Arranging simple neural networks to solve complex classification problems

Reza Ghaderi

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

UniS

Centre for Vision, Speech and Signal Processing (CVSSP)
School of Electronic Engineering, Information technology and Mathematics
University of Surrey
Guildford, Surrey GU2 5XH, U.K.

October 2000

© Reza Ghaderi 2000
Summary

In "decomposition/reconstruction" strategy, we can solve a complex problem by 1) decomposing the problem into simpler sub-problems, 2) solving sub-problems with simpler systems (sub-systems) and 3) combining the results of sub-systems to solve the original problem. In a classification task we may have "label complexity" which is due to high number of possible classes, "function complexity" which means the existence of complex input-output relationship, and "input complexity" which is due to requirement of a huge feature set to represent patterns.

Error Correcting Output Code (ECOC) is a technique to reduce the label complexity in which a multi-class problem will be decomposed into a set of binary sub-problems, based on the sequence of "0"s and "1"s of the columns of a decomposition (code) matrix. Then a given pattern can be assigned to the class having minimum distance to the results of sub-problems. The lack of knowledge about the relationship between distance measurement and class score (like posterior probabilities) has caused some essential shortcomings to answering questions about "source of effectiveness", "error analysis", "code selecting", and "alternative reconstruction methods" in previous works. Proposing a theoretical framework in this thesis to specify this relationship, our main contributions in this subject are to: 1) explain the theoretical reasons for code selection conditions 2) suggest new conditions for code generation (equi-distance code) which minimise reconstruction error and address a search technique for code selection 3) provide an analysis to show the effect of different kinds of error on final performance 4) suggest a novel combining method to reduce the effect of code word selection in non-optimum codes 5) suggest novel reconstruction frameworks to combine the component outputs. Some experiments on artificial and real benchmarks demonstrate significant improvement achieved in multi-class problems when simple feed forward neural networks are arranged based on suggested framework.

To solve the problem of function complexity we considered AdaBoost, as a technique which can be fused with ECOC to overcome its shortcoming for binary problems. And to handle the problems of huge feature sets, we have suggested a multi-net structure with local back propagation. To demonstrate these improvements on realistic problems a face recognition application is considered.

Key words: decomposition/reconstruction, reconstruction error, error correcting output codes, bias-variance decomposition

WWW: http://www.eim.surrey.ac.uk/
Acknowledgements

One who does not thank people, does not thank God

prophet Mohammed

But to whom shall I start?! 
Imam Ali-bin Abi-Taleb is saying;

One who teaches me a single word, is making me his/her slave

So first of all, I am deeply grateful to my supervisor Dr. Terry Windeatt. Not only for being a good supporter, advisor and criticiser but also a good friend over the past three years.

My gratitude goes also to the manager team and the members of the Centre for Vision, Speech, and Signal processing (CVSSP) for providing such a great environment for research.

I have many reasons to thank my wife Farzaneh. In this particular case I can remember her worries about me when I was engaged my research. Living with a researcher is not always easy, now in this stage she may think “shall we relax now?”

I am afraid, this is not even “the beginning of the end”, I hope it could be “the end of beginning”.


Contents

1 Introduction

1.1 Neural networks in classification tasks
   1.1.1 Automatic classification
   1.1.2 Neural networks approach
   1.1.3 Learning
   1.1.4 Generalisation

1.2 Motivation for combining neural networks

1.3 Homogeneous classifier combining

1.4 Approach to the problem

1.5 Guide to the Dissertation

2 Combining classifiers: A literature review

2.1 Introduction
   2.1.1 Main requirements in combining

2.2 Correlation Reduction strategies
   2.2.1 Different learning machines
   2.2.2 Different representation of patterns
   2.2.3 Partitioning the training set
   2.2.4 Different labelling in learning

2.3 Mathematical frameworks of Combining
   2.3.1 Input categorising
2.3.2 Function categorising ................................................................. 24
   2.3.2.1 Linear Combining .............................................................. 24
   2.3.2.2 Non-linear combining ........................................................ 28
2.3.3 Task categorising ........................................................................ 30
2.4 Theoretical base on the effectiveness of combining ....................... 30
   2.4.1 Regression viewpoint ............................................................ 31
   2.4.2 Classification viewpoint .......................................................... 32
      2.4.2.1 Investigating the effect of variance on misclassification .... 32
      2.4.2.2 Generalisation of bias-variance decomposition for 1/0 loss function ................................................................. 33
      2.4.2.3 Margin criteria ................................................................. 35
2.5 Summary of chapter ......................................................................... 36

3 Label decomposition: ECOC code matrices ....................................... 38
   3.1 Introduction .................................................................................. 38
      3.1.1 Decomposition strategies ...................................................... 39
   3.2 Motivation for label decomposition .............................................. 40
      3.2.1 Simplifying the problem ........................................................ 40
      3.2.2 Label separation ................................................................... 40
         3.2.2.1 The effect of decomposition matrix ................................ 42
   3.3 ECOC: General idea ....................................................................... 43
      3.3.1 Error detection ..................................................................... 43
      3.3.2 Error correction ................................................................... 44
      3.3.3 Consideration in information theory ...................................... 45
      3.3.4 Error correction limits in ECOC ............................................ 45
   3.4 Code generation methods .............................................................. 46
      3.4.1 Existing methods .................................................................. 46
   3.5 ECOC in classification ................................................................. 49
      3.5.1 Classification considerations in code generation .................... 52
   3.6 Summary of chapter .................................................................... 53
5.2.1.1 Base classifiers .............................................................. 96
5.2.1.2 Selection of code matrix ................................................. 97
5.2.1.3 Reconstruction error ....................................................... 98
5.2.1.4 Effect of common error (biased measurement) in experts ...................................................................................... 99
5.2.1.5 Effect of variance ............................................................ 101
5.2.1.6 General cases ................................................................. 102
5.2.1.7 Circular ECOC ............................................................... 103
5.3 Variance reduction ............................................................... 104
5.4 Visualising decision boundaries .............................................. 106
5.5 Experiments with real problems with MLP ................................ 111
  5.5.1 Random Codes performance ............................................. 111
5.6 Summary of chapter ............................................................. 118

6 Fusing ECOC with other methods ........................................... 120
6.1 Introduction ............................................................................ 120
6.2 Input decomposition ............................................................. 123
  6.2.1 Local error back propagation .......................................... 124
6.3 Boosting Binary classifiers ...................................................... 125
  6.3.1 Traditional Boosting ....................................................... 127
  6.3.2 Adaptive Boosting .......................................................... 127
  6.3.3 A quick Summary of AdaBoost ....................................... 129
  6.3.4 Some Notes about AdaBoost .......................................... 131
  6.3.5 Examples of dynamic weighting ....................................... 132
6.4 Experimental results on fusion of these methods ..................... 136
  6.4.1 Experiments on Face4 face data base ................................. 137
    6.4.1.1 Preprocessing .......................................................... 137
    6.4.1.2 System description .................................................. 138
    6.4.1.3 Discussion ............................................................. 139
  6.4.2 The ORL face data base ................................................. 140
6.5 Summary of chapter ............................................................. 146
# Summary and Conclusion

## 7.1 Introduction

## 7.2 Contributions

## 7.3 Summaries of chapters

## 7.4 Conclusion and recommendations on future works
List of Figures

1.1 Block diagram of a fully connected feed forward neural network with two hidden layers ........................................... 4

2.1 Block diagram of a composite classifier ........................................ 15
2.2 Block diagram of mixture of experts combining method ............ 19
2.3 Block diagram of Linear combining ............................................. 25

3.1 Error reduction due to error correction capability for $\epsilon = .3$ ........ 46
3.2 The table of accepted values of added bits and correction capability 55

5.1 Decision boundaries of multi-class Bayesian classifier ............... 97
5.2 Decision boundaries of C1 (a) individual boundaries (b) composite boundaries .................................................. 107
5.3 Decision boundaries of C4 (a) individual boundaries (b) composite boundaries .................................................. 108
5.4 Decision boundaries of ECOC in the presence of Gaussian noise with sigma=.25 (a) C1 (b) C4 ................................. 109
5.5 Decision boundaries in the presence of Gaussian noise with sigma=.25 (a) ECOC (b) circular ECOC .......................... 110

6.1 Block diagram of a composite network using separation of feature set ...................................................................... 125
6.2 Error reduction in learning (a) single net. with 70 hidden nodes (b) 4 net. with 10 hidden nodes ................................. 126
6.3 Decision boundaries: a) individual experts b) composite classifier with optimum fixed weighting factors c) composite classifier with dynamic weighting factors .................................................. 134
6.4 Comparison between decision boundaries of AdaBoost and Dynamic-AdaBoost .................................................. 135
6.5 images of a subject from “Face4” face data base ................. 138
6.6 The rise of performance in “Face4” face data base .................. 139
6.7 OLR face data base at a glance ........................................... 141
6.8 The rise of performance by increasing the number of binary clas­
sifiers in ECOC and AdaBoost+ECOC a) for images with size of
8×8 b) for images with size of 12×12 ............................................. 143
6.9 The rise of performance by increasing the number of binary clas­
sifiers in ECOC and AdaBoost+ECOC a) for images with size of
24×24 b) for images with size of 32×32 .............................................. 144
List of Tables

4.1 Assignment table of ECOC and Centroid method .................. 75

5.1 Distribution parameter of data used artificial benchmark ........... 96

5.2 Matching rate (%) of different methods with Bayesian classifier when there is no noise .......................................................... 99

5.3 Matching rate(%) of different methods with Bayesian classifier in the presence of Gaussian noise with mean of .5 and zero variance 100

5.4 Matching rate (%) of different methods with Bayesian classifier in the presence of Gaussian noise with mean of zero and variance of .5 101

5.5 Matching rate (%) of different methods with Bayesian classifier in the presence of Gaussian noise with the mean of .2 and the variance of .5 .......................................................... 102

5.6 Comparison of Centroid and circular centroid(second rows) in the presence of Gaussian noise ($\sigma=.5$, mean of .2) .................. 103

5.7 Comparison of ECOC and circular ECOC(second rows) in the presence of Gaussian noise ($\sigma=.5$, mean of .2) .................. 104

5.8 Comparison of least squares and circular least squares(second rows) in the presence of Gaussian noise ($\sigma=.5$, mean of .2) ........... 105

5.9 Databases used in this experiment ........................................... 112

5.10 Specification of problems ....................................................... 112

5.11 Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over "zoo" data base .................................................. 113

5.12 Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over "car" data base .................................................. 114

5.13 Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over "vehicle" data base .................. 114
5.14 Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "glass" data base ............................................................ 115

5.15 Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "satellite" data base ...................................................... 115

5.16 Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "Gaussian5" data base ....................................................... 116

5.17 Mean and standard deviation of correct classification rate of MLP with given number of hidden nodes with ECOC and circular ECOC 117

6.1 distribution parameter of five group of data used in training and test set .................................................. 133

6.2 classification rates (%) for 300,500 and 1000 pattern in training set 133

6.3 Classification rate (%) on artificial database 20 independent run and 10 experts -100 sample for training and 900 sample for test set 135

6.4 Classification rate (%) on diabetic database 20 independent run and 10 experts -570 sample for training and 214 sample for test set 136

6.5 Classification rate of combining experts made by Boosting .... 136

6.6 correct classification rate of ECOC on ORL face data base. .... 145

6.7 correct classification rate of local experts of ECOC on ORL face data base. .................................................... 147
Chapter 1

Introduction

1.1 Neural networks in classification tasks

1.1.1 Automatic classification

Computers help us to solve many problems, their electronic elements operate much faster than our brains and their memory management is much better, but surprisingly, in many cases they can hardly perform some of our simplest actions which are categorised as "cognition" tasks, such as recognition of the face of a person from an angle not seen before. This fact resulted in a quest by scientists to create an artificial mind, a machine which operates like a human in these cases. One of these tasks is "classification" which can be explained as:

*assigning a given pattern to one of several possible classes*

An implicit assumption in this task is that each sample has some "essential properties" which are common to all members of a certain class, and some "individual properties" which distinguish a pattern from other members of the class to which it belongs. Using essential properties we can classify a pattern.

Practically, in most cases these essential properties are unknown, and all we have is a set of patterns with their class information. An automatic classification system should extract these essential properties.
On the other hand, patterns can be of different kinds. To use computers for classification we need to show patterns by appropriate symbols. If we consider recognition system as a block, then representation of patterns and class information (class label for each pattern) can be considered as the environment. In this analysis this block should be capable of:

- Presenting the knowledge, which is divided in two categories:

  1. Representation of patterns, which means using symbols to represent the patterns (input) and class information (output).

  2. Reasoning or inferring, which means the way of responding to the information coming from environment (input) in a certain mathematical or logical way and producing the output.

- Learning, which means modifying some parts of the reasoning (inferring) stage to simulate the environment. In other words, this means changing the response of the system to make its input-output closer to the training samples.

Hence, the design of an automatic classification system includes the following steps:

1. Finding an appropriate presentation of patterns to make possible the extraction of essential properties (feature extraction and selection).

2. Finding information of essential properties for each class as some mathematical functions fitted to the training patterns which can distinguish different classes (learning).

3. Finding the output of these functions when new (test) patterns are given to the system (inferring).

4. Assigning a given pattern to one of several possible classes based on the results of inferring stage (decision making).
1.1.2 Neural networks approach

At about the same time digital computers were emerging in the world, the idea of neural networks was introduced as a different approach of calculating, which essentially was based on mimicking the human brain. The works of McCulloch and Pitts,[49] Hebb [29], Rosenblatt [56], Widrow and Hoff [72] expand the idea of employing neural networks in processing applications. But three events are normally identified as creating a new atmosphere in the field. The efforts of Hopfield [31, 32, 33], the introduction of back propagation training algorithm[59], and the increase in computing power have made the simulation of neural networks popular and practical.

At first, most efforts were in the field of “intelligent machines”, but intelligence seems hardly possible to implement with current knowledge and technology. Designers quickly found that wrong answers had been due to asking the wrong questions. Rather than focusing on nebulous concepts such as “reasoning” and “intelligence”, efforts were gradually directed toward more practical and reachable goals such as pattern classification.

Although statistical pattern recognition methods have long been available for developing specific algorithms for given tasks, neural networks offer an alternative which can provide robust classification even when data is poorly understood or detailed processing steps are not known.

Our interest in this discussion is confined to a certain class of neural network having two main properties:

- They perform their useful computations through a process of Learning.

- Their inner units have a feed forward connection, which means they are separated in layers and the input of each unit comes only from the previous layer. There are three kind of layers in such a neural network(figure 1.1):

  1. An input layer whose inputs are the networks input.

  2. An output layer whose output are the networks output.
Chapter 1. Introduction

Figure 1.1: Block diagram of a fully connected feed forward neural network with two hidden layers

3. Hidden layers which are not input or output.

Such a neural network employs a massive interconnection of simple computing cells referred to as neurons or processing units. So we can define neural networks from the viewpoint of adaptive machines as[28]:

A neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use.

It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.

2. Inter-neuron connection strengths known as synaptic weights are used to store the knowledge.

Neural networks differ from conventional algorithms in following ways:
• The style of processing is completely different. It is parallel, includes less data manipulation, and in many cases signals do not need to be converted to numeric symbols for processing. In a conventional computer, most of the time is spent on data transfer between CPU and memory and this is why their actual speed is so much lower than neural networks.

• Information is stored in a set of weights in the whole network rather than a program that is located in a block of memory. This is a critical advantage when needed operations are difficult to be implemented in a program.

• These weights are supposed to adapt when the network is shown examples from a training set (learning phase). So the training phase may be considered as analogous to making programs in conventional computers.

• Neural networks are robust in the presence of noise and local hardwire failure, because only a local part of the network may be affected by them.

1.1.3 Learning

Among the many interesting properties of a neural network, is the ability of the network to learn from its environment and improve its performance through learning. There are different sides to this concept, for example learning viewed by a psychologist is different from learning in a classroom scene. In the context of neural networks learning has been defined as follows ([28]):

Learning is a process by which the free parameters of a neural network are adapted through a continuing process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.

In fact, in such a neural network there are two phases of actions:

1. The learning phase, in which the weights are modified by an input training set (in unsupervised learning) or pairs of input-desired output (in supervised learning).
learning which is our concern in this discussion). During the learning phase a network sets its inner weights to fit itself to the environment (training samples). This is done via a search in weight space to optimise a cost function.

2. The function (application) phase, in which the network responds to a new set of patterns (test set) and provides the outputs.

1.1.4 Generalisation

The neural networks’ capability to learn from examples should give them the ability to “predict” the outcome of a certain event not seen before, based on “experience”.

This capability which provides an expected performance over data not seen before is called generalisation. In fact without generalisation, learning is nothing more than making a look up table, that is not really useful in most applications.

An underlying assumption in this discussion is that the “training set” is “representative” of the “test set” so there are some “relationships” between them. For example they can be extracted from the same source with certain distribution.

In classification task, if we consider each pattern as a point in the “input space”, each class has its own region, in which patterns most likely belong to this class. These regions may have overlap based on pattern representation. Relationships exist between classes and partitions and generalisation may be regarded as the ability to extract the rules of these relationships from the training set, usually not explicitly defined.

In training, learning machines provide functions which can simulate the “boundaries” between these areas (decision boundaries). When a test pattern is applied to the trained system, its position in input space will be determined; then using the learned boundaries, the pattern will be assigned to the corresponding class.

To provide a precise boundary, we need all the points of this boundary. This means we need infinite points (training examples) which is practically impossible
in two ways. First, the number of training patterns in real world is limited. Furthermore, for huge training sets, the learning machine should be capable of using them.

If we could provide a consistent learning machine and infinite training samples we would expect a constructed classifier to generalise perfectly (like a Bayesian classifier). But in practical cases, different systems generalise differently based on a number of factors such as: structure (say for neural networks number of layers and nodes), learning paradigm cost function, initial values, feature set, training set,... .

The traditional design method is based on training a few networks with different sets of parameters, and testing them over another set of samples (validation set) to find the best possible combination of these parameters.

1.2 Motivation for combining neural networks

Although in the halcyon days of neural computing, neural networks were popularly supposed to provide good generalisation for all problems, after a while it became apparent that there are many tasks that cannot be efficiently achieved by a well trained single network. Many experiments were reported for problems that involve noisy data, large number of highly overlapped classes and limited number of training samples. They indicated that particular choices for the problem at hand relating to training set, feature set, learning algorithm structure and parameters can hardly lead to good generalisation, even if the best set of parameters has been chosen (which is generally a big if).

The proper selection of a large array of design parameters, however, is crucial for the satisfactory performance of a complex neural network which is needed for complex problems. Considering the fact that in a neural network there has been a lack of understanding on the specific inner workings, (this is why they have earned a "block box" label) it is clear that finding the perfect network for a certain complex task which means finding the best set combination of size,
structure, learning paradigm, and parameters is difficult. In fact the traditional way to find such a network which is based on trying different combinations of mentioned items, and testing the achieved networks on validation set can only show us the best tested network and not necessarily the best possible one. Some shortcoming in this strategy can be mentioned as follows:

- **Non-overlapped errors**
  Noticing the fact that mis-classified patterns in different networks are not necessarily overlapped, we can conclude that the "best" estimator is only best on average. For each classifier (even the "best" one) some patterns are difficult, which may not be difficult for other classifiers. It shows that different classifiers have extracted (and stored) different information in training phase. So there is the risk of removing parts of important information in choosing the "best" classifier and throwing aside the others.

- **Finding the best classifier problem**
  Finding the "best" classifier is not easy in all manners, particularly in neural networks. This is because of the fact that the relationship between the output of network (or its internal weight set which dictates the output) and the mentioned parameter set is very complex (and unknown in many cases).

- **Learning-Generalisation Dilemma,**
  For a given network, a large number of patterns in training phase may be learned very well but may give poor generalisation. Using too few patterns could also cause unrepresentative learning, there is a trade off between error and generalisation, and using a combination could prevent over-fitting and improve performance.

- **Regularization**
  If a truly model free estimator (a network with a large complexity) is used to minimise the mean square error over training set, the estimator may simply interpolate the data. This leads to high variance, and this is why
Chapter 1. Introduction

In the cost function, a penalty term is sometimes suggested to avoid unnecessarily complicated models (regularization). This approach may increase the number of needed patterns in the training phase, but as noted, a larger training set is not always a good alternative. Combining methods may help to solve this problem.

• Lack of Effective Learning
  Ineffective learning is another problem in some neural networks such as Multi Layer Perceptron. For example, the increase of hidden nodes does not cause the appropriate increase in learning capability, and in these circumstances combining could be a good solution for the problem.

• Different Sources of Information
  In some applications, there are different sources of information for an identical phenomena, so a high dimensional vector should be used for representation of data to include all information. An important phase in pattern recognition is to choose the feature set, because a large number of features may give rise to the curse of dimensionality, i.e. exponentially increase in calculation with respect to input dimension. Choosing too few features means wasting valuable information. Using a combination of networks with different feature sets could avoid this problem.

• Using the decomposition/reconstruction strategy
  In this way complex problems which are decomposed into simpler subproblems can be solved by a team of simpler networks. This can provide better performance with lower cost.

1.3 Homogeneous classifier combining

Although there have been many different techniques for combining classifiers, most of them heuristically designed, recently designers exhibit more interest in “homogeneous experts combining”. In this category, using a standard learning
Chapter 1. Introduction

machine (say neural network with certain number of layers, nodes and learning paradigm) different classifiers (experts) will be made. The results of these experts then should be combined to implement the final output. Two main goals of this strategy are simplifying the problem and increasing the performance.

The main advantage of this idea is the fact that in many cases repeating a simpler algorithm could be more economical than making a complex one. In fact one practical goal is:

to construct a combination of homogeneous simple networks to solve a complex problem which normally needs a complex network.

1.4 Approach to the problem

One possible strategy to solve complex problems is to reduce the complexity of the inner parts of networks, because we can not control the inner works of a complex network appropriately. This may reduce the capability of solving complex problems. So we increase controlled complexity by combining a team of simple networks.

Three main sources of complexity are:

1. Label complexity; which means that the number of possible classes is high. This causes some problem in decision making stage.

2. Input representation; which means having a huge feature set.

3. Function complexity; which means that complex functions may be needed to infer the input, so complex learning machines are needed to provide them.

So for a complex problem we address these three issues:

1. Decomposing a multi-class problem into binary sub-problems to reduce label complexity. Then the results of sub-problems should be combined to
provide a solution of original problem. We focus on the concept of error correcting output code (ECOC), an idea from information theory, whose application in classification has been introduced in [15, 16]. In this technique, new labels (super classes) are defined based on the sequence of “1” and “0” of columns of a certain matrix with binary elements (code). Although this technique has exhibited good performance in many applications, there are some unanswered questions:

(a) What is the source of effectiveness of this framework?
(b) Which decomposition can provide better performance (code matrix selection)?
(c) How should we combine the result of components?
(d) How does final error relate to component errors?

In chapter 3 we explain this technique and existing methods to create decomposition matrix and discuss the shortcomings in answering the above questions in previous works. It has been due to the lack of understanding relationship between the output of experts and final class scores needed for assigning. In chapter 4, providing a framework to explain this relationship, appropriate answers will be given to the above questions.

2. We suggest a multi-network system with local back propagation learning algorithm to reduce the input complexity of binary sub-problems (chapter 6). Such a network which has not been used with ECOC previously exhibits great improvement in performance over a complex system.

3. Boosting type methods like AdaBoost have exhibited good capability to overcome function complexity in many applications. We suggest fusing this algorithm with ECOC and multi-net system for complex problem such as face recognition (chapter 6).
1.5 Guide to the Dissertation

The structure of the thesis is as follows:

- In chapter 2 we introduce the combining problem and present a literature review of reported works on combining estimators and their different categories.

- Chapter 3 deals with decomposition of a multi class problem into a few binary ones to be solved by binary classifiers, focusing on Error Correcting Output Coding (ECOC). Existing code generation techniques are introduced and their shortcomings in classification task are discussed which are addressed in chapter 4.

- Chapter 4 starts with a discussion about previous investigations in ECOC, its evaluation and developments. We discuss that the main weak point of previous works has been ignoring the relationship between distance measurement (which is the basis of decision making in ECOC) and class scores (say posterior probabilities). Finding a theoretical framework to specify this relationship leads to:
  
  - New condition set for code selection
  - New technique to get better performance from a certain code
  - Analysis of different kinds of error including "reconstruction error", "bias" and "variance" and their effects on final performance.
  - Alternative combining frameworks

- Some experimental examples are presented in chapter 5 to demonstrate improvements achieved by considering points discussed in chapter 4.

- In chapter 6 some shortcomings of ECOC when used with complex problems are introduced and to overcome them, fusing with other techniques is investigated. A realistic application to face recognition is described.
• Conclusion and suggested directions to future works are in Chapter 7. Contributions resulting from this thesis are listed in section 7.2.
Chapter 2

Combining classifiers: A literature review

2.1 Introduction

Combining is an approach to improve the performance in classification particularly for difficult problems such as those involving a considerable amount of noise, limited number of patterns, high dimensional feature sets, and highly overlapped classes. Some reported experiments indicate that a composite system outperforms the best individual classifier (single component) while others report the contrary to be true. The main question is:

"when does a composite system outperform the single component?"

We are not proposing that we can answer this question generally in this thesis. In fact the existence of such a question arises from the fact that most of the works in this subject have been heuristic.

General structure of a composite classifier is shown in figure 2.1. Pattern to be classified (p) is represented by different feature sets (x_1, x_2, ..., x_n) which may have some common parts. Each feature set applies to an individual classifier (expert) and the output of these experts (Y_1, Y_2, ..., Y_n) are combined to build the final output Y_{com}. 
In this chapter we have a review of the problem of combining, its different parts and methods suggested to solve them and theoretical analysis to evaluate how successful a combination is.

2.1.1 Main requirements in combining

Although the design of a composite classifier has been largely heuristic until now, there are essential requirements that should be satisfied in any combination:

1. The first requirement is to have some individual classifier, that not only has an acceptable performance but also exhibits independence in decision making. In other words, we need experts which have different information and are not redundant, otherwise, they give the same results in all manners and there is hardly any benefit from combining. These experts should yield the same results from some patterns (which are correctly assigned) and
disagree upon the error cases. If some patterns are wrongly assigned by all individual classifiers, it would be hardly possible to assign them correctly by composite classifier.

Generally speaking, the parts of error that are more likely removed, are not common to all components. This is why correlation reduction between classifiers is so important, as explained in section 2.2

2. The second requirement of a composite system design is to have a suitable mathematical framework to combine the results of individual classifiers in such a way that takes their strengths and avoids their weakness.

2.2 Correlation Reduction strategies

Since classifiers are made through a training procedure, to have classifiers which generalise diversely, they should be trained differently. The training procedure can be affected by input representation of patterns, training samples, learning procedure and supervision strategy, on which correlation reduction techniques will be based.

Based on these items, different strategies to make diverse classifiers exist:

2.2.1 Different learning machines

Changing learning procedure can be done by:

1. Using different learning algorithms.
   For example we can use multi-layer perceptron, radial basis function network and decision trees in an ensemble to make a composite system [41].

2. Using certain algorithms with different complexity.
   For example using networks with different number of nodes and layers or nearest neighbour classifiers with different number of prototype.
3. Using different learning parameters.

For example in back-propagation different initial weights, number of epochs, learning rate and momentum or cost function can affect generalisation of network [74].

2.2.2 Different representation of patterns

There are three ways to perform this task:

1. Using different feature sets or rule sets.

   This method is useful particularly when more than one source of information is available([41])

2. Using decimation techniques

   Even when only one feature set exists, we can produce different feature sets [71] to be used in training the different classifiers by removing different parts of this set.

3. Feature set partitioning in a multi-net system

   This method can be useful if patterns include parts which are independent. For example different parts of an identification form or a human face. In this case, patterns can be divided into sub-patterns each one can be used to train a classifier. One of theoretical properties of neural networks is the fact that they do not need special feature extraction for classification. Feature sets can be the same real valued measurements (like gray levels in image processing). The number of these measurements can be high so if we apply all of them to a single network, the curse of dimensionality will occur. To avoid this problem they can be divided into parts each one used as the input of a sub-network[58, 57] which are independent.
2.2.3 Partitioning the training set

There are some free parameters in any learning machine which should be set during training. The final set of these parameters depend on the training set, so even for a given structure and an identical representation of patterns, different training sets could cause different generalisations. Using identical representation of patterns has the advantage that the decision boundary of individual classifiers are in the same axis set (space). Therefore the effect of each sample, or expert in composite classifier can be investigated. In these circumstances correlation reduction is based on partitioning the main training set into a few subsets, and using these partitions to train different experts. If different partitions are separated (non-overlapped), the independence of classifiers will be increased, but in most practical cases because of limited number of training samples, these partitions made by perturbing the original training set, have some overlap.

There are a few methods for such a partitioning of the training set, some of the more popular are as follows:

1. **Random partitioning** such as:

   - **Cross validation**, [21] is a statistical method in which the training set is divided into $k$ subsets of approximately equal size. A network is trained $k$ times. Each time one of subsets is "left-out", so changing the "left-out" subset we will have $k$ different trained networks. If $k$ equals the number of samples, it is called "leave-one-out"

   - **Bootstrapping**, in which the mechanism is essentially the same but subsets (patterns) are chosen randomly with replacement [18, 36]. This method has been used in the well known technique of Bagging (Bootstrap Aggregation) [9, 53, 7, 46]. It has yielded significant improvement in many real problems.

2. **Partitioning based on spatial similarities**
Chapter 2. Combining classifiers: A literature review

- **Mixture of Experts**, is based on the divide and conquer strategy [35, 34], in which the training set is partitioned according to the spatial similarity rather than random partitioning in the former methods. During training, different classifiers (experts) try to model different parts of the input space.

![Figure 2.2: Block diagram of mixture of experts combining method](image)

There is a *competitive gating* mechanism that localises the experts, using a gating network (figure 2.2).

3. Partitioning based on experts’ ability

- **Boosting**, is a general technique for constructing a classifier with good performance and small error from some individual classifiers which have performance just a little better than free guess.[17]

Here, the main idea is to create conditions in such a way that “difficult cases” have more chance to appear in the training set of some
individual classifiers.

In the original versions of boosting ([17]) the training procedure is as follows:

- Train the first classifier (R1) on a subset of training data with the original distribution of $D_1$.

- Filter another subset of original data (with distribution of $D_1$) through R1 to generate a new subset with distribution of $D_2$, in such a way that 50% of its members are correctly classified cases and 50% of errors, upon which the second classifier (R2) should be trained.

- Filter another subset with distribution of $D_1$ through R1 and R2 to generate a new subset with distribution of $D_3$ consisting of the patterns upon which two classifiers disagreed. The third classifier (R3) must be trained over this subset.

In operating mode, each pattern is shown to R1 and R2, if their assignments agree it is accepted otherwise the decision of R3 would be chosen.

• AdaBoosting, is the new version of Boosting in which the number of individual classifiers is increased. In each iteration the chance of appearance of “hard” samples in training of next classifier will be increased. The results of classifiers then will be combined in a weighted voting structure[19, 20, 63]. Other approaches of this idea has been suggested [54, 11, 10] as well. Adaboosting is further discussed in chapter 6 section 6.3.

In these methods the main idea is to give more chance to “difficult patterns” to appear in the training set of some experts in order to specialise them to distinguish similar instances.
2.2.4 Different labelling in learning

As mentioned earlier, in classification a given pattern $x$ should be assigned to one of several possible classes. Based on this fact, in supervised learning, pairs of input-desired target are used. So changing desired target can change the environment. Two main methods have been introduced in this way:

1. Randomising the outputs[12]

   (a) In regression, *smearing* is used in which first we estimate the standard deviation of the output($\sigma$). Then Gaussian noise with zero-mean and standard deviation of $\sigma$ will be added to each output.

   (b) In normal classification, a class label will convert into a binary vector (one per class). For example if the class label of a pattern for a 5 class problem is 2, then its label will be a 5 element vector in which all elements are “0” but the second one which is “1”. Now in training we can add Gaussian noise with zero mean and the standard deviation twice the sample standard deviation of that particular output. This method is called *output flipping*.

2. If we redefine the classes for example by giving the same label to a group of classes, the classifiers which are made by different defining of classes can generalise diversely. Methods like ECOC [15, 16], binary clustering [73] and pairwise coupling [27] use this method.

Binary classification interests us because:

- Some accurate and efficient algorithms for solving 2-class classification problems don’t naturally scale up to multi-class (e.g. perceptrons, Bayesian classifiers and logical analysis of data). Furthermore some useful algorithms which make the classifiers more capable are designed for binary classifiers( like Boosting type algorithms). Although some attempts were made to scale them up, their performance on binary cases was better.
• Binary classifiers have some advantage in software and hardware implementa­tion because of their similarity with binary logics.

• There are mechanisms in other fields of data processing (like information theory) which can be employed in conjunction with binary classification in order to improve the performance of the system.

To use this technique, the designer is faced with two main questions:

(a) How can we decompose the main problem into sub-problems to solve them by a binary base classifier? in other words, which set of sub-problems is suitable?

(b) How should the results of sub-problems be combined to achieve better performance? It is obvious that the reconstruction method should be appropriate for decomposition, otherwise some other errors may occur.

2.3 Mathematical frameworks of Combining

Having a set of individual classifiers (experts), the second step is to make a composite system by combining the output of these experts in a suitable mathematical framework. The goal is to achieve a better performance. No one could say which framework is the "best" because it is based on the problem and other practical considerations. However they can be categorised based on the methods used in different parts of the combining procedure.

2.3.1 Input categorising

The input of combiner block (figure 2.1) comes from the output of components. The procedure of combining will be based on the type of this input. Let us review the problem of learning in supervised mode. Supervised learning is based on extracting knowledge from a set of examples like \( S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), where \( x_i \) are input patterns, typically vectors of the form \( x_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T \),
whose components are discrete or real valued features such as weights, colour, age, etc. In other words input set can be shown as a matrix $\mathbf{X} = \{x_i\}$ where $i$ indicates a certain sample and $j$ indicates certain features.

The $d_i$ values on the other hand are desired targets. In a classification task, $d$ contains class(label) information, discrete sets of labels each one belongs to one of possible classes. For example “0” and “1” or “1” and “-1” can be used to show if a pattern belongs to a class or not.

In most cases, applying a pattern $p$ to a trained classifier (the $j$th one), results in a numerical score $Y_j = (p)$. Then through a binarizing procedure it makes a hypothesis $C_j(p)$, a vector contains its judgement about the class of the pattern.

So there are three different kinds of information available from the output of individual classifiers (figure 2.1):

1. Soft level information, the output of inferring stage which are normally real numerical scores ($Y_j$).
2. Hard level label information, The final outputs of classifier which are discrete values ($C_j$).
3. Rank information, which shows the relative position of classes for each expert.

From this viewpoint combining techniques could be divided into three categories:

1. **Hard-level combining**, in which, the assigning results of individual experts are used as input to the combiner. The advantage of these techniques is their simplicity. Considering the experts as "black boxes" we don't need to be concerned about the classification procedure. On the other hand, this method can combine the results of classifiers which produce different types of scores in the inferring stage. Voting schemes lie in this category. At first glance it seems that when using this strategy only patterns can be assigned correctly if at least a minimum number of experts are correct.
This is due to the fact that these methods do not combine scores which can be interpreted as the confidence of reasoning. But recent analysis states the contrary [63]. Two common ways to manage the combining in order to increase the rate of correct classification are to use weighted voting and rejecting low confidence cases in individual classifiers [6].

2. **Soft-level combining**, in which the results of the *inferring stage* of individual classifiers (before making a decision about assigning) are used as the input to the combiner. The advantage of this technique is the fact that by combining the results of experts it could be possible to provide more confident assignment.

One disadvantage of this method is the fact that this technique can only combine the output of classifiers which provide the same type of scores in their inferring stage.

3. **Rank based combining** is another approach to combining. [2, 30]. In some applications that use different sources of information, it is necessary to employ different kinds of classifiers, which provide different numerical scores such as: *distances to prototypes, values of an arbitrary discriminant, estimates of posterior probabilities and confidence measures*. In these circumstances, the rank of classes in the output vector of each classifier, have more meaningful information and are more appropriate for combining. This method is particularly good for the problems which have a very high number of classes such as word recognition.

### 2.3.2 Function categorising

From this viewpoint combinations would be divided into:

#### 2.3.2.1 Linear Combining

The most popular method for combining is Linear combining, due to its simplicity and good properties. There is a linear relationship between the input and
the output of a linear combiner (and not the input and the output of composite classifier).

The block diagram of such a system is shown in figure 2.3, in which the phenomenon to be classified is presented with some feature set (or rule set), that may be overlapped. The final output is a linear function of the experts outputs:

\[ Y(x) = \sum_{i=1}^{n} W_i(x) \cdot Y_i(x) \quad ; \quad \sum_{i=1}^{n} W_i = 1 \]

Here \( Y_i \) is the output vector of \( i \)-th expert, \( W_i(x) \) is the weighting factor of this expert and \( Y \) is the final output vector.

The design procedure is based on making a weighting factors space and search to find an optimum by defining a cost function. Here there are few methods to find the appropriate weighting factors set:

1. Let us consider general case of linear combining when experts provide the
probability of class memberships. Given a certain set of weighting factors, a joint probability can be estimated:

\[ \hat{y}(w_{ij}|x) = \sum_{j=1}^{m} w_{ij}(x) \hat{y}_j(w_{ij}|x) \]  

(2.1)

Where \( \hat{y}_j \) is the output of the \( j \)th (hard level for voting and soft level for averaging), \( \hat{y} \) is the output of the ensemble (composite estimator) and \( w_{ij}(x) \) is the appropriate weighting factor which is a function of classes (i), experts (j) and input (x).

Here there are a few strategies to find such a weighting factor set:

(a) *on line calculation* In most cases, particularly when the number of experts is too high, the natural way is selecting the factors based on the performance of the expert without considering the behaviour of others [40, 26]. The main advantage of such a strategy is its simplicity. To shed light on this problem let us consider a soft level combining (averaging) in which the goal is to estimate the probability of class membership of each class for a given example \( x \):

\[ \hat{p}(w_i|x) = \frac{1}{m} \sum_{j=1}^{m} \hat{p}_j(w_i|x) \]

where \( \hat{p}_j(w_i|x) \) is the estimation of probability of class \( j \) membership provided by the \( j \)th expert. If the experts are independent and the variance of experts' estimation are \( \sigma_j(w_i|x) \), a natural weighting factor selection can be made based on the values of variance:

\[ \hat{p}(w_i|x) = \frac{1}{\sum_{j=1}^{m} \sigma_j(w_i|x)} \sum_{j=1}^{m} \hat{p}_j(w_i|x) \]

In such a selection more accurate (less variance) experts are given higher weights. A practical problem here is the fact that we can have an estimate of error (\( e \)) but we have no precise information about its distribution so \( \sigma \) is unknown.
In some techniques the weighting factor selection is closely related to the selection of the training samples to improve performance of composite system. For example in AdaBoost a logarithmic function of error $\log \frac{1}{\sigma}$ is used as weighting factor. This technique is based on choosing the weight of appearance probability of samples in training set and weighting factor of the experts made by this training set in which the final performance should become maximum and the necessary number of expert minimum.

(b) optimum weights in pool The second strategy is based on competition between experts on a set of examples[40]. Assume that in using equation 2.1 the following conditions are satisfied:

i. Each expert has the same performance on all classes. In other word the number of hard patterns in all classes are the same. In this case we can say that $w_{ij}(x) = w_j(x)$.

ii. The performance of each expert has little change in different parts of input space so it can be assumed independent of $x$, so $w_j(x) = w_j$

Subject to these conditions a search in the space of weighting factor can give us the appropriate set of weighting factors.

- If we use a soft level combining, quadratic methods like MSE can be used.
- Having few experts, an exhaustive search can be used to find appropriate weights.
- Another approach is using a genetic search to find the weighting factors.

2. Dynamic weighting factors has exhibited significantly better performance in some real and artificial problems [74, 75]. The idea of Dynamic weighting factors comes from the fact that satisfaction of mentioned conditions is not always possible. In fact in many cases the assumption of having the
same performance over the whole input space opposes our attempt to make
diverse classifiers each one being expert in a certain part. On the other hand
performance of an expert usually shows its average ability over different
parts. Having the best weighting factors over a set of examples means
this set provide the best average performance over whole set. So in some
parts it is possible that other sets exhibit better performance which means
different partitions of a set can have their own optimum weights. In dynamic
weighting factors an arbitrator dictates which weighting factor set is better
for any partition (or even any single sample) it can improve the performance.
Such an oracle can be implemented using a neural network [74, 75].

2.3.2.2 Non-linear combining

Here, a non-linear function relates the output of experts to the final output. Most
popular non-linear methods are as follows:

1. Stacked generalisation, addressed by Wolpert [78], is one of general
structures that explains the behaviour of many composite systems. The
main idea is to use a generaliser (or classifier) to correct the error of previ­
ous one. In a simple form of stacked generalisation there are two kinds of
generalisers, named level-0 and level-1, that lie sequentially, like layers in
an MLP (figure 1.1).

In training, the level-0 layer generalisers should be trained diversely. Wolpert
has used leave-one-out cross validation for correlation reduction between
them, but other techniques may be used also. The level-1 classifier should
learn what the correct output is, for each specific arrangement of level-0
outputs.

In operation mode, the pattern is given to level-0 classifiers and a vector is
constructed from their outputs which is used as the input to level-1 classifier,
which in turn provides a decision.

The main distinction between MLP and stacked generaliser appears from
the type of information passed from the input layer to the succeeding layer. In a neural network the outputs of each layer may (or may not) have a certain interpretation. Based on the main motivation of neural networks, they are made of simple computational units. In a stacked generaliser on the other hand, each of its components can be a complex system by itself.

The main point in this discussion is the fact that this framework includes many popular designed composite systems. For example, ECOC in chapters 3 and 4 can be assumed as a special form of stacked generalisation in which level-0 classifiers are binary classifiers made by different learning machines and level-1 classifier is a nearest neighbour classifier (in original version) [67]. Such an explanation helps designers to decompose many popular systems in order to develop them, and that is what we have done in this thesis.

2. **Order statistics combining**, is one of the most popular types of combining because of its relative simplicity. The main point in this technique is the fact that it originally came from regression and the analytical evaluations can not be used for classification directly. Some theoretical attempts exist, Kittler et. al show that these combiners can be interpreted as an approximation of generalised Bayesian decision rule in estimation of probability of class memberships [39, 41, 40], and Turner and Ghosh [69, 71, 70] have provided a framework to evaluate this framework from classification viewpoint.

3. **Belief-based methods**, for knowledge based systems, various methods of integrating evidence have been developed. For example, *Dempster-shafer theory* provides not only an answer to each query, but also a numeric value representing the belief of classification. These methods can be used to combine the results of experts by decoupling the classifier output from the confidence that it has, for this output. In these circumstances not only results but also beliefs could be combined, so aiming at separating ignorance and uncertainty, the final answer can use as much available data as possible [55]. For ECOC in which binary experts should be combined the
implementation of techniques like Dempster-Shafer theory is simpler as we will show in coming section 4.7.

2.3.3 Task categorising

From this viewpoint we can divide the existing methods into two categories[66]:

1. In some composite systems, given a pattern $x$ all the components (individual classifiers) are faced with the same problem.

   In fact a combination of "redundant" classifiers is proposed. Terms like ensemble, or committee or committee machines have been used for such a system. In these circumstances any expert provides a solution of the problem and the goal of combining is to improve the accuracy of solution or the efficiency of system.

2. In another approach, a complex problem will be changed to some sub-problems. For example a few two-class problems will be made from the multi-class problem at hand, each one should be solved by an expert. This technique has been named the modular approach in which the output of each expert by itself can not provide a solution of the original problem, but a combination of them can[73, 27, 16, 24, 77, 76, 44].

In both the mentioned approaches each expert or component can be a composite classifier by itself.

2.4 Theoretical base on the effectiveness of combining

To explain how combining can improve the performance, different viewpoints have been introduced:
2.4.1 Regression viewpoint

Since the decision making procedure is based on the soft-level output of experts, if the estimator (regression machine) provides a more precise output, the final performance will improve. So to evaluate a composite classifier, we can see how close soft level output is to the output of a perfect estimator.

The main framework of such an analysis is based on the idea of Bias-Variance decomposition [5]. Let us assume that for a given pattern \( x \), the output of perfect estimator is \( p \) which is a random variable while our practical estimator provides an estimation \( y \) for this point. If we have a quadratic loss function, the expected loss is : \( \mathcal{E}[(p - y)^2] \). It has been proved that [5] this expectation can be expressed as follows:

\[
\mathcal{E}[(p - y)^2] = \mathcal{E}[(p - \bar{p})^2] + \mathcal{E}(y - \bar{p})^2
\]

where \( \bar{p} = \mathcal{E}(p) \). The first term of this equation calling Variance of \( p \), is independent of our estimation. It will disappear if for any \( x \), there is only one \( p \); Otherwise this term shows the irreducible part of the loss function.

The second part depends on our estimating. If the estimator is made by training over a training set \( D \), in the same way, it can be expressed as:

\[
\mathcal{E}_D[(p - y)^2] = \mathcal{E}_D[(y - \bar{y})^2] + (\mathcal{E}_D(y) - \bar{p})^2
\]

where the first term is the Variance of our estimation \( y \) and the second term is called the squared Bias. The main goal of design is to reduce one part (say variance) without increasing the other one. Most ensembles which are built by learning machine with low bias and relatively high variance, are constructed in such a way that their variance is reduced.

To shed light on this idea, consider a group of estimators (an ensemble) in which \( b \) different experts make estimations of the function \( y = f(x) \) as \( \{f^1, f^2, ... f^b\} \). Each estimation has some error and the mean square error is:
Now if we calculate the mean value of the estimations:

\[ f_{\text{mean}}(x) = \frac{1}{b} \sum_{i=1}^{b} f^i(x) \]

if we present the mean value of \( MSE \) of all experts in the ensemble as:

\[ MSE[Ens] = \frac{1}{b} \sum_{i=1}^{b} MSE[f^i] \]

It has been proved [51] that for independent estimators (\( f(x) - f^i(x) \) and \( f(x) - f^j(x) \) are independent for any pattern \( x \)) we have:

\[ MSE[f_{\text{mean}}] = \frac{1}{b} MSE[Ens] \]

It means \( MSE \) in regression will be reduced \( b \) times (number of experts in the ensemble) if experts are independent. For correlated experts this formula does not apply, and we can only confirm that the error will be reduced.

### 2.4.2 Classification viewpoint

In a classification task the performance is the probability of correct classification. It means that the loss function can adopt binary values ("1" or "0") for a given pattern, so there is not a simple relationship between the precision of soft level estimation and the performance. Three different attempts have been reported to solve such a problem:

#### 2.4.2.1 Investigating the effect of variance on misclassification

It is obvious that all errors (or their variance) do not cause misclassification. This aspect can be explained by the idea of decision boundary.
Chapter 2. Combining classifiers: A literature review

For a certain problem any classifier makes a decision boundary between any pair of classes (say \(i\) and \(j\)), which is the loci of all points \(x^*\), upon which the scores for the two classes provided by the classifier are equal. Because of imperfect training, the calculated loci are different from the perfect one. For example the score of classes is the same in the point of \(x^* + e(x^*)\), rather than \(x^*\) as it should be. Where the error is \(e(x^*) = b + e(x^*)\), which means there is a shift in boundary at this point. Now for the points which are located between these two loci, mis-classification will occur. To find the probability of mis-classification, all these regions should be considered. Turner has shown that with a linear approximation for linearly combined ensembles[71] (mean combiner) the variance of \(e(x^*)\) will be reduced \(b\) times if the experts are independent. Another analysis has been given for order statistic (medium) combiner as well [69].

In another attempt, Kittler [40] has shown that if the probability estimations have a Gaussian distribution, the probability of point-wise error between two classes (if others are not comparable) will be reduced by the factor of:

\[
\frac{1 - erf\left(\frac{\Delta P_{ij}(x)}{2\hat{\sigma}}\right)}{1 - erf\left(\frac{\Delta P_{ij}(x)}{2\hat{\sigma}}\right)}
\]

where \(\Delta P_{ij} = p_i - p_j\), \(\hat{\sigma}\) is the variance of local classifiers and \(N\) is the number of classifiers in the ensemble.

2.4.2.2 Generalisation of bias-variance decomposition for 1/0 loss function

Bias variance decomposition was essentially designed for squared error regression. There are attempts to suggest new definitions in classification task. Some of most popular works in this area are as follows:

1. Kohovi and Wolpert [42] define the cost function for 1/0 loss for the classifier \(C\) as:

\[
E(C) = \sum [1 - \delta(y_f, y_h)]p(y_f, y_h)
\]

33
where $y_f$ is the assignment of perfect classifier and $y_h$ is the assignment of the hypothesis made by $C$, $\delta(y_f, y_h)$ is “1” if they are equal and “0” otherwise. This function can be separated into three parts:

$$\frac{1}{2} \sum [p(y_f = y|x) - p(y_h = y|x)]^2 \quad \text{bias}_x$$

$$\frac{1}{2} (1 - \sum p(y_h = y|x)^2) \quad \text{variance}_x$$

$$\frac{1}{2} (1 - p(y_f = y|x)^2) \quad \sigma^2_{\text{noise}}$$

2. Dieterich and Kong define the error of algorithm $A$ over training sets with size of $m$ at the point $x$ when they are in an ensemble[43]:

$$Error(A, m, x) = 1 - p_j \quad \text{for} \quad c_j = f(x)$$

The ideal voting hypothesis chooses the class with the highest probability—the class with the most votes from individual assignments $f_{x_i}$:

$$\hat{f}^* = \text{Argmax}_{c_j} p_j$$

now

$$Bias(A, m, x) = 1 - \delta(f(x), \hat{f}^*(x))$$

and

$$Variance(A, m, x) = Error(A, m, x) - Bias(A, m, x)$$

3. Breiman has defined these terms using the term of “aggregated” classifier $C_A$. First he defines a classifier unbiased if $C_A(x) = C^*(x)$, where $C^*$ is the Bayesian classifier output[10].

If $U$ is the set on which $C$ is Unbiased and $B$ is the set on which $C$ is Biased on training set $T$:

$$Bias(C) = p_x,Y(C^*(x) = Y, x \in B) - E_{T}p_x,Y(C(x) = Y, x \in B)$$

and

$$Variance(C) = p_x,Y(C^*(x) = Y, x \in U) - E_{T}p_x,Y(C(x) = Y, x \in U)$$
and the predicted mis-classification will be:

\[ PE(C) = PE(C^*) + Bias(C) + Variance(C) \]

4. Tibshirani [68] has defined Bias and Variance as follows:

\[ Bias(C) = PE(C_0, C_A) = PE(Y, C_A) - PE(Y, C_0) \]

where \( C_0 \) is ideal classifier and \( C_A \) is the aggregation one.

\[ Variance(C) = PE(C, C_A) \neq PE(Y, C) - PE(Y, C_A) \]

He also defined the "Aggregation effect" quantity as:

\[ AE(C) = PE(Y, C) - PE(Y, C_A) \]

and states that this term (not variance) figure directly into prediction error.

In [22] a good comparison of these works has been provided.

### 2.4.2.3 Margin criteria

The margin of an ensemble at the point \( x \) in the predictor space is defined as:

\[ M(x) = \text{weighted proportion of classification to the correct class} - \text{maximum weighted proportion of classification to any other classes} \]

For example if the ensemble consists of 100 classifiers for a three class problem, and the correct class is class 1, and 50 classifier assigned it to this class while 30 of others assign it to class 2 and 20 to class 3, the margins of classes are 0.2, -0.2, and -0.3; if we assign the pattern to the first class the margin will be positive.

The margin of classes is always between -1 and 1, and a large positive margin can be interpreted as a confident classification.

Margin can simply relate to the classification error, if we consider it as a random variable over a given set, the probability of error is the probability of having negative margin.
The main idea of such an aspect is the fact that if we can prove the existence of relationship between the bound of margin (error rate) over training and test sets [63], providing a classifier with very low error rate upon training set (like AdaBoost) the test error rate will be low.

It has been proved that for AdaBoost this relationship exists [63] so it could be a large step towards explaining the success of Boosting.

2.5 Summary of chapter

In this chapter we provide a summary of the more popular methods in classifier combining. The lack of knowledge in combining is not due to too few reported works, [66, 65] the problem is, methods are mostly heuristically designed. To extract the strength of methods and avoid their weakness, we choose an approach to the problem of classifier combining, based on its requirements. The main requirements of combining in a general manner are:

1. Producing individual classifiers with lower level of correlation between their error. For this task we have found four basic strategies:

   (a) Using different learning machines
   (b) Using different representation of patterns
   (c) Using different training samples
   (d) Using different labelling methods

2. A mathematical framework to combine the results of individual classifiers.
   We categorised the existence of combining methods in three different ways;

   (a) The first viewpoint of categorising is based on the type of information used in combining. From this viewpoint methods can be divided into three groups:

      i. Soft level combining which uses the output of inferring stage.
ii. Ranked based combining which uses the rank of classes in the output of inferring stage.

iii. Hard level combining which uses the output of decision making stage of individual classifiers in combining.

(b) The second way of categorising the combining frameworks is based on the relationship between input and output of combiner. From this viewpoint there are two main categories:

i. Linear combining, which is more popular in applications and evaluation because of the existence of theoretical frameworks and practical consideration.

ii. Non-linear combining, includes some popular methods such as:
   A. Stacked generalisation
   B. Order statistics combing
   C. Belief based methods

(c) The third way of categorising is based on the task of composite classifier and components. In this way, methods are divided into:

i. Ensemble methods, in which all components and composite system share in the same certain task.

ii. Modular methods, in which the task of components are different.

Furthermore basic theories to explain source of effectiveness of combining classifiers have been reviewed.
Chapter 3

Label decomposition: ECOC code matrices

3.1 Introduction

In chapter 2, we mentioned that one strategy to improve the performance can be based on dividing a large-scale classification problem into a few simpler sub-problems. One possible source of complexity can be the existence of high number of possible classes. This can mean the necessity of more precise inferring to achieve an acceptable decision making. Many problems in classification, are polychotomies, i.e. the data set is organised into $k$ classes with $k > 2$. Although in a wide variety of learning machines like neural networks or decision trees, there are tools capable of handling polychotomies (direct multi-class approach), interesting advantages exist in converting a multi-class problem into a few binary sub-problems, as discussed in section 3.2. This chapter is organised as follows:

In section 3.1.1, we look at the strategies for decomposing multi-class problems and in section 3.2 discuss motivation for label decomposition. In sections 3.3 3.4 and 3.5 we introduce the idea of ECOC and existing code generating methods. Section 3.5.1 discusses shortcomings of these methods in classification task.
3.1.1 Decomposition strategies

There are two main strategies to define binary sub-problems from the original problem:

- The first strategy is based on the fact that the final decision boundary is made of decision boundaries of pairs of classes. So what we need is a team of base classifiers with each one providing the decision boundary of just a pair of classes. Given a test pattern, each base expert should compare just two classes (the probability of class-membership of these classes or distance between this point to each classes' prototypes). A combination of the results of all experts provides the final solution. Methods like *binary decision clustering*[^73], *pairwise coupling*[^27] have such a viewpoint.

- The second viewpoint is based on defining a few pairs of super-groups. Here we need a team of base classifiers, each one judging between two super-groups. *Error Correcting Output codes*[^16, 44, 38, 62] (ECOC) is in this category.

The main advantage of the first approach is the fact that in most cases sub-problems are "simpler", so a simpler learning machine can be used. But to cover the whole decision boundary, all possible pairs of classes should be considered. It means for a $k$ class problem the number of base classifiers is: $b = \frac{k^2}{2(k-2)} = k \times (k-1)$. Furthermore, the fact that some patterns do not belong to any classes in the pair, may cause some practical difficulties in training of base classifiers.

In the second approach the designer can use lower number of base classifiers ($2^b \geq k$). Defining pairs of super-groups should be directed in such a way that all classes will be covered fairly. On the other hand, sub-problems are more complex, and more capable learning machines are needed. But the ability to repeat the training of different part of decision boundary seems interesting. Furthermore, other kind of error reduction technique (error correcting) and bias cancelling which help to achieve better performance are available in these methods as described in section 3.2.
3.2 Motivation for label decomposition

Label decomposition in classification task interests designers from two different viewpoints:

3.2.1 Simplifying the problem

Binary problems which are extracted from the original multi-class problem, are usually simpler, so they are easier to solve in practice. This fact can be viewed from the following points:

1. For simpler binary problems less complex learning machines can be used. In this case, the design problem which is to select the structure and parameters of learning machine, is easier.

2. The cost of calculating (hard-ware, time, ...) is less.

3. The system can benefit from parallel processing capability.

4. There are some soft-ware and hard-ware advantages for binary classifiers due to the similarities with binary logics.

5. The local classifiers are more independent even when the same training set, learning machine and feature set are used. So the combining can be more effective.

6. There are a variety of algorithms for constructing the binary classifiers or improving their performances which are not naturally scaled up for multi-class problems.

3.2.2 Label separation

For a certain problem, a classifier should provide a decision boundary to distinguish the regions of different classes. If the classes are more separable in this
space, it will improve the performance. So the idea is to transfer the problem into the space in which classes are more separable. Let us shed light on this idea by looking at the popular mechanisms of assignment. To assign a given pattern \((x)\) to one of \(k\) possible classes, the inferring block of the classifier evaluates a function, \(F(x)\), which has been found in training phase. Assigning will be based on this evaluation. There are different possibilities in this case:

1. In **one-per-class** labelling, the label of classes are \(k\) dimensional vectors with binary elements: \(C_i = [c_{i1}, c_{i2}, \ldots, c_{ik}] = [c_{ij}], \quad j = 1, 2, \ldots, k\), where:

\[
    c_{ij} = \begin{cases} 
        1 & \text{if } i = j \\ 
        0 & \text{otherwise}
    \end{cases}
\]

The inferring stage provides an evaluation of the function \(F\) which has been learned in training phase: \(F(x) = [f_1(x), f_2(x), \ldots, f_k(x)]\). Each element of this vector \((f_j(x))\) belongs to one of classes \((j)\). The final decision will be made based on a comparison between these elements to output a hypothesis \(\mathcal{H} = [h_1, h_2, \ldots, h_k]\) with binary elements. If classifier assigns \(x\) to class \(i\), The \(i\)th element of \(\mathcal{H}\) will be “1” while all others are “0”.

In other words, the decision making is based on matching between \(\mathcal{H}\) and labels \((C_i, \text{for } i = 1, 2, \ldots)\). The weak-point of this method is the fact that the system can not detect the error which leads to wrong assignment. This is because the number of possible \(\mathcal{H}\) equals the number of labels and any change in elements of \(\mathcal{H}\) cause its match with another label which means wrong decision making.

2. **Distributed output code** is an alternative approach in which the number possible \(\mathcal{H}\) can be more than the number of classes. In these circumstances, changing some of elements of \(\mathcal{H}\) will produce a vector which does not match with any of labels. It makes the system able to detect the occurrence of error. The bit values in this method can be meaningful ([64]) or not ([16, 24, 77, 76, 44]).
Chapter 3. Label decomposition: ECOC code matrices

If the number of bits for each label \((b)\) is sufficiently greater than number of classes \(k\), it will be possible to remove parts of errors due to imperfect generalisation. In other words we may convert the problem into the space in which classes are more separable. The main condition to use such an advantage is the independence of bits errors in \((H)\). Otherwise when an error occurs in one bit, it may change other bits simultaneously in such a way that the changed output matches to another label. Note that in a learning machine which provides multi-class assignment (say neural network), in training, all the output units have been driven by the same cost function, so the error of output units are not independent. So we need to decompose the multi-class problem into some binary sub-problems.

This idea leads us to the concept of Error correcting output coding.

3.2.2.1 The effect of decomposition matrix

To redefine classes into new super-classes a decomposition matrix (code) will be used. The set of binary sub-problems is defined based on sequence of “1” and “0” in columns of this matrix while its rows are the new label set.

This matrix can affect the system through three procedures:

1. Defining new space into which the original problem is mapped. If classes in the new space are more separable, it improves the performance. Generally speaking separability of classes in new space is related to the problem and the basis vectors (rows) which define the space. So having greatest possible difference between rows of the code matrix may to lead better performance.

2. Experts’ independence is based on difference between columns because they are trained by them. More independence in columns (super-classes choosing in different iterations) cause more diverse experts.

3. Reconstruction of final solution is based on combining method and code matrix. Consider the case that experts provide the ideal result for binary sub-problems. A system is reconstructible if combination of these results
Chapter 3. Label decomposition: ECOC code matrices

provides an ideal multi-class solution. It depends on both combining framework and code matrix. Having a certain combining method, code matrix should be selected in such a way that minimises the reconstruction error.

3.3 ECOC: General idea

ECOC is an information theoretic concept which suggests that there may be some advantage to employing error-correcting codes to represent different signals which should be distinguished from each other when they are corrupted after passing through a transmission channel[52]. The main idea is based on adding some redundant cases in the possible output set which do not match with any of the acceptable labels. If one of these cases appear in the output, the system realises the occurrence of error.

Let us shed light on the main concepts of this idea by an example.

3.3.1 Error detection

Assume that we want to transmit a binary variable (x) through a transmission channel, the level of electronic signal being changed because of the presence of noise (in general). We can represent this signal with one bit, and decide if the received signal is more than a certain threshold, we will assign it to class "a" and otherwise "b". When noise changes the level of signal in such a way that it violates the threshold, we will have error in assignment.

If we represent the signal in more (say three as example) bits, it would increase the redundancy of system and lower the efficiency but it can provide us with the ability of error detection.

As an instance assume we show "a" by [0 1 1] and "b" by [1 0 0]. These labels have three bits distance from each other in Hamming measurement. After transmitting three bits via the channel separately, if just one of signal levels change by the noise and violates the threshold, we can detect it and do not attempt wrong
assignment. For example if we receive [1 0 1], it is not an acceptable signal so we can realise the occurrence of error (even if we do not know what the original signal was). This is true when two bit errors occur. The only case in which the error won't be detected, is the case that three errors happen. But the probability of such a case is very much lower than the other cases.

3.3.2 Error correction

For the example in 3.3.1 two possibilities exist:

1. The original signal had been [0 1 1] ("a") and two errors have occurred, (the two first bits have been changed).

2. That original signal was [1 0 0] ("b"), and one error has occurred (the third bit has been changed).

If the probability of error for a certain bit is \( \sigma \) (equal for all bits), the probability of two bits error is \( \sigma^2 \) (when they are independent), which is much lower if \( \sigma \ll 1 \), so it can be ignored.

Assigning the received signal to class whose label has minimum distance, one bit error can be tolerated. In other words this label set has the capability of one bit error correction.

In general, if signals are presented by code words with the length of "b", each of them having a guaranteed difference (in Hamming distance) "2d + 1" to any other, assuming the transmission error for each bit is independent:

1. We can recognise the occurrence of error if the number of wrong bits in receiver is less than "2d + 1".

2. Furthermore if the number error bits is less than "d", the probability of error occurrence can be much lower by assigning the received signal to the label which has minimum difference with it.

In other words such a code has the capability of "d" bits error correction.
3.3.3 Consideration in information theory

From the viewpoint of information theory, there are two main considerations to select a code matrix;

- **error correction capability** which is due to the Hamming distance between pairs of rows (row separation).

- **error independence** is a second consideration. If the error of bits are highly correlated, the above procedure cannot correct it. We have little knowledge about the cause of error and its relation to the value of signals, but if the sequence of “1”s and “0”s in different columns are different, we can say that the error of different bits is more independent. This leads us to the second consideration of code production, column separation.

There are a few methods to find a set of code words (code) with a guaranteed minimum distance between any pair, the most popular being the BCH codes[16].

3.3.4 Error correction limits in ECOC

Apparantly for better performance we need greater distance between pairs of code words, but now we show that error reduction due to error correction capability is limited in two ways:

1. Consider two binary vectors of the length of \( b \) which have \( 2d + 1 \) different bits. When we transmit them through a channel bit by bit, an error in distinguishing will occur if the number of wrong bits of different parts is more than \( d \) (error on the common bits does not affect error in distinguishing). If the probability of wrong bit for all bits is \( \epsilon \), the probability of wrong assignment will be:

\[
\epsilon_{ECOC} = \sum_{j=d+1}^{2d+1} \frac{1}{2^{(j-1)}} \times \frac{(2d + 1)!}{j!(2d + 1 - j)!} \epsilon^j
\]
This formula shows that for a given $\epsilon$, when we increase $d$, at first $\epsilon^{ECOC}$ drops quickly, but after a while a saturation region appears in which increasing $d$ can not cause significant change in $\epsilon^{ECOC}$, as shown in figure 3.1.

![Figure 3.1: Error reduction due to error correction capability for $\epsilon = .3$](image)

2. For a given number of classes $k$, the maximum number of different columns is $2^k$. Now for a given length ($b \leq 2^k$) the probability to have $2d+1$ different bits is reduced when we increase $k$.

### 3.4 Code generation methods

#### 3.4.1 Existing methods

In information theory, the main considerations of code selection is to have good separation between labels. The primary required information for this step includes:
Chapter 3. Label decomposition: ECOC code matrices

1. The length of code words \((b)\)

2. The difference between codewords \((d)\)

3. The number of transmission signals or classes \((k)\)

Four reported methods to generate a group of codeword with good separation are\[16\]:

1. **Exhaustive generation**

   This method is simple but effective. If the number of classes is less than seven \((3 \leq k \leq 7)\) the length of codewords will be: \(b = 2^{k-1} - 1\). For the first row, we put “1” for all elements. In the second row, the \(2^{k-2}\) first elements are “0” and others are “1”. The third row consists of \(2^{k-3}\) “0” elements followed by \(2^{k-3}\) “1” and \(2^{k-3}\) “0” and \(2^{k-3} - 1\) “1”. All other labels will be built in the same way. The codeword separation (difference between any pair of rows) in this group is: \(d = \frac{b}{2}\).

2. **Column selection from exhaustive codes**

   For the cases that include higher number of classes \((8 \leq k \leq 11)\), we have to use longer codewords, the number of possible codewords is rising exponentially, but not the number of classes, so we have to pick up some code words. It can lead to shorter columns so the column separability will be limited. Here we should choose the rows which make more separated columns. This requirement may be satisfied by a search.

3. **Randomised hill climbing**

   Random search is another way to produce a code, but simple random search (or genetic search) for finding a set of codewords with a minimum distance is not simple. An alternative approach is based on starting with a random code and changing the elements in such a way that improves its row and column separation. As an instance we will find the intersection points of two closest rows and two closest rows. We should change the values of these points in such a way that increases the distance between rows and columns.
Chapter 3. Label decomposition: ECOC code matrices

Repeating this procedure finally we may have a codewords set with good separation in rows and columns.

4. Bose & Ray-Chaudhuri; Hocquenghem (BCH) code The BCH algorithm employs algebraic methods to generate a code matrix with minimum distance between any pair of rows. This method is based on adding certain sequential of binary values ("1"s and "0"s) to BCD numbers [47]. In this method the length of codewords and number of codewords are always a power of two. So for most classification problems we need to shorten the columns (pick up rows). It causes some problem in column separation. In this algorithm information needed in advance are number of signals (classes) and error correction capability (T) or minimum distance between codes and length of codewords (N). The following steps are used:

(a) Find the needed number of added bits from the table of BCH codes (K).

(b) If the number of classes (k) is more than $(2^K - 1)$, we have to have less error correction capability or use longer codewords (figure 3.2).

For example if we want to use codewords with 15 bits, we can cover $2^{11} - 1$ classes if we need one bit error correction (three bits minimum distance between any pair of codewords). For 2 bits error correction (five bits minimum distance) and the same length of codewords only $2^7$ classes can be covered. And for three bits error correction only $2^5 = 32$ classes can be covered.

(c) Having the appropriate information we can make the ECOC code using the table of this code. The number of code words in this set is a power of 2.

(d) Based on the number of classes we should pick up more suitable labels which means that the columns should be cut. This may cause some problem so we should consider the following points:

i. columns consist of all "1" all "0" elements are not valid. Because
in classification task they are meaningless. Furthermore in some combining methods these columns will cause some practical problems.

ii. Repeated columns are not useful in stable classifiers (like nearest neighbour or radial basis function network with exact procedure of centre selection). So they should be removed to decrease the calculating cost. This may destroy the label separation. In unstable classifiers, although the column separation provides more independence of base classifiers, repeating the columns may improve the performance by reducing the variance. So removing repeated columns is not necessary.

3.5 ECOC in classification

Although discussing about the reconstruction methods will come in next chapter we need to consider some points in decomposition which is related to combining framework in classification. This is why we present a quick review of ECOC classification algorithm here. Dietterich and Bakiri suggest a framework based on the idea of ECOC to solve a multi-class classification problems [15, 16].

Let $Z$ be a $k \times b$ code matrix with binary elements, where $k$ is the number of classes and $b$ is the number of binary classifiers.

Each row of $Z$ is a code word (with length of $b$) that is used as a label for one of classes and each column is a map to convert the multi-class problem to binary subproblems by defining a pair of super-classes.

In training phase binary classifiers are constructed over these sub-problems.

In test(application) phase, for a given pattern $x$, the set expert provides an output vector $y = [y_1 y_2 ... y_b]$.

The distance of this vector from all labels is:

$$L_i = \sum_{j=1}^{b} |z_{i,j} - y_j|$$ (3.1)

49
In decision making, the pattern \( x \) should be assigned to the class with minimum distance.

The greater distance between rows can cause better decision making, and distance between columns affects the covariance of local errors.

This algorithm can be reviewed as follows:

**ECOC Algorithm**

**In training phase:**

for each column:

- Map the training patterns into *super classes* according to the sequence of "1s" and "0s" in the column
- Train learning machine to make a *base classifier* with the patterns based on defined *super classes*

So we have \( b \) trained base classifiers.

**In inferring:**

- Apply a test pattern (\( x \)) to all base classifiers giving a vector:
  \[
  \vec{y} = [y_1, y_2, \ldots, y_b]^T
  \]
  in which \( y_j \) is the output of \( j \)th base classifier.

**For decision making (reconstruction)**

- Find the distance between the output vector with representative (label) of each class:
  \[
  L_i = \sum_{j=1}^b |Z_{i,j} - y_j|
  \]
- Assign \( x \) to the class corresponding to the code word having minimum distance with \( \vec{y} \)

Here, some important points should be considered:

1. **Softmaxing the base classifiers output**
   
   In general, the output of base classifiers can vary over a wide range. so
they can not be used as probabilities to be combined in the first method.
In the second one, distance from the labels which are “1”, and “0” should
be used. But this kind of comparison seems not to be fair.

For example let us assume that the output of a binary expert varies between
-1 and 1. For the point just on the bound the output of expert is 0, so the
distance to first super-class with the label of “1” is 1, and to the second
super-class with the label of “0” it is 0. It means naturally this classifier is
biased to select the second class.

This is why a softmax stage (normalising the outputs into some score be­
tween “0” and “1”) is very popular. In this case, we can interpret the output
as the posterior probability of super-class membership in many cases.

2. **New interpretation of Training**

We model the prediction task (inferring stage in classification) as a com­
munication problem in which the channel consists of “input features”,
“training samples” and “learning algorithm”. In this viewpoint the noise
can be considered as:“finite training samples”, “shortcomings in presen­
tation(feature extraction) which cause overlapped classes” and “flaws in
learning procedure”, that make corruption in classifiers. Just as in infor­
mation theory, by encoding class information (labelling) according to ECOC
and pass them through the channel, composite system can recover parts of
error if the error of bits are independent.

3. **Usage in polychotomy learning machines**

In some multi-class classifiers like neural networks or decision trees there
is the capability of providing hypothesis for all classes, now can we use the
ECOC code to make the problem more separable to get better performance?
The problem of such a suggestion is the fact that in these algorithms, all
output units have been set by using the same cost function. Therefore
errors in different output units are not independent so they can not benefit
from ECOC [14, 25]. However using “b” binary classifiers (base experts)
which have been trained independently, ECOC suggests a straight forward
technique to convert a multi-class problem into a few binary classification problems.

4. **Soft level information**

In information theory the output level of channel in the absence of the noise will be just a binary value ("1" or "0"), but in classification task, these values are the output of inferring stage and have some physical or logical interpretation. For example they can be the distance of the pattern to the prototype of a class or the probability of class memberships. This means that this kind of information can be used in other parts of system and not just in assigning the pattern to appropriate class.

5. **Hard level and soft level combining**

In information theory, hard level combining is used, it means, the output of channel for each bit, separately used for assigning (interpreted as "1" or "0"), then Hamming distance between binary values is used for error detection and correction. In classification on the other hand, the output of different bits before binarising are combined to make another score which is used for final assigning.

3.5.1 **Classification considerations in code generation**

Choosing Code matrix in classification differs from information theory in the following ways:

1. The repetition of the columns in information theory makes the error more correlated. In classification task on the other hand particularly for unstable classifiers, this repetition can improve the performance by reducing the variance of error. But it can not benefit stable classifiers.

2. All "0" or all "1" columns can exist in code matrix in information theory, but from the viewpoint of classification they are meaningless.
3. Using complementary columns makes the same binary classifiers in which just the labels of classes is changed, so it seems useless particularly for stable classifiers.

4. The main shortcoming in code generation techniques mentioned in section 3.4.1 is the fact that in information theory, system only concerns with hard-level information, and when received signal is different from “1” or “0”, we consider it as the result of noise. In classification on the other hand soft-level outputs can be meaningful and containing valuable information. For example the can be the probability of class membership or other scores. So in many cases we are interested in reconstructing them, because in this way not only we can use other combining frameworks (which could provide better performance) but also they can be useful in other tasks. This fact results in new demand of “free reconstruction error code”. In next chapter we will introduce the condition under which this demand can be satisfied in ECOC.

3.6 Summary of chapter

In this chapter we reviewed the problem of converting a multi-class problem into a few binary sub-problems. We have introduced two main strategies in this field:

1. In the first strategy we consider just two possible classes in each iteration and given a pattern, compare them on having this pattern.

2. Second strategy is based on redefining labelling in which some classes are gathered under a new label.

We have focused on the second strategy, particularly the method of error correcting output codes in which a code matrix is used to map the problem into another space.
Some main considerations in generating code matrix in information theory have been mentioned in this chapter and popular methods of production are introduced.

From the viewpoint of classification the main shortcoming of existing techniques for code generation is the fact that they are not concerned with soft-level scores needed for assigning. So there is the possibility of occurrence of reconstruction error. This fact demands a new condition set in code generating techniques. Since this error is related to combining framework as well as decomposition, discussion about the researches in code selection in classification task will come in chapter 4.
## Chapter 3. Label decomposition: ECOC code matrices

<table>
<thead>
<tr>
<th>N</th>
<th>K</th>
<th>T</th>
<th>N</th>
<th>K</th>
<th>T</th>
<th>N</th>
<th>K</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>255</td>
<td>199</td>
<td>7</td>
<td>511</td>
<td>358</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>1</td>
<td>191</td>
<td>8</td>
<td>349</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>187</td>
<td>9</td>
<td>340</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>179</td>
<td>10</td>
<td>331</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>26</td>
<td>1</td>
<td>171</td>
<td>11</td>
<td>322</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>163</td>
<td>12</td>
<td>313</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>155</td>
<td>13</td>
<td>304</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>147</td>
<td>14</td>
<td>295</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>139</td>
<td>15</td>
<td>286</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>57</td>
<td>1</td>
<td>131</td>
<td>18</td>
<td>277</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>2</td>
<td>123</td>
<td>19</td>
<td>268</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>3</td>
<td>115</td>
<td>21</td>
<td>259</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>4</td>
<td>107</td>
<td>22</td>
<td>250</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>5</td>
<td>99</td>
<td>23</td>
<td>241</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>91</td>
<td>25</td>
<td>238</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>7</td>
<td>87</td>
<td>26</td>
<td>229</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>79</td>
<td>27</td>
<td>220</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>71</td>
<td>29</td>
<td>211</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>63</td>
<td>30</td>
<td>202</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>55</td>
<td>31</td>
<td>193</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>120</td>
<td>1</td>
<td>47</td>
<td>42</td>
<td>184</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>2</td>
<td>45</td>
<td>43</td>
<td>175</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>3</td>
<td>37</td>
<td>45</td>
<td>166</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>4</td>
<td>29</td>
<td>47</td>
<td>157</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>5</td>
<td>21</td>
<td>55</td>
<td>148</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>6</td>
<td>13</td>
<td>59</td>
<td>139</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>7</td>
<td>9</td>
<td>63</td>
<td>130</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>9</td>
<td>511</td>
<td>502</td>
<td>1</td>
<td>121</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>10</td>
<td>493</td>
<td>2</td>
<td>112</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>11</td>
<td>484</td>
<td>3</td>
<td>103</td>
<td>61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>475</td>
<td>4</td>
<td>94</td>
<td>62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>14</td>
<td>466</td>
<td>5</td>
<td>85</td>
<td>63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>15</td>
<td>457</td>
<td>6</td>
<td>76</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>21</td>
<td>448</td>
<td>7</td>
<td>67</td>
<td>87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>439</td>
<td>8</td>
<td>58</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>27</td>
<td>430</td>
<td>9</td>
<td>49</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>31</td>
<td>421</td>
<td>10</td>
<td>40</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>247</td>
<td>1</td>
<td>412</td>
<td>11</td>
<td>31</td>
<td>109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>239</td>
<td>2</td>
<td>403</td>
<td>12</td>
<td>28</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>3</td>
<td>394</td>
<td>13</td>
<td>19</td>
<td>119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>4</td>
<td>385</td>
<td>14</td>
<td>10</td>
<td>121</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>215</td>
<td>5</td>
<td>376</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>6</td>
<td>367</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

N: code word length; K: message length; T: error–correction capability

Figure 3.2: The table of accepted values of added bits and correction capability
Chapter 4

Reconstruction in Error Correcting Output Code

4.1 Introduction

In chapter 3, we reviewed the problem of decomposition of a multi-class problem into a few binary (two-class) sub-problems. The specification of such a mapping is presented by a matrix. Binary sub-problems can be solved by local experts and the remaining problem which is under discussion in this chapter, is to reconstruct the final solution by combining the results of these experts. We mentioned that the code generation techniques addressed in chapter 3 are essentially based on considerations of information theory where hard-level outputs ("1"/"0" values) is the only valuable information. This is why have some shortcomings in classification task.

In this chapter we look at the problem of reconstruction of soft-level information (class scores) in ECOC. It begins with a discussion on previous works (section 4.1.1) and their shortcomings. Then an evaluation of ECOC comes which includes code selection consideration, error analysis and introducing a technique to improve its performance (section 4.2). Then alternative combining frameworks (including two new methods) are introduced (sections 4.3, 4.4, 4.5, 4.6, and
There are three main goals in this chapter:

1. As mentioned before, in a good decomposition, the system should be reconstructible. It means that for a perfect solution of sub-problems, it should provide a perfect final solution. As the first goal, we have tried to provide an analysis of the relationship between “decomposition matrix”, “combining method”, and “reconstruction error” in ECOC classifiers.

2. For practical cases, when the local experts are imperfect, how is the final solution related to precision of local experts? In other words, we aim to determine how sensitive the composite classifier is to its components. Bias-variance decomposition, is an appropriate tool for this error analysis.

3. The third goal is to investigate application of other combining frameworks in ECOC. Original method of ECOC [15, 16], can be viewed as a stacked generaliser [67]. The level zero classifiers in this system are the binary experts and the level one classifier is a nearest neighbour in which prototypes are the label of classes. Introducing other possibilities of combining is another goal of this chapter.

In statistical pattern classifications based on the output of inferring stage, two main methods of decision making exist:

- The first method is based on the evidence of belonging to each class. For example, many techniques like feed forward neural networks or decision trees are supposed to provide an estimation of the probability of class membership of each class. So the decision making can be based on the selection of the class whose probability is maximum.

- The second method is based on distance measurements between the output due to the pattern with the representative or prototypes of different classes. As an example, nearest neighbour classifiers use this technique. The class with minimum distance with the output should be selected in this approach.
Combining the results of experts can be viewed as a decision making procedure so these strategies are appropriate.

4.1.1 Discussion on previous works

After introducing the idea of ECOC classifiers research has been done in the following directions:

1. In most reported works this technique is used in a certain application or using a different learning machines to compare the results with single classifiers or other methods [45, 3, 4, 1, 8, 48].

2. The second area is to find condition set for a good code.

In[15, 16] the viewpoint of information theory essentially dominates, which is based on greater distance between rows and columns of code matrix. As we mentioned before the weakness of this viewpoint is the fact that although a soft-level combining framework has been used, code generating techniques are not concerned with reconstruction of these scores.

While random code has been suggested in [62] to avoid the cost of code generation, in [37] random code is considered as the best code.

(a) The idea of “Bayes consistency” is introduced which is essentially a hard-level concept and is defined as follows: “A PICT (Plug In Classification Technique) is said to be Bayes consistent for any test set if it always classifies to the Bayes class when the base classifier is a Bayesian classifier”.

(b) It is claimed that: “for any computational feasible and practically useful deterministic matrix, ECOC PICT will not be Bayes consistent”.

(c) Asymptotically as \( b \) (number of columns) approaches infinity, ECOC PICT will become Bayes consistent provided a random code matrix is used”.

58
Chapter 4. Reconstruction in Error Correcting Output Code

It seems that the main reason for insisting on a random code is to provide an error analysis for the new method (substitution). This method is essentially an ensemble combining of ECOC components based on finding an estimation of class membership probabilities from each expert independently and combining them by simple averaging[38, 37]. Considering the fact that substitution did not outperform ECOC in most reported experiments, and that his attempt to prove it is asymptotically equivalent to ECOC when \( b \) approaches infinity and that there exists a simple variance reduction analysis for simple averaging, it seems the main motivation for introducing this method and insisting on random code is to provide such an error analysis for ECOC.

3. Another field of investigation on ECOC is its evaluation. In [43], suggesting new definition of bias and variance for binary loss functions, some experimental examples show that ECOC can reduce both variance and bias. In [37] a variance reduction analysis is provided for the substitution which essentially is based on the analysis of variance in simple averaging. But it seems that some parts of chain have been missed, because the final variance is a function of what we want to have (class probability estimation) and not a function of what we actually have (super-class probability estimation provided by the output of local experts), so it is not simply related to the precision of learning procedure.

4. Finding alternative combining frameworks to provide final solution of problem is another field of research. Methods like least-squares [44], centroid[37] and substitution[37] are results of these investigations.

Our main contribution in this area has been:

1. providing a framework to show how can we found final scores from distance measurement in ECOC. This framework leads to:
   
   (a) Finding new condition set for code generating and practical technique of search in codes (section 4.2.1)
Chapter 4. Reconstruction in Error Correcting Output Code

(b) Finding new framework of combining (section 4.5)
(c) Suggesting a method to improve the performance of ECOC with code at hand (section 4.2.6)
(d) Providing complete error analysis concerning variance, bias and reconstruction error in ECOC (sections 4.2.3, 4.2.2 and 4.2.4)
(e) Finding condition set for code in least-squares (section 4.4).

2. Suggesting evidence combining for ECOC (section 4.7).

4.1.2 The structure of the chapter

In this chapter we introduce the following subjects:

1. The Original ECOC combining which is suggested by [15, 16]. We have introduced concept of reconstruction error and the condition under which this error will be minimised. An error analysis based on bias-variance decomposition is provided for this framework as well (section 4.2).

2. Alternative combining frameworks has been introduced as follows:

   (a) The Centroid algorithm [37]
   (b) The least squares method [37, 44]
   (c) Linear combining of superclass membership probabilities to find class membership probabilities
   (d) An analytic approach to estimate the class membership probabilities based on distance measurements
   (e) Dempster-Shafer based combining

The centroid algorithm is distance based method like original ECOC, which means in both methods the final decision making is based on the distance of the output with representatives of classes but the representatives are different (section 4.3).
In the third approach, an estimation of probability is made by using least square method (section 4.4). The fourth algorithm is based on estimating the probability of class membership with a linear combining in which the weighted factors will be found based on the concept of conditional probabilities (section 4.6).

In the fifth algorithm we have proposed an analytical approach to relate the posterior probability of class membership for each class, with the distance of the output from the label of classes (section 4.5).

In the sixth method, the soft-level output of binary classifiers are interpreted as evidence of a given pattern \( x \) belonging to the super-class and Dempster-Shafer theory of evidence is used to combine these evidences (section 4.7).

### 4.2 Evaluation of original ECOC

Certain items in this method affecting performance are:

- The way of converting the main problem into binary sub-problems. It can be viewed from two viewpoints:
  
  - Code selection, which means finding a matrix with binary elements with some properties which cause error reduction as described in section 3.4.
  
  - Code word selection which means having a suitable code, which code word (rows of matrix) is more suitable to represent a class.

- A measurement vector provided by local experts as input of the decision making procedure. This output will be passed through a softmaxing procedure to make it between “1” and “0”. Here there are two points:

  1. Unstable base classifiers, like decision trees and neural networks are preferred because they have low bias error, and their variance which is relative high, can be reduced by repeating the training over small perturbed training set.
Chapter 4. Reconstruction in Error Correcting Output Code

2. Although using a soft-maxing stage can help to have more comparable distances, it can not guarantee a fair distance set in all cases. If there is some bias in soft level outputs, the distance of this output from label elements ("1" and "0") are different which means system is biased to choose some classes more than others. This can cause some kind of error as we will show in section 4.2.4.

4.2.1 Code selection in ECOC

Obviously the main goal of such a composite system is to minimise the classification error. Having a certain learning machine (homogeneous combining) and training set, we will try to find the code matrix that can minimise the error.

Generally speaking three kind of errors can be distinguished in such a framework:

1. **Bayesian error**, which is due to overlapped classes and can not be removed by combining or error correcting.

2. **experts' added error**, which is related to the training of base classifiers. We hope to remove some parts of the effect of this error in final assigning by error correcting capability of codewords. This can be done in two ways:
   - for an unstable classifier like neural network with random initial weights, (where a small change in training samples or parameters could cause a big change in the whole structure) the variance of error (part which is due to the training samples) is rather high, and repeating the training phase can reduce the error even if the columns are the same.
   - On the other hand error due to the learning machine(bias) will not be reduced in this way. This is the common problem in homogeneous combining (combining the results of classifiers with the same structure of learning). We will show that in some circumstances this kind of error can be reduced in ECOC (section 4.2.4).

3. **Combining error**, which can be divided into two items:
(a) reconstruction error which means even if all our base classifiers are perfect (for example they provide Bayesian classifiers decision boundary), some error may be added based on the way we combine their results.

(b) measurement error which is due to the sensitivity of distance measurement (3.1) to some parts of experts' error (bias).

We can conclude that the main procedures involved in ECOC error are:

1. Error correcting mechanism, which is related to the error correcting capability of code (distance between rows of code matrix). The interpretation of error correction in this framework differs from what we have said in information theory because we are using a soft-level information in combiner. Error correcting can be interpreted in the following ways:

   (a) Each expert concentrates on small part of decision boundary. The higher the number of experts means the smaller part any one should learn. It means they will be faced with easier problem.

   (b) Transforming the problem into higher dimensional space in which (we hope) classes are more separable and so lead to more precise decision making.

2. Voting or averaging to reduce the variance, which is related to repetition of different parts of decision boundary in subproblems (number of base experts or length of code words) and reducing the bias which is related to the independence of base classifiers (distance between columns) [43].

3. Decision making strategy which differs from Bayesian rule may cause combining added error.

Now which code can provide the best possible performance?
4.2.1.1 Maximum distance code

From equation 3.1 one can simply argue that greater distance between labels (code words) can cause less error, because in this case the difference between $L_i$ and $L_j$ is greater so having the output of experts ($y$), judgement between these two classes is less sensitive to local (experts) error. Longer code words means higher number of experts and greater inferring cost. So for a certain length, the first condition for a good code is:

having the greatest possible distance between any pair of code words

there are few methods to create codes with maximum guaranteed distance between code words. One of the most popular ones is using BCH code as described in chapter 3.

4.2.2 Introduction to reconstruction error

Reconstruction of soft-level information is a goal in regression task. In classification, when we use an alternative way of estimation, (for example by new algorithm or decomposition method) all we concern about, is the rank of correct class. To shed light on this concept of consider a $k$ class problem, having posterior probability of each class represented by $q = [q_1, q_2, \ldots q_k]^T$. The Bayesian decision rule for input $x$ says:

assign $x$ to $\text{ArgMax}_i(q_i)$.

In ECOC on the other hand, the decision rule is based on distance of the output of classifiers from all code words $L = [L_1, L_2, \ldots L_k]^T$:

assign $x$ to $\text{ArgMin}_i(L_i)$. Now if for a pattern $\text{ArgMax}_i(q_i)$ and $\text{ArgMin}_i(L_i)$ are different ECOC wont be Bayes consistent[37]. So the main idea is keeping the rank of correct class. A more general idea could be keeping the rank of all classes or even reconstruction of $q$ from $L$.

Consider the case that base classifiers provide posterior probability of super class membership precisely ($y = [y_1, y_2, \ldots y_b]$). Super classes are constructed over the
columns of \( Z \), so in matrix equation form we will have:

\[
y = Z^T.q
\]  

(4.1)

or

\[
y_j = \sum_{l=1}^{k} q_l.Z_{lj}, \quad (l = 1...k)
\]

since:

\[
L_i = \sum_{j=1}^{b} |Z_{ij} - y_j|
\]

we have:

\[
L_i = \sum_{j=1}^{b} |(\sum_{l=1}^{k} q_l.Z_{lj}) - Z_{ij}|
\]

now if we separate the case \( l = i \) and consider the fact that \( \sum_{l=1}^{k} q_l = 1 \), it leads to:

\[
L_i = \sum_{j=1}^{b} |q_i.Z_{ij} - Z_{ij} + \sum_{l \neq i} q_l.Z_{lj}|
\]

\[
= \sum_{j=1}^{b} |q_i.Z_{ij} - (\sum_{l \neq i} q_l + q_i).Z_{ij} + \sum_{l \neq i} q_l.Z_{lj}|
\]

\[
= \sum_{j=1}^{b} |\sum_{l \neq i} q_l.(Z_{ij} - Z_{lj})|
\]

so:

\[
L_i = \sum_{l \neq i} q_l.\sum_{j=1}^{b} |Z_{ij} - Z_{lj}|
\]

(4.2)

In equation 4.2, the inner part \( (\sum_{j=1}^{b} |Z_{ij} - Z_{lj}|) \), is the Hamming distance between \( l \)th and \( i \)th rows. Let us show this distance by \( d_{il} = \sum_{j=1}^{b} |Z_{ij} - Z_{lj}| \). For first row (as example) from equation 4.2 we will have:

\[
L_1 = (0).q_1 + d_{12}.q_2 + ... + d_{1k}.q_k
\]

having \( q_k = 1 - q_1 - q_2 - ... q_{(k-1)} \) we will have:

\[
L_1 - d_{1k} = (-d_{1k}).q_1 + (d_{12} - d_{1k}).q_2 + ... + (d_{1(k-1)} - d_{1k}).q_{k-1}
\]

and in general case:

\[
L_i - d_{ik} = (d_{i1} - d_{ik}).q_1 + ... + (-d_{ik}).q_i + ... + (d_{i(k-1)} - d_{ik}).q_{k-1}
\]

The only way to keep the rank of \( L_i \) for any value of \( q \) and \( i \) is the case that all coefficients of \( q_l \) will be zero when \( i \neq l \), which means: \( d_{i1} = d_{ik}, d_{i2} = d_{ik}, ... \)
In other words the distance measurement $L_i$ is a function of the posterior probability vector $\bar{q}$ and the Hamming distance of this row to all other rows. If we want to have a decision rule like the Bayesian one, we need the case in which $L_i$ is a monotonic function of just $q_i$, so we better make all the other variables fixed. This means having a code with the same distance between all pairs:

$$\sum_{j=1}^{b} |Z_{ij} - Z_{lj}| = M$$

for any $i \neq l$

which means:

$$L_i = M \cdot (\sum_{l \neq i} q_l)$$

if a softmaxing stage has been used, we will have

$$\sum q_l = 1$$

so

$$\sum_{l \neq i} q_l = 1 - q_i$$

and:

$$L_i = M \cdot (1 - q_i)$$

This means, as expected, in this framework, $L_i$ has a simple linear relationship with $q$. In other words, in fact this framework provides an alternative way to estimate the posterior probability $q$ in which the sensitivity to noise (error) is reduced.

Rather than estimating $q$ directly, the system estimates super-class membership probability. In this estimation problem classes are more separable so the effect of noise is less. Then in reconstruction the final estimation will be made. The decision boundary of such a system is based on the Bayesian decision rules:

$$\text{ArgMax}_i(q_i) = \text{ArgMin}_i(L_i)$$

Here we can find the second condition for a good code:

**having equi-distance between all pairs of rows**
4.2.3 Analysis of variance reduction

To show how ECOC can reduce the variance of prediction, let us suppose that the \( j \)th local expert prediction (\( y_j \)) has an unbiased error \( e^y_j \) with the variance of \( \sigma^y_j \). The composite classifier performance will be affected by the variance of error of \( L_i \) (\( \sigma^L_i \)) which is related to the error of experts. From equation 3.1, since \( Z_{ij} \) is deterministic for a certain code matrix, having one of the values of “1” or “0”, assuming the error of each expert is independent, we can conclude:

\[
\text{Var}(e^L_i) = \text{Var}(\sum_{j=1}^{b} e^y_j) \tag{4.4}
\]

where \( \mathcal{E}(\cdot) \) is the expected value operator. Since we assume that error has the mean value of zero, the second term will be removed.

\[
= \mathcal{E}(\sum_{j=1}^{b} e^y_j)^2 + \sum_{j=1}^{b} \mathcal{E}(e^y_j)\sum_{i \neq j} \mathcal{E}(e^y_i, e^y_j)
\]

So

\[
\text{Var}(e^L_i) = \sum_{j=1}^{b} \sigma^y_j + \sum_{i \neq j} \text{Cov}(e^y_i, e^y_j)
\]

To compare variance in ECOC with other methods such as multi-class networks, we consider the case in which the probability can be found from the distance directly.

Let all experts have the same variance \( \sigma^y \) and the covariance between any pair of experts error is \( \rho.\sigma^y \). We will have:

\[
\text{Var}(e^L_i) = b\sigma^y + b(b - 1)\rho.\sigma^y = b\sigma^y(1 + (b - 1)\rho)
\]

From equation 4.3 we can estimate the probability of class membership:

\[
\hat{q}_i = 1 - \frac{\hat{f}_i}{M}
\]
so the variance of error of this estimation is:

$$Var(c_i^0) = Var \left( \frac{c_i^L}{M} \right)$$

$$\sigma_{ecoc}^2 = \frac{b}{M^2} \sigma^2 (1 + (b - 1) \rho)$$  \hspace{1cm} (4.5)

This equation says that the variance of ECOC can be affected by:

1. The precision of local experts ($\sigma^y$). Using more precise experts will reduce the variance of ECOC.

2. Code word (row) separation will decrease the variance of framework by the factor of $(M^2/b)$.

3. Column selection which appears as the factor $\rho$.

This conclusion is completely in agreement with our conditions to select a code word.

4.2.4 The effect of bias

We have seen that the distance measurement is the seed of decision making (equation 3.1 $L_i = \sum_{j=1}^{M} |Z_{ij} - y_j|$). Here $Z_{ij}$ is zero or one, now if in the output of the estimator (trained learning machine) all the members of first super-class are below .5 and all patterns belong in second super-class have an output more than .5, this measurement will be fair, because the distance between the bound (threshold) from any of the values is the same.

Consider the point $(z)$ which is just on the boundary of these classes (probability of belonging to these classes is equal), Suppose in the learning machine which builds all base classifiers, the bound between two super-classes (Threshold of inferring stage) is more than .5 (e.g. .6). If we had used a "hard-level" combining, this would be solved by just a shift in threshold level but in this framework,
naturally there is a bias for the code words which have more “1”, because the
distance of “1” to the threshold is less then the threshold of “0”. So if the number
of “1”s in jth row is more than ith, the $L_j$ will be less than $L_i$, even if $q_i = q_i$,
as we supposed.

The only way to prevent such a problem is to use a code with equal number of
“1”s in all code words.

To shed light on this problem let us have an example:

4.2.5 Example

Consider a three class problem. A sample $x$ is a common point of all the classes,
so the posterior probability vector will be: $q = [.33, .33, .33]^T$.

we use a code $z$ to represent these classes:

$$z_1 = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
\end{bmatrix}$$

In decomposition stage, the precise values of the posterior probability of super­
class membership can be found as:

$$p = z_1^T \times q \quad \text{so} \quad y = p = [.33, .66, .66, .33]^T$$

it means the distance vector is $L = [2, 1.66, 1.68]^T$. System assigns $x$ to second
class. This is due to reconstruction error.

Now assume that our estimator has a bias in estimation, for example it adds a
fixed value to all the values: $y = \tilde{p} = [.43, .76, .76, .43]^T$

and if we find the distance between $\tilde{y}$ and code words, we will have: $L =
[2, 1.86, 1.48]^T$ which means system assign $x$ to Third class!

In this example, it is obvious that:
1. In both cases assigning is wrong, in first case reconstruction error has occurred.

2. In second case the effect of bias in base classifier causes another error.

Now consider another code

\[ z_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \]

in this code the distance between code words are different but the number of “1”s is the same and the distance between \( y \) and code words in both cases (without bias and with bias) are: \( L = [2, 2, 1.34] \). It means that the reconstruction error exists, because system will assign \( x \) to third class, but the bias has more effect (distance of classes are the same).

In third case, we use an equi-distance code:

\[ z_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \]

In this code the distance between any pair of code words is equal, and the number of “1”s is the same in all code words. So \( y = p = [0.66, 0.66, 0.66]^T \) and the distance vector of two cases are \( L_1 = [1.34, 1.34, 1.34] \) and \( L_2 = [1.24, 1.24, 1.24] \) respectively. It shows that there is no reconstruction error (because three classes have the same distances and the bias does not effect decision making as well.

One interesting point is the fact that in third case (equi-distance code) the number of “1”s is the same, (but not vice versa), now is there any relationship between them?

4.2.5.1 Unbiased measurement in Equi-distance code

Let \( Z \) be a equi-distance code matrix in which:
\[ \sum_{i=1}^{b} |Z_{il} - Z_{jl}| = M \quad \text{for any pair of } i \text{ and } j \]

having a look at truth table, for any binary variable \( a \) and \( b \), we know that:

\[ |a - b| = a + b - 2ab \]

so for two rows (ith and jth) we will have:

\[ \sum_{i=1}^{b} Z_{il} + \sum_{j=1}^{b} Z_{jl} - 2 \sum_{i=1}^{b} Z_{il} Z_{jl} = M \quad (4.6) \]

For (ith and Kth) rows:

\[ \sum_{i=1}^{b} Z_{il} + \sum_{k=1}^{b} Z_{kl} - 2 \sum_{i=1}^{b} Z_{il} Z_{kl} = M \quad (4.7) \]

and for (jth and Kth) rows also:

\[ \sum_{j=1}^{b} Z_{jl} + \sum_{k=1}^{b} Z_{kl} - 2 \sum_{j=1}^{b} Z_{jl} Z_{kl} = M \quad (4.8) \]

subtracting equations 4.6, 4.8:

\[ \sum_{i=1}^{b} Z_{il} - \sum_{k=1}^{b} Z_{kl} = 2\left(\sum_{i=1}^{b} Z_{il} Z_{jl} - \sum_{i=1}^{b} Z_{il} Z_{kl}\right) \quad (4.9) \]

where \( \sum_{i=1}^{b} Z_{il} \) is the number of “1”s in ith row, and \( \sum_{i=1}^{b} Z_{il} Z_{jl} \) is the number of common “1”s between two rows.

Let us divide bits upon the “1”s and”0”s of jth row, for example \( \sum_{(j1)} Z_{i} \) be the number of “1”s in ith row where corresponding bits in jth row are “1”. Equation 4.9 can be rewritten as follows:

\[ \sum_{(j1)} Z_{i} + \sum_{(j0)} Z_{i} - \sum_{(j1)} Z_{k} - \sum_{(j0)} Z_{k} = 2 \sum_{(j1)} Z_{i} - 2 \sum_{(j1)} Z_{k} \quad (4.10) \]

\[ \sum_{(j0)} Z_{i} + \sum_{(j1)} Z_{i} - \sum_{(j0)} Z_{k} - \sum_{(j1)} Z_{k} = 2 \sum_{(j0)} Z_{i} - 2 \sum_{(j0)} Z_{k} \quad (4.11) \]

or:
The number of different bits between $i$th and $j$th pair (Hamming distance) is sum of the positions in which the bit is "1" in $i$th row but "0" in the $j$th one, and vice versa. On the other hand in an equi-distance code we have the same distance between $k$th and $j$th rows, so:

\[ \sum_{(j0)} Z_i - \sum_{(j1)} Z_i = \sum_{(j0)} Z_k - \sum_{(j1)} Z_k \quad (4.12) \]

subtracting 4.12 and 4.13 we will have:

\[ \sum_{(j1)} (Z_i' + Z_i) = \sum_{(j1)} (Z_k' + Z_k) \quad (4.13) \]

but $\sum_{j1}(Z_i' + Z_i) = \sum_{i=1}^{b} Z_{il} \cdot Z_{jl}$ and since $i$, $j$ and $k$ are dummy variable:

\[ \sum_{l=1}^{b} Z_{il} \cdot Z_{jl} = \sum_{l=1}^{b} Z_{kl} \cdot Z_{jl} = m \quad (4.15) \]

where $m$ is number of common bits in code word. Combining 4.9 and 4.15:

\[ \sum_{l=1}^{b} Z_{il} = \sum_{l=1}^{b} Z_{kl} = \sum_{l=1}^{b} Z_{jl} = n \quad (4.16) \]

So for a equi-distance matrix the number of "1"s in different rows are the same, which means this code provide an unbiased measurement in ECOC. Furthermore the number of common "1"s between any pair of rows is the same.

**4.2.5.2 Considerations in code generation**

Code generation methods which came from information theory can not provide equal distance between all rows and we have to search to find rows with equal distance. This search is not easy for longer codes, and this is why it has been said
that equi-distance codes can not be useful practically ([37]). Knowing the fact that in equi-distance code all rows have the same number of “1” can be a great help in this problem. To find an equi-distance code, the only things we should do are:

1. Generating a code with enough distance between code words by using one of the methods we have mentioned in chapter 3.
2. Counting the number of “1”s in all rows
3. Selecting a set of rows having the same number of “1”s.

This approach is much easier in comparison with searching for equal distance code words.

4.2.6 Label selection dilemma: Circular ECOC

One major weak point of mentioned technique (ECOC) is the fact that the code selection is completely independent of the problem. It is obvious that the error is dependent on the separability of different classes. In a certain problem the probability of confusing between classes is not equal. For example the probability of confusing the number "4" with "9" is higher than with "5". So in code selection it is important to have greater distance between the labels of more overlapped classes but we increase the distance between all cases.

In a certain code, this problem exists as well. How should we select labels (code words) for different classes? This division can be based on separability measurement of classes (e.g. Mahalonobis distance) but this problem relates to a template matching in a very high dimensional space and practically unsolved. One possible alternative to reduce the sensitivity of error to code word selection is using different labels of a code for any class.

In such an approach we use two stages of combining, with an ensemble whose components are composite classifiers made by modular method of ECOC.
Having a code matrix $Z$ with $k$ codewords the algorithm can be explained as follows:

\begin{center}
\textbf{Circular ECOC Algorithm}
\end{center}

\textit{For any class $n$, $n = 1 \ldots k$ :}
- produce $Z_n$ by shifting the rows of $Z$ $(n-1)$ times.
- Train base as original ECOC
- Apply test patterns $(x)$ to find the distance:
  \[ L_{ni} = \sum_{l \neq i} q_l \cdot \sum_{j=1}^{b} |Z_{nij} - Z_{nij}| \]

\textit{Find distance measurement} for each class by averaging its distances:
  \[ L_i = \frac{1}{k} \sum_{n=1}^{k} L_{ni} \]

\textit{Assign} $x$ to class with minimum $L_i$

In this approach all the code words have been used as labels of each class to reduce the sensitivity to code word selection. Furthermore this approach provides the opportunity to repeat the training (perturbing) by the same training set, feature set and learning machine but different labelling.

### 4.3 Centroid of classes

In ECOC labels are used not only as desired output of training phase but also as representative of classes in distance measurements. In this way for patterns which belong to first superclass the output can be between .5 to 1 but we compare them with “1”. If we assume that outputs of all members of a certain class scatter around a centre, it seems reasonable to choose this centre as representative of this class and use it to find the distance measurement.

For example consider that we want to classify six patterns with given probabilities
Chapter 4. Reconstruction in Error Correcting Output Code

<table>
<thead>
<tr>
<th>pattern No.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>true assign</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ECOC assign</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Centroid. assign</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Assignment table of ECOC and Centroid method

as follows:

\[
q = \begin{bmatrix}
.22 & .30 & .38 \\
.20 & .30 & .40 \\
.30 & .37 & .33 \\
.31 & .36 & .33 \\
.37 & .36 & .27 \\
.38 & .35 & .27 \\
\end{bmatrix}
\]

We use \( z_1 \) to make ECOC and centroid frameworks to solve this problem:

\[
z_1 = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
\end{bmatrix}
\]

the results of ECOC and centroid methods are given in table 4.1 which indicate that this concept can reduce some parts of error in classification. Having a certain code matrix \( \mathbf{Z} \), the algorithm ([37]) is as follows:
Chapter 4. Reconstruction in Error Correcting Output Code

Centroid Algorithm

In training:
- Train \( b \) experts as it is done ECOC algorithm
  - For each class \( i \):
    - Apply training sample of this class \((l = 1, 2, \ldots, n)\)
      to all experts \((j = 1, 2, \ldots, b)\) to make an output matrix:

\[
P_i = \begin{bmatrix}
p_{i1} & p_{i2} & \cdots & p_{ib} \\
\vdots & \vdots & \ddots & \vdots \\
p_{ni} & \cdots & \cdots & p_{nb}
\end{bmatrix}
\]

where each row belongs to one of patterns
- Make an average (or median) of each column to make a centroid
  for this class:

\[
C_i = \begin{bmatrix}
\bar{p}_{i1} & \bar{p}_{i2} & \cdots & \bar{p}_{ib}
\end{bmatrix}
\]

In test (application):
- In inferring:
  - Apply the new pattern \( x \) to all experts to make an output:

\[
Y = [y_1 \ y_2 \ \cdots \ y_b]
\]

- For decision making:
  - For any class \( i \):
    - find the distance between the output \( Y \) and the representative
      of this class \( C_i \):

\[
L_i = \sum_{j=1}^{b} |C_{ij} - y_j|
\]

* Assign the pattern \( x \) to class its corresponded distance
  is minimum.
4.4 Least squares ECOC

As noted before (3.2), decomposition of a multi-class classification problem into binary sub-problems can be interpreted as transferring the original output \( q \) to another space \( p \). In Matrix form we will have:

\[
p = Z^T . q \quad j=1...b
\]

Having \( y_j \) (the estimation of posterior probability \( p_j \) provided by \( j \)th expert), this matrix equation can be solved to find an estimation of class membership probabilities \( \hat{q} \) and then the Bayesian decision rule can be used for assigning. Here there are two main points:

- \( Z^T \) is not a square matrix.

- In general, base classifiers will not produce correct probability and some errors will exist:

\[
y_j = \sum_{i=1}^{k} Z_{ij} \cdot q_i + \sigma_{y_j}
\]

A natural unbiased solution for such a problem is based on using least square method which means finding the \( \hat{q} \) which minimises a cost function such as:

\[
R_y = \sum_{j=1}^{b} \sigma_{y_j}^2 = \sum_{j=1}^{b} (y_j - p_j)^2.
\]

But

\[
\sum_{j=1}^{b} \sigma_{y_j}^2 = \sum_{j=1}^{b} (y_j - \sum_{i=1}^{k} Z_{ij} \cdot q_i)^2
\]

The optimum point will be:

\[
q^* = (Z.Z^T)^{-1}.Z.y
\] (4.17)

Having a certain code matrix \( Z \), we can define our classification algorithm as
Chapter 4. Reconstruction in Error Correcting Output Code

follows:

**Least Squares ECOC Algorithm**

*In training:*
- Define superclasses based on the sequence of “1”s and “0”s in each column of code matrix
- Train $b$ binary classifiers (base expert) based on defined superclasses

*In inferring:*
- Apply the new pattern $x$ to all classifier to make the output vector:
  
  $$y = [y_1, y_2, \ldots, y_b]$$

*For decision making:*
- Estimate the class membership probability by solving the following matrix equation (equation 4.17):
  
  $$\hat{q} = (Z.Z^T)^{-1}.Z.\bar{y}$$

This method will prevent occurrence of reconstruction error regardless of choosing $Z$. The only condition is to have a code in such a way that equation 4.17 has a solution ($ZZ^T$ is not singular).

It means that having a precise estimation of $p$ (perfect binary experts), we will find the $q$ precisely. But in the presence of noise the sensitivity of solution to the code matrix is important.

In ECOC the elements of $Z$ is “0” or “1”, so the elements of $A = ZZ^T$ are:

$$a_{ij} = \sum_{l=1}^{r} z_{il} z_{lj}$$

are all non negative. Some times this matrix is singular, for example when:

1. all elements of the $i$th row are “0” ($z_{il} = 0$ for any $l$) , elements of $(a_{ii}$ for all $l$) will be zero.

2. two rows (or columns) are equal in $Z$, which cause two rows in $A$ will be equal.

78
4.4.1 Error consideration for code selecting

For any certain $Z$, this solution will minimise the $R_y$, (sum of squares error of $y$), now the question is:

"which $Z$ will minimise the square error of $q$ ($R_q$)?"

In other words, each $Z$ matrix will make a transformation from $q$ space into $y$ space in which optimum $q$ for minimising error of $y$ is given by equation 4.17, but is there a transformation in which we can minimise $R_q$ and $R_y$ simultaneously? To answer this question we should find the relationship between $R_q$ and $R_y$.

In matrix forms $R_y$ can be shown as follows:

$$R_y = (\hat{y} - y)^T(\hat{y} - y)$$
$$= \hat{y}^T\hat{y} - 2\hat{y}^T.y + y^Ty$$

and since $y = Z^T.q$:

$$R_y = q^T.ZZ^T.q - 2q^T.ZZ^T.q + q^T.ZZ^T.q$$

(4.18)

At first glance it seems that if $ZZ^T$ be an identity matrix.

$$R_y = m.I.(q^T.q - 2q^T.q + q^T.q)$$
$$= m.I.R_q$$

where $m$ is a constant and I is identity matrix.

This case does not interest us much because for a code matrix with elements of "1"s, "0"s, the only possible case is to have a unitary matrix for $Z$. Otherwise non diagonal elements will not be zero.

For this case (one-per-class) the length of code words just equals the number of classes which limits the number of repetitions of training. Many other matrices have exhibited better performance as we will see in next chapter.

4.4.1.1 Matrix in form of $m_1*1+n_1I$

Suppose that $A = ZZ^T$ can be written in the above form:
\[ a_{ij} = \begin{cases} 
 n & \text{if } i=j \\
 m & \text{otherwise} 
\end{cases} \quad (4.19) \]

which means:

\[
A = \begin{bmatrix}
 n & m & \cdots & m \\
 m & n & \cdots & m \\
 & & \ddots & \vdots \\
 m & m & \cdots & n
\end{bmatrix}
\]

now we are going to calculate the last term in equation 4.18. Let

\[ B = A.q = \begin{bmatrix}
 n & m & \cdots & m \\
 m & n & \cdots & m \\
 & & \ddots & \vdots \\
 m & m & \cdots & n
\end{bmatrix} \begin{bmatrix}
 q_1 \\
 \vdots \\
 q_k
\end{bmatrix}
\]

where: \( b_i = nq_i + \sum_{j \neq i} m.q_i \), we know that \( \sum_{j \neq i} m.q_i = m(1 - q_i) \). It means:

\[ b_i = (n-m)q_i + m \quad \text{and:} \]

\[
q^T.ZZ^T.q = [q_1 q_2 \cdots q_k] \begin{bmatrix}
 q_1(n-m) \\
 \vdots \\
 q_k(n-m)
\end{bmatrix} + [q_1 q_2 \cdots q_k] \begin{bmatrix}
 m \\
 \vdots \\
 m
\end{bmatrix}
\]

\[
q^T.ZZ^T.q = (n-m).q^Tq + m \quad (4.20)
\]

through the same path for the second and first terms in equation 4.18:

\[-2\hat{q}^T.ZZ^T.q = -2(n-m).\hat{q}^T\hat{q} - 2m \]

and:

\[
\hat{q}^T.ZZ^T.\hat{q} = -2(n-m).\hat{q}^T\hat{q} + m
\]

A simple substitution in equation 4.18 will make:

\[ R_y = (m-n).R_q \quad (4.21) \]
which means:

_for a matrix with the form of m*1+nI, using equation 4.17, we can minimise sum squares error of both q and y simultaneously._

Now we are ready to ask another question:

"what kind of matrix (Z) has such a property"?

### 4.4.2 Equi-distance code

Suppose that Z is a matrix with binary (1 and 0) elements (code matrix) and for elements of \( A = ZZ^T \) we have:

\[
A_{ij} = \sum_{l=1}^{b} Z_{il}Z_{lj}^T = \sum_{l=1}^{b} Z_{il}Z_{jl} = \begin{cases} 
n_i & \text{if } i=j \\
m_{ij} & \text{otherwise} 
\end{cases}
\]

where \( n_i = \sum_{l=1}^{b} Z_{il}^2 \) is the number of "1"s in ith row, and \( m_{ij} = \sum_{l=1}^{b} Z_{il}Z_{jl} \) is the number of common "1"s between ith and jth rows.

It is clear that Z can be our choice if:

\( n_i = n \) for any \( i \)

\( m_{ij} = m \) for any pair of \( i, j \)

We have shown before that for equi-distance matrix the number of "1"s in all rows is the same, furthermore the number of common "1"s between any pair of rows are the same (equations 4.15 and 4.16 in section 4.2). It means:

_**using an equi-distance code matrix in least squares ECOC the sum squares error of local estimators(y) and global estimation(q) are minimised simultaneously.**_
4.5 Analytic approach to estimate probabilities from distance measurement

Both distance measurement \( L_i \) in original ECOC, and probability estimation \( \hat{q} \) in least squares, are calculated from the experts output \( p \). It can be useful to find their relationship directly. It is useful not only because in some real application we may need them in different parts of problem, but also to understand their behaviours. For example it can be useful to compare the composite system error to the component errors.

We have seen that there is a relationship between the values of the output \( p \) and the real probability, and code matrix elements (recalling equation 4.2):

\[
L_i = \sum_{l \neq i} q_l \sum_{j=1}^b |Z_{ij} - Z_{lj}|
\]

The inner summation is the Hamming distance between pair rows of \( i \) and \( l \). Since the Hamming distance of a row from itself is zero, we can change the equation as follows:

\[
L_i = \sum_{l=1}^k q_l \sum_{j=1}^b |Z_{ij} - Z_{lj}| \quad \text{for} \quad i = 1, 2, \ldots k
\]  

(4.22)

This set of equation can be rewritten in matrix form:

\[
L = D \times \hat{q}
\]

(4.23)

where the elements of \( D \), \( (d_{il}) \) represent the Hamming distance between the pair rows of \( i \) and \( l \):

\[
d_{il} = \sum_{j=1}^b |Z_{ij} - Z_{lj}|
\]

and \( L = [l_1, L_2, \ldots, L_k]^T \) is the vector of distance measurement of the output \( y \) (an estimation of super class membership) from the label of classes.

All the elements of \( D \) have non negative values so it is not a singular matrix and the equation has always a solution to provide an estimate for \( q \)
\[ \hat{q} = D^{-1} \times L \] (4.24)

One advantage of this method is the fact that the elements of \( D \) are not "1" or "0", so this matrix is less likely to be singular than \( ZZ^T \).

Note when we use equi-distance code matrix, this framework is exactly the same as original ECOC, but for other code words it is different.

---

**Analytic approach to find probabilities from distance measurements**

**In training:**
- Fine the distance matrix \( D \) in such a way that \( d_{ij} = \sum_{j=1}^{b} |z_{ij} - z_{ij}| \)
- Train \( b \) base classifier based on superclasses defined by columns of \( Z \)

**In inferring**
- Applying new pattern \( x \), find the output of all experts: \( y = [y_1, y_2, \ldots, y_b] \)
- Find the distance of the output vector to all labels: \( L_i = \sum_{j=1}^{b} |z_{ij} - y_j| \) for \( i = 1, 2, \ldots, k \)

**For decision making:**
- Solve matrix equation (equation 4.24):
  \[ \hat{q} = D^{-1} \times L \]
- Assign \( x \) to the class which its corresponded probability is maximum

---

4.6 Linear combining of estimate probabilities

As we have seen before in ECOC, for a given input \( x \) we will have the output vector \( y = [y_1, y_2, \ldots, y_b] \) which is an estimation of superclass membership \( p = \ldots \)
Although least squares and analytic approach give us the ability to estimate the probability of class membership \( q = [q_1, q_2, \ldots, q_k] \), in many cases we are interested in estimating this probability as a linear function of \( y \) which means:

\[
\hat{q}_i = \sum_{j=1}^{b} W_{ij} y_j \quad \text{for} \quad i = 1, 2, \ldots, k \tag{4.25}
\]

where

\[
\sum_{j=1}^{b} w_{ij} = 1 \quad \text{for} \quad i = 1, 2, \ldots, k.
\]

The problem is to find the weighting factor matrix \( W \):

\[
W = \begin{bmatrix}
w_{11} & w_{12} & \cdots & w_{1b} \\
\vdots & \vdots & \ddots & \vdots \\
w_{k1} & \cdots & \cdots & w_{kb}
\end{bmatrix}
\]

where each row of this matrix belongs to one of classes.

There are different methods to choose the weighting matrix \( W \) as we explained in section 2.3. but at first glance it seems that this will not lead us to a solution, Rather than estimating a \( k \times b \) dimensional vector \( q \), we need to having estimate of a \( k \times b \) matrix \( W \), which seems a more difficult problem. But practically it would be based on the way we use to do this.

One simple approach is from the view of conditional probability.

The probability of pattern \( x \) belonging to class \( i \) can be found by simple averaging over all the conditional probabilities of \( x \) belonging to class \( i \) and (for example) first super classes defined by columns of code matrix \( P_{\text{conij}}(x) \):

\[
\hat{q}_i(x) = \frac{1}{b} \sum_{j=1}^{b} P_{\text{conij}}(x)
\]

We can decompose \( P_{\text{conij}}(x) \) into two terms:

1. The probability of \( x \) belonging to first superclass
2. The probability of the members of first superclass belonging to class \( i \)
To find the above terms we should note that:

- The $j$th expert has already provided the first item ($y_j$ which is an estimation of $p_j$).

- The second item can be found by a simple assumption which is acceptable in any generalisation problem.

If the distribution of training and test patterns are the same we can found the second item by counting the number of training samples which belongs to both the $i$th class and first superclass of sub-problem defined by the $j$ column.

The elements of the weighting factors matrix can be found by normalising the numbers of each class in first super class of different partitioning. The main reason to use such a method to find the weighting factors is its simplicity that make it practical particularly for longer code matrices. On the other hand for longer code matrices, the normalising procedure is more precise so it can improve performance in this way. One advantage of such an approach is the fact that here our combining framework (reconstruction) will be affected by training samples.

In all previous methods of ECOC (except centroid relatively) neither decomposition nor reconstruction are affected by the problem. In other words we were not concerned with this problem, when we designed the system. It could be a critical dis-advantage if the problem is unusual. For example when the number of patterns in different classes is highly different, or the overlap between some classes is much more than others.

Linear combining of estimates can compensate parts of this shortcoming when the number of patterns in each class is not the same. This is due to the fact that training samples (representing some information about the problem) affect the
weighting factors and change the behaviour of composite classifier.

### Linear estimating Algorithm

**In training**
- Train \( b \) binary classifier based on superclasses defined by code matrix \( Z \)
- Find the number of pattern in each class \( i \) which belongs to first super class defined by \( j \) column of \( Z \)
- Normalise all the numbers in each class to make them as probability values to make the weighting factors matrix \( W \)

\[
W = \begin{bmatrix}
    w_{11} & w_{12} & \ldots & w_{1b} \\
    \ldots & \ldots & \ldots & \ldots \\
    w_{k1} & \ldots & \ldots & w_{kb}
\end{bmatrix}
\]

**In inferring**
- Applying new pattern \( x \), find the output of all experts:

\[
y = [y_1, y_2, \ldots, y_b]
\]

**For decision making:**
- Estimate probably of class membership \( i \) as a linear combination (weighted averaging) of the probability of all first super classes (the following formula):

\[
\hat{q}_i = \sum_{j=1}^{b} y_j w_{ij} \quad \text{for} \quad i = 1, 2, \ldots, k
\]
- Assign \( x \) to the class which corresponded probability is maximum

### 4.7 Dempster-Shafer mechanism

The theory of evidence combining has been introduced by Glenn Shafer in 1976. This theory is the basis of a mathematical framework to represent and solve problems involving uncertainty. This theory is mostly used in knowledge-based
expert system, because many other existence techniques in such systems cannot deal with uncertainty.

On the other hand this theory can be assumed as a generalised form of Bayesian statistics. So it can be used in problems which deal with probabilities. As we have shown in ECOC, the final solution should be found by combining the probabilities estimated by binary experts, so this technique seems capable of doing this task.

The main point here is the fact that all base classifiers are binary and we assume there is no uncertainty in their results, so combining their results need simple procedure. Let us shed light on this idea by an example:

4.7.1 An example on application of Dempster-Shafer theory in ECOC

Consider our previous three class problem in which we want to assign three samples with the true probability of class memberships given as follows:

<table>
<thead>
<tr>
<th>class</th>
<th>pattern1</th>
<th>pattern2</th>
<th>pattern3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>0.4*</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.5*</td>
<td>0.25</td>
<td>0.15</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>0.35</td>
<td>0.55*</td>
</tr>
</tbody>
</table>

The class nominated by Bayesian classifier is marked by (*):

We use $z_1$ to decompose the problem into three binary problems:

$$ z_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} $$

Based on $z_1$, for first binary classifier, first super-class is made by members of class 1 and 2 while second super class contains members of class 3. This can be done for all other columns as well. The exact value outputs of binary classifiers
can be found by adding the probabilities:

<table>
<thead>
<tr>
<th>pattern</th>
<th>expert1</th>
<th>expert2</th>
<th>expert3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8</td>
<td>0.65</td>
<td>0.45</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>3</td>
<td>0.7</td>
<td>0.6</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Now having the output of perfect base classifiers, and decomposition matrix code $z1$ we will combine these outputs one by one to achieve final assignment.

As an example for the first pattern:

1. Starting with the first column, the probability of belonging to class ('1' or '2') is 80% and to class '3', is 20%.

2. The information of second column can be added according to the following table:

<table>
<thead>
<tr>
<th></th>
<th>('1', '2')</th>
<th>('3')</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%80</td>
<td>%20</td>
</tr>
<tr>
<td>('1', '3')</td>
<td>%65</td>
<td>%52</td>
</tr>
<tr>
<td></td>
<td>%35</td>
<td>%28</td>
</tr>
<tr>
<td></td>
<td>%35</td>
<td>%28</td>
</tr>
</tbody>
</table>

In this table the sequence of classes defined by the decomposition matrix, is compared to find the common sub-sequences. For example the common part of ('1', '2') and ('1', '3') is ('1'), and its probability (mass function) of this sequence is the product (.65 × .8 = .52). If a sequence appears more than once the probabilities are cumulated, and for the case that the common part is $\emptyset$ corresponding mass function should not be considered and other mass functions should be normalised. In general mass functions can be found from the following formulas:

$$m(A) = \frac{1}{K} \times \left( \sum_{X \cap Y = A} m_1(X) \times m_2(y) \right)$$  \hspace{1cm} (4.26)
where $K$ is the normalising factor:

$$K = 1 - \sum_{X \cap Y = \emptyset} m_1(X) \times m_2(y)$$

(4.27)

3. Applying the third column, we will have:

\[
\begin{array}{c|c|c|c}
\text{Column} & ('1') & ('2') & ('3') \\
\hline
('2', '3') & 0 & ('2') & ('3') \\
%70 & 39.14 & 21.08 & 9.79 \\
\text{('1')} & ('1') & 0 & 0 \\
%30 & 16.77 & 9.03 & 4.19 \\
\end{array}
\]

4. For the fourth column:

\[
\begin{array}{c|c|c|c}
\text{Column} & ('1') & ('2') & ('3') \\
\hline
('3') & 0 & 0 & ('3') \\
%20 & 7.04 & 8.85 & 4.11 \\
('1', '2') & ('1') & ('2') & 0 \\
%80 & 28.16 & 35.4 & 16.44 \\
\end{array}
\]

The final mass function of these classes after normalising is: $[0.4161 \ 0.5231 \ 0.607]$. It means that class '2' will be assigned by composite classifier which is the correct one. the same procedure leads to correct assignment for the second and third patterns as well.

There are two more advantages for this framework:

1. In this framework it is possible to apply the uncertainty, for example if we use other decomposing methods (like pairwise coupling) in which a given pattern can belong to the first class, the second one or non of them, this framework can deal with this information.

2. If we have some information about the reliability of results of binary classifiers this framework can use them to improve performance.
Chapter 4. Reconstruction in Error Correcting Output Code

### Dempster-Shafer Algorithm for ECOC

**In training**
- Train $b$ binary experts based on sub-problems defined by columns of $Z$

**In inferring**
- Apply given pattern $x$ to all experts to make the output vector: $y = [y_1, y_2, ..., y_b]^T$
- Set $A_1$ as the set of “1”s in first column
- Set mass function of first expert $m_1(A_1) = [y_1, 1 - y_1]$
- Until the “$b$”th column:
  - Set $A_2$ as the set of “1”s in the “$j$”th column
  - Set mass function of $j$th expert $m_2(A_2) = [y_j, 1 - y_j]$
  - Combine the mass functions by **orthogonal summation**
    $$m(A_{12}) = \frac{1}{K} \times (\sum_{A_1 \cap A_2 = A_{12}} m_1(A_1) \times m_2(A_2))$$
    where $K$ is the normalising factor:
    $$K = 1 - \sum_{A_1 \cap A_2 = \emptyset} m_1(A_1) \times m_2(A_2)$$
- Set $A_1 = [A_{12}]$ and $m_1(A_1) = m(A_{12})$

**For decision making**
- Assign $x$ to class with maximum corresponding mass function

### 4.8 Summary of chapter

In this chapter we reviewed the problem of combining the results of sub-problems of ECOC. Providing a quick review of previous works some shortcomings of them were:

1. The relationship between distance measurement and class scores in ECOC was not found.
2. Code generation methods are not concerned with reconstruction of scores.

3. There was no theoretical framework to evaluate this method.

4. For alternative combining framework like least squares, it was not clear what is the condition set for a good code.

What we have reported in this chapter is:

1. Introducing a framework to explain the relationship between scores and distances in ECOC; which leads us to:
   
   (a) New condition set for code selection (equi-distance codes) (section 4.2)
   (b) A practical technique to select code words (section 4.2.5.2)
   (c) A complete framework for error analysis (sections 4.2.2, 4.2.3 and 4.2.4)
   (d) A new framework to improve ECOC performance with any code (section 4.2.6)

2. Finding the condition set of good code for least squares method (section 4.4.2)

3. Introducing the evidence combining framework for ECOC components (section 4.7)

In chapter 5, we experimentally show the effect of different codes on different combining frameworks with the intention of determining how the strategies developed in this chapter can change the performance of this technique.
Chapter 5

Some Experimental Examples

5.1 Introduction

After presenting some theoretical analysis of previous works on ECOC in sections 4.2, we concluded that their main shortcoming is the fact that they are not concerned with reconstruction of soft-level information (scores). We can therefore make the following comments:

1. In code selection the main consideration has been row and column separation [15, 16] which is based on information theoretic concept or simply recommending random code to provide suitable condition for a simple error analysis [37].

2. It is not clearly explained how code selection will affect the final performance. This is mainly due to lack of theory in error analysis in ECOC which is a modular combining method.

3. Two main developments for ECOC have been:

   (a) Least-squares methods in which soft-level information will be reconstructed. But the effect of decomposition (code) matrix in its performance has not been investigated [44].
Chapter 5. Some Experimental Examples

(b) Substitution (discussed in 4.1.1) which is essentially an equivalent to ECOC in which an estimation of class membership will be made independently using some weights for each expert output. Then using a simple averaging a final estimation can be found. The error analysis can cover the class estimation and not expert outputs.

We have provided a theoretical framework which explains the relationship between final scores and distance measurements in ECOC. This leads us to suggest that:

1. New condition set for code selection in ECOC (using equi-distance codes) can provide:
   
   (a) Zero reconstruction error (section 5.2.1.3)
   
   (b) Minimum bias effect (section 5.2.1.4)
   
   (c) An estimation of final variance (section 5.3)

2. Equi-distance code can optimise the performance of least-squares error (sections 5.2.1.5, 5.2.1.4 and 5.2.1.3)

3. Circular perturbing can improve the performance of ECOC for any code (section 5.2.1.7)

4. New combining frameworks (solving matrix equation and using Dempster-shafer mechanism can combine the binary expert outputs (sections 5.2.1.5, 5.2.1.4 and 5.2.1.3)

Now we provide some experiments to demonstrate that these novel ideas do in fact lead to improved performance for artificial and real data.

5.1.1 Benchmark considerations

In many cases, evaluation of the techniques that we suggest for improving performance is not analytically easy (or even possible). In these cases a popular
approach to evaluate the method is to test it over some problem, and to compare its performance with existing benchmarks. Benchmarks are artificial or real problems for which the results of various methods are available.

There are some points to note about these benchmarks:

1. It would be desirable to have a target and then show how close our design is to this goal as reference. In most cases there is no known reference or ideal performance for the problem. Although for posterior probability based classifiers, the Bayesian classifier is the best one, its implementation is not necessarily easy for many problems.

2. In many cases a comparison with current methods is used to evolve a new technique; and if the method exhibits better performance on a few examples, it is regarded as acceptable. But we may ask: how much better and over how many samples? Although there is no universal theoretical criteria for this purpose, significance test is a popular approach. Having the rate of error (percentage) of previous method $e_1$ over certain number of test samples $N$, the significance parameter can be calculated from an experimental formula:

   \[ S = \sqrt{e_1(1-e_1)}/N, \]

   If the error rate of new design is less than $e_1 - S$, this design is acceptable [13].

3. Evaluation of different designs is mostly based on average rate of error which is not useful to investigate the behaviour of system in different parts of input space. In some reported works the performance over certain partitions of problem is needed. For example error rate over the members of a certain class ([74] can be used in designing the combiner. Visualising the decision boundary of a classifier can be very useful but for multi-dimensional input is not practical. To investigate the points we have mentioned in previous chapter we have designed a two dimensional problem in order to visualise the decision boundaries.
5.2 Experiments in ECOC

In this part our goal is to investigate the role of following items in affecting performance:

1. Different combining techniques which were introduced in chapter 4.

2. Effect of code matrices with different properties on different parts of error, particularly:
   (a) Reconstruction error
   (b) Variance of binary estimators
   (c) Bias

It is obvious that the error of system relates to learning machine and training procedure, but our discussion has been general in previous chapter, so we should try to have experiments in which the effect of training is under control. In other words, here we want to evaluate just the above mentioned points, so we do not want to be involved with the effect of learning machine in our experiment. Therefore we use Bayesian binary classifiers as experts. To simulate the effect of imperfect learning, Gaussian noise with certain bias and variance will be added.

5.2.1 Introducing an artificial benchmark

We introduce an artificial benchmark in which we can find the result of Bayesian classifier as reference and visualise the decision boundaries to show the behaviour of ECOC. It is helpful to understand the behaviour of composite system in mimicking the Bayesian classifier.

Consider five groups of two dimensional random vectors having normal distribution as:

\[ p(x|c_i) = \frac{1}{2\pi\sigma_i^2} \exp\left(\frac{||x - \mu_i||^2}{-2\sigma_i^2}\right) \text{ for } i = 1, 2, ..., 5 \]

with parameters given in table 5.1.
Chapter 5. Some Experimental Examples

<table>
<thead>
<tr>
<th>class</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_i$ (mean)</td>
<td>[0,0]</td>
<td>[3,0]</td>
<td>[0,5]</td>
<td>[7,0]</td>
<td>[0,9]</td>
</tr>
<tr>
<td>$\sigma_i^2$ (variance)</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>25</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 5.1: Distribution parameter of data used artificial benchmark

Having a set of patterns consisting of equal number of patterns from each group, our goal is to classify them.

5.2.1.1 Base classifiers

Here our base classifiers are not made by training, but using the parameters from table 5.1 we will just find the posterior probability of class (or super-class) membership for each sample. Using equal number of patterns from each group for test set (equal prior probability for classes); Bayesian decision rule says:

\[
\text{assign } x \rightarrow w_i \quad \text{if} \quad P(w_i|x) = \text{ArgMax}_i(P(c_i|x))
\]

where $P(c_i|x)$ is the posterior probability of class membership for class $c_i$, and can be found by the Bayesian formula:

\[
P(c_i|x) = \frac{P(x|c_i)P(c_i)}{p(x)}
\]

Where $P(c_i)$ is the prior probability of class $i$ and $p(x)$ is the same for all classes (because of equal number of samples), so the decision rule can be changed:

\[
\text{assign } x \rightarrow c_i \quad \text{if} \quad p(x|c_i) = \text{ArgMax}_i(p(x|c))
\]

The Bayesian decision boundaries of different classes in this problem are shown in figure 5.1.

To simulate the behaviour of the system, Gaussian and uniform random data with different parameters will be added to the output of experts. For a fair comparison between different methods, the noise for each code matrix is produced once and is used in all reconstruction methods.
5.2.1.2 Selection of code matrix

To find a code with desired properties, we have used BCH method.

As mentioned before (section 4.2) error correcting capability is due to difference between labels, using an exhaustive search, codewords with equal number of "1" which have the same distance with each others are picked up to build our code matrices ($C_4, C_6, C_7$).

The following code matrices are in this experiment:

**C1**: a $5 \times 5$ unitary matrix (one per class which is very similar to normal classification)

**C2**: a $5 \times 7$ matrix with randomly chosen binary elements

**C3**: a $5 \times 7$ BCH code matrix with minimum distance of 3 bits between any pair of rows (1 bit error correction capability), but the distance is not the same in all cases
Chapter 5. Some Experimental Examples

C4: a $5 \times 7$ BCH code matrix with equal distance of 4 bits between any pair of rows (1 bit error correction capability)

C5: a $5 \times 15$ matrix with randomly chosen elements

C6: a $5 \times 15$ BCH code matrix with equal distance of 8 bits between any pair of rows (3 bit error correction capability)

C7: a $5 \times 31$ BCH code matrix with equal distance of 16 bits between any pair of rows (7 bit error correction capability)

Columns with all "0" or "1" element have been removed because they seems useless from classification viewpoint for stable classifiers. Only in C7 code matrix complementary columns (which might appear to be useless in classification) exist in order to test this statement.

5.2.1.3 Reconstruction error

In first step we attempt to investigate reconstruction error in different combining methods. Here we assume that our binary experts are perfect classifiers, so we expect the behaviour to be like Bayesian classifier from composite system. We have used 5000 patterns (1000 sample from each group) for testing and the rate of correct classification for Bayesian classifier is 74.28%. The rate of matching for different methods (original ECOC, Centroid, Least squares, Analytic approach, Linear estimating and Dempster-Shafer mechanism) with Bayesian classifier are presented in table 5.2.

Analysis:

1. Methods of least squares and analytical solution for finding the probabilities has no reconstruction error for any code matrix. It means that for very low rate of errors, they provide better performance.

2. Equi-distance code matrices (C1, C4, C6, C7) provide zero reconstruction error for ECOC, and linear estimation.
3. Reconstruction error for non equi-distance code matrices in ECOC is lower than linear estimation because in these cases, the sequence of “1”s in different rows is not the same, and when we normalise the counted patterns of first super-classes, they will vary in different labels.

4. The amount of reconstruction error is related to the diversity of distance between rows, for example in C2 this distance is between 2 and 6, in C3 it is 3 or 4 and in C5 it is between 3 and 11. This is why the reconstruction error in C3 is lower and for C5 it is higher.

5.2.1.4 Effect of common error (biased measurement) in experts

To investigate the concept of biased distance measurement, we have added a fixed value (.5) to the normalised output of all binary experts. In this case the rate of correct classification for Bayesian classifier is 72.08%, rate of matching for different combining methods with Bayesian classifier are shown in table 5.3.

**Analysis**

1. Equi-distance code matrices which have the same number of “1” in all rows provide very low rate for methods of ECOC, least squares, analytical
Chapter 5. Some Experimental Examples

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>100.00</td>
<td>58.96</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>C2</td>
<td>53.88</td>
<td>49.08</td>
<td>86.88</td>
<td>63.90</td>
<td>74.68</td>
<td>57.66</td>
</tr>
<tr>
<td>C3</td>
<td>83.06</td>
<td>49.74</td>
<td>31.52</td>
<td>51.30</td>
<td>97.24</td>
<td>96.14</td>
</tr>
<tr>
<td>C4</td>
<td>98.64</td>
<td>76.72</td>
<td>98.64</td>
<td>98.64</td>
<td>98.64</td>
<td>94.40</td>
</tr>
<tr>
<td>C5</td>
<td>87.50</td>
<td>73.26</td>
<td>88.30</td>
<td>75.22</td>
<td>95.24</td>
<td>92.68</td>
</tr>
<tr>
<td>C6</td>
<td>100.00</td>
<td>86.08</td>
<td>100.00</td>
<td>100.00</td>
<td>91.82</td>
<td>91.82</td>
</tr>
<tr>
<td>C7</td>
<td>100.00</td>
<td>80.38</td>
<td>100.00</td>
<td>100.00</td>
<td>91.82</td>
<td>91.82</td>
</tr>
</tbody>
</table>

Table 5.3: matching rate(%) of different methods with Bayesian classifier in the presence of Gaussian noise with mean of 0.5 and zero variance

solution and linear estimation. But there is no guarantee to remove this error completely. Linear estimation shows the best performance in these circumstances followed by ECOC. This fact can be explained as follows:

In distance based methods (ECOC, centroid, and Analytic approach) a bias in \(y\) directly affects \(L\), so if this effect is not the same for all classes, (non-equidistance codes) it will cause some error and least squares essentially has been driven with the assumption that unbiased estimation are provided by experts. In linear estimation on the other hand, the effect of bias will be reduced by a normalising procedure.

2. Centroid seems to be more sensitive to bias. This is because the fact that in these circumstances an average output of experts can not make a model of distribution of patterns in various classes, so centroids found by simple averaging, are not suitable representatives of classes.

3. C3 in which the number of “1”s in rows are 3 and 4 provide better performance for ECOC but for linear estimation C3 looks better (the number of “1”s in rows is between 5 to 7). This leads us to the idea that for longer random codes, where the probability of “0” and “1” is the same, and normalised variance of distance between code words merged decreases, the
Table 5.4: matching rate (%) of different methods with Bayesian classifier in the presence of Gaussian noise with mean of zero and variance of .5

effect of code selection will be reduced.

5.2.1.5 Effect of variance

Adding Gaussian noise with variance of .5 and zero bias, the performance of Bayesian classifier is 71.82% while the rate of matching for our methods are presented in table 5.4.

Analysis

1. For equi-distance code results of ECOC, analytic approach and linear estimation we obtain almost the same performance. This fact has been observed in other experiments as well and can give us these points:

(a) The distance measurement $L_i$, calculated in ECOC (section 4.2) has a simple linear relationship with probability estimated in analytic approach $\hat{q}_i$ (section 4.5). So in fact there is same classification procedure in these two frameworks.

(b) Least squares method can reach the performance of other methods when it minimise the least squares of $\hat{q}_i$ (equi-distance codes) but not in other cases.
Table 5.5: matching rate (%) of different methods with Bayesian classifier in the presence of Gaussian noise with the mean of .2 and the variance of .5

2. With the same number of columns, equi-distance codes have better performance (comparison of C2, C3, and C4, or C5 and C6) in all these techniques. It seems that in this problem, equi-distance codes provide more separable transformation for classes.

3. For a 5 class problem, there are $2^5 = 32$ different cases, removing the all “1” and all “1” cases and considering 14 pairs of complementary sequences, the number of independent columns which can be used in classifier training is 14. Although the number of independent columns in C6 and C7 are the same, the results of C7 are significantly better than C6 in all cases (except Dempster-Shafer). This fact can be explained as a result of variance reduction in averaging. For Dempster-Shafer method the complementary columns provide opposite evidences so they can not be useful.

5.2.1.6 General cases

As the last example consider the case that noise is Gaussian with mean of .2 and variance of .5. The rate of Bayesian classifier is 71.88% and the matching rates of different methods are shown in table 5.5
5.2.1.7 Circular ECOC

As noted before for distance based methods (ECOC and Centroid), circular perturbing can lead to better error reduction by reducing the sensitivity to the code word selection and providing the possibility to repeat the training procedure resulting in lower variance.

To investigate this concept we applied this algorithm to 10000 samples of this problem. The noise which is added to the outputs is kept fixed in all steps of the algorithm so error reduction can not be due to simple repeating.

The rate of matching the result of Centroid and circular Centroid are given in table 5.6 and for ECOC and circular ECOC in table 5.7.

As it has been shown clearly this perturbing can improve the performance significantly.

Probability based methods can not benefit for this algorithm as it has been shown in table 5.8 for least squares method as an example.
Chapter 5. Some Experimental Examples

Table 5.7: Comparison of ECOC and circular ECOC (second rows) in the presence of Gaussian noise ($\sigma=.5$, mean of .2)

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.00</td>
<td>91.80</td>
<td>92.60</td>
<td>100.00</td>
<td>0.9720</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>100.00</td>
<td>99.00</td>
<td>99.80</td>
<td>100.00</td>
<td>0.9980</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>.25</td>
<td>8.560</td>
<td>83.80</td>
<td>85.40</td>
<td>91.00</td>
<td>91.20</td>
<td>93.40</td>
<td>95.80</td>
</tr>
<tr>
<td></td>
<td>100.00</td>
<td>98.40</td>
<td>99.40</td>
<td>100.00</td>
<td>99.40</td>
<td>100.00</td>
<td>99.80</td>
</tr>
<tr>
<td>.5</td>
<td>59.80</td>
<td>63.80</td>
<td>63.60</td>
<td>66.20</td>
<td>77.00</td>
<td>81.00</td>
<td>89.80</td>
</tr>
<tr>
<td></td>
<td>99.80</td>
<td>99.40</td>
<td>99.20</td>
<td>99.80</td>
<td>99.80</td>
<td>99.80</td>
<td>100.00</td>
</tr>
<tr>
<td>.75</td>
<td>46.60</td>
<td>48.60</td>
<td>48.60</td>
<td>54.60</td>
<td>60.40</td>
<td>64.80</td>
<td>79.20</td>
</tr>
<tr>
<td></td>
<td>99.80</td>
<td>99.20</td>
<td>99.00</td>
<td>99.20</td>
<td>99.20</td>
<td>99.40</td>
<td>99.00</td>
</tr>
</tbody>
</table>

5.3 Variance reduction

The aim of this experiment is to exhibit the application of variance analysis (section 4.2, equation 4.5). To show the effect of ECOC on final variance, we have considered an ECOC framework using an equi-distance code (C4) in decomposition stage:

$$C_4 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

In this code the number of columns ($b$) is 7 and Hamming distance between any pair of rows is 4. In this experiment, we apply 100 patterns from each class to Bayesian binary experts; and add Gaussian noise with zero mean and fixed variance.

The variance of soft-level error of base classifiers are:

$$\sigma^y = \{.029 .0263 .0293 .0273 .0230 .0205\}, \quad \bar{\sigma}^y = .028$$
Chapter 5. Some Experimental Examples

Table 5.8: Comparison of least squares and circular least squares (second rows) in the presence of Gaussian noise ($\sigma = .5$, mean of .2)

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.00</td>
<td>94.44</td>
<td>99.10</td>
<td>100.00</td>
<td>97.74</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>.25</td>
<td>83.54</td>
<td>23.76</td>
<td>86.18</td>
<td>90.60</td>
<td>92.40</td>
<td>94.18</td>
<td>95.94</td>
</tr>
<tr>
<td></td>
<td>83.88</td>
<td>23.76</td>
<td>87.68</td>
<td>90.76</td>
<td>91.88</td>
<td>94.12</td>
<td>96.50</td>
</tr>
<tr>
<td>.5</td>
<td>56.28</td>
<td>50.38</td>
<td>62.80</td>
<td>69.00</td>
<td>75.18</td>
<td>80.96</td>
<td>89.96</td>
</tr>
<tr>
<td></td>
<td>57.26</td>
<td>52.74</td>
<td>62.76</td>
<td>69.90</td>
<td>71.96</td>
<td>82.16</td>
<td>89.08</td>
</tr>
<tr>
<td>.75</td>
<td>43.62</td>
<td>41.92</td>
<td>49.00</td>
<td>52.66</td>
<td>60.88</td>
<td>65.56</td>
<td>79.06</td>
</tr>
<tr>
<td></td>
<td>44.12</td>
<td>41.66</td>
<td>49.06</td>
<td>52.84</td>
<td>56.40</td>
<td>65.36</td>
<td>78.68</td>
</tr>
</tbody>
</table>

First let us estimate the final variance using the equation 4.5:

$$\sigma_{eoc}^2 = \frac{b}{M^2} \sigma^2 (1 + (b - 1) \rho)$$

Considering the fact that the covariance of error between any two columns is zero make $\rho = 0$, and $\sigma_{eoc}^2 = \frac{L}{16} \cdot 0.0283 = .0123$.

Now we can find the distance measurements, and estimate the posterior probability:

$$L_t = M(1 - q_t) \quad \text{or} \quad \hat{q}_t = 1 - \frac{L_t}{M}$$

Comparing this output with the precise values of posterior probabilities we can find the real value of variance:

$$\sigma_{eoc}^2 = [0.0132 \ 0.0113 \ 0.0111 \ 0.0139 \ 0.0132] \quad \sigma_{eoc}^2 = .0125$$

which is very close to the estimated value.

In practical cases the problem is to estimate the covariance between the error of base classifiers.
5.4 Visualising decision boundaries

Visualising decision boundaries has been our main motivation to design this benchmark because it can show the behaviour of classifier in different parts of input space.

For understanding the difference between one-per-class and ECOC in decomposition we consider C1 and C4 codes and decision boundaries of super classes defined by them which are shown in figure 5.2(a) and 5.3(a) respectively.

Comparing these figures with figure 5.1, boundary provided by a multi-class classifier we can extract the following points:

1. In multi-class classifier, all the boundaries should be learned at once. But when the problem is decomposed, some parts of boundaries should be learned by individual classifiers. Figure 5.2(a) and 5.3(a) show that the decision boundaries in the latter case is simpler.

2. In one-per-class (Figure 5.2) each part of boundary is learned twice but in distributed code with higher number of columns (like ECOC) each part may be learned more times as it is shown in figure 5.3.

3. Figures 5.2(b) and 5.3(b) shows the mechanism of reconstruction in ECOC for no noise condition. It seems that lower number of columns can construct the whole decision boundary. In fact for a 5 class problem it can be accomplished by just 3 bits. But in the presence of noise (figures 5.4(a) and 5.4(b)) more columns provide better performance.

4. Figure 5.5 compares the boundaries of ECOC and circular ECOC. in which the boundary made by circular ECOC seems much more precise than ECOC. It is completely with agreement with the the results of table 5.7.
Figure 5.2: Decision boundaries of C1  (a) individual boundaries  (b) composite boundaries
Figure 5.3: Decision boundaries of C4  (a) individual boundaries  (b) composite boundaries
Figure 5.4: Decision boundaries of ECOC in the presence of Gaussian noise with \( \sigma = 0.25 \)  
(a) C1  (b) C4
Figure 5.5: Decision boundaries in the presence of Gaussian noise with $\sigma=0.25$

(a) ECOC    (b) circular ECOC
5.5 Experiments with real problems with MLP

Feed-forward neural networks such as multi layer perceptron and Radial basis functions have exhibited the following characteristics in many classification applications. They are not sensitive to noise, have the capability of parallel processing and do not need many different kinds of pre processing stages. Furthermore in many cases complex networks can exhibit a generalisation which can hardly be bettered by combining. The only unsolved problems are selecting the parameters and regularisation particularly for problems which need complex network. Using simpler network in a composite system is an alternative approach.

For practical experiments, we have used simple multi layer perceptron with one hidden layer including very low number of nodes to solve some popular classification problems. The specification of problems can be found in table 5.9. At first each problem is solved by direct method (using a single network with multi-class output). Because of shortage in number of samples each problem has been solve with different splits and the mean and standard deviation of correct classification rates are presented in table 5.10 (6th column). The results of ECOC with different code and combining methods can be found in tables which mentioned in 7th column.\footnote{Thanks to UCI Machine Learning repository}

5.5.1 Random Codes performance

In long random codes the average distance between codewords which provides the error correcting capability is increased. On the other hand in this case codeword length can have any value. Here we are going to find the performance of ECOC with random codes over these data bases. To show the effect of experts, their performance over binary sub-problems (standard deviation of correct classification rates) and final performance of ECOC and circular ECOC are presented in table 5.17. each problem has been solved 10 time to get more reliable results.
### Table 5.9: Databases used in this experiment

<table>
<thead>
<tr>
<th>data base</th>
<th>tot. samp.</th>
<th>Features</th>
<th>classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>1728</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>cmc</td>
<td>1473</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>gaus5</td>
<td>1000</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>segment.</td>
<td>2310</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>vehicle</td>
<td>846</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>zoo</td>
<td>101</td>
<td>18</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table 5.10: Specification of problems

<table>
<thead>
<tr>
<th>Data</th>
<th>classes</th>
<th>tr. samp.</th>
<th>ts. samp.</th>
<th>hid. nods</th>
<th>S. net rate/std(%)</th>
<th>table</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoo</td>
<td>7</td>
<td>50</td>
<td>51</td>
<td>1</td>
<td>49.02/7.07</td>
<td>5.11</td>
</tr>
<tr>
<td>car</td>
<td>4</td>
<td>50</td>
<td>1678</td>
<td>1</td>
<td>69.98/.19</td>
<td>5.12</td>
</tr>
<tr>
<td>vehicle</td>
<td>4</td>
<td>350</td>
<td>496</td>
<td>5</td>
<td>23.45/1.54</td>
<td>5.13</td>
</tr>
<tr>
<td>glass</td>
<td>6</td>
<td>100</td>
<td>114</td>
<td>2</td>
<td>31.58/3.16</td>
<td>5.14</td>
</tr>
<tr>
<td>satel.</td>
<td>6</td>
<td>1000</td>
<td>5435</td>
<td>2</td>
<td>22.53/17.27</td>
<td>5.15</td>
</tr>
<tr>
<td>gaus5</td>
<td>5</td>
<td>200</td>
<td>800</td>
<td>2</td>
<td>46.4/28.46</td>
<td>5.16</td>
</tr>
</tbody>
</table>

Table 5.10: Specification of problems
Chapter 5. Some Experimental Examples

Table 5.11: Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over “zoo” data base.

<table>
<thead>
<tr>
<th>code</th>
<th>bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic</th>
<th>L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>7</td>
<td>89.54</td>
<td>89.54</td>
<td>89.54</td>
<td>89.54</td>
<td>89.54</td>
<td>89.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.9903</td>
<td>5.9903</td>
<td>5.9903</td>
<td>5.9903</td>
<td>5.9903</td>
<td>5.9903</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>77.78</td>
<td>77.78</td>
<td>75.82</td>
<td>43.14</td>
<td>46.41</td>
<td>77.12</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>88.89</td>
<td>88.89</td>
<td>86.27</td>
<td>84.97</td>
<td>88.89</td>
<td>88.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.3030</td>
<td>6.3030</td>
<td>5.1877</td>
<td>2.2641</td>
<td>6.3030</td>
<td>6.8860</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>86.27</td>
<td>86.27</td>
<td>86.27</td>
<td>86.27</td>
<td>84.97</td>
<td>10.7992</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.9854</td>
<td>8.9854</td>
<td>8.9854</td>
<td>8.9854</td>
<td>8.9854</td>
<td>8.9854</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>94.77</td>
<td>94.77</td>
<td>94.12</td>
<td>94.12</td>
<td>77.12</td>
<td>94.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.9951</td>
<td>2.9951</td>
<td>3.3962</td>
<td>3.3962</td>
<td>9.6723</td>
<td>2.9951</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>93.46</td>
<td>93.46</td>
<td>93.46</td>
<td>93.46</td>
<td>93.46</td>
<td>92.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.9951</td>
<td>2.9951</td>
<td>2.9951</td>
<td>2.9951</td>
<td>2.9951</td>
<td>4.0817</td>
</tr>
</tbody>
</table>
Table 5.12: Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over “car” data base.

<table>
<thead>
<tr>
<th>code</th>
<th>bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic</th>
<th>L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>4</td>
<td>72.15</td>
<td>72.15</td>
<td>72.15</td>
<td>72.15</td>
<td>72.15</td>
<td>72.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.83</td>
<td>4.83</td>
<td>4.83</td>
<td>4.83</td>
<td>4.83</td>
<td>4.83</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>73.60</td>
<td>73.60</td>
<td>73.14</td>
<td>72.63</td>
<td>38.99</td>
<td>73.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.93</td>
<td>0.93</td>
<td>0.84</td>
<td>1.21</td>
<td>27.42</td>
<td>0.70</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>72.96</td>
<td>72.96</td>
<td>72.11</td>
<td>71.90</td>
<td>72.96</td>
<td>73.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.62</td>
<td>3.62</td>
<td>2.86</td>
<td>2.79</td>
<td>3.70</td>
<td>3.69</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>74.16</td>
<td>74.16</td>
<td>74.16</td>
<td>74.16</td>
<td>74.16</td>
<td>74.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.70</td>
<td>3.70</td>
<td>3.70</td>
<td>3.70</td>
<td>3.70</td>
<td>3.32</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>74.33</td>
<td>74.33</td>
<td>74.41</td>
<td>74.55</td>
<td>71.22</td>
<td>74.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.35</td>
<td>2.35</td>
<td>2.31</td>
<td>2.39</td>
<td>0.81</td>
<td>1.52</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>72.79</td>
<td>72.79</td>
<td>72.79</td>
<td>72.70</td>
<td>72.79</td>
<td>74.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.65</td>
<td>2.65</td>
<td>2.65</td>
<td>2.65</td>
<td>2.65</td>
<td>2.78</td>
</tr>
</tbody>
</table>

Table 5.13: Results (average classification rate on 10 independent run and standard deviation) of different methods of reconstruction and different codes over “vehicle” data base.

<table>
<thead>
<tr>
<th>code</th>
<th>bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic</th>
<th>L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>4</td>
<td>62.77</td>
<td>62.77</td>
<td>62.77</td>
<td>62.77</td>
<td>62.77</td>
<td>62.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.65</td>
<td>8.65</td>
<td>8.65</td>
<td>8.65</td>
<td>8.65</td>
<td>8.65</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>66.94</td>
<td>66.94</td>
<td>67.41</td>
<td>61.22</td>
<td>67.20</td>
<td>68.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.42</td>
<td>4.42</td>
<td>4.86</td>
<td>5.54</td>
<td>3.44</td>
<td>2.33</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>53.02</td>
<td>53.02</td>
<td>58.74</td>
<td>57.12</td>
<td>54.23</td>
<td>51.88</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>69.15</td>
<td>69.15</td>
<td>69.15</td>
<td>69.15</td>
<td>69.15</td>
<td>68.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.62</td>
<td>5.62</td>
<td>5.62</td>
<td>5.62</td>
<td>5.62</td>
<td>4.96</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>73.32</td>
<td>73.32</td>
<td>73.72</td>
<td>73.72</td>
<td>68.88</td>
<td>73.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.78</td>
<td>2.78</td>
<td>2.92</td>
<td>3.82</td>
<td>4.51</td>
<td>2.52</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>75.34</td>
<td>75.34</td>
<td>75.34</td>
<td>75.34</td>
<td>75.34</td>
<td>76.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.81</td>
<td>1.81</td>
<td>1.81</td>
<td>1.81</td>
<td>1.81</td>
<td>1.65</td>
</tr>
</tbody>
</table>
Table 5.14: Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "glass" data base.

<table>
<thead>
<tr>
<th>Code</th>
<th>Bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>7</td>
<td>58.77</td>
<td>11.40</td>
<td>58.77</td>
<td>58.77</td>
<td>58.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.77</td>
<td>11.40</td>
<td>58.77</td>
<td>58.77</td>
<td>58.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.77</td>
<td>11.40</td>
<td>58.77</td>
<td>58.77</td>
<td>58.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.77</td>
<td>11.40</td>
<td>58.77</td>
<td>58.77</td>
<td>58.77</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>48.54</td>
<td>11.90</td>
<td>47.08</td>
<td>30.41</td>
<td>19.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48.54</td>
<td>11.90</td>
<td>13.93</td>
<td>9.64</td>
<td>13.16</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>38.30</td>
<td>6.22</td>
<td>39.18</td>
<td>30.18</td>
<td>37.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>38.30</td>
<td>6.22</td>
<td>3.51</td>
<td>3.54</td>
<td>6.22</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>52.63</td>
<td>9.48</td>
<td>52.63</td>
<td>52.63</td>
<td>52.63</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>59.94</td>
<td>4.83</td>
<td>60.53</td>
<td>59.94</td>
<td>59.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59.94</td>
<td>4.83</td>
<td>4.01</td>
<td>2.53</td>
<td>4.64</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>52.92</td>
<td>4.14</td>
<td>52.92</td>
<td>52.92</td>
<td>52.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>52.92</td>
<td>4.14</td>
<td>4.14</td>
<td>4.14</td>
<td>4.14</td>
</tr>
</tbody>
</table>

Table 5.15: Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "satellite" data base.

<table>
<thead>
<tr>
<th>Code</th>
<th>Bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>7</td>
<td>65.05</td>
<td>17.29</td>
<td>65.05</td>
<td>65.05</td>
<td>65.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65.05</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65.05</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65.05</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
<td>17.29</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>80.29</td>
<td>6.915</td>
<td>48.73</td>
<td>23.91</td>
<td>41.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80.29</td>
<td>6.915</td>
<td>4.88</td>
<td>2.30</td>
<td>7.77</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>70.06</td>
<td>10.42</td>
<td>68.94</td>
<td>62.67</td>
<td>65.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>70.06</td>
<td>10.42</td>
<td>10.92</td>
<td>6.96</td>
<td>10.71</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>69.48</td>
<td>3.88</td>
<td>69.48</td>
<td>69.48</td>
<td>69.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.48</td>
<td>3.88</td>
<td>3.88</td>
<td>3.88</td>
<td>3.88</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>77.74</td>
<td>4.31</td>
<td>77.74</td>
<td>77.74</td>
<td>76.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.74</td>
<td>4.31</td>
<td>5.04</td>
<td>4.98</td>
<td>2.08</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>80.43</td>
<td>1.73</td>
<td>80.43</td>
<td>80.43</td>
<td>80.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80.43</td>
<td>1.73</td>
<td>1.73</td>
<td>1.73</td>
<td>1.73</td>
</tr>
</tbody>
</table>
Chapter 5. Some Experimental Examples

### Table 5.16: Results (average and standard deviation of classification rate on 10 independent run) of different methods of reconstruction and different codes over "Gaussian5" data base.

<table>
<thead>
<tr>
<th>code</th>
<th>bit number</th>
<th>ECOC</th>
<th>Cent.</th>
<th>Analytic</th>
<th>L. sqrs</th>
<th>Linear</th>
<th>Demsp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-P-C</td>
<td>7</td>
<td>65.47</td>
<td>65.47</td>
<td>65.47</td>
<td>65.47</td>
<td>65.47</td>
<td>65.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.70</td>
<td>1.70</td>
<td>1.70</td>
<td>1.70</td>
<td>1.70</td>
<td>1.70</td>
</tr>
<tr>
<td>random</td>
<td>7</td>
<td>61.90</td>
<td>61.90</td>
<td>64.15</td>
<td>61.76</td>
<td>52.96</td>
<td>62.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.64</td>
<td>8.64</td>
<td>5.34</td>
<td>3.63</td>
<td>5.47</td>
<td>5.61</td>
</tr>
<tr>
<td>BCH</td>
<td>7</td>
<td>63.10</td>
<td>63.10</td>
<td>62.28</td>
<td>62.56</td>
<td>61.03</td>
<td>63.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.6851</td>
<td>6.6851</td>
<td>7.0741</td>
<td>5.8713</td>
<td>5.6883</td>
<td>6.7900</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>7</td>
<td>65.22</td>
<td>65.22</td>
<td>65.22</td>
<td>65.22</td>
<td>65.22</td>
<td>65.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>1.46</td>
<td>0.63</td>
</tr>
<tr>
<td>random</td>
<td>15</td>
<td>62.92</td>
<td>62.92</td>
<td>62.76</td>
<td>63.13</td>
<td>64.47</td>
<td>62.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.66</td>
<td>4.66</td>
<td>5.57</td>
<td>5.39</td>
<td>5.02</td>
<td>4.15</td>
</tr>
<tr>
<td>eq-BCH</td>
<td>15</td>
<td>68.41</td>
<td>68.41</td>
<td>68.41</td>
<td>68.41</td>
<td>68.41</td>
<td>68.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.73</td>
<td>1.73</td>
<td>1.73</td>
<td>1.73</td>
<td>1.73</td>
<td>1.33</td>
</tr>
</tbody>
</table>

Results in table 5.9 and 5.17 clearly show that both original and circular algorithm can improve the performance of classification. This improvement in circular ECOC is significantly greater in all cases (significance test given in section 5.1.1 thresholds are(%): car 77.67; cmc 51.08; gaus5 67.6; glass 54.6; iris 85.41; segmentation 79.13; vehicle 71.44; zoo 92.58). On the other hand in original ECOC performance is more sensitive to the code matrix (compare the results in each row in table 5.9). Furthermore, using a simple network in this framework, instead of a single but more complex network which perhaps can exhibit comparable performance, we have by-passed the problems of parameter selection and regularisation. This is critical particularly in automatic algorithms where less interaction with designer is important.
### Table 5.17: Mean and standard deviation of correct classification rate of MLP with given number of hidden nodes with ECOC and circular ECOC

<table>
<thead>
<tr>
<th>data base</th>
<th>tr. samp</th>
<th>h.nds</th>
<th>cols.</th>
<th>single</th>
<th>Orig.</th>
<th>Circ</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>100</td>
<td>1</td>
<td>15</td>
<td>75.43</td>
<td>77.25</td>
<td>78.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.82</td>
<td>3.98</td>
<td>3.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmc</td>
<td>100</td>
<td>1</td>
<td>15</td>
<td>44.8</td>
<td>49.5</td>
<td>51.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.37</td>
<td>4.3</td>
<td>4.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gaus5</td>
<td>200</td>
<td>2</td>
<td>15</td>
<td>62.95</td>
<td>67.27</td>
<td>67.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>1.37</td>
<td>1.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>glass</td>
<td>100</td>
<td>2</td>
<td>15</td>
<td>47.37</td>
<td>53.56</td>
<td>56.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.27</td>
<td>2.03</td>
<td>4.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iris</td>
<td>50</td>
<td>1</td>
<td>15</td>
<td>79.8</td>
<td>84.9</td>
<td>95.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>27.88</td>
<td>14.45</td>
<td>2.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>segment.</td>
<td>200</td>
<td>1</td>
<td>50</td>
<td>44.12</td>
<td>78.84</td>
<td>85.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.33</td>
<td>5.84</td>
<td>2.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vehicle</td>
<td>350</td>
<td>5</td>
<td>15</td>
<td>56.82</td>
<td>70.67</td>
<td>74.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.94</td>
<td>4.83</td>
<td>4.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zoo</td>
<td>51</td>
<td>1</td>
<td>15</td>
<td>70.4</td>
<td>92.2</td>
<td>94.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14.01</td>
<td>4.26</td>
<td>2.95</td>
</tr>
</tbody>
</table>

Chapter 5. Some Experimental Examples
5.6 Summary of chapter

In this chapter using some real and artificial benchmarks, we have proposed to investigate three subjects:

1. Behaviour of ECOC in decomposition and reconstruction with various code matrices, with different levels of bias and variance

2. Comparison of original ECOC with ECOC using other combining methods introduced in sections 4.3, 4.4, 4.5, 4.6 and 4.7.

3. Evaluating our variance analysis (section 5.3)

We have designed an artificial problem in which it is possible to find the behaviour of Bayesian classifier and other classifiers both in performance and visualising the decision boundaries. Such a problem gives us the opportunity to decompose different parts of error and investigate the effect of different combining methods and code matrices.

The results on some popular real classification data bases as well as our artificial benchmark indicate that:

1. For original ECOC using an equi-distance matrix can minimise the reconstruction error (table 5.2) and the effect of biased distance measurement (table 5.3).

For least-square method, equi-distance code can provide better performance as well (tables 5.2, 5.3, 5.4 and 5.5).

2. Circular ECOC, a novel technique introduced in section 4.2.6, can improve the performance of distance based methods like original and centroid ECOC (table 5.7).

3. The final variance in original ECOC with equi-distance code is function of codeword separation and column selection (section 5.3). So the next step
to improve code selection is to investigate the effect of columns on the covariance of their error.

4. Estimation of final scores (probabilities) can provide acceptable performance (tables 5.2, 5.3, 5.4 and 5.5). So they can be considered as alternatives of original distance based method.

5. Dempster-Shafer mechanism can be used to combine the results of local experts (tables 5.2, 5.3, 5.4 and 5.5). This technique opens up a new possibility to solve multi-class problems involving uncertainty.
Chapter 6

Fusing ECOC with other methods

6.1 Introduction

In chapters 3 and 4 we have explained ECOC concept and its source of effectiveness in improving the performance of classification. In chapter 5 we demonstrated experimentally on artificial and real data that theoretical points in chapter 4 are effective. However the data sets in chapter 5 are relatively small. Now we consider a more realistic problem with huge data set. There are some problems which are not suitable for ECOC:

1. ECOC can not improve the performance of dichotomous problems. In other words, binary classifiers can not be improved by this algorithm. This is because of the fact that the main procedure of this algorithm is to reduce the complexity of learning by mapping the multi-class problem into a few binary sub-problems and this can not happen for binary problems.

2. If the complexity of problem is not due to number of classes (for example involve large feature set), it is hardly possible to improve the performance significantly by converting it into a few dichotomous sub-problems.
On the other hand, one of the main advantages of ECOC is the fact that it does not need high number of training samples. This is because:

1. Subproblems in ECOC are less complex, and usually the learning machine has lower number of free parameters to set in training. As an example in neural networks, based on experimental criteria it is desirable that the number of training samples be ten times number of inner connections [28]. Fewer output nodes causes fewer connections and fewer required training samples.

2. While in many combining methods the use of different samples in training set is the main technique to make diverse experts, subproblems in ECOC use the same training set.

These points indicate that the shortcomings of ECOC can be compensated by adopting an appropriate procedure for designing the binary experts by:

1. Input decomposition for lowering the required complexity of experts.

Representing a pattern with a high number of features will cause problems such as curse of dimensionality. There are few methods to escape such a trap by manipulating the input features:

(a) Selecting more important patterns for a single classifier (feature selection). The main weak point of this method is the fact that some valuable parts of information can be missed by removing some features.

(b) Using an ensemble of experts each one using different feature sets and trained individually, then combining the results (for example [71]). The main problem of this approach is the fact that there is no guarantee that sub-sets can provide acceptable and diverse results.

(c) Decomposing the feature set and using them as input to local experts but they will be trained together.
Chapter 6. Fusing ECOC with other methods

We concentrate on decomposition of feature set for use in a multi-net system. The main idea is to train local networks as the component of a modular composite system simultaneously. This idea has been used in some methods which are based on divide and conquer strategy. For example in “mixture of experts” [35] local experts are trained together but with different samples. Here all the experts use the same training samples but with different representation. Introducing a multi-network system with local back propagation algorithm we will show that this system can be useful in learning of problems involving huge feature sets.

2. Improving the performance of binary classifiers can be achieved by using ensemble methods especially Boosting.

In [62] Schapire explain that one of the difficulties of AdaBoost is the fact that this algorithm is essentially designed for binary systems. Although methods have been introduce to use this algorithm for multi-class systems, Schapire believes that “the hybrid system he suggest (a fusion between ECOC and AdaBoost) may be significantly faster and require less programming efforts in creating the base learning algorithm”. In this new algorithm in each iteration, both training set and defined super classes will be changed. One possible short coming of this suggestion is the fact that in this way, it is possible that the number of training samples is not enough to train base classifiers. We have suggest an algorithm in which for a certain defined super classes (any column in code matrix) AdaBoost is used to reinforced the local experts learning.

In this chapter first we will explain these ideas and then will provide some experimental evidence to show effectiveness of these methods for a complex problem with a huge feature set.
6.2 Input decomposition

From the viewpoint of neural networks, relationship between desired output and input (features), should be learned and recorded as connection weights in training phase. In this viewpoint training is a search in the space of weights. Higher number of connections means more difficult search.

If some features have a special relation together and have some common properties that the others do not, it means some information can be extracted from these features (and not necessarily from all). So only a certain part of feature set is needed to train the network in training phase or provide the output in test phase. Otherwise the whole set is needed but if the features include relatively independent information, for example they are from different position of an image, some weights which connect them to next layer will be negligible or zero. Although for a network with high number of nodes and good learning procedure theoretically we suppose that the network will find all relations and record them in their weights (even those which are zero), in practice training such a network is not easy. In these circumstances focusing on local relationship between features which means ignoring some connections in advance, can be more effective, because:

1. For quadratic problems the derivation of cost function with respect to the weight vector will be too complex (other kinds of search like Genetic search or exhaustive search will have their difficulties as well).

2. A complex relation between cost function and weights vector may cause many local minima which causes difficulties in training. It needs more care in choosing learning parameters.

3. The required number of training example will increase as a function of number of connections. For example an experimental heuristic suggests a ten fold value for back propagation [28].

These facts led us to use a feature set decomposition in order to reduce the
number of inner connections. The problem we may face comes from the fact that in general the output of each local network is insufficient for solving the original problem. Using simple combining methods such as order statistics, linear combining and voting hardly benefits us, because the outputs are essentially from different types. Combining the output from different types could possible use ranked based methods([30]), only if each output were capable for giving the final answer.

6.2.1 Local error back propagation

The approach we suggest to solve such a problem is to use a network in which the values of some weights have been chosen in advance (say zero for example), so the search problem in training is simpler. Such a technique is shown in figure 6.1. In such a structure the original feature set $X$ is divided into three subsets $S1, S2$ and $S3$ each one is applied to one of local networks, $N1, N2$ or $N3$. The outputs of these networks will combined in a weighted averaging structure to provide the final output. The local weights as well as weighting factors which are used in combining can be found by methods like error back propagation.

To show this method we have used a data set("optdigits") with feature set of 64 features that is divided into 4 groups. From 3741 samples, 741 sample are used as training set and others for test set. Once a single network with 70 hidden nodes is used and another time a composite multi-net system (figure 6.1).

The learning parameters (learning rate, and momentum constant) have been chosen in such a way that the best possible training will occur for the single network.

The sum square error of two systems are shown in figure 6.2. The rate of correct classification for single network is 91.4%, and for composite system it is 94.2%.
6.3 Boosting Binary classifiers

Boosting refers to a general idea of producing an accurate prediction rule by combining rough and moderately inaccurate rules.

The essential question is whether a "weak" learner, which can performs just slightly better than random guessing, can be "boosted" into a strong learner.

As noted, training samples have a critical role in automatic classification systems. The assumption of some similarities between training samples and test samples is the base motivation of learning. The system is expected to extract the rules to distinguish between members of different classes, but because of the existence of some "private properties which separate the members of a certain class, it is not so easy. In learning tasks there are important phrases such as "enough examples" and "sufficiently many examples", but they are not quantitatively defined.

Having a set of examples $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ and a certain learning machine $\mathcal{L}$ we need a good generaliser. The point is, our learning machine can
Chapter 6. Fusing ECOC with other methods

Figure 6.2: Error reduction in learning (a) single net. with 70 hidden nodes (b) 4 net. with 10 hidden nodes
not extract all needed information for good generalisation. The main idea is the fact that if learning machine concentrates on difficult patterns it will learn to classify them. Boosting tries to implement such a strategy by answering two questions:

- To make each classifier, which samples should be picked?
- How should the result of each classifier be weighted in final decision making?

### 6.3.1 Traditional Boosting

Drucker and Schapire in 1993 had suggested the first version of Boosting as follows[17, 61]:

Ensemble is made by three classifiers. The first one is supposed to classify normal patterns but obviously it will fail on some patterns. The second one is expert on patterns the first one has failed over them, and the third classifier will judge between the two first.

To make such an ensemble, first we use a training set with the distribution of the whole data to train the first expert. Then we use some data to test this classifier. The error cases are used to train the second classifier but to avoid inverse correlation the training set of second expert includes 50% of original distribution and 50% of error cases. Some other samples should be used to test these two classifiers and then the third training set will contain 50% of original data and 50% patterns two previous experts did not agree over them.

This technique needs high number of training patterns and in many cases it causes some improvement in performance.

### 6.3.2 Adaptive Boosting

One main point in the Boosting described here is the fact that different examples have not the same effect in learning. For example we can divide patterns into
easy and hard to distinguish. From the practical viewpoint of classification, noise can affect the points near decision boundary seriously. It led us to pay more attention on hard points in training. In Boosting it has been done by providing more probability in appearance of error cases in second and third experts.

From this discussion we can conclude that the weights of different patterns in training should be different.

Here there are three main items in this kind of ensemble:

1. Finding appropriate weight for each pattern in training The only way to find appropriate weight of each sample is to repeat serial training. This is because we have no information about their importance in training in advance, but after looking at the behaviour of the trained expert we can find that they are “easy” or “hard”.

In practice in first iteration all patterns are hard so they have the same weight of importance, but after making the first trained expert, we can find which one is “easy” and reduce its weight and increase the weight of “hard” samples.

2. Applying the weight of samples in training Given a weight vector of samples $\mathbf{w} = \{w_1, w_2, ... w_N\}$, there are two ways to apply it in training of the next expert;

(a) In some learning machines like back propagation neural networks, a cost function is used to change the inner weights and behaviour of the network which mostly is sum squares of error of samples. (say $SSE = \sum_{i=1}^{N} (y_i - y_{di})^2$) where $y_i$ is the output of network given $x_i$ and $y_{di}$ is the desired output of this pattern. Since this cost function is the only way different patterns can change the direction of learning in training phase, when the cost function is simply related with the effect of input($y_i$), we can change the effect of each sample by just adding a weight in this cost function: $WSSS = \sum_{i=1}^{N} w_s (y_i - y_{di})^2$.
(b) The second way to change the direction of training with this vector is to change the distribution of training set by resampling with replacement. In this way we increase the probability of appearance of more important samples. This technique has some advantage in comparison with the first one:

i. In many learning machines the relation between the effect of inputs and cost function is not so simple so the first technique is not applicable.

ii. This method changes the effect of more important samples both with the presence in training of next expert and the appearance in other experts. In fact because of voting regime, the effect of hard sample will be duplicated.

iii. Resampling with replacement can make improvement by itself. For example in Bagging the weight of all samples is equal but because of resampling with replacement, any two sets have 63.2% overlap. It means that they can make different experts. If we have used the previous method in Bagging no improvement would be achieved.

3. **Weighted averaging or voting** The third main fact in such a composite system is the fact that the effect of different experts in final decision is not the same.

### 6.3.3 A quick Summary of AdaBoost

notations: $D$ the training set (original) with $N_1$ sample

$D^i$ the training set of $i$th iteration which is the original at first for other iterations the samples are picked up by resampling from the original patterns weighted by $p^i$

$p^i$ the vector including probability of occurrence of each pattern in $D^i$

$L(D^i)$ the learning machine (expert) that is trained by $D^i$
Chapter 6. Fusing ECOC with other methods

\( h^j \) hypothesis (decision) made by \( j \)th expert

\( h^j \) combined hypothesis of first \( j \)th experts

\( c(i) \) correct decision about \( i \)th input pattern

---

**AdaBoost Algorithm**

**Initialisation:**

\[ p^1(i) = \frac{1}{N_1} \text{ for } i = 1, \ldots, N_1 \]

FOR \( j = 1 \) to \( N_{\text{exp}} \)

\[
\begin{align*}
& h^j = L(D^j) \\
& \varepsilon^j = \sum_{i=1}^{N_1} |h^j(i) - c(i)|p^j(i) \\
& \beta^j = \frac{\varepsilon^j}{1 - \varepsilon^j} \\
& w^{j+1}(i) = \begin{cases} \\
   w^j(i) \beta^j & \text{if } \quad h^j(i) = c(i) \\
   w^j(i) & \text{otherwise} \\
\end{cases} \\
& p^{j+1} = \text{normalise}(w^{j+1}) \\
& h_f^{j+1}(i) = \begin{cases} \\
   1 & \text{if } \quad \sum_{m=1}^{j}(\log \frac{1}{p_m^j})h^j(i) \geq \frac{1}{2} \sum_{m=1}^{j} \log \frac{1}{p_m^j} \\
   0 & \text{otherwise} \\
\end{cases}
\end{align*}
\]

END-FOR

Each iteration of this algorithm can be separated into two main parts:

- **updating** in which learning machine \( L \) is trained on a new training set \( (D^j) \) following by changing \( p^j \) according to the decision of \( L(D^j) \) in current iteration.

- **decision making** in which each classifier \( L(D^j) \) has an associated weight based on the rate of correct classification, we call this a weighting factor to distinguish it from the weights of pattern in updating phase and the final decision is made by weighted voting.
6.3.4 Some Notes about AdaBoost

1. Learning-generalisation trade off

AdaBoost is designed to make a strong learner from weak learners. A learning machine is weak either because there are not enough free parameters to capture the essential properties or the search procedure in free parameter space (learning algorithm) fails to fit the samples. In other words the capability of weak learner is low and it can provide only simple hypothesis.

AdaBoost adds some complexity to system by using multi weak learners with different weighting factors, each of them is specialised on some set of patterns so the composite learner can fit to a complex distribution of data in training set by making a complex hypothesis. Two main rules in AdaBoost are:

(a) More chance to appear in training set for “difficult” patterns
(b) More weight in voting for successful experts

There are some comments about the way AdaBoost applies these rules:

- The influence of difficult patterns in final decision making is much more than normal patterns and the composite classifiers will be suitable for them, so it can destroy its performance over normal patterns. This is why a penalty term has been suggested to regularize the algorithm by preventing too much influence of difficult patterns[54].

- When an expert exhibits low error in updating mode, it will find a heavy weighting factor for all patterns. It seems that using a dynamic weighting can lead to better performance[75].

- Even when we use an alternative framework instead of logarithmic weighting factors, the generalisation improvement is not as good as expected; it means that the correlation reduction between experts is not successful in these circumstances, and so choosing the training set based on correlation between experts can be useful[75].
• In some cases when the error reaches to a certain bound such as zero (i.e., complete fitness), algorithm will terminate, but it is observed that carrying on the procedure will improve the generalisation in some cases [63].

• In some cases overfitting occurs with high error on training set, [53, 54], and it seems that it is an open area for investigation particularly in more complex experts like neural networks.

2. Using other combining frameworks

Other combining frameworks such as median, max and mean combiner exhibit better performance than weighted averaging in some applications, so it is well worth comparing the result of combining these methods with AdaBoost.

In updating phase AdaBoost tries to adapt each pattern individually and this is why each pattern has an independent weight, but in combining, each expert has a weighting factor which is fixed for all patterns. It seems that if the weighting factors of experts also adapt, perhaps performance of the system will improve. Dynamic weighting factors introduced in [74] is such an approach.

6.3.5 Examples of dynamic weighting

To show the idea we have used two artificial examples:

1. For traditional Boosting, we use a problem involving the classification of five groups of two dimensional random vectors (as in previous chapter) with following distribution parameters [74]:

The performance of Bayesian classifier is about 55.75%. Three MLP with 4 hidden nodes made by Boosting algorithm trained over different number of training samples (10% for training the first experts, 30% filtered by first expert to be used in training of second one, 60% filtered by these two to be used in training of the third one).
Chapter 6. Fusing ECOC with Other Methods

<table>
<thead>
<tr>
<th>class</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>[0,0]'</td>
<td>[0,2]'</td>
<td>[2,0]'</td>
<td>[4,0]'</td>
<td>[0,4]'</td>
</tr>
<tr>
<td>var.</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 6.1: distribution parameter of five group of data used in training and test set

A weighted averaging framework is used to combine the results. First an exhaustive search is used to find the optimum weighting factors and in the second approach a neural network “oracle” dictates the dynamic weighting factors. The rate of classification for experts and composite classifiers over 5000 test patterns are shown in table 6.2.

<table>
<thead>
<tr>
<th>perf.</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>cfix</th>
<th>cdyn</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>29.9</td>
<td>30.3</td>
<td>22.7</td>
<td>38.4</td>
<td>41.5</td>
</tr>
<tr>
<td>std</td>
<td>.359</td>
<td>.564</td>
<td>.385</td>
<td>.382</td>
<td>.189</td>
</tr>
<tr>
<td>mean</td>
<td>32.3</td>
<td>32.2</td>
<td>18.7</td>
<td>40.7</td>
<td>43.6</td>
</tr>
<tr>
<td>std</td>
<td>.617</td>
<td>.628</td>
<td>.486</td>
<td>.334</td>
<td>.235</td>
</tr>
<tr>
<td>mean</td>
<td>38.5</td>
<td>32.3</td>
<td>18.3</td>
<td>44.0</td>
<td>47.6</td>
</tr>
<tr>
<td>std</td>
<td>.42</td>
<td>.51</td>
<td>.385</td>
<td>.33</td>
<td>.19</td>
</tr>
</tbody>
</table>

Table 6.2: classification rates (%) for 300, 500 and 1000 pattern in training set

Although as it is shown in figures 6.3 and table 6.2 Dynamic weighting improves performance, using more accurate and powerful networks, this improvement will stop. In fact it is not hard to find a single network (more complex) that outperforms such a composite system.

2. For AdaBoost a problem including artificial set of two groups of two dimensional random vectors with Gaussian distribution (mean [0, 0], variance 1 and mean [2, 0], variance 4) [79] has been used.

Figure 6.4 shows decision boundaries of first expert, Bayesian classifier as
Chapter 6. Fusing ECOC with other methods

Figure 6.3: Decision boundaries: a) individual experts b) composite classifier with optimum fixed weighting factors c) composite classifier with dynamic weighting factors
Table 6.3: Classification rate (%) on artificial database 20 independent run and 10 experts -100 sample for training and 900 sample for test set

<table>
<thead>
<tr>
<th># centres</th>
<th>single</th>
<th>H-log</th>
<th>S-log</th>
<th>median</th>
<th>min/max</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>61.33</td>
<td>74.04</td>
<td>72.06</td>
<td>61.99</td>
<td>60.24</td>
<td>73.48</td>
</tr>
<tr>
<td>5</td>
<td>69.11</td>
<td>72.52</td>
<td>74.27</td>
<td>72.63</td>
<td>72.24</td>
<td>73.81</td>
</tr>
<tr>
<td>10</td>
<td>78.89</td>
<td>73.10</td>
<td>74.61</td>
<td>70.16</td>
<td>68.77</td>
<td>72.9</td>
</tr>
</tbody>
</table>

Figure 6.4: Comparison between decision boundaries of AdaBoost and Dynamic-AdaBoost

reference, AdaBoost and Dynamic weighting factors combiner, when RBF with different number of centres that are randomly chosen, is used as learning machine and 20 experts are produced by AdaBoost.

It is shown that although both AdaBoost and dynamic weighting combiner successfully mimic the Bayesian classifiers decision (central circle) specially where the number of patterns is high, dynamic weighting is closer to the Bayesian classifier.
Chapter 6. Fusing ECOC with other methods

<table>
<thead>
<tr>
<th># centres</th>
<th>single</th>
<th>H-log</th>
<th>S-log</th>
<th>median</th>
<th>min/max</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>65.66</td>
<td>66.92</td>
<td>67.74</td>
<td>68.12</td>
<td>69.26</td>
<td>70.39</td>
</tr>
<tr>
<td>20</td>
<td>73.23</td>
<td>68.8</td>
<td>68.18</td>
<td>66.1</td>
<td>69.99</td>
<td>67.61</td>
</tr>
<tr>
<td>30</td>
<td>73.23</td>
<td>73.18</td>
<td>73.12</td>
<td>72.39</td>
<td>71.60</td>
<td>73.11</td>
</tr>
</tbody>
</table>

Table 6.4: Classification rate (%) on diabetic database 20 independent run and 10 experts - 570 sample for training and 214 sample for test set

<table>
<thead>
<tr>
<th></th>
<th>single</th>
<th>H-Boost</th>
<th>S-Boost</th>
<th>median</th>
<th>min/max</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>66.41</td>
<td>67.27</td>
<td>67.9</td>
<td>68.14</td>
<td>66.24</td>
<td>87.67</td>
</tr>
<tr>
<td>std</td>
<td>8.46</td>
<td>7.25</td>
<td>7.35</td>
<td>7.52</td>
<td>8.86</td>
<td>8.03</td>
</tr>
</tbody>
</table>

Table 6.5: Classification rate of combining experts made by Boosting

As another example we have considered a popular classification data base ("diabetics") on which AdaBoost has shown relatively weak performance.

A comparison between tables 6.4 and 6.5 shows that the problem is in the way we have used to make the experts. This leads us to the idea that rather than focusing on the combining framework to improve the AdaBoost, we better find a way to reduce correlation between experts as much as possible. The idea of ECOC can be helpful in this manner in multi-class problems.

6.4 Experimental results on fusion of these methods

As mentioned before (section 6.1), ECOC has two main shortcomings:

1. Although ECOC can reduce the complexity of problem related to high number of classes, for problems with huge feature sets, there is not much gain to be achieved by ECOC. This is because the binary classifiers can not provide an output with desired precision.

1 Thanks to UCL Machine Learning repository
2. For binary classifiers ECOC can not do anything to improve the performance.

On the other hand, one interesting property of ECOC is the fact that this framework is independent of the inner works of its binary experts. So it seems that the first shortcoming can be solved in binary experts designing stage. To show these ideas we have introduced in the previous chapter, we solve a relatively complex problem in which such difficulties exist. The databases were chosen so that experiments which shows aspects mentioned here.

6.4.1 Experiments on Face4 face data base

We have developed our system on small Face4 face data base. The task here involves classifying camera images of faces of various people in various poses. Images of 20 different people were collected, including approximately 32 images per person, varying the person's expression (happy, sad, angry, neutral), the direction in which they are looking (left, right, straight ahead, up), and whether or not they were wearing sunglasses. There is also variation in background behind the person, the clothing worn by the person and the position of the person's face within the image.

In total, there are 624 grey scale images with a resolution of 120×128, with each image pixel described by a grey scale intensity value between 0 (black) and 255 (white).

6.4.1.1 Preprocessing

We have intended not to use any preprocessing stages which can affect the neural networks. However to reduce the volume of calculations we have reduced the size of images to 28×28. Resampling was based on nearest neighbours interpolation and no filter has been used to reduce the effects of aliasing during resampling. The resulting gray levels are directly applied to neural network as features.

\footnote{used in \cite{50} available in \url{http://WWW.cs.cmu.edu/~tom/mlbook.html}}
Images of a subject have been shown in figure 6.5 as example.

6.4.1.2 System description

The reported result of a neural network with 30 hidden nodes on images with the size of 32×30 to recognise face pose (left, straight right, up), a four class problem, has been 93% [50]. For face recognition, we used a single net with 550 hidden node which results in 45% correct recognition (higher number of nodes does not lead to better performance).

We have used a multi-network system with 30 hidden nodes boosted by AdaBoost in 5 iterations, as binary classifier in an ECOC framework with a random 20×100 code matrix. The multi-networks are trained by local back propagation algorithm with 100 epoch, and learning rate of 0.0001 with momentum of .5. The final result of the system is more than 99%.

The increase in performance as a function of number of columns is shown in figure 6.6 for normal ECOC and ECOC with binary experts using 5 iterations of
6.4.1.3 Discussion

In figure 6.6 we witness two different cases:

1. The area in which the number of local experts is under 20. In this area adding a new binary expert cause a considerable improvement in performance.

2. The second area (after about 20 experts) the improvement of performance is less.

To explain this, we should consider that ECOC decomposition is a technique to control the injection of label (class) information to the system. In fact since our system can handle all the class information, we divide this information and provide different parts in different iterations.

In the first area (when the number of experts is less than 20), the provided class information is not enough to solve the final problem (even if we use perfect
experts), this is why adding a single column of code matrix can improving the performance so dramatically.

In the second part on the other hand, all class information needed for classification has been provided but the effect of imperfect experts causes some error which should be removed by adding redundancy (more columns).

For longer code matrices the effect of code selection will decrease because even for a random code all information will be given sooner or later.

### 6.4.2 The ORL face data base

A second set of experiments were carried out using the “Olivetti Research Ltd (ORL) face data base” [60](figure 6.7. The database consists of 400 images, 10 each of 40 different subjects. The subjects are either Olivetti employees or Cambridge University students. The age of the subjects ranges from 18 to 81, with the majority subjects being aged between 20 and 35. There are 4 female and 36 male subjects. Subjects were asked to face the camera and no restrictions were imposed on expression; only limited side movement and limited tilth were tolerated. For most subjects the images were shot at different times and with different lightning conditions, but always against a dark background. Some subjects are captured with and without glasses. The images were manually cropped and rescaled to a resolution of 92×112, 8-bit grey levels. Five images of each subject were selected randomly for use in training and five for testing which means a total 200 training and 200 test images. This database has been used in experiments reported in [60] with HMM with an identification rate between 97% to 98.5%.

Like the previous experiment the only pre processing stage includes a resampling based on nearest neighbours interpolation to reduce the size of images, and the resulting gray levels are directly applied to neural network as features. Four different systems are compared on this data base:

1. A single network to solve multi-class problem. Back propagation algorithm with constant learning rate (.00001) and momentum(.85) with 500 epochs
Figure 6.7: OLR face data base at a glance
Chapter 6. Fusing ECOC with other methods

is used to train the network which includes an input layer consisting of 64 units (for the image with size of 8x8); a hidden layer with 1500 units and the output layer with 40 units. The mentioned parameters have been chosen by a trial and error procedure to get the best result. The rate of correct classification in this method is 17%. Although this performance is much better free guess (5%) it can not be acceptable for many applications. For lower number of hidden nodes, network error will remain high, and we checked that higher number of nodes does not lead to better performance. It seems that the existence of high number of local minima in error surface has caused serious difficulties for the search procedure of back propagation.

2. A single multi-network consist of four sub-networks to solve multi-class problem as describe in chapter 6 (figure 6.1). The number of hidden nodes for sub-networks is 250, having the same learning parameters, the correct classification rate will increase to 31.5%.

3. An ECOC structure consists of local binary experts made by multi-networks (figure 6.1).

4. An ECOC structure in which binary experts are reinforced by AdaBoost with 10 iterations as described in previous example.

The results of the third and the fourth cases (ECOC and ECOC with boosted binary experts) for different size of images and learning parameters are repeated in table 6.6. These results clearly show that with relatively simple networks which are appropriately arranged to we can outperform much more complex single networks. Although we cannot say that there is not a single network that outperforms this composite system, finding such a network is a real problem.

1. The role of more experts

To have a better understanding, it is worth looking at the increasing performance when we add binary classifiers (figures 6.8 and 6.9). Theses figures indicate that:
Figure 6.8: The rise of performance by increasing the number of binary classifiers in ECOC and AdaBoost+ECOC a) for images with size of 8×8 b) for images with size of 12×12
Chapter 6. Fusing ECOC with other methods

Figure 6.9: The rise of performance by increasing the number of binary classifiers in ECOC and AdaBoost+ECOC a) for images with size of 24x24 b) for images with size of 32x32
Chapter 6. Fusing ECOC with other methods

Table 6.6: correct classification rate of ECOC on ORL face data base.

<table>
<thead>
<tr>
<th>size</th>
<th>h-nod</th>
<th>lr/mum</th>
<th>epochs</th>
<th>ECOC</th>
<th>ECOC+AdBst</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 x 8</td>
<td>20</td>
<td>.002/.65</td>
<td>130</td>
<td>79</td>
<td>82.5</td>
</tr>
<tr>
<td>12 x 12</td>
<td>20</td>
<td>.0007/.85</td>
<td>400</td>
<td>88.5</td>
<td>90.5</td>
</tr>
<tr>
<td>24 x 24</td>
<td>25</td>
<td>.0002/.85</td>
<td>400</td>
<td>94.5</td>
<td>94.5</td>
</tr>
<tr>
<td>32 x 32</td>
<td>25</td>
<td>.005/.5</td>
<td>30</td>
<td>94.5</td>
<td>97</td>
</tr>
</tbody>
</table>

(a) At first the performance is not much better than free guess, which indicates that too few experts cause increases in classification error. This is due to lack of class information in system as discussed in previous experiment. This is why we see a rapid increase in performance when the number of experts is between 1 and 20.

(b) Above 20 binary experts, improvement of performance levels off.

2. The role of experts’ precision

One main question here is:

Can more precise local classifiers guarantee better performance in final classification?

To discuss this question we should note that in ECOC, class information (learning the super-class labels) is not the only information the binary experts provide. In fact since ECOC uses a soft-level combining, the ability of experts to achieve better performance in sub-problems (indicative of good hard-level information) does not necessarily lead to better performance of composite system.

For example for lower resolution images, it is possible to get better performance in binary classifiers by using more complex sub-networks or to boost them. The precision of local experts for each case are shown in table 6.7. This table indicates that for the experiments on resolution of 24 x 24, base classifiers have better performance than the case of 32 x 32. But the final performance in latter case is much better (table 6.6).
This shows that the decomposing procedure helps to extract more information from the original problem in this case.

3. Error reduction of ECOC

Although ECOC with boosted binary experts exhibits better performance, it seems the difference between two cases (without and with boosting) decreases for more experts. It means that the procedure of decreasing the variance in boosting can be replaced by ECOC if we increase the number of columns in code matrix. So when we are using a random code and there is no problem in finding new columns can we ignore fusion of ECOC with Boosting?

Answering this question is not easy in general. But in some cases, when the number of classes is not too high we have a limitation in increasing the number of columns. The maximum number of independent columns (different, not complementary and not all "0" or all "1" columns) for a $k$ class problem is $2^k - 2$. Now for lower values of $k$, if the learning machine can not provide a precise base classifier sometimes the variance reduction of ECOC for this number of columns does not satisfy us. For example for an equi-distance code we showed that the final variance will be:

$$\sigma_{ecoc}^2 = \frac{b}{M^2} \sigma^2 (1 + (b - 1)\rho)$$

if we cannot provide a code matrix with lower $\frac{b}{M^2}$, an alternative way to reduce the variance is to lower $\sigma^2$. Using Boosting can be suggested in these circumstances.

6.5 Summary of chapter

In this chapter we have considered the application of ECOC technique to a more realistic problem than those reported in chapter 5.

Two main weak points of ECOC are:
Chapter 6. Fusing ECOC with other methods

<table>
<thead>
<tr>
<th>size</th>
<th>ECOC(mean)</th>
<th>ECOC(std)</th>
<th>ECOC+AdBst(mean)</th>
<th>ECOC+AdBst(std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 × 8</td>
<td>73.06</td>
<td>4.44</td>
<td>79.14</td>
<td>3.56</td>
</tr>
<tr>
<td>12 × 12</td>
<td>80.64</td>
<td>3.36</td>
<td>86.29</td>
<td>2.99</td>
</tr>
<tr>
<td>24 × 24</td>
<td>83.57</td>
<td>4.33</td>
<td>88.89</td>
<td>2.63</td>
</tr>
<tr>
<td>32 × 32</td>
<td>81.71</td>
<td>4.81</td>
<td>88.05</td>
<td>3.74</td>
</tr>
</tbody>
</table>

Table 6.7: correct classification rate of local experts of ECOC on ORL face data base.

1. ECOC can not reduce the complexity of pattern representation (feature set). This is a critical shortcoming when patterns did not pass a pre-processing stage and a multi-network structure has been suggested to build the binary experts. A local back propagation algorithm can be used to train this system. An example shows that this multi-net can outperform a much more complex single network under certain conditions (section 6.2.2 and tables 6.6 and 6.7).

2. This algorithm can not improve the performance of binary classifiers. To overcome this, fusion with ensemble methods particularly Boosting is suggested (table 6.7).

Providing some experiments (section 6.4) it has been shown that multi-net system can improve the performance of a direct classification system (multi-class classifier), for example from 2.5% to 17% on ORL data base. But much greater improvement will be achieved when it is fused with ECOC (about 80%). This technique is useful only if the feature set can be divided into sub-sets with common properties. AdaBoost can improve the performance of ECOC particularly when we use simpler binary classifiers and shorter codes (figures 6.8 and 6.9 and table 6.6).
Chapter 7

Summary and Conclusion

7.1 Introduction

Although in any research some results will be found, in many cases the way the researcher has carried out the work, can be more important than the results themselves. In fact in this way we can answer this essential question:

*What was the motivation of these works?*

The work described in this thesis started with the problem of finding the optimum set of weights for linearly combining the outputs of a set of neural networks. The main problem here is that when the networks have the same structure, with the same presentation of patterns and same training set, they will be correlated so the classical quadratic methods (like least squares) can not be used. Other search methods like exhaustive or genetic search have their own difficulties such as high calculation cost.

Although the idea of dynamic weighting (section 6.3.4) outperforms other methods in many situations, it was realised that in most cases, the way we have chosen to make independent networks is more important than the way we combine their results. [74, 23]. The Boosting method (particularly AdaBoost) provides a solution to both the problems of correlation reduction and weighting in linear combining. But still there were a few difficulties:
Chapter 7. Summary and Conclusion

1. In weighting the outputs, AdaBoost uses logarithmic weights which can be proved to converge to the optimum set for infinite number of iterations. In most reported works improvement can be achieved in this manner. But for neural networks which are powerful learning machines themselves, using high number of experts does not seem efficient. Modifying AdaBoost with dynamic weighting factor which results in lower number of required experts can solve such the problem in some cases as we have shown in [75].

2. For correlation reduction it is hardly possible to find a large set of independent networks by the weighted resampling of patterns. Although neural network classifiers are categorised as “unstable” classifiers, which means a small change in their parameters leads to large change in outputs, any attempt to increase the precision of networks like higher number of epochs, or hidden nodes (which could be necessary to satisfy conditions in Boosting), can lead to relatively stable networks. So small changes in some patterns may not change the behaviour of networks. This is why as it has been reported that traditional Boosting has been more successful than AdaBoost in some cases[75].

For neural networks, the problem is mainly due to the fact that most correlation reduction techniques are based on changing “the feature set”, “the training samples” or “the learning algorithm”. But because of lack of understanding of the inner workings of neural networks, the effect of these attempts on the output of network is not known well.

The ECOC concept has opened up a new possibility to make a large number of diverse networks with certain structure and training set. The idea is to decompose a complex problem into many sub-problems (mostly binary sub-problems). Even if we use the same feature set, training set and network structure, diverse networks can be made, each one is trained to solve one of several sub-problems. In other words, different labelling in ECOC can improve the performance by decomposing the problem into binary classification [77, 25, 76, 76] (sections 3 and 4). In fact the source of effectiveness of ECOC can be stated as:
1. It can provide the possibility of producing several independent classifier with the same feature set, training set and learning algorithm.

2. It can provide more accurate inferring by simplification of a complex problem. In other words binary sub-problems produced by ECOC are supposed to be simpler than the original multi-class problem.

3. It can provide more accurate assigning, by correcting some parts of error.

Although some points have been investigated experimentally by others, there has been a lack of theoretical explanation for them. For example in [16] using code with separated rows and column has been suggested, but the effect of row separation is not theoretically explained.

### 7.2 Contributions

The main weak point in previous works on ECOC has been the fact that the relationship between soft-level score needed for classification and distance measurements used in ECOC decision making, is not clear (section 4.1). This caused some shortcomings in:

1. Code selection; In which main previous works were dependent on information theoretic concepts ([16]) or using random codes ([37]).

2. Label selection for different classes; which was not discussed before.

3. Finding alternative frameworks in reconstruction; The main previous methods were least squares [44] and substitution [37]

4. Error analysis; which was limited to:

   (a) Experimental analysis (without theoretical explanation) based on bias variance decomposition for 1/0 loss function [43]

   (b) An error analysis of substitution [37] assuming that it is an asymptotically equivalent to ECOC.
In this thesis we have provided such a framework, leading to the following contributions:

1. New condition set and practical considerations for code generation (using equi-distance code for ECOC) (section 4.2).

2. A new algorithm (circular ECOC) which is a solution to the problem of label selection and shown to have superior performance to the original ECOC (section 5.2.1.7).

3. Introduction of the concepts of “reconstruction error” and “biased distance measurement” and their effects on final performance [77, 25, 76](section 4.2).

4. Bias-variance decomposition to explain the roots of effectiveness of distance-based ECOC particularly for the equi-distance code (section 4.2.3).

5. For least squares method, the equi-distance code for minimising the least squares error of both the binary experts output and the final output simultaneously(section 4.4).

6. Two new frameworks to combine the results of binary experts in ECOC

   (a) An analytic approach to find the probabilities from distance measurements (section 4.5)

   (b) Dempster-Shafer mechanism to combine the output of binary experts as evidences (section 4.7)

7. Fusing ECOC with other frameworks to overcome the shortcomings of ECOC. As an example we tried to solve the face recognition problem using relatively simple NNs without any preprocessing stage (section 6.4).

### 7.3 Summaries of chapters

This thesis has been organised as follows:
Chapter 1 introduce the application of NNs in classification task and the motivations to use homogeneous classifiers combining methods, particularly ECOC technique.

Chapter 2 includes a literature review of the most general reported works in the field of combining classifiers. In this chapter we have introduced the main quests of a combining task and popular techniques to satisfy them. The theoretical frameworks to explain how combining can be effective (evaluation frameworks) are presented as well.

Chapter 3 deals with decomposition of multi-class problems into two class sub-problems. The main ideas of error detection and correction as well as popular method to generate ECOC codes are introduced in this chapter. The main shortcomings of these methods is the fact that they came from information theory where hard-level information is the only valuable information. This leads to them to ignore the reconstruction of soft-level class scores.

Chapter 4 starts with a discussion on the previous works on ECOC classifiers and concludes that lack of understanding on the relationship between the output of components and class scores is the main shortcoming of previous works. Providing such a framework leads us to suggest:

1. For original ECOC, equi-distance code provides
   (a) zero reconstruction error
   (b) minimum bias
   (c) framework for variance analysis
2. For original ECOC, circular perturbing can improve the performance of any code
3. For least-squares, equi-distance code provides better performance
4. Class score can be found by solving matrix equations
5. Dempster-Shafer mechanism can be used to combine the output of local experts
Chapter 7. Summary and Conclusion

• **Chapter 5** provides experimental evidence for strategies and perditions made in chapter 4 on:
  1. effectiveness of equi-distance codes
  2. circular perturbing in ECOC
  3. variance analysis
  4. alternative combining frameworks

• **Chapter 6** deals with main shortcomings of ECOC in more realistic problems. The problem of input complexity is suggested to be solved by multinet local back propagation technique and AdaBoost for improving the performance of local binary classifiers in ECOC. Then a more realistic problem (face recognition) is chosen to be solved. Results indicate that simple subnetworks can be organised based on ECOC framework fused with feature decomposition and AdaBoost to solve the face recognition problem which normally needs much more complex network.

7.4 Conclusion and recommendations on future works

The main point of our investigation on ECOC is to find the relationship between scores of sub-problems produced by decomposition stage and the scores of original classes. This fact gives us the viewpoint of decomposition/reconstruction of these scores to this algorithm. In this way we have performed the following tasks not reported in previous works:

1. We introduce the concept of reconstruction error (section 4.2) which is the source of some parts of error which is not due to imperfect learning. To remove this error in ECOC we have to use a code with equal Hamming distance between all rows (labels).

2. Having a certain code (non equi-distant) we find that with a circular perturbing, not only the effect of non optimal code is reduced and label selec-
Chapter 7. Summary and Conclusion

tion dilemma is solved, but also the opportunity of repeating the training phase can provide better performance (sections 5.2.1.7 and 4.2).

3. We have shown that for equi-distance code, the effect of bias in final assigning will be minimised (section 4.2).

4. We have shown how the variance of original scores is related to:

   (a) Variance of local scores
   (b) Hamming distance between code words (labels)
   (c) Correlation between local errors

In other words, we have provide a theoretical explanation of "error correction" in classification task and the effect of row and column separation in ECOC (section 4.2.3).

5. We have investigated the effect of code selection in least-squares method and found that equi-distance code provides the minimum squares error not only in local (sub-problem) scores but also in original (class) scores (section 4.4).

6. We have provided new combining frameworks (analytic solution (section 4.5), linear estimation (section 4.6) and Dempster-Shafer mechanism (section 4.7)) to reconstruct original scores from the results of sub-problems.

7. We have pointed to ECOC shortcomings and suggest methods to overcome them in certain circumstances (section 6).

Furthermore our works have opened up new research areas:

1. Error analysis

   Although we have found a formula to predict the final variance (section 4.2.3) for equi-distance code, work can be carried out in following directions:

   (a) Investigating the effect of column selection on correlation of local errors
(b) Using bias-variance decomposition for other combining frameworks of ECOC

(c) Applying different definition of bias and variance in classification task ("1" /"0" loss functions) in ECOC

2. As noted before one weak point of ECOC is the fact that for problems with high number of classes the decision boundary of binary sub-problems can still be complex. Methods based on pairwise coupling do not suffer this weakness. This is because of the fact that in any sub-problem just two classes will be considered. The main practical difficulty of this method in training is the patterns which do not belong to any of classes. It seems that Dempster-shafer structure can be used in these problems.

3. In ECOC the way we decompose the original problem (code matrix) is completely independent of the problem. Even in equi-distance codes, we do not consider the existence of different level of error in different pairs of classes. It seems that if we can choose the code words based on the classes of the problem at hand, it can improve the performance.
Bibliography


Bibliography


Bibliography


