CHAPTER IV

GENERALIZED CODE CONCATENATION FOR MULTIPLE CLASSIFIER SYSTEMS

The ECOC technique for solving multi-class pattern recognition problems can be broken down into two distinct stages – encoding and decoding. Given a pattern vector of unknown class, the encoding stage consists in constructing a corresponding output code vector by applying to it each of the base classifiers in the ensemble. The decoding stage consists in making a classification decision based on the value of the output code. The main focus of this chapter is on the former stage. In particular, we propose a series of concatenated output codes for MCS based on MDC derived from the previous chapter. In Section 4.2, we start to lay some basic knowledge in channel coding. The first part of the section introduces the threshold decoding technique for T-repetition code, where the decoding of T-repetition output code is equivalent to Adaboost [8]. Based on this knowledge, in Section 4.3 we propose to cascade MDC with Adaboost for MCS, leading to one of the two-stage T-repetition output code schemes. Thus, this classification scheme can be easily fitted into classical code concatenation scheme. Moreover, it can be considered as the MCS method where we try to both manipulate the input features and resample training examples according to Adaboost algorithm (see section 2.2 in Chapter 2). We also propose to cascade MDC with ECOC for MCS, whereas it should be fitted with classical code concatenation scheme. Finally, we explore the possibility to use the MDC and Bagging methods with ECOC in order to generalize them to the generalized code concatenation scheme for MCS.

4.1 Introduction

Actually, the common approach used in channel coding is to introduce redundancy to information before sending on the channel. For example, giving S information sequence, a linear block code $X^S \rightarrow X^R$ is applied to information, and the R resulting codeword are sent on the channel. The information system works by producing a linear dependence between the transmitted bit symbols. When the distorted symbol is received, it is consistent with exactly one valid element of X^R, so the information symbols is decoded and then known.

On the other hands, one can build a coverage set of classifiers by repeatedly partitioning the complete set of classes into pairs of super-classes and, for each pair, training a base classifier to distinguish between these two super-classes. When the partitions of super-classes are chosen carefully, redundancy is thus introduced to class information in a more systematic

way. To classify an unknown pattern we present it to all of the trained classifiers. With the classifier outputs we create a string which corresponds to the unknown pattern and the designed output codes. In making an overall classification decision, the decoder assigns the string to the class by means of minimum distance decision rule, or probability estimation, or likelihood decoding rule [77]. This is thus a match between the capabilities of decoding redundant information in coding theory and the needs of partitioned hypotheses decoding in pattern recognition. One of the possibilities in adding redundancy to channel sequence is by repeating exactly the same information and sending it over one single channel sequentially or over multiple channels (in this case, all coded sequence are maximally correlated).

In analogy to channel coding in information theory, the error rate of MCS can be controlled by adding redundancy to the original patterns before final decision. Underlying the MCS framework, a teacher could try to repeat target concept M times, resembling the transmission of same information over M channels. This scheme is sometime called T-repetition code. Note that many MCS are based on this T-repetition code model. For example, decision of boosting based MCS [8], called *Adaboost*, could be interpreted as an instance of threshold decoding for a T-repetition code under the assumption of a binary discrete memoryless channel constructed by the ensemble of binary weak classifiers.

In the case that the redundancy information with maximum correlation are presented to multiple classifiers, the decisions among classifier members become highly dependent and inefficient. In this case, we thus face a major problem in optimal combining the classifiers, called *harmful collinearity* [78]. Anyway, we could alternatively try to minimize the correlation between all information channels (classifiers) by splitting the information into distinct parts by simple spatial sampling. In this case, all classifiers are independent to each other, but each of them makes inaccurate decision from partial information presented to them.

Demanding for solving complex classification problems has directed toward the optimal design (trade-off) between the above two cases. One way of achieving effective results is through *sharing* resources such as information and components. There are at least two other methods for adding redundancy (sharing resources) into classification systems. The first method is known as ECOC, which exploits the conventional concept of channel coding, called *forward error correcting code (FEC)*. The ECOC based MCS is inspired by error correcting code transmission technique from communication theory. In the second method, redundancy is added into classification system in a more general way. This method is called *multiple description coding model* [44,53], which is based on the concept of frame expansion (linear transform). Since the linear block code is a linear transform. The difference between two methods is that the transform and coding blocks are interchangeable. Specifically, the expansion from S to R dimensions is done in the original, continuous domain of the data.

There are three broad approaches to MCS. One is based on combining strong learners. The second approaches is based on combining weak learners. The last approach is based on modular nets and hybrids. The effectiveness of the combining strong learners approach

is aimed at variance reduction, while the combining weak learned approach is aimed at reducing both bias and variance. The ensemble error of the combining strong learners approach is come from the average error minus by the average of mismatch. This way, coverage and prediction optimizations become the key ingredient for the success of the combining strong learners approach. More importantly, the main design criterion of MCS is to optimize the performance of the system subject to classifiers' characteristics (strong or weak), and restrictions on the complexity. Our proposed method learns an ensemble of classifiers that are biased to have high precision (as opposed to, for example, boosting, where the ensemble members are biased to ignore portions of the instance space). Giving that the aim of coverage optimization method presented in Chapter 3 is satisfied. We further modify the idea of combining strong learners to concatenated error correcting output codes.

In this chapter, we are interested in the equivalency between decoding redundancy information sent over multiple channels and combining multiple hypotheses classified by multiple classifiers. Based on such an equivalency, we describe two extensions to the ECOC to select super-classes partitions suitable for classification problems. The first method is based on the classical code concatenation, whereas the second method is based on a more generalized concatenated codes.

4.2 Channel Coding

In this section, we first describe likelihood decoding of T-repetition code and some notations needed in coding theory. The likelihood decoding of T-repetition code described here is somewhat akin to Adaboost [8]. Moreover, this decoding technique currently becomes one of the favorite decoding method used in ECOC [77]. Then, we detail the concept of classical and generalized code concatenations based on the material taken from [49].

4.2.1 Coding Theory

Often the code parameters (n, k, d_m) are used to distinguish between different block codes, where d_m is the minimum distance between all code words in the code. In a special case, transmitting single information bit n times is the $(n, 1, n)$ block code.

Definition 4.1 (Dual code) *If $\mathcal{C}$ is a (n, k) code we define the $(n-k, n,)$ dual code by $\mathcal{C}^{\perp}$*

$$\mathcal{C}^{\perp} = \{x \in \mathcal{F}_2^n; |; cx^T =: 0, ; c \in \mathcal{C}\}, \tag{4.1}$$

where $\mathcal{F}_2^n$ denotes the binary vector space of length n.

Definition 4.2 (Repetition code) *The length n repetition code is the set of two code words consisting of the all-zero code word* c_0= 00...0 and c_1= 11...1.

Definition 4.3 ($n-1$ **parity check code**) *$n-1$ parity check code (parity check matrix of repetition code) of length n is the set of all code words satisfying $cH^T = 0$, where the $n-1 \times n$ parity check matrix is given by*

$$H = \begin{pmatrix} 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \\ 1 & & & & 1 \end{pmatrix}. \tag{4.2}$$

Definition 4.4 (Generator matrices of repetition codes) *A generator matrix that maps the single information symbols to any repetition code components of length n is given by*

$$G = [I; |; P], \tag{4.3}$$

where $I = 1$, and P is a $n-1$ vector with all of its elements are equal to 1. In other words, $G = [1; |; 11 \dots 1]$, and H is its dual code.

Definition 4.5 (Majority logic decision) *Consider a set of parity check vectors $\mathcal{M}_J$ of a linear binary code $\mathcal{C}$, where $\mathcal{M}_J$ spans $C^\perp$ and position j is equal to one for every vector. The majority logic decoding of position $j \in [1, n]$ is defined by*

$$\widehat{c}_j = \begin{cases} r_j \oplus 1 & \text{if } \sum_{\mathbf{b} \in \mathcal{M}_J} \langle \mathbf{r}, \mathbf{b} \rangle \ mod\ 2 \ > \lfloor \frac{J}{2} \rfloor \\ r_j & otherwise, \end{cases} \tag{4.4}$$

where $\oplus$ is addition modulo 2, $\langle \cdot\, :, \cdot \rangle$ is scalar product of two binary vectors, and a parity check vector $\mathbf{b} \in \mathcal{C}^\perp$. In other words, $\mathbf{b}$ also spans the $n-1$ parity check code.

In fact, the above decoding rule can be formulated in a different, but equivalent manner. We may write

$$\widehat{\mathbf{c}} = \begin{cases} 1 & \text{if } \sum_{j=1}^{J} r_j > \lfloor \frac{J}{2} \rfloor \\ 0 & otherwise. \end{cases} \tag{4.5}$$

Definition 4.6 (Error probability) *The block error probability P_e^b gives the probability that a transmitted code word does not correspond to the decoded code word, and the bit error probability p_e specifies the probability that a transmitted information bit is incorrect.*

If more than e errors occur during the transmission of a binary code word consisting of n bits, then

$$\begin{aligned} P_e^b &= \sum_{j=e+1}^{n} \binom{n}{j} p_e^j (1-p_e)^{n-j} \quad e \text{ odd} \\ &= \frac{1}{2} \binom{e}{\frac{e}{2}} p_e^{\frac{e}{2}} (1-p_e)^{\frac{e}{2}} \\ &\quad \sum_{j=e+1}^{n} \binom{n}{j} p_e^j (1-p_e)^{n-j} \quad e \text{ even.} \end{aligned} \tag{4.6}$$

Example 4.1 *The simple error-correcting code is the "triplication" code [42]. Here, the transmitter sends the information bit as the triplet* (i, i, i)*, and the corresponding received triplet of binary numbers with probable error is denoted as* (r_1, r_2, r_3)*. Following Definition 4.3, we can obtain the parity check matrix as*

$$H = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \tag{4.7}$$

and $G = (1; 1; 1)$.

We can obtain three parity check vectors

$$\begin{array}{rcl} \mathbf{b_1} = & h_1 = & (1 \quad 1 \quad 0), \\ \mathbf{b_2} = & h_2 = & (1 \quad 0 \quad 1), \end{array} \tag{4.8}$$

and theirs corresponding syndromes $S_1; \text{and}; S_2$ *computed as the scalar products of the received and parity check vectors (specifically, received vector is derived from the scalar product of the information bit and noise bit* e_j*):*

$$\begin{array}{rcl} S_1 & = & \langle \mathbf{r} :, \mathbf{b_1} \rangle \\ & = & (r_1 \oplus 1) \oplus (r_2 \oplus 1) \oplus (r_3 \oplus 0) \\ & = & (i \oplus e_1) \oplus (i \oplus e_2), \\ S_2 & = & \langle \mathbf{r} :, \mathbf{b_2} \rangle \\ & = & (i \oplus e_1) \oplus (i \oplus e_3). \end{array} \tag{4.9}$$

Following Definition 4.4, if the received bit r_1 is corrupted by noise bit e_1, we see that the noise bit e_1 can be correct by the rule

$$\widehat{c_1} = \begin{cases} r_1 \oplus 1 & \text{if}; S_1 + S_2 > 1.5 \\ r_1 & otherwise. \end{cases} \tag{4.10}$$

Similarly, the next j received bit can be correctly detected by exploiting $\widehat{c_l}, ; l = 1, \ldots, j-1$ (see [79] in details).

Furthermore, if the probability of error for each bit is p_e, then the probability of error of the triplication code (3rc) is

$$\begin{array}{rcl} P_e^{3rc} & = & \sum_{j=2}^{3} \begin{pmatrix} n \\ j \end{pmatrix} p_e^j (1 - p_e)^{n-j}, \\ & = & 3p_e^2(1 - p_e) + p_e^3, \end{array} \tag{4.11}$$

when we use the bound minimum distance decoding method.

Next, we can get an elementary proof that the block error probability of triplication code P_e^{3rc} is always less than bit error probability p_e as long as $p_e \leq \frac{1}{2}$. Since $3p_e^2(1-p_e)+p_e^3 \leq p_e$ is equivalent to $3p_e(1 - p_e) + p_e^2 \leq 1$ or $2p_e^2 + (-3)p_e + 1 \geq 0$, which is an upward-going parabola, we need only check where its roots are. Using the quadratic formula, we find that they are located at $\frac{1}{2}$ and 1. Consequently in the range $0 \leq p_e \leq \frac{1}{2}$ the error rate produced by coding is smaller.

Definition 4.7 (Threshold decoder) *The threshold decoding of position $j \in [1, n]$ is defined by*

$$\widehat{c}_j = \begin{cases} r_j & \text{if } \sum_{l=1}^{J} w_l s_l \leq \frac{1}{2} \sum_{l=1}^{J} w_l \\ r_j \oplus 1 & otherwise, \end{cases} \tag{4.12}$$

where

$$s_j = L^H(s(\mathbf{b_j})) = \begin{cases} 0 & L(s(\mathbf{b_j})) \geq \mathbf{0} \\ 1 & otherwise, \end{cases} \tag{4.13}$$

$w_l = |L(s(\mathbf{b_j}))|$, and $L(s(\mathbf{b_j}))$ denotes the *log likelihood ratio* for the j^th information bit. In fact, $L(s(\mathbf{b_j}))$ is called the *soft-output* of the decoder and can be interpreted as reliably information of the decoded information bit. For the sake of brevity, we refer the reader to Reference [49] for the derivation of $L(s(\mathbf{b_j}))$, which we can introduce the term here as

$$\begin{aligned} L(s(\mathbf{b})) &= \ln \frac{P_{even}}{P_{odd}} \\ &= \ln \frac{1+\prod_{l \in supp(\mathbf{b})} \tanh(\frac{1}{2} L(c_l, r_l))}{1-\prod_{l \in supp(\mathbf{b})} \tanh(\frac{1}{2} L(c_l, r_l))}. \end{aligned} \tag{4.14}$$

Note that threshold decoding technique was first introduced by Massey [42] as the extension of the majority decoding technique. It is also called *weighted majority decoding* and *a posteriori probability (APP) decoding*. In fact, equation (4.14) is likely to be equal to the combining weight of the Adaboost algorithm.

4.2.2 Generalized Code Concatenation

The traditional codes cannot be used for several communication channels, e.g., wireless channel. The major difficulty of traditional codes is that, in an effort to approach the theoretical limit for Shannon's channel capacity, there is need to increase the codeword length of a linear block code, which, in turn, causes the computational complexity of a decoder to increase exponentially. Random codes are known to achieve Shannon limit performance as k gets large, but at the price of a prohibitively complex decoding algorithm. Ultimately, a point is reached where complexity of decoder is so high that it becomes physically unrealizable. The way to combat the problem is to use concatenated coding, where two (or more) constituent codes are used in serial or in parallel. To obtain high coding gains with moderate decoding complexity, concatenation of codes has thus proved to be an attractive scheme.

The idea of code concatenation was first investigated in 1966 by Forney. As shown in Figure 4.1, he defined an *inner code* together with the channel as a *superchannel*. A second code, called the *outer code*, encoded the data to be sent over the superchannel. One can view the construction of code concatenation according to Forney as *classical* concatenation. The perspective of the classical concatenation of the $\mathcal{B}$ inner and $\mathcal{A}$ outer codes can be easily modified to the generalized code concatenation paradigm, as shown in Figure 4.2. This perspective was proposed by Blokh and Zyablov (see [49, page 290] and the References therein).

To gain the knowledge of generalized concatenation, we are going to study the concept through the following examples. Here, the code parameter $(q; n, k, d_m)$ are used to distinguish between concatenated block codes, where q indicates the code alphabet, n is the number of codeword, k is the number of information bits, and d_m is the minimum distance between all code words in the code. This is with the purpose to differentiate between binary and non-binary codes.

Example 4.2 *(**Classical Concatenation**) Let us consider a single parity check code $\mathcal{B}(2; 4, 3, 2)$ with length 4 as the inner code. The outer code is the repetition code $\mathcal{A}(2^3; 8, 1, 8)$. Thus, the codeword of the concatenated code C_c consists of an 8 where the encoded inner codeword are listed in the rows. The length of the codeword is thus become $n = 4 \times 8 = 32$ for $k = 3$ information bits. The minimum distance is $d = d_a d_b = 8 \times 2 = 16$. The classical code concatenation technique is shown in Figure 4.3. Therefore the concatenated code has parameters*

$$C_c(2; 32; 3; 16). \tag{4.15}$$

In the following example, we construct a generalized concatenated code based on the same code as above.

Example 4.3 *(**Generalized Concatenation**) Let us also consider a single parity check code $\mathcal{B}(2; 4, 3, 2)$ as the inner code, which we term it the first inner code $\mathcal{B}^{(1)}$. We partition this code into four subcodes by carrying out in a way that the minimum distance within a subcode is maximized. Each subcode is partitioned into two codeword of the original code set. All 8 optimized codeword of $\mathcal{B}^{(1)}(2; 4, 3, 2)$ are listed below:*

$$\begin{aligned} &(0000), \quad (0011), \quad (0101), \quad (0110), \\ &(1001), \quad (1010), \quad (1100), \quad (1111), \end{aligned} \tag{4.16}$$

We can rearrange these possible codeword to subcode with best minimum distance as

$$\begin{aligned} \mathcal{B}^{(2)}(2; 4, 1, 4) &= \{\ (0000), \quad (1111)\ \}, \\ \mathcal{B}^{(2)}(2; 4, 1, 4) &= \{\ (0011), \quad (1100)\ \}, \\ \mathcal{B}^{(2)}(2; 4, 1, 4) &= \{\ (0101), \quad (1010)\ \}, \\ \mathcal{B}^{(2)}(2; 4, 1, 4) &= \{\ (0110), \quad (1001)\ \}. \end{aligned} \tag{4.17}$$

Based on this partitioning, we can define how to identify the four subcodes and their codeword within each subcode by using two outer codes:

$$a^{(1)} \in GF(2)^2 \ \text{ and } \ a^{(2)} \in GF(2). \tag{4.18}$$

For example, when $a^{(1)} = (10)$ and $a^{(2)} = 1$, the selected codeword is $\mathcal{B}^{(3)}_{10,1} = (1010)$. In Figure 4.4, we graphically demonstrate the partitioning of the code $\mathcal{B}^{(1)}(2; 4, 3, 2)$. The basic idea of this method is to protect the enumeration of the partitioning using the outer repetition code $\mathcal{A}^{(1)}(2^2; 8, 1, 8)$ and an extended Hamming code. This generalized code concatenation technique is shown in Figure 4.5. Finally, it is not difficult to see that the parameter of the generalized code is

$$C_{gc}(2; 32; 6; 16). \tag{4.19}$$

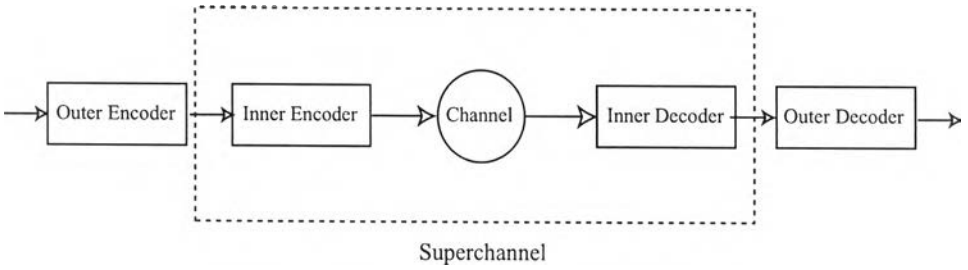

Figure 4.1 Code concatenation according to Forney.

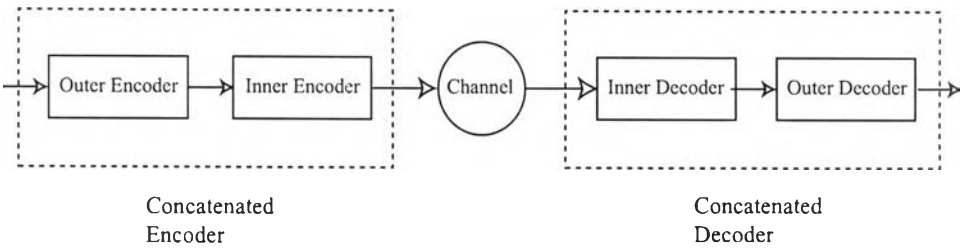

Figure 4.2 Code concatenation according to Blokh and Zyablov.

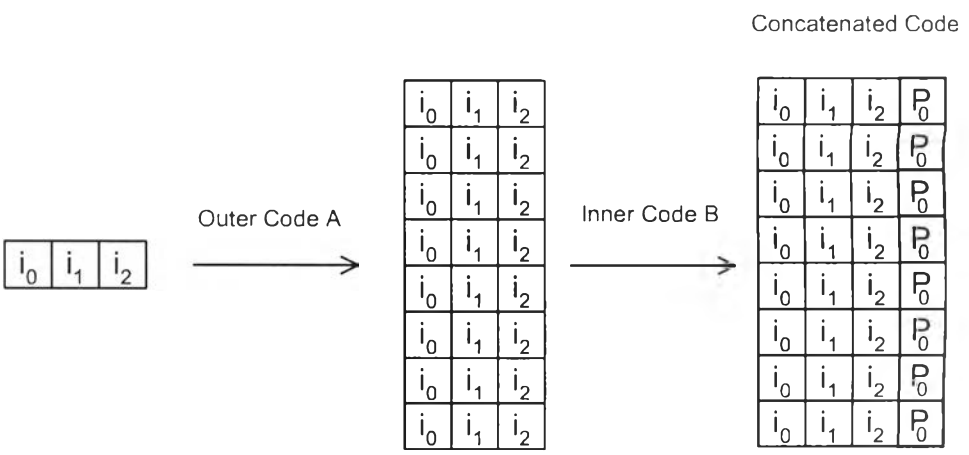

Figure 4.3 Example of classical concatenation (systematic).

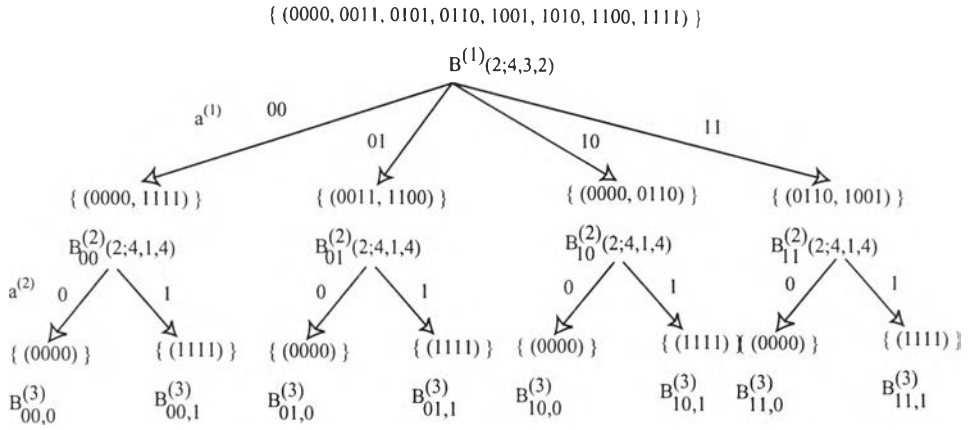

Figure 4.4 Example of codeword scheme for generalized code concatenation.

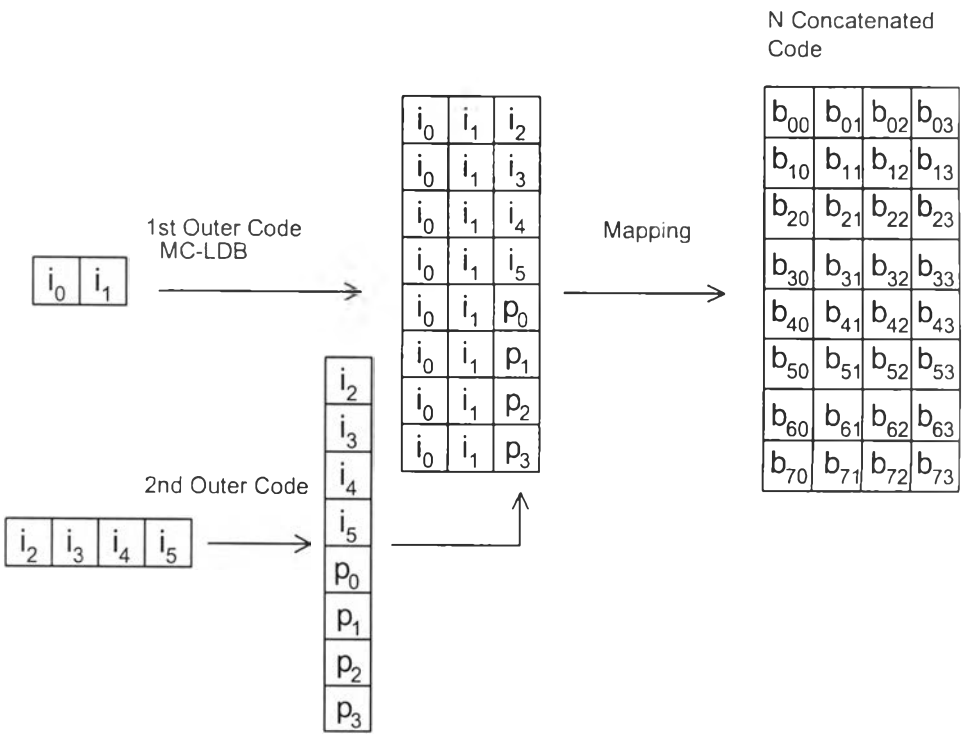

Figure 4.5 Example of generalized concatenated encoding.

4.3 Applications to Multiple Classifier Systems

In this section, we first describe two classical code concatenation schemes for M-CS. The implementations of both classical code concatenation schemes for MCS are just straightforward. For example, we can simply use two cascaded T-repetition codes for implementing the first classical code concatenation scheme. Both T-repetition code can be derived from different coverage MCS construction methods, e.g., outer code from MDC or Bagging, and inner code from Adaboost. This way, two cascaded MCS can be easily cast into the classical code concatenation framework. Alternatively, we can also cascade T-repetition code with FEC to implement the second scheme according to the classical code concatenation framework. In fact, this technique can be viewed as the method that hybridly manipulates the input features and resamples training examples according to Adaboost, and also viewed as the *input-output coding method for MCS*.

In generalized code concatenation MCS scheme, we derive two of the outer codeword based on either MDC or Bagging method. Then, each outer codeword is mapped to the inner codeword by using SVM-ECOC.

4.3.1 Classical Concatenated Input-Output Code with Multiple Description Coding and Adaboost

As previously discussed, we can simply use two cascaded T-repetition codes to construct a multiple classifier system. One of the advantages of using the proposed cascaded MCS scheme is that it can overcome one of the short comings in using just only one single MCS method, especially Adaboost. As argued in [27], noise handling is a crucial issue of Adaboost. From experimental results presented in Chapter 3, we have also seen that Adaboost with LDB can help improve the recognition accuracy. In particular, the main purpose of local discriminant bases and its variants [80] are usually aimed at denoising signal. They also believe that denoising, compression, and pattern recognition are very closed fields. This way, LDB can be used with great success to denoise signal before using Adaboost. It is also reported that more resistance to overtraining is obtained when we use classifier with LDB. However, *Coiflets* seem to be less resistant to overtraining than other wavelet filters, as they are adapted too well to training data [15]. In order to further reduce the overfitting and increase the efficiency of Adaboost, we thus use Adaboost with multiple description. This technique is shown in Figure 4.6.

4.3.2 Classical Concatenated Input-Output Code with Multiple Description Coding and Block Codes

In the second scheme, we can use two concatenated codes, where the outer code is from T-repetition code and inner code is from parity check bits or FEC. On the implementation of

this scheme for MCS, the implementation can be done by using Bagging with SVM-ECOC. In this case, bagged data sets can be considered as the encoded T-repetition outer codeword and SVM-ECOC is then used for implementing the inner codeword. we can also further use MDC as the T-repetition outer code and SVM-ECOC as the inner code. As discussed in previous sections, this technique can exploit the advantages of MDC over the conventional Bagging method. This technique is viewed as an extension method for the MDC and Adaboost. The technique is thus illustrated in Figure 4.7 .

4.3.3 Generalized Concatenated Input-Output Code

There are problems in using generalized concatenation for MCS. One of the difficulties in implementing the generalized code concatenation scheme for MCS is that the power of a code to correct errors is directly related to the row separations. However, the error correcting codes only succeed in MCS if the errors made in the individual bit positions are relatively uncorrelated. This way, large row separation preferred for generalized coding matrix in coding theory will not be the only separation we prefer in MCS. This is from the fact that there are many simultaneously correlated errors in many bit positions.

On the implementation of generalized concatenation for MCS, the length of the output codeword for each subcode will be dramatically reduced, since there are less super-classes partitions to be selected for each subcodes. For example, on the implementation of classical concatenation of 7-class classification problem, we can use (15,7) BCH code matrix (inner code) to encoded the repetition code (outer code). This way, if it is assumed that 4-repetition code is used in the classification system, the length of the codeword is thus become $n_c = 15 \times 4 = 60$.

On the other hand, the encoding of 3 information bits is carried out as follows. One information bit i_0 are encoded by $\mathcal{A}^{(1)}$, giving the first four super-class partitions $\mathbf{a^{(1)}} = (c_0^{(1)}, c_1^{(1)}, c_2^{(1)}, c_3^{(1)})$ obtained from the 7 original classes. The remaining two information bits (i_1, i_2) are encoded by $\mathcal{A}^{(2)}$, giving the second four codeword super-class partitions $\mathbf{a^{(2)}} = (c_0^{(2)}, c_1^{(2)}, c_2^{(2)}, c_3^{(2)})$. In other words, information bit i_0 is mapped by 8-repetition outer code $\mathcal{A}^{(1)}$, and information bits (i_1, i_2) are mapped to *encoded* bits $(i_1, i_2, i_1, i_2, i_1, i_2, i_1, i_2)$ by the outer code $\mathcal{A}^{(2)}$. Evidently, the maximum number of super-class partitions is 4. In this case, we can use 7 bit exhaustive code matrix for encoding $\mathbf{a^{(2)}}$. This way, if it is assumed that 8-repetition code is used in the classification system, the length of the codeword is thus become $n_{gc} = 7 \times 8 = 56$. This way, we can see that the numbers of base classifiers of both proposed scheme become comparable. It can be seen that in MCS we deal with the super-class partitions, while in coding theory we deal with the codeword partitions. This technique is shown in Figure 4.8 .

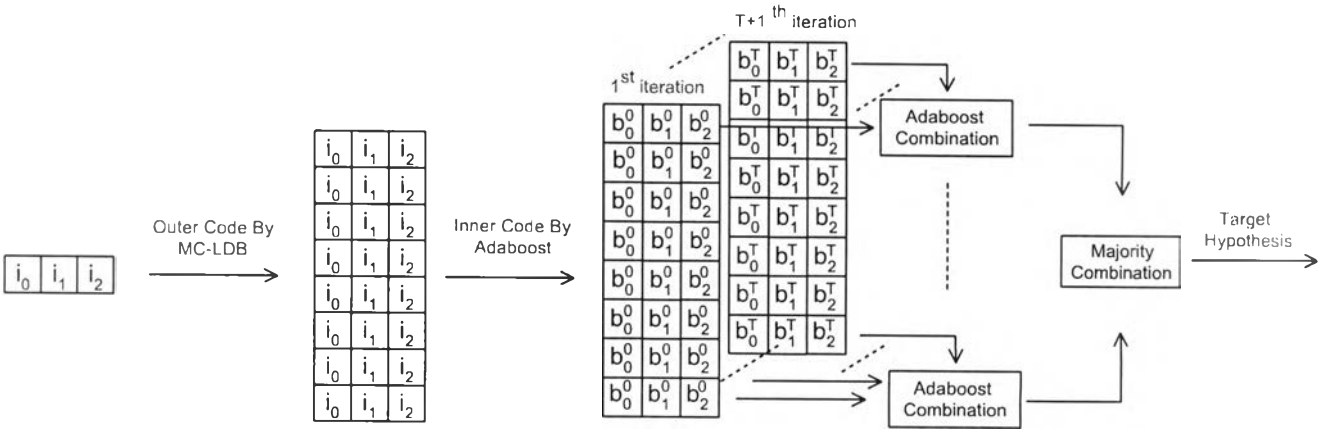

Figure 4.6: Example of classical code concatenation based multiple classifier systems using MDC and Adaboost.

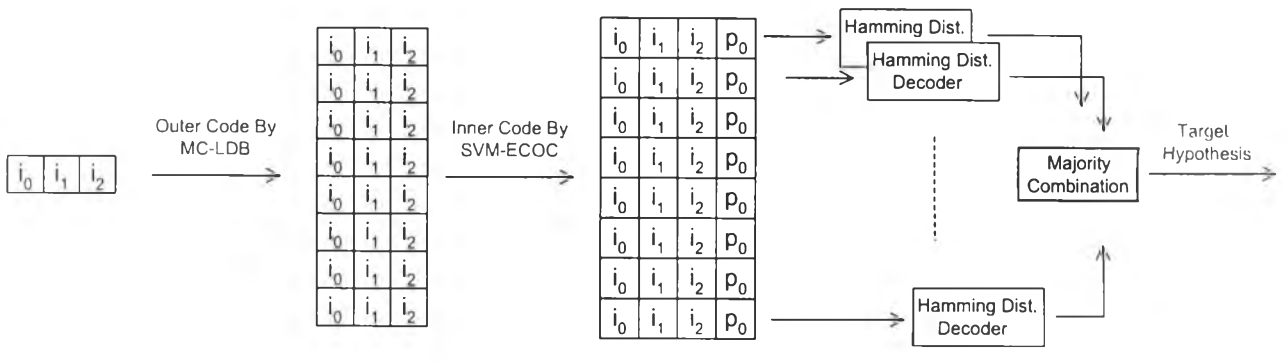

Figure 4.7: Example of classical code concatenation based multiple classifier systems using MDC and ECOC.

4.4 Experimental Results

For classical code concatenation schemes, we conduct the experiments on the same SAR ATR problem discussed in Chapter 3 of this dissertation. In the experiments, we also used the first order Coiflet filters for the LDB algorithm. Four different image sizes from 32x32 to 80x80 had also been used in our experiments. We also used the same number of hidden nodes for Adaboost algorithm, and same 3-class exhaustive output code for ECOC as well.

For generalized code concatenation schemes, we first generalize the classical code concatenation based on MDC and Adaboost using the generalized concatenated BCH code. For the second method of this approach, we exploited the bagged ensemble of SVM-ECOC with the generalized concatenated BCH code. Both methods are applied to classical and generalized code concatenation schemes of the 7-class Satimage problem obtained from UCI repository, respectively. We compared a series of our proposed methods with Boosting, random subspace method, random tree methods over the Satimage data set examined for classification performance in [81].

4.4.1 Classical Concatenation Codes

Here, we first present the experimental results regarding to the classical code concatenation based on MDC and Adaboost. As presented in Table 4.1, our proposed method gives better overall performance than both the single output code schemes: T-repetition-like code scheme (Adaboost) and ECOC. As shown from Figures 4.9 to 4.12, we obtained the percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost, whereas each recognition accuracy is plotted as a function of number of weak classifiers and descriptions used in Adaboost at window size 32 to 80.

Note that the performance improves continuously as the number of weak classifiers increases with large number of descriptions. As shown in Figure 4.13, we obtained the percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost. Each recognition accuracy is plotted as a function of image size and number of descriptions used. At window size 64, the performance of the scheme was better than when it was evaluated at window size 80. This may be from the fact that too much noise was included at the large window size.

Here is the second set of experimental results regarding to the classical code concatenation based on MDC and ECOC. As presented in Table 4.2, our proposed method improve the overall performance over the single output code scheme of ECOC. As shown in Figure 4.14, we obtained the percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost. Each recognition accuracy is plotted as a function of image size and number of descriptions used. It is also happened that the performance of the scheme was not so good at window size 80 as well.

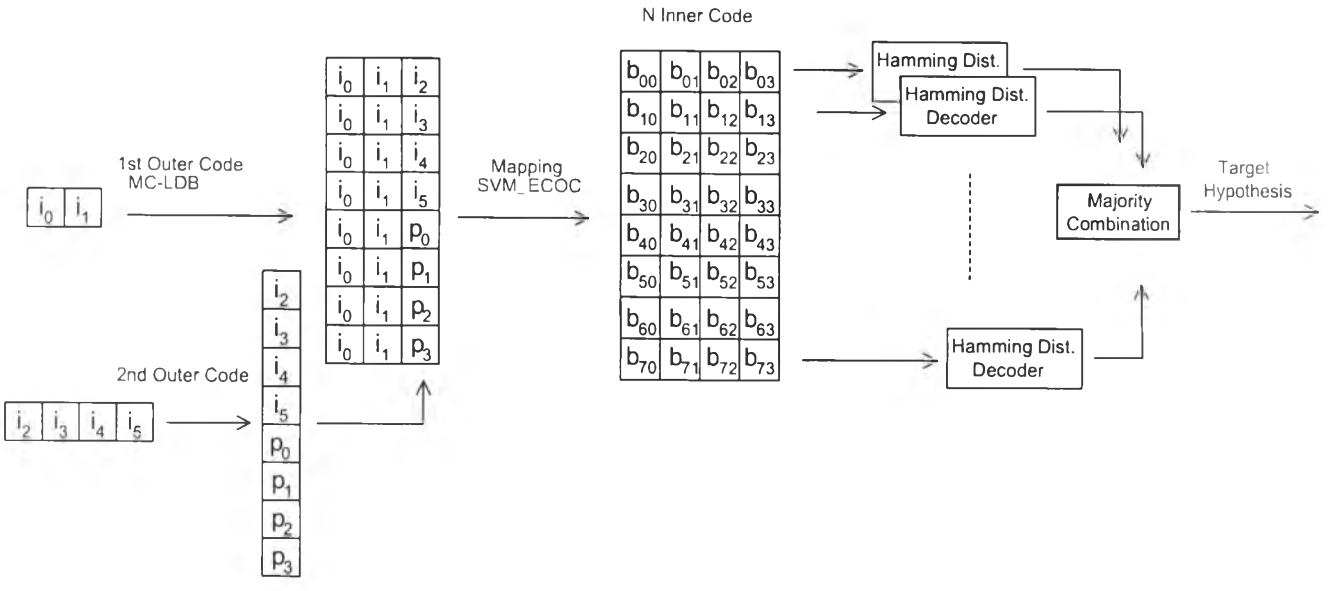

Figure 4.8 Generalized concatenation code scheme for multiple classifier systems.

Table 4.1: The performance of classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Comparison of difference methods is in overall percentage of images correctly recognized as a function of image size.

Methods / Image Size	32x32	48x48	64x64	80x80
MDBC+NN	75.88	N/A	N/A	N/A
CGSM	N/A	N/A	N/A	**98.53**
SVM-ECOC (original)	84.46	90.16	91.76	92.70
SVM-ECOC (LDB)	85.42	90.81	92.51	92.14
Adaboost (original)	88.24	93.35	93.48	93.68
Adaboost (LDB)	89.66	93.16	94.26	93.97
MDC-Adaboost (LDB)	**91.27**	**95.64**	**96.36**	96.25

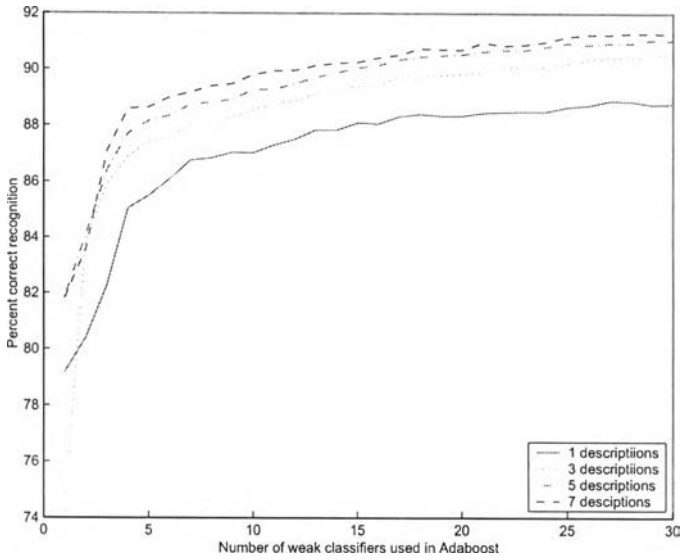

Figure 4.9: Percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of number of weak classifiers and descriptions used in Adaboost at window size 32.

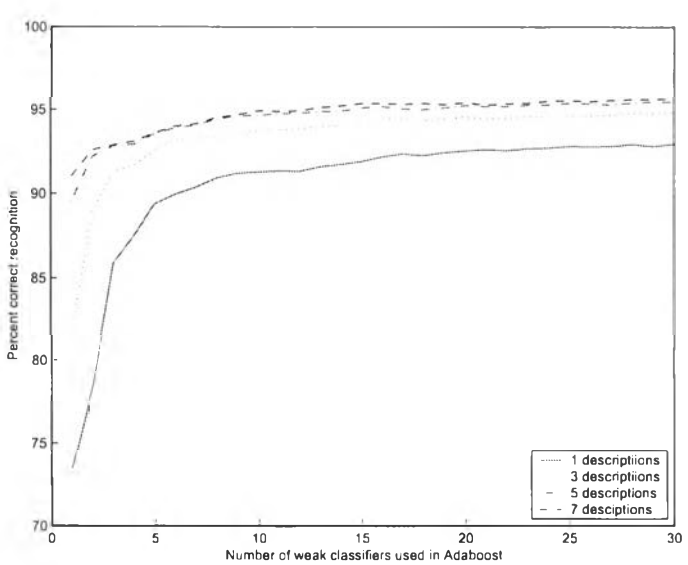

Figure 4.10: Percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of number of weak classifiers and descriptions used in Adaboost at window size 48.

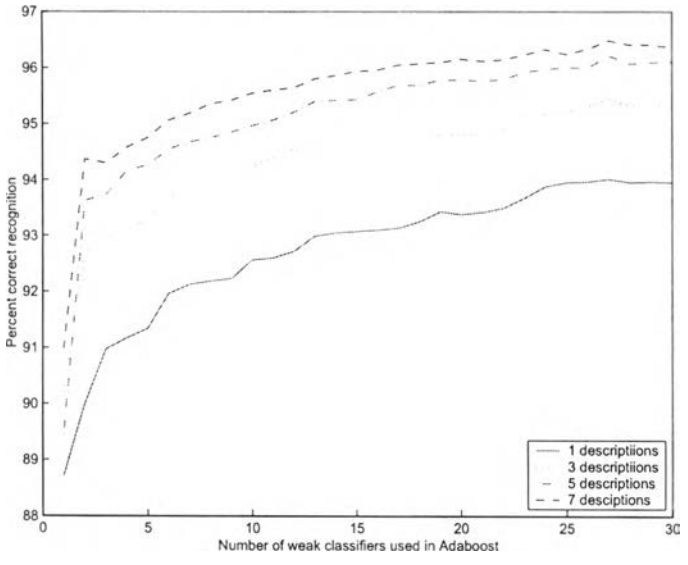

Figure 4.11: Percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of number of weak classifiers and descriptions used in Adaboost at window size 64.

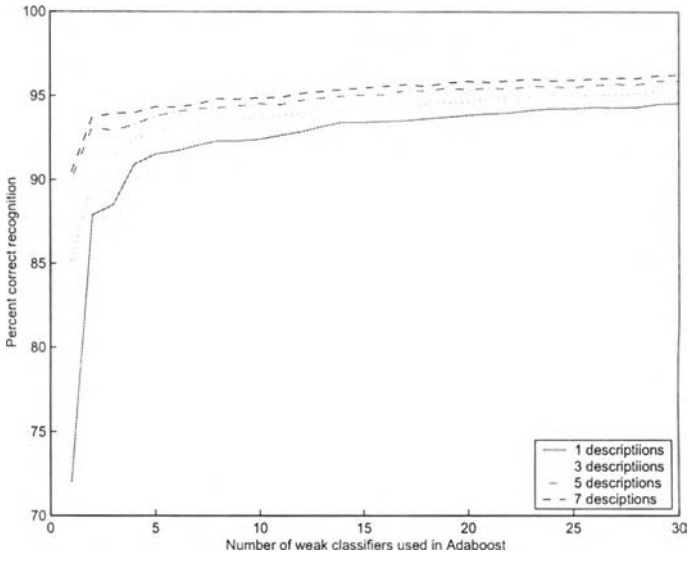

Figure 4.12: Percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of number of weak classifiers and descriptions used in Adaboost at window size 80.

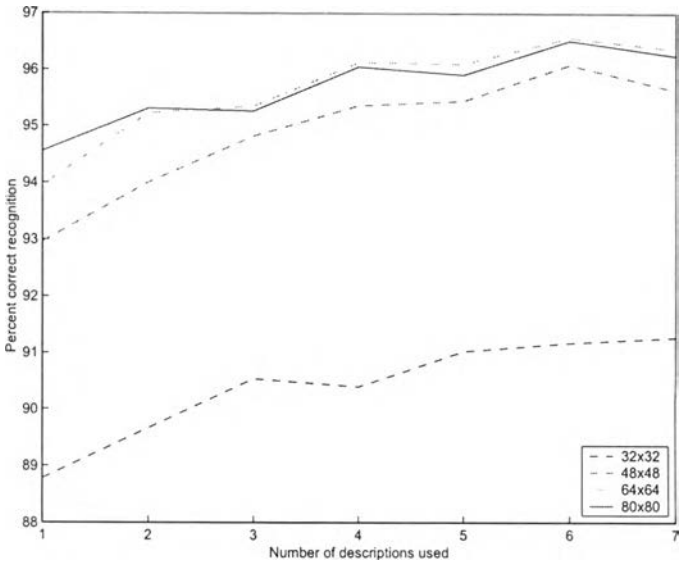

Figure 4.13: Percent of recognition accuracy of the classical code concatenation method based on MDC and Adaboost for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of image size and number of descriptions used.

Table 4.2: The performance of classical code concatenation method based on MDC and SVM-ECOC for 3-class MSTAR data set. Comparison of difference methods is in overall percentage of images correctly recognized as a function of image size.

Methods / Image Size	32x32	48x48	64x64	80x80
MDBC+NN	75.88	N/A	N/A	N/A
CGSM	N/A	N/A	N/A	**98.53**
SVM-ECOC (original)	84.46	90.16	91.76	92.70
SVM-ECOC (LDB)	85.42	90.81	92.51	92.14
MDC-SVM-ECOC	**87.47**	**92.96**	**93.77**	93.11

4.4.2 Generalized Concatenation Codes

Here two experiments were conducted and compared for both multiple classifier systems based on the classical and generalized code concatenation schemes for the Satimage data set.

4.4.2.1 Multiple Description Ensembles of SVM-ECOC

We used the third order Daubechies filters for computing discrete wavelet packet transform into three decomposition levels. On the implementation of the MDC, we created 8 redundant versions of the wavelet packet decomposition functions for each of the data subset. We computed and selected LDB for each transform version of each data subset, following the same procedure presented in Chapter 3. We then sorted the selected LDB and create 8 descriptions, also following the same procedure presented in Chapter 3. We evaluated the MCS methods using various numbers of MDB (from 29 to 33) for training the SVM-ECOC. However, the number of MDB used here is 33. We found that the more number of MDB used, the more recognition accuracy increased. Note that the SVMs with radial basis function (rbf) kernel were used in this evaluation. In fact, the kernel widths of 32, 40, 48, and 56 were used in these experiment settings.

Intuitively, the numbers of base classifiers used in both proposed multiple classifier systems should be comparable. For example, when comparing the generalized code concatenation schemes as in these experiment settings, we can use the (15,7) BCH output code for 7-class code matrix together with (7,4) exhaustive output code for 4-class code matrix. For the classical concatenation scheme, the multiple classifier system can be implemented by using (15,7) BCH output code for 7-class code matrix for encoding the 4 description data subsets. This way, the number of the base classifiers is thus become $n_{cc} = 15 \times 4 = 60$. For generalized concatenation scheme, we apply the procedure presented in Subsection 4.3.3 using (7,4) exhaustive output code matrix to encode 4 super-class partitions of each multiple descriptions (all of the 8 descriptions). In detail, information bit i_0 is mapped by 8-repetition outer code $\mathcal{A}^{(1)}$, and information bits (i_1, i_2) are mapped to encoded bits $(i_1, i_2, i_1, i_2, i_1, i_2, i_1, i_2)$ by outer code $\mathcal{A}^{(2)}$. Thus, the first row information bits (i_0, i_1) is used to encode 4 super-class partitions of classes. The second row information bits (i_0, i_2) is also used to encode another 4 super-class partitions of classes. This repeats for the other row sequentially as well. As a consequence, the number of the base classifiers is thus become $n_{gc} = 7 \times 8 = 56$. However, one may avoid the ambiguous decision usually occurred when the number of the classifiers is even. In the following experimental results, we thus summarize the MCS classification accuracy based on their best performance.

As shown in Figure 4.15, the MDC classical code concatenation scheme gives better performance than the single SVM-ECOC methods and the generalized code concatenation scheme. As shown in Table 4.3, the experimental results of the MCS methods using the

Table 4.3: The performance of generalized code concatenation method based on MDC and SVM-ECOC for Satimage data set. Comparison of difference methods is in overall percentage of patterns correctly recognized.

Methods / Data Set	Boosting	Random Subspaces	Random Tree	SVM-ECOC (original)	SVM-ECOC (LDB)	CCC (LDB)	GCC (LDB)
Satimage	91.89	92.19	92.24	91.2	90.5	91.3	90.1

kernel width of 40 is presented. It should be noted that the results in the table is the best classification performance with 8 and 3 descriptions of the classical and generalized schemes, respectively. The proposed methods provide comparable results compared with the state-of-the art MCS methods.

The reason why the classical code concatenation scheme outperformed the generalized scheme lies on the fact that in theory the minimum distance of the former scheme is larger than the latter scheme. This comes from the fact that when we use 4 descriptions of (15,7) BCH output code matrix in the classical scheme, the minimum distance is $d_{cc} = d_a d_b = 4 \times 8 = 32$. On the other hand, when we evaluate the generalized scheme with 8 descriptions with (7,4) exhaustive output code matrix, the minimum distance is $d_{gc} = 4 \times 4 = 16$ (the minimum between superclass codewords is 4 for each (7,4) exhaustive output subcode, and this label is protected with an outer code (i_1, i_2) with distance 4). Moreover, the column separation of the subcode is less separated, so there may be more correlation among classifiers trained by the generalized scheme than the classical scheme. From the above explanation, we can see why the classical scheme is outperformed the generalized scheme when the number of classes for the classification problem is small. However, when the number of classes for the classification problem is increased (greater than 8), the difference between both classification performance should be negligible, and at some point the performance of the generalized scheme should be outperformed the classical scheme. For example, when the number of classes is 16, the minimum distances of both schemes become 28 and 16, respectively. In particular, we then end up using (15,16) BCH output code matrix for classical scheme (the minimum distance becomes 7), and (15,8) BCH output code matrix for encoding superclass (the minimum distance becomes 8) with the protection of 2-repetition outer code (for the comparable number of base classifiers, we have to reduce the number of descriptions to 4 instead of 8). It is in this context that we are continuing in investigation the generalized scheme for a large classification problem.

4.4.2.2 Bagged Ensembles of SVM-ECOC

In this experiment, a collection of N bootstrapped samples with n elements, generated by choosing at random examples in the training set, according to a uniform probability distribution. We also created 8 description data subsets. We also conducted a series

Table 4.4: The performance of generalized code concatenation method based on Bagged ensembles of SVM-ECOC for Satimage data set. Comparison of difference methods is in overall percentage of patterns correctly recognized.

Methods / Data Set	Boosting	Random Subspaces	Random Tree	SVM-ECOC (original)	BSVM-ECOC (CCC:original)	BSVM-ECOC (GCC:original)
Satimage	91.89	92.19	92.24	91.2	91.22	89.91

of experiments in a similar way to the generalized concatenation scheme with multiple description coding. As presented in Table 4.4, our proposed method has comparable results compared with the state-of-the art MCS methods. Similarly, the classical scheme is outperformed the generalized scheme using the Bagging approach.

4.5 Conclusions

There are three broad approaches to MCS. One is based on combining strong learners. The second approaches is based on combining weak learners. The last approach is based on modular nets and hybrids. Giving that the aim of coverage optimization method presented in Chapter 3 is satisfied. Our proposed method learns an ensemble of classifiers that are biased to have high precision (as opposed to, for example, boosting, where the ensemble members are biased to ignore portions of the instance space).

Our proposed methods for coverage construction of multiple classifier systems are developed and applied to the public MSTAR database and the UCI repository data set. The main task of these methods is to construct multiple classifier systems based on two extensions of the ECOC approach: (1) classical code concatenation and (2) generalized code concatenation so that the highest classification performance is attained. An evaluation of the proposed methods on the classification of two public data sets provides additional proofs of the prospect of the improvement of our proposed multiple classifier systems.

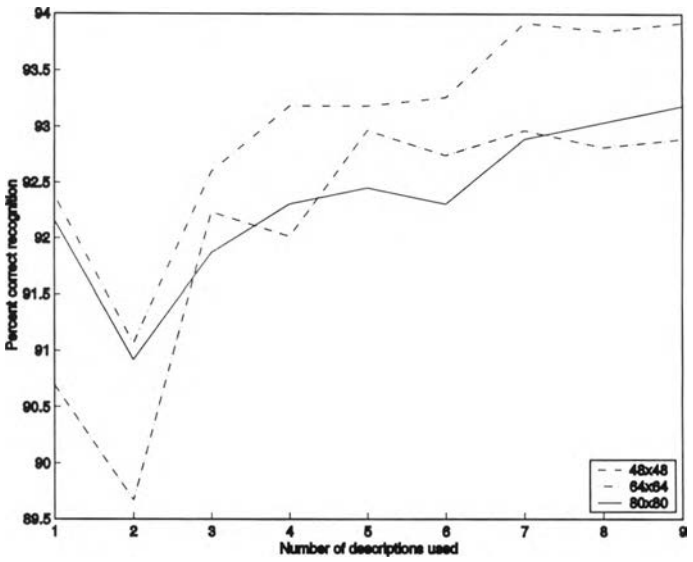

Figure 4.14: Percent of recognition accuracy of the classical code concatenation method based on MDC and SVM-ECOC for 3-class MSTAR data set. Each recognition accuracy is plotted as a function of image size and number of descriptions used.

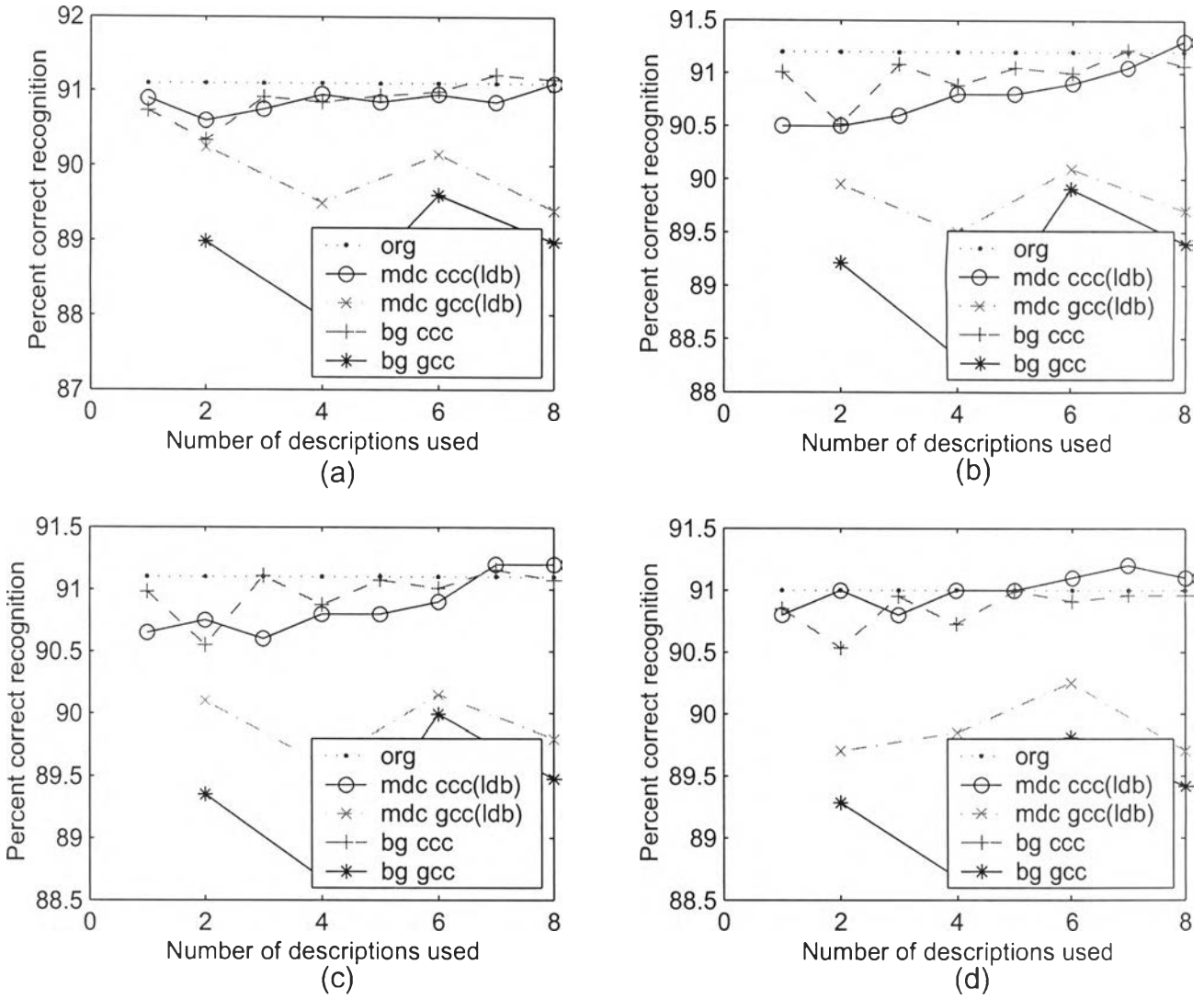

Figure 4.15: Percent of recognition accuracy of all of the code concatenation methods for Satimage data set. Each recognition accuracy is plotted as a function of image size and number of descriptions used. a) kernel width 32. b) kernel width 40. c) kernel width 48. d) kernel width 56.