Cooperative Training in Multiple Classifier Systems

by

Rozita Alaleh Dara

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Rozita A. Dara

Abstract

Multiple classifier system has shown to be an effective technique for classification. The success of multiple classifiers does not entirely depend on the base classifiers and/or the aggregation technique. Other parameters, such as training data, feature attributes, and correlation among the base classifiers may also contribute to the success of multiple classifiers. In addition, interaction of these parameters with each other may have an impact on multiple classifiers performance. In the present study, we intended to examine some of these interactions and investigate further the effects of these interactions on the performance of classifier ensembles.

The proposed research introduces a different direction in the field of multiple classifiers systems. We attempt to understand and compare ensemble methods from the cooperation perspective. In this thesis, we narrowed down our focus on cooperation at training level. We first developed measures to estimate the degree and type of cooperation among training data partitions. These evaluation measures enabled us to evaluate the diversity and correlation among a set of disjoint and overlapped partitions. With the aid of properly selected measures and training information, we proposed two new data partitioning approaches: Cluster, De-cluster, and Selection (CDS) and Cooperative Cluster, De-cluster, and Selection (CO-CDS). In the end, a comprehensive comparative study was conducted where we compared our proposed training approaches with several other approaches in terms of robustness of their usage, resultant classification accuracy and classification stability.

Experimental assessment of CDS and CO-CDS training approaches validates their robustness as compared to other training approaches. In addition, this study suggests that: *1)* cooperation is generally beneficial and *2)* classifier ensembles that cooperate through sharing information have higher generalization ability compared to the ones that do not share training information.

Acknowledgments

I am indebted to my supervisor, Professor Mohamed Kamel, for being an indefinite source of encouragement and inspiration. He taught me to first be a human being and a scientist. I could not have completed this thesis without his support.

I am grateful to Dr. Hamid Tizhoosh for his support and guidance throughout my graduate studies.

I would like to thank the members of my committee Dr. Fakhri Karray and Dr. Magdy Salama for serving on my thesis committee.

I also wish to express my gratitude to my colleague, Masoud Makrehchi, for his enthusiasm and stimulating discussions.

I feel a deep gratitude to my parents, Nayereh Estrabadi and Mohammed Dara, who have sacrificed so much to provide me with the quality of life from which I could choose what I wanted to be or to do. This thesis is dedicated to both of them, with love and sincere thanks for all they have done for me throughout my life and as well as during the life of this project.

I owe immeasurable thanks to Azita Dara, my sister, and Armin Dara, my brother, for their love and continuous support.

My especial thanks go to my colleagues and friends Mahtab Kamali, for her support, as well as Roshanak Moradi and Parisa Bohlouli who made the time I spent in Waterloo enjoyable.

Finally, financial support from the Natural Sciences and Engineering Research Council of Canada, Ontario Graduate Scholarship, and Faculty of Engineering, University of Waterloo, is greatly acknowledged.

Dedication

I would like to dedicate this thesis to my husband, Shayan, for living with the thesis as well as me, and for countless ways he ensured that I would finish every bits and pieces. Without his patience, encouragement and understanding, this work would have not been possible.

Contents

List of Tables

List of Figures

List of Abbreviations

A-BST	Ada-boosting Approach
AVE	Average Vote Rule
Bay	Bayesian Combining rule
BKS	Behaviour-Knowledge Space
CB-CNT	Class Border Contour Overlap
CC-CNT	Class Center Contour Overlap
CDS	Clustering, De-clustering, and Selection Approach
CDD-space	Class Diversity and Distance Space
CELS	Cooperative Ensemble Learning System
CO-CDS	Cooperative Clustering, De-clustering, and Selection Approach
CNT-RAN	Contour-Random Overlap
DAC	Divide-and-conquer Approach
DA	Distance Array
DCS	Dynamic Classifier Selection Approach
DT	Decision Template Approach
FBDF	Feature-Based Decision Fusion Approach
GA	Genetic Algorithm
HME	Hierarchical Mixtures of Experts
MAJ	Majority Vote Rule
MCS	Multiple Classifier System
MLP	Multi-layer Perceptron
OB	Out-of-Bag Approach
PROD	Product Rule
RAN-OVR	Random-Overlap strategy
RP	Random Partitioning
RSM	Random Subspace Method
SB	Single Best Classifier
SG	Stacked Generalization Approach
SLC-OVR	Slice-Overlap strategy
SLC-RAN	Slice-Random Overlap
SOM	Self Organizing Maps
SP	Stratified Partitioning
STD	Standard Deviation

Chapter 1

Introduction

1.1 Background

The field of Multiple Classifier Systems (MCS) falls within the supervised learning paradigm. This task orientation assumes that a set of training patterns has been given to the system. These training patterns are presented by feature vectors. Each training pattern is labeled with a class target, which is a member of a finite set of class labels. The goal of supervised learning is to predict the class labels of unseen patterns and to do the task accurately and efficiently.

It is a well-known fact that in supervised learning a good performance on training data does not necessarily mean accurate generalization ability for unseen data patterns. Several classifiers can have similar training performances, but they may not have similar generalization performances. In fact, it has been observed that although one design may outperform the others, the patterns that are misclassified by different classifiers are not necessarily the same [61]. This observation suggests that the use of multiple classifiers can reduce the risk of using a poor performing classifier. The use of multiple classifiers may or may not outperform the best classifier in the ensemble, but it reduces the chances of selecting an inaccurate classifier. This approach is common in everyday human life. Asking for the opinion of several experts is a common practice in our everyday life. We seek the opinion of several users on the quality of a product, or consult several doctors for a diagnosis.

There have been numerous studies in the field of MCS. A great number of meth-

ods for combining classifiers have been proposed in the past [81, 59, 60]. Some researchers have concentrated on solving particular problems [3, 7, 58, 64, 82, 97, 99]. Few of these studies have explicitly focused on sequential pattern recognition problems such as time series [19], speech recognition [78] and hand recognition [62]. These approaches usually differ from each other with respect to the specific procedure used for generating individual classifiers, and/or the strategy employed for combining the classifiers. Only a few researchers have considered the theoretical issues behind these developments [61, 72, 101].

1.2 Objective and Approach

The primary interest of this research was to study cooperation in MCSs. Our intention was to examine the impact and gain in the efficiency, when various components of MCS cooperate. Some important questions originally motivated this project:

- Does combination of classifiers always result in higher accuracy?
- What criteria should the MCS meet in order to improve the accuracy over the best single classifier?
- What type of cooperation among the components is the most effective?
- How can cooperation in MCS be evaluated?

Discussions on issues regarding cooperation such as type, effectiveness, and measures are the focal point of this study.

Although the issue of cooperation in MCS have been previously raised by other researchers (e.g. [91]), there is dearth of studies in the literature that have investigated cooperation in a systematic manner. We distinguished four levels of cooperation: decision, architecture, feature, and training. The degree and method by which classifier ensembles "share" resources was used as measures of cooperation. We narrowed down our focus on the training level and examined the effects of sharing training patterns and sharing training information on the performance of multiple classifiers. We focused on the training level since this is considered to be the most important step in the construction of MCSs. The importance of this level

stems from the fact that generalization ability of the ensemble highly depends on the performance of the individual classifiers.

We first developed measures to assess cooperation (sharing) among training data partitions. Furthermore, we examined the advantages of cooperation through sharing training information. With the aid of properly selected measures and training information, we developed two new MCS training methods. In the end, a comprehensive comparative study was conducted where we compared our proposed training methods with several other existing methods in terms of classification accuracy and classification stability. The findings of this comparative study suggest that, first, cooperation is generally beneficial and, second, classifier ensembles that cooperate through sharing information are more accurate and stable.

In general, the importance of this research is threefold: 1) we introduce a different direction in which the ensemble methods are categorized and studied from the cooperation perspective, 2) novel evaluation measures are proposed with which the type and degree of cooperation among training partitions can be estimated, 3) a new classifier ensemble training technique, in which training information is utilized to obtain sub-optimal training partitions, is developed and empirically evaluated.

1.3 Thesis Overview

Chapter 2 provides an overview of the multiple classifier system field. This includes a discussion of different design criteria, introduction to various MCS design approaches, and a review of the current categorization schemes.

In Chapter 3, a novel categorization of MCS techniques is proposed. In this categorization, MCS techniques are grouped based on their type of cooperation into four levels of training, feature, architecture and decision. Each of these levels is discussed in detail.

The proposed approach is discussed in Chapter 4. A number of evaluation measures and partitioning strategies are introduced. Two new training partitioning methods are proposed.

A comprehensive comparative analysis of different MCS training approaches is presented in Chapter 5. MCS training approaches are compared using different criteria. The experimental results are presented in the same chapter.

Chapter 6 highlights the conclusion with a discussion of the contributions made in this study and discusses the future work.

Appendix A provides detailed information about the datasets and additional results.

Chapter 2

Multiple Classifier Systems

Information fusion refers to the fusion of data from multiple sources. The goal of information fusion is to provide knowledge and to improve the accuracy that is not achievable from individual sources [24, 25]. Exponential growth of information and data in the recent years has immensely contributed to the popularity of information fusion and the proposed methodologies. Technological and societal developments have increased the need for information fusion, filtering or extraction. Various research areas have recently emerged from information fusion including sensor fusion, image fusion, decision fusion, knowledge fusion, and classifier fusion. The focus of this study is on classifier fusion.

Several lines of evidence suggest that combining individual classifiers is an effective technique for improving accuracy of classification [1, 2, 16, 52, 61, 93, 105]. The interest in multiple classifier systems (MCSs) stems from the fact that the classical approaches for designing pattern recognition systems were primarily focused on searching for the best individual classifier [53]. This approach has some serious drawbacks. The main drawback is that the best individual classifier for the classification task at hand is very difficult to find, unless there is some prior knowledge about the problem. In addition, with the use of single best classifier, the chance of loss of information is high. MCSs have the promise of reducing these caveats of standard classification approaches. MCS is referred to by a variety of names including classifier ensembles [93], combination of multiple classifiers [61, 75], classifier fusion [41], multiple experts, mixture of experts [55, 56], dynamic classifier selection [43], divide-and-conquer classifiers [38, 40], composite classifier system [95], committees

of classifiers, decision combination, and consensus aggregation [68].

2.1 Multiple Classifier Systems Design Criteria

Three general criteria have been employed in the design of multiple classifiers: *1)* accuracy of the individual classifiers, *2)* diversity of the individual classifiers, and *3)* efficiency of the entire MCS [95]. In the following sections, we discuss each of these criteria in more details.

2.1.1 Accuracy

The accuracy of the individual classifier is an important criterion for MCS design. A great deal of effort usually goes in training the individual classifiers. In the past, some studies have emphasized on the construction of highly accurate, but independent, classifiers [88, 92], while a few others have investigated the combination of weak classifiers [37, 54].

It is expected that each individual classifier misclassifies some of the data patterns. Kittler *et al.* [61] showed that the patterns misclassified by different classifiers are not necessarily the same. This important observation suggests that combining the prediction of several classifiers can be beneficial, if the impact of errors made by the classifiers is minimized. It has been previously proven by Tumer and Ghosh [101] that by increasing the number of distinct base classifiers L, the MCS error ($Error_{MCS}$) decreases. They showed that MCS error is related to the Bayesian error as

$$Error_{MCS} = \frac{1 + \sigma(L-1)}{L} E_{added} + E_{bayes} \tag{2.1}$$

where E_{bayes} is the Bayesian error, E_{added} is the added error due to the specific classifier, and σ is the correlation factor. As the correlation factor σ decreases, the MCS performance improves. For example, if $\sigma = 1$, the ensemble performance is similar to a single classifier. Therefore, there is no advantage in combining base classifiers that are highly correlated. If $\sigma = 0$, the ensemble error decreases. It can also be concluded that combining the prediction of base classifiers with high error

rates is not beneficial either. If base classifiers are highly correlated because of their poor performances, the combined prediction will also be inaccurate. MCSs are most useful if the base classifiers make independent errors and the error rates are less than 50% [94].

There are several ways to generate classifiers with different generalization ability. The commonly used methods are mostly based on manipulation of training parameters such as initial conditions, the training data, classifier topology, and the training algorithm. These methods are briefly reviewed in the next section.

2.1.2 Diversity

The idea of "diversity of failure" can be used to improve the performance of MCS. It is obvious that there is no advantage of combining classifiers that show identical behaviour. Some studies [94, 96] have shown that a good classifier combination system is the one where individual classifiers are both accurate and make their errors on different parts of the input space. This property is known with many names in the literature including: disagreement, diversity, and independence.

Diversity among the classifiers in the MCS has been recognized as an important criterion and it has been well studied and addressed in the literature [67, 70]. A set of diverse classifiers may be generated by utilizing different methods:

- Different initializations: In some classifiers (e.g. Neural Networks), different training initializations may result in diverse classifiers.
- Different parameter choices: Classifiers such as Decision Trees, Neural Networks, and k-nearest neighbour are parameter-dependent. Change of parameters in these classifiers can generate diverse classifiers.
- Different architectures: Changes in the architecture can produce diverse classifiers. For example, different size of hidden nodes or hidden layers can result in diverse neural network committee.
- Different types of classifiers: Training different types of classifiers on the same dataset is one of the popular techniques in producing diversity in MCS.

- Different training sets: The most commonly used method for construction of individual classifiers is training data alteration. A diverse pool of classifiers may be generated by training each individual classifier on different subsets of the data (e.g. boosting, bagging and k-fold crossvalidation).
- Different feature sets: In some applications, training classifiers on random subsamples of a large feature set has been shown to be a successful method for generating diversity.

Measuring Diversity

Kuncheva and Whitaker [70] provide an overview of eight diversity measures. They have empirically studied the relationship between the majority vote combination method and different diversity measures. Two categories of diversity measures exist: pairwise and non-pairwise. In the pairwise measure, diversity between two classifiers is estimated. An overall estimate is obtained by averaging the diversity of all pairs. In the non-pairwise measure, however, diversity among all the classifiers is calculated. In this section, two diversity measures, Q statistic and Entropy, are introduced.

Q statistics

This is the pairwise symmetrical measure of diversity. For two classifiers C_i and C_j where $i, j \in 1, ..., L$, Q statistic is defined as

$$Q_{ij} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{2.2}$$

where N^{11} is the number of patterns that both C_i and C_j correctly classify,
N^{10} is the number of patterns that C_i correctly classifies and C_j misclassifies,
N^{01} is the number of patterns that C_j correctly classifies and C_i misclassifies,
N^{00} is the number of patterns that both C_i and C_j misclassify,
and $N^{11} + N^{01} + N^{10} + N^{00} = N$ (Table 4.4).

For statistically independent classifier $Q_{ij} = 0$; Q varies between -1 and 1. The higher the absolute value of Q, the less diverse the team of classifiers.

Entropy

Table 2.1: Q Statistic Measure

Classifiers	C_j correct	C_j wrong
C_i correct	N^{11}	N^{10}
C_i wrong	N^{01}	N^{00}

Entropy is an information-theoretic measure that captures the variability of classifiers decisions. This method is a non-pairwise measure defined as

$$E = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{(L - \lceil \frac{L}{2} \rceil)} min\{l(x_j), L - l(x_j)\} \tag{2.3}$$

where n is the number of patterns, x_j is a data pattern, and $l(x_j)$ is the number of classifiers that correctly classifies x_j. E varies between 0 and 1, where 0 indicates no difference and 1 indicates the highest possible diversity.

2.1.3 Efficiency

Avoiding costly classifiers has been considered as a design goal. If all other parameters are similar, it is preferred to have *1)* fewer classifiers over more, and *2)* computationally inexpensive classifiers or algorithms over more expensive ones. Clearly, if they all perform equally, fewer classifiers is preferable, since training and costs will be lower.

Although it has been theoretically proven that large ensemble size results in lower error, this may not be feasible for real-world problem domains. Availability of large and complex datasets is a common characteristic of most real-world domains. As a result, training a large pool of base classifiers for these problems can be a costly process. This issue has not been closely examined previously, however, some studies have indirectly addressed the problem. Distributed leaning type approaches have been proposed [18, 76] in which training subset sizes and the number of classifiers are adjusted according to the datasets and learning computation cost. In addition, selection of classifier type based on the application in hand can effectively reduce the cost of such systems.

Not only the computational resources of the base classifiers, but also the com-

plexity of aggregation rule or the MCS architecture can cause problems. This issue is in particular more evident where the MCS architecture consists of several layers of multi-layer perceptron classifiers, e.g. error-correcting output code [30]. Little attention has been given to this issue. Amongst a few, feature-based decision fusion algorithm [104] tackles the problem of reducing the computational resources by generating weak learners in the initial steps, and reusing and adjusting algorithm parameters throughout the retraining iterations.

2.2 Data Decomposition

One of the popular techniques in MCS construction is data decomposition methods. In some of the techniques, multiple hypotheses are generated by modifying the dataset in three levels of training patterns, features, and output patterns. In each one of these levels, data patterns are partitioned into smaller sets or new patterns are added to the old ones to make the problem simpler or to reduce correlation among classifiers. Some of these techniques are discussed in the following sections.

2.2.1 Training Partitioning

Training data partitioning is a popular method for constructing MCS. Classifiers, in this approach, are each trained with a different subset of the training examples. As stated before, ensemble of diverse classifiers may have a better potential to outperform a single best classifier. By training classifiers with different subsets of training patterns, highly correlated classifiers can be avoided. Some of the well-known data sampling techniques, Bagging, Boosting, and k-fold-crossvalidation, are discussed in more detail.

Bagging

One of the popular ways of manipulating the training set is called Bagging [17]. On each run, Bagging presents the classifiers with a training set that consists of a sample of training examples selected randomly with replacement from the original training set. Such a training set is called a bootstrap replicate of the original training set. Each bootstrap replicate contains, on average, 63.2% of the original training

set, with several training examples appearing multiple times. Bagging modifies the training dataset, builds classifiers on these modified training sets and then combines them into a final decision.

Ada-Boosting

Ada-Boosting, proposed by Freund and Schapire [37], is a technique to combine a set of weak classifiers, with the objective of obtaining a classification algorithm with a better performance. In Ada-boosting, classifiers and training sets are obtained in a deterministic way. At each step, weights are assigned to data patterns in such a way that more weight is placed on training examples that are misclassified by classifiers and less weight on examples that are correctly classified. This way, classifiers get focused on more difficult learning problems in each step. Boosting modifies the training dataset, trains the classifiers on these modified training sets and then combines them into a final decision.

k-fold-crossvalidation

In k-fold-crossvalidation, the training set is randomly divided into k subsets. Then, *k-1* of these overlapping subsets are used to train the classifiers and results are tested on the subset that is left out of the training. By changing the subset that is left out of the training process, k classifiers can be constructed, each of which is trained on a different training set [85].

2.2.2 Feature Decomposition

The use of different feature subsets, for training classifiers, has been recognized as a promising design method for the MCS. Feature space may be partitioned by random selection, genetic algorithm, input decimation, or other statistical approaches. Tumer and Oza [100] applied an input decimation technique to select features and reduce correlation between classifiers. In their proposed approach, classifiers were trained on an equal number of feature partitions. Input decimation applies principle component analysis to the feature space to generate subsets that each correspond to a specific class. Windridge and Kittler [106] investigated prior selection of features via forward selection technique applied to the classifiers that were functioning

individually or as a team. They also examined the backward selection technique examined on the individually optimized feature sets. Kuncheva and Jain [73] suggested genetic algorithm (GA) for selection of features. They implemented two versions of GA algorithms to select disjoint and overlapping feature subsets as well as types of individual classifiers. Rodriguez *et al.* [86] proposed a MCS construction method called rotation forest. Similar to input decimation, rotation forest utilizes principle component analysis to obtain diverse and accurate classifier. This method consists of generating several random feature subsets, in the first step, and regenerating these feature subsets by applying principle component analysis on each of them separately.

Instead of using feature subset selection techniques, some researchers have suggested a reconstruction scheme for the feature space by the addition of new attributes. Cascade generalization, [42], is an example of such techniques. Cascade generalization combines classifiers in an iterative way. At each iteration, a classifier is generated. The feature space is extended with new attributes.

2.2.3 Output Decomposition

Another level of data manipulation is on the output patterns. When the number of classes in the problem is larger than 2, they can be broken down into multiple sub-problems. The main motivation for this methodology is that the sub-problems are typically much simpler than the original problem. In the machine learning community, an output space decomposition method called error-correcting output coding is introduced by Dietterich and Bakiri [30]. In this method, each member of an ensemble solves a 2-class problem obtained by partitioning the original classes into two groups based on error correcting codes. Error correcting code method codes the classes to binary strings and assesses the redundancy in the resulting coding to optimize classification performance. Error correcting code works best for small training sizes.

2.3 Architectures

An alternative approach to MCS construction is in terms of architecture topology. Categorization of combination methods, in this level, can be made according to

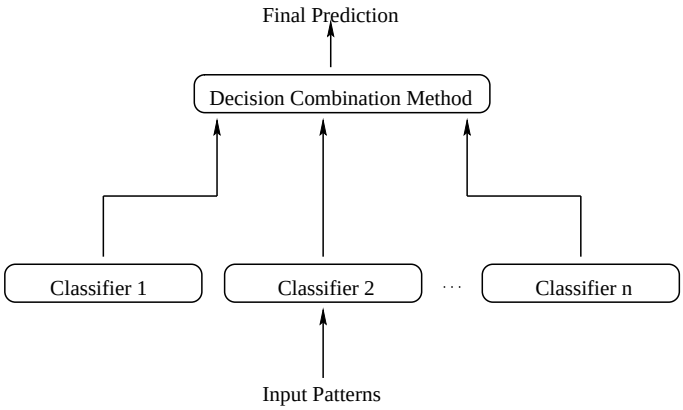


Figure 2.1: Parallel Topology

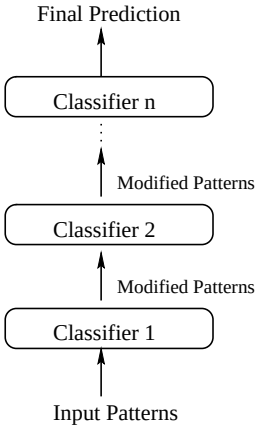


Figure 2.2: Sequential Topology

the way classifiers interact. Architecture topologies can be classified as parallel, sequential, hierarchical, and conditional [74].

2.3.1 Parallel Topology

This is the most commonly used topology in the MCS and has been studied, both empirically and theoretically [61, 59, 72, 81, 103]. In this category, classifiers first operate in parallel to predict label for an unknown pattern. Subsequently, their decisions are combined for the final prediction (Figure 2.3.1). Parallel combinations can be implemented using different strategies and the type of information produced by the classifiers. Combination schemes may vary from simple ones such as maximum, minimum, sum, product, median, majority vote, and averaging to the more complicated ones such as fuzzy integrals, weighted averaging, decision templates, and logistic regression (Section 2.4).

2.3.2 Sequential Topology

Classifiers in this topology are applied in sequence where each classifier produces a modified set of possible classes for each dataset. Through this process, a complicated problem is progressively reduced to simpler ones (Figure 2.2).

An example of this approach is discussed in [107]. Stacked Generalization is a framework for classifier combination in which different layering may be created

by iteration. Each layer is used to combine the predictions of the classifiers at the immediately lower layer. A single classifier at the top-most level will do the final prediction. The information passed from layer to layer may take the form of vectors of predictions, confidence values, or other data. Another example of this architecture is Cascade generalization [42]. The basic idea behind this architecture is to use classifiers in sequence, with the difference of using an extension of training data for the upper levels. At each iteration, new attributes are added to the training patterns. These new attributes represent the probability that patterns belong to specific classes. The single layer at the final level will make the decision.

2.3.3 Hierarchical Topology

In this topology, the system has multiple layers and for each layer, classifiers produce a reduced set of possible classes for each pattern. This way, classifiers become more focused on the problems (Figure 2.3). An example of this topology is Hierarchical of Mixtures of Experts (HME) [55, 56]. The HME has a tree structure and classifiers sit at the nodes of the tree, in which the input space is divided into sets of regions. Division of regions are done recursively and regions have soft boundaries which means that data points may lie in different sections. Part of the HME learning process is to assign experts (classifiers) to the different regions and then to use a gating method to decide which classifier should be used to assign the label. Kumar and co-workers [64] propose a modular learning system based on and automatically generated binary hierarchy of the classifiers. Each of the classifiers solves a two-class problem with a specific feature space that belongs to those classes. The dataset is first partitioned into two disjoint subsets. These disjoint sets are further partitioned recursively until each partition is reduced to one of the original classes. The output of the classifiers are combined based on some weights which can depend on the input or predictor.

2.3.4 Conditional Topology

Under this structure, a primary classifier is first applied to the data. If the classification is made with low confidence, another classifier is utilized. This structure has the advantage of computational efficiency when the initial classifier is a low cost type and the second one can be more sophisticated for difficult patterns. Asker and

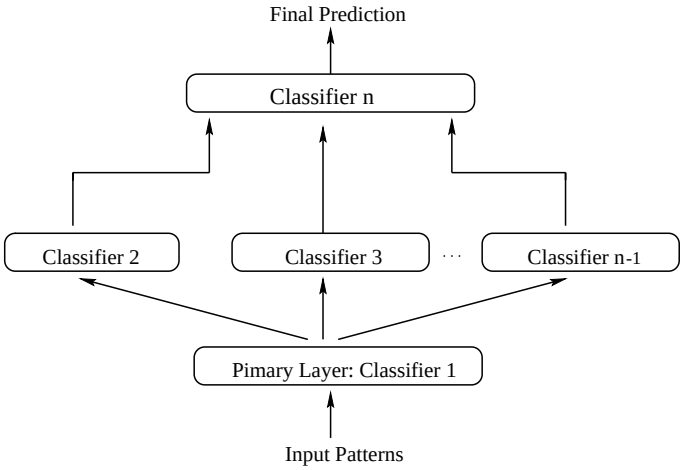


Figure 2.3: Hierarchical Architecture

Maclin [4] propose a method which is based on a simple estimation of each classifiers performance. The classifiers are grouped into an ordered list where each classifier has a corresponding threshold. In the classification phase, the first classifier on the list is consulted. If the prediction confidence of that classifier is above the predefined threshold, then that classifier is used for final decision making. Otherwise, the next classifier and its threshold is considered. If none of the predictions made by the classifiers is above the threshold, then averaging technique will be performed on all the predictions.

2.4 Aggregation

In addition to methods that focus on architectures and data decomposition, a distinction can be made among methods that manipulate outputs of the base classifiers. The type of information provided by base classifier's outputs can be categorized into three levels: abstract (or crisp), ranked and measurement [98]. Abstract-level classifier generates a unique predefined class label for each input pattern. Ranked-level classifier ranks all the labels in a queue with a label at the top being the most probable class. On the other hand, measurement-level classifier estimates a set of confidence values, each belonging to a different class, for an input pattern. Each classifier ensemble method uses a different type of output level. In general, two clas-

sifier combination strategies are identified: classifier *selection* and classifier *fusion* [68].

2.4.1 Fusion

Classifier fusion assumes that all classifiers are equally experienced in the whole feature space and the decisions of all classifiers are taken into account to assign label to an unknown pattern. Two sets of classifier fusion methods have been distinguished: fixed and trained. In the fixed aggregation scheme, the outputs of individual classifiers are considered as a clear interpretation of the decision [103]. While in the trained aggregation scheme, outputs are considered as input to a second-level classifier in some intermediate feature space [33].

Fixed Aggregation Scheme

Fixed combiners have been extensively studied (e.g. [1] and [72]). In the fixed scheme, outputs of the classifiers are directly used in prediction of label for an unknown pattern. The following sections summarize some of the well known simple fixed rules.

Let $X \in \mathbb{R}^m$ be the data-space and y_i be a set of class labels $\forall y_i \in \Omega = 1, 2, ..., k$. We consider a training set

$$S = \{(x_1, y_1), ..., (x_n, y_n)\}, \tag{2.4}$$

of size n drawn form an unknown distribution D. Let $C = C_1, C_2, ..., C_L$ be a set of classifiers. For each classifier trained with S, the output is a k dimensional vector of $C_i(x) = [c_{i,1}(X), c_{i,2}(X), ..., c_{i,k}(X)]$, where $c_{ij}(X)$ is the confidence value of classifier C_i for class j and for a given input pattern X. The entry c_{ij} can be either crisp $c_{ij} \in \Omega$ or a confidence value estimated as:

$$c_{ij}(x) = P(y_j|x). \tag{2.5}$$

- Product Rule

$$Q_j(x) = \prod_i c_{ij}(x)$$

where j represents the classes and i represents the classifiers. The final prediction of combined classifier Q is made by hardening the decision using maximum membership formula. Product rule works well with independent classifiers [61]. However, achieving independency among classifiers that are trained with the same data, is hardly ever possible.

- Sum Rule

$$Q_j(x) = \sum_{i=1}^{L} c_{ij}(x) \tag{2.6}$$

The final prediction of combined classifier Q is made by hardening the decision using maximum membership formula. The sum rule is equivalent to product rule for small deviations in the classifier outcomes. This rule may be used to improve classification using similar classifiers with independent behaviour [61].

- Maximum Rule

$$Q_j(x) = max_i\{c_{ij}(x)\} \tag{2.7}$$

The maximum rule selects the classifier with higher training accuracy. This may seem reasonable. However, there is always a chance that some classifiers might be more overtrained than others. In this case, they will dominate the outcome without having a better performance.

- Minimum Rule

$$Q_j(x) = min_i\{c_{ij}(x)\} \tag{2.8}$$

The minimum rule will select the outcome of the classifier that has the least objection against a certain class. This rule has the same drawback as the maximum rule.

- Majority Vote

The majority rule is the most popular way of combining classifiers. The final decision is made by selecting the label that is most represented by the individual classifiers.

- Average Vote

  This method averages the individual outputs of the classifiers for each class. The final decision is the class that has the highest average value.

$$Q_j(x) = max_j\{\frac{1}{L}\sum_{i=1}^{L} c_{ij}(x)\}$$

Trained Aggregation Schemes

Instead of using fixed rules, a training rule can be used to adapt the MCS to the classification problem. Outputs of the individual classifiers can be treated as the input to a second-level learner. The advantages of trained approaches have been previously demonstrated in [33, 83, 98]. Two different strategies can be employed for training of the combiner. One is to use the original training data to train both base classifiers and the combiner. Another strategy is to use a different set, referred to as validation set, to train the combiner.

Neural Network classifiers have been widely used as a second level combiner. Wanas and Kamel [104] propose a novel trained aggregation rule that makes the decision fusion a more adaptive process. In their proposed method, the aggregation scheme is divided into two phases: a learning phase, and a decision making phase. In the learning phase, a multi-layer perceptron is used to assign a weight factor to each prediction to support each decision maker. This weighting factor represents a confidence in the output of each classifier. These confidences are then aggregated using fixed classifier-combining methods. Behavior-Knowledge Space (BKS), Decision Template, and Bayesian rule [68] are other examples of such trained aggregation schemes.

- Weighted Average:

  This method is similar to the average vote, with the exception that the output of base classifiers are multiplied by a weight. Combination of various classifiers is constructed by forming weighted sums of the classifier outputs.

$$Q_j(x) = max_j\{\frac{1}{L}\sum_{i=1}^{L} w_i c_{ij}(x)\} \tag{2.9}$$

The combination weights, w_i, are obtained by minimizing the mean squared error of the classifiers on the training data.

- Fuzzy Integral: Two basic types of fuzzy integrals have been proposed: Sugeno and Choquet [46]. Let H be a fuzzy set on C. The Sugeno fuzzy integral with respect to a fuzzy measure g is obtained by

$$Q_g = max_\alpha\{min(\alpha, g(H_\alpha))\} \tag{2.10}$$

where H_α is the α-cut of H, and g is called a λ-fuzzy measure. The fuzzy measure g can be calculated from a set of L values g^j, representing the individual importance of $D_1, ...D_L$. The value of λ is obtained as the unique real root greater than -1 of the polynomial

$$\lambda + 1 = \Pi_{j=1}^{L}(1 + \lambda g^j) \tag{2.11}$$

the value of g can be computed recursively as follows:

$$g(t) = g^{i_t} + g(t-1) + \lambda g^{i_t} g(t-1) \quad for \quad 1 < i < L. \tag{2.12}$$

The final degree of support of class y_j can be calculated by

$$Q_j = max_{i=1}^{L}\{min(c_{ij}(x), g(t))\}. \tag{2.13}$$

- Behaviour-Knowledge Space (BKS): Every possible combination of class labels is an index which is represented as a cell in the look up table. The class label most often encountered amongst the elements of this table during training is selected for the cell. The decisions generated by each classifier are compared against this lookup table. The decision for unknown pattern is made based on the class stored in the cell [**?**].

- Decision Templates (DT): Similar to the BKS, the DT approach generates a template for a given combination of classifier decision outputs. Each classifier in the ensemble outputs a degrees of support for each data pattern x. These outputs are first organized into a decision profile, in which the columns represent the support from all classifiers for a specific class, and the rows represent

the support from a particular classifier for all classes. The decision templates (DT) are then obtained for each class j as the average decision profile among all class j instances of the training data:

$$DT_j = \frac{1}{N_j} \sum_{X_j \in \Omega} X_j \tag{2.14}$$

where N_j is the number of class j and X_j is a set of training patterns that belong to true class Ω. For a given unknown pattern, a similarity measure is used to compare the decision of the classifiers for the unknown input to the templates previously generated. The most common similarity measure applied in decision template approach is squared Euclidean distance.

- Bayesian Rule: This method assumes that all base classifiers are mutually independent [109]. Let $\sum_{i=1}^{L} c_{ij}$ denote the total number of data that are assigned class j and c_{ij}^{c} denote the number of data with actual class j. Then, the soft label for class y_j is calculated by

$$P(y_j|x) = \frac{c_{ij}^{c}}{\sum_{i=1}^{L} c_{ij}}. \tag{2.15}$$

2.4.2 Selection

The assumption about the classifier selection is that each classifier is "an expert" in some local area of the feature space. When a pattern vector is submitted for classification, the classifier responsible for the given pattern is given the highest credit to label. Either exactly one classifier or more than one local expert can be nominated to make the decision. Two types of classifier selection systems have been distinguished: static and dynamic [71].

Static

In this system, selection regions are specified during the training phase, prior to the final step (classification of unlabelled data). In the operation phase, two strategies can be utilized: *1)* the region may be partitioned into different sections, then, an

appropriate classifier is assigned to that region, or *2)* based on the classifier, a region is found where each classifier has its best performance [71].

Static classifier selection has been discussed in several studies [71, 103]. The entire data space is partitioned into several regions using a clustering technique, i.e. fuzzy c-mean or frequency-sensitive competitive. Then, the reference point in each region is found by simply considering the center of the clusters or other methods. The accuracy of all the individual classifiers in each region is estimated, and the classifier with the highest accuracy is assigned to the region.

Dynamic

The choice of selection of a classifier to label the unknown pattern is made during the operation phase. The choice is made based on the certainty of the classifier decision in a region. The classifier with higher certainty would be nominated for labelling the unknown patterns [43]. Woods and co-workers [108] proposed a method for dynamic classifier selection based on classifier local accuracy estimates. For each classifier, an estimate of the accuracy in local regions of feature space around the unknown sample is computed. Local regions are defined in terms of k-nearest neighbours in the training data. The classifier with higher accuracy in training is nominated to assign a label to the unknown data.

The idea of combining fusion and selection approaches has also been discussed in Kuncheva and Rodriguez [65]. Random linear oracle works as following. Each classifier is substituted by a mini-ensemble of a pair of classifiers and an oracle. Oracle acts as a random linear function. The base classifiers are first trained by resampling or reweighting of the training patterns. In the classification phase, the oracle for the respective classifier decides which classifier in the mini-ensemble to use for classification of an unseen pattern. It has been claimed that this approach encourages extra diversity in the ensemble.

2.5 Methods for Designing Multiple Classifier Systems

Although some of the current design methods have shown to be very successful, a clear guideline for choosing the best design method is not yet available. There

have been some efforts in this direction. An "overproduce and choose" [88] (or "test and select" [92]) method has been proposed in order to design a MCS that has the optimal accuracy for the task at hand. The basic idea in this proposed paradigm is to produce a large set of initial classifiers using bagging or boosting (overproduction phase), and to select the subset of classifiers that can be combined to obtain optimal accuracy (choice phase). The optimal subset of classifiers can be selected using one of these four evaluation rules/functions:

- Heuristic Rules: "Choose the best" technique has been proposed as a heuristic rule by Partridge and Yates [84]. "Choose the best" technique selects n most accurate classifiers from the initial pool of classifiers, where n is a predefined value. Partridge and Yates also proposed another technique, referred to as "choose the best in the class". In this method, for each classifier type, the most accurate one is selected. For example, if we generate a large pool of classifiers of three different types, such as multi-layer perceptron, k nearest neighbor, and linear quadratic classifier, based on "choose the best in the class" technique, a subset of three different classifiers with the highest accuracy is selected. Such heuristic rules reduce the computational cost of selecting optimal classifiers, however, their efficiency can not be validated.

- Diversity Measure Criteria: Diversity can also be employed as an evaluation criterion to select the base classifiers. Various diversity measures have been proposed, such as Q-statistics, Entropy, within-set generalization diversity [67, 70, 84] (Section 2.1.2).

- Search Algorithms: Search algorithms can be used to select base classifiers. Sharkey *et al.* [92] proposed an exhaustive search algorithm with the assumption of having a small pool of classifiers. As effective as exhaustive search algorithms can be, they can be computationally costly when dealing with a large pool of candidate classifiers. Feature selection search algorithms can be adopted for the purpose of classifier selection. Forward search, backward search and tabu search are examples of such algorithms. Diversity measures and the accuracy assessed by the majority voting rule have been used as evaluation functions [88].

- Clustering Methods: An effective searching technique, with limited computational effort, can be achieved by clustering the pool of classifiers into several

disjoint subsets [88]. This method is based on the assumption that the error correlation between any two classifiers belonging to the same subset is greater than the one between any two classifiers belonging to different subsets. Based on this assumption, classifiers are clustered using a diversity measure (see Section 2.1.2) into several disjoint subsets. In each iteration of clustering, an ensemble is created by selecting one classifier from each cluster that has the largest distance with the other classifier members in the same cluster. Classifiers are combined with majority voting rule.

2.6 An Overview of the Current Categorization Schemes

Combination of multiple classifiers is a rich research area that can be considered from many different perspectives and combination techniques can be grouped and analyzed in different ways. There have been several attempts to classify and compare different types of MCS. These categorizations are based on different design techniques, methods of creation of ensemble and modular structures, implementation and topology of the architecture, and type of aggregation schemes. This section reviews some of these categorizations.

Classifier combination methods can be classified either as the ensemble combination of classifiers or the modular combination of classifiers [93]. The term ensemble is commonly used for combining redundant classifiers. The redundancy occurs because each classifier provides a solution to the same problem. This is in contrast to the modular approach, in which the problem is decomposed into a number of sub-tasks. Each module is concerned with completing a solution for a sub-task. To complete the whole task, each component is expected to contribute. In addition to modular-ensemble, top-down and bottom-up systems are identified [91]. In the top-down method, the outputs of the component modules are not used to make the final decision, while in the bottom-up system the outputs are applied. Most parallel architectures are bottom-up, whereas modular systems with control switch are top-down.

Ho [49] categorized the methods for creation of ensemble into generating coverage optimization and decision optimization. In the coverage optimization, a large set of weak classifiers, each trained on a different proportion of training data is created. Bagging and Boosting are two common methods for generating such systems.

Alternatively, decision optimization method focuses on the design of the combination rule. It assumes a given set of already trained classifiers and tend to find an optimal combination method. In addition to this categorization, discriminators fusion and selection is used to categorize the way in which the components of MCS are combined [71]. Selection scheme was further subdivided into static and dynamic structure and also the fusion scheme into fixed and trained methods. Sharkey [91] also made the same distinction using different terms, competitive and cooperative instead of selection/fusion. Duin [33] provides an introduction into combination schemes and makes distinction between fixed and trained methods.

Auda and Kamel [5] provided a survey of the Modular Neural Networks (MNN) design. In this study, different motivations for the construction of MNN were discussed and three general steps in the design of MNN were identified. These steps were task decomposition, training, and multi module decision-making. Furthermore, decoupled, cooperative, and competitive decision-making were distinguished and discussed. In this study, the advantages and disadvantages of the surveyed methods are pointed out, in addition to some recommendations for future designs.

MCS can be categorized based on their dependence on the data [57]. Kamel and Wanas categorized data dependency into implicit and explicit. Fuzzy integral, weighted averaging, boosting, modular approaches, and stacked generalization methods are identified as implicit data dependent approaches. While dynamic classifier selection methods, HME (hierarchical mixture of experts), and feature based decision [55, 104, 108] are explicitly data dependent techniques. In the implicit data dependent approaches, the performance of each classifier is considered based on all of the data space or in comparison to each other. On the other hand, explicit data dependency is recognized by the local superiority of certain classifiers.

MCS has also been categorized based on the method of mapping between the input and output of the fusion module. This mapping may be linear or non-linear. Linear combinations are the simplest approaches, in which a weighting factor is assigned to the output of each expert being combined. This weighting factor can be specific to the input pattern, or can be specific to the expert. Weighted average and fuzzy integrals are among the linear combination methods. Non-linear methods include approaches such as, majority or maximum votes. The feature-based decision fusion approach [104], stacked generalization [107], or rank based methods [52], which involve a more complex mapping of the input, also use a nonlinear mapping

in the combining method.

Base classifiers arrangement can be viewed as another categorization scheme, in which two basic categories are: serial and parallel architecture [68]. The parallel architecture consists of a set of classifiers that are consulted in parallel. The decision of the various experts are combined by an aggregation function. On the other hand, the serial architecture consists of a set of classifiers arranged in series, or in tandem. This architecture is designed to deal with situations where the different experts have a ternary decision scheme.

Lam [74] discussed the combinations from an entirely different perspective. Their proposed categorization is based on the topologies and structures used for implementation of the combination methods. These topologies are classified as multiple, conditional, hierarchical, and hybrid. Construction of ensemble in the machine learning has been reviewed by Dietterich [28]. Construction methods applied to different learning algorithms are classified as Bayesian voting, manipulating the training patterns, manipulating the features, manipulating the output targets, and injecting randomness.

2.7 Linear and Quadratic Classifiers

Statistical classifiers are popular techniques in the field of pattern recognition. Linear Discriminant [90] and Quadratic Discriminant classifiers [39] have been widely used in the MCS as the base classifiers [72]. The linear discriminant analysis method consists of a search for a linear combinations of selected variables which provide the best separation between the classes of the problem in hand. These different combinations are called discriminant functions. The linear discriminant function F for true mean vector μ and covariance matrix Σ may be obtained by

$$F_i(x) = (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i). \tag{2.16}$$

where i is a class or group and x is unknown pattern. In some cases, the linear separation of patterns in the feature space is not adequate and a nonlinear separation is needed. Quadratic classifier is based on the normal class-conditional probability

densities.

$$F_i(x) = ln \mid \Sigma_i \mid +(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) - 2lnp_i. \tag{2.17}$$

where p_i is a prior probability associated with class i.

2.8 Multi-Layer Perceptron

Multi-layer perceptron (backpropagation) is the a popular supervised neural network that is based on the error correcting method. This network has successfully been used in many different problems. Given enough training data, appropriate initial conditions and architecture, multi-layer perceptron has been shown to be capable of learning the mapping of any function to satisfactory accuracy [48]. Neural network classifiers have been extensively used in the area of combining classifiers. Many data decomposition techniques, such as boosting and bagging, have demonstrated their best performances using multi-layer perceptron.

2.9 Fuzzy C-Means Clustering

Fuzzy c-means clustering is a popular clustering technique. The origins of fuzzy c-means can be found in Bezdek *et al.* [11]. Fuzzy c-means determines the class prototypes for an existing data set and a specified number of classes. Each of the so-called cluster centers represents the typical object for one class. The Fuzzy c-means algorithm assigns a classification of 0 to 1 between each object to be classified and each class. The fuzzy c-means algorithm is based on minimizing the following objective function:

$$J_q(U,V) = \sum_{j=1}^{n}\sum_{i=1}^{k}(u_{ij})^q d^2(X_j, V_i) \tag{2.18}$$

where V is a set of k centroides and $q > 1$. d is any inner product metric (distance between X_j and V_i), U is fuzzy k-partitioned data set, u_{ij} is the degree of membership of X_j in the ith class, X_j is the jth m-dimensional feature vector and V_i is the centroid of ith cluster. k is number of clusters and n is number of data patterns. V_i

and u_{ij} are calculated as following:

$$V_i = \frac{\sum_{j=1}^{n}(u_{ij})^q X_j}{\sum_{j=1}^{n}(u_{ij})^q} \tag{2.19}$$

and

$$u_{ij} = \frac{[\frac{1}{d^2(X_j,V_i)}]^{(1/q-1)}}{\sum_{h=1}^{k}[\frac{1}{d^2(X_j,V_h)}]^{(1/q-1)}} \tag{2.20}$$

2.10 Measure of Entropy

Entropy is the basic concept of information theory. The entropy of a random variable can be interpreted as the degree of information that the variables provide. Entropy permits assessment of the partial or total information content of a random variable or event. Entropy is denoted by

$$E = -\sum^{x} p(a)\log p(a) \tag{2.21}$$

where $p(a)$ is the probability of occurrence of a. The more unpredictable and unstructured (random) the variable is, the larger its entropy will be. This measure evaluates the information contained in a single random event. The mutual information [22] I provided by a pattern $a_j \in \mathbb{R}^n$, $j = 1, 2, ..., n$, about class y_i, $i \in \Omega$, is calculated as

$$I(y_i; a_j) = \log \frac{p(y_i|a_j)}{p(y_i)}. \tag{2.22}$$

2.11 Summary

In this chapter, a review of relevant MCS design strategies and techniques has been presented. We have discussed three design criteria, four MCS construction strategies including data decomposition, architectures, and aggregation approaches, along with several algorithms and architectures. In addition, a brief overview of the proposed categorizations has been provided. In the following chapter, we propose a novel categorization scheme where we study *cooperation* in MCSs. We categorize MCS

techniques and design approaches with respect to the type of cooperation among their components.

Chapter 3

Cooperative Multi-Classifiers

Classifier ensembles has been proposed with the goal of improving the accuracy and reliability of classification results. In the past decade, various methods and architectures have been reported. Most of the efforts have focused on development of novel combining methods that achieve higher or equal accuracy compared to a single classifier. Few others have introduced new training method that result in diverse classifiers. Other aspects of MCSs have also been investigated such as different architectures, new categorizations, and novel applications. A summary of these studies has been discussed in the previous chapter.

Despite all the accomplishments, no attempt has been made to understand and compare combination methods from the *cooperation* perspective. In general, the structure of a system is shaped by the actual relations among its components. As a result, in order to understand a system, it is essential to study interactions among the components of that system on a larger scale. A thorough understanding of the system allows reasoning about the inter-component interactions and provides guidelines for optimization. For example, systematic investigation on statistical characteristics of the training data and its impact on the performance of individual classifiers and the entire system can provide insight into MCS training techniques and achieving better performance. One way of looking at the issue of cooperation is the concept of *sharing*. A clear picture of the behaviour of MCS may emerge by identification of the resources and the analysis of the gains and drawbacks of sharing between the resources. This knowledge will increase our ability to design and implement a system that is more reliable and flexible for different problems.

The main aim of this research is to have an in-depth investigation and discussion of *sharing* in MCS.

Given the lack of comprehensive studies of different combination schemes from the *sharing* point of view, the objectives of this research are: *1)* to develop an initial vocabulary and a set of terms with which MCS can be described and categorized, *2)* to advance schemes that motivate comparison of different types of structures, *3)* to obtain insights into how a new system can be constructed, and finally *4)* to develop a new MCS based on the findings in the initial experiments.

In the following sections, first, some definitions and terminologies are presented. This will be followed by promoting a new categorization on the MCS based on the new definitions. In each category, some of the ensemble methods, that are used throughout this study, are discussed in detail.

3.1 Definitions

- **System** S**:** A *system* is considered to be any entity that maps inputs to outputs. A system may be composed of subsystems. A *Multiple Classifier System* is a system in which converts a set of input patterns into a set of predictions. The MCS are composed of a set of classifiers $C = (C_1, C_2, ..., C_L)$ where each classifier, C_i, is a system itself.

- **Resource** R**:** A multiple classifier system consists of a number of resources. These resources can be listed as follows:

  - *Input feature:* an attribute of the data which is denoted by x. Multiple variables are denoted by x_i, $x \in \mathbb{R}$, $i = 1, 2, ..., m$.
  - *Input pattern:* a set of feature variables forms an input pattern $X = (x_1, x_2, ..., x_m)$.
  - *Class (target):* each input pattern is associated with a class, y_i, $i = 1, ..., c$. A class has a set of possible values $\Omega = (1, 2, ..., c)$.
  - *Training data:* a set of feature variables along with their associated class forms a training pattern. A combination of n training patterns form the training data $X = (X_1, X_2, ..., X_n)$, $X \in \mathbb{R}^n$.

– *Classifier output:* prediction of a classifier for a given input pattern X corresponding to its class target y_j, $j \in \Omega$, can be denoted as $c_{ji} = p(y_j|X)$, where $i = 1, ..., L$. c_{ji} can either be restricted within the interval of [0,1], soft label, or belong to the set of $\{0, 1\}$, crisp label.

- **Sharing:** Resource R of a system is being shared if, and only if, R is used by more than one component of the system.
- **Level of Sharing:** Sharing in the MCS may take place at four different levels of Decision, Architecture, Feature and Training.

3.2 Levels of Sharing

3.2.1 Decision Level

Let the prediction of classifiers C_i, $i = 1, ..., L$ for a given input pattern X be denoted as a vector $C_i(x) = [c_{1i}(x), ..., c_{ki}(x)]$. Resources ($R$) at this level are the base classifier' outputs c_{ji}. Then, sharing at this level may be distinguished by combining method ($\mathcal{G}$) denoted as

$$Q(X) = \mathcal{G}(C_i(X)) \tag{3.1}$$

that makes the final decision

- according to individual classifiers prediction. These methods make use of the fact that the outputs of the base classifiers are a clear interpretation of the final decision (e.g. majority vote, maximum, minimum).
- based on a function that processes the predictions of classifiers into a final decision (e.g. averaging, decision template, and trainable aggregation methods).

3.2.2 Architecture Level

The next level is sharing at the architecture level. Sharing at this level may be distinguished by:

- Using homogeneous classifiers: many parallel architecture designs use homogenous classifiers (Section 2.3.1).
- Using information that is passed from one layer to the next: stacked generalization and cascade generalization ([107] and [42]) are two popular methods in which classifiers are arranged in sequence and pass information from one layer to the next. In these methods, the first classifier generates possible recognition indices about the classes and passes the information to the next classifier. Hierarchical mixture of experts ([55] and [40]) are another example of this scheme where a prior knowledge of behaviour of the classifiers is passed through the layers. Each classifier at the lowest level is an expert in a local area of the feature space. A prior knowledge of the behaviour of the classifiers is passed through the layers and a gating method is used to decide which classifier should assign the label.
- Using classifiers that work in parallel but exchange information about their individual task. Auda and Kamel [6] developed a novel modular architecture, Cooperative Modular Neural Networks (CMNN) in which classifiers are utilized to cooperate by forcing them to exchange information about each others task. Unlike other modular approaches in which classifiers focus on predicting their own task, in CMNN, classifiers are forced to learn about other modules tasks at the same time. In this approach, classifiers are trained with a subset of data that consists of subset of patterns belonging to predetermined set of classes. If the unknown pattern does not fit in the range of the classifiers responsibility, the classifier will point out the other responsible classifiers with the output unit that is in charge of doing that.

3.2.3 Feature Level

The next potential level of sharing occurs in the input feature space. Different patterns of generalization can be achieved by modification of the feature space. For a given high dimensional data set, there are various types of features which can be extracted from the data for the same classification task. Feature partitioning has been shown to be an effective way to produce diverse classifiers. MCS construction methods on this level may share feature instances instead of using totally disjoint instances.

Random Subspace Method (RSM) is a combining technique proposed by Ho [51]. In this method, features are randomly divided into subspaces and classifiers are built on different training feature sets. Other techniques designed for feature space modification are genetic algorithm, input decimation, and different selection schemes [73, 86, 100, 106]. In the input decimation techniques, each classifier is trained on a subset of input features that correspond to a specific class. This method aims at reducing the dimensionality of the data as well as correlation between classifiers. Forward and backward selections methods add/delete one feature to/from the set at a time in order to reduce correlation among classifiers. Genetic algorithm performs a feature selection procedure in which the classifiers produce the highest accuracy as a combination and individually. In other words, individual fitness and group fitness are applied simultaneously.

Let each training object X_i, $i = 1, ...n$ in the training data X be a m dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{im})$, described by m features. The features can be divided to N different partitions. The modified training set $X^N = (X_1^N, X_2^N, ..., X_n^N)$ consists of r-dimensional training objects $X_i^N = (x_{i1}^N, x_{i2}^N, ..., x_{ir}^N)$ where $r < m$ and $i = 1, 2, ..., n$. The N partitions may be disjoint or overlapped. Several feature modification approaches have been proposed in the field of MCS.

3.2.4 Training Level

This is the most basic level of sharing which occurs in the training process. Classifier combining methods in this level may share:

- Training patterns: most of the current MCS in the literature concentrate on the use of shared or identical training data. More training data may result in higher accuracy for individual classifiers. However, many popular techniques in MCS have been designed based on division or modification of the training data. There are several reasoning behind these approaches: *1)* to achieve different patterns of generalization, *2)* to be able to work with large datasets (time and cost), *3)* to obtain classifiers that are experts in specific part of the data space. Classifiers that are trained on different training sets have a higher chance to display different patterns of generalization than the ones that are trained on the same data. Even in the case of neural network classifiers, initial

conditions or different algorithms may not be able to induce different patterns of generalization.

Training data modification is a popular method for constructing MCS. Bagging is, perhaps, the most well-known sampling approach. The bagging algorithm, relies on varying the data, through the bootstrap sampling procedure [17]. Each training set is generated by randomly selecting, with replacement, n examples. Many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set. Bootstrapping is based on random sampling with replacement. Bootstrap replicates $X = (X_1^b, X_2^b, ..., X_n^b)$ of the training set $X = (X_1, X_2, ..., X_n)$ and build a classifier on each of them. In bagging, bootstrapping and aggregating techniques are implemented in the following way.

1. Repeat for $b = 1, ..., L$.
   (a) Take a bootstrap replicate X^b of the training dataset X.
   (b) Construct a classifier $C_b(x)$ on X_b.
2. Combine classifiers $C_b(x), b = 1, ..., L$, by the simple Majority vote to a final decision rule $\beta(x) = \begin{cases} +1 \text{ if } \sum_b sgn(c_b(x)) > 0 \\ \text{-1 otherwise} \end{cases}$

Rvote, a variation of pasting small votes [15], is similar to bagging and has been designed for large datasets. In Rvote, a large dataset is partitioned into smaller subsets, referred to as bites, each of which is used to train a classifier in a different processor. Rvote requires creations of very small size bags, randomly, and is a fast and simple approach.

The general principle of divide-and-conquer is based on partitioning the training data. Frosyniotis *et al.* [38] proposed a divide-and-conquer method in which training data is partitioned into overlapping clusters using fuzzy c-means and greedy-EM. Classifiers are assigned to overlapping regions from the beginning, and are trained with data sets that are representative of the regions that they are assigned to.

The idea of partitioning the training data using clustering has also been applied to building classifier selection methods, where classifiers are trained on the

same training set and are subsequently assigned to different regions according to their accuracy [71]. Classifier selection based on hard boundary points has been proposed by Lipnickas [77]. In this method, the entire data space is partitioned into several regions using a clustering technique. Then, hard boundary points are selected as reference points. Hard boundary points are generated based on the misclassified patterns by base classifiers. The accuracy of all the individual classifiers in each region is estimated using the reference points, and the classifier with the highest accuracy is assigned to the region.

Prototype Reduction Scheme (PRS) [58] has also been applied to combine a pool of individual classifier. Prototype Reduction Scheme is a way of reducing the number of training vectors while performing as well as or similar to the original data. Various PRS methods have been proposed including condensed nearest neighbor, random selection, genetic algorithm, and search techniques. Other techniques designed to produce different training sets include non-linear transformations, injection of noise, and data from different sensors [94]. These methods involve changing the patterns in a training set such that classifiers approximate different functions for the same problem.

- Training information, instead of being trained individually and independently: Wanas *et al.* [104] proposed an evolving training algorithm where classifiers share the outcome of an aggregation method in order to perform an adaptive training process. The algorithm can determine if further training is needed by evaluating the result in the aggregation layer. This method directs the training of each of the classifiers by partitioning the training patterns based on the performance of the ensemble. This way, classifiers share the aggregation results which helps to retrain and improve the quality of the training. Liu and Yao [79] proposed a cooperative ensemble learning system (CELS) in which classifiers are encouraged to learn different parts or aspects of the training data. This method emphasizes interaction among individual base classifiers and utilizes an unsupervised penalty term in the error function to produce biased neural networks that are negatively correlated.

  In the Boosting algorithm [37], successive classifiers are trained on input-output pairs that have been filtered by previous classifiers. The sampling of training data patterns is in such a way that misclassified patterns have a higher likelihood to be selected for the next step. AdaBoost algorithm works

as follows:

1. Initialize the parameters, weights w_i $i = 1...N$, ensemble $C = 0$, size of ensemble L
2. Repeat this step L times
   - Take a sample S^b from X and train classifier C_b
   - Calculate the weighted ensemble error at step k

$$\epsilon_b = \sum_{j=1}^{N} w_j^b l_b^j \quad l_b^j = \begin{cases} 1 \text{ if x is misclassified} \\ 0 \text{ otherwise} \end{cases} \tag{3.2}$$

   - Calculate

$$\beta_b = \frac{\epsilon_b}{1 - \epsilon_b} \tag{3.3}$$

   - Update the weights after classification

$$w_j^{b+1} = \frac{w_j^b \beta_b^{(1-l_b^j)})}{\sum_{i=1}^{N} w_i^b \beta_b^{(1-l_b^i)}} \quad j = 1, ..., N. \tag{3.4}$$

3. Calculate the output for class y_j

$$Q_j(X) = \sum_{j=1}^{k} ln(\frac{1}{\beta_b}) \tag{3.5}$$

4. Select the class with maximum support.

Ivote is another variation of pasting small vote which has been designed for large databases [15]. Ivote is similar to Adaboosting in a sense that the training data (bite) of each subsequent classifier relies on the performance of previous classifier. The sampling of training patterns rely on the out-of-bag error, where the classifiers are tested on the patterns that they have not been trained on. If a pattern in the training subset is misclassified by a majority vote of the out-of-bag classifiers, it will be selected for the next training subset. If not, then it

is rejected with the probability $\frac{e_t}{1-e_t}$, where e_t is the error at step t. Cascade generalization is another example of methods that share training information.

The first step in the construction of classifier ensembles, or any classification system, is to train the base classifiers. In some studies, this is considered the most important step since generalization ability of the ensemble highly depends on the performance of the individual classifiers. For example, some approaches have focused on construction of diverse base classifiers [67, 69], while few others were concerned with generation of a large pool of base classifiers with different characteristic, e.g. learning method [92]. Despite all the effort, the issue of cooperation at this level has been overlooked. This study focuses on the training level and various aspects of cooperation at this level. In order to achieve reliable assessment of the cooperation at training level, we minimized the impact of other external parameters including cooperation at other three levels.

3.3 Summary

The goal of this chapter was to highlight the needs to investigate cooperation among MCS resources. We identified one of the principal types of cooperation, sharing, and categorized ensemble techniques into four levels of sharing including aggregation, architecture, feature and training. It is important to note that some of the methods mentioned above cooperate at different levels. For example, in boosting techniques, sharing is at the architecture level as well as the decision level. The feature-based decision fusion approach [104] is another example where sharing happens at all levels. Cooperation in MCS, as defined in this chapter, is a new concept that has not been studied before. In this thesis, we focus on training level cooperation and investigate the advantages and disadvantages of sharing data and training information. The following chapters present the details of methodologies and findings at the training level.

Chapter 4

Cooperation at Training Level

When the use of multiple classifiers is considered, different strategies can be adopted to train the base classifiers. Identical training sets can be used by all the base classifiers, or modified subsets may be generated for training. The main objective of training data partitioning is to improve the training of MCSs, and their performance. This objective can be achieved in two ways. One approach may be achieved by directly sharing the training information of the base classifiers or by the aggregation method in the training data partitioning process. While in the other approach, the training data partitioning is a pre-classification process, where partitions are generated prior to training of the individual classifiers and without sharing training information. With this analogy, we categorize MCSs training methods into two groups: methods that are independent of classifiers and methods that share training information. This chapter provides a detailed analysis of these two categories.

Despite research in the area and a significant number of empirical studies, the advantages offered by different MCS training methods are still not clear. In addition, no analysis or measurement of training methods effectiveness has been discussed in the literature. The main objective of this work is to shed some light on these and closely related issues. The present study introduces a new direction for the training of multiple classifier systems.

The primary contribution of this work is the development of novel data partitioning evaluation measures. The intent is to investigate how training patterns are utilized in MCSs and to investigate the advantages and disadvantages of sharing

training information. We propose two categories of evaluation measures: class-based and feature-based. The class-based measures only consider the distribution of classes among training partitions. Feature-based measures evaluate training subsets with respect to classes (or training partitions) correlation. Feature-based and class-based evaluation measures enabled us to estimate the degree and type of information provided by a set of disjoint and overlapped partitions for the training of MCSs. Moreover, we empirically assessed both types of measures using several benchmark databases. The findings allowed us to reason about interactions between training partitions and MCS performance, and provided guidelines for the generation of sub-optimal partitions. We proposed two approaches for generation and selection of sub-optimal training partitions. One approach is a pre-classification process where training information is not utilized in selection of sub-optimal training partitions, while in the other approach, training information is used to select sub-optimal training partitions. We compare these two approaches as well as other existing MCS training approaches using several benchmark datasets.

4.1 Cooperation without Sharing Training Information

In some of multiple classifier training methods, training subsets are partitioned before individual classifiers are trained (Figure 4.1). Bagging [17] is a well-recognized training method that partitions data before constructing the base classifiers. Bagging has been shown to be very effective for unstable classifiers, since small changes in the training data produces a large change in the output. Another method that helps force diversity, is by using k-fold cross validation. A more deterministic partitioning approach has been proposed in [38], where original training data is partitioned using clustering techniques. Each cluster is then used to train a new classifier. Other training data modification methods belonging to this category are random forests, pasting votes, and static classifier selection [14, 29].

Ensemble methods that do not share training information are less expensive and the selection of the architecture and base classifiers does not have to be made in advance. In addition, classifiers will not be engaged in partitioning and selection procedures which result in lower computational cost. However, the difficulty with these approaches is that all the parameters and their interactions should be considered. Sophisticated functions are needed to evaluate all the parameters and their

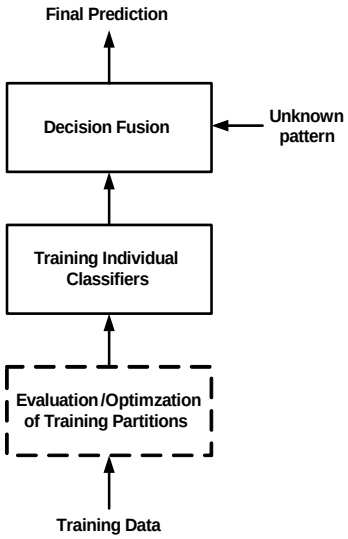


Figure 4.1: Cooperation without Sharing Training Information

impact on the multiple classifiers performance.

When classifiers do not share training information, selection and optimization of the training partitions become a pre-classification process and cannot be accomplished automatically. Therefore, appropriate tools and measures are required to generate and optimize training subsets. Currently, only a few of the existing ensemble methods optimize their partitioning processes, e.g. divide-and-conquer method [38]. Most of the popular training methods operate using random partitioning, e.g. bagging. Instead of randomly distributing data patterns among training subsets, prior knowledge about the training partitions, data attributes (e.g. distribution) and their impact on the classification performance is needed to obtain sub-optimal partitions and boost training of base classifiers. To achieve this goal, we have proposed several measures with which interdependency and correlation of the training partitions were estimated. By applying these measures, we examined the impact of the type and degree of information provided by the data on the training of classifier ensembles and the resultant accuracy.

4.2 Evaluation Measures

A fundamental aspect of data partitioning, that remains overlooked by random selection, is the use of information in the data itself. With the use of properly selected measures, we will gain better insight about the data which leads to the development of more sophisticated methods to explore the correlation between cooperation and accuracy.

We aimed to develop several measures to examine and rank various types of training partitioning based on MCS performance. As previously discussed, we were concerned with pre-classification partitioning methods. Therefore, it was necessary to take into account many parameters: the number of classes, number of data partitions, diversity within each partition and total diversity of a set of training subsets, interdependency among these subsets, information shared by them, the amount of overlap among partitions, correlation between classifiers, the type of classifiers, and decision fusion methods were examples of such parameters. Since examining all these parameters and their impact is beyond the scope of this study, we narrowed our focus to class diversity and the data/information correlation distributed across training partitions. To draw reliable conclusions, we had to investigate one factor at a time, and hold constant the remaining parameters which include: the type of classifiers, number of classifiers, and size of partitions. We conducted the analysis in two stages, one for disjoint partitions (0% overlap) and one for overlapped partitions.

In the following, we use the following terminologies and rules. Partitioning of the training data means dividing the data space into m subsets. Every partition is associated to one classifier. Every partitioning scheme covers the whole training space. Since we are concerned with disjoint partitions, for every training pattern, there exists one and only one partition to which this pattern belongs to. In the present work, the number of partitions is assumed to be equal to the number of base classifiers. Every set of partitions is used to train a multi-classifier system, in such a way that each partition trains one and only one base classifier. A set of partitions consists of:

- a pattern x_{ij} that belongs to class i in partition j
- k_i^j, the i^{th} class in the j^{th} partition

- P_j, the j^{th} partition
- k_i, class i in the training data
- the whole training data which is divided into m partitions. In this study, we refer to each set of partitions as a *solution*.

We employed a procedure to generate a set of sub-optimal training partitions. This procedure had two essential components, *(i)* a large pool of solutions (or sets of training partitions), and *(ii)* a set of evaluation measures by which we could rank all the solutions. In this study, we examined the impact of the two parameters: class diversity and correlation. For these two parameters, we introduced several measures and grouped them into two main categories of feature-based and class-based. For the class-based measures (or class diversity), information provided by the feature space is not considered and only class labels of the patterns are incorporated into the measure. While for feature-based methods, the measures are calculated with respect to feature attributes (correlation).

Feature-based and class-based measures can be constructed differently, given the class and location of data patterns x_{ij}. Subsequent to the partition of the training data, data patterns may locate:

- in one partition *intra-partition*, among the members of one class *intra-class*
- in one partition *intra-partition*, among member of two or more classes *inter-class*
- among two or more partitions *inter-partition*, among members of one class *intra-class*
- among two or more partitions *inter-partition*, and all the classes *inter-class*.

These conditions are summarized in Table 4.1.

4.2.1 Class-based Measures

While *classifier diversity* has been extensively discussed in the MCS community, the issue of *class diversity* (among training partitions) has been largely overlooked.

Table 4.1: Four different conditions for location of data patterns

	same partition	**different partitions**
same class	intra-class intra-partition	intra-class inter-partition
different classes	inter-class intra-partition	inter-class inter-partition

In this study, we plan to address this issue through a systematic and analytical procedure. We define a partition with a high degree of diversity as a subset in which all data patterns belong to one class. Obviously, a less diverse partition contains a low information content. A partition with a higher degree of diversity is more informative and represents a larger area of the space associated with the training data. The class diversity of a partition or a set of partitions can be estimated with different techniques, including Berger-Parker, entropy, and standard deviation.

The following notations are used in this section: n_{ij} is the size of class i in partition j, M_j is the size of partition j, m is the number of partitions, N is the size of training data, k is number of classes, and N_i is the size of class i in the training data.

- **Berger-Parker Measure:** One of the simplest diversity measures is the Berger-Parker index [9]. This measure is mostly used for estimating biological diversity,

$$b_{p_j} = max(p_{ij}), \quad p_{ij} = \frac{n_{ij}}{M_j}. \tag{4.1}$$

  The overall measure of diversity for a solution (m partitions) can be calculated as:

$$b_{overall} = \sum_{j=1}^{m} \frac{M_j}{N} b_{p_j}. \tag{4.2}$$

- **Entropy:** The Shannon's diversity index, mostly referred to as entropy, is applied to measure the information content of the training partitions. Generally, the use of such information theoretic measures are very popular in partitioning, and sub-space clustering [20]. According to the information theory, higher probability for a class in a partition, or less diverse partitions, represents a lower entropy and, consequently, a lower information content. Two different strategies can be employed to estimate entropy: within one partition among different classes (inter-class, intra-partition) or within one class distributed among several partitions (intra-class, inter-partition). Intra-partition entropy,

h_{p_j}, estimates the information content or diversity of the j^{th} partition

$$h_{p_j} = -\sum_{i=1}^{k} p_{ij} log(p_{ij}), \quad p_{ij} = \frac{n_{ij}}{M_j}. \tag{4.3}$$

Inter-partition, intra-class diversity can be estimated by

$$h_{k_i} = -\sum_{j=1}^{m} p_{ij} log(p_{ij}), \quad p_{ij} = \frac{n_{ij}}{N_i}. \tag{4.4}$$

For both entropies, the overall measure of diversity can be calculated by

$$H_p = \sum_{j=1}^{m} h_{p_j}, \quad H_k = \sum_{i=1}^{k} h_{k_i}. \tag{4.5}$$

- **Standard Deviation:** Another measure of diversity is standard deviation, which, again, can be viewed in both inter-class/intra-partition and intra-class/inter-partition level.

$$s_{p_j} = (\sum_{i=1}^{k} |n_{ij} - \frac{M_j}{k}|)^{1/2}, \quad s_{c_i} = (\sum_{j=1}^{m} |n_{ij} - \frac{N_i}{m}|)^{1/2}. \tag{4.6}$$

The overall class diversity for a solution based on standard variation is calculated by

$$S_p = \sum_{j=1}^{m} s_{p_j}, \quad S_k = \sum_{i=1}^{k} s_{k_i}. \tag{4.7}$$

4.2.2 Feature-Based Measures

Unlike class-based measures, feature-based measures make use of feature space to qualitatively estimate similarities/differences among two or more partitions in a solution set. In this section, we introduce models which can be applied to estimate distances of two or more objects. By object, we refer to patterns, classes, or partitions. Obviously, to compute the distance between two data patterns, the use of feature attributes is inevitable. Therefore, all the proposed measures of this type are referred to as feature-based. We employed Euclidean, Bhattacharyya, Mahalanobis,

and Patrick-Fisher distance measures to estimate the similarity (difference). The lack of studies on data distribution and MCS accuracy has prompted us to examine several distance measures.

- **Euclidean Distance:** The most popular and simple distance measure is the Euclidean distance [32]. Defining a metric space, the Euclidean measure is denoted as:

$$d_E = \sqrt{(\bar{x} - \bar{y})^T(\bar{x} - \bar{y})} \tag{4.8}$$

  where $\bar{x}$ and $\bar{y}$ are two sample means selected from a predefined data space. Each vector has m features, representing the dimension of the feature space.

- **Mahalanobis Distance:** One drawback of the Euclidean distance is its sensitivity to features that have different scales and variances. Mahalanobis distance overcomes this problem by expressing the distance between two vectors in units of their covariances. The Mahalanobis distance between two sample means $\bar{x}$ and $\bar{y}$ is represented as:

$$d_M = \sqrt{(\bar{x} - \bar{y})^T \Sigma^{-1} (\bar{x} - \bar{y})} \tag{4.9}$$

  where Σ is the covariance matrix of two samples. Σ is defined as

$$\Sigma = \frac{(n_x - 1)\Sigma_x + (n_y - 1)\Sigma_y}{n_x + n_y - 2} \tag{4.10}$$

  where, n_x and n_y are the sizes of matrixes Σ_x and Σ_y are the covariance matrices of x and y accordingly. Mahalanobis distance is the same as Euclidean Distance if Σ is the identity matrix.

- **Bhattacharyya Distance:** Similar to Mahalanobis, Bhattacharyya distance measure is widely used as a measure of class separability, because of its precision and relation to the Bayes error [32].

$$d_B = \frac{1}{8}\sqrt{(\bar{x} - \bar{y})^T (\frac{\Sigma_x + \Sigma_y}{2})^{-1} (\bar{x} - \bar{y})} + \frac{1}{2} ln(\frac{|(\Sigma_x + \Sigma_y)/2|}{\sqrt{|\Sigma_x| . |\Sigma_y|}}). \tag{4.11}$$

- **Patrick-Fisher Distance:** Patrick-Fisher measure [32] was also considered

to measure correlation between classes and partitions.

$$d_{PF} = (\frac{\Sigma_x + \Sigma_y}{2})^{-1}(\bar{x} - \bar{y}) \tag{4.12}$$

The aforementioned measures calculate the distance between two vectors. In the present study, we calculated distances between two vectors from the following categories (see Table 4.1):

- identical classes in one partition (Figure 4.2: d_{c1})
- different classes in one partition (Figure 4.2: $d_{(c1,c2)}$)
- identical classes in two different partitions (Figure 4.2: $d^{c2}_{(P1,P2)}$)
- two different classes in two different partitions (Figure 4.2: $d_{(P1,P2)}$).

Figure 4.2 depicts the possible scenarios between two partitions P1 and P2, where μ_1 and μ_2 are the means of these partitions respectively.

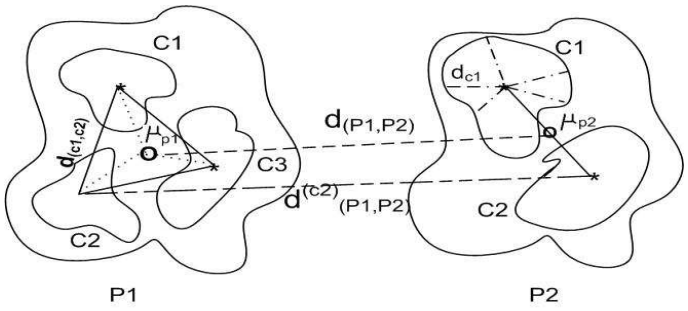


Figure 4.2: Feature-Based Evaluation Measures: d_{c1} Intra-class distance/Inter-partition, $d_{(c1,c2)}$ Inter-class/Intra-partition distance, $d^{c2}_{(P1,P2)}$ Intra-class/Inter-partition distance, and $d_{(P1-P2)}$ Inter-class/Inter-partition distance

These proposed distance measures were each designed to satisfy a different objective. For example, intra-class/intra-partition distance computes and evaluates class compactness in a partition. On the other hand, the intra-class/inter-partition distance formulates a criterion for the compactness of a class distributed across a set of partitions. Inter-class/intra-partition distance assesses class separability in

each training subset. While inter-class/inter-partition distance evaluates similarity between two or more training subsets.

Using the weighted average of all pairwise distances among two or more objects, a quantitative value was derived for a set of partitions. The pair-group centroid, center of the objects, was used because of its simplicity. This simplification reduced the number of operations. All the measures were weighted to take into consideration the difference in the size of objects.

The following notations are used in this section: μ_i^j represents the mean (center) of the i^{th} class in the j^{th} partition, μ_i represents the mean of the j^{th} class, μ^j is the mean of the j^{th} partition, ν is the mean of the training data, $x_{k,i}^j$ is considered the k^{th} sample of i^{th} class in the j^{th} partition, and $d(x, y)$ represents the distance between x and y data patterns. In addition, D_i^j is the correlation among the i^{th} class in partition j, D_i is the correlation among class j, and finally, D^j is the correlation of the th partition. Proposed evaluation measures are discussed bellow, and summarized in Table 4.2.

Intra-class, Intra-partition Distance: In this category, distances among patterns from the same class are calculated. This measure can also be referred to as density. This distance is calculated as

$$D = \sum_{j=1}^{m} \sum_{i=1}^{k} p_{k_i} \sum_{k=1}^{n_i j} d(x_{k,i}^j, \mu_i^j), \quad p_{k_i} = \frac{n_{ij}}{M_j}. \tag{4.13}$$

Inter-class, Intra-partition Distance: Here, correlation among two or more classes in a partition is considered. Instead of a class centroid, we can also consider mean of the partition as follows

$$D = \sum_{j=1}^{m} \sum_{i=1}^{k} p_{k_i} d(\mu^j, \mu_i^j), \quad p_{k_i} = \frac{n_{ij}}{M_j}. \tag{4.14}$$

Intra-class, Inter-partition Distance: The distribution of one class among m partitions is the basis of this evaluation. It is important to note that instead of the mean, μ, actual patterns can be considered as well

$$D = \sum_{i=1}^{k} \sum_{j=1}^{m} p_{p_j} d(\mu_i, \mu_i^j), \quad p_{p_j} = \frac{n_{ij}}{N_i}. \tag{4.15}$$

Table 4.2: Summary of the Measures

Data Patterns Location/Measure	*Class-based*	*Feature-based*
intra-class/intra-partition	–	$D = \sum_{j=1}^{m} \sum_{i=1}^{k} p_{k_i} \sum_{k=1}^{n_{ij}} d(x_{k,i}^{j}, \mu_i^j)$
intra-class/inter-partition	$s_{k_i} = (\sum_{j=1}^{m} \lvert n_{ij} - \frac{N_i}{m} \rvert)^{1/2}$ $h_{k_i} = -\sum_{j=1}^{m} p_{ij} log(p_{ij})$	$D = \sum_{i=1}^{k} \sum_{j=1}^{m} p_{p_j} d(\mu_i, \mu_i^j)$
inter-class/intra-partition	$s_{p_j} = (\sum_{i=1}^{k} \lvert n_{ij} - \frac{M_j}{k} \rvert)^{1/2}$ $h_{p_j} = -\sum_{i=1}^{k} p_{ij} log(p_{ij})$ $b_{overall} = \sum_{j=1}^{m} \frac{M_j}{N} b_{p_j}$	$D = \sum_{j=1}^{m} \sum_{i=1}^{k} p_{k_i} d(\mu^j, \mu_i^j)$
inter-class/inter-partition	–	$D = \sum_{j=1}^{m} \sum_{i=1}^{k} d(\nu_i^j, \mu_i^j)$

Inter-class, Inter-partition Distance: The overall correlation of a solution of partitions is calculated by considering the distances of the centers of all partitions from the overall mean

$$D = \sum_{j=1}^{m} d(\nu, \mu^j). \tag{4.16}$$

4.3 Description of Datasets

This section provides an overview of the datasets that were used in this study. We gathered 13 benchmark databases from different sources from different sources UCI [12], and ELENA project [35]. These data sets were collected because of their diverse statistical and distributional characteristics. To provide more information about the data sets, Table 4.3 summarizes some of their properties. We applied a coefficient measure to highlight data complexity in terms of data size, dimensionality and the number of classes (Equation 4.17). Column #5 illustrates the results for this complexity measure.

$$data_{coef} = \frac{data \;\; size}{(number \;\; of \;\; features) \;\; . \;\; (number \;\; of \;\; classes)}. \tag{4.17}$$

Class imbalance is another factor that is demonstrated in Table 4.3 in column six, where one represents equal class sizes.

$$imbalance = \frac{smallet \;\; class}{largest \;\; class}. \tag{4.18}$$

Table 4.3: Summary of the Data Sets

Data Sets	# of Classes	# of Features	Size	$data_{coef}$	imbalance	Source
			Low			
Clouds	2	2	5000	1250	0	[35]
Concentric	2	2	2500	625	0.58	[35]
Iris	3	4	150	12.5	0	[12]
Ionosphere	2	34	351	5.61	0.56	[12]
Glass	6	10	214	0.12	0.19	[12]
Wine	3	13	178	0.68	0.61	[35]
Wisconsin Breast Cancer	2	9	699	0.59	[12]	
			Low			
20 Class Gaussian	20	2	2000	50	1	[104]
German	2	20	1000	25	0.43	[12]
Phoneme	2	5	5404	54.04	0.41	[35]
80-D Gaussian	2	80	1000	6.25	1	[71]
Pima Indian Diabetes	2	8	768	48	0.54	[12]
Satimage	6	36	6435	29.79	0.41	[35]
Vehicle	4	18	846	11.75	0.91	[12]
Vowel	11	13	990	6.92	1	[12]

The Self-organizing Map [63] has been applied to all the datasets to highlight complexity of the data in terms of class compactness, isolation or linear class separability in a two dimensional space (see Appendix A.1).

4.4 Disjoint Partitioning

We grouped the experiments into disjoint partitioning and overlapped partitioning. With this distinction, we were able to study cooperation through sharing training patterns in classifier ensembles. This section highlights the results and discussion for the proposed class-based and feature-based measures on disjoint training partitions.

4.4.1 Empirical Assessment of Class-based Measures

We carried out a set of experiments to examine the proposed class-based measures. In these experiments, we first generated a large population of solutions, and then calculated class diversity for each set of training partitions and ranked them according to their impact on MCS performance. This section highlights a summary of the important observations.

An ensemble of Naive Bayesian classifiers [34] was used for empirical evaluation of the proposed measures. For each data set, we used a different number of ensemble members. The selection of ensemble sizes was based on the size and complexity of

the data. The use of disjoint training subsets prohibited us from applying large ensemble sizes. The product of the estimated posterior probabilities by the base classifiers was used as combining rule [68].

Training Solutions Generation Scheme

The goal was to generate partitions with different degrees of class diversity. As a result, an initial set of partitions S_1^i was obtained in such a way that each partition contained only one class. If the number of classes was smaller than the number of partitions, larger classes were distributed among two or more subsets. Furthermore, if the number of classes was larger than the number of partitions, two or more classes were grouped into one training subset, depending on the size of the classes. These sets remained the baseline for the generation of succeeding set of solutions. We referred to this method as decoupled strategy (Figure 4.3).

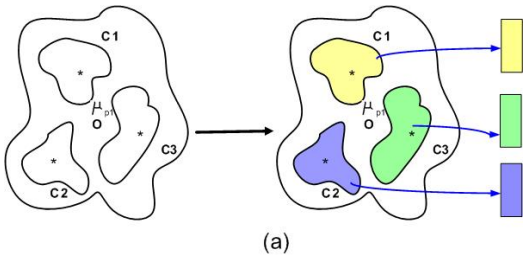


Figure 4.3: *a)* Decoupled Strategy

By randomly selecting a certain amount of patterns from the training data and distributing these among initial subsets, new subsets were generated (Algorithm 1). The size of training partitions remained balanced throughout the experiments.

Subsequent to the generation of a pool of solutions, these solutions were used to train classifier ensembles (Algorithm 2). We applied class-based measures to estimate the degree of information provided by each set. These estimates were used as criteria to rank and evaluate training subsets according to the performance of the constructed ensemble.

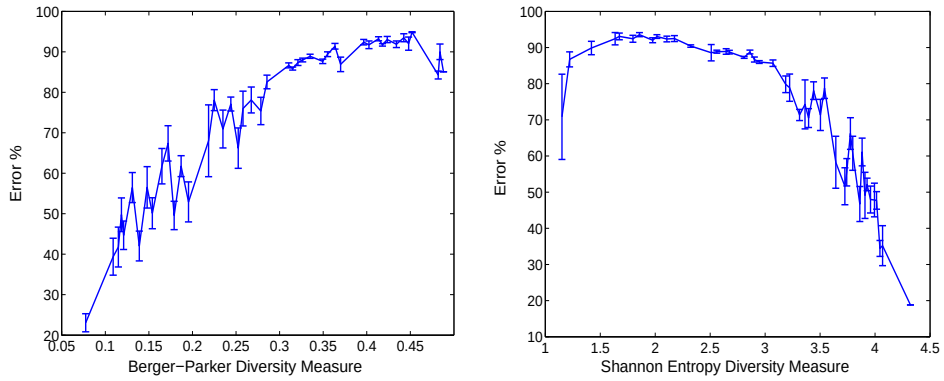
Algorithm 1 *Disjoint Training Partition Generation Scheme*

X_{train}: training data
X_{test}: test data
S_1^i: an initial set of training partitions generated in iteration i
S_t^i: a set of training partitions
M and b : are predefined integer values
r : is a predefined integer between 1 to 10
m: number of partitions

for i=1:b
  generate X_{train}^i and X_{test}^i
  generate $S_1^i = P_1^i, ..., P_m^i$ using decoupled strategy
  initialize $frac\%$ to an integer between 0 to 100
  $t = 2$
  while $frac \leq 100$
    for $n = 1 : M$
      generate X_{random} by randomly selecting $frac\%$ from X_{train}^i
      generate S_t^i by distributing X_{random} among S_1^i subsets
      $t = t+1$
    end for
    $frac = frac + r$
  end while
end for

Algorithm 2 *Evaluation of Training Data Partitions*

construct ensembles on all S_i
calculate the class-based measure for all S_i
estimate mean of ensemble error and the class-based measure for n iterations
rank S_i according to the performance of constructed ensembles



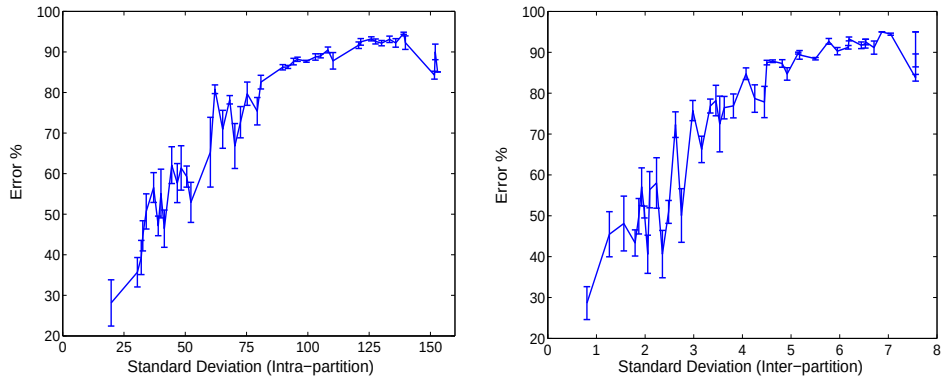
(a) (b)

Figure 4.4: 20-class Gaussian (Disjoint): a) Berger-Parker, b) Shannon Entropy

Results & Discussion

A large number of solutions and MCSs were generated using Algorithms 1 and 2 for class-based related experiments. These experiments were carried on all datasets (see 4.3); however, the results for only 80-D Gaussian and 20-class Gaussian datasets are highlighted, since all of these showed identical behavior. Test error rates, shown on the y-axis, against class diversity, shown on the x-axis, are summarized in Figures 4.4-4.5 and Figures A.9-A.10 in Appendix A.2. The presented results have been averaged over 10 training partitioning iterations, as well as 25 different sets of partitions (solutions). Standard deviation over these error points has been added to the plots.

Overall, Berger-parker, entropy, and standard deviation measures (Equations 4.1-4.7) illustrated that by increasing the degree of class diversity among training partitions, classifier ensemble error decreases. This observation yields an important conclusion; multiple classifiers that are built on partitions containing diverse classes result in a better generalization ability. This is due to the fact that partitions with less diverse classes could not cover the entire information space associated to the training data. By distributing classes among all subsets and increasing the degree of diversity, a better estimate of the posteriori class probabilities is calculated. Increasing the accuracy of individual classifiers resulted in overall improvement in the system.



(a) (b)

Figure 4.5: 20-class Gaussian (Disjoint): a) STD Inter-class/intra-partition b) STD Intra-class/inter-partition

4.4.2 Empirical Assessment of Feature-based Measures

Empirical assessment of feature-based measures were also conducted on all datasets introduced in Section 4.3. In these experiments, a large population of training subsets with different class distributions and distances were generated. These subsets were evaluated according to their impact on MCS performance.

Training Solutions Generation Scheme

In the feature-based set of experiments, the training solutions generation scheme was similar to our class-based experiments (Algo. 1). However, the generation of an initial set of training partitions was completely different. Since we were mainly concerned with data distance (correlation), we had to play with the feature space of the data instead of the class space. We developed two strategies to generate the initial set, which we called Slice and Contour. Our aim in both proposed strategies was to obtain highly uncorrelated training partitions. These sets remained the baseline to generate succeeding sets of solutions.

1. We aimed on initializing highly separable partitions. This objective was achieved by employing a k-means clustering method to cluster each class to m sub-clusters (slice strategy). Then, pairwise distances of each sub-cluster from other sub-clusters located in nonidentical classes were calculated and sorted.

Sub-clusters with the largest distances were located in one partition (Figure 4.6 (a)). This set of partitions has remained the baseline for constructing additional sets of partitions using Algorithm 1.

2. Training sets were generated based on distances of the data patterns from the center of their classes. We first sorted the patterns based on distances. Then, we distributed the sorted patterns in such a way that the first partition contained the largest distanced patterns, the second partition contained the second largest distanced patterns, and so on (Figure 4.6(b)). By randomly selecting training patterns and dividing them across partitions, new sets of solutions were generated (Algorithm 1). It is important to note that, in this method, class distribution was kept fixed throughout the generation procedure.

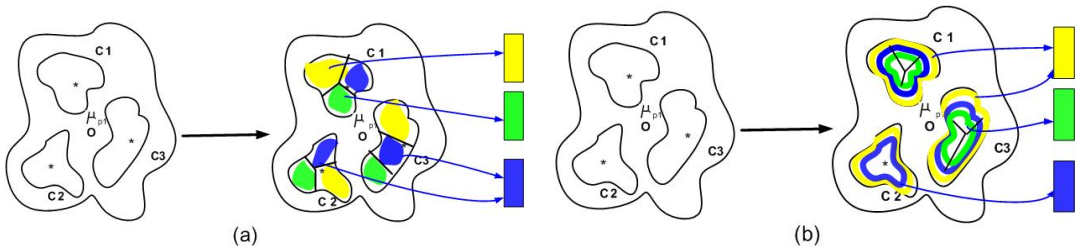


Figure 4.6: *a)* Slice Strategy, *b)* Contour Strategy

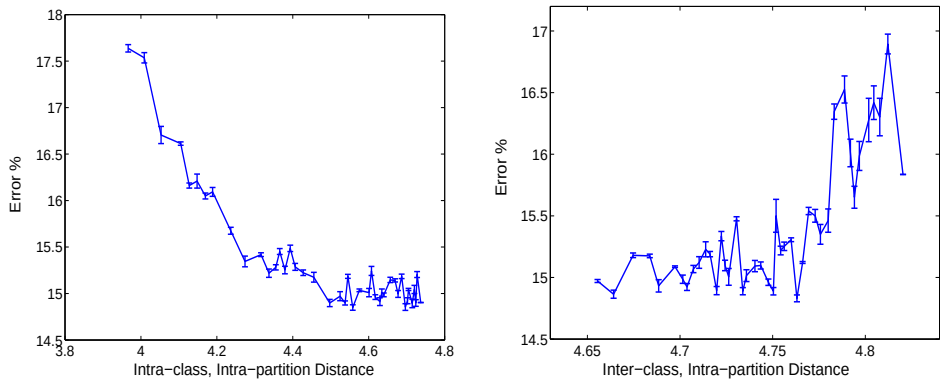
Subsequent to the generation of the solutions pools, these training sets were used to construct classifier ensembles (Algorithm 2). For base classifiers this time, in addition to Naive Bayesian classifiers, we examined the use of an unstable classifier: Multilayer Perceptron (MLP) [48]. MLP is known to be sensitive to the data, which is suitable for our purpose. We intended to apply MLP to gain insight into the effect data that distribution has on the combination of such classifiers. Furthermore, we applied both the product combining rule as well as a dynamic classifier selection (DCS) method.

Results & Discussion

As we discussed, we applied Euclidean, Mahalanobis, Bhattacharyya, and Patrick-Fisher distances, Equations 4.8-4.12, to estimate the distances among the training sets. Among these measures, Euclidean distance has shown to have the largest inconsistency and fluctuations, while the other three were more stable. Since feature attributes do not have the same variance in each dimension, it is necessary to use a normalized distance measure. Mahalanobis distance has this property, as well as being a good representation of the Bayes error. The results presented in this section are based on this measure.

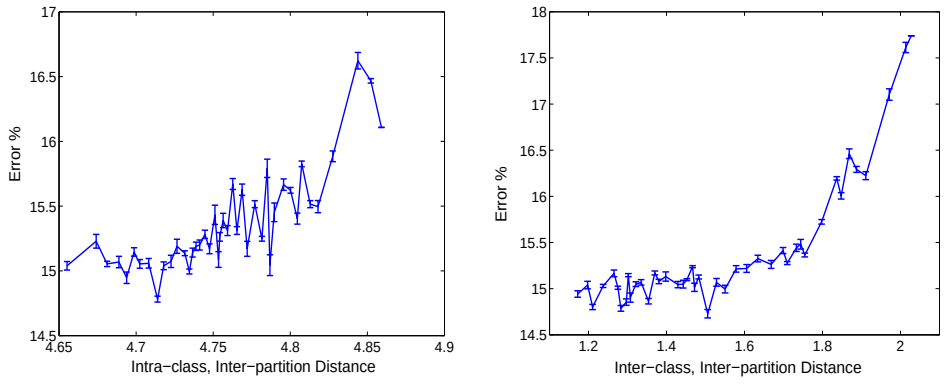
Two classes are called highly separable if the distance between the features within a class is minimum, and the distance between any two classes is maximum. By applying the aforementioned generation schemes (Algorithms 1 and 2) and partitioning strategies, we aimed to generate highly separable classes within the partitions in a solution. We increased the correlation among the partitions by randomly selecting and distributing data patterns among partitions. Relationships between the error rates of generated MCS and proposed distance measures (Equations 4.13-4.16) are highlighted in Figures 4.7-4.14 as well as Figures A.11-A.14 in Appendix A.2. Since all datasets showed similar trends, we only illustrate the results for 20-class Gaussian, Vehicle, Breast Cancer, and German datasets. The test error rates, shown on the y-axis, have been averaged over 10 training data partitioning iterations, as well as 25 different sets of solutions. First set of data partitions had been generated by contour strategy.

In Figures 4.7 (a) and 4.9 (a), the intra-class distance evaluates the compactness of a class in each partition. As we have shown, the smaller the distances between the patterns and their class mean, the less accurate were the recognition rates. A similar investigation was carried out on intra-class/inter-partition measure. This time, the pairwise distances among identical classes distributed across the set of partitions were calculated (Figures 4.8 (a) and 4.10 (a)). The error tended to increase as the sum of the distances of a class distributed among the partitions increased. These behaviors are likely due to the fact that generated ensembles were tested on a subset of the original data which had a similar distribution to the training data. When we distributed the training data among disjoint sets and forcing well-separable classes to accumulate in one subset, even though partitions with dense



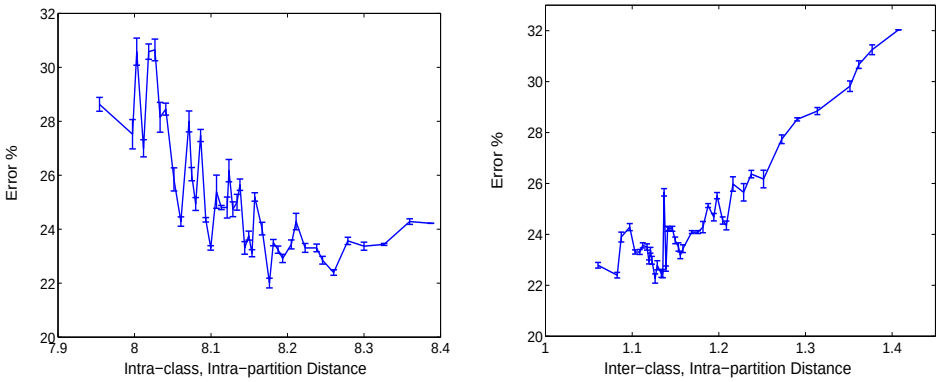
(a) (b)

Figure 4.7: 20-class Gaussian (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



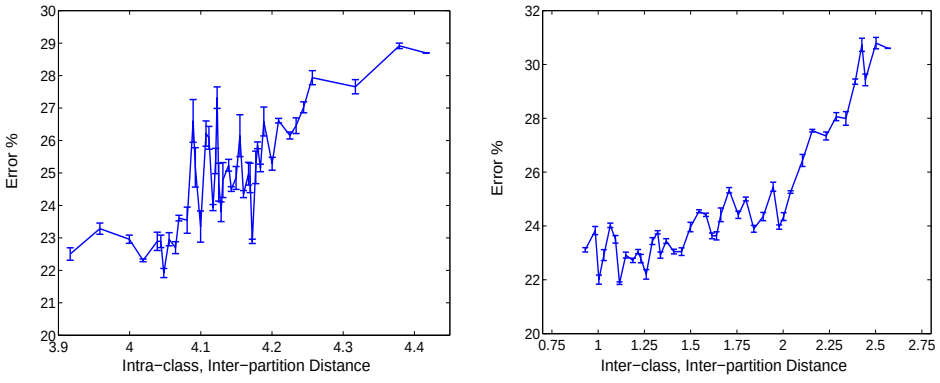
(a) (b)

Figure 4.8: 20-class Gaussian (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure 4.9: Vehicle (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



(a) (b)

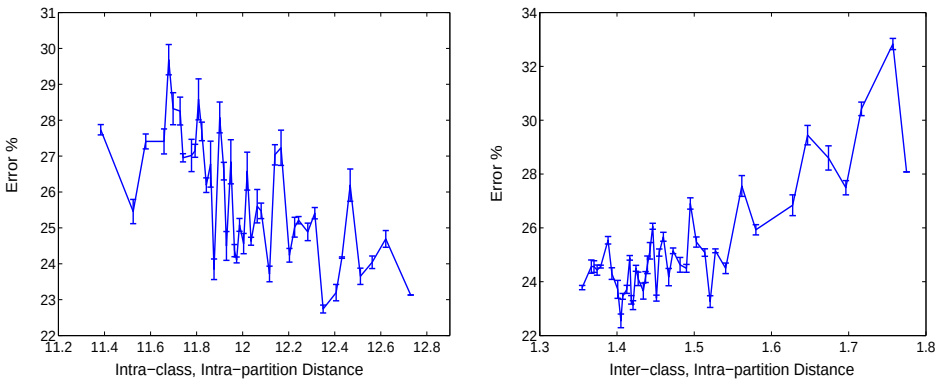
Figure 4.10: Vehicle (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

classes were obtained, they were not the actual representatives of the original training data. Individual classifiers trained on these subsets did not have an accurate view of the test subset and were biased towards their own training subset. Furthermore, we considered inter-class, intra-partition distance measures (Figures 4.7 (b) and 4.9 (b)). The overall trend in these cases showed that training partitions with smaller inter-class distances outperformed the ensembles generated on large distances. This is due to the aforementioned analogy, where base classifiers trained on partitions with similar distribution to the testing set resulted in higher accuracy. With inter-partition/inter-class measures (overall collection), we examined the distances among training partitions. The results suggested that a smaller distance among the means improves MCS classification accuracy, which is consistent with our other observations (Figures 4.8 (b), 4.10 (b)).

We ran the similar set of experiments using a pool of Multi-Layer Perceptron classifiers, instead of stable classifiers. Figures 4.11 and 4.12 depicts the result for the Vehicle dataset. As shown in this figure, the type of individual classifiers has not affected the trend of the classification error. Regardless of the type of classifiers, MCSs performed better when base classifiers have an accurate view of the problem space. Other than classifiers, combining rules can also be data dependent [57]. Dynamic classifier selection (DCS) methods are examples of such rules. We implemented Woods *et al.*'s DCS method, in which the local accuracy is utilized to select the classifier of choice for a presented input pattern. In this method, when an unknown pattern is presented, it is clustered with the closest training patterns, and the most accurate classifier is selected for classification. The results presented in Figures 4.13 and 4.14 depicts the error for a dynamic selection of a pool of Multi-Layer Perceptron classifiers. Although the accuracy was improved, the pattern of change for all feature-based measures was the similar to ones that were combined with a fixed combining rule. We observed the same behavior for all other datasets. Comparing Slice and Contour strategies, we found that Slice strategy outperformed or showed identical performance to Contour strategy in almost all cases.

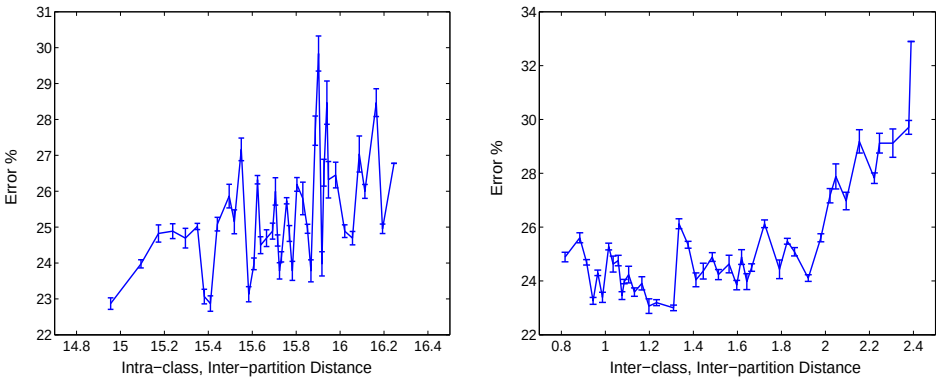
4.5 Overlapped Partitioning

Similar experiments to disjoint partitioning were performed on overlapped training subsets. The experimental setup was similar in such a way that, first, a large



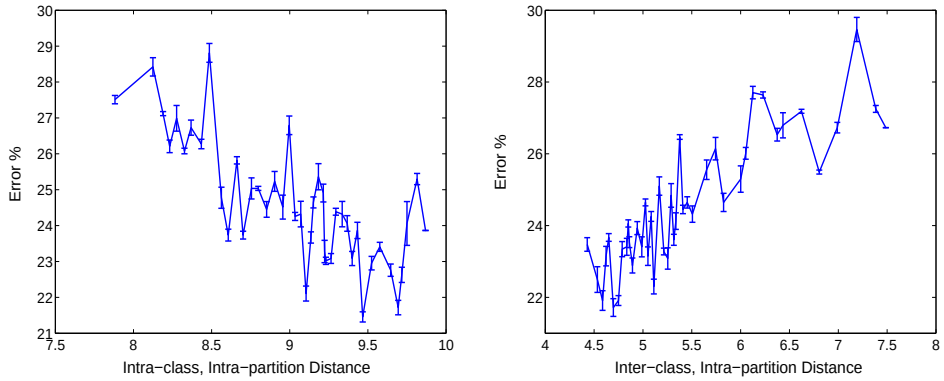
(a) (b)

Figure 4.11: Vehicle using Backpropagation (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



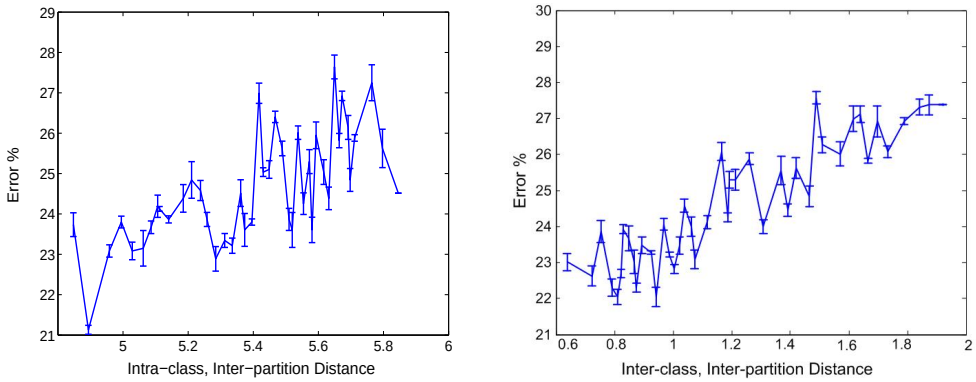
(a) (b)

Figure 4.12: Vehicle using Backpropagation (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure 4.13: Vehicle using Backpropagation & Dynamic Classifier Selection (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



(a) (b)

Figure 4.14: Vehicle using Backpropagation & Dynamic Classifier Selection (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

population of overlapped subsets were generated with different class distribution and correlations. Then, class-based and feature-based measures were calculated for each set of partitions and their impacts on MCS performance were examined and ranked. The main difference was in the initial training partitions as well as overlapping partition generation method.

An ensemble of seven Naive Bayesian classifiers was used for all the datasets. The product of the estimated posterior probabilities by the base classifiers was used as combining rule.

4.5.1 Empirical Assessment of Class-based Measures

Training Solutions Generation Scheme

The goal was to generate overlapped partitions with different degrees of class diversity. Similar to disjoint partitioning, an initial set of partitions was obtained in such a way that each partition contained only one class (decoupled strategy). Unlike disjoint partitioning, however, elements of each partition were randomly selected and added to the partition until the size of each partition was equal to the size of the original training data. Importantly, there was no overlap between this initial set of partitions (Figure 4.15 (a)). These sets remained the baseline for the generation of succeeding set of solutions.

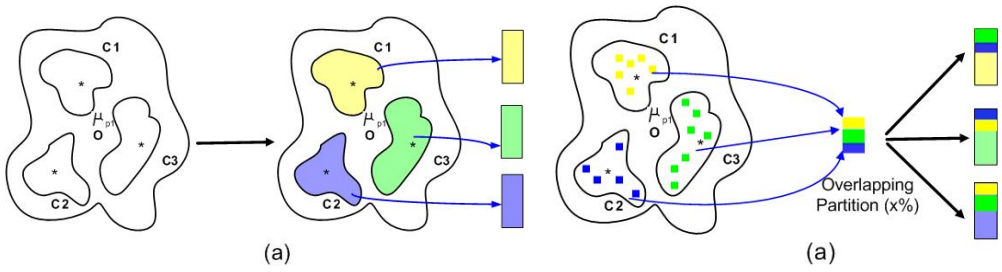


Figure 4.15: *a)* Decoupled Strategy, *b)* Generating x% random overlapping partition

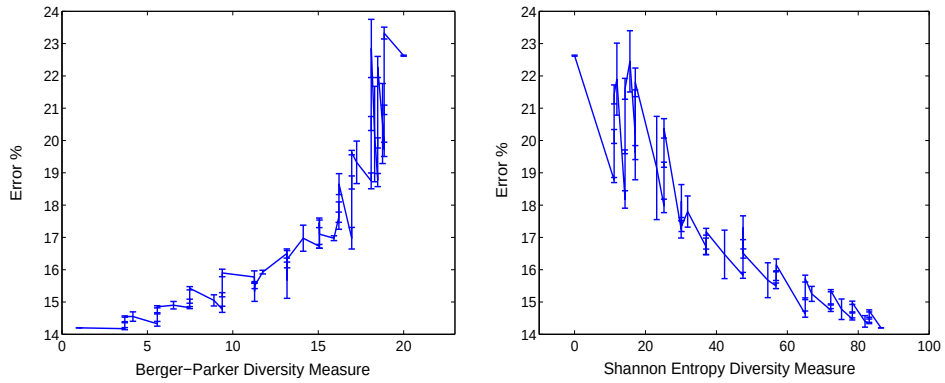
By randomly selecting a certain amount of patterns from the training data, the overlapping partition was generated (Figure 4.15 (b)). The same overlapping partition was added to all the training partitions. The size overlapping partitions varied between 1%-100%; 100% overlap represents identical training partitions. The size of training partitions was constant throughout the experiments and was equal to the size original training data (Algorithm 3).

Algorithm 3 *Overlapped Training Partition Generation Scheme*

X_{train}: training data
X_{test}: test data
S_1^i: an initial set of training partitions generated in iteration i
S_t^i: a set of training partitions
m: number of partitions
P_{j^i}: training partition j generated in iteration i
M and b : are predefined integer values
r : is a predefined integer between 1 to 10
X_{random}: overlapping partition

for i=1:b
  generate X_{train}^i and X_{test}^i
  generate $S_1^i = P_1^i, ..., P_m^i$ using decoupled strategy
  for $j = 1 : m$
    adjust size(P_j^i)=size(X_{train}^i), where $j = 1, ..., m$
  end for
  initialize *overlap*% to an integer between 0 to 100
  $t = 2$
  while $overlap \leq 100$
    for $n = 1 : M$
      generate X_{random} by randomly selecting *overlap*% from X_{train}^i
      generate $S_t^i = P_1^{i,t}, ..., P_m^{i,t}$ by combining X_{random} with all S_1^i subsets
      adjust size($P_j^{i,t}$)=size(X_{train}^i), where $j = 1, ..., m$
      $t = t + 1$
    end for
    $overlap = overlap + r$
  end while
end for

Subsequent to the generation of a pool of training solutions, they were used to train classifier ensembles (Algorithm 4). Class-based measures were applied to



(a) (b)

Figure 4.16: 20-class Gaussian (Overlapped): a) Berger-Parker, b) Shannon Entropy

rank and evaluate training subsets according to the performance of the constructed ensembles.

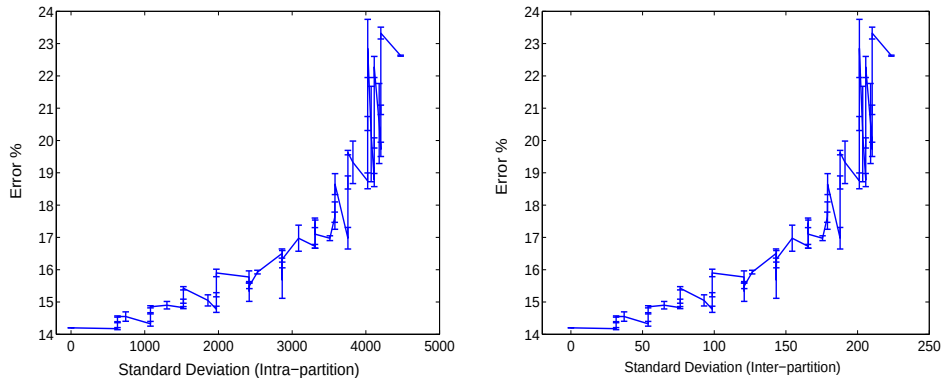
Algorithm 4 *Evaluation of Training Data Partitions*

construct ensembles on all S_i
calculate the feature-based measure for all S_i
estimate mean of ensemble error and the feature-based measure for $n = 25$ iterations
rank S_i according to the performance of constructed ensembles

Results & Discussion

A large sets of training partitions and ensembles were generated using Algorithms 2 and 3 for class-based related experiments. These experiments were carried out on all datasets (see 4.3). Since all of the datasets demonstrated identical patterns of change, we highlight the results for the same datasets in Section 4.4. Figures 4.16-4.17 and A.15-A.16 in Appendix A.2 depict the error rates verse represents class-based measures. Each error point represents mean of 25 different sets of partitions (solutions) for one size of overlap (between 1%-100%), repeated 10 times. Error bars represent standard deviation of mean.

Overlapped partitioning results were similar to those of disjoint partitioning. By increasing the degree of class diversity among training partitions, performance of classifier ensembles improved accordingly.



(a) (b)

Figure 4.17: 20-class Gaussian (Overlapped): a) STD Inter-class/intra-partition b) STD Intra-class/inter-partition

4.5.2 Empirical Assessment of Feature-based Measures

Empirical assessment of feature-based measures was also conducted on overlapped partitions.

Training Solutions Generation Scheme

For overlapped partitioning feature-based analysis, construction of the training partitions was different from the the disjoint partitioning. The main difference was in the way overlapping partitions were generated. Our aim in the proposed strategies was to generate overlapping partitions that represent different parts of the data-space so that MCS performance can be evaluated with respect to various types of training data distribution. The initial set of partitions was generated using the previously discussed strategies: decoupled, slice and contour. Overlapping partitions were generated according to the following strategies:

1. Overlapping partitions of different sizes, 1%-100% of the training data, were constructed by randomly selecting a certain percentage of the training patterns. Selection of training patterns was accomplished with respect to the original distribution of the classes. This strategy is referred to as Random-Overlap strategy (RAN-OVR) (Figure 4.18 (a)).

2. A k-means clustering method was used to group each class to sub-clusters. Overlapping partitions of different sizes, 1%-100% of the training data, were constructed by randomly selecting a certain percentage of the training patterns. Selection of training patterns was accomplished with respect to the original distribution of the classes. This strategy is referred to as Slice-Overlap strategy (SLC-OVR) (Figure 4.18 (b)).

3. Overlapping sets were generated based on distances of the data patterns from the center of their classes. We first sorted the patterns based on their distances to the center. Then, gradually, a certain percentage of the farthest patterns from the center of the class was selected to construct overlapping partitions. Selection of training patterns was accomplished with respect to the original distribution of the classes. Because of the similarity of this strategy to contour strategy, we refer to it as Class Center Contour Overlap (CC-CNT) (Figure 4.19 (c)).

4. This strategy was similar to the Decoupled Class Center Contour Overlapping strategy. The main difference was that instead of selecting farthest patterns from the class center, this time, we selected the nearest patterns to the class center to construct overlapping partitions. We refer to this strategy as Class Border Contour Overlap (CB-CNT) (Figure 4.19 (d)).

5. The initial set of training partitions was generated using the slice strategy (Figure 4.6 (a)). This set remained the baseline for the generation of the succeeding set of solutions. Overlapping partitions of different sizes, 1%-100% of the training data, were constructed by randomly selecting a certain percentage of the training patterns. Selection of training patterns was accomplished with respect to the original distribution of the classes. This strategy is referred to as Slice-Random Overlap (SLC-RAN).

6. This strategy was similar to the Slice-Random Overlapping. The only difference was that the initial set of training solution was generated using contour strategy (Figure 4.6 (b)). This strategy is referred to as Contour Random Overlap (CNT-RAN)

Training solution creation algorithm was similar to the one presented in Algorithm 3. However, construction of the overlapping partitions varied based on the

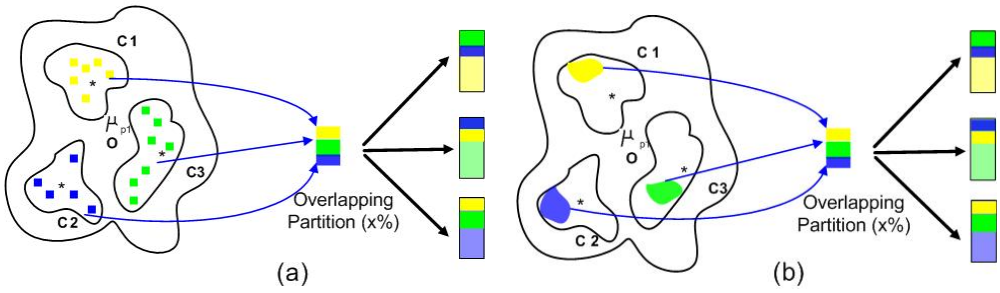


Figure 4.18: *a)* Random-Overlap (RAN-OVR), *b)* Slice-Overlap (SLC-OVR)

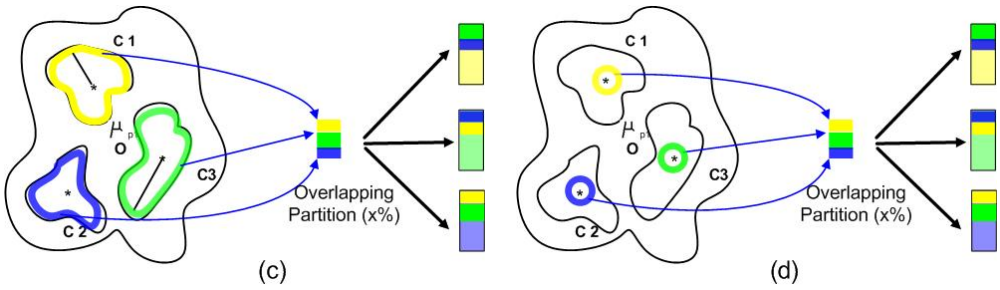
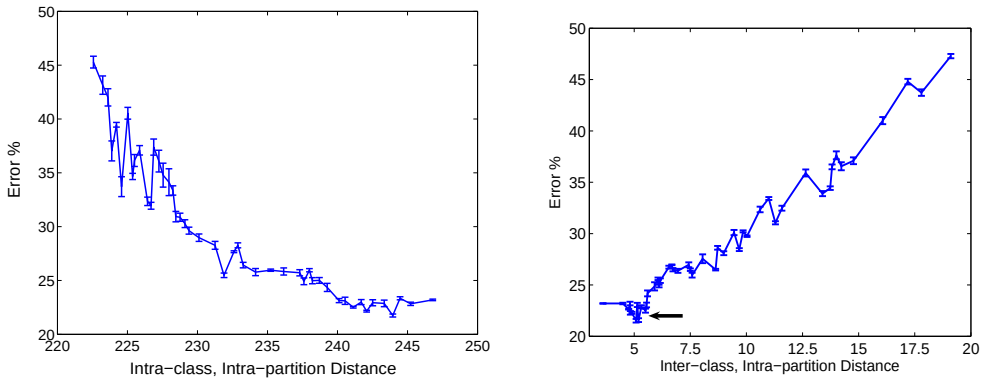


Figure 4.19: *c)* Class Border Contour Overlap (CB-CNT), *d)* Class Center Contour Overlap (CC-CNT)



(a) (b)

Figure 4.20: Vehicle (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition

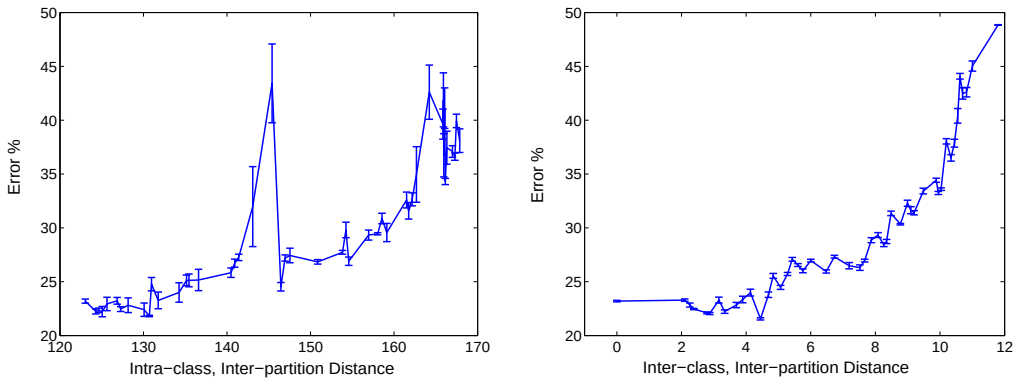
strategies discussed above.

Subsequent to the generation of pool of training partitions, they were used to construct classifier ensembles (Algorithm 2). For base classifiers, in addition to Naive Bayesian classifiers, we examined the use of an unstable classifier: Multilayer Perceptron (MLP) [48]. Furthermore, we applied both the product combining rule as well as a dynamic classifier selection (DCS) method.

Results & Discussion

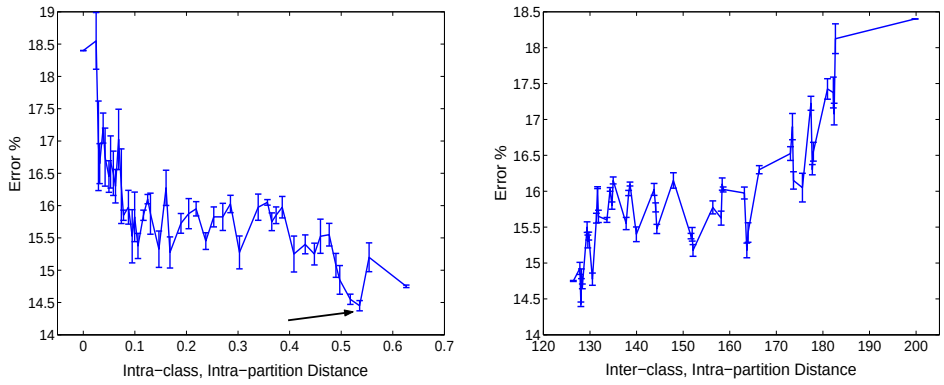
Decoupled, slice and contour strategies were designed to generate highly separable classes within the partitions in the first solution. Overlapping strategies, however, were designed to gradually increase correlation among training partitions by presenting overlapping patterns from different parts of the data space. We measured this correlation using feature-based distance measures introduced in Equations 4.13-4.16. Mahalanobis distance (Equation 4.10) was applied to calculate feature-based measures. MCSs error rates against feature-based measures are highlighted in Figures 4.20-4.27 as well as Figures A.17-A.20 in Appendix A.3. In these figures, the first set of training solutions was generated with contour strategy.

Once again, the trend of MCS error rates changes with respect to feature-based measures was similar to the trend observed for disjoint partitioning. In Figures 4.20 (a) and 4.22 (a), the smaller the distances between the patterns and their



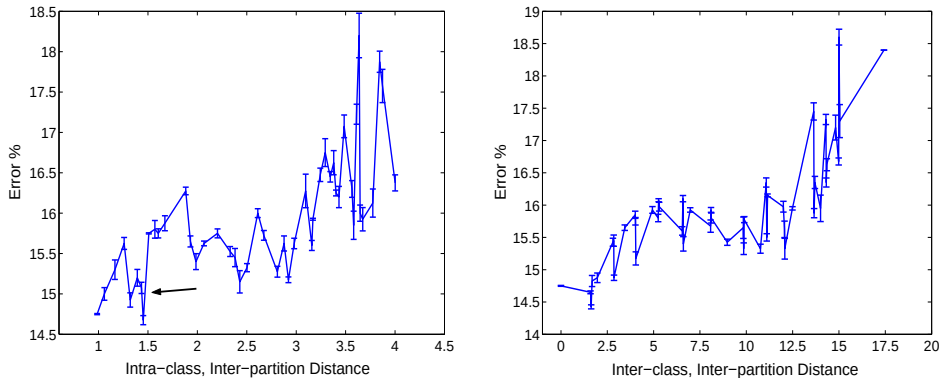
(a) (b)

Figure 4.21: Vehicle (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure 4.22: 20-class Gaussian (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



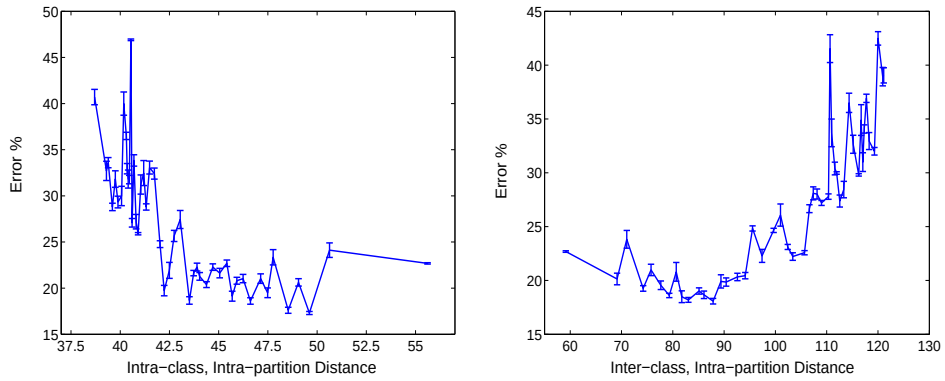
(a) (b)

Figure 4.23: 20-class Gaussian (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

class mean, the less accurate the recognition rates. For intra-class, inter-partition distance (Figures 4.21 (a) and 4.23 (a)), the error tended to increase as the sum of the distances of a class distributed among the partitions increased. These results verified our previous observations for disjoint partitioning that training partitions with dense classes were not the accurate representatives of the original training and test data. Consequently, classifiers constructed on these biased training partitions were not able to perform accurately. Results in Figures 4.20 (b), 4.22 (b), 4.21 (b) and 4.23 (b) confirmed our previous observation, where base classifiers trained on partitions with similar distribution to the testing set resulted in higher accuracy.

For large overlap sizes, we were expecting lower error rates compared to disjoint partitions since classifiers were trained with larger training sets. However, that was not the case for most of the datasets used in this study. Disjoint partitions with large class diversity demonstrated a similar performances to overlapping partitions.

An important observation from Figures 4.20-4.23 is that classifier ensembles with the lowest error rates were mostly trained with overlap sizes between 65% to 96% (marked by black arrows in the figures). Identical training data (100% overlap) did not always result in lower error. To validate this observation and to make sure that this behavior was not due to the use of stable classifiers, we ran a similar set of experiments using a pool of Multi-Layer Perceptron classifiers. Figures 4.24 and 4.25 depicts the results for the Vehicle dataset. We also applied a different combining method than product rule, Dynamic Classifier Selection method [108], and searched



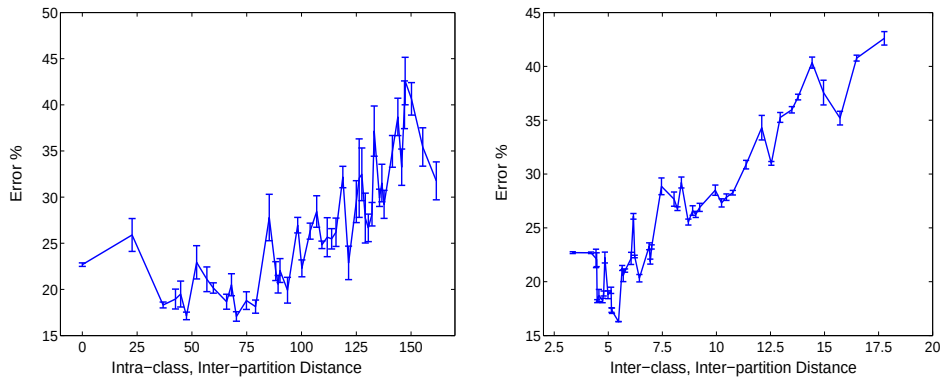
(a) (b)

Figure 4.24: Vehicle using Backpropagation (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition

for the best performing classifier ensembles (Figures 4.26 and 4.27). Interestingly, the type of individual classifiers or combining method did not change the trend of the classification error. Regardless of the type of the algorithm, MCSs performed better when base classifiers had an accurate view of the problem space. The best performing ensembles were constructed on overlapping partitions of size 65% to 96% of the training data. This is an important finding that could shed new light on the effectiveness of some of the existing MCS training techniques (e.g. bagging) as well as construction of nearly optimal training partitions.

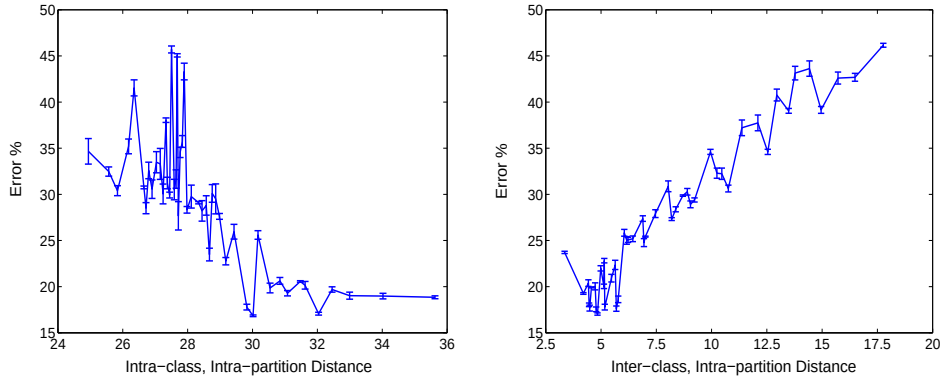
We examined the impact of overlap sizes on ensemble accuracy by assessing base classifier diversity for each of generated ensembles. Measure of Entropy (Equation 2.3) was applied to calculate classifier diversity, where 0 indicates no difference and 1 indicates the highest possible diversity. Classifier diversity versus error rate for Vehicle, 20-class Gaussian, and Breast Cancer is highlighted in Figures 4.28 and 4.29. As illustrated in these figures, stable classifiers resulted in lower diversity compared to unstable classifier such as multi-layer perceptron (MLP) (Figure 4.28 (a) and (b)). This behavior was expected because of the randomness in MLP parameters.

There was a general trend shown in all these figures by the boosted ensembles, where lower diversity meant generally higher accuracy. This behavior was due to the distribution of training partitions and the base classifier performance that were constructed on biased partitions. Although highly diverse base classifiers were generated using biased partitions, most of these classifiers performed poorly on the test



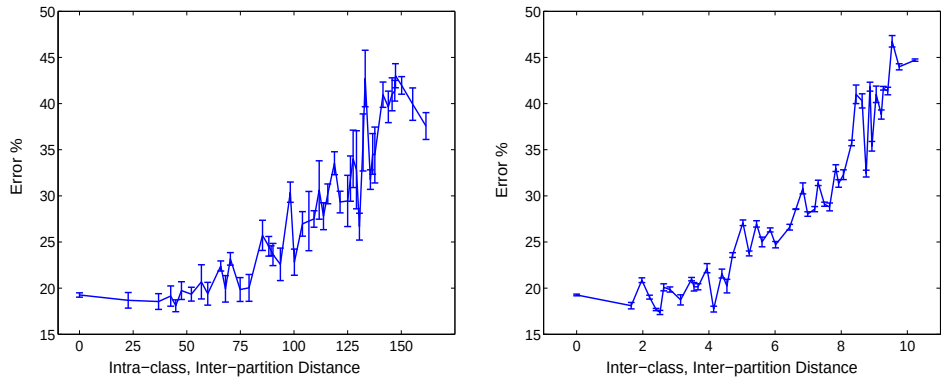
(a) (b)

Figure 4.25: Vehicle using Backpropagation (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure 4.26: Vehicle using Backpropagation & Dynamic Classifier Selection (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



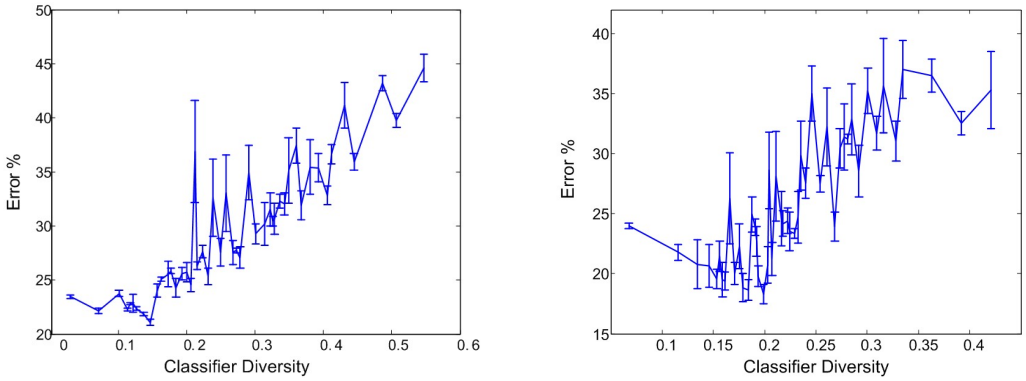
(a) (b)

Figure 4.27: Vehicle using Backpropagation & Dynamic Classifier Selection (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

set. As a result, their combination could not improve the overall generalization ability either. A common pattern in all these figures is that, for large overlap sizes, no consistent trend between classifier diversity and ensemble error can be observed. This behavior suggests that the diversity generated by biased training partitions is not beneficial for the MCSs. The most useful diversity is achieved by perturbing a small subset of original training data instead of imposing larger variations. In bagging, each bootstrap replicate contains 63.2% of the original training set. The same type of functionality is common among other popular MCS training methods.

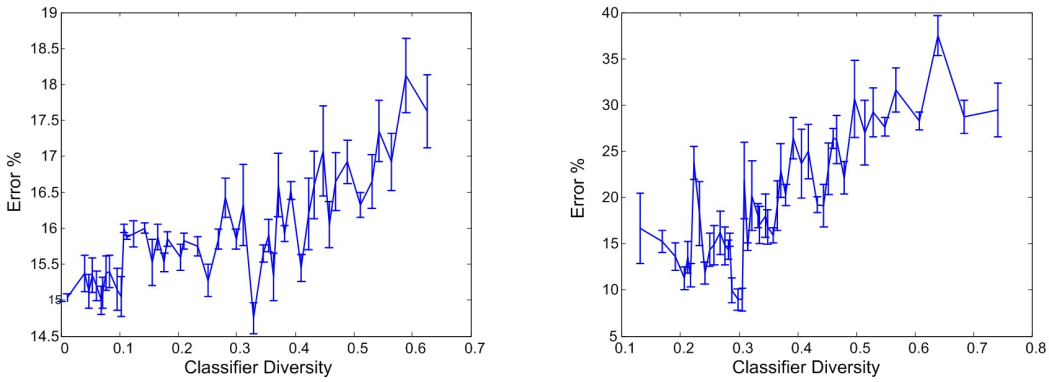
4.6 Training Multiple Classifier Systems

Results in the previous sections illustrated that a high degree of class diversity in a training partition improves classifier accuracy. Another conclusion was that MCS error is non-linearly correlated with the proposed feature-based measures. These observations suggest that class-based and feature-based measures provide the objective functions which can be used to find sub-optimal training data partitions through an optimization procedure. Therefore, we expect to generate sub-optimal training partitions using these measures, where MCS performance is at its best. To verify this assumption, a combined analysis of feature-based and class-based measures with respect to ensemble performance was performed. The key goals of this analysis was to: *1)* to examine the functionality of feature-based and class-



(a) (b)

Figure 4.28: Classifier Diversity: a) Vehicle dataset using stable base classifiers b) Vehicle dataset using MLP base classifiers



(a) (b)

Figure 4.29: Classifier Diversity: a) 20-Class Gaussian dataset using MLP base classifiers b) Breast Cancer using MLP base classifiers

based measures and their rules as objective functions and *2)* to investigate how these measures can be utilized to generate sub-optimal training partitions.

To achieve the first goal, we plotted a class-based measure (class diversity) against inter-class/inter-partition distance (Figure 4.30). We refer to this two-dimensional space, represented by *c*lass *d*iversity and *d*istance, as CDD-space. We then divided CDD-space into four regions. Each region is characterized according to class diversity and distance values. For example, region (1,1) represents low diversity and small distance area, while region (2,2) depicts high diversity and large distance area. Based on our observations from the previous sections, we expected to see MCS best performances in the high class diversity and small distance region (2,1).

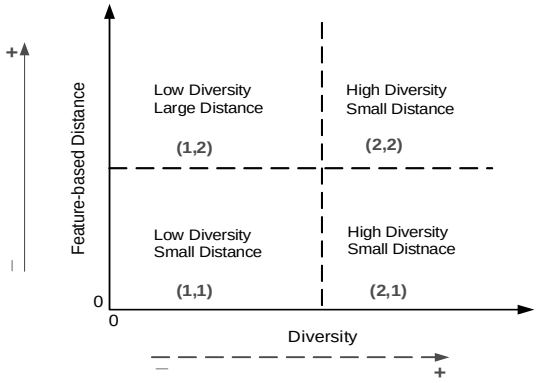
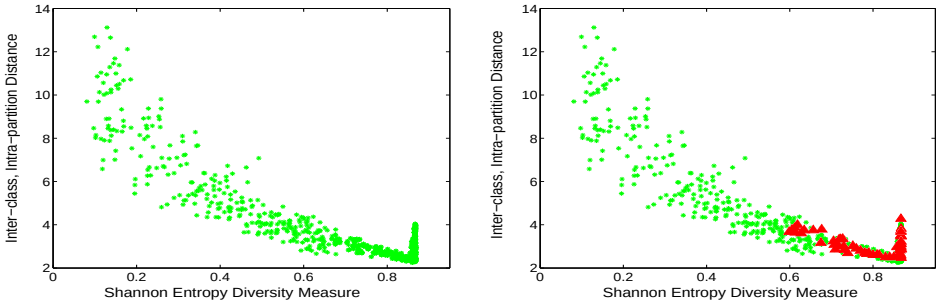


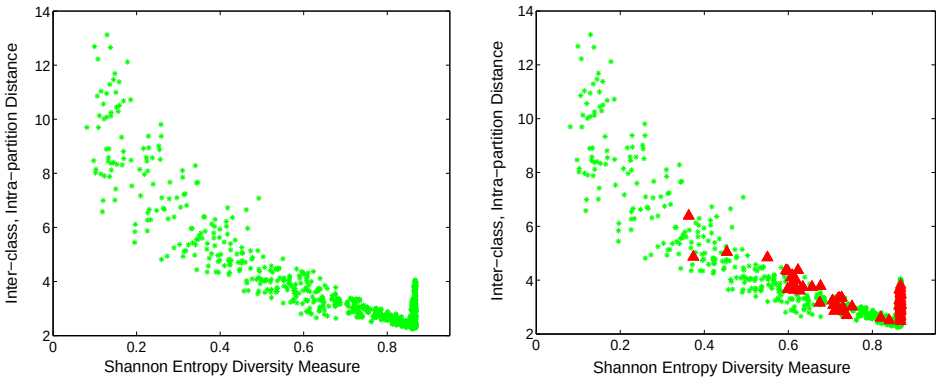
Figure 4.30: Training Partitions Class Diversity vs Distance

Training and overlapping partitions strategies proposed in Section 4.5 were all used to generate a diverse and large pool of training solution. We calculated the Shannon Entropy measure and the inter-class/intra-partition distance and plotted the CDD-space. Figures 4.31-4.36 (a) highlight CDD-space for Vehicle, Breast Cancer and 80-D Gaussian datasets. The training solutions were used to train MLP base classifiers and their outputs were combined using product rule as well as the dynamic classifier selection method. We then ranked the solutions according to the MCS performance and selected the top 150 solutions with lowest MCS error. Top 150 solutions are highlighted on CDD-space with dark triangles (in red) for product rule (Figures 4.31 (b), 4.33 (b), and 4.35 (b)) and the dynamic classifier selection method (Figures 4.32 (b), 4.34 (b), and 4.36 (b)).



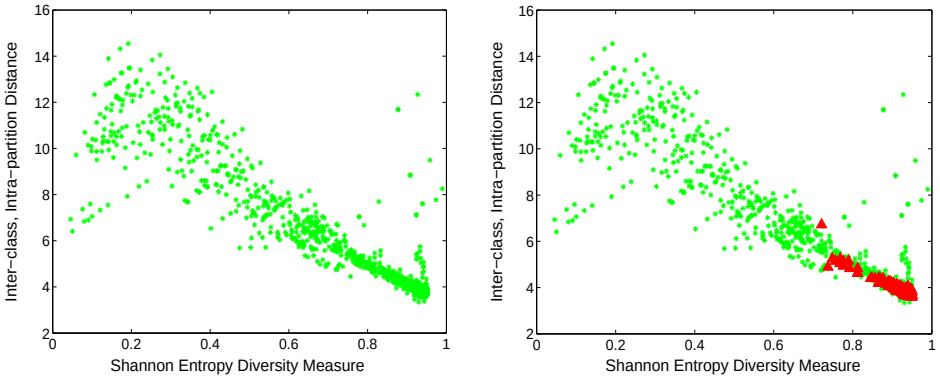
(a) (b)

Figure 4.31: Vehicle: a) CDD-space, b) best MCS performances using Product Rule



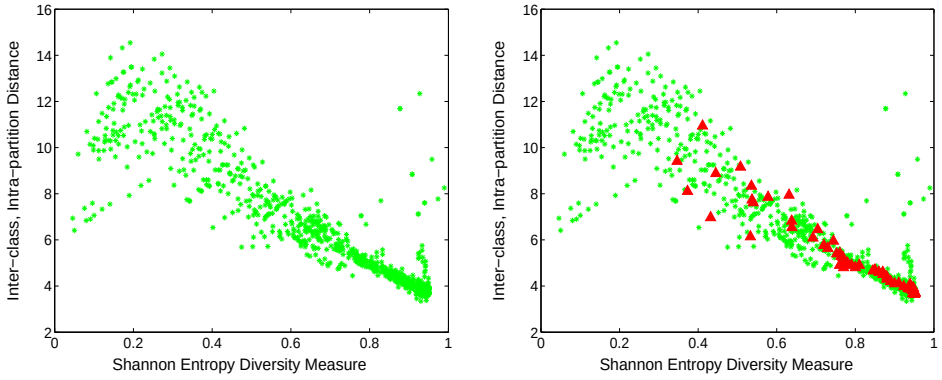
(a) (b)

Figure 4.32: Vehicle: a) CDD-space, b) best MCS performances using Dynamic Classifier Selection



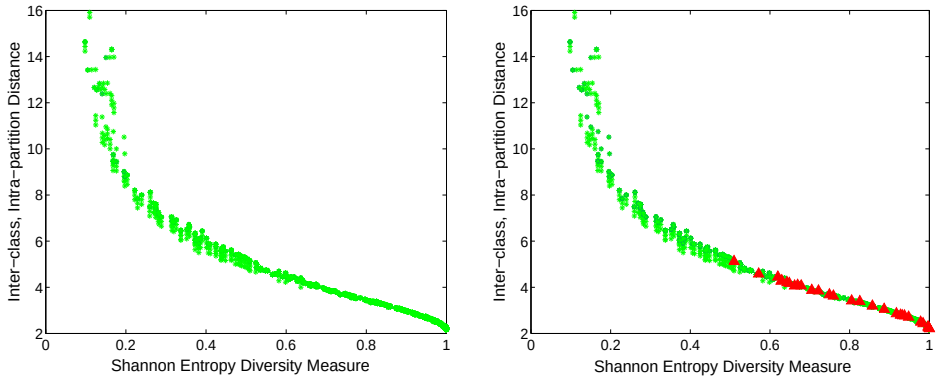
(a) (b)

Figure 4.33: Breast Cancer: a) CDD-space, b) best MCS performances using Product Rule



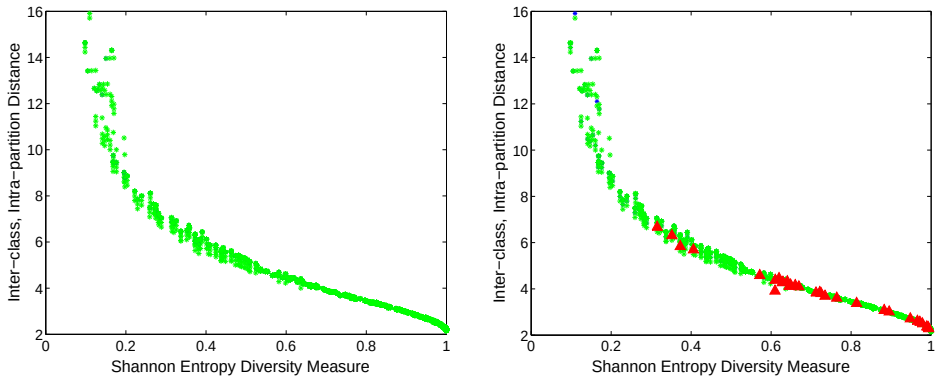
(a) (b)

Figure 4.34: Breast Cancer: a) CDD-space, b) best MCS performances using Dynamic Classifier Selection



(a) (b)

Figure 4.35: 80-D Gaussian: a) CDD-space, b) best MCS performances using Product Rule



(a) (b)

Figure 4.36: 80-D Gaussian: a) CDD-space, b) Best MCS performances using Dynamic Classifier Selection

The results illustrated in Figures 4.31-4.36 confirmed and extended the findings in the previous sections. Our first observation was that, as expected, the top 150 selected solutions were located in region (2,1); which again shows the best partitions for training MCS should be highly diverse with small feature distances. Another important observation was that the top 150 selected solutions for DCS method were more spread out in the CDD-space rather than being clumped together as they were in the case of the product combining rule. This phenomenon can be explained by the fact that classification of the test patterns in DCS does not depend on the output of all classifiers. Therefore, even if one classifier possesses the best performance, then that may be sufficient for DCS method to outperform other classifier ensemble methods. It is important to note, however, that even for DCS method, most of the top 150 partitions were located in the same (2,1) region.

Although the aforementioned findings confirmed our assumption and provided insight into the class-based and feature-based measures and their rule as objective functions, they did not satisfy our second goal. We know that the sub-optimal solutions will locate in region (2,1), but we do not have the tools to recognize and select them. To this end, we developed the *distance array* as depicted in Figure 4.37. Distance array contains four feature-based values and their corresponding MCS error.

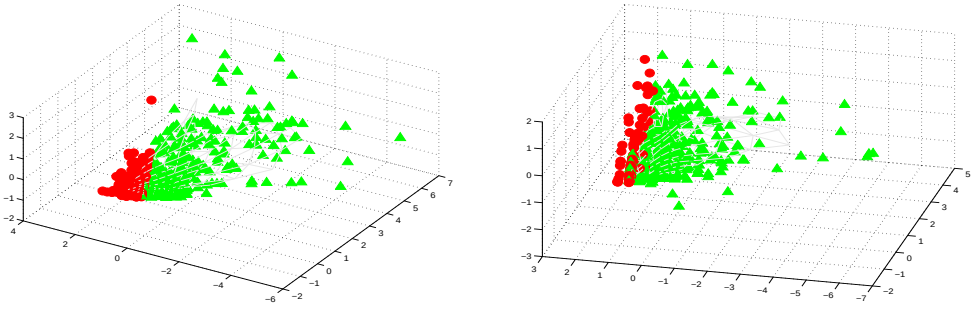
Berger Parker	Inter-class Intra-partition	Intra-class Inter-partition	Inter-class Inter-partition	MCS Error

Figure 4.37: Distance Array

We used the same pool of solutions that we generated for previous experiments and constructed the distance array for each set of training solution set. We then used a Self-Organizing Feature Map (SOM) [48] to cluster the distance arrays obtained from training partition solutions. The same top 150 solutions, discussed above, were again located and highlighted on the SOM map. The three dimensional SOM for Vehicle, Breast Cancer and 80-D Gaussian are depicted in Figures 4.38-4.40. The dark circles (in red), in these figures, show the top 150 solutions and light triangles (in green) illustrate the remaining solutions.

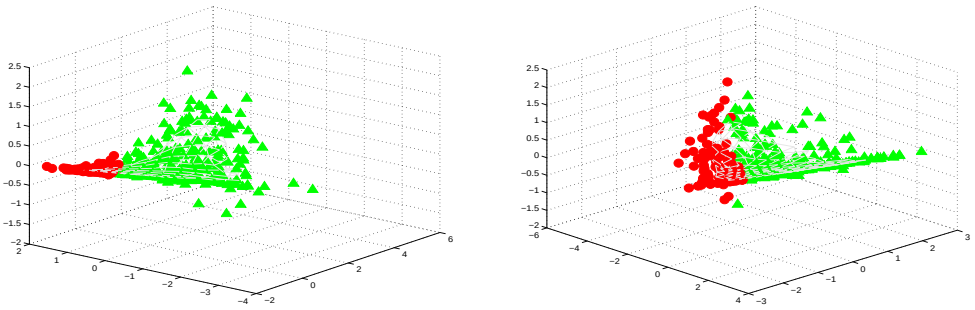
Figures 4.38-4.40 illustrate that, for all the datasets and combining methods, we obtained two distinct clusters located side-by-side. In the case of the product combining rule, clusters were clearly separated, while for the DCS method, clusters were slightly overlapped but completely distinguishable. This again confirms our previous observation regarding the DCS method and its functionality. Based on these observations, we can conclude that all top 150 solutions should have some features in common in order to be grouped in one cluster.

We looked closely at the training partitions and checked their distance arrays. We took the average of the distance arrays for all the datasets and examined for



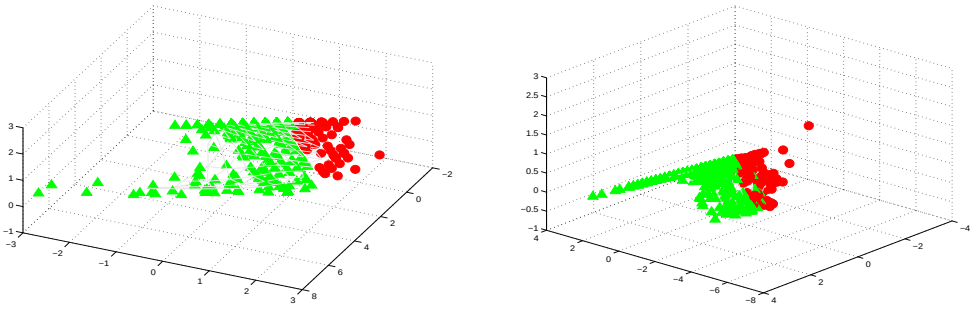
(a) (b)

Figure 4.38: Vehicle, Training Partition Cluster Space: a) Product Rule , b) Dynamic Classifier Selection



(a) (b)

Figure 4.39: Breast Cancer, Training Partition Cluster Space: a) Product Rule , b) Dynamic Classifier Selection



(a) (b)

Figure 4.40: 80-D Gaussian, Training Partition Cluster Space: a) Product Rule , b) Dynamic Classifier Selection

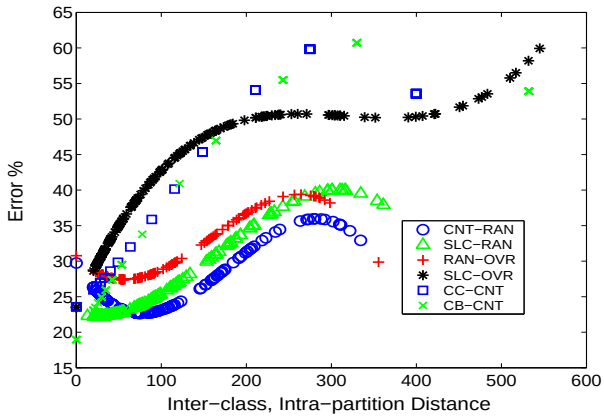


Figure 4.41: Vehicle: Empirical assessment of proposed partitioning strategies

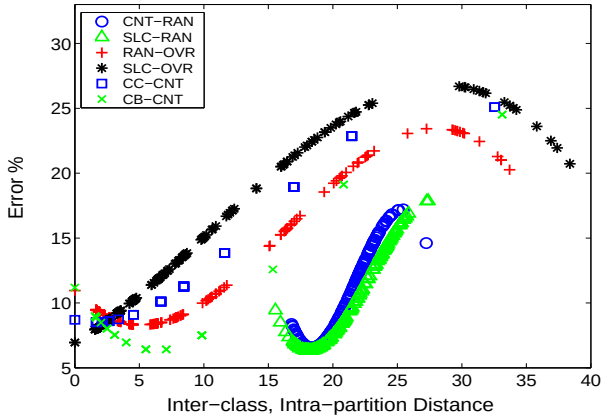


Figure 4.42: Breast Cancer: Empirical assessment of proposed partitioning strategies

the trend. We found that the average value for all 150 solutions was the lowest amongst all the solutions in the pool. This finding largely satisfied our second goal mentioned above. However, we still do not know whether any of the proposed training partition generation strategies has a superior performance over the others and which one results in the highest accuracy. Training partition generation strategies were empirically analyzed and the results are presented in Figures 4.41, 4.42, and 4.43. These figures display the best-fitted curves of the ensemble error versus the feature-based distance.

The first conspicuous observation from Figures 4.41 to 4.43 is that all parti-

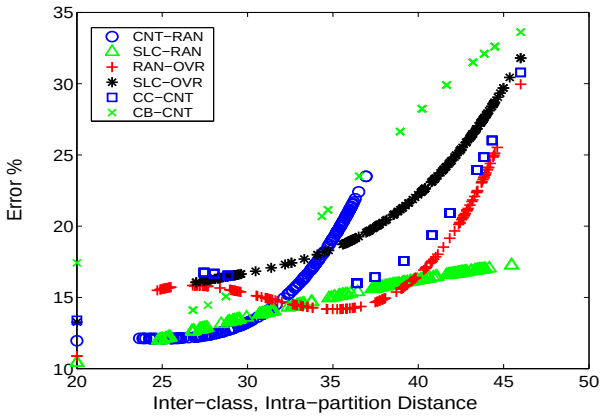


Figure 4.43: 80-D Gaussian: Empirical assessment of proposed partitioning strategies

tioning strategies behaved similarly with respect to the increase of overlap sizes. It is difficult to distinguish which strategy outperformed the rest in the existence of large overlaps. However, it can be concluded that contour (CTN-RAN) and slice (SLC-RAN) strategies outperformed other strategies in the presence of small size overlaps. Amongst these two strategies, SLC-RAN performed slightly better than CTN-RAN.

4.6.1 Cluster, De-cluster, and Selection Approach

We pooled all the observations from the previous section together and developed a new data partitioning technique called the Clustering, De-clustering and Selection (CDS) method (Algorithm 5). In this algorithm, training partitions are generated through three steps: clustering, de-clustering, and selection. In the first step, we initialize a set of training subsets by slice strategy where we cluster the classes and distribute them among partitions. Resultant partitions contain classes with high density and low correlation, and are highly diverse. We then replicate the patterns in each partition to the point that the size of partition is equal to the original training data. In the de-clustering step, we randomly select 70% of the training data and generate an overlapping partition. The overlapping partition is added to all the partitions in each step to construct a new set of training solution. Overlapping partition is added in such a way that the union of all the training partitions, $j = 1, ..., m$, in a solution i is equal to the original training data:

$$X_{train} = \bigcup_{j=1}^{m} P_{ij}.$$

This step is repeated until a stopping criterion, M, is met. Finally, a set of training partitions with the smallest distance-array is selected and used to train an ensemble. The choices of overlap sizes (70%) and slice partitioning strategy are made based on our findings discussed in previous sections.

Algorithm 5 *Clustering, De-clustering, Selection (CDS) Algorithm*

Initialize:

X_{train}: training data	X_{test}: test data
P_{ij}: j^{th} partition in the i^{th} training solution	k: number of classes
M: predefined number of iterations	DA: distance array
m: number of partitions	$size(P_{ij})$: size of partition P_{ij}

1. ***Clustering:***
   Slice strategy:
   for $j = 1 \leq k$
   cluster class $j \subset X_{train}$ to m disjoint subsets
   end for
   for $j = 1 \leq m$
   select one cluster from each class and construct P_{1j}
   adjust size(P_{1j})=size(X_{train})
   end for
2. ***De-clustering:***
   for $i = 1 \leq M$
   $f = 70$
   construct T by randomly selecting f% of X_{train}
   adjust size(P_{ij}) so that size(P_{ij})+size(T)=size(X_{train})
   add T to all partitions of P_i to construct $P_{(i+1)}$
   construct *distance-array* (DA) for each P_i
   end for
3. ***Selection:***
   for $i = 1 \leq M$
   calculate $A_i = \sum(DA_i)$
   end for
   select $s = index(min(A))$
   train m MLP with P_s and use product rule to build the ensemble E
   test E on X_{test}

It is important to note that the proposed CDS approach is a pre-classification process, where training information is not utilized to select sub-optimal training partitions. In the next section, we propose another approach to optimize selection of training partitions. We compare our proposed approaches to several existing MCS training approaches using several benchmark datasets in the following chapter.

4.7 Cooperation through Sharing Training Information

One type of cooperation at training level is through sharing training information. In this type of cooperation, training subsets are evaluated and partitioned based on the output of the base classifiers or aggregation rule before making the final decision (Figure 4.44). Boosting and Adaboosting, proposed by Freund and Schapire [37], are popular training methods that fall into this category. Schapire demonstrated that a series of weak learners can be converted to strong learners as a result of training the members of an ensemble on previously modified patterns. These patterns are modified by trained members of the ensemble. Boosting requires a large training data. As a result, Freund and Schapire [37] have proposed an algorithm, Adaboost, that largely avoids this problem. Essentially, in Adaboost, the training sets are adaptively re-sampled, so that the weights in the re-sampling rates are increased for those cases which are most often misclassified. Kamel and Wanas [104] proposed an evolving training algorithm (feature-based decision fusion) where base classifiers utilize the outcome of an aggregation method to rearrange the training subsets. In this method, repartitioning of the training data is not only dependent on the output of the base classifiers, but also it depends on the prediction of the aggregation rule. Some other techniques have been discussed in Chapter 3. Liu and Yao [79] proposed a cooperative ensemble learning system (CELS) which emphasizes interaction among individual base classifiers. CELS utilizes an unsupervised penalty term in the error function to produce biased neural networks that are negatively correlated.

The most obvious benefit of the this type of cooperation is that it automatically considers all the parameters and interactions that may influence system performance, including interdependency and correlation among data partitions, without any prior knowledge or users' intervention. The wrapper approach has three drawbacks, however: *1)* it can be time demanding and expensive since the population of partitions has to be evaluated and optimized step by step before finalizing the

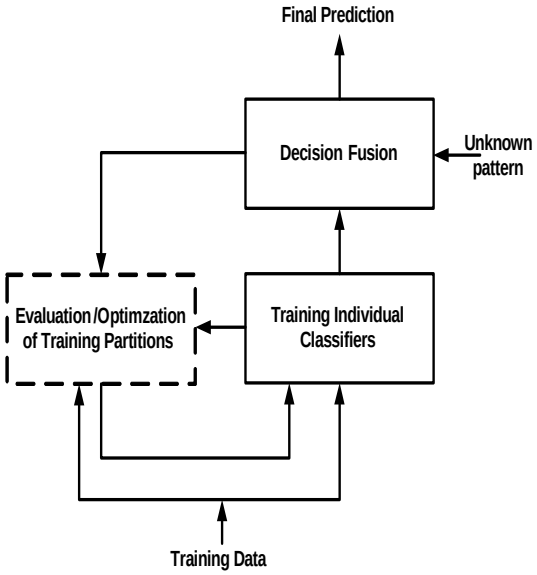


Figure 4.44: Cooperation through Sharing Training Information

training process, *2)* decisions on base classifier and the MCS architecture have to be made in advance, and *3)* a powerful architecture and/or base classifiers may be traded off by the weakness of selected partitions.

4.7.1 A Cooperative Approach to Optimization of Classifier Ensemble Training

We optimized the proposed sub-optimal training partition generation algorithm (Clustering, De-clustering, and Selection) by adding additional steps which incorporate training information into the selection of the training partitions. In other words, we enforced cooperation by sharing the output of the base classifiers with the other modules in the system, e.i. "selection" module, to improve training partition generation process. The rationale behind enforcing such cooperation is to direct the system toward generating more optimal training sets. We refer to this approach as Cooperative Clustering, De-clustering, and Selection (CO-CDS) algorithm.

CO-CDS works as follows: similar to CDS, in the clustering step, partitions with high class diversity and low correlation are generated. This training solution is de-clustered by randomly, with respect to class distribution, selecting 70% of the training data and generating a new set of training partitions. This is done in such

a way that the union of all the training partitions, $j = 1, ..., m$, in a solution i is equal to the original training data:

$$X_{train} = \bigcup_{j=1}^{m} P_{ij}.$$

This step is repeated M times. M is an integer that can be set to any predefined number by the user. By the end of de-clustering the step, the algorithm generates M training solutions. Distance-array DA_i is generated for all the training solutions. The selection step is designed to aim at finding sub-optimal training partitions. This step is optimized in CO-CDS and incorporates training information in the process. First, all distance-arrays DA_i $i = 1, ..., M$ are used to cluster the training solutions into two disjoint or overlapping clusters CL_1 and CL_2, similar to the training partition cluster space presented in Figures 4.38 to 4.40. Any clustering technique such as SOM, fuzzy c-means or kmeans can be applied. Then, between two clusters, the one with the smallest distance-arrays is selected as the candidate for further improvement

$$CL^* = min_i(\sum(DA_{Ci})) \;\; CL_i = 1, 2.$$

For each training solution belonging to cluster CL^*, there are m partitions. These training partitions are used to train a single classifier. This classifier is then tested on the "out-of-bag" training patterns. Out-of-bag data patterns are the ones that do not appear in the training partition. All training subsets in a solution have 70% overlap. The chances are that the 30% or less of the training patterns do not appear in each partition. The union of these patterns are referred to "out-of-bag" subset X_{ob}, and the classifiers tested on X_{ob} as out-of-bag classifier C_{ob}. The training solution with the smallest out-of-bag error e_{ob} is selected as the sub-optimal set of partitions

$$s = index(min(\sum_{i=1}^{m} e_{i,ob}))$$

where

$$e_{i,ob} = 1 - P(c|X_{i,ob}) \quad X_{i,ob} \subset X_{train}.$$

Base classifiers are constructed on P_s and combined with product rule. The resultant ensemble is tested on X_{test}. CO-CDS steps is presented in Algorithm 6.

The overall process of the CO-CDS algorithm is illustrated in Figure 4.45. CO-CDS and CDS were implemented and compared along with several other existing MCS training techniques. It was expected that CO-CDS outperforms CDS. Results are presented in the next section.

Algorithm 6 *Cooperative Clustering, De-clustering, and Selection (CO-CDS) Algorithm*

Initialize:

X_{train}: training data	X_{test}: test data
P_{ij}: j^{th} partition in the i^{th} training solution	k: number of classes
M: predefined number of iterations	DA: distance array
m: number of partitions	$size(P_{ij})$: size of partition P_{ij}
C_{ob}: out-of-bag classifier	$e_{i,ob}$: error rate of out-of-bag classifier

1. ***Clustering:***
   Slice strategy:
   for $j = 1 \leq k$
   cluster class $j \subset X_train$ to m disjoint subsets
   end for
   for $j = 1 \leq m$
   select one cluster from each class and construct P_{1j}
   adjust size(P_{1j})=size(X_{train})
   end for
2. ***De-clustering:***
   for $i = 1 \leq M$
   $f = 70$
   construct T by randomly selecting f% of X_{train}
   adjust size(P_{ij}) so that size(P_{ij})+size(T)=size(X_{train})
   add T to all partitions of P_i to construct $P_{(i+1)}$
   construct DA for each P_i
   end for
3. ***Selection:***
   cluster DA space into two clusters CL_1 and CL_2
   select CL^* as $min_i(\sum(DA_{CLi}))$, $i = 1, 2$
   train C_{ob} on all CL^* training partitions
   select $s = index(min(\sum_{i=1}^{m} e_{i,ob}))$
   train m MLP with P_s and use product rule to build the ensemble E
   test E on X_{test}

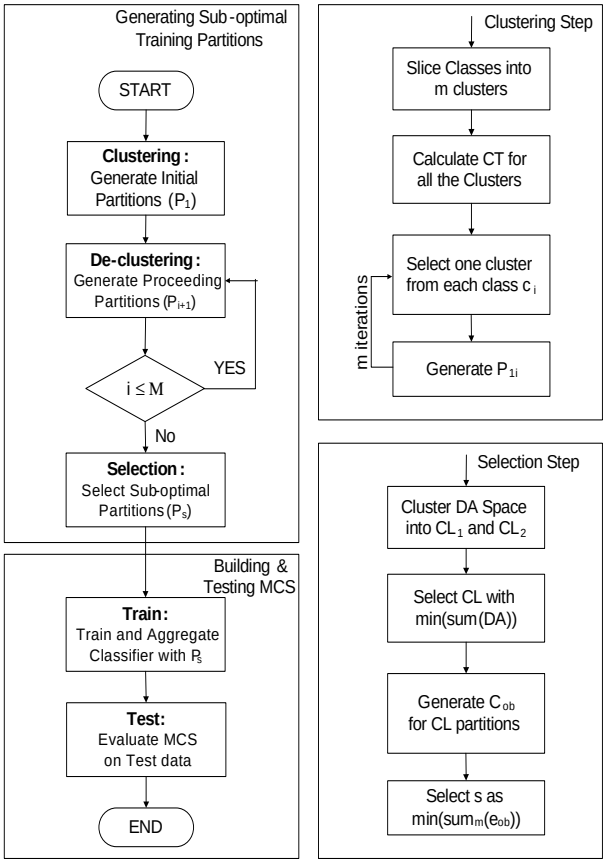


Figure 4.45: Flowchart of Cooperative Clustering, De-clustering, and Selection Algorithm

4.8 Summary

In this chapter, we examined various aspects of cooperation at training level. A summary of the findings is provided in the following:

- *Class-based and feature-based measures:* These measures enabled us to examine relationship between training data distribution and MCS performance. They provided the means to estimate class diversity and correlation for a set of training partition and to evaluate them with respect to classifier ensemble

performance. The empirical results suggested higher class diversity is always beneficial, regardless of the type of the base classifier or aggregation methods. Furthermore, we noticed that class diversity was not sufficient to obtain efficient training partitions. Training partitions should provide an accurate view of the problem space to the base classifiers. Training partitions that were generated using slice and contour strategies were highly uncorrelated. However, base classifiers trained with those subsets were biased and were not able to generalize well individually and collectively in an ensemble. This behavior was observed for both disjoint and overlapped partitions.

- *Training Partitioning Strategies:* Several training partitioning strategies were proposed and evaluated on different datasets. Contour and slice strategies outperformed the other strategies for smaller size overlaps, because of the highly diverse partitions they generate in the "clustering" step. For large overlaps, they demonstrated identical performances.

- *Overlap size and the issue of classifier diversity:* Empirical results illustrated that MCS error is non-linearly correlated with the feature-based measures. We also observed that best performed ensembles were not necessarily trained with identical training subsets (100% overlap). For many of the datasets, the best ensembles were constructed on overlapping partitions of size 65% to 96% of the training data. We also examined classifier diversity and its relationship with respect to training subsets distribution and ensemble generalization error. Although highly uncorrelated training partitions transferred extra diversity to the ensemble, resultant ensembles did not perform well. The most useful diversity was achieved by perturbing a small subset of original training data (less that 35%). This type of functionality is common among the most popular MCS training methods. This important finding provided us with the guidelines for developing new training partitioning algorithms: CDS and CO-CDS.

Chapter 5

Comparative Study and Results

We proposed two novel MCS training methods, CDS and CO-CDS, in the previous chapter. This chapter presents the results of a comprehensive comparative study of different MCS training methods. These training methods were compared based on three criteria: generalization ability, stability, and robustness to different degree of problem difficulty.

5.1 Experimental Objectives

Our experiments were aimed at the following objectives:

- evaluating the effectiveness of the proposed MCS training approaches: CDS and CO-CDS
- comparing CDS and CO-CDS training approaches with other existing approaches
- evaluating the stability of various MCS training approaches
- investigating the advantages of cooperation through sharing training information against other types of cooperations at training level.

The first and second objectives were satisfied by implementing our proposed CDS and CO-CDS approaches as well as several other MCS training approaches.

Effectiveness of these approaches was evaluated and compared with respect to their generalization ability. Generalization ability is the ability to correctly classify unseen data. This the most common evaluation criterion for the performance of classification algorithms.

For the third objective, we examined reproducibility of the classification results (stability). A classifier is considered stable if it is capable of producing repeatable results for the same problem [36]. We considered a pair-wise method to estimate stability: measure of similarity or agreement [66]. Let $D_1, ..., D_r$ be a set of multiple classifier systems. Then, we measure the stability index among a pair of D_i and D_j as

$$S_{ij} = \frac{1}{t} \sum_{k=1}^{t} agree(D_i(x_k), D_j(x_k))$$

where t is the number of test samples, and $agree$ is 1 if classifier ensembles i and j agree on the class of test sample x_k and 0 otherwise. An overall measure of stability among ensembles can be obtained by averaging the degree of agreement across the pairs. S varies between 0 and 1, where 0 represents the lowest degree of stability and 1 the highest.

It is a well known fact that the robustness and effectiveness of any pattern recognition problem is dependent on the data. Therefore, we collected 15 datasets with different statistical characteristics and used them to evaluate MCSs training approaches. A summary of datasets and their attributes can be found in Section 4.3 and in Appendix A.1. We grouped the MCS approaches based on their functionality, whether they cooperate with or without sharing training information, and compared them with respect to datasets level of difficulty.

5.2 Experimental Setup

The size of ensembles (or partitions) was set to $m = 7$ for all experiments. Each base classifier was a one-hidden layer multi-layer perceptron. These classifiers were trained using the error backpropagation algorithm. The number of hidden units for the classifiers was set to 6, 7, 8, 9, 10, 11, and 12 for each base classifier in the ensemble. The original dataset was divided into two subsets of training and

test using the holdout method. This was accomplished with respect to the original distribution of the classes. We ran each experiment 10 times for each dataset and for each ensemble technique. It is important to note that, in this study, we are mostly interested in to investigate the effectiveness of the training approach used for construction of the ensemble techniques, rather than comparing the ensemble techniques in terms of their architectures.

The following ensemble methods were implemented and compared in these experiments. Approaches that cooperated without sharing training information were:

- Majority Vote (Maj): Classifiers were trained independently on an identical training data. Then, majority vote aggregation rule was used to combined base classifiers output.
- Product Rule (Prod): Classifiers were trained independently on an identical training data. Then, product aggregation rule (Equation 2.4.1) was used to combined base classifiers output.
- Average Rule (Ave): Classifiers were trained independently on an identical training data. Then, average vote aggregation rule (Equation 2.4.1) was used to combined base classifiers output.
- Bagging (Bag): In this method, multiple versions were formed by making bootstrap replicates of the training data and using these new sets to train the classifiers. Bagging algorithm is discussed in detail in Section 3.2.4.
- Divide-and-Conquer (DAC): Divide-and-conquer method proposed in [38] was implemented in this study. Fuzzy c-means clustering method was applied to partition the training data (see Section 3.2.4).
- Random Partitioning (RP): In this method, m training partitions were generated by random sampling. Sampling was performed without considering the original class distribution. The size of partitions was equal to the size of the original training data, where some data patterns appeared several times in a partition and few others did not. The union of the training partitions was identical to the original training data. Partitioning was repeated 100 times. Training sets with the smallest distance-array were selected to construct an ensemble.

- Stratified Partitioning (SP): In this method, m training partitions were generated by random sampling. Sampling was performed with respect to original class distribution. The union of the training partitions was identical to the original training data. Partitioning was repeated 100 times. Training sets with the smallest distance-arrays were selected to construct an ensemble.

- Clustering, De-clustering, and Selection (CDS): This algorithm is discussed in detail in Section 4.6.1. Fuzzy c-means clustering algorithm was used to generate the initial set of training solutions.

Approaches that cooperated through sharing training information were:

- Ada-boosting (A-BST): Training data sampling is done is in such a way that misclassified patterns have a higher likelihood to be selected for the next step. Ada-boosting algorithm is discussed in Section 3.2.4.

- Feature-based decision fusion (FBDF): Feature-based decision fusion approach has been discussed in [104]. For this algorithm, average rule was used for final decision making. Detector and aggregation module were similar to the individual classifiers and had 6 hidden units. For each set of FBDF ensembles, detector and aggregation module were trained four time.

- Cooperative Clustering, De-clustering, and Selection (CO-CDS): This algorithm is discussed in details in Section 4.7.1. Fuzzy c-means clustering algorithm was used to generate the initial set of training solutions. Fuzzy c-means algorithm was also used to cluster distance array into two groups. A simple linear classifier was applied to generate out-of-bag classifiers.

- Cooperative ensemble learning system (CELS): CELS method proposed in [79] was implemented for this comparative study. CELS emphasizes cooperation among individual base classifiers and utilizes an unsupervised penalty term in the error function to produce negatively correlated neural networks (see Section 3.2.4).

To decrease the likelihood of external factors, all the experiments were set up using the same computing platform and language. All neural networks parameters were similar, for the same dataset, and all utilized the standard backpropagation

scheme. The same training and test sets were used to generate and evaluate the ensemble methods in each iteration. For CDS and CO-CDS methods, all steps were identical, except the "selection" step.

5.3 Results

5.3.1 Generalization Error

Tables 5.1 and 5.2 summarize the test results for 12 different methods and the single best classifier. The error results reported in these tables are averaged over 10 iterations. Standard deviations are also highlighted in parentheses for each mean value. Table 5.1 illustrates the results for low difficulty datasets and Tables 5.2 & 5.3 illustrate the results for high difficulty datasets. Each table is divided into two sections based on the type of cooperation: "Cooperation Without Sharing Training Information" and "Cooperation With Sharing Training Information". The lowest error rate for each dataset is underlined.

As stated before, one of the objectives of these experiments was to examine and compare CDS and CO-CDS effectiveness. We assessed the "performance" based on generalization error. This criterion was tested with respect to levels of problem difficulty. CO-CDS outperformed CDS almost for all problems, except for satimage and vowel. Considering CDS and CO-CDS only operated differently in the last step of the algorithm (selection step), it can be concluded that superiority of CO-CDS is mostly due to the use of training information in the selection of sub-optimal training partitions. CDS significantly outperformed bagging in 40% of the cases and bagging only outperformed CDS in 13% of the cases. For the other datasets, their performances were almost identical. Performances of A-BST and CO-CDS were comparable for 53% of the datasets. CO-CDS had the lowest error rates in 27% of the cases when compared to A-BST, while A-BST outperformed on the remaining 20% of the datasets.

Overall, amongst all the approaches used in this study, CO-CDS had the best generalization error. This approach significantly outperformed the other approaches for 27% of datasets, with the average error rate of 4.6% for low-difficulty datasets and 12.56% for high-difficulty datasets. A-BST was the second best with the average error rate of 5.11% for low-difficulty datasets and 14.15% for high-difficulty datasets.

Table 5.1: Test Error (%) and Standard Deviation for various MCS methods and low difficulty datasets

Methods	Clouds	Concentric	Iris	Ionosphere	Glass	Wine	B Cancer
SB	15.33(4.11)	1.01(0.45)	5.07(3.51)	11.2(2.16)	10.43(6.43)	3.9(1.44)	3.74(0.8)
	Cooperation	Without	Sharing	Training	Information		
Maj	11.96(1.18)	0.68(0.33)	4.53(2.82)	10.06(2.34)	8.12(3.25)	2.38(1.73)	3.38(0.81)
Ave	11.96(1.18)	0.71(0.28)	4.8(2.92)	10.26(2.08)	8.12(2.98)	2.03(1.85)	3.38(0.81)
Prodc	12.14(1.57)	0.71(0.34)	6.2(3.94)	9.03(2.2)	9.86(4.27)	3.79(3.77)	2.45(0.56)
Bag	12.22(0.52)	0.84(0.37)	5.33(4.52)	11.66(1.18)	6.96(2.38)	2.41(0.94)	3.24(0.16)
DAC	14.65(2.92)	1.16(0.39)	6.8(6.3)	11.09(1.6)	9.36(3.78)	3.79(2.72)	3.66(1.61)
RP	16.45(5.47)	3.82(1.18)	9.47(7.75)	13.74(6.13)	13.06(7.07)	5.41(2.54)	5.7(1.57)
SP	13.45(3.47)	0.73(0.68)	4.97(5.17)	9.94(2.73)	7.06(3.07)	3.45(1.07)	3.9(0.75)
CDS	13.69(1.27)	0.98(0.23)	4.53(3.12)	9.71(0.94)	5.8(1.67)	1.85(0.75)	3.18(0.25)
	Cooperation	With	Sharing	Training	Information		
A-BST	11.22(1.3)	0.75(0.21)	4.53(1.93)	8.91(1.81)	6.25(3.69)	1.72(1.72)	3.52(0.52)
FBDF	11.35(0.22)	0.95(0.26)	5.07(1.12)	10.74(1.63)	5.35(2.51)	2.41(1.48)	3.59(0.54)
CO-CDS	12.21(1.57)	0.79(0.28)	1.33(1.33)	9.37(1.66)	4.35(2.91)	1.52(1.58)	2.62(0.6)
CELS	12.51(1.19)	0.78(0.35)	4.53(1.79)	9.43(2.52)	7.93(3.76)	1.72(1.72)	3.31(0.47)

Table 5.2: Test Error (%) and Standard Deviation for various MCS methods and high difficulty datasets

Methods	20-Class	German	Phoneme	80-D Gauss
SB	16.4(0.9)	24.6(1.02)	19.16(1.64)	12.06(2.16)
	Cooperation	Without Sharing	Training	Information
Maj	14.64(0.9)	22.6(1.98)	17.82(1.82)	9.4(1.11)
Ave	15.3(0.9)	22.6(2.02)	17.82(1.21)	9.52(1.1)
Prodc	15.04(0.9)	22.6(2.8)	16.98(1.59)	9.4(0.65)
Bag	14.48(0.7)	23.44(0.88)	17.35(1.93)	12.53(1.5)
DAC	14.56(0.71)	23.88(1.22)	17.84(1.11)	12.99(2.08)
RP	16.04(6.8)	26.32(3.9)	20.36(3.1)	13.77(4.38)
SP	15.1(1.11)	23.32(1.9)	17.79(1.17)	9.77(1.38)
CDS	14.76(0.58)	24.1(0.9)	17.09(0.82)	8.25(1.03)
	Cooperation	With Sharing	Training	Information
A-BST	13.86(0.86)	21.92(1.02)	16.37(1.5)	10.18(1.12)
FBDF	15.26(0.72)	22.1(0.92)	16.45(0.48)	10.18(0.78)
CO-CDS	14.65(0.46)	21.96(1.04)	16.17(0.84)	7.95(0.85)
CELS	14.64(0.9)	22.6(1.02)	17.54(0.86)	9.7(0.69)

Fixed aggregation methods such Maj, Prod, and Ave performed surprisingly well in comparison to Bag and DAC. This observation was particularly interesting since these fixed aggregation approaches were trained on identical training data. The only source of diversity was different parameters of the base classifiers. Bagging generally performed better for high-difficulty datasets.

We noticed that amongst all the approaches, RP had the worst generalization accuracy. This behavior is consistent with our findings in Section 4.6, since class distribution was ignored when generating the training partitions. The resultant training partitions had low class-diversity. Although we implemented the same strategy as CDS for sub-optimal training partitioning selection, ensemble methods trained with these biased partitions were not able to generalize well the unseen data. Stratified partitioning approach, on the other hand, performed equally as well as bagging and, to some extent, CDS. Unlike the findings presented in [38], divide-and-conquer approach had the second worst error rates after RP. This could be due to the differences in experimental setup of our study and Frosyniotis *et. al.*

Table 5.3: Test Error (%) and Standard Deviation for various MCS methods and high difficulty datasets

Methods	Diabetes	Satimage	Vehicle	Vowel
SB	24.71(2.08)	15.67(1.55)	21.29(2.9)	13.94(3.82)
	Cooperation	Without Sharing	Training	Information
Maj	22.59(1.39)	14.23(1.27)	19.48(2.96)	13.94(2.24)
Ave	22.07(1.38)	14.29(1.2)	18.25(2.79)	13.94(3.38)
Prodc	22.06(1.28)	14.42(0.99)	19.27(2.28)	13.94(3.49)
Bag	21.17(1.58)	14.54(0.81)	19.81(1.12)	13.94(2.32)
DAC	24.67(1.08)	15.3(1.39)	19(2.68)	14.89(2.83)
RP	26.1(4.41)	19.49(3.94)	25.97(4.08)	18.91(4.77)
SP	22.1(1.41)	14.8(1.26)	19.12(3.45)	14.18(2.66)
CDS	23.92(1.16)	12.67(1.08)	19.34(3.36)	12.04(1.97)
	Cooperation	With Sharing	Training	Information
A-BST	21.19(1.88)	12.98(1.17)	17.35(2.9)	13.18(3.49)
FBDF	23.25(0.8)	12.84(0.51)	18.01(2.23)	13.94(3.15)
CO-CDS	21.4(0.87)	12.69(0.82)	18.01(2.49)	12.57(2.59)
CELS	22.11(1.05)	14.39(0.83)	18.53(2.84)	13.94(1.82)

Overall performances of MCS training methods categories are depicted in Figures 5.1 and 5.2. We averaged the error rates for all the datasets and displayed them as dot plots for each category: cooperation without sharing training information, cooperation with sharing training information. Results for low and high difficulty datasets are plotted in two separate charts. In addition to presenting the error for each individual approach in Figures 5.2 and 5.2, we averaged the error rates of all approaches in each category and displayed the mean values in the graphs.

An important observation that stands out is that average error rates for approaches that cooperated through sharing training information were lower than those that did not. This observation holds true for both types of datasets. The dotted line represents the median value of the error rates for all the categories. It appears that most or all of training methods that cooperated without sharing training information are grouped around or above the line, while the training methods that cooperated through sharing training information are located under the line. This again validates the advantages of using training information. No concrete conclusion can be drawn regarding the differences between the two categories with respect to data complexity. As discussed, in both Figures 5.1 and 5.2, the lowest error rate was obtained by CO-CDS and the highest error rate by RP.

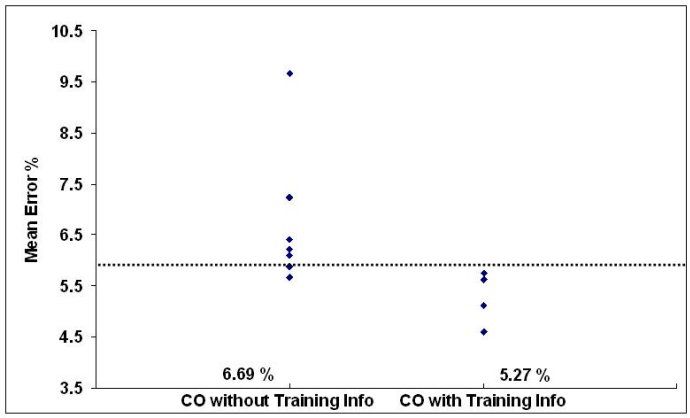


Figure 5.1: Mean error rates (%) for MCS training approaches categories (low-difficulty datasets)

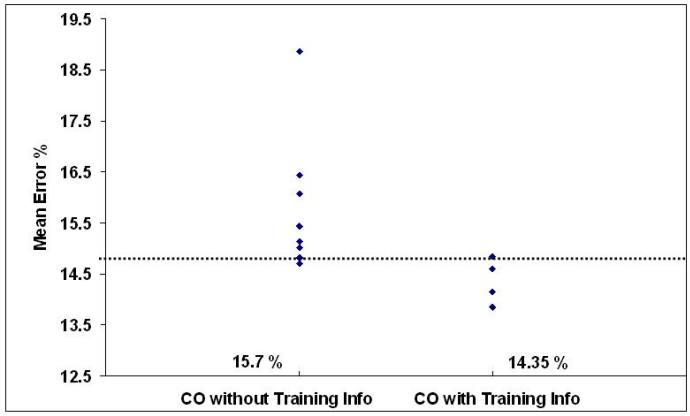


Figure 5.2: Mean error rates (%) for MCS training approaches categories (high-difficulty datasets)

5.3.2 Ensemble Stability and Generalization Error

The scatter plot of standard deviations and error rates for each ensemble method on a plane is depicted in Figures 5.3 and 5.4. Each point on these graphs represents the average error rate and standard deviation of all datasets, in a specific data category, for an ensemble approach. We treated RP as an outlier, since it had the highest error rate and standard deviation, and excluded it from these graphs.

In these Figures 5.3 and 5.4, we considered standard deviation as an indicator of variability of the results. Lower standard deviation shows less variability, or in other words, better ability of showing repeatable results. An interesting observation is that, for low-difficulty datasets, most of the training methods that operated through sharing information are clustered in the left corner of the scatter-plots (Figure 5.3). The left corner represents multiple classifier methods with more stable behavior. FBDF was the most stable training method and CO-CDS was the second stable approach. A-BST, Maj, and Ave were more stable for low-difficulty datasets than high-difficulty problems. Overall, three out of four training methods that cooperated by sharing training information presented stable behavior.

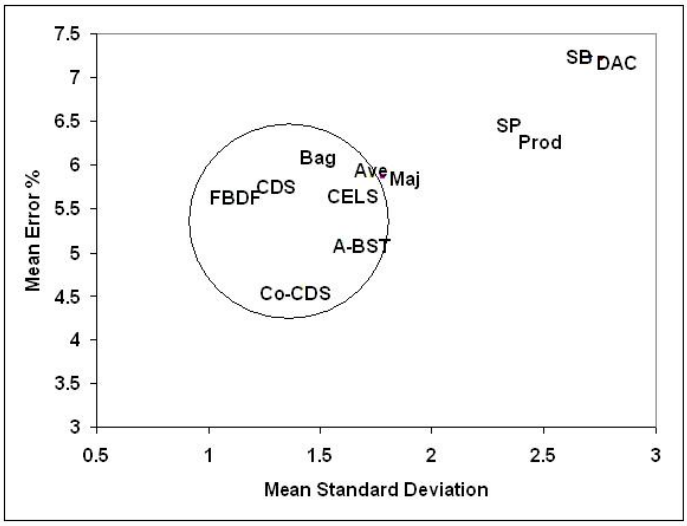


Figure 5.3: Scatter-plot of mean ensemble error rates (%) and standard deviation (low-difficulty datasets)

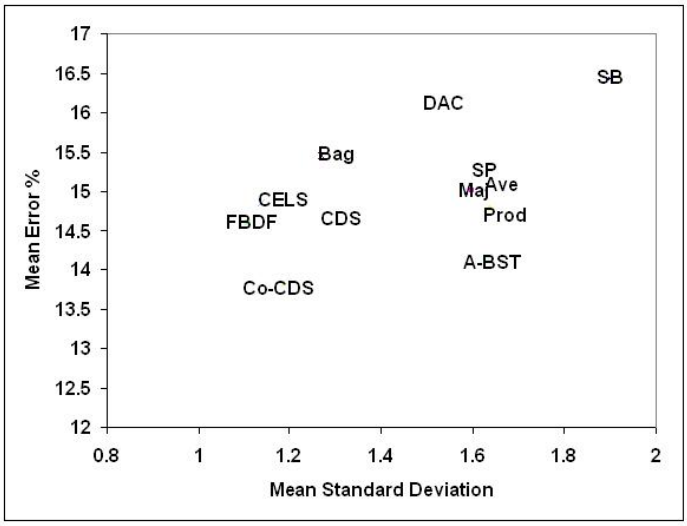


Figure 5.4: Scatter-plot of mean ensemble error rates (%) and standard deviation (high-difficulty datasets)

An accurate approach for evaluating MCS stability is using Equation (5.1). As mentioned before, we repeated the experiments 10 times for each ensemble approach. Using Equation 5.1, we examined the ability of generating identical classes for similar patterns in each iteration. We measured stability between two ensembles that were generated in sequence. Results for this analysis is presented in Figures 5.5 to 5.7 for Vehicle, Breast Cancer and 80-D Gaussian datasets. Each point in these figures indicates the mean, minimum and maximum value of stability measure for each ensemble approach. Error rates are depicted on x-axis.

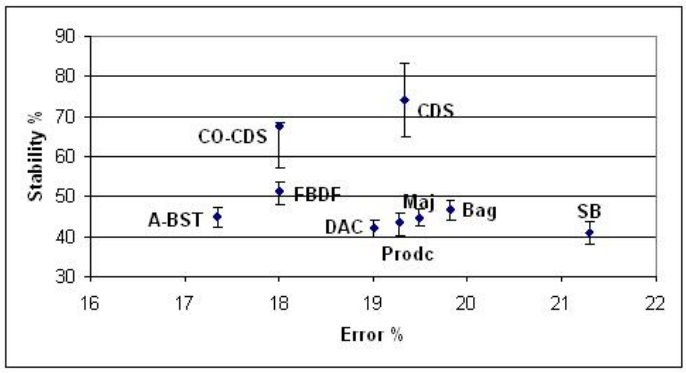


Figure 5.5: Stability vs Error for various ensemble methods (Vehicle dataset)

CDS was the most stable approach. This is most likely due the fact CDS approach selects the sub-optimal training partitions, in a large population of solutions, based on low distance-array values. The resultant training partitions have similar statistical characteristics compared to other random partitioning approaches. CO-CDS was the second stable approach. Since selection of sub-optimal training partitions in CO-CDS is dependent on out-of-bag classifier performance, this dependency results in extra instability among ensembles generated in different iterations. FBDF has also shown to be stable compared to other approaches. The reason for this stability lies in the way FBDF partitions the training data. FBDF partitions data by assigning a weight to each training pattern. Weights are calculated based on the performance of individual classifiers and the aggregation rule. Repartitioning of the

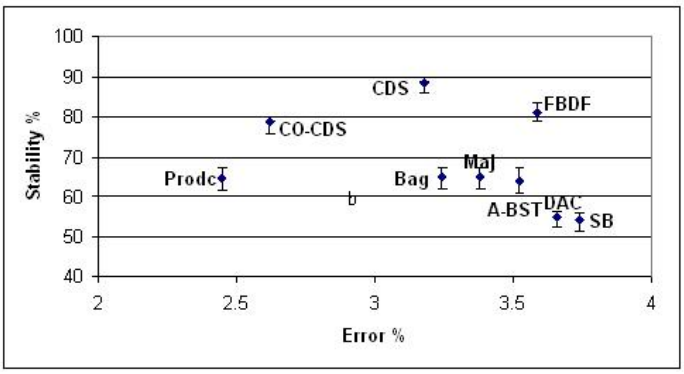


Figure 5.6: Stability vs Error for various ensemble methods (Breast Cancer dataset)

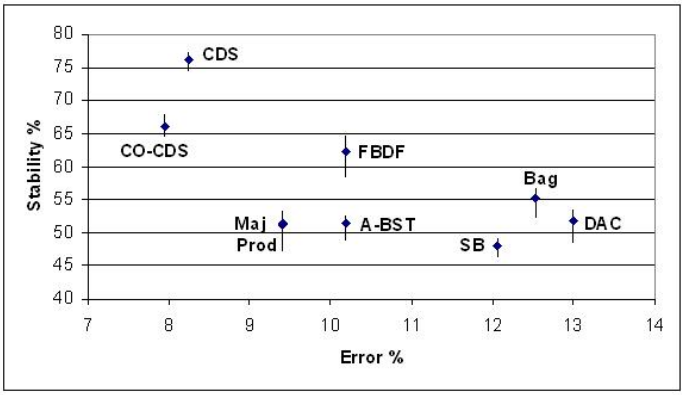


Figure 5.7: Stability vs Error for various ensemble methods (80-D Gaussian dataset)

Table 5.4: Test error (%) and standard deviation for various training methods (Breast Cancer datasets)

Aggregation Schemes	Fixed			Trained		
Training Methods	Maj	Prodc	Ave	SG	DT	Bay
Identical	3.38(0.81)	2.45(0.56)	3.38(0.81)	3.41(0.81)	3.39(0.65)	3.3(0.79)
Bootstrapping	3.24(0.16)	3.16(0.75)	3.33(0.69)	3.17(0.78)	3.04(0.95)	3.19(0.79)
CDS	3.1(0.31)	3.18(0.25)	2.97(0.33)	2.86(0.46)	2.55(0.41)	3.34(0.58)
CO-CDS	2.67(0.52)	2.62(0.6)	2.8(0.49)	2.31(0.67)	2.46(0.77)	3.08(0.77)

Table 5.5: Test error (%) and standard deviation for various training methods (Vehicle datasets)

Aggregation Schemes	Fixed			Trained		
Training Methods	Maj	Prodc	Ave	SG	DT	Bay
Identical	19.48(2.96)	19.27(2.28)	17.75(2.79)	18.65(2.62)	16.81(3.11)	18.2(2.38))
Bootstrapping	19.81(1.12)	17.88(3.2)	19.37(2.61)	16.92(2.57)	18.72(2.7)	18.48(2.41)
CDS	19.03(2.04)	19.34(3.36)	18.59(2.41)	17.1(1.98)	15.87(2.41)	18.01(2.03)
CO-CDS	18.12(2.39)	18.01(2.49)	17.83(2.65)	18.03(2.44)	16.58(2.61)	15.4(2.66)

data is performed several times until a stoping criterion is met. SB was the most unstable approach, but its difference from Maj, DAC, Bag, and A-BST was not significant.

5.3.3 Robustness and Efficiency

The robustness of some of the training methods used in this study was further explored by examining their impact on the performance of several ensemble methods. We considered three fixed ensemble methods including majority vote (Maj), product rule (Prodc), and average rule (Ave) as well as three trained ensemble methods: stacked generalizations, decision template and bayesian rule (see Section 2.4). The base classifiers in these ensemble methods were trained using identical training data, bootstrapping with replacement, CDS and CO-CDS. Experimental results for Breast Cancer, Vehicle and 80-D Gaussian datasets are presented in Tables 5.4 to 5.6. The lowest error rates for each aggregation scheme is underlined.

An obvious observation is that training the base classifiers with CO-CDS ap-

Table 5.6: Test error (%) and standard deviation for various training methods (80-D Gaussian datasets)

Aggregation Schemes	Fixed			Trained		
Training Methods	Maj	Prodc	Ave	SG	DT	Bay
Identical	9.4(1.11)	9.4(1.15)	9.52(1.1)	8.73(1)	8.99(0.79)	9.3(0.81)
Bootstrapping	12.53(1.5)	9.69(0.9)	9.29(0.97)	9.61(0.81)	8.17(1.13)	9.03(0.94)
CDS	8.43(0.9)	8.25(1.03)	8.13(0.81)	7.53(0.95)	7.22(1.01)	7.83(0.82)
CO-CDS	7.93(0.96)	7.95(0.85)	8.22(1.08)	6.32(0.99)	7.89(0.85)	8.02(0.97)

proach resulted in lower generalization error for both fixed and trained aggregation schemes. This observation was even more evident in the case of 80-D Gaussian dataset. Ensemble methods trained with CDS approach also illustrated better performances compared to the ones constructed on identical or bootstrapped training data. This finding demonstrates another advantage offered by CO-CDS and CDS approaches. Unlike some of the MCS training approaches such as A-BST, DAC and CELS, CO-CDS and CDS have the flexibility of being applied to any ensemble method.

5.4 Summary

The purpose of the comparative study presented in this chapter was to explore the strengths and the weaknesses of different MCS training methods. A summary of our observations is listed in the following.

- *CDS approach:* CDS performance was significantly better than divide-and-conquer, random partitioning, and stratified partitioning. In addition, its performance was slightly better than bagging and fixed combining approaches. This behavior could be related to the fact that CDS encourages extra classifier diversity, as compared to Bag, Maj, or other methods, while allowing high accuracy for the individual ensemble members. This extra classifier diversity can be related to non-overlapping data patterns (30% of each partition) that are highly uncorrelated in terms of class distribution. CDS approach does not depend on the classifier and it is simply a method for forming training partitions. Furthermore, it can be used to construct any ensemble method.

- *CO-CDS approach:* The process of selecting training partitions in CO-CDS is dependent on the output of a classifier. Unlike CELS, FBDF and A-BST, however, CO-CDS is a pre-classification process and does not depend on the architecture or the individual classifiers of the ensemble. CO-CDS only requires a computationally inexpensive classifier such as linear discriminant to generate out-of-bag classifiers. Our prime conclusion is that CO-CDS is a promising approach for generating sub-optimal training partitions on the datasets that we experimented with.

  The increase in accuracy using CO-CDS as compared to CDS was interesting. Although CO-CDS is computationally more demanding than CDS, this finding emphasizes that utilizing training information in the selection of training partitions pays off in terms of reduced generalization error. CO-CDS outperformed most of the MCS training approaches compared in this study. The combined effects of good data space coverage, efficient data sampling, and classifier diversity boosted CO-CDS generalization ability in comparison to other approaches. CO-CDS approach has the flexibility of training different ensemble method.

- *Cooperation through sharing training patterns:* The most obvious conclusion is that cooperation through sharing training patterns is vital to achieving accurate classifiers and ensemble systems. Such cooperation can be in any form of sharing training patterns, sharing the type of classes, or a combination of both.

- *Cooperation through sharing training information:* The advantages and disadvantages of employing training information for building MCS have been empirically investigated. Approaches that cooperate through sharing training information (A-BST,CO-CDS, FBDF, and CELS) were able to outperform other approaches on 11 of the 15 datasets. The presented results suggest that although some of the simple aggregation techniques have been shown to be effective, they are vulnerable to the incorrect information provided by the misclassified patterns. By sharing training information, such vulnerability may be avoided.

Chapter 6

Conclusions and Future Work

6.1 Contributions

In this thesis, we studied classifier ensemble from a new perspective "cooperation". We distinguished four levels of cooperation: decision, architecture, feature, and training. The degree and method by which classifier ensembles "share" resources were used as measures of cooperation. We narrowed down our focus on the training level and examined the effects of sharing training patterns and sharing training information on the performance of multiple classifiers. It is expected that this analysis will contribute to the advancement of our current understanding of MCS, its components and their interactions.

We developed two sets of measures to estimate class diversity and correlation among training partitions: class-based and feature-based. These evaluation measures enabled us to evaluate the degree and type of sharing among a set of disjoint and overlapped partitions. Class-based measures examine the issue of class diversity among training partitions. We empirically assessed the measures using several benchmark datasets. The results suggested that MCSs constructed on highly diverse training partitions have more accurate generalization ability. Feature-based measures, on the other hand, examine the correlation among training partitions. We developed and implemented several training partitioning strategies. Our aim in these proposed strategies was to generate overlapping partitions that represent different parts of the data-space so that MCS performance can be evaluated with

respect to various types of training data distribution. Empirical assessment of the feature-based measures showed that the combination of base classifiers which have a more accurate view of the problem space result in a superior performance.

These findings allowed us to reason about the interactions between training partitions and ensemble performance, and provided guidelines for the generation of sub-optimal partitions. With the aid of properly selected measures and training information, we proposed two new data partitioning approaches: CDS and CO-CDS. In CDS, training partitions are selected based on their close correlation. While in CO-CDS, in addition to correlation, performance of out-of-bag classifiers was also taken into consideration. A comprehensive comparative study was conducted where we compared our proposed training approaches with several other ensemble training methods. CO-CDS and A-BST (Ada-boosting) were among the best training approaches. Performance of CDS was marginally better than bagging and was similar to that of many fixed decision combining methods.

We categorized CDS, CO-CDS, and several existing MCS training approaches into two groups and compared them in terms of classification accuracy, classification stability, and their robustness for training different ensemble methods. Training approaches were categorized based on their functionality; whether they take into account training information provided by the classifier or the aggregation rule when training partitions are generated. The results of this comparative study suggest that cooperation is generally beneficial. Furthermore, it was shown that most of the ensemble approaches that cooperate through sharing information are firstly more accurate in terms of their ability of generalizing unseen data and secondly are more stable in terms of reproducing similar results. Among training approaches studied in this thesis, CDS and CO-CDS have shown to be more robust and efficient.

6.2 Future Work

The following issues remain open to future explorations:

- Despite various advantages of cooperation through sharing information, these approaches can be computationally more demanding. The trade-off between accuracy and computational efficiency of different approaches needs to be explored in future studies.

- While the success of CDS and CO-CDS validates the robustness of these training approaches, more research needs to be conducted to better understand and improve this behavior.

- Feature-based and class-based measures along with training information can be used as fitness functions for guiding the "selection" of sub-optimal training partitions using genetic algorithm. This approach can be compared with CO-CDS and other MCS training approaches.

- As pointed out previously, classifier diversity that is achieved by highly uncorrelated training partitions does not improve classifier ensemble performance. However, findings of this study suggest that there is a sweet spot for training partitions correlation at which classifier diversity is at its best. This issue can be explored further.

As discussed in Chapter 3, we distinguished four levels of cooperation in MCS. We only covered cooperation at training level in this thesis. Other levels and types of cooperations can be the future direction of this research:

- Cooperation at feature level: Feature partitioning has been shown to be an effective way to produce diverse classifiers. The same type of analysis at training level should be transferable to study cooperation at the feature level. Evaluation measures can be developed to estimate correlation among disjoint and overlapped feature subsets. The findings of such evaluation can be used to generate sub-optimal feature subsets for training MCS base classifiers.

- Cooperation at decision level: Decision level is perhaps the most extensively studied level among the other three. There has been some attempt to compare different decision combination techniques. However, none has examined combining methods from the cooperation perspective. A comparative study needs to be conducted to evaluate relative effectiveness of techniques that do not share classifier outputs (such as: maximum, minimum, and majority vote) versus techniques that share outputs (such as: averaging, weighted majority vote, decision template, and trained methods).

- Cooperation at architecture level: Since the significance of the above qualities may depend on the architecture used, they should be studied in conjunction

with several architectures. Sharing at training, feature and decision levels can be considered with different architectures to examine the circumstances under which they are most effective.

6.3 Publications Resulting from this Work

Journal

1. Nayer Wanas, Rozita Dara, and Mohamed Kamel, "Adaptive Fusion and Cooperative Training for Classifier Ensembles," *Pattern Recognition*, vol 39, no. 9, pp. 1781–1794, 2006.

2. Rozita Dara, Masoud Makrehchi, and Mohamed Kamel, "Data Partitioning for Training Multiple Classifier Systems," *IEEE Trans. on Knowledge and Data Engineering,* (under review).

3. Rozita Dara, Nayer Wanas, and Mohamed Kamel, "Data Dependence in Multiple Classifier Systems," *Pattern Recognition,* (under review).

4. Rozita Dara and Mohamed Kamel, "Cooperation in Multiple Classifier Systems," (to be submitted).

Conference

1. Rozita A. Dara, Hanan Ayad, Yanmin Sun, and Mohamed Kamel, "Ensemble Methods for Knowledge Discovery in LORNET Data," *Proc. of the 2th Annual Scientific Conference - LORNET Research Network,* Vancouver, Canada, 2005.

2. Rozita A. Dara, and Masoud Makrehchi, and Mohamed S. Kamel, "Data Partitioning Evaluation Measures for Classifier Ensemble," *Proc. of 6th Int. Workshop on Multiple Classifier Systems,* Monterey Bay, California, USA, 2005.

3. Rozita A. Dara, and Mohamed S. Kamel, "Effect of Sharing Training Patterns on the Performance of Classifier Ensemble," *Proc. of the Int. IEEE Conf. on System, Man, and Cybernetics,* The Hague, The Netherlands, pp 4826–4831, 2004.

4. Rozita A. Dara, and Masoud Makrehchi, and Mohamed S. Kamel, "An information-Theoretic Measure to Evaluate Data Partitions in Multiple Classifiers," *Proc. of Int. IEEE Conf. On Systems, Man, and Cybernetics,* The Hague, The Netherlands, pp. 1220–1225, 2004.

5. Rozita A. Dara, and Mohamed S. Kamel, "Sharing Training Patterns in Neural Network Ensembles," *Proc. of IEEE Int. Joint Conf. on Neural Networks (IJCNN),* Budapest, Hungry, 2004.

6. Nayer Wanas, Rozita A. Dara, and Mohamed S. Kamel, "Co-operative Training in Classifier Ensembles," *Proc. of the 7th Int. Conf. on Information Fusion,* Stockholm, Sweden, pp. 74–78, 2004.

7. Rozita A. Dara, and Mohamed S. Kamel, "Sharing Training Patterns among Multiple Classifiers," *Proc. of the 5th Int. Workshop on MCS,* LNCS 3077, Cagliari, Italy, pp. 243-252, 2004.

Other Publications

1. Hamind Tizhoosh, Farhang Sahba, and Rozita Dara, "Naive Bayesian Approach to Poem Recognition," *Pattern Recognition Letters,* (under Review).

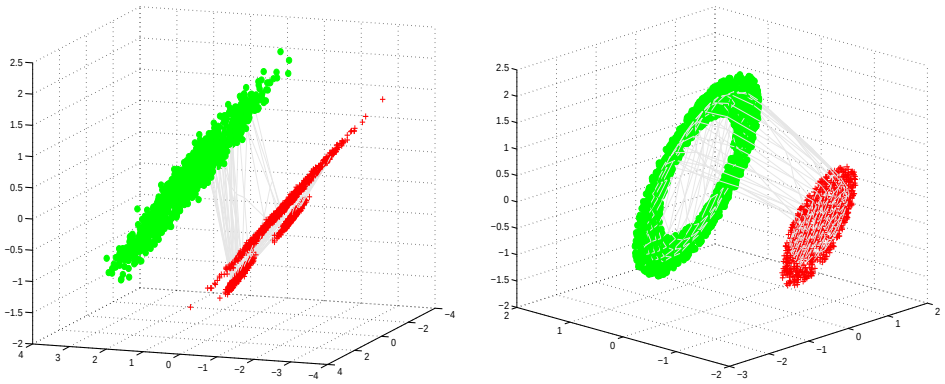
2. Hamid Tizhoosh, and Rozita Dara, "On Poem Recognition," *Pattern Analysis and Applications,* vol. 9, no. 4, pp.325– 338, 2006.

Appendix A

Additional Results

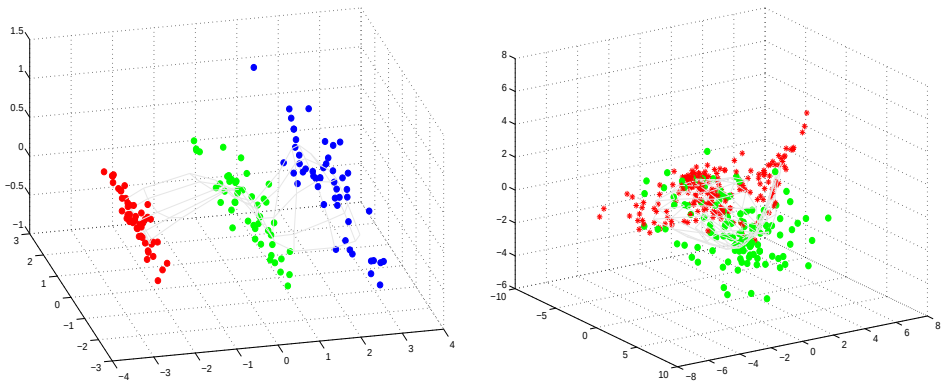
A.1 Data Difficulty

We applied Self-organizing Map [63] to all the datasets to highlight complexity of the data in terms of class compactness, isolation or linear class separability in a two dimensional space (Figs. A.1-A.8). We grouped the datasets according to their degree of difficulty into high and low (Table 4.3).The degree of difficulty was determined based on data dimensionality, number of classes, and class separability highlighted in Figures A.1-A.8. Regardless of data dimensionality, all the datasets were projected in a two dimensional SOM space, some datasets happened to have highly separable classes (e.g. Clouds, Iris, and Concentric), while some others did not such as Phoneme, German, and Vowel.



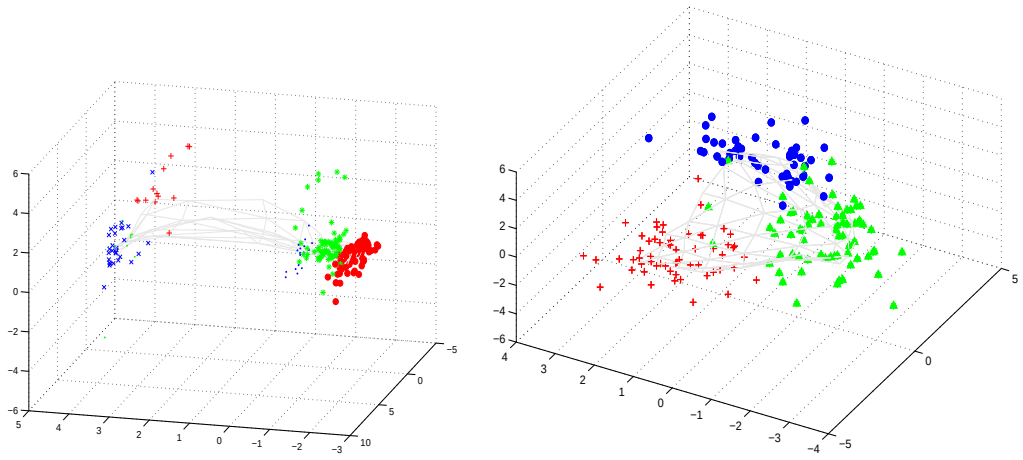
(a) (b)

Figure A.1: Datasets: a) Clouds, b) Concentric



(a) (b)

Figure A.2: Datasets: a) IRIS, b) Ionosphere



(a) (b)

Figure A.3: Datasets: a) Glass, b) Wine

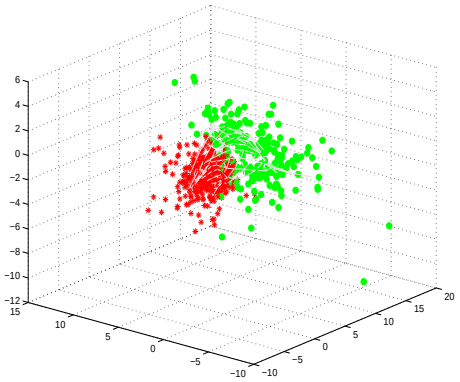
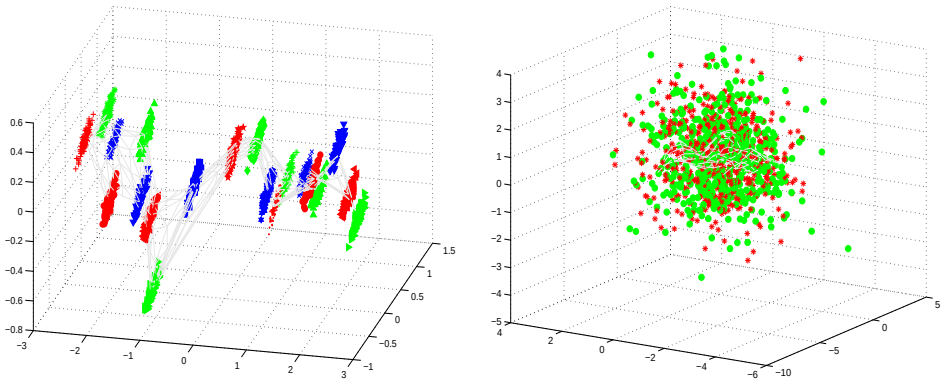
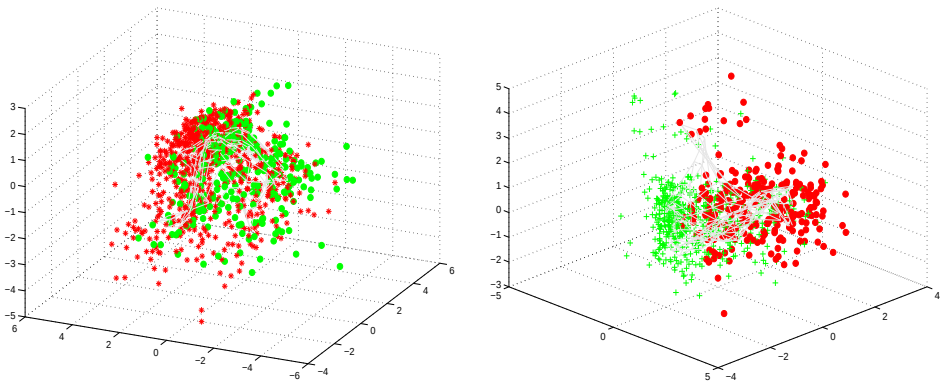


Figure A.4: Datasets: Breast Cancer



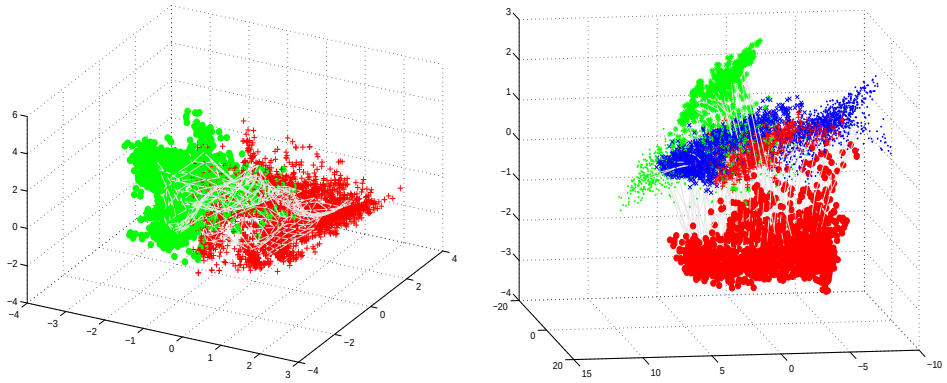
(a) (b)

Figure A.5: Datasets: a) 20 Class Gaussian, b) 80-D Gaussian



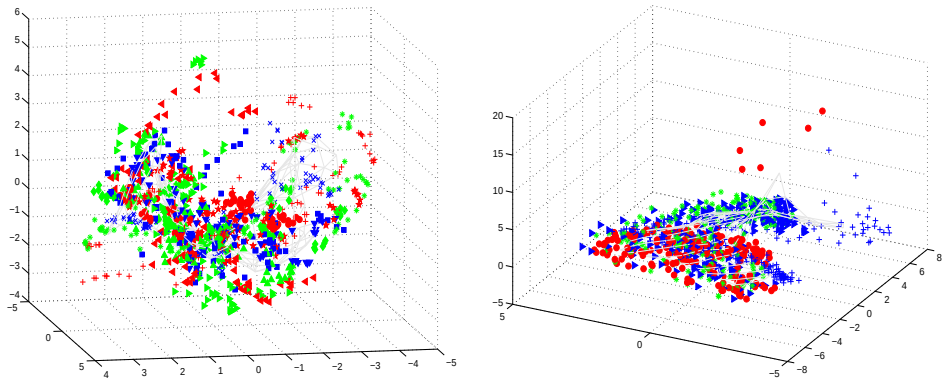
(a) (b)

Figure A.6: Datasets: a) German, b) Pima Indians Diabetes



(a) (b)

Figure A.7: Datasets: a) Phoneme, b) Satimage

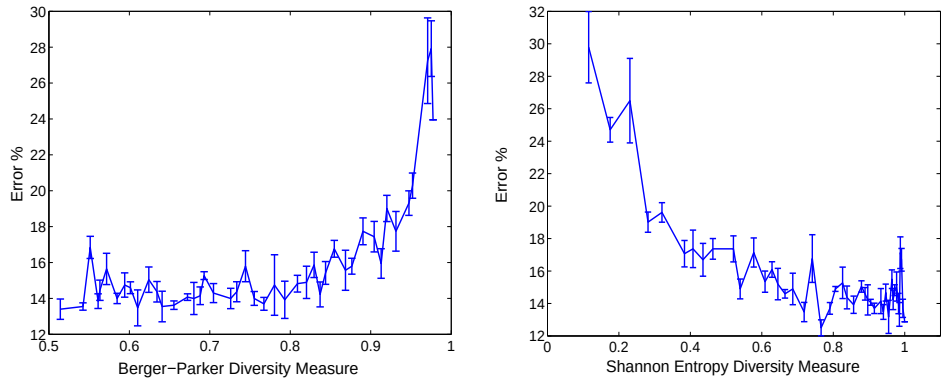


(a) (b)

Figure A.8: Datasets: a) Vowel b) Vehicle

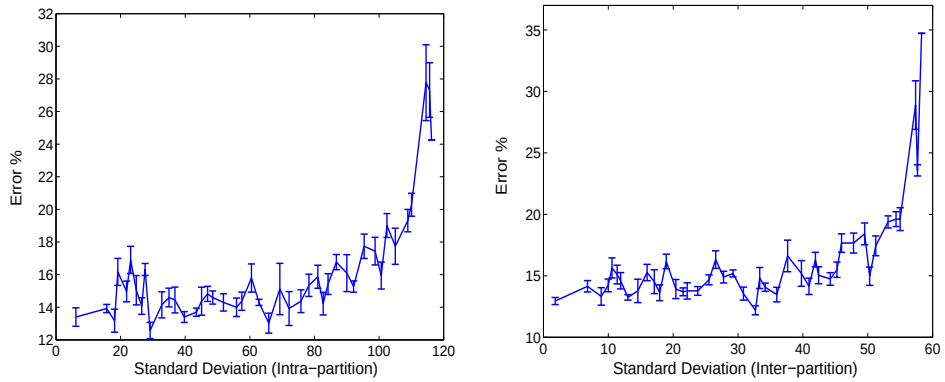
A.2 Disjoint Partitioning

A.2.1 Class-based Results



(a) (b)

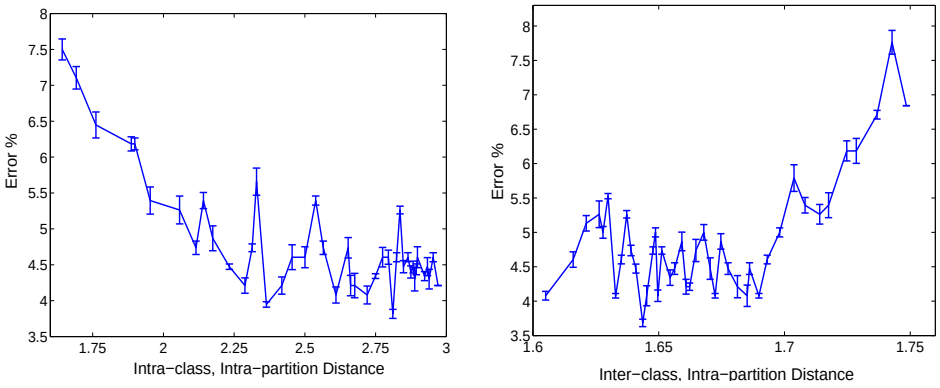
Figure A.9: 80-D Gaussian (Disjoint): a) Berger-Parker, b) Shannon Entropy



(a) (b)

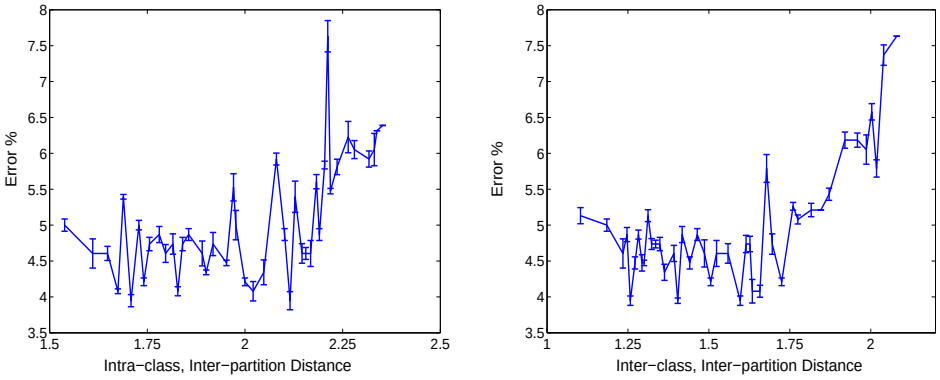
Figure A.10: 80-D Gaussian (Disjoint): a) STD Inter-class/intra-partition b) STD Intra-class/inter-partition

A.2.2 Feature-based Results



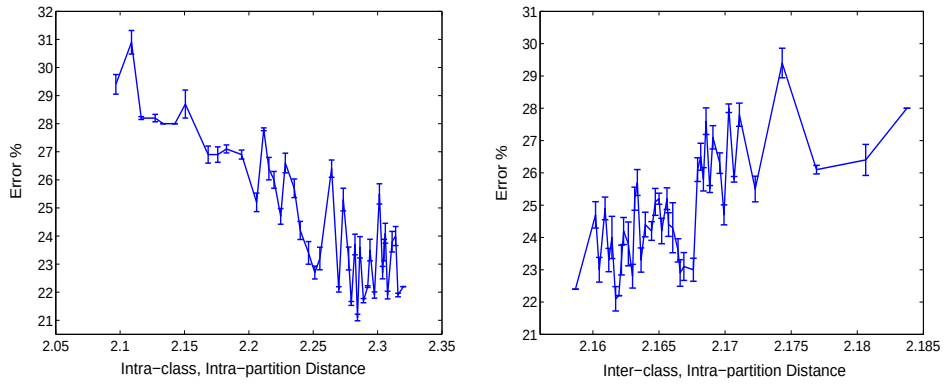
(a) (b)

Figure A.11: Breast Cancer (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



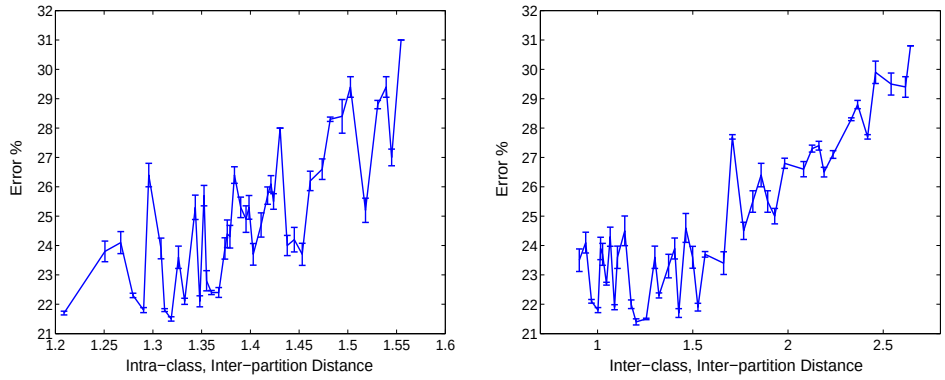
(a) (b)

Figure A.12: Breast Cancer (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure A.13: German (Disjoint): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition

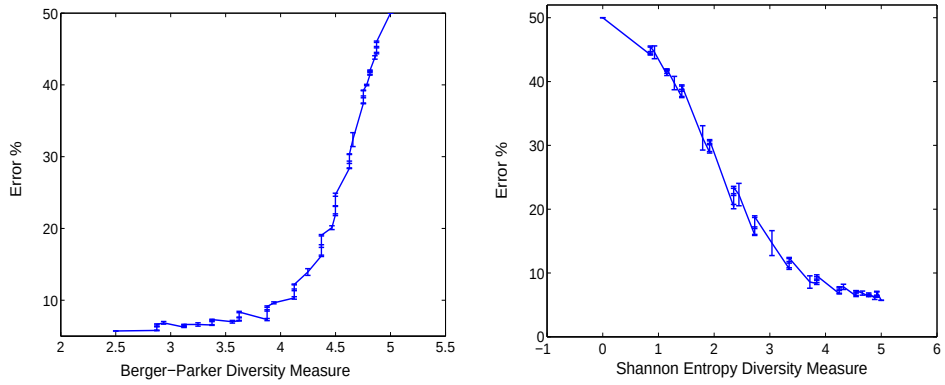


(a) (b)

Figure A.14: German (Disjoint): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

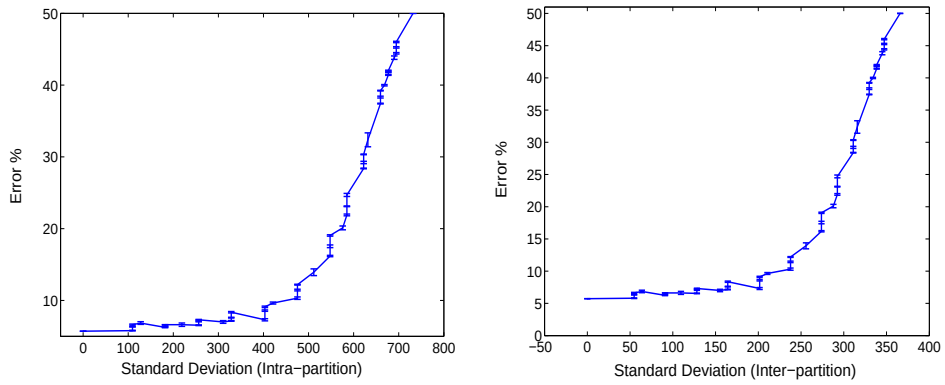
A.3 Overlapped Partitioning

A.3.1 Class-based Results



(a) (b)

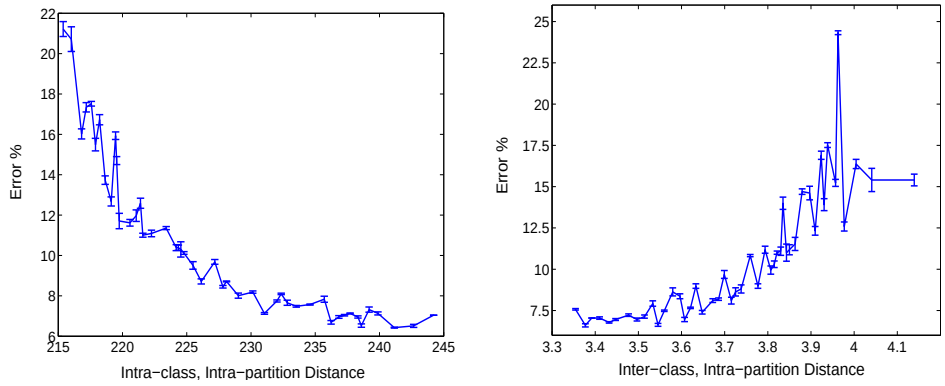
Figure A.15: 80-D Gaussian (Overlapped): a) Berger-Parker, b) Shannon Entropy



(a) (b)

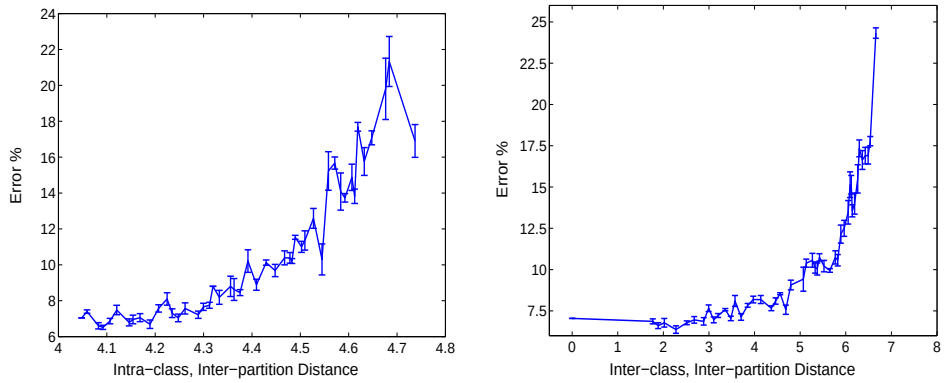
Figure A.16: 80-D Gaussian (Overlapped): a) STD Inter-class/intra-partition b) STD Intra-class/inter-partition

A.3.2 Feature-based Results



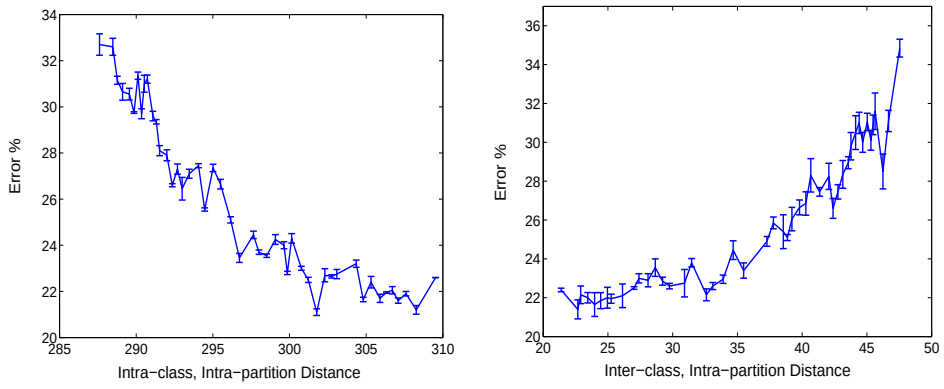
(a) (b)

Figure A.17: Breast Cancer (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



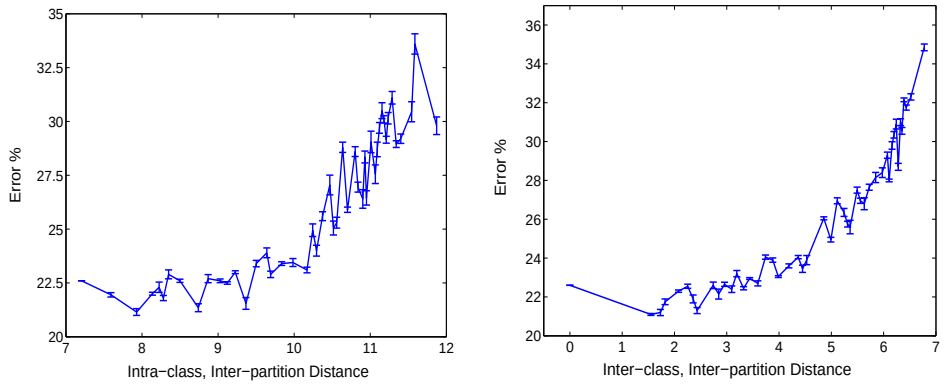
(a) (b)

Figure A.18: Breast Cancer (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition



(a) (b)

Figure A.19: German (Overlapped): a) Intra-class/Intra-partition, b) Inter-class/Intra-partition



(a) (b)

Figure A.20: German (Overlapped): a) Intra-class/Inter-partition b) Inter-class/Inter-partition

Bibliography

[1] F. Alkoot, and J. Kittler, "Experimental evaluation of experts fusion strategies," *Pattern Recognition Letters,* vol. 20, pp. 1361-1369, 1999.

[2] K. Ali, and M. Pazzani, "Error reduction through learning multiple descriptions," *Machine Learning,* vol. 24, pp. 173–202, 1996.

[3] N. Arshadi, and I. Jurisica, "Data mining for case-based reasoning in high-dimensional biological domains," *IEEE Trans. on Knowledge and Data Engineering,* Vol. 17, No. 8, pp. 1127–1137, 2005.

[4] J. Asker, and R. Maclin, "Ensemble as a sequence of classifiers," *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence,* vol. 2, pp. 860-865, 1997.

[5] G. Auda, and M. Kamel, "Modular neural networks: a survey," *Journal of Neural Systems*, vol. 9, pp. 129–151, 1999.

[6] G. Auda, and M. Kamel, "CMNN: Cooperative Modular Neural Networks for pattern recognition," *Pattern Recognition Letters*, vol. 18, no. 11-13, pp. 1391–1398, 1997.

[7] D.A. Bell, J.W. Guan, and Y. Bi, "On combining classifier mass functions for text categorization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 17, no. pp. 1307–1319, 2005.

[8] P. Bennett, S. Dumais, and E. Horvitz, "The combination of text classifiers using reliability indicators," *Information Retrieval*, vol. 8, no. 1, pp. 67–100, 2005.

[9] W.H. Berger, and F.L. Parker, "Diversity of planktonic foraminifera in deep sea sediments," *Science*, pp. 1345-1347, 1970.

[10] *Benchmark Repository*, [http://users.rsise.anu.edu.au/ raetsch/data/].

[11] J. Bezdek, and S. Pal, "Fuzzy models for pattern recognition," IEEE press, New Jersey, 1991.

[12] D. Blake, and C. Merz, *Repository of machine learning databases*, [http://www.ics.uci.edu/ mlearn/MLRepository.html].

[13] A. Blum, and P. Langley, "Selection of relevant features and examples in machine learning", *Artificial Intellignce*, vol. 97, no. 1-2, pp. 245–271, 1997.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] L. Breiman, "Pasting small votes for classification in large databases and online," *Machine Learning*, vol. 36, no. 1-2, pp. 85-103, 1999.

[16] L. Breiman, "Combining predictors," Combining Artificial Neural Nets. Springer-Verlag, London, 1999.

[17] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[18] N. Chawla, T. Moore, L. Hall, L. Bowyer, P. Kegelmeyer, and C. Springer, "Distributed learning with bagging-like performance," *Pattern Recognition Letters*, vol. 24, pp. 455–471, 2003.

[19] L. Chen and M. Kamel, "Design of Multiple Classifier Systems for Time Series Data," in *Proc. of 6th Int. Workshop on Multiple Classifier System*, pp. 216–225, 2005.

[20] C.H. Cheng, A.W. Fu, and Y. Zhang, "Entropy-based subspace clustering for mining numerical data," in *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pp. 84-93, 1999.

[21] F. Chu, Y. Wang, C. Zaniolo, D.S. Parker, "Improving mining quality by exploiting data dependency," *Proc. of the 9th Int. Conf. on Principles and Practice of Knowledge Discovery in Databases*, pp. 486–499, 2005.

[22] R. Cilibrasi and P. Vitnyi, "Clustering by compression", *IEEE Trans. on Information Theory,* vol. 51, no. 4, pp. 1523–1545, 2005.

[23] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning,* vol. 15, no. 2, pp. 201–221, 1994.

[24] B. Dasarathy (Ed.) Multisensor, multisource information fusion: architectures, algorithms and applications. Society of Photo-Optical Instrumentation Engineers, 2005.

[25] B. Dasarathy, Decision Fusion. IEEE Computer Society Press, 1994.

[26] T. Dietterich, The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, Ensemble learning, 2002.

[27] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning,* vol. 40, no. 2, pp.139-157, 2000.

[28] T. Dietterich, "Ensemble methods in machine learning," in *Proc. of 1th Int. Workshop on Multiple Classifier System*, pp. 1–15, 2000.

[29] T. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–158, 2000.

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

[31] V. Dmitryuk and K. Dmitryuk, "Data-dependent classifier fusion for construction of stable effective algorithms," *Proc. of the 17th IEEE Int. Conf. on Pattern Recognition*, vol 1, pp. 144–147, 2004.

[32] R. Duda, P. Hart, and D. Strok, Pattern Recognition (2nd ed.), John Wiley and Sons, 2000.

[33] R. Duin, "The combining classifier: to train or not to train?," in *Proc. of 16th Int. Conf. on Pattern Recognition*, pp. 765–770, 2002.

[34] R.P. Duin, P. Juszczak, P. Paclik, E. Pekalska, and D. de Ridder, D.M.J. Tax, PRTools3, a Matlab toolbox for Pattern Recognition [http://prtools.org], Delft Univ. of Techn., 2003.

[35] *ELENA Project*, [http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm].

[36] T. Evgeniou, M. Pontil, and A. Elisseeff, "Leave one out error, stability, and generalization of voting combinations of classifiers," *Machine Learning,* vol. 55, no. 1, pp. 71–97, 2004.

[37] Y. Freund Y, and R. Schapire, "Experiments with a new boosting algorithm " in *Proc. of the 13th Int. Conf. on Machine Learning*, pp. 148–156, 1996.

[38] D. Frosyniotis, A. Stafylopatis, and A. Likas, "A divide-and-conquer method for multi-net classifiers," *Pattern Analysis and Applications*, vol. 6, pp. 32–40, 2002.

[39] K. Fukunaga, Introduction to Statistical Pattern Recognition (2th ed.), Oxford University Press, 1990.

[40] H. Fu, C. Chiang, Y. Lee, and H. Pao, "Divide-and-conquer learning schemes and modular perceptron networks," *IEEE Trans. on Neural Networks,* vol. 12, no. 2, 2001.

[41] P. Gader, M. Mohamed, and J. Keller, "Fusion of handwritten word classifiers", *Pattern Recognition Letters,* vol. 17, pp. 577–584, 1996.

[42] J. Gama, and P. Brazdil, "Cascade generalization," *Machine Learning,* vol. 41, no. 3, pp. 315–343, 2000.

[43] G. Giancinto, and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition,* vol. 34, pp. 1879–1881, 2001.

[44] J. Ghosh, "Multiclassifier systems: back to the future," *Proc. of 3th Int. Workshop on Multiple Classifier Systems*, LNCS 2364, pp. 1–15, 2002.

[45] D. Goldberg, Genetic Algorithms, Addison Wesley, 1988.

[46] M. Grabisch, "On equivalence classes of fuzzy connectives - the case of fuzzy integrals," *IEEE Trans. on Fuzzy Systems,* vol. 3, no. 1, pp. 96–109, 1995.

[47] S. Hashem, "Optimal linear combinations of neural networks", *Neural Networks*, vol. 10, no. 4, pp. 599–614, 1997.

[48] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd edition, Prentice Hall, 1999.

[49] T. Ho, "Multiple Classifier Combiation: Lessons and Next Steps." HybridMethods in Pattern Recognition. World Scientific Press, 2002.

[50] T. Ho, "Data complexity analysis for classifier combination," *Proc. of 2th Int. Workshop on Multiple Classifier Systems,* LNCS 2096, pp. 53–67, 2001.

[51] T. Ho, "Random subspace method for constructing decision forests," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[52] T. Ho, J. Hull, S. Srihari, "Decision combination in multiple classifier systems", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66–75, 1994.

[53] A.K. Jain, P.R.W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. on Pattern Analysis and Machine Intelligence* vol. 22, no. 1, pp. 4–37, 2000.

[54] R. Jin, L. Si, Y. Liu, A. Hauptmann, and J. Carbonell, "A new boosting algorithm using input dependent regularizer," *Proc. of the 20th Int. Conf. on Machine Learning*, 2003.

[55] W. Jiang, and M. Tanner "Hierarchical mixtures of experts for generalized linear models," *Neural Computation*, vol. 11, pp. 1183–1198, 1999.

[56] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

[57] M. Kamel, N. Wanas, "Data dependence in combining classifiers," in *Proc. 4th Int. Workshop on Multiple Classifier System*, pp. 1–14, 2003.

[58] S.W, Kim and B.J. Oommen, "On using prototype reduction schemes and classifier fusion strategies to optimize kernel-based nonlinear subspace methods," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 27, no. 3, pp. 455–460, 2005.

[59] J. Kittler, F. Roli, and T. Windeatt (eds), "Multiple Classifiers Systems," *Proc. 5th Int. Workshop on MCS*, LNCS 3077, Springer-Verlag GmbH, 2004.

[60] J. Kittler, and F. Roli (Eds.), "Multiple Classifiers Systems," *Proc. 4th Int. Workshop on MCS*, LNCS 2709, Springer-Verlag GmbH, 2003.

[61] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On Combining Classifiers," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[62] A. Koerich, R. Sabourin and C. Suen, "Recognition and verification of unconstrained handwritten words," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1509–1522, 2005.

[63] T. Kohonen, "Self-Organizing Maps", 2001, 3d Edition, Springer.

[64] S. Kumar, J. Ghosh, and M. Crawford, "Hierarchical fusion of multiple classifiers for hyperspectral data analysis," *Pattern Analysis and Applications,* vol. 5, pp. 210–220, 2002.

[65] L. Kuncheva and J.J. Rodriguez, "Classifier ensembles with a random linear oracle," *IEEE Trans. on Knowledge and Data Engineering,* vol. 19, no. 4, pp. 500–508, 2007.

[66] L. Kuncheva, and D. Vetrov, "Evaluation of stability of k-Means cluster ensembles with respect to random initialization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1798–1808, 2006.

[67] L. Kuncheva (ed.), "Diversity in multiple classifier systems", *Information Fusion*, vol. 6, no. 1, 2005.

[68] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley and Sons, 2004.

[69] L. Kuncheva, "That elusive diversity in classifier ensembles," in *Proc. of 1th Iberian Conf. on Pattern Recognition and Image Analysis*, LNCS 2652, pp. 1126–1138, 2003.

[70] L. Kuncheva, and C. Whitaker, "Measures of diversity in classifier ensembles," *Machine Learning*, vol. 51, pp. 181–207, 2003.

[71] L. Kuncheva, "Combining classifiers by clustering, selection and decision templates: an experiment," *IEEE Trans. on Systems Man and Cybernetics,* vol. 32, no. 2, pp. 146–156, 2002.

[72] L. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 2, pp. 281–286, 2002.

[73] L. Kuncheva, "Genetic algorithm for feature selection for parallel classifiers," *Information Processing Letters*, vol. 46, pp. 163–168, 1997.

[74] L. Lam, "Classifier combinations: implementations and theoretical issues," *Proc. of 1th Int. Workshop on Multiple Classifier Systems*, pp. 77–86, 2000.

[75] L. Lam, and C. Suen, "Optimal combination of pattern classifiers," *Pattern Recognition Letters,* vol. 16, pp. 945–954, 1995.

[76] A. Lazarevic and Z. Obradovic, "Boosting algorithms for parallel and distributed learning," *Distributed and Parallel Databases Journal,*, vol. 11, pp. 203-229, 2002.

[77] A. Lipnickas, "Classifier fusion with data-dependent aggregation schemes," *Proc. of the 7th Int. Conf. on Information Netowrks, Systems and Technologies*, pp. 147–153, 2001.

[78] R. Linares, G. Mateo, and A. Castro, "On combining classifiers for speaker authentication," *Pattern Recognition*, vol. 36, no. 2, pp. 347–359, 2003.

[79] Y. Liu, and X. Yao, "A cooperative ensemble learning system", *Proc. of the Int. Joint Conf, on Neural Networks*, Anchorage, AK, USA, pp. 2202–2207, 1998.

[80] D. Mashao and M. Skosan, "Combining classifier decisions for robust speaker identification," *Pattern Recognition,* vol 39, no 1, pp. 147–155, 2006.

[81] N.C. Oza, R. Polikar, J. Kittler, and F. Roli (eds) "Multiple Classifiers Systems," *Proc. 6th Int. Workshop on MCS*, LNCS 3541, Springer-Verlag GmbH, 2005.

[82] D. Parikh, M. Kim, J. Oagaro , S. Mandayam and R. Polikar, "Combining classifiers for multisensor data fusion," *Proc. of IEEE Int. Conf. on System Man Cybernetics*, pp. 1232–1237, 2004

[83] P. Paclk , T. Landgrebe , D. Tax, and R. Duin, "On deriving the second-stage training set for trainable combiners," *Proc. 6th Int. Workshop on MCS*, LNCS 3541, pp. 136-146, 2005.

[84] D. Partridge, and W. Yates, "Engineering multiversion neural-net systems," *Neural Computation,* vol. 8 pp. 869–893, 1996.

[85] B. Pramanto, P. Munro, and H. Doyle, "Improving committee diagnosis with resampling techniques," *Advances in Neural Information Processing Systems,* vol. 8, pp. 882–888, 1996.

[86] J. Rodriguez, L. Kuncheva, and C. Alonso, "Rotation forest: a new classifier ensemble method," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 28, no. 10, pp. 1619–1630, 2006.

[87] F. Roli, "Semi-supervised multiple classifier systems: background and research directions" *Proc. of 6th Int. Workshop on Multiple Classifier Systems*, LNCS 3541, pp. 1-11, 2005.

[88] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," *Proc. 2th Int. Workshop on MCS,* pp. 78–87, 2001.

[89] D. Ruta, and B. Gabrys, "An overview of classifier fusion methods," *Computer and Information Systems,* vol. 7, no. 1, pp. 1–10, 2000.

[90] L.G. Rueda and B.J. Oommen, "On optimal pairwise linear classifiers for normal distributions: the d-dimensional case," *Pattern Recognition,* vol. 36, pp. 13 23, 2003.

[91] A. Sharkey, "Types of multinet systems," *Proc. of 3th Int. Workshop on Multiple Classifier Systems*, LNCS 2364, pp. 108–117, 2002.

[92] A. Sharkey, N. Sharkey, U. Gerecke, and G. Chandroth, "The "Test and Select" approach to ensemble combination," *Proc. of 1th Int. Workshop on Multiple Classifier Systems*, LNCS 1857, pp. 30–44, 2000.

[93] A. Sharkey, Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, Springer-Verlag, 1999.

[94] A. Sharkey, and N. Sharkey, "Combining Diverse Neural Nets," *The Knowledge Engineering Review*, vol. 12, no. 3, pp. 231-247, 1997.

[95] D. Skalak, "Prototype selection for composite nearest neighbor classifiers," Ph.D. dissertation. Dept. of Computer Science, Technical Report 96-89, University of Massachusetts, Amherst, Massachusetts.

[96] M. Skurichina, L. Kuncheva, and R. Duin, "Bagging and boosting for the nearest mean classifier: effects of sample size on diversity and accuracy," *Proc. of 3th Int. Workshop on Multiple Classifier Systems*, LNCS 2364, pp. 62–71, 2002.

[97] X. Song, Y. AbuMostafa, J. Sill, H. Kasdan, and M. Pavel, "Robust image recognition by fusion of contextual information," *Information Fusion,* vol. 3, pp. 277–287, 2002.

[98] C.Y. Suen, and L. Lam, "Multiple classifier combination methodologies for different output levels," *Proc. of 1th Int. Workshop on Multiple Classifier Systems,* LNCS 1857, pp. 52–66, 2000.

[99] K. Toh, and A. Wei-Yun Yau, "Combination of hyperbolic functions for multimodal biometrics data fusion," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 34, no. 4, pp. 1196–1209 , 2004.

[100] K. Tumer, and N. Oza, "Input decimated ensembles," *Pattern Analysis and Applications*, vol. 6, no. 1, pp. 65–77, 2003.

[101] K. Tumer, and J. Ghosh, "Estimating the bayes error rate through classifier combining," *Proc. of the 13th Int. Conf. on Pattern Recognition*, vol. 2, pp. 695–699, 1996.

[102] N. Ueda, "Optimal linear combination of neural networks for improving classification performance", *Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 207–215, 2000.

[103] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis, "Soft combination of neural classifiers: a comparative study," *Pattern Recognition Letters*, vol. 20, pp. 429–444, 1999.

[104] N. Wanas, and M. Kamel, "Feature based decision fusion in ensemble of neural networks," *Pattern Recognition Letters,* vol. 20, no. 11, pp. 1353-1359, 1999.

[105] G. Webb, Z. Zheng, "Multistrategy ensemble learning: reducing error by combining ensemble learning techniques," *IEEE Trans.on Knowledge and Data Engineering,* Vol. 16, No. 8, pp. 980–991, 2004.

[106] D. Windridge, and J. Kittler, "Combining classifier optimization via feature selection", *Proc. 1th Int. Workshop on MCS*, LNCS 1867, pp. 687–695, 2000.

[107] D. Wolpert, "Stacked generalization", *Neural Networks*, vol. 5, pp. 241-259, 1992.

[108] K. Woods, W. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[109] L. Xu, A. Krzyzak, and C.Y. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.