance between pair i, j is the sum of the number of 1s in row i and row j minus number of common 1s between i, j we may write

$$\sum_{l=1}^{b} Z_{il} + \sum_{l=1}^{b} Z_{jl} - 2\sum_{l=1}^{b} Z_{il} Z_{jl} = 2d + 1 \tag{8}$$

similar equations can be written for pair i, k and pair j, k

$$\sum_{l=1}^{b} Z_{il} + \sum_{l=1}^{b} Z_{kl} - 2\sum_{l=1}^{b} Z_{il} Z_{kl} = 2d + 1 \tag{9}$$

$$\sum_{l=1}^{b} Z_{jl} + \sum_{l=1}^{b} Z_{kl} - 2\sum_{l=1}^{b} Z_{jl} Z_{kl} = 2d + 1 \tag{10}$$

From (8), (9),(10) after re-arranging

$$\sum_{l=1}^{b} Z_{il} Z_{jl} = \sum_{l=1}^{b} Z_{kl} Z_{jl} = \sum_{l=1}^{b} Z_{kl} Z_{il} = m \tag{11}$$

where m is number of common 1s in code word, and

$$\sum_{l=1}^{b} Z_{il} = \sum_{l=1}^{b} Z_{kl} = \sum_{l=1}^{b} Z_{jl} = n \tag{12}$$

where n is the number of 1s in each row.

Therefore if Z is an equidistant matrix, there is the same number of 1s in any row, and the same number of common 1s between any pair of rows, and Z provides an unbiased distance measurement. For example, if ZZ^T can be written in the form

$$ZZ^T = \begin{bmatrix} n & m & \cdots & m \\ m & n & \cdots & m \\ \cdots & \cdots & \cdots & \cdots \\ m & m & \cdots & n \end{bmatrix} \tag{13}$$

then properties (11) and (12) are automatically satisfied since

$$ZZ^T = \sum_{l=1}^{b} Z_{il} Z_{lj}^T = \sum_{l=1}^{b} Z_{il} Z_{jl} = \begin{cases} n & \text{if i=j} \\ m & \text{otherwise} \end{cases}$$

4.4 Summary of Constraints

The main constraints for designing set of binary strings that make up the code matrix are therefore as follows

- minimum Hamming Distance between rows (error-correcting capability)
- variation of Hamming Distance between rows (effectiveness of combining)
- number of columns (repetition of different parts of sub-problems)
- Hamming Distance between columns and complement of columns (independence of base classifiers)

From the theory of error-correcting codes [19] we know that finding a matrix with long code words, and having maximum and equal distance between all pairs of rows is complex. In the example in Section 6, random, equidistant and non-equidistant code matrices are compared as number of columns is varied. It shows that as code length increases the difference in performance between equidistant and random codes becomes less significant. Equidistant codes can be produced by using the BCH method [19], which employs algebraic techniques from Galois field theory. The number of rows is over-produced (BCH requires number to be power of 2), before using properties (11) and (12) to select a subset of strings. Of course these properties provide only necessary conditions for equidistant strings and so cannot be used to generate them in isolation.

4.5 Code Sensitivity

There is another issue, in addition to those described in Section 4.4, that may affect performance. Since in practice the overlap between classes can be different, we might suspect that certain code words are better suited to some super-classes more than others. Although it would seem a good idea to use a labelling in which distance between labels is based on distance between classes (e.g. Mahalanobis distance), the additional constraint is likely to be difficult to implement. Taking a different approach [27] the following modification to ECOC is suggested, in which all code words are used for each class to reduce sensitivity to code word selection.

In training, circularly shift one row of ECOC matrix k times, and for each single shift train b base classifiers. In testing, compute the distance $L_{n_i}^1$ for class

w_i between b-dimensional base classifier outputs and string corresponding to class w_i. After averaging the final distance measurement, assign pattern to class w_i given by $Argmin_i \sum_{n=1}^{k} L_{n_i}^1$.

5 Alternate Combining Strategies

A variety of combining strategies have been proposed for Output Coding besides L^1 norm, referred to here as L1-OC. Other combining strategies discussed in this section include three approaches for recovering individual class probabilities, which are Least Squares (LS-OC), Inverse Hamming Distance (IHD-OC) and Linear estimation (Lin-OC). For these three methods the classification decision is based on maximising the probability estimate. One further method is also described, Dempster-Shafer (DS-OC) based on maximising probability mass function [22]. The original Hamming Distance combining strategy (HD-OC) was shown in [13] to be analogous to majority vote over the classes. Besides L1-OC, LS-OC has been the most extensively investigated. The justification for LS-OC in terms of probability estimation was reported in [14], and in [10] LS-OC was extended by incorporating ridged regression when b is small.

5.1 Least squares (LS-OC)

Recovering individual class probabilities from super-class probabilities is easily accomplished if the individual probability estimates are exact and if the k independent equations that result from equation (3) can be solved. In practice, base classifiers will not produce correct probabilities but we can model the error assuming that the expected value of the error is zero for unbiased probability estimates [8]. In general Z^T is not a square matrix, and so does not have an inverse, but $\mathbf{q}^*$, an optimal value of $\mathbf{q}$ in equation (3), can be found using the method of least squares

$$\mathbf{q}^* = (ZZ^T)^{-1}Z\mathbf{y} \tag{14}$$

using a cost function such as $(\mathbf{y} - Z^T\mathbf{q})^T(\mathbf{y} - Z^T\mathbf{q})$.

5.2 Inverse Hamming Distance (IHD-OC)

From equation (2) we may write the matrix equation

$$\mathbf{L^1} = \Delta\mathbf{q} \tag{15}$$

where $\mathbf{L^1} = [L_1^1, L_2^1, ...L_k^1]$ and Δ is a matrix whose element Δ_{ij} represents the Hamming Distance between row i and row j. Since the elements of Δ are non-negative, Δ can be inverted to find an estimate for $\mathbf{q}$.

5.3 Linear estimation (Lin-OC)

The assumption here is that there is a linear relationship between outputs of the base classifiers and individual class probabilities.

$$q_i = \sum_{j=1}^{b} W_{ij} y_j \qquad i = 1...k \tag{16}$$

where

$$\sum_{j=1}^{b} W_{ij} = 1 \qquad i = 1...k \tag{17}$$

Although there are several ways to estimate the weights, a simple approach is to count, for each column, the number of times that a training pattern belongs both to class w_i and to superclass Ω_1. The count is repeated for all classes ($i = 1, ...k$) and normalised as in (17).

5.4 Dempster-Shafer (DS-OC)

The Dempster-Shafer approach to combining provides a mathematical framework which may be regarded as a generalised form of Bayesian statistics [22]. It is based upon the concept of probability mass function (basic probability assignment) which assigns a value to a subset of propositions. We assume that there are k elemental propositions, one each for the k classes to which a pattern can be assigned. In the context of Output Coding each column of the code matrix Z defines a partitioning into two superclasses as described in Section 3.2. A superclass is a subset of the k classes and represents one of 2^k possible subsets in Dempster-Shafer theory. Therefore, each column can be regarded as providing evidence for class membership, which can be combined with evidence from previous columns.

The mass function of the jth base classifier is set to $m_j(A_j) = [y_j \quad 1 - y_j]$ for the two superclasses Ω_1 and Ω_2, where y_j is the jth base classifier output (defined in (1)), and A_j represents the decomposition defined by the jth column of Z. The evidence for the $(j+1)th$ base classifier can be combined recursively with evidence from the previous j classifiers by invoking Dempster's rule of combination to calculate the orthogonal sum $m_j \oplus m_{j+1}$

$$m_{j+1} = [m_j \oplus m_{j+1}] = 1/K \sum_{A_j \cap A_{j+1}} m_j(A_j) m_{j+1}(A_{j+1}) \qquad (18)$$

where normalisation factor K is given by $1 - \sum_{A_j \cap A_{j+1}=\{\}} m_j(A_j) m_{j+1}(A_{j+1})$. Initially with $j = 1$ equation (18) gives $m_2 = m_1 \oplus m_2$, and equation (18) is applied recursively for $j = 1, ...b-1$ to find the mass function for the combination of b classifiers. The decision strategy is to assign the class corresponding to the maximum probability mass function.

6 Examples

6.1 Synthetic Data

An artificial five-class overlapping Gaussian problem is defined by five groups of two-dimensional random vectors having normal distribution, shown in Table 1. The classifiers are not trained, but using the parameters from Table 1, the posterior probability of super-class membership is computed. The Bayes rate for this five class problem is 74.28%, and the Bayes (optimum) decision boundary is shown in figure 1.

class	w_1	w_2	w_3	w_4	w_5
μ_i(mean)	[0,0]	[3,0]	[0,5]	[7,0]	[0,9]
σ_i^2(variance)	1	4	9	25	64

Table 1: Distribution parameters of data used in artificial benchmark

The following code matrices are used in these experiments:

C1: $k \times k$ one-per-class (OPC: 1s on main diagonal 0 elsewhere))
C2: $k \times 7$ matrix with randomly chosen binary elements
C3: $k \times 7$ BCH code (minimum distance of 3, non-equal)
C4: $k \times 7$ BCH code with equal distance of 4
C5: $k \times 15$ matrix with randomly chosen binary elements
C6: $k \times 15$ BCH code with equal distance of 8

To determine robustness of the Output Coding framework, Gaussian random noise with different mean (μ_n) and variance (σ_n^2), is added to the soft-level (real-valued) output of base classifiers to simulate noisy imperfect classifiers. Decision boundaries in figure 2(a) (code C1) and figure 2(b) (code C4) demonstrate the difference between one-per-class (OPC) and the ECOC concept. By comparing with

the Bayes boundary shown in figure 1, we can see that each classifier concentrates on different part of the input space. So each classifier has a less complex problem to handle compared with the complete boundary that would, for example, be handled by a single multiclass classifier. Figure 2(b) also shows how some parts of the boundary appear in more than one classifier, and so are learned repeatedly, as investigated in [13, 25]. For added noise ($\mu_n = 0$, $\sigma_n^2 = 0.25$) figure 3(a) and figure 3(b) show respectively the C1 and C4 decision boundaries of the ensemble as number of individual classifiers is increased. A comparison shows that code C4, by virtue of repeated learning of parts of decision boundaries, more effectively reduces variance.

Figure 4 shows a plot of the percentage match with Bayes rate as level of noise is increased and demonstrates the robustness of ECOC when it has been modified to reduce sensitivity to code word selection (Section 4.5). The error-correcting capability for codes C2, C4, C6, C5 is 1,3,7,2 bits respectively. From figure 4 the following points can be observed:

- if no noise is added, the equidistant matrices C4, C6 match Bayes rate perfectly
- if the number of classifiers is the same, the equidistant code performs slightly better than random code, shown by comparing C5 with C6 (b = 15) and C4 with C2 (b = 7)
- if longer random code is used, repetition of sub-problems gives improved performance even if error-correction capability is reduced, shown by comparing C5 with C4

Figure 5 compares original and modified decision boundaries when noise ($\mu_n = 0$, $\sigma_n^2 = 0.25$) is added to each individual classifier, and confirms that error variance has been reduced.

6.2 Popular Benchmark Data

Simple two-dimensional data shown in the example in Section 6.1 is not necessarily representative of real data, so it is interesting to see performance on some popular benchmarks (See [2] for a description of the datasets). The base classifier is a conventional single hidden layer MLP trained by Levenberg-Marquardt algorithm, with default training parameters.

There are fifty epochs of training and the number of hidden nodes for the datasets is (zoo/1, car/1, vehicle/5, glass/2, satellite/2). Table 2 gives a comparison of codes and combining strategies over the five datasets for codes C1 to

Code	L1	LS	IHD	Lin	DS
C1*	0.91	0.91	0.91	0.91	0.91
C2	0.89	0.83	0.61	0.58	0.90
C3	0.84	0.84	0.83	0.82	0.84
C4*	0.92	0.92	0.92	0.92	0.92
C5	0.97	0.97	0.97	0.94	0.97
C6*	0.97	0.97	0.97	0.97	0.99

Table 2: Mean value over five databases of ratio of classification rate of specified coding/combining (* equidistant) strategy to maximum classification rate over all strategies for a particular database

C6, where each entry denotes the mean ratio of classification rate at the specified code/combining strategy to the best classification rate for that problem over all strategies. From table 2, we can observe that longer codes generally perform better but combining strategy appears to have little impact on performance. However, for code C2 the results for IHD-OC and Lin-OC demonstrate that these two combining strategies are more sensitive to shorter codes than the others. For 7-bit code, equidistant performs best (C2 and C3 compared with C4). For the 15-bit code, the individual results show that random is better for zoo (error rate 5.2% compared with 6.5%) and for car (error rate 25.5% compared with 27.2%). while equidistant is better for the other three datasets, but there is no difference in mean ratio.

On seven data-sets (zoo, vehicle, segment, iris, glass, cmc, car) with ten independent runs the improvement in test error rate of modified (Section 4.5) over original ECOC is 5% mean and 30% standard deviation averaged over the seven data-sets. Further details may be found in [26].

6.3 Face Verification

Automation of a system that performs personal identity verification may use a variety of biometric modalities including facial features, voice characteristics, iris scan, fingerprints. Facial images are a popular source of biometric information since they are relatively easy to acquire, and provide discriminatory features routinely used by humans for recognition. However automated face verification systems often have poor levels of performance and improving them is known to be a difficult task.

The extended M2VTS (XM2VTS) database contains 295 subjects, who were recorded in four separate sessions uniformly distributed over a period of 5 months, and within each session a number of shots were taken including both frontal-view and rotation sequences. Some sample profiles are shown in figure 6, and further de-

tails of this database can be found in [18]. [1] The experimental protocol (known as Lausanne evaluation protocol) provides a framework within which the performance of vision-based person authentication systems running on XM2VTS database can be measured. The protocol assigns 200 clients and 95 impostors, with two shots of each session for each subject's frontal or near frontal images [17], giving 3 training, 3 evaluation and 2 test images for each client. The impostor set is partitioned into 25 evaluation and 70 test impostors. Details of the this protocol can be found in [15]. [2]

Each image is first projected to a lower dimensional feature space using Principal Components and Linear Discriminant Analysis (PCA + LDA) so that each pattern is represented by a vector with 199 elements. The base classifier is a MLP with one hidden layer containing 199 input nodes, 35 hidden nodes, two output nodes and with fixed learning rate, momentum and number of epochs. The dual output is mapped to a value between 0 and 1 to give an estimation of probability of super-class membership. Further details of the system can be found in [12].

There are 200 clients, so from the identification viewpoint it is a 200 class problem. An equidistant 200x511 code matrix is generated by the BCH method, as described in Section 4.4, with Hamming Distance between any pair of rows being 256 bits. In the ECOC method, the classifier outputs represent estimates of super-class probabilities and these are the estimates used to represent clients and impostors for face verification. Specifically, each client i is represented by a set Y_i of N ECOC classifier output vectors, i.e.

$$Y_i = \{\underline{y}_i^l | l = 1, 2, ..., N\} \tag{19}$$

where N is the number of ith client patterns available for training.

The verification task reduces to ascertaining whether classifier outputs representing an image are jointly consistent with a claimed identity. In order to test the hypothesis that the client claim is authentic it is necessary to adopt a test statistic such as the average distance between $\underline{y}$ and the elements of set Y_i. The metric in (2) is preferred to Euclidean distance since it is believed to be more robust to outliers, that is

$$d_i(\underline{y}) = \frac{1}{N} \sum_{l=1}^{N} \sum_{j=1}^{b} |y_j^l - y_j| \tag{20}$$

where y_j is the jth binary classifier output for the test pattern, and y_j^l is the jth classifier output for the lth member of class i. The distance in (20) is checked against a decision threshold, which can be set using the evaluation set, to determine if the

[1] *http://www.ee.surrey.ac.uk/Research/VSSP/xm2fdb.html*

[2] *http://www.idiap.ch/∼m2vts/Experiments/xm2vtsdb_protocol_october.ps*

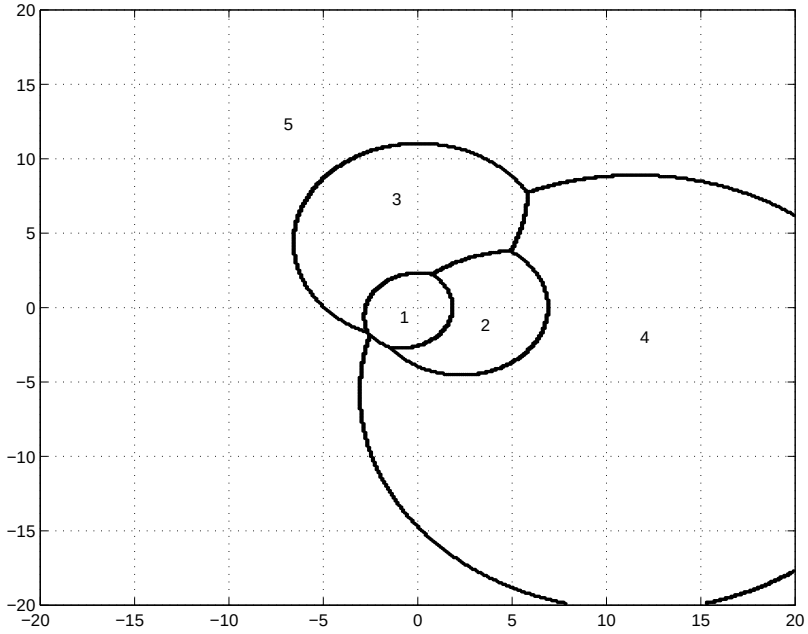
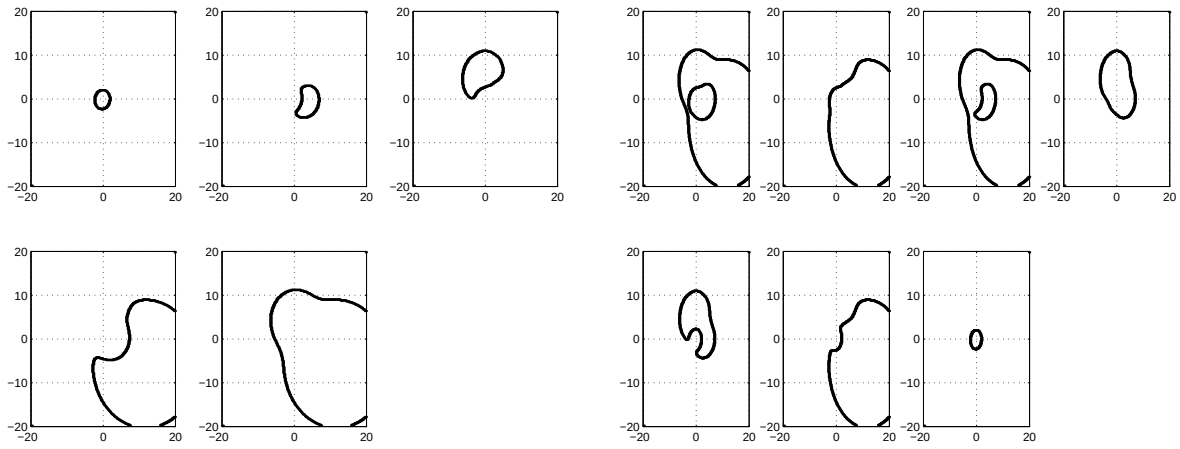


Figure 1: Optimum decision boundary for five class artificial data

client's claim is accepted, Using this ECOC framework the best performance to date has been achieved using this publicly available benchmark (False Acceptance 0.75 % and False Rejection 1.25%).

7 Conclusion

Output Coding is a method for solving multiclass learning problems that uses a set of binary strings to decompose the multiclass into complementary two-class problems. A two-class classifier is then sufficient to solve the multiclass problem. Although error-correcting codes were originally proposed for designing binary strings, random codes achieve similar performance to optimal codes provided they are long enough. It is shown here that while a variety of combining strategies are possible in the Output Coding framework performance is not very dependent on the choice.



(a) (b)

Figure 2: Individual decision boundaries made by several base classifiers (a) C1 Code 5 classifiers (b) C4 code 7 classifiers

Acknowledgments The support received for the face database from Omniperception Ltd, EPSRC grant GR/M61320 and EU framework Project Banca is gratefully acknowledged.

References

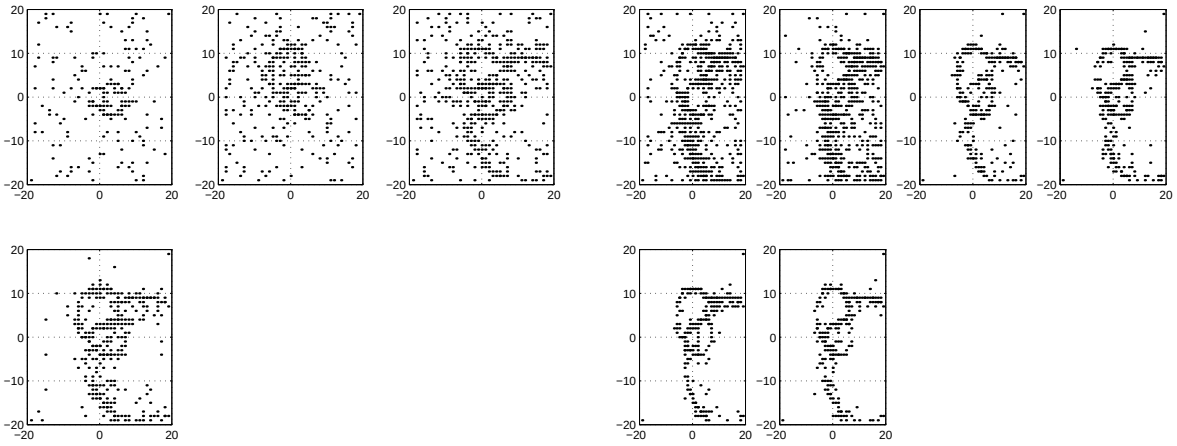
[1] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multi-class to binary: A unifying approach for margin classifiers. *Machine learning research*, 1:113–141, 2000.

[2] C.L. Blake and Merz C.J. Uci repository of machine learning databases. Technical report, Irvine, Univ. of Calif., Inf. and Comp. Scienc, 1998.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1997.

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

[5] T.G Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceed-*



(a) (b)

Figure 3: Decision boundaries of ensemble as number of base classifiers is increased in presence of noise ($\mu_n = 0$, $\sigma_n^2 = 0.25$). The noise-free Bayes boundary is shown in figure 1. (a) C1 code 2,3,4,5 classifiers (b) C4 code 2,3,4,5,6,7 classifiers (numbered left to right)

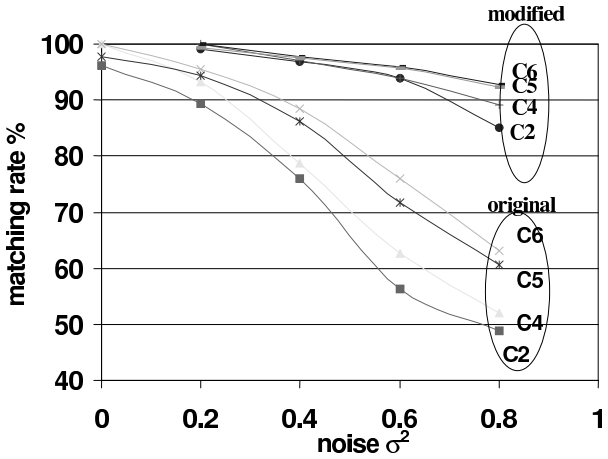


Figure 4: Percentage match with Bayes rate for increasing levels of Gaussian noise modified ECOC (top 4 curves) and original method

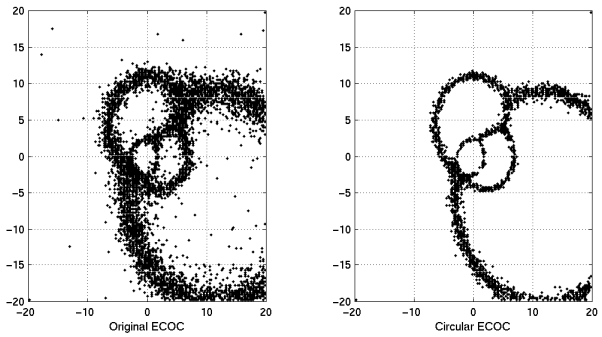


Figure 5: Comparison of decision boundaries for added noise when ECOC has been modified (right) to make it less sensitive to code word selection

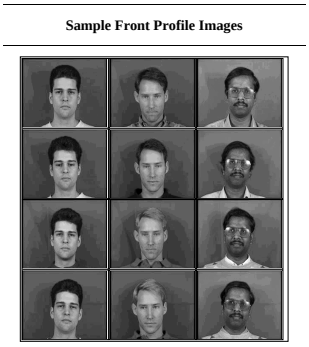


Figure 6: Images from Extended M2VTS Database

ings of the Ninth National Conference on Artificial Intelligence (AAAI-91), pages 572–577. AAAI Press, 1991.

[6] 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.

[7] Y. Freund and R.E. Schapire. A decision-theoretic generalisation of on-line learning and application to boosting. *Journal of computer and system science*, 55:119–139, 1997.

[8] R. Ghaderi and T. Windeatt. Least squares and estimation measures via error correcting output code. In *2nd Int. Workshop Multiple Classifier Systems,Lecture notes in computer science, Springer-V erlag*, pages 148–157, Jul 2001.

[9] T. Hastie and R Tibshirani. Classification by pairwise coupling. *The annals of statistics*, 2:451–471, 1998.

[10] G. M. James. *Majority Vote Classifiers: Theory and Applications.* PhD thesis, Dept. of Statistics, Univ. of Stanford, Calif., May 1998.

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

[12] J. Kittler, R. Ghaderi, T. Windeatt, and G. Matas. Face verification using error correcting output codes. In *Computer Vision and Pattern Recognition CVPR01*, volume 1, pages 755–760, Hawaii, December 2001. IEEE Press.

[13] 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.

[14] E.B. Kong and T.G. Diettrich. Probability estimation via error-correcting output coding. In *Int. Conf. of Artificial Inteligence and soft computing*, Banff,Canada, 1997.

[15] J Luettin and G. Maitre. *Evaluation Protocol For The Extended M2VTS Database (XM2VTS)*. Dalle Molle Institute for Perceptual Artificial Intelligence, P.O. Box 592 Martigny, Valais, Switzerland, July 1998. IDIAP-Com 98-05.

[16] F. Masulli and G. Valentini. Effectiveness of error correcting output codes in multiclass learning problems. In J.Kittler and F.Roli, editors, *Multiple*

Classifier Systems, MCS2000, pages 107–116, Cagliari, Italy, 2000. Springer Lecture Notes in Computer Science.

[17] J. Matas, M. Hamouz, M. Jonsson, J. Kittler, Y. Li, C. Kotroupolous, A. Tefas, I. Pitas, T. Tan, H. Yan, F. Smeraldi, J. Bigun, N. Capdevielle, w. Gerstner, S. Ben-Yacoub, Y. Abduljaoued, and Y. Majoraz. Comparison of face verification results on the xm2vts database. In A. A Sanfeliu, J.J Villanueva, M. Vanrell, R. Alqueraz, J. Crowley, and Y. Shirai, editors, *Proceedings of the 15th ICPR*, volume 4, pages 858–863, Los Alamitos, USA, September 2000. IEEE Computer Soc Press.

[18] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In *Proc. of AVBPA'99*, pages 72–77, 1999.

[19] W.W. Peterson and J.R. Weldon. *Error-Correcting Codes*. MIT press, Cambridge,MA, 1972.

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

[21] T.J. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex systems*, 1:145–168, 1987.

[22] G. Shafer. *A mathematical theory of evidence*. Princeton Univ Press, Princeton, New Jersey, 1976.

[23] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science, special issue on combining artificial neural ne tworks: ensemble approaches*, 8(3):385–404, 1996.

[24] C.L. Wilson, P.J. Grother, and C.S. Barnes. Binary decision clustering for neural-network-based optical character recognition. *Pattern Recognition*, 29(3):425–437, 1996.

[25] T. Windeatt and R. Ghaderi. Binary codes for multi-class decision combining. In *14th Annual International Conference of Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 4051, pages 23–34, Florida,USA, April 2000.

[26] T. Windeatt and R. Ghaderi. Multi-class learning and error-correcting code sensitivity. *Electronics Letters*, 36(19):1630–1632, Sep 2000.

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