A New Design of Multiple Classifier System and its Application to Classification of Time Series Data

by

Lei Chen

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer 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.

Abstract

To solve the challenging pattern classification problem, machine learning researchers have extensively studied Multiple Classifier Systems (MCSs). The motivations for combining classifiers are found in the literature from the statistical, computational and representational perspectives. Although the results of classifier combination does not always outperform the best individual classifier in the ensemble, empirical studies have demonstrated its superiority for various applications.

A number of viable methods to design MCSs have been developed including bagging, adaboost, rotation forest, and random subspace. They have been successfully applied to solve various tasks. Currently, most of the research is being conducted on the behavior patterns of the base classifiers in the ensemble. However, a discussion from the learning point of view may provide insights into the robust design of MCSs. In this thesis, Generalized Exhaustive Search and Aggregation (GESA) method is developed for this objective. Robust performance is achieved using GESA by dynamically adjusting the trade-off between fitting the training data adequately and reducing the risk of overfitting. Besides its learning algorithm, GESA is also distinguished from traditional designs by its architecture and level of decision-making. GESA generates a collection of ensembles and dynamically selects the most appropriate ensemble for decision-making at the local level.

Although GESA provides a good improvement over traditional approaches, it is not very data-adaptive. A data-adaptive design of MCSs demands that the system can adaptively select representations and classifiers to generate effective decisions for aggregation. Another weakness of GESA is its high computation cost which prevents it from being scaled to large ensembles. Generalized Adaptive Ensemble Generation and Aggregation (GAEGA) is an extension of GESA to overcome these two difficulties. GAEGA employs a greedy algorithm to adaptively select the most effective representations and classifiers while excluding the noise ones as much as possible. Consequently, GAEGA can generate fewer ensembles and significantly reduce the computation cost. Bootstrapped Adaptive Ensemble Generation and Aggregation (BAEGA) is another extension of GESA, which is similar with

GAEGA in the ensemble generation and decision aggregation. BAEGA adopts a different data manipulation strategy to improve the diversity of the generated ensembles and utilize the information in the data more effectively.

As a specific application, the classification of time series data is chosen for the research reported in this thesis. This type of data contains dynamic information and proves to be more complex than others. Multiple Input Representation-Adaptive Ensemble Generation and Aggregation (MIR-AEGA) is derived from GAEGA for the classification of time series data. MIR-AEGA involves some novel representation methods that proved to be effective for time series data.

All the proposed methods including GESA, GAEGA, MIR-AEGA, and BAEGA are tested on simulated and benchmark data sets from popular data repositories. The experimental results confirm that the newly developed methods are effective and efficient.

Acknowledgments

I wish to express sincere thanks to all the people who have provided invaluable help and support in my thesis research.

I would like to thank my supervisor, Professor Mohamed S. Kamel. He has generously helped me in my Ph.D program. I have learned much from him both in knowledge and in research attitude. Without his encouragement, guidance, and support, I would not have accomplished my thesis research. I owe a great deal of gratitude to him.

I also want to express my appreciation to Dr. Reda Alhajj for serving as my external examiner. Special thanks are due to Dr. Mohamed-Yahia Dabbagh, Dr. Otman Basir, and Dr. Ali Ghodsi for serving on the exam committee. Their comments and suggestions have greatly improved my work.

I am greatly indebted to my parents and my wife, Ni Ding. She has encouraged me, believed in me, helped me, and supported me. Without her love and support, I would not have been able to complete my work.

Dedication

This is dedicated to Ni Ding, my beloved wife.

Contents

List of Tables

List of Figures

List of Abbreviations

ARMA	Autoregressive Moving Average
BKS	Behavior Knowledge Space
BP	Back Propagation
CART	Classification And Regression Tree
DES	Double Exponential Smoothing
DFT	Discrete Fourier Transformation
DT	Decision Template
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transformation
HME	Hierarchical Mixture of Experts
HMM	Hidden Markov Model
KNN	K-Nearest Neighbor
LPC	Linear Predictive Coding
MCSs	Multiple Classifier Systems
ML	Machine Learning
MLP	Multiple Layer Perceptron
MV	Majority Voting
NN	Neural Network
NBF	Naive Bayesian Fusion
PAA	Piecewise Aggregate Approximation
PLA	Piecewise Linear Approximation
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SOM	Self-Organizing Mapping

Chapter 1

Introduction

1.1 Preface

To solve the challenging pattern classification problems, machine learning researchers have extensively studied Multiple Classifier Systems (MCSs) [17, 42]. The interest in classifier combination is evident in the literature [27, 45, 64, 68]. Kittler et al. [49] have identified two reasons for using combination of classifiers: efficiency and accuracy. It is observed that different classifiers do not necessarily misclassify the patterns simultaneously. It is this information that is exploited to improve classification performance. MCSs have also been preferred for practical reasons. For example, the combination of *simple* classifiers (e.g. linear classifier) can achieve a similar performance as that of a more *complicated* classifier (e.g. Neural Network (NN)) and significantly reduce the computation cost. Several researchers have demonstrated the motivation to combine classifiers from the perspectives of expected errors, bias and variance [33, 92]. In addition, Dietterich [22] has investigated the possible advantages of using a combination of classifiers from the statistical, computational and representational aspects of the learning algorithms. Finally, MCSs have less stringent demands on the initial condition and the parameter tuning for the classifiers, which simplifies the model selection process. Although the results of classifier combination does not always outperform the best individual classifier in the ensemble, empirical studies have demonstrated its superiority

for various applications, including handwriting recognition [104], speaker recognition [65], face recognition [15], signature verification [7] and finger-print verification [72, 89].

1.2 Motivation

Most of the current research on MCSs are conducted by studying the behavior patterns of the base classifiers in the ensemble [26, 94, 101]. For example, Majority Voting (MV) assumes that a decision made by the majority of the classifiers in the ensemble is more likely to be correct. Therefore, aggregating the classifiers with MV helps to achieve a better performance [102]. Unlike MV, more advanced techniques [99] adaptively learn the relative importance of each base classifier and label the data with the most appropriate classifier or combinations of them. However, MCSs have seldom been discussed from the learning point of view. Theoretical and empirical studies demonstrate that the success of a supervised learning method heavily depends on how to appropriately tweak the trade-off between fitting the training data adequately and reducing the risk of overfitting, or, more mathematically, the balance between the bias and variance.

For instance, it is well known that the linear classifier and nearest neighbor classifier are not very effective although the reasons for their ineffectiveness vary. For the linear classifier, since the decision boundary must be a linear function and is not very adaptive, the data cannot be fitted adequately; that is, the learning of the data is not sufficient. In contrast, the nearest neighbor classifier does not appear to rely on any stringent assumption about the underlying data, and can adapt to any situation to fit the data well. However, the overfitting problem is easily incurred. Any particular subregion of the decision boundary depends on a handful of input points and their particular positions and is thus wiggly and unstable [42]. Similarly, since MCS is a type of the supervised learning methods, it also needs to balance the goodness of fit on the training data and the risk of overfitting to achieve a robust and good performance, even if the design of MSCs is very different from that of individual classifiers in terms of architecture, topology, and the level of decision making.

A number of methods, including bagging [17], adaboost [43], rotation forest [83] and random forest [10], have been developed. These algorithms are based on different learning methods and have been successfully applied to various tasks. However, traditional MCSs are not designed to adapt the data very well. Typically, the data can be represented in multiple ways, each of which extracts a certain type of information. Obviously, the classification system can benefit from aggregating the decisions based on different representations. However, methods such as adaboost and bagging are limited to only one of the representations, although *any* of them is available. Since it is difficult to know which representation is effective without prior knowledge, the information in the data might be distorted or not sufficiently utilized for the classification purpose.

Although other methods [19] can assume multiple representations, these techniques are still not completely data-adaptive. It is true that those approaches can employ all the possible representations and let the learning algorithm to assign the weights to the decisions based on the effective representations. However, these effective representations are only part of the possible representations and the rest act as *noise* representations, which can provide false information about the data. It is reasonable to assume that the bigger the portion of *noise* representations, the more likely the learning algorithm is distracted by the false information and assign inappropriate weights to the decisions based on the different representations. The inability of traditional approaches to discern the *noise* representations, somehow, limits the application of the traditional approaches to real problems. On one hand, it is desirable to employ different representations of the data to cover all the possibilities. On the other hand, it is likely that some *noise* representations will be introduced, degrading the classification performance. In short, a data-adaptive design should not only be able to select effective representations, but also exclude *noise* representations as much as possible. As it is known, the performances of the classifiers are usually data-dependent. Similar discussions can be applied to the selection of the base classifiers in the ensemble. A data-adaptive design should be able to select the effective classifiers and exclude the *noise* classifiers.

As a specific application, the classification of time series data is chosen for the research reported in this thesis. The sources of time series data includes finance,

medicine, biometrics, chemistry and speech. There is extensive work on the classification of time series data in the machine learning and data mining communities. Keogh et al. [55] have conducted empirical studies on the time series data mining which shows that nearest neighbor classifier gives an impressive performance for time series data. Povinelli et al. [71] have proposed Gaussian mixture models for the classification of time series data. Other classification methods for time series data are found in the literature [19, 31, 32, 90]. Such type of data is known for its high dimensionality and complicated underlying model. The success of the classification task is determined by how well the temporal information is extracted and utilized for the classification purpose. Chen et al. [14] have experimentally shown that different representations of the time series data can provide complementary information, which helps to enhance the classification performance. MCSs have a unique advantage for the classification of time series data. A successful design enables the classification system to adaptively determine the effective representation and base classifiers for the classification purpose, thus achieving a reliable performance. Although the classification of time series data with MCSs is not new, an adaptive and robust design of MCSs for this task is absent due to the previous problems.

1.3 Objective

The objectives and the investigation described in this thesis are to propose a robust and data-adaptive design of MCS as well as its application to the classification of time series data. The focus is not on the superiority of a certain representation, or similarity measure, or individual classifier because that they have already been available for the design of the new MCS. By appropriately aggregating their decisions, the newly devised MCS performs well for classifying data.

Unlike the design of other MCSs, the proposed Generalized Exhaustive Search and Aggregation (GESA) approach generates a collection of different ensembles of classifiers which are the power set of all the available classifiers, excluding the null set. A score function, employed to select the appropriate ensemble for classification at the local level, is composed of two parts. One measures the goodness of fit of

the ensemble. The other measures the risk of overfitting. It is anticipated that GESA can achieve a good and robust performance by dynamically balancing the trade-off between fitting the data adequately and reducing the risk of overfitting. In addition, it is shown that GESA is asymptotically optimal, given the ensemble of available classifiers.

The discussion of GESA is theoretically interesting, but it is not very practical due to its high computation cost to generate and search for the appropriate ensemble. Also, GESA is not completely data-adaptive, since the noise representations and classifiers still exist in some of the generated ensembles. The Generalized Adaptive Ensemble Generation and Aggregation (GAEGA) approach is proposed to overcome these difficulties. In GAEGA, only the effective representations and classifiers are selected to build the ensemble. In addition, the computation cost to generate and search for the appropriate ensemble is in the polynomial order of all the available classifiers.

Multiple Input Representation-Adaptive Ensemble Generation and Aggregation (MIR-AEGA) is derived from GAEGA, applying for the classification of time series data. MIR-AEGA adopts some special representation techniques which are effective for time series data. In addition, based on the conclusion from previous studies [55], the emphasis should be on the base classifiers such as nearest neighbor classifier and Dynamic Time Warping (DTW).

Bootstrapped-Adaptive Ensemble Generation and Aggregation (BAEGA) is another extension of GESA. Unlike GAEGA, BAEGA adopts the bootstrapped sample to train the base classifiers, and the entire training data set to train the aggregation rule. The purpose of BAEGA is to first avoid a fixed split of the training data to train the base classifier and aggregation rule separately, which needs to be pre-determined. Therefore, the performance of BAEGA is more general. Secondly, BAEGA aims to improve the diversity of the generated ensembles and utilize the information in the data more effectively.

1.4 Contributions

The principal contribution of this thesis is the development of the robust and data-adaptive MCS. Another main contribution is the application of the proposed method to the classification of time series data. The contributions have been realized through the following steps.

- Propose a measure to evaluate the goodness of fit of MCSs.
- Propose an Adaptive Aggregation approach for the decision combination.
- Propose GESA as a robust design of MCSs.
- Propose GAEGA to reduce the computation cost and achieve a data-adaptive performance.
- Develop MIR-AEGA to classify time series data.
- Propose BAEGA to utilize the information in the data more effectively and achieve a more general performance.

1.5 Organization of the Thesis

The organization of the thesis is as follows: the literature on MCSs and the classification of time series data are reviewed in Chapter 2. Chapter 3 discusses the GESA approach. In Chapter 4, the GAEGA approach, which is more practical than GESA, is proposed. In addition, the application of the proposed approach to classify time series data is discussed. In Chapter 5, the BAEGA approach is introduced as another extension of GESA. After the experimental results are discussed in Chapter 6, some conclusions and suggestions are presented for future research.

Chapter 2

Background

In the machine learning research, the decision aggregation has proved to be effective for improving the performance of various multiagent systems [1, 30, 37, 46]. Among them, MCS may be one of the most well-studied. In the literature, terms such as *combination of multiple classifier, classifier fusion, and divide and conquer classifiers* have been used interchangeably in MCSs [47]. The validity of MCSs is based on the observation that the sets of patterns misclassified by the different classifiers do not necessarily overlap. This indicates that different classifiers, when combined, provide complementary information about the patterns to be classified. In addition, the use of MCSs increases the reliability, since each base classifier provides redundant information for the whole system. Consequently, the failure of one base classifier does not necessarily cause the failure of the entire.

In this chapter, the current progress in MCSs is first reviewed. Then, the related techniques for the classification of time series data is discussed. In particular, the focus is on the available MCSs techniques for this task.

2.1 Multiple Classifier Systems

Many MCSs are reported in the literature. The approaches are distinguished by their structure, topology, aggregation strategy, aggregation procedure, and ensem-

ble generation. From these perspectives, the current research on MCSs is reviewed in the following sections.

2.1.1 Diversity

Diversity is regarded as the ability of the base learners in MCSs to exhibit different local behaviors to a given problem. Kuncheva claimed that the concept of diversity is one of the cornerstones in the study of MCSs.

"There is consensus among the researchers in classifier combination that the major factor for a better accuracy is the diversity in the classifier team ..." [52].

Previous researches have reported that the performances of most classifiers are *ad hoc*; that is, they outperform others only for a specific problem or for a specific subset of the input data. It is difficult to find a single expert for achieving the best results in the overall problem domain [97]. However, several independent classifiers seldom fail on a certain input coincidentally. Therefore, it is possible to enhance the accuracy and reliability by combining a set of classifiers, each of which performs differently for a given problem. Clearly there is no advantage to combine identical classifiers, and the performances of MCSs heavily depend on the diversity of the base classifiers.

Types of Diversity

Sharkey and Sharkey [92] have identified four types of diversity for the combination of NN classifiers in terms of the extent to which members of the ensemble exhibit the coincidental failure and function coverage. A function is covered if at least one classifier produces the correct output for each input that is tested.

- *Type 1 Diversity*: There are no coincidental failures and the function is covered. Type 1 diversity achieves an ideal performance for the ensemble on a test set. For any input, there is, at most, one classifier that produces an incorrect output.

- *Type 2 Diversity*: There are some coincidental failures, but the majority are correct, and the function is covered. Type 2 diversity does not meet all the criteria of Type 1. However, for any input, the majority of the classifiers produce the correct answer. In this situation, an aggregation strategy such as MV can still work effectively.

- *Type 3 Diversity*: The majority of the classifiers do not consistently give the correct answer; however, the classifiers in the ensemble still cover the function such that the correct output for each input pattern is consistently produced by at least one classifier. Obviously, MV does not work and more complex aggregation methods are expected.

- *Type 4 Diversity*: The test set is not entirely covered by the ensemble of classifiers. In this situation, the performance of MCSs can be unreliable. However, they can still be used to improve generalization, provided the aggregation methods are appropriately designed.

Sharkey and Sharkey [92] claim that such an assessment of the diversity indicates the best way of combining classifiers, and provide an upper bound on the level of generalization performance that can be expected from an ensemble.

Evaluation of Diversity

In the section, several popular methods are reviewed to measure the diversity of the base classifiers.

Within-Set Generalization Diversity (GD) Partridge and Yates [75] have proposed the GD method to measure the diversity of classifiers. This measure is computed as follows:

$$GD = 1 - \frac{p(A)}{p(B)}, \tag{2.1}$$

where $p(A)$ indicates the probability that any two classifiers which are randomly selected from set C will both fail on a randomly selected input, and

$p(B)$ indicates the probability that one randomly selected classifier will fail on a randomly selected input. GD takes values in the range [0,1]. Clearly, the larger GD is , the more diversity the selected classifiers exhibit.

Q-Statistics Kuncheva et al. [57] have proposed the Q-statistics to assess the pairwise similarity of the classifiers. For classifiers c_a and c_b. Q-Statistics is defined as follows:

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

where N^{11} is the number of times both classifiers are correct, N^{00} is the number of times both classifiers are incorrect, N^{10} is the number of times that classifier c_a is correct and c_b is incorrect, and N^{01} is the number of times that classifier c_a is incorrect and c_b is correct. Q_{ab} varies between [-1,1].

Compound Diversity (CD) Giacinto and Roli [36] have developed the Compound Diversity (CD) based on the compound error probability for two classifiers c_a and c_b. The equation is as follows:

$$CD = 1 - p(A), \tag{2.3}$$

where $p(A)$ is the probability that c_a and c_b will both fail on a randomly selected input. CD takes values in the range [0,1]. Clearly, the larger the CD is, the greater diversity of the selected classifiers.

Mutual Information (MI) Kang and Lee [53] have suggested that the pairwise mutual information can be used as a measure of the diversity between classifiers c_a and c_b such that

$$MI(a,b) = \sum_{i=1}^{L}\sum_{j=1}^{n} p(\omega_i,\omega_j) log \frac{p(\omega_i,\omega_j)}{p_a(\omega_i)p_b(\omega_j)}, \tag{2.4}$$

where L is the total number of classes and $\omega_i, i = 1...L$ are the class labels. It is evident that the larger $MI(a,b)$ is, the more the diversity for classifiers c_a and c_b have.

Correlation between Errors (CE) The correlation of the errors for the member classifiers is also a good choice for a measure of the diversity. Aksela [4] has introduced the CE method, represented by ρ_{ij}, to measure the diversity between classifiers c_i and c_j. It is calculated by

$$\rho_{ij} = \frac{Cov[v_e^i, v_e^j]}{\sqrt{Var[v_e^i]Var[v_e^j]}}, \tag{2.5}$$

where v_e^i and v_e^j are the vector of the error occurrence in classifiers c_i and c_j respectively. Cov and Var refer to the covariance and the variance, respectively.

Other diversity measures include the Ratio between Different and Same Errors (RDSE), Weighted Count of Errors and Correct Results (WCEC), Exponential Error Count (EEC), Measure of Closeness (MC) and Conditional Entropy (CE) [4, 48, 106], each conveying strength in certain applications.

Diversity Approaches

Once the diversity has been assessed, approaches to produce diversity among the base classifiers are considered. Windeatt and Ardeshir [98] have identified four categories of approaches. The first category includes the methods that reduce the dimension of the training set to attain different feature sets. Oza and Tumer [70] have proposed the input decimation approach which decouples the base classifiers by training them with different subsets of the input features. Thus, the correlation among the classifiers is significantly reduced. A possible weakness of the approaches in this category is that the decomposition of the feature space might conceal some high-dimensional patterns, degrading the performance of the entire system.

The second category includes methods that incorporate an ensemble of heterogeneous or homogeneous classifiers with different parameters. In these approaches, it is assumed that each base classifier tends to identify different discriminating patterns for the same input. Therefore, the performance of the entire system is enhanced through aggregation. However, the work of Valentini and Masulli [97]

has demonstrated a tradeoff between the accuracy and the independence of the base classifiers; that is, it is difficult to guarantee the accuracy and independence of the base classifiers at the same time. Therefore, the capability of MCSs to achieve diversity at the training level seems to be limited.

The third category includes techniques that resample the training set, and thereby, specifically apply each classifier on a different subset of data. The popular resampling techniques include bagging and boosting [17, 69]. The common weakness of the approaches in this category is that the decomposition of the input space reduces the available information for the individual classifiers. As a result, the classifiers might not be well trained.

Finally, the fourth category includes the output coding methods that create complementary two-class problems from MCSs. The most popular output coding approach is the Error-Correcting Output Coding (ECOC) [67, 74]. The appropriate design of the coding matrix is one of the most important issues for the output coding approach.

2.1.2 Architectures

From the architecture perspective, MCSs are divided into the modular and ensemble systems, depending on whether or not the tasks are decomposed [88].

Modular Systems

In modular systems, the problem is decomposed into a series of sub-problems. Each base classifier focuses on one of the sub-problems, and the solution to the entire task can not be provided by any individual member. The complete solution requires the coordination of all of the modules. Typically, the combination of the modular system relies on some switching mechanism, in which the output for each input data is taken from the most relevant component, or even the most relevant blend of modules.

The Hierarchical Mixture of Experts (HME) [44] is a popular modular system. Its architecture is a tree structure in which the classifier selection is conducted

according to a pre-defined distribution, as described in Fig. 2.1. Each of the base classifiers is an expert in the local area of the feature space. The outputs of the experts go up and are mixed by the gating network.

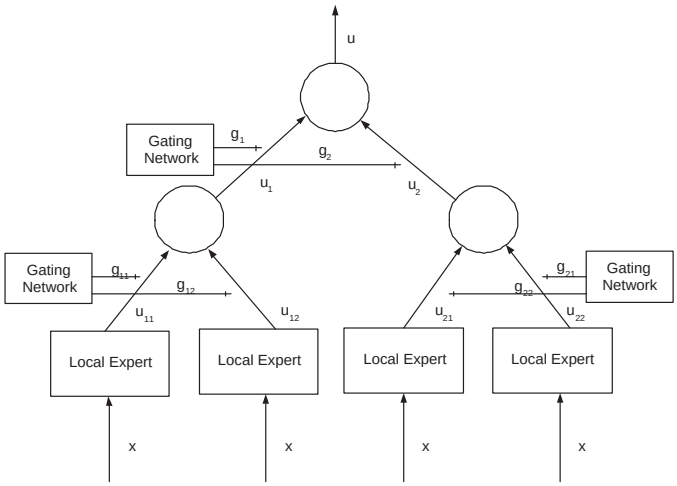

Figure 2.1: Two-level hierarchical mixture of experts architecture

Ensemble Systems

In an ensemble system, several redundant approximations to the same function are combined to yield a single unified outcome. In contrast to the modular system, each component of the ensemble system can provide a solution to the entire task, although it may not as good as the solution achieved by the ensemble. The combination of the ensemble system takes all the outputs of the base classifier in some forms such as the voting and averaging.

2.1.3 Ensemble Generation

Ensemble generation is crucial for the design of MCSs. There are various ways to generate the classifier ensemble. Wanas and Kamel [100] have summarized the methods for generating an ensemble of NN classifiers as follows.

- *Varying the Initial Condition*: A set of networks is created by varying the initial conditions such as the initial weights.

- *Varying the Network Topology*: Various network topologies and architectures can be used. These variations are in the number of hidden layers or nodes.

- *Varying the Training Algorithm*: Various algorithms, including the Back Propagation (BP) and the Radial Basis Function (RBF), are chosen to train each network.

- *Varying the Training Data*: The training data, presented to each member of the ensemble, can differ. Several alternative methods can be carried out. These alterations can be done by resampling, decomposing the feature space, varying the data sources or pre-processing methods. One example is bagging. Bagging is a "bootstrap" ensemble method that creates the ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples, where N is the size of the original training set. Then, the decisions of the base classifiers are combined by MV.

In order to overcome the problem of the over-generation of the ensemble, approaches such as *test and select* and *overproduce and choose* [38, 93] have been proposed in previous research. The selection criteria and search strategy are pivotal in deciding the optimal subset of classifiers. The selection criteria include the maximization of diversity, maximization of accuracy, and minimization of mean squared error [38, 80, 93]. The search strategies include the exhaustive search, forward search, backward search, tabu search, genetic algorithm, population-based incremental learning, and clustering and search [38, 79, 93].

2.1.4 Topology and Structure

Lam [61] has categorized MCSs in terms of their topologies or structures. Four different topologies are recognized in Lam's work.

Conditional Typology

In conditional topology, a series of classifiers are sequentially applied to the data set, until satisfying results are achieved. Usually, the system begins with the primary classifier. Therefore, this structure has the advantage of computational efficiency when the primary classifier is a fast one, as it processes most of the easily recognizable patterns.

Hierarchical (Serial) Topology

In the hierarchical topology, the classifiers are applied in succession such that each classifier produces a reduced set of possible classes for each pattern. The distinction between this topology and the previous one is that the former stops applying the classifier to the data set, once the satisfying results are obtained, whereas the latter must apply all the designed classifiers to the data set.

Hybrid Topology

Certain classifiers will be more effective on particular patterns. This information is used to select the classifiers which are appropriate to identify the given patterns. In the hybrid topology, the selection strategies are driven by various criteria, including reliability, classifier agreement (obtained from the results on a training set), and the features or parameters of the patterns.

Multiple (Parallel) Topology

For multiple topology, multiple classifiers first operate in parallel to produce the classification results of a given pattern, and the decisions are combined to yield a final decision. The additional computational cost for the individual classifier in this architecture can be reduced by using a parallel hardware architecture. In addition, the operation of each classifier allows for the development and introduction of new classifiers without requiring major modifications to the fusion process.

2.1.5 Aggregation Strategy

Although the previous research on MCSs has focused on combining different classifiers, the current interest has shifted to finding the appropriate aggregation strategies. Woods et al. [101] have categorized combination approaches into two groups: dynamic classifier selection and classifier fusion. This scheme has been further developed by Kuncheva et al. [47], in which the difference between static selection and dynamic selection are distinguished. This categorization is further extended into the selection-fusion scheme [51]. In the following section, several well known aggregation strategies are discussed.

Selection

The classifiers in the selection approach are regarded as complementary and each classifier is an expert in some local regions of the feature space. Kuncheva [51] further divided the selection scheme into the static and dynamic structure. The subdivision depends on whether the selection is made upon statically or dynamically.

- *Static Selection*

  In a static selection system, feature regions are specified during a training phase. Two possible training approaches are specified by Kuncheva [51]: (1) specify the regions and then assign a responsible classifier for each region (2) given the data set, find the region where each classifier is the best. The second approach might be more effective, but it is difficult to implement.

- *Dynamic Selection*

  Dynamic selection systems predict which single classifier is most likely to be correct for a given sample. Only the output of the selected classifier is considered in the final decision [101]. In dynamic selection systems, the selection of the classifier to label the data is made during the operation phase. Typically, this choice is based on the certainty of the current decision. The advantage of the dynamic selection system is that the error-dependency can be eliminated.

Woods et al. [101] have proposed the Dynamic Classifier Selection by Local Accuracy (DCS-LA). In this research, "local regions" are defined as the K-nearest neighbors in the training data. The idea is to estimate each classifier's accuracy in the local regions of the feature space, surrounding an unknown test sample; then, the decision is made on the most locally accurate classifier. Two methods for estimating the local accuracy are also proposed in this research. One is simply the percentage of training samples in the region that are correctly classified which is referred as the "overall local accuracy". Another possibility is to estimate the local accuracy with respect to some output class, which is referred to as the "local class accuracy". Other dynamic selection approaches including the prior selection method, posterior selection method, and Dynamic Classifier Selection based on Multiple Classifier Behavior (DCS-MCB), are reported in the literature [36, 37, 38].

Fusion

Instead of a search for the most appropriate classifier for the local regions, the fusion approaches assume that all the classifiers are equally experienced, and the decisions of all of the classifiers are taken into account. Sharkey [88] further divided the fusion scheme into fixed and trained methods. The sub-division depends on whether the weight for the combination of the multiple classifiers is fixed or dynamically trained.

- *Fixed Fusion*

  In the fixed fusion systems, the weight for the combination of the multiple classifiers is fixed. There is no learning process for the system to learn the weight for each classifier.

  The most simple combination strategy is to apply simple operators such as sum, simple average, product, and min-max to the outputs of all the base classifiers. Then, the decision is based on the max or min value of the final results. Another important fixed fusion approach is MV, in which the final decision is made by selecting the label that is most represented by the individual classifiers. The advantage of the fixed fusion approach is its simplicity

and lower computational cost; the disadvantage is the lack of flexibility in the combination process.

- *Trained Fusion*

  In the trained fusion system, the weight for the combination of the multiple classifiers is learned through training. Usually, there is a learning process for the system to learn the weights of each classifier.

  Unlike simple averaging and MV, the Weighted Aggregation approaches such as Weighted Averaging (WA) and Weighted Majority Voting (WMV) adaptively assign the weights to the base classifiers. Clearly, the performances of WA depend on the way that the weights are learned. Wanas and Kamel [58] have developed a novel scheme for decision aggregation. A detector is employed to estimate the weights for the base classifiers. The weight factor represents the confidence in the output of each classifier. These confidences are then aggregated by using fixed classifier-combining methods.

  Stacked Generalization (SG) [95] requires a higher level classifier to learn the error of the base classifiers as described in Fig. 2.2. The outputs of the base classifiers are used as the training data for the stacked classifier at the higher level. The purpose of this approach is to minimize the generalization of the errors of the base classifiers. Obviously, the performance of this aggregation approach heavily depends on the decision distribution of the base classifiers and the choice of the stacked classifier at the higher level.

  In the Naive Bayesian Fusion (NBF) approach [104], the independence of the base classifiers is assumed. For each classifier, a confusion matrix is learned from a validation data set, representing the decision distribution of each classifier. For the given test data, a soft label vector, which represents the support for each category, is generated by each classifier. The final label is obtained by applying the max-sum or max-product operator to these soft label vectors. This approach is built on the bayesian theory. However, an assumption about the independence of the base classifiers is too strong and can not be satisfied in many applications.

  Behavior Knowledge Space (BKS) combines the decision of the multiple clas-

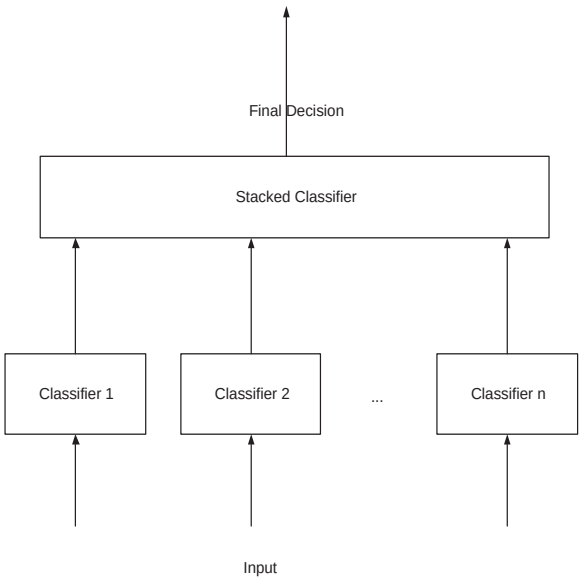

Figure 2.2: Stack Generalization (SG)

sifiers by building a lookup table for each candidate. The lookup table comprises all possible combinations of the class labels, and each cell of the table is one possible combination. The final decision is reached by applying a maximum operator on the set of class label in each cell. Unlike NBF, BKS does not assume the independence of the base classifiers. However, large set of training data is required [41].

Decision Template (DT) [47] is similar to BKS. For any input, DT builds a profile table from the ensemble of the classifiers. For each class, there is a desirable decision template. Then, the profile table is compared with each decision template. Finally, the decision is made by applying a maximum operator on the set of similarity measures. DT does not assume the independence of the base classifiers either. Compared to BKS, DT does not require such a large amount of training data. The empirical research has proven that DT generally has a better performance compared to that of other methods. However, DT estimates the joint decision distribution of the base classifiers based on the average performance. It can be ineffective in some situations

such as the decision XOR problem, discussed in Chapter 6.

Other aggregation approaches include the Dempster-Shafer (DS) and Fuzzy Integral (FI) [6, 13].

2.2 Methods for the Classification of Time Series Data

The progress in MCSs continues to bring new ideas for traditional problems such as the classification of time series data, which is challenging but motivating. Although the classification of time series data is not a discipline in its own right, it has been extensively explored in many fields such as statistics, signal processing, control theory, and machine learning. Several traditional techniques have been employed for classifying time series data including Hidden Markov Model (HMM), Recurrent Neural Network (RNN), and Dynamic Time Warping (DTW).

Hidden Markov Model (HMM) It has been mostly used for problems such as speech recognition, gesture recognition, and handwriting recognition. An HMM is a statistical model in which the system being modelled is assumed to be a Markov process with unknown parameters. In a standard Markov model, the state is directly visible, and the state transition probabilities are the only parameters. In HMM, the state is not directly visible. However, the variables that are influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by a HMM gives some information about the sequence of states. The major task for the use of HMM is to determine the unknown parameters [17]. Fig. 2.3 dipicts an HMM. w_i is the hidden units, a_{ij} is the transition probability, and b_{ij} is the probability of the emission of a visible state.

Recurrent Neural Network (RNN) Unlike the feedforward NN, an RNN forms directed cycles among the neurons. In principle, an RNN can exhibit almost

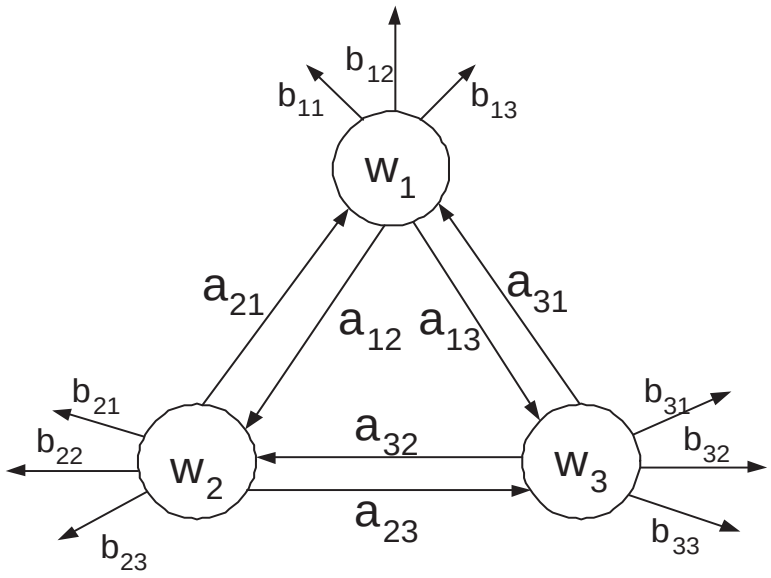

Figure 2.3: Example of Hidden Markov Model (HMM)

arbitrary sequential behavior, which makes it promising for adaptive robotics, speech recognition, music composition, attentive vision, and many other applications. Fig 2.4 illustrates an example of an RNN. The output unit values are fed back as auxiliary inputs. During the classification, pattern $\mathbf{x}$ is presented to the input units. The feedforward flow is computed and the outputs are fed back as auxiliary inputs. This leads to a different set of hidden unit activations, new output activations, and so on. Ultimately, the activations stabilize, and the final output values are used for classification. RNNs have been proven effective in learning short time-dependent signals. However, RNNs are less successful for long time-dependent signals because of the *diluted* effects [17]. The application of RNNs on the classification of time series data is found in literature [34, 40].

Dynamic Time Warping (DTW) It is a traditional algorithm for measuring the similarity among the sequences of time series data. Its many applications include speech recognition and word recognition. Unlike the simple Eucledian distance measure, the match of the sequences of time series data is not always

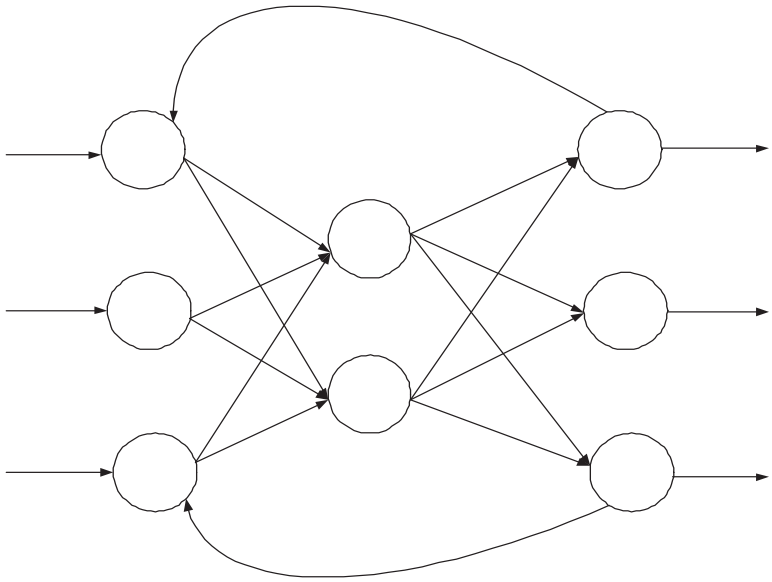

Figure 2.4: Example of Recurrent Neural Network (RNN)

linear in DTW. It allows the sequences to be *warped* non-linearly in the time dimension to find an optimal match for the given sequences. Fig.2.5 illustrates an example of DTW as the similarity measure for the sequences of time series data.

Keogh et al. [50, 55] have conducted empirical studies on time series data mining which shows that the Euclidean distance and DTW outperform many other similarity measures. With the nearest neighbor classifier, the classification performances are impressive for time series data. Kadous [54] has summarized a few weaknesses when DTW is applied as the similarity measure for the classification of time series data. One of the fundamental issues for DTW is that this similarity measure might not be very reliable. In the following chapter, an example is introduced to show how DTW fails to solve an easy problem for time series classification.

One of the common weaknesses of traditional approaches is that they can only be applied to certain types of time series data. However, the sources and underlying models of time series data are very diversified. Therefore, the performance of those

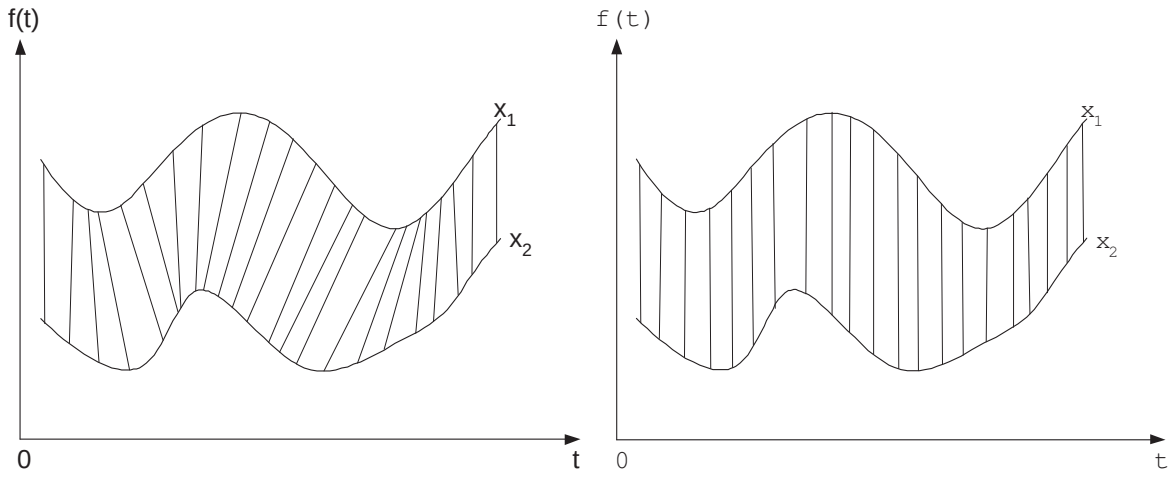

Figure 2.5: Example of Dynamic Time Warping (DTW) and euclidian distance

traditional methods are data-dependent, and are difficult to apply. MCSs have an unique advantage in classifying time series data, since MCSs allow the data to be modelled in different ways. By adaptively selecting the most appropriate method or combinations of them, a good and reliable performance can be achieved. With the recent advances in this area, many MCSs have been proposed for the classification of time series data.

Sanchos et al. [90] have employed an ensemble of five base classifiers, each of which is either Self-Organizing Mapping (SOM) or Multiple Layer Perceptron (MLP) for the classification of time series data. Each SOM or MLP shares the same architecture, but has different initial weights. For MLP, the outputs of the net are considered as an estimation of the posteriori probability of the categories. For SOM, the Euclidian distance between the input vector and the weight vector of each node is first computed. For each category L_i, the corresponding winner node has the minimum Euclidean distance for all the nodes which belong to L_i. Then, the posterior probability of each category is estimated by applying a softmax function on the Euclidean distance between the input vector and the weight vector of the winner node of each category. Finally, the category which has the maximum average posterior probability is assigned to the input data.

Although it is quite innovative to combine supervised and unsupervised learning in Sanchos' research, there are several weaknesses. At first, the Euclidean distance is used to calculate the posteriori probability of each class in the SOM classifiers. However, it is known that the Euclidean distance is not very reliable for measuring the similarities of time-dependent data. Therefore, it is questionable to adopt this method to generate the posteriori probability. In addition, the mapping between the cluster and the class label might not be a one-to-one or many-to-one mapping which is assumed implicitly by the authors.

Ghosh et al. [29, 32] have employed an ensemble of NNs with the WA fusion technique to identify and classify underwater acoustic signals. The feature space of the input data is composed of a 25-dimensional feature vector extracted from the raw signals. In the feature vectors, there are 16 wavelet coefficients, 1 value denoting the signal duration, and 8 other temporal descriptors and spectral measurements. One of the weaknesses for this approach is that the feature extraction is *ad hoc*. The number and type of the features might be optimal for only certain applications such as oceanic signal data.

González et al. [31] and Diez et al. [16] have proposed a classification system for time series data which is based on the boosting technique. The base classifiers are very simple classifiers which consists of only one literal. The background temporal predicates can be interval-based, point-based, or distance-based. Each time series data is represented by a set of literals, and the final classification results are obtained according to the weighted average of the outcomes of the base classifiers. The objective of this approach, as claimed by the authors, is to find a non-domain specific temporal classification technique. However, the selection and definition of the temporal predicates are themselves *ad hoc*. In addition, the boosting technique is sensitive to the noisy data, limiting its performance in real applications.

Dietrich et al. [19, 21] have developed three architectures for the fusion of decisions based on the local features of time series data: Classification, Decision Fusion and Temporal Fusion (CDT), Decision Fusion, Classification and Temporal Fusion (DCT) and Classification, and Temporal Fusion, Decision Fusion (CTD). A set of the local features for time series is calculated within a local sliding window $W^t(t = 1..T)$ which covers a small part of the time series and moved over it. From

each window, W^t, a set of p features are extracted: F^t= (F_1^t,F_2^t,... F_p^t). CDT and CTD adopt the hierarchical fusion architecture. They differ from each other by firstly fusing the decisions of the base classifiers on different types of features in the same window or the same type of features over various windows. DCT first connects the p features in each window to a vector. The final decision is made, based on the fusion of the decisions of the base classifiers on the p-feature vector in each window. Dietrich et al. [20] further proposed three other architectures for the fusion of decisions: Multiple Decision Template (MDT), Decision Template (DT) and Clustered Multiple Decision Template (CMDT). MDT and DT are similar to CDT and DCT, respectively. CMDT improves MDT by assigning a set of multiple templates $\{DT_i^1, DT_i^2, ..., DT_i^k\}$ to each class w_i. There are weaknesses to this approach. First, the determination of the parameters such as the size of the sliding window is data-dependent. Secondly, the sliding window method can cause the loss of information for high dimension temporal patterns.

Hsu et al. [40] have devised an hierarchical mixture model, called the specialist-moderator network for the classification of time series data. This method combines RNNs in a bottom-up way. The primary contribution of this work is the model's ability to identify the intermediate targets, based on a partition of the input channels. One drawback of the hierarchical classifier system, such as the hierarchical mixture of experts, is that the system is not robust and the failure of one base classifier immediately affects the performance of the entire system.

2.3 Summary

In this chapter, the MCSs in the literature are first presented from different perspectives, including diversity, architecture, ensemble generation, and aggregation. Also, various methods for the classification of time series data were reviewed. Compared with traditional methods such as HMMs, RNNs and DTWs, MCSs have specific advantages and can achieve a better and more reliable performance. In the following chapters, a new design of the MCS and its application to the classification of time series data are introduced. The motivation to develop such a design is examined, and the results of the empirical study on benchmark data sets are presented.

Chapter 3

Generalized Exhaustive Search and Aggregation (GESA) Approach

3.1 Motivation

3.1.1 General Principles of Supervised Learning

Although MCS is a special type of classification system in terms of the architecture, topology, and level of decision making, the design should still follow some general principles of the supervised learning methods. Therefore, it makes sense to review the supervised learning from the machine learning point of view.

Supervised learning is an attempt to learn model f by example through a teacher, which observes the system under study, and assembles a training set of observations (x_i, y_i), $i = 1, ..., N$. The observed input values x_i are also sent to an artificial system, known as the learning algorithm. It produces outputs $\hat{f}(x)$ in response to the inputs. The learning algorithm can modify the input and output relationship $\hat{f}$ in response to the differences between Y and $\hat{f}(x)$, which are defined by a loss function. This process is known as learning by example. After the learning process is finished, it is anticipated that the artificial and real outputs will be close

enough in value to be useful for all the sets of inputs likely to be encountered in practice [42].

3.1.2 Overview of Overfitting Problem

Numerous empirical and theoretical studies have confirmed that it is also critical to reduce the risk of overfitting in the learning process. Overfitting, a major concern in machine learning, is recognized as a violation of Occam's razor [17]. Usually a learning algorithm is trained by using some set of training samples, in which the desired output is known. The learner is assumed to reach a state where it can also predict the correct output for the other samples such that the learner can be generalized for situations not presented during training. However, the learner can adjust to very specific random features of the training data that have no causal relation to the target function. In this process of overfitting, the performance on the training samples still increases, whereas the performance on the test data worsens. Overfitting occurs, for example, because the model is too complicated, the training sample is too small or its dimension is too high, and the learning process is too long. It is somewhat suspicious that a learning method, without any mechanism to reduce the risk of overfitting, can achieve a robust performance. For example, linear regression is usually considered as a robust supervised learning method. However, empirical studies have shown that this conclusion is valid only when the dimensions of the data are relatively low. In a typical case, methods such as stepwise regression, lasso regression and ridge regression are preferred. Various regularization methods such as penalization, selection and shrinkage can be employed to reduce the risk of overfitting [42]. Although there is no proof that certain loss functions or regularization methods are superior to others, an explicit discussion from this perspective provides some insight into a robust design of the supervised learning algorithm.

3.1.3 Understanding MCSs from the Learning Perspective

Boosting is one of the well studied MCSs. Hastie et al. [42] have discussed this method from the learning point of view. They have concluded that adaboost is

equivalent to the forward stage-wise additive modelling, based on the exponential loss function. The relative good performance of adaboost is partly explained by the robustness of the exponential loss function, compared with that of other types. However, it does not mean that adaboost could completely avoid the overfitting problem. Hastie et al. [42] have also discussed several ways to reduce the risk of overfitting when training a more generalized version of boosted trees: use a small-size tree as the base learner to reduce the model complexity and use the shrinkage method to reduce the learning speed. For other MCSs, there are few explanations by the designers from the learning perspective. It is likely that these designed MCSs are not robust. Compared with the individual classifiers, MCSs succumb to the overfitting problem more easily, since they are more complex and have a higher degree of freedom.

In the proposed Generalized Exhaustive Search and Aggregation (GESA) approach, the "over-generation and selection" strategy is adopted. A collection of various ensembles of classifiers are first generated, each of which fits the validation data with different degrees[1]. The test data are then classified by each of the generated ensembles. The final decision is made by taking into consideration the ability of each ensemble to fit the validation data locally and reducing the risk of overfitting. By appropriately balancing the trade-off between these two parts, it is expected that the performance of this newly devised method is robust over general situations.

3.2 Details of GESA

To simplify the discussion, a set of heterogeneous classifiers, each of which takes the raw input representations, is considered. The architecture of GESA is exhibited in Fig. 3.1. The knowledge space Ω is defined as all the available classifiers. GESA first generates a collection of different ensembles. This motivation is based on the observation that each ensemble has a different capability to fit the data. Each ensemble makes a decision from the data. Then, the most appropriate decision is

[1]The validation data set denotes the data which are used to train the aggregation rule

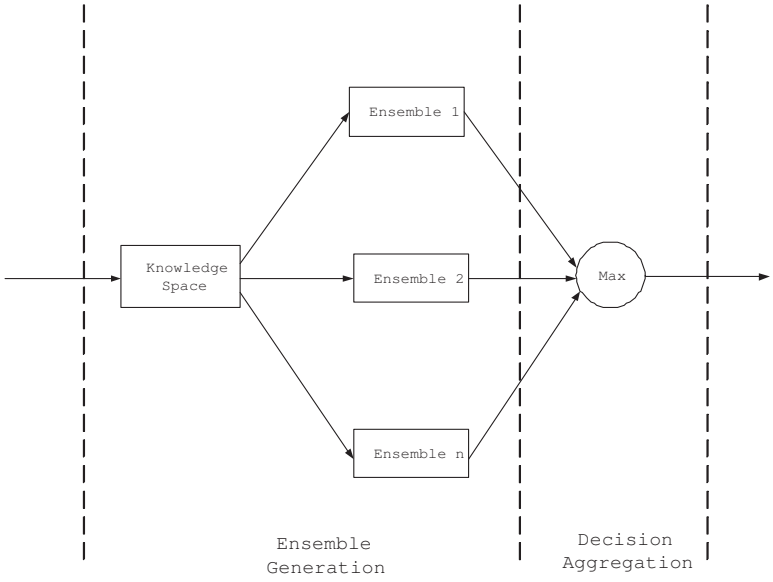

Figure 3.1: Architecture of GESA

selected to label the data by maximizing the proposed score function. The score function is composed of two parts. One measures the goodness of fit of the ensemble, and the other measures the risk of overfitting.

3.2.1 Loss Function

In this section, a loss function is defined to measure the goodness of fit on the validation data by the ensemble. In this thesis, the decision boundary of the base classifiers in the ensemble represents a partition on the data space X into a collection of disjoint local regions $\{X_i\}$. Each of the local regions is determined by the decision boundary of the base classifiers in the ensemble. Fig. 3.2 gives an example in which the data space is partitioned by 3 linear classifiers into 7 disjoint local regions. In Fig. 3.2, it is noted that the data in some local regions come from several categories and the decision boundary doesn't separate the data accurately. However, since each local region could be labeled with only one category in the end, some data in these local regions are unavoidably misclassified.

Let $L = \{l_1, l_2..., l_r\}$ be the categories of the data. Given an ensemble ϕ ($\phi = \{c_1, c_2, ..., c_h\}$), $\hat{\theta}_\phi(X_i)$ is adopted to measure the goodness of fit of ensemble ϕ on the validation data in the local region X_i. The definition of $\hat{\theta}_\phi(X_i)$ is as follows. Let p_i^j be the probability of the data with category j in X_i and $\hat{p}_i^j$ is the estimation of p_i^j. The most frequent category l_{mi} is the category with the highest probability in local region X_i, which is defined as follows.

$$l_{mi} = l_{ki} \quad k = \arg\max_j p_i^j.$$

Consequently, $\hat{l}_{mi}$ is used to denote the estimation of l_{mi} and defined as

$$\hat{l}_{mi} = l_{ki} \quad k = \arg\max_j \hat{p}_i^j.$$

Suppose X_i is labelled with l_{mi}, the loss function for $\mathbf{x} \in X_i$ is defined as

$$I(\omega(\mathbf{x}), l_{mi}) = \begin{cases} 1 & \omega(\mathbf{x}) = l_{mi}; \\ 0 & \text{Otherwise}, \end{cases} \tag{3.1}$$

where $\omega(\mathbf{x})$ is the actual label of $\mathbf{x}$. It is readily shown that

$$E(I(\omega(\mathbf{x}), l_{mi})) = p_i^m,$$

where p_i^m is the probability of the data with label l_{mi} in X_i. $\theta_\phi(X_i)$ is therefore defined by calculating.

$$\begin{aligned} \theta_\phi(X_i) &= 1 - E(I(\omega(\mathbf{x}), l_{mi})) \quad \mathbf{x} \in X_i \\ &= 1 - p_i^m. \end{aligned}$$

A straightforward estimator for p_i^m is $\hat{p}_i^m$ where $\hat{p}_i^m$ is the frequency of the validation data from $\hat{l}_{mi}$ in X_i. The estimation of $\theta_\phi(X_i)$ is computed by

$$\hat{\theta}_\phi(X_i) = 1 - \hat{p}_i^m.$$

The goodness of fit of ensemble ϕ on the entire validation data set is then measured by $\hat{\Theta}_\phi$. It is the weighted average of $\hat{\theta}_\phi(X_i)$ over all the local regions. Therefore,

$$\hat{\Theta}_\phi = \sum_i \hat{\theta}_\phi(X_i) \times \frac{n_i}{N}, \tag{3.2}$$

where n_i is the number of observations in the local region X_i and N is the total number of observations in the validation data set. The larger the value of $\hat{\Theta}_\phi$ indicates that the validation data is less fitted by ensemble ϕ.

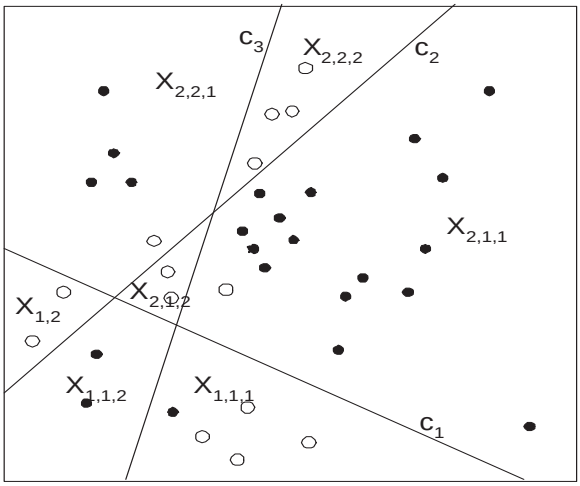

Figure 3.2: Partitioning of the data space by an ensemble of 3 classifiers

3.2.2 Ensemble Generation

To simplify the discussion, a set of heterogeneous classifiers, each of which takes the raw input representation, is firstly considered in GESA. The ensemble generation of GESA is straightforward. In the previous section, how to measure the goodness of fit of the ensemble on the validation data set is established. It is desirable that

the ensembles that are generated differ as much as possible. The advantage is that GESA can explore the different levels of goodness of fit on the validation data set by the ensembles. Given the knowledge space Ω, all of the possible subsets are 2^{Ω}. Evidently, null set $\emptyset$ is not useful. So, the collection of ensembles F, which is generated in GESA, is

$$F = 2^{\Omega} - \emptyset. \tag{3.3}$$

3.2.3 Decision Aggregation

The proposed GESA adopts the novel Adaptive Aggregation (AA) approach for the decision aggregation. After the reasons for using the AA approach are given, its major developing steps are discussed.

Goal for the Proposed AA Approach

In the ensemble generation layer, a collection of ensembles F is generated with different ability to fit the validation data. For the test data $\mathbf{x}$, each ensemble accordingly has a local region which contains $\mathbf{x}$. Given the collection of generated ensembles $F = \{\phi_i\}(i = 1...n)$, suppose $H_x = \{X_i\}(i = 1...n)$ is all the local regions which contain $\mathbf{x}$. Clearly, for each local region X_i in H_x, there is $\mathbf{x} \in X_i$. As discussed in Section 3.2.1, the goodness of fit of each ensemble for the local region is measured by $\hat{\theta}_{\phi}(X_i)$. The AA approach labels the test data $\mathbf{x}$ by balancing the trade-off between fitting the validation data in local region adequately and reducing the risk of overfitting. The key is to decide which ensemble is appropriate for the decision-making for test data $\mathbf{x}$. Let's further elaborate on this idea with the examples in Fig. 3.3. To simplify the discussion, without the loss of generality, a subset of the collection of generated ensembles $F' = \{\phi_1, \phi_2, \phi_3\}$ is considered, where $\phi_1 = \{c_1\}$, $\phi_2 = \{c_1, c_2\}$ and $\phi_3 = \{c_1, c_2, c_3\}$. If the principle to select the appropriate ensemble from F' is clarified, it is easy to extend the principle to F. If the test data $\mathbf{x}$ locates inside the region $X_{2,1,2}$, how should it be labelled? In this situation, $H_x = \{X_2, X_{2,1}, X_{2,1,2}\}$ where $X_{2,1} = X_{2,1,1} \bigcup X_{2,1,2}$ and $X_2 = X_{2,1} \bigcup X_{2,2}$.

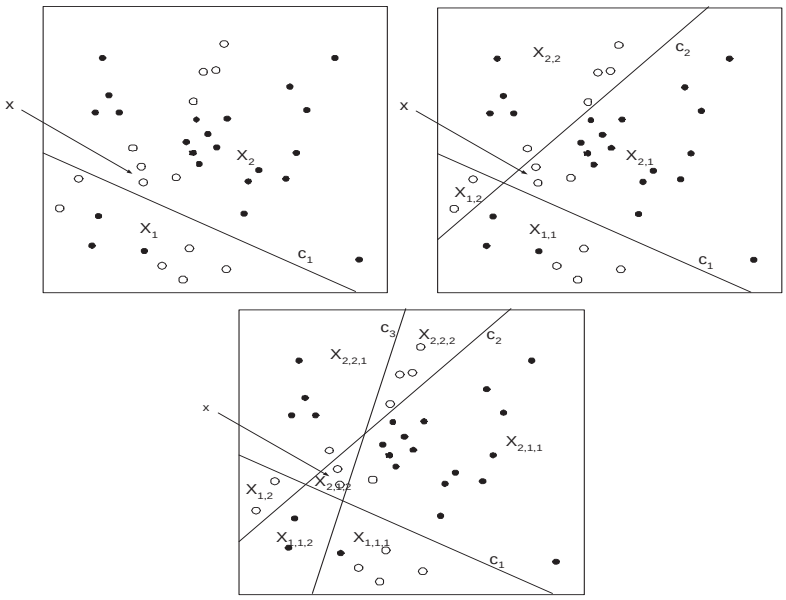

Figure 3.3: Partitioning of the data space by ϕ_1, ϕ_2, ϕ_3

A straightforward way is to first label $\mathbf{x}$ with the estimated most frequent category of each local region in H_x and select the most appropriate one from them as the final decision. For example, when using ϕ_3, the local region is $X_{2,1,2}$. When using ϕ_2, the local region is $X_{2,1}$. Suppose that it is more appropriate to make the decision with ϕ_2. Then $\mathbf{x}$ is finally labelled with the category, say l_1. Obviously, the final decision of $\mathbf{x}$ depends on how to determine the most appropriate ensemble. From Fig. 3.3, there is less percentage of data to be mis-classified in $X_{2,1,2}$ than $X_{2,1}$ in the in-sample test. It indicates that ϕ_3 fits the validation data better than ϕ_2 locally. Normally, it seems to be advantageous to select ϕ_3 to label $\mathbf{x}$ since it fits the data better at the local level. However, it is likely that ϕ_3 may over-fit the data in the local region $X_{2,1,2}$, which makes the inference on the label of $\mathbf{x}$ unreliable. In contrast, if the ensemble ϕ_2 is selected, the in-sample fitting may not be adequate and the final outcome can be poor. Therefore, there should be a mechanism to adaptively balance the trade-off between fitting the validation data adequately and reducing the risk of overfitting. The basic idea of the AA approach is examined as follows.

- If the value of p_i^m is known for each X_i in H_x, we claim that the following aggregation strategy provides an optimal performance: (1) among all the local regions $X_k \in H_x$, find the local region X_i which has the maximum value of p_i^m; (2) label $\mathbf{x}$ with the category l_{mi} which has the probability p_i^m in X_i. In (1), this aggregation strategy searches for an ensemble that has the best fit for the local region which contains $\mathbf{x}$. The decision-making in (2) is actually based on the real probability distribution of the data in the local region. Therefore, this decision should be optimal. However, in most of the situations, p_i^m is not known and can only be estimated through $\hat{p}_i^m$. According to the law of the large number, $\hat{p}_i^m$ converges to p_i^m when the sample size is large enough. In this situation, $\hat{l}_{mi}$ is exactly the category l_{mi}. Therefore, in the above aggregation strategy, if p_i^m is replaced by $\hat{p}_i^m$, the decision aggregation should still be optimal given the sample size is large enough.

- When the sample size in the local region X_i is small, $\hat{p}_i^m$ can significantly deviate from p_i^m. In this situation, the previous aggregation strategy is unreliable since $\hat{l}_{mi}$ is less likely the same with the category l_{mi}. To solve this problem, we consider the hypothesis test

$$H_0 : p_i^m = \hat{p}_i^m$$
$$H_1 : p_i^m \neq \hat{p}_i^m$$

  and intend to determine how confident it is to estimate p_i^m by $\hat{p}_i^m$. This estimator is denoted as $G(p_i^m, \hat{p}_i^m, n_i)$, where n_i is the number of sample observations in local region X_i. Since each hypothesis test is associated with type I and II errors, $G(p_i^m, \hat{p}_i^m, n_i)$ is evaluated by the probability that both type I and II errors are controlled under a pre-determined threshold. There is a reason for choosing this method. When type I and II errors of the hypothesis test are within the threshold, the null hypothesis is more likely to be valid. Therefore, it is less risky to estimate p_i^m by $\hat{p}_i^m$.

- Finally, for each local region X_i, the estimator is computed by

$$S_i = \hat{p}_i^m G(p_i^m, \hat{p}_i^m, n_i). \tag{3.4}$$

The ensemble for labelling test data $\mathbf{x}$ is selected by finding the one such that S_i is the highest value in the local region which contains $\mathbf{x}$. This score function can be understood from two different perspectives. From the estimation point of view, it is desirable to have a bigger value of $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$ at the same time. It can be shown that the range of $G(p_i^m, \hat{p}_i^m, n_i)$ is [0,1]. When $G(p_i^m, \hat{p}_i^m, n_i) \to 1$, it is less risky to estimate p_i^m with $\hat{p}_i^m$. Also, the ensemble with a larger value of $\hat{p}_i^m$ is more likely to be the most appropriate one. From the learning point of view, $\hat{p}_i^m$ determines the estimation of $\theta_\phi(X_i)$ and measures the goodness of fit on the validation data in X_i. It can be shown that $G(p_i^m, \hat{p}_i^m, n_i)$ is positively correlated to n_i. Consequently, the larger value of $G(p_i^m, \hat{p}_i^m, n_i)$ favors the local region with a larger number of observations. Therefore, the risk of overfitting can be reduced.

Adaptive Aggregation (AA) Approach

In this section, we provide a detailed description of the AA approach. In the training phase, a lookup table is built for each ensemble ϕ_i in collection F. The decisions of the base classifiers in cell (entry) e_i determine a local region X_i. Each e_i records a score and the label of the estimated most frequent category $\hat{l}_{mi}$ for the local region X_i. Table 3.1 is an example of the lookup table in the AA approach, where $\hat{c}_i$ is the estimation from classifier c_i. In the test phase, for input vector $\mathbf{x}$, the corresponding cell e in each lookup table is checked. The cell with the maximum score is the winner, and $\mathbf{x}$ is labelled with the label of the winner cell. Given $F = \{\phi_1, \phi_2, \phi_3\}(\phi_1 = \{c_1, c_2, c_3\}, \phi_2 = \{c_2, c_3\}, \phi_3 = \{c_2\})$, the decision aggregation with the AA approach is demonstrated in Fig. 3.4.

For local region X_i, suppose that l_{mi} denotes the most frequent category and n_i denotes the total number of samples. Suppose that the probability of the data from l_{mi} in X_i is p_i^m, and $\hat{p}_i^m$ is the estimation of p_i^m. Then, the score function for cell e_i is evaluated by computing.

$$S(e_i) = \hat{p}_i^m G(p_i^m, \hat{p}_i^m, n_i), \tag{3.5}$$

Table 3.1: Example of the lookup table for the AA approach

Local Region	$\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$	Score of the Estimated Most Frequent Category	Cell Label
$X_{1,2}$	l_1,l_1, l_1	0.8	l_1
$X_{2,2,1}$	l_1,l_1, l_2	0.6	l_2
$X_{2,2,2}$	l_1,l_2, l_2	0.5	l_1
$X_{2,1,2}$	l_2,l_2, l_2	0.7	l_2
$X_{2,1,1}$	l_1,l_2, l_1	0.2	l_2
$X_{1,1,2}$	l_2,l_2, l_1	0.1	l_1
$X_{1,2,1}$	l_2,l_1, l_1	0.9	l_2

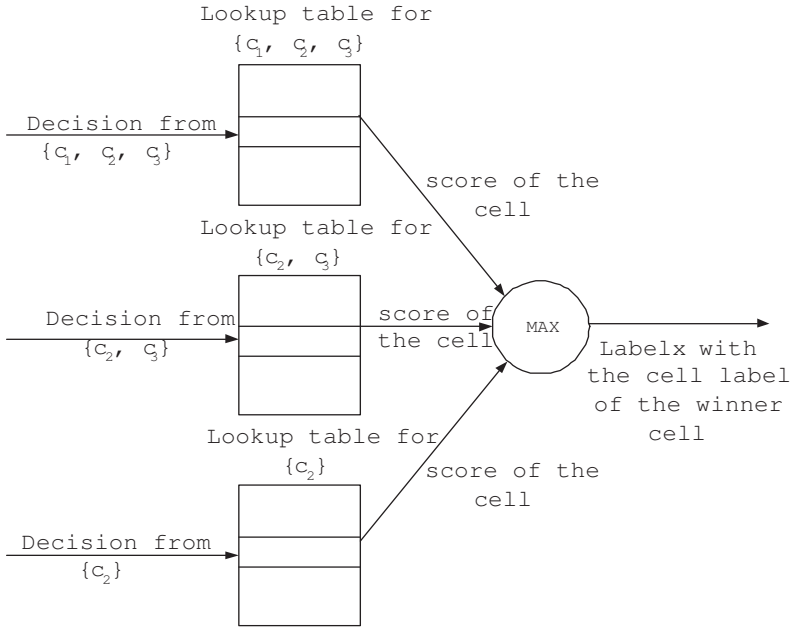

Figure 3.4: Example of the AA approach

where $G(p_i^m, \hat{p}_i^m, n_i)$ evaluates the confidence to estimate p_i by $\hat{p}_i^m$. To estimate $G(p_i^m, \hat{p}_i^m, n_i)$, a hypothesis test is constructed as follows:

$$H_0 : p_i^m = \hat{p}_i^m$$
$$H_1 : p_i^m \neq \hat{p}_i^m$$

Then, $G(p_i^m, \hat{p}_i^m, n_i)$ is estimated by the possibility that both type I and II errors of this test are below the thresholds α and ϵ (e.g. $\alpha = 0.05$, $\epsilon = 0.05$), respectively. For the estimation, the size of the test α, which is the lowest upper bound of type I error, is first fixed. Then, power function $\beta(\theta)$ for the test is constructed[2]. Finally, the probability that type II error is below threshold ϵ is estimated according to the power function of this hypothesis test. The power function is estimated as follows.

$$\beta(\theta) = P(W_1 > a + \sqrt{n_i}b) + P(W_2 < -a + \sqrt{n_i}b), \tag{3.6}$$

Where

$$a = 1.96\sqrt{\frac{\hat{p}_i^m(1-\hat{p}_i^m)}{p_i^m(1-p_i^m)}} \quad b = \frac{\hat{p}_i^m - p_i^m}{\sqrt{p_i^m(1-p_i^m)}}$$

and

$$W_1 \sim N(0,1) \quad W_2 \sim N(0,1)$$

The proof is given as follows. Let $\omega(\mathbf{x})$ represent the label of $\mathbf{x}$. For any $\mathbf{x} \in X_i$, we define the indicator function

$$I(\omega(\mathbf{x}) = l_{mi}) = \begin{cases} 1 & \omega(\mathbf{x}) = l_{mi} \\ 0 & \text{Otherwise.} \end{cases} \tag{3.7}$$

and the sum of the indicator functions

$$Y_i^m = \sum_{k=1}^{n_i} I(\omega(\mathbf{x}_k) = l_{mi}).$$

Suppose that the probability of the data from l_{mi} in X_i is p_i^m. There is

$$Y_i^m \sim BIN(n_i, p_i^m)$$

Consider the following test:

[2] θ is used to denote the unknown parameter in the model. For the specific problem in this thesis, θ is parameter p_i^m

$$H_0 : p_i^m = \hat{p}_i^m$$
$$H_1 : p_i^m = \hat{p}_i^{m'} (\hat{p}_i^{m'} > \hat{p}_i^m)$$

According to Neyman-Pearson Lemma, for a constant c, the critical region is defined by:

$$\begin{aligned} R_1 &= \{Y_i^m : \frac{\binom{n_i}{Y_i^m}(\hat{p}_i^{m'})^{Y_i^m}(1-\hat{p}_i^{m'})^{n_i - Y_i^m}}{\binom{n_i}{Y_i^m}(\hat{p}_i^{m})^{Y_i^m}(1-\hat{p}_i^{m})^{n_i - Y_i^m}} > c\} \\ &= \{Y_i^m : \left(\frac{\hat{p}_i^{m'}(1-\hat{p}_i^m)}{(1-\hat{p}_i^{m'})\hat{p}_i^m}\right)^{Y_i^m} > \frac{c(1-\hat{p}_i^m)^{n_i}}{(1-\hat{p}_i^{m'})^{n_i}}\} \\ &= \{Y_i^m : Y_i^m > k\}(\quad k \quad is \quad some \quad constants). \end{aligned}$$

Under H_0 hypothesis and the given test of size α_1, there is:

$$\begin{aligned} \alpha_1 &= P(Y_i^m > k) \\ &= P\left(\frac{Y_i^m - n_i\hat{p}_i^m}{\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)}} > \frac{k - n_i\hat{p}_i^m}{\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)}}\right). \end{aligned}$$

According to the normal approximation to the binomial distribution, there is

$$\frac{Y_i^m - n_i p_i^m}{\sqrt{n_i p_i^m(1-p_i^m)}} \to W \sim N(0,1).$$

Given the H_0 hypothesis, there is

$$\frac{Y_i^m - n_i\hat{p}_i^m}{\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)}} \to W \sim N(0,1).$$

We set $\alpha_1 = 0.025$. Therefore, the value of k can be solved as

$$k = 1.96\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)} + n_i\hat{p}_i^m.$$

Therefore, the most powerful critical region for this test is

$$R_1 = \{Y_i^m : Y_i^m > 1.96\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)} + n_i\hat{p}_i^m\}.$$

Since this conclusion holds for any $\hat{p}_i^{m'} > \hat{p}_i^m$, we further claim that R is also the most powerful critical region for the test

$$H_0 : p_i^m = \hat{p}_i^m$$
$$H_1 : p_i^m > \hat{p}_i^m$$

Therefore the power function of the test is as follows:

$$\begin{aligned}
\beta_1(\theta) &= P_\theta(Y_i^m \in R_1) \\
&= P(Y_i^m > 1.96\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)} + n_i\hat{p}_i^m) \\
&= P\left(\frac{Y_i^m - n_i p_i^m}{\sqrt{n_i p_i^m(1-p_i^m)}} > a + \sqrt{n_i}b\right) \\
&= P\left(W_1 > a + \sqrt{n_i}b\right)(W_1 \sim N(0,1)),
\end{aligned}$$

where

$$a = 1.96\sqrt{\frac{\hat{p}_i^m(1-\hat{p}_i^m)}{p_i^m(1-p_i^m)}} \quad b = \frac{\hat{p}_i^m - p_i^m}{\sqrt{p_i^m(1-p_i^m)}}$$

Similarly, we can get the most powerful critical region R_2 for the test

$$H_0 : p_{ij}^m = \hat{p}_{ij}^m$$
$$H_1 : p_{ij}^m < \hat{p}_{ij}^m$$

Given the size of test $\alpha_2 = 0.025$, R_2 is given as follows:

$$R_2 = \{Y_i^m : Y_i^m < -1.96\sqrt{n_i\hat{p}_i^m(1-\hat{p}_i^m)} + n_i\hat{p}_i^m\}.$$

and the power function is

$$\begin{aligned}
\beta_2(\theta) &= P_\theta(Y_i^m \in R_2) \\
&= P\left(W_2 < -a + \sqrt{n_i}b\right)(W_2 \sim N(0,1)).
\end{aligned}$$

Finally, let us consider the most powerful critical region and the power function for the hypothesis

$$H_0 : p_i^m = \hat{p}_i^m$$
$$H_1 : p_i^m \neq \hat{p}_i^m$$

Unfortunately, since $R_1 \bigcap R_2 = \phi$, there is no uniformly most powerful test for the hypothesis above. Instead, we construct a reasonable test for the hypothesis with the most powerful critical region R and the power function β as follows:

$$R = R_1 \cup R_2.$$

Then the power function $\beta(\theta)$ could be estimated as:

$$\begin{aligned} \beta(\theta) &= \beta_1(\theta) + \beta_2(\theta) \\ &= P(W_1 > a + \sqrt{n_i}b) + P(W_2 < -a + \sqrt{n_i}b). \end{aligned}$$

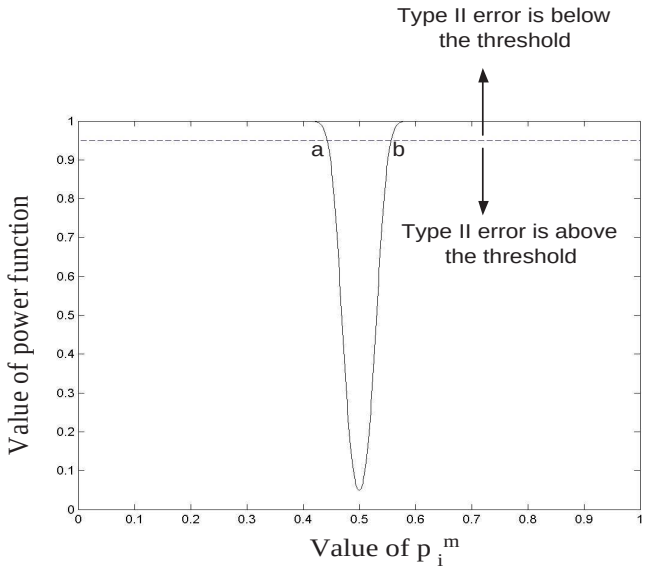

Figure 3.5: Example of power function

Type II error is evaluated by $1 - \beta(\theta)$. It is noted that the value of $\beta(\theta)$ is a function of p_i^m, $\hat{p}_i^m$, and n_i. In a specific test, $\hat{p}_i^m$ and n_i are known. Therefore, the

value of $\beta(\theta)$ is a function of p_i^m. Fig. 3.5 shows an example of power function, given $\hat{p}_i^m = 0.5$, $\alpha = 0.05$, $\epsilon = 0.05$, and $n_i = 1000$. Obviously, when $p_i^m \in [a, b]$, type II error is higher than the threshold. It is noteworthy that the range of p_i^m is $[0, 1]$[3]. If the value of p_i^m is considered as a random variable, the estimation of how likely type II error is below the threshold equals estimating how likely p_i^m is outside of $[a, b]$. Since there is no information about the distribution of p_i^m, it can be simply assumed that p_i^m follows a uniform distribution ($p_i^m \sim UNIF(0, 1)$). Now, the explicit equation for $G(p_i^m, \hat{p}_i^m, n_i)$ is given as:

$$G(p_i^m, \hat{p}_i^m, n_i) = 1 - |ab|, \tag{3.8}$$

where $|ab|$ is the length of segment ab. It is not difficult to show that $|ab|$ decreases, when n_i increases. Therefore, when there are more observations, the estimation of p_i^m by $\hat{p}_i^m$ is more likely to be correct. Finally, the detail of estimating $|ab|$ is explained. First, set a large number N_g (e.g., N_g is set to 10000 by default). Then, set $p_i^m = \frac{j}{N_g} \quad (j = 1, ..., N_g - 1)$, and the value of $\beta(\theta)$ for each p_i^m can be obtained. Finally, the ratio of p_i^m, whose corresponding value of $\beta(\theta)$ is bigger than $1 - \epsilon$ is computed. $|ab|$ is estimated by this ratio.

$$|ab| = \frac{\#\{p_i^m : \beta(\theta) > 1 - \epsilon\}}{\#\{p_i^m : 0 \leq p_i^m \leq 1\}}. \tag{3.9}$$

An algorithm for the AA approach can be described as follows:

Algorithm: *AA Approach*

Input: $F = \{\phi_1, \phi_2, ..., \phi_n\}$
l_m = -1;
Temp = -1;
For i =1 to n
Build the lookup table for ϕ_i
Given the input vector $\mathbf{x}$, find the corresponding cell e_i

[3]Since the data may be noisy, it is likely that the value of p_i^m can be very small in certain situations

in the lookup table of ϕ_i
For the cell e_i, identify its label $\hat{l}_{mi}$ and calculate the score $S(e_i)$
If $S(e_i) >$ Temp
Temp $= S(e_i)$
$l_m = \hat{l}_{mi}$
End
End
Label the input vector $\mathbf{x}$ with l_m
Output: class label of $\mathbf{x}$

Extension of the Score Function

Now, more general ways are examined to construct the score function for the proposed AA approach.

Multiplication Rule In Eq. 3.5, the score function is constructed as the multiplication of $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$, and can be extended to a more general case,

$$S(e_i) = \hat{p}_i^m G^k(p_i^m, \hat{p}_i^m, n_i) \quad (k > 0). \tag{3.10}$$

In Eq. 3.10, k is regarded as the weight that is assigned to control the risk of overfitting. Since $G(p_i^m, \hat{p}_i^m, n_i) \in [0, 1]$, the larger the k value is, the weaker the system is for reducing the risk of overfitting. It is clear that Eq. 3.5 is a special case of Eq. 3.10 when $k = 1$.

Addition Rule Similarly, the score function can be constructed by the addition rule as follows.

$$S(e_i) = \hat{p}_i^m + k \times G(p_i^m, \hat{p}_i^m, n_i) \quad (k > 0). \tag{3.11}$$

In Eq. 3.11, k is also the weight to control the risk of overfitting. The larger the k value is, the better the system can reduce the risk of overfitting.

The score functions that are built with the multiplication and addition rules might exhibit different properties. For example, the score function with the multiplication rule is less sensitive to the change of $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$. The reason is that both $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$ are in the range of [0,1]. A change in either of them is mitigated by the other part through multiplication. However, both of these two rules can effectively reduce the risk of overfitting through $G(p_i^m, \hat{p}_i^m, n_i)$. Therefore, the score function, built with either of them, should help the whole system to achieve a robust performance.

GESA with Multiple Representations

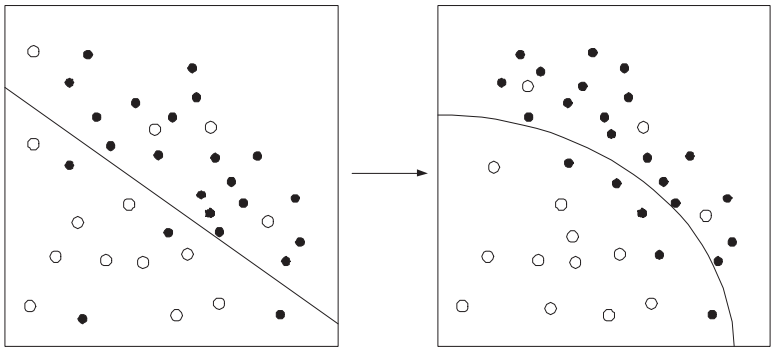

Figure 3.6: Decision boundary in feature and data space

In the previous discussion, GESA is assumed to takes the raw representation of the data. Also, GESA can be easily extended to incorporate multiple representations. When multiple representations of the data are used, the base classifiers of GESA conduct the classification in the feature space. However, the decision boundary in the feature space can be mapped into the original data space. The decision of the base classifiers in GESA can still be regarded as a partition of the data space as described in Fig. 3.6. Therefore, the previous conclusions on GESA are still valid with multiple representations. The architecture of GESA with multiple representations is demonstrated in Fig. 3.7. In this situation, the concept of the knowledge space Ω should also be extended. The classifiers in the knowledge

space Ω are not only distinguished by the training algorithm, initial conditions, and parameters but also by the input representations they take.

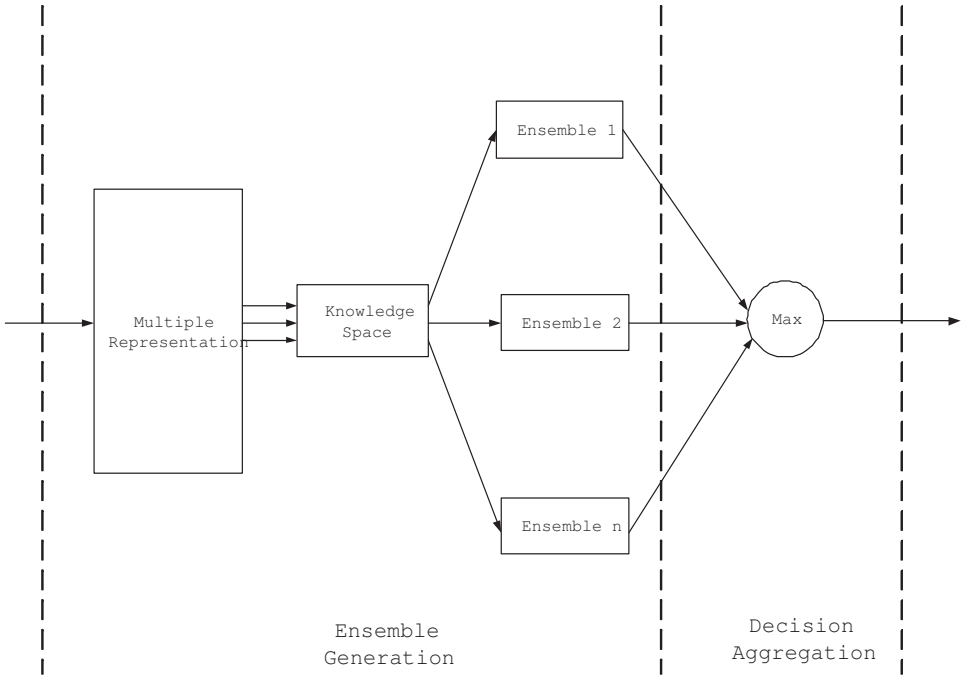

Figure 3.7: Architecture of GESA with multiple representations

3.3 Discussion on the Properties of GESA

3.3.1 Capability to Reduce the Risk of Overfitting

GESA reduces the risk of overfitting by dynamically selecting an appropriate ensemble of classifiers. The score function $S(e_i)$ in Eq. 3.5 is composed of two parts: $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$. A larger ensemble partitions the data space finer, which might lead to a better fit of the data and favors the value of $\hat{p}_i^m$. However, each local region X_i has fewer observations for the finer partition. It is noticed that $G(p_i^m, \hat{p}_i^m, n_i)$ is a function of n_i, but their relationship is not very evident. First, consider the relation between n_i and type II error. The conclusion that type II error

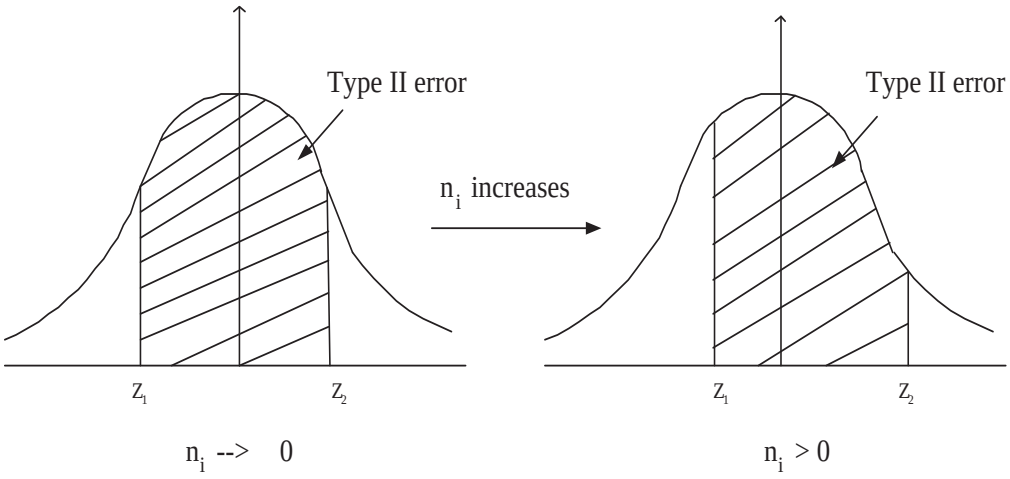

Figure 3.8: Relationship between type II error and the number of observations n_i

is reduced when n_i increases is intuitive. When p_i^m is given (assume $\hat{p}_i^m > p_i^m$), the relation between n_i and type II error is shown in Fig. 3.8 ($Z_1 = -a + \sqrt{n_i}b$ and $Z_2 = a + \sqrt{n_i}b$). Type II error is represented by the shaded area in Fig. 3.8. When n_i increases, the shaded area moves to the right, along the axis. Since the interval between Z_1 and Z_2 is always $2a$, the shaded area decreases when the area moves to the right, along the axis; that is, type II error monotonically decreases with the value of n_i. The same conclusion is obtained by assuming $\hat{p}_i^m < p_i^m$. Since $G(p_i^m, \hat{p}_i^m, n_i)$ is calculated by the probability that type II error is below the threshold given α is fixed, it is conclude that $G(p_i^m, \hat{p}_i^m, n_i)$ monotonically increases with n_i. Obviously, the larger value of $G(p_i^m, \hat{p}_i^m, n_i)$ demands a coarser partition of the data space. Recall that the maximum value of $S(e_i)$ must be achieved by establishing a balance between $\hat{p}_i^m$ and $G(p_i^m, \hat{p}_i^m, n_i)$. Therefore, the term $G(p_i^m, \hat{p}_i^m, n_i)$ ensures that the data space will not be partitioned too fine by the classifier ensembles, reducing the risk of overfitting.

3.3.2 Asymptotic Performance

Another issue is the asymptotic performance of the proposed GESA. The previous discussion shows that the value of n_i is positively correlated with that of

$G(p_i^m, \hat{p}_i^m, n_i)$. In fact, when $n_i \to \infty$,

$$\beta(\theta) = P(W_1 > +\infty) + P(W_2 < +\infty)$$

or

$$\beta(\theta) = P(W_1 > -\infty) + P(W_2 < -\infty),$$

where $W_1, W_2 \sim N(0, 1)$. In both situations, there is

$$\beta(\theta) = 1. \tag{3.12}$$

Therefore, when $n_i \to \infty$, the probability that type II error is above the threshold is 0, which indicates that $G(p_i^m, \hat{p}_i^m, n_i) = 1$. At the same time, when $n_i \to \infty$, $\hat{p}_i^m \to p_i^m$. Then, the score function is reduced to

$$S(e_i) = p_i^m. \tag{3.13}$$

Consistently, the AA approach chooses the ensemble which generates the local region with the highest value of $\hat{p}_i^m$. In addition, $\hat{p}_i^m$ converges to the real probability p_i^m. From the the discussion in Section 3.2.3, it is concluded that the decision aggregation is asymptotically optimal over all the ensembles generated in F. Since F contains all the possible subsets of Ω, it is safe to conclude that GESA is asymptotically optimal over any subsets in Ω. In other words, given Ω, the newly devised GESA is one of the asymptotically optimal MCSs.

3.4 Summary

In this chapter, the objective is to understand MCSs from the learning point of view. It is also crucial for MCSs to follow some general principles of the supervised learning methods. Although there is no general conclusion on the superiority of the loss function and the regularization method, such a discussion is still beneficial to design a robust MCS. In this thesis, the claim is that GESA achieves a reliable and

good performance, since it can dynamically tweak the trade-off between the ability to fit the data adequately and reducing the risk of overfitting at the local level. Also, the discussion demonstrates that GESA is asymptotically optimal. However, it is not very data-adaptive and its computation cost is high. In Chapter 4 and 5, some more practical designs of MCSs, which are regarded as the extension of GESA, are elaborated.

Chapter 4

Generalized Adaptive Ensemble Generation and Aggregation (GAEGA) Approach

4.1 Objective

The focus of GESA is on the robustness of the classification system. Other important factors, which can also affect the performance of the classification system, are overlooked. It is known that the effectiveness of the input representation and type of classifiers is data dependent. Duda et al. [17] have reported the No Free Lunch Theorem and Ugly Duckling Theorem which shows that no one representation and classifier is universally better than any other one. One advantage of the use of MCSs for pattern classification is that the system is able to dynamically select the effective representations and classifiers for the classification purpose. However, many approaches, including the proposed GESA, are not completely data-adaptive. For methods such as adaboost, Random Subspace (RS), and bagging, only one type of representation and classifier is allowed to build the classification system. It seems that the data-adaptiveness is not fully explored in this type of design. It is true that there are many designs which take multiple representations and heterogeneous classifiers. However, they are still not completely data-adaptive.

In this thesis, it is proposed that the data-adaptive design of MCSs should not only generate and select the effective decisions by correctly recognizing the effective representations and classifiers, but also exclude the noise representations and classifiers as much as possible. One the one hand, it is more likely that an ensemble can generate the effective decisions, when there are more representations and classifiers available. Therefore, the odds that the classification system can label the data correctly is better. On the other hand, it is also more likely that the noise representations and classifiers which generate the noise decisions are introduced. Of course, it can be argued that many aggregation methods are able to adaptively learn the weight of the decisions and select the appropriate classifier or combinations of them to label the data. However, the learning algorithms themselves might not be perfect. When there are more noise decisions, the algorithm's performances can worsen. To clarify this point, consider the analogy between the decision aggregation and classification by the individual classifiers. The decision aggregation can be regarded as a special type of classification task, in which the decisions of the base classifiers act as the input data and the aggregation method works as a special type of classifier. Table 4.1 indicates that the error rate of the Support Vector Machine (SVM) increases, after the six noise features are introduced [42]. The argument is that the noise decisions play the same role for the decision aggregation method as the noise features do for the SVM classifier in the example. Obviously, the data-adaptive design can improve the performance of MCSs significantly. Another difficulty of the newly developed GESA is its high computation cost. Again, for simplicity, consider the case in which MCSs take a heterogeneous ensemble of classifiers with the raw input representation. Let h denote the size of the knowledge space Ω. For GESA, the computation costs for creating and searching the ensemble are $O(2^h)$. Obviously, GESA is time consuming and not scalable to a large set of Ω.

To overcome the difficulty of GESA, the Generalized Adaptive Ensemble Generation and Aggregation (GAEGA) approach is further proposed for the design of MCSs. Compared with GESA, GAEGA still achieves a robust performance while having a less expensive computation cost and being data-adaptive.

Table 4.1: Effects of noise feature (from Table 12.2 in Page 385 in [42])

	Test Error	(SE)
Methods	No Noise Features	Six Noise Features
1 SV Classifier	0.450(0.003)	0.472(0.003)
2 SVM/poly 2	0.078(0.003)	0.152(0.004)
3 SVM/poly 5	0.180(0.004)	0.370(0.004)
4 SVM/poly 10	0.230(0.003)	0.434(0.002)
Bayes	0.029	0.029

4.2 Details of GAEGA

4.2.1 Basic Assumption and Design Concept

To simplify the discussion, we first focus on designing MCSs with a set of heterogeneous classifiers, each of which takes the raw input representation. As we discussed previously, it is advantageous to combine the decisions of the classifiers by varying the input representations, initial condition of the architecture, the training algorithm of the classifiers etc. GAEGA doesn't put any constraints on the way to process the input and generate the classifiers. The goodness of fit of the ensemble on the validation data, measured by criteria $\hat{\Theta}_\phi$ in this thesis, is used to determine which classifier or the combination of them are effective. Then, a collection of different ensembles are generated, each of which fits the validation data with a different degree.

A test data point $\mathbf{x}$ is classified by all the generated ensembles. GAEGA adaptively selects the decision of the most appropriate ensemble to label the data point, which is determined by two factors: (1) how well the ensemble fits the data in the local region which contains $\mathbf{x}$. The local region is determined by the decision boundaries of the base classifiers in the ensemble. (2) how likely the ensemble over-fits the data in the local region.

4.2.2 Ensemble Generation

GESA simply generates all the possible subsets of Ω. Therefore, the computation costs for generating and searching the ensemble is in the exponential order of h[1]. For GAEGA, the ensemble generation needs to fulfill two requirements: (1) the computation cost to generate the ensemble must be reasonable. For example, the computation cost with $O(h^k)$ ($k = 1, 2, 3$ ..) is acceptable . The computation cost for searching the decisions among the ensembles must also be in the polynomial order of h. This property allows GAEGA to be scalable to the large set of Ω. (2) The generated ensembles must fit the validation data set with different degrees as much as possible. At least one of the ensembles must minimize $\hat{\Theta}_\phi$. In addition, each ensemble needs to include the effective classifiers adequately, and exclude the noise classifiers as much as possible. The purpose is to help the whole system to achieve the desired data-adaptive performance. Consider the following proposition

Proposition 1 Given two ensembles ϕ_i and ϕ_j, if $\phi_i \subseteq \phi_j$, then $\hat{\Theta}_{\phi_i} \geq \hat{\Theta}_{\phi_j}$.

If the ensemble consists of more base classifiers, the data space has the potential to have more partitions. From the definition of $\hat{\Theta}_\phi$, it is not difficult to prove the above proposition with some simple algebra. This proposition indicates that larger ensemble tends to fit the validation data better. As we discussed in the previous section, GAEGA generates a collection of different ensembles, each of which fits the validation data globally with a different degree. We want the collection of ensembles $F = \{\phi_i\}$ to have the property that

$$\phi_1 \subseteq \phi_2 ... \subseteq \phi_n,$$

Where $n \leq h$. Then, from Proposition 1,

$$\hat{\Theta}_{\phi_1} \geq \hat{\Theta}_{\phi_2} ... \geq \hat{\Theta}_{\phi_n}.$$

In addition, GAEGA demands that ϕ_n is large enough to minimize $\hat{\Theta}_{\phi_n}$ as

[1] h is the size of the Ω

much as possible. The collection F is generated iteratively in the training phase by a greedy algorithm. In the kth iteration, a new ensemble ϕ_k is generated by adding a classifier c_j ($c_j \in \Omega$, Ω is a pool of available classifiers) to ensemble ϕ_{k-1}, which is generated in the $(k-1)$th iteration. That is,

$$\phi_k = \phi_{k-1} \cup c_j.$$

The classifier c_j is selected to minimize $\hat{\Theta}_{\phi_k}$. There are two stopping criteria in the training phase: (1) $\hat{\Theta}_\phi$ is reduced to zero, or (2) $\hat{\Theta}_\phi$ could not be further decreased. In both of these two situations, $\hat{\Theta}_\phi$ is minimized. The algorithm for the ensemble generation is summarized as follows.

Algorithm: *Greedy Search*

Ω: all the available classifiers, each of which is trained on a certain input representation
c_j: the jth classifier
ϕ_i: the ith ensemble
F : the collection of the ensembles
$\hat{\Theta}_{\phi_k}$: the value of $\hat{\Theta}_\phi$ for ensemble ϕ_k
Input: $\Omega = \{c_1, ...c_h\}$
$\phi_0 = NULL$
$\hat{\Theta}_{\phi_0} = LargeNumber$
$F = NULL$
$k = 0$
Do
    $k = k + 1$
    Find the classifier c_j such that $\phi_k^j = \phi_{k-1} \cup c_j \quad (c_j \in \Omega)$
    and $\hat{\Theta}_{\phi_k^j}$ is minimized
    $\phi_k = \phi_k^j$
    $F = F \cup \phi_k$
    $\hat{\Theta}_{\phi_k} = \hat{\Theta}_{\phi_k^j}$

While ($\hat{\Theta}_{\phi_k} > 0$ and $\hat{\Theta}_{\phi_k} < \hat{\Theta}_{\phi_{k-1}}$)
Output: collection of ensembles F

After explaining the ensemble generation, several architectures of GAEGA are examined.

4.2.3 Architecture of GAEGA

The architecture of the proposed GAEGA consists of two layers, where the data is sequentially processed. In the ensemble generation layer, a set of heterogeneous classifiers is applied on the data, and a greedy algorithm is proposed to adaptively generate a collection of possible ensembles of classifiers. In the decision aggregation layer, the AA approach is employed to compare the decisions of all the generated ensembles, and reach the final decision, based on the proposed score function.

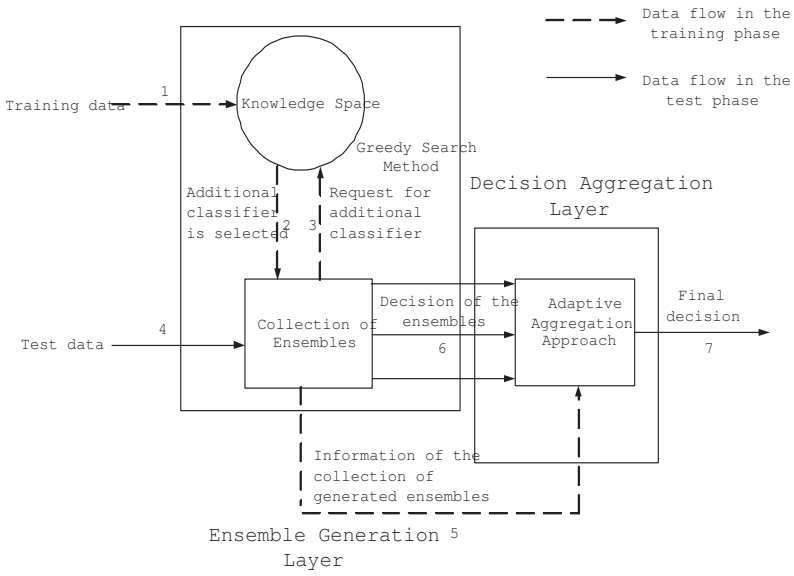

Figure 4.1: Architecture of GAEGA

The architecture of GAEGA is signified in Fig. 4.1. The dash lines denote the

data flow in the training phase and the solid lines indicate the data flow in the test phase. The numbers on the lines represent the steps in the process. The data is forwarded to the knowledge space Ω, which is a pool of available classifiers (step 1). A greedy algorithm is applied to iteratively generate a collection of possible ensembles (step 2 and 3). In addition, the information about the process of the ensemble generation is sent to the decision aggregation layer for the decision combination (step 5). The test data is fed into the generated ensembles (step 4). The multiple decisions are generated by all the ensembles, which are created in the ensemble generation layer (step 6). The final decision is made by combining this information using the AA approach (step 7).

4.3 Discussion on the Properties of GAEGA

There is a concern about the following properties of GAEGA: (1) how does it reduce the risk of overfitting? (2) what is the asymptotical performance? (3) what is the computation cost for creating and searching the ensemble?

Since GAEGA also adopts the AA approach for the decision aggregation, the mechanism for GAEGA to reduce the risk of overfitting is similar to that for GESA. Also, like the discussion in Section 3.3.2, it is readily shown that GAEGA is asymptotically optimal over all the ensemble generated in F. However, since the ensembles in F does not cover all the local regions which can be generated by the subsets of Ω, GAEGA is not asymptotically optimal over the knowledge space Ω. The reason for such a design is to balance the performance and the computation cost. If all the local regions are generated, the computation cost for the ensemble generation is $O(2^h)$. In contrast, the computation cost for the ensemble generation in GAEGA is only $O(h^2)$. In addition, the computation cost to search the ensemble in GAEGA is $O(h)$. Since both the computation cost for generating and searching the ensemble is in polynomial order of h, significantly lower than GESA, it is expected that GAEGA is much faster than GESA.

4.4 GAEGA with Multiple Representations

In the previous discussion, it is assumed that MCSs take the raw data as the input. Similar to GESA, GAEGA can also be easily extended to incorporate multiple input representations.

4.4.1 Architecture

We first extend the concept of knowledge space Ω. In the extended knowledge space Ω, the classifiers are distinguished by the training algorithm, as well as the input representations, initial conditions, and parameters. The architecture of the extended GAEGA consists of three layers, in which the data is sequentially processed. In the input representation layer, the raw data is preprocessed by various representation methods. Then, the transformed data is fed into the ensemble generation layer. In this layer, a set of heterogeneous classifiers is applied on different representations and a greedy algorithm is proposed to adaptively generate a collection of possible ensembles of classifiers. Since each base classifier takes a certain representation, the selection of the base classifiers is also the selection of the useful input representations. Finally, in the decision aggregation layer, the AA approach is employed to compare the decisions of all the generated ensembles and make the final decision.

The architecture of GAEGA is conveyed in Fig. 4.2. The dash lines denote the data flow in the training phase and the solid lines indicate the data flow in the test phase. The number on the lines represent the steps in the process. Each Input Representation Processor (IRP) is an expert to process a certain type of input information. After the training data is fed into various IRPs (step 1), various representations of the data are forwarded to the knowledge space Ω (step 2). The greedy algorithm is applied to iteratively generate a collection of possible ensembles (step 3 and 4)[2]. The information about the selected input representations is fed

[2]Strictly speaking, the base classifiers of GAEGA conduct the classification in the feature space. However, the decision boundary in the feature space can always be mapped into the original data space. Therefore, the decision of the base classifiers in GAEGA is still be regarded as a partition of the data space

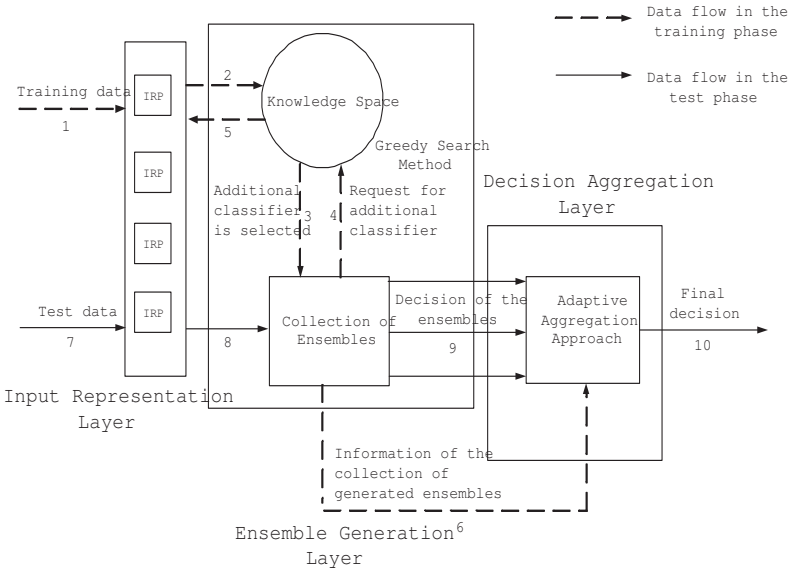

Figure 4.2: Architecture of GAEGA with multiple representations

back to the input representation layer to process the test data (step 5). In addition, the information about the process of the ensemble generation is sent to the decision aggregation layer for the decision combination (step 6). Test data is first processed by the selected representation methods (step 7), and then fed into the generated ensembles (step 8). Multiple decisions are generated by all the ensembles, which are created in the ensemble generation layer (step 9). The final decision is made by combining these information using the AA approach (step 10).

4.4.2 Application of GAEGA to the Classification of Time Series Data

Motivation

One applications of GAEGA with multiple representations is to classify time series data. MCSs have unique advantages for this task. Time series data contains dynamic information and its underlying models vary from one application to another. There are two disadvantages in regarding time series data as the general type of

data: (1) the temporal information can not be fully utilized for the classification purpose, (2) time series data is usually high dimensional. In this situation, the dimension reduction is necessary, since it can cause the overfitting problem, and render the computation cost unacceptably high. However, it is difficult to build an individual representation to handle all kinds of time series data due to its extremely diversified sources. Therefore, it makes sense to adopt a strategy, where time series data is represented in several ways and all the decisions of the classifiers are aggregated in the final stage. With this strategy, the complete information of the temporal patterns can be well identified by the aggregation of the classifiers trained on different representations of the input data. Most importantly, this strategy does not assume the comprehensiveness of any one representation; that is, each representation processes only part of the temporal information. Consequently, the complexity of the design of the representation method is significantly reduced. It is noteworthy that certain representations do not seem to be useful or reasonable for some problems. However, a data-adaptive design of the MCS enable it to consistently select the effective representations and filter the noise ones. Multiple Input Representation-Adaptive Ensemble Generation and Aggregation (MIR-AEGA) is used to denote the classification of time series data with GAEGA, since it involves those special input representations for the time series data.

Discussion on the Representation Methods for Time Series Data

In the following, several popular representation techniques for time series data are discussed.

Piecewise Approximation Representation The idea of piecewise approximation representation is to partition the initial sequence into several segments. In each segment, the subsequence is represented by a linear function or a constant. Then, the overall sequence is represented by a collection of linear functions or constants. The most popular piecewise approximation representations are the Piecewise Linear Approximation (PLA) [66] and Piecewise Aggregate Approximation (PAA) [105]. In PAA, the series is represented as a sequence of box basis functions. Each of the box the is same length. Then,

the subsequence in the box is represented by its mean value. In PLA, time series data is segmented into a collection of straight lines. Then, the sequence is represented by their lengths and slopes. In this thesis, the original PLA is slightly revised so that the lengths of all the straight lines are equal. After a linear regression is applied for each interval to generate a straight line function, time series data is represented by the slopes of these straight lines. One advantage of the piecewise approximation representations is that they can significantly reduce the dimensionality of the time series data. The weakness is that some important local information might be missing.

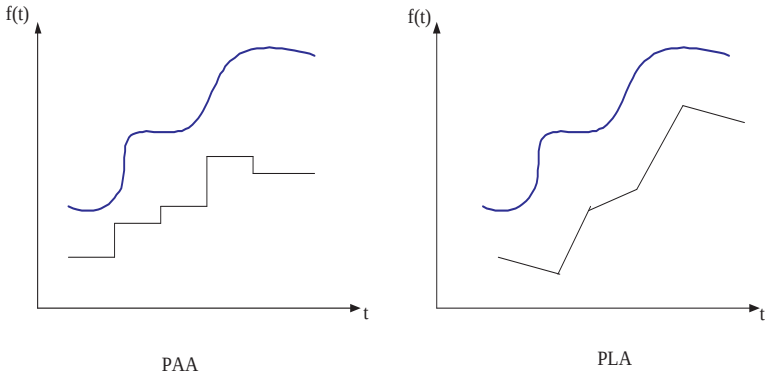

Figure 4.3: PAA and PLA

Simple Preprocessing Representation One possible solution is to use the data, with only minimal transformation to obtain manageable sequences. Keogh and Pazzani [56] have identified six types of distortions in time series data, including linear drift, noise, offset translation, and amplitude scaling. To obtain more manageable sequences for traditional classifiers, it is necessary to remove one or more of the distortions from the original data. In this thesis, three techniques are adopted to remove the possible distortions. The first method is used to remove the distortion of offset translation. Let X represent the original time series such that processed series Y is obtained as follows:

$$Y = X - \overline{X},$$

where $\overline{X}$ is the mean value of series X. The second method is employed to remove the linear trend in the series. First, a linear regression between X and T (T is the variable for time) is constructed. Let $\hat{X}$ represent the regressor. Y is obtained as follows.

$$Y = X - \hat{X}.$$

The third technique removes the noise in the data and represents the data by its derivatives. How to represent time series data with the combination of the Double Exponential Smoothing (DES) and Differentiation Generator (DG) has been discussed in the literature [14]. Among the popular schemes to produce a smoothed time series is the exponential smoothing scheme which weights past observations using exponentially decreasing weights. The family of the exponential smoothing schemes includes Single Exponential Smoothing (SES), Double Exponential Smoothing (DES) and Triple Exponential Smoothing (TES). Their difference lies in the number of parameters that are used to represent the trend and seasonal information in the time series data. For example, the function of DES is given as.

$$\begin{aligned} y_t &= \alpha f_c(t) + (1-\alpha)(y_{t-1} + b_{t-1}) \\ b_t &= \gamma(y_t - y_{t-1}) + (1-\gamma)b_{t-1} \\ b_0 &= \frac{(f_c(t_2) - f_c(t_1)) + (f_c(t_3) - f_c(t_2)) + (f_c(t_4) - f_c(t_3))}{3}, \end{aligned}$$

where, y_i is the estimated value space, f_c is the input time series data, and α is the smoothing rate in the interval [0,1]. When $\alpha \to 1$, the smoothed data is exactly the same as the original data. γ is the learning rate. The larger the value of γ is, the more the estimated data is affected by the previous learning. Obviously, different combinations of the parameters generate different series of temporal data. The DES(α, γ) denotes for the algorithm with parameters α and γ. Another simple processing technique involves mapping the original sequences into their first derivative spaces [28]. This approach is further extended by introducing the DG [14]. $DG(p, q)$ has two parameters: *order*

and *distance*, represented by p and q, respectively. DG is mathematically expressed as

$$DG_t(p,q) = DG_{q+t}(p-1,q) - DG_t(p-1,q).$$

The DG has several functionalities. It can remove the effects of the initial values. For example, the effects of the shift can be eliminated by DG(1,1). In addition, DG can remove the trend information in the time series data by differentiating the given temporal data, until it becomes stationary. This is helpful to focus on the stationary information in the time series data. The advantage of the simple preprocessing techniques is their simplicity and effectiveness in correcting certain types of distortions.

Transformation-Based Representation The idea behind transformation-based representations is to map the initial sequences from the time domain to another domain which is more manageable, and then represent each original sequence with a point in the new space [5].

One of the most widely used techniques is the Fourier Transformation (FT). It is based on the principle that any time series data can be represented by combining a collection of sine and cosine functions. In the FT, time series data is mapped from the time domain to the frequency domain. FT provides the periodicity information for the data, and is effective in analyzing the periodic, time-invariant, and stationary time series data.

The Wavelet Transformation (WT) is another important transformation-based representation. It expands the function by a series of more generalized base functions, other than the sine and cosine functions, called wavelets. The localizing property of wavelets makes it possible to model the transient event with a small number of coefficients by the wavelet expansion. The wavelet systems, which form a complete orthogonal system for $L^2(R)$, are generated from a single scaling function or wavelet by simple scaling and translation as follows [9]:

$$\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi(2^j t - k) \quad j,k \in Z,$$

where Z is the set of all integers and factor $2^{\frac{j}{2}}$ maintains a constant norm independent of scale j. Wavelet systems are distinguished from others by their different base functions. One of the most popular systems is the Haar wavelet system which expands function $f(t)$ with a series of scaling functions $\varphi(t)$ and wavelet functions $\psi(t)$ such that

$$f(t) = \sum_{k=-\infty}^{\infty} c_k \varphi(t-k) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi(2^j t - k),$$

where

$$\varphi(t) = \begin{cases} 1 & t \in [0,1) \\ 0 & \text{otherwise} \end{cases}$$

and

$$\psi(t) = \begin{cases} 1 & t \in [0, \frac{1}{2}) \\ -1 & t \in [\frac{1}{2}, 1) \\ 0 & \text{otherwise.} \end{cases}$$

Both of the FT and WT have been widely applied in processing time series data including, indexing, retrieval, and classification [12, 73, 76, 103].

Finally, one of the vital representations for time series data is the Autoregressive Moving Average (ARMA) model which represents a potentially long time series by a limited dimensional vector [18]. Generally, time series data represented by ARMA(p,q) is written as

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + \beta_1 \zeta_{t-1} + \beta_2 \zeta_{t-2} + ... + \beta_q \zeta_{t-q} + \zeta_t,$$

where $\zeta_k (k = 1, ..q)$ is the noise of the time series data. A similar representation is the coefficient of the Linear Predictive Coding (LPC), which is computed by minimizing the difference between linear auto regressive model and the actual time series data [81].

Subspace Projection Representation Also, time series data can be simply regarded as high dimensional data. In this situation, it is beneficial to reduce the dimensionality of the data by projecting it into a subspace with a lower dimension, as observed in Fig. 4.4. Statistically, it is optimal to project the data into an orthogonal subspace that captures as much of the variations in the data as possible. One of the most popular methods is the Karhunen Loeve (KL) transformation [25]. The axis of the space are represented by the eigenvectors of the covariance matrix of the data, whose corresponding eigenvalues are above a certain threshold. In the random projection, the original high dimensional data is projected into a lower dimensional subspace by using a random matrix [11]. Achlioptas [2] has shown that j^{th} element of i^{th} axis r_{ij} can be generated with a very simple distribution as follows

$$r_{ij} = \begin{cases} \sqrt{3} & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -\sqrt{3} & \text{with probability } \frac{1}{6}. \end{cases}$$

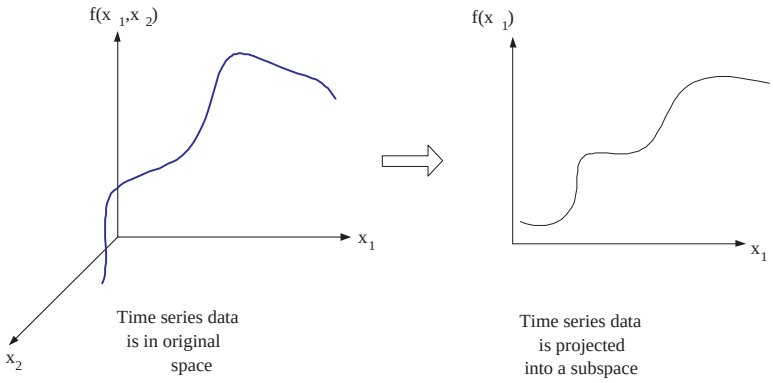

Figure 4.4: Subspace projection representation

Discussion of the Classifiers for Time Series Data

Although MIR-AEGA emphasizes the importance of the input representation for the classification of time series data, it is also crucial to pre-select the appropriate

base classifiers. Keogh et al. [55] have conducted empirical studies on time series data mining. The results indicate that the nearest neighbor classifier gives an impressive performance for time series data. Although the nearest neighbor classifier by itself is not robust for classifying time series data, it might be a good candidate for the base classifier in MIR-AEGA. Another criterion for the base classifier in MIR-AEGA is the computation cost. Since time series data are usually high dimensional, it seems impractical to use the classifiers that are computationally complex. For example, although DTW is also effective, its computation cost is $O(nd^3)$ [3]. Moreover, the empirical study shows that DTW is very time-consuming.

Architecture and Working Procedures

Similar to that of GAEGA, the architecture of MIR-AEGA consists of three layers, in which the data is sequentially processed. MIR-AEGA is characterized by the design of the input representation layer. It is composed of a collection of Temporal Representation Processors (TRPs), each of which is an expert in a certain type of temporal information. Any of the representation methods for time series data or their combinations can be used to design TPR, including FT, smoothing, and random projection.

4.5 Summary

In this chapter, the weakness of GESA is identified and GAEGA is proposed to overcome the recognized difficulties. GAEGA is designed to take into account the robustness, data-adaptiveness, and computation cost of the system. Therefore, GAEGA is more practical. In addition, as an application of GAEGA to the classification of time series data, MIR-AEGA is also discussed. In Chapter 6, the strengths of GAEGA and MIR-AEGA are exhibited by conducting the experiments on the benchmark data sets.

[3] n is the number of the training data and d is the dimension of the data

Chapter 5

Bootstrapped Adaptive Ensemble Generation and Aggregation (BAEGA) Approach

5.1 Motivation

Bootstrap is a general purpose approach to statistical inference, falling within a broader class of resampling methods. Bootstrap is used for estimating the sampling distribution of an estimator by resampling with replacement from the original sample. Usually, the purpose is to derive estimates of the standard errors and confidence intervals of a population parameter such as a mean, median, proportion, odds ratio, correlation coefficient, and regression coefficient. The bootstrap method has also been widely applied in the design of MCSs. In bagging, one of the simplest examples, each bootstrapped data set is used to train a different base classifier, and the final classification decision is established by the vote of each base classifier. Similarly, the bootstrap method can also be used to improve the classification performance of our proposed methods, but in a different way. The most straightforward way is to bootstrap the training data, which is used to train the base classifier. Then, the base classifier of GESA is trained by the bootstrapped sample. However, the information might not be fully utilized in this way. In each round, there are

observations which are left out of the bootstrapped samples. Clearly, those observations can help the aggregation of the decisions of the base classifiers. Rao and Tibshirani [86] have discussed a similar idea in the *out of bootstrap*. Therefore, it is not very appropriate to directly apply the bootstrap method to GESA.

In this section, the Bootstrapped Adaptive Ensemble Generation and Aggregation (BAEGA) approach is proposed for the design of MCSs. BAEGA is an alternative improvement of GESA. Although BAEGA shares characteristics with GAEGA, there is a fundamental difference between them in how the data is manipulated.

5.2 Background of Bootstrap Methods

The bootstrap method is a general tool for assessing statistical accuracy. The idea is to randomly draw the data N times with replacement from the sample, where N is the sample size. This is done B times, producing B bootstrap data sets. Then, the statistical model is examined with each of the B data sets [17]. Fig. 5.1 is a schematic diagram of the bootstrap method from [24]. On the left is the real world. An unknown distribution F generates the observed data $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..\mathbf{x}_n)$ by random sampling. A statistic of interest from $\mathbf{x}$ could be computed, say, the standard deviation of $\mathbf{x}$. On the right side of the diagram is the bootstrap world. In the bootstrap world, the empirical distribution F' generates bootstrap samples $\mathbf{X^*} = (\mathbf{x^*}_1, \mathbf{x^*}_2, ..\mathbf{x^*}_n)$ by random sampling, from which the bootstrap replications of the statistic interest can be computed. According to [24], the advantage of the bootstrap world is that the replication of the statistic interest can continuously be computed many times. Consequently, the probabilistic calculation is done directly. One of the examples is to use the observed variability of the replication of the statistic interest to estimate its standard deviation. There are two practical problems in the bootstrap analysis: (1) The entire probability mechanism P needs to be estimated from the data. In the diagram, it is the step indicated by $\mathbf{X} \rightarrow P'$ (2) The bootstrap data is simulated from P', according to the relevant data structure. In the diagram, it is the step indicated by $P' \rightarrow \mathbf{X^*}$ [24]. Besides the resampling with replacement, other bootstrap schemes have also been reported in the literature, in-

cluding the smooth bootstrap, parametric bootstrap, case resampling, resampling residuals, and wild bootstrap [17].

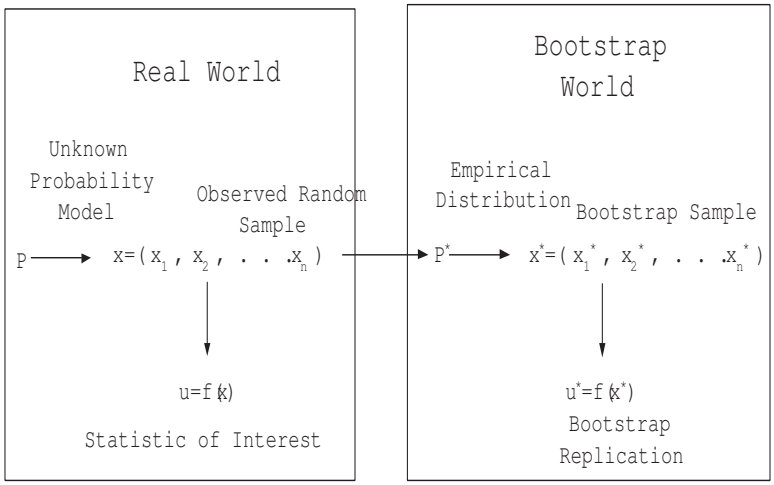

Figure 5.1: Schema of bootstrap method [24]

5.3 Details of BAEGA

With BAEGA, the goal is to achieve a data-adaptive and robust design of MCSs with a reasonable computation cost. BAEGA adopts the same ensemble selection and decision aggregation methods as GAEGA. However, BAEGA employs a different way to manipulate the data to train the base classifiers and aggregation rule.

5.3.1 Data Manipulation

Training the Base Classifiers

BAEGA does not split the training data to train the base classifier and aggregation rule separately. Instead, BAEGA re-samples the training data with replacement

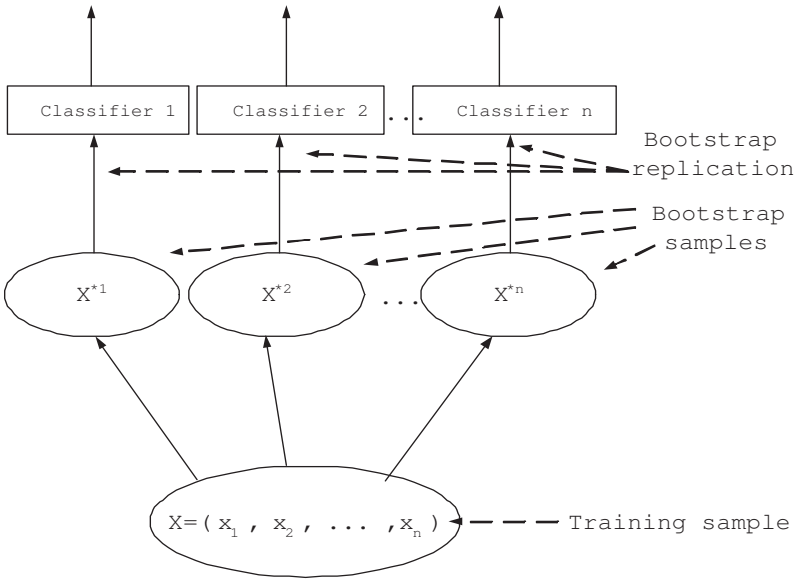

Figure 5.2: Train base classifiers with bootstrap samples [42]

and generates B set of bootstrap samples. Each set of the bootstrapped sample is used to train a base classifier, as illustrated in Fig. 5.2.

The probability that observation $\mathbf{x}_i$ is in the set of bootstrapped sample $\mathbf{X}^{*j}$ is calculated as follows [42]:

$$\begin{aligned} Pr[\mathbf{x}_i \in \mathbf{X}^{*j}] &= 1 - (1 - \frac{1}{N})^N \\ &\approx 1 - \frac{1}{e} \\ &= 0.632, \end{aligned}$$

where N is the size of the training data set. In other words, for any $\mathbf{X}^{*j}$, it is expected that there are $0.632N$ different observations in the bootstrapped sample to train each base classifier.

Training Aggregation Rule

Like GAEGA, BAEGA still adopts the AA approach for the decision aggregation. However, instead of using a separate data set to train the aggregation rule, all the training data are employed in BAEGA. The data flow of BAEGA is demonstrated in Fig. 5.3.

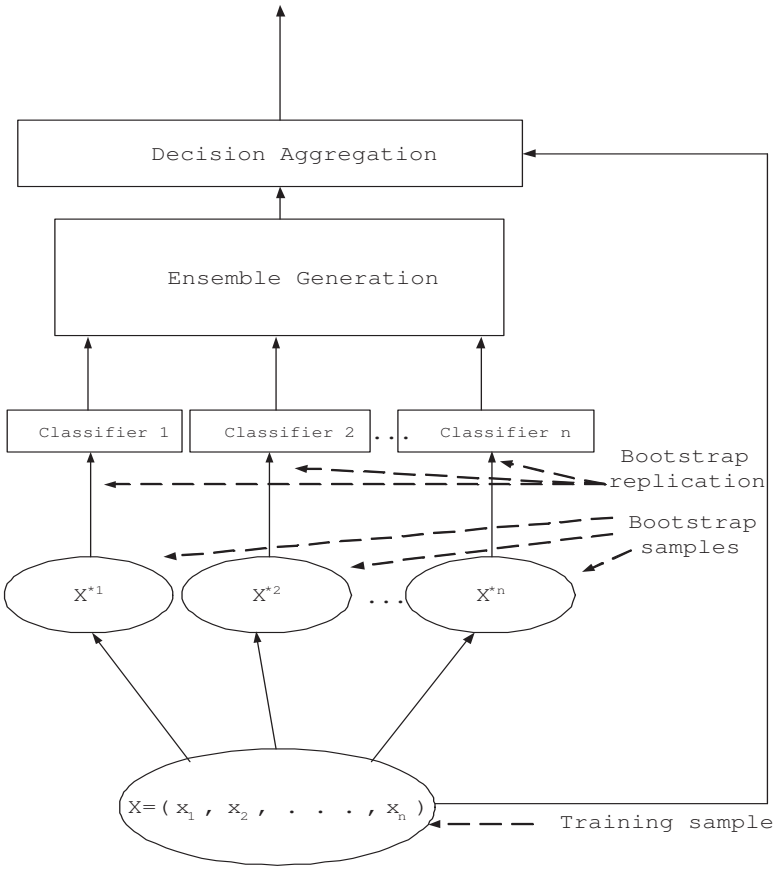

Figure 5.3: Data flow of BAEGA

The motivation to use the entire training data to train the aggregation rule is explained as follows. The previous discussion shows that only around $0.632N$ of the training data are used to train each base classifier. Obviously, the remaining data can be utilized for other training purposes, say, the aggregation rule. However, the AA approach requires the same set of validation data to evaluate the performance

of each generated ensemble. Since each bootstrap sample has a different set of remaining data, it is difficult to directly apply the remaining data to train the AA approach. The solution of BAEGA is to use the entire training data. Consequently, the goodness of fit of each generated ensemble is evaluated over the entire training data instead of over a separated validation data. A possible criticism for this approach is that the training data might be overfitted by BAEGA. It is true that BAEGA may use the same subset of the training data to train both the base classifier and aggregation rule, causing the overfitting problem. However, the AA approach is able to take care of this issue by dynamically tweaking the balance of the goodness of fit and the risk of overfitting. Therefore, it is expected that the performance of BAEGA will not be hurt by the overfitting problem. In Chapter 6, the experimental results prove the robustness of BAEGA.

5.3.2 Architecture of BAEGA

In this section, several architectures of BAEGA are examined. First, BAEGA is constructed with a set of homogeneous classifiers, each of which takes the raw input representation. Then, this architecture is extended to incorporate heterogeneous classifiers. Finally, a more general architecture in which BAEGA takes heterogeneous classifiers with multiple representations is presented.

BAEGA with Homogeneous Classifiers

Fig. 4.2 reflects the simplest architecture of BAEGA. There are three layers, consisting of the input processing layer, ensemble generation layer, and decision aggregation layer. In the input processing layer, a collection of Bootstrap Sample (BS) are generated by resampling the training data with replacement. Then, a set of homogeneous classifiers are applied on the bootstrapped samples, which generate a set of diverse decisions. The ensemble generation layer and decision aggregation layer are close to those of GAEGA.

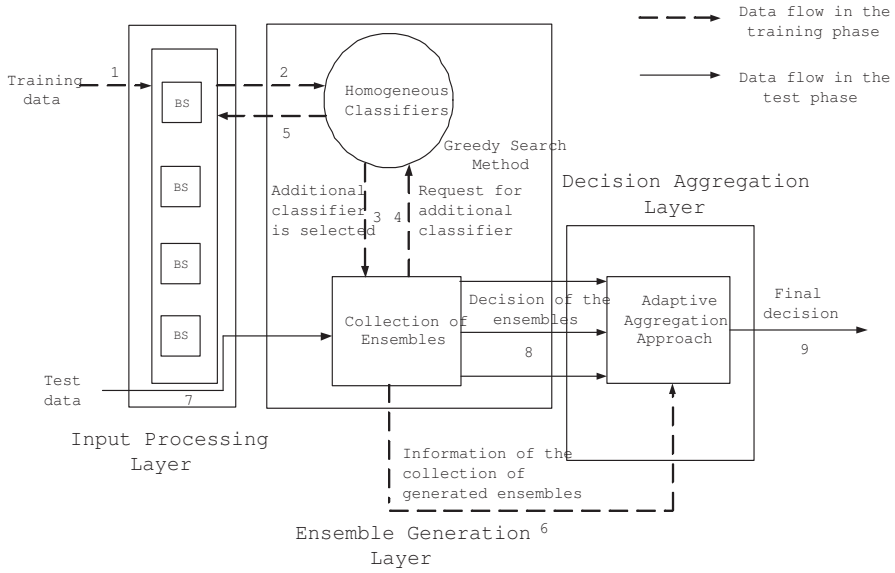

Figure 5.4: Architecture of BAEGA with homogeneous classifiers

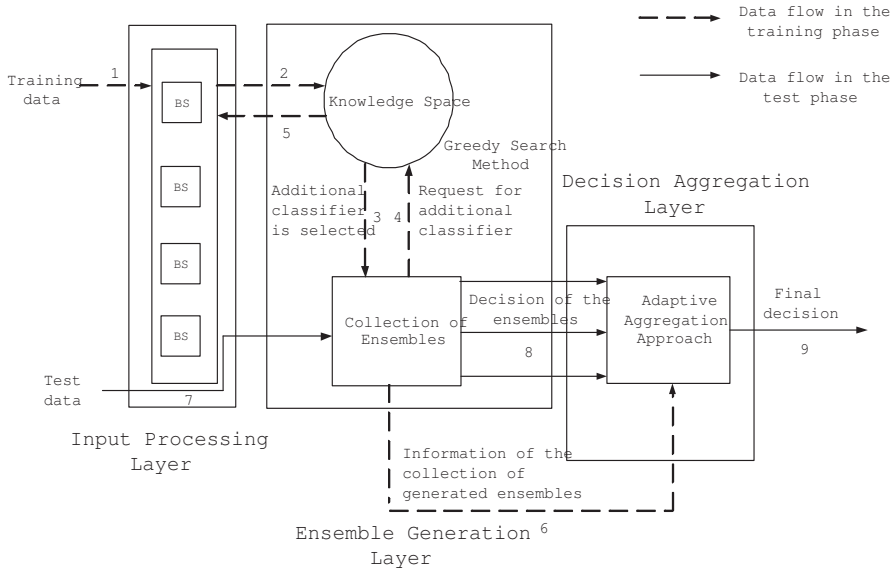

Figure 5.5: Architecture of BAEGA with heterogeneous classifiers

BAEGA with Heterogeneous Classifiers

The architecture in Fig. 5.5 is an extension of Fig. 5.4 where heterogeneous classifiers are employed in BAEGA. The definition of the knowledge space Ω is extended such that the classifiers are distinguished not only by the training algorithm and input representations, but also by the different bootstrap samples. In the knowledge space Ω, the classifiers are created by varying the learning algorithm, the initial conditions, and the parameters. The rules to combine the bootstrapped sample and the base classifiers are a little tricky. For example, the full combination increases the probability that the effective decisions are generated. However, the computation cost is also higher in this case. Therefore, the trade-off between the performance and the computation cost needs to be determined in terms of specific applications.

BAEGA with Multiple Representations

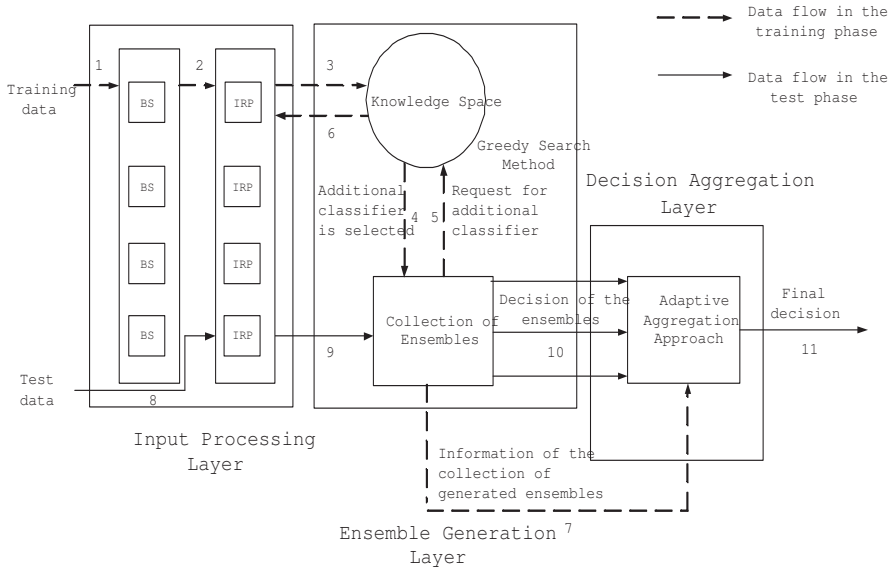

Figure 5.6: Architecture of BAEGA with multiple representations

Fig. 5.6 offers an even more general architecture, which allows BAEGA to

incorporate heterogeneous classifiers and multiple representations. Since this architecture exploits different strengths of the representations and classifiers to build the MCSs, it appears to be more powerful. Similarly, the rules to combine the bootstrapped sample and the input representations, and the input representations and the base classifiers need to be determined to balance the performance and computation cost.

5.4 Summary

In this section, BAEGA is presented as an alternative improvement of GESA. BAEGA incorporates the bootstrap method in the ensemble generation. In addition, BAEGA does not require a fixed split of the training data to train the base classifiers and the aggregation rules separately. Instead, it adopts a similar idea to that of the *out of bootstrap* method. BAEGA employs the bootstrap sample to train the base classifiers, and the entire training data set to train the aggregation rules. Several possible architectures of BAEGA are discussed to exploit the strength of the bootstrap method, as well as the heterogeneous classifiers and multiple input representations for the ensemble generation. In Chapter 6, the experiments are designed to compare the proposed approaches with the popular MCSs on the benchmark data sets.

Chapter 6

Experimental Results and Discussion

6.1 Introduction

In the previous chapters, the robust and data-adaptive designs for MCSs and the application of the proposed approach to classify time series data are presented. In this chapter, the performances of the proposed methods are empirically examined with the benchmark data sets. The experiments are organized as follows. Section 6.2 discusses the data sets which are used in the experiments of this thesis. In Section 6.3, the robustness of the proposed GESA is examined by varying the size of the training data set. It is expected that the performance of GESA should be reliable in various situations. Section 6.4 compares MIR-AEGA with other approaches on the benchmark time series data. In Section 6.5, the performances of GAEGA and BAEGA are tested on general types of data[1].

[1]All the classifiers used can be found in the classification toolbox version 2.0 of Matlab [17].

Table 6.1: Characteristics of time series data sets

Data Set	Source	Classes	Instances	Frames
CBF	[96]	3	600	200
Control Chart (CC)	[96]	6	600	60
Waveform (WF)	[8]	3	600	21
WF+Data Noise (WF+DN)	[8]	3	600	40
Two Patterns (TP)	[55]	4	5000	128
Gun Point (GP)	[55]	2	200	150
Trace (TR)	[55]	4	200	275
simu	Eq. 6.5	2	402	60

6.2 Data Description

The experiments are conducted on 2 synthetic and 15 benchmark data sets, which have been used by previous researchers and are available from the UCI repository or other related references [23, 55, 96]. According to the characteristics of the data, the data sets are categorized as time series data sets and non-time series data sets.

6.2.1 Time Series Data Sets

The characteristics of time series data sets are summarized in Table 6.1.

CBF The CBF data has been introduced by Saito [87] as an artificial problem. The learning task is to distinguish the data from the three classes, including cylinder(c), bell(b) and funnel(f). The models of the three classes are

$$
\begin{aligned}
c(t) &= (6+\eta)\cdot\chi_{[a,b]}(t)+\varepsilon(t)\\
b(t) &= (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\frac{t-a}{b-a}+\varepsilon(t)\\
f(t) &= (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\frac{b-t}{b-a}+\varepsilon(t).
\end{aligned}
$$

η and $\epsilon(t)$ are obtained from the standard normal distribution $N(0,1)$, a is an integer obtained from the uniform distribution in [16,32], c is another integer obtained from the uniform distribution in [32,96]. b is an integer that is the sum of a and c, and $\chi_{[a,b]} = 1$ if $t \in [a,b]$. Otherwise, $\chi_{[a,b]} = 0$.

Control Chart The CC data contains 600 examples of control charts which are synthetically generated. There are six different classes of control charts, including decreasing trend, cyclic, normal, upward shift, increasing trend, and downward shift.

Waveform Breiman et al. [8] have introduced the waveform data. The purpose is to identify three classes, defined by the evaluation in the time frames $t = 1, 2...21$, of the following models:

$$x_1(t) = uh_1(t) + (1-u)h_2(t) + \varepsilon(t)$$
$$x_2(t) = uh_1(t) + (1-u)h_3(t) + \varepsilon(t)$$
$$x_3(t) = uh_2(t) + (1-u)h_3(t) + \varepsilon(t),$$

where $h_1(t) = max(6 - |i-7|, 0)$, $h_2(t) = h_1(t-8)$, $h_3(t) = h_1(t-4)$. u is a random variable with a uniform distribution in (0,1), and $\varepsilon(t)$ is the standard normal distribution.

Waveform+Data Noise The Waveform+Data Noise (WF+DN) [31] is generated in the same way as the previous models, except that 19 frames are added to the end of each time series. The value of the 19 frames follows the standard normal distribution $N(0,1)$. This data set is used to test the performance of the proposed approach in the presence of data noise in the data set.

Tow Patterns The two pattern problem is designed by Geurts [35]. In this problem, each scenario is described by a single temporal input attribute A which is defined on the integer time axis 0, 1, ..., 127 by the following function:

$$A(o,t) = \begin{cases} \epsilon(t) & \text{if } 0 \leq t < t_1 \\ s_1(t-t_1, I_1) & \text{if } t_1 \leq t < t_1 + I_1 \\ \epsilon(t) & \text{if } t_1 + I_1 \leq t < t_2 \\ s_2(t-t_2, I_2) & \text{if } t_2 \leq t < t_2 + I_2 \\ \epsilon(t) & t_2 + I_2 \leq t < 128. \end{cases} \tag{6.1}$$

where time functions s_1 and s_2 are defining the patterns of respective lengths I_1 and I_2. $\epsilon(t)$ is some noise surrounding the patterns, drawn from a standard normal distribution N(0,1). I_1 and I_2 are integers uniformly drawn in [16,32], and t_1 and t_2 are integers randomly drawn in [0,127] such that the two patterns are not overlapping each other, and are fully restricted in the interval [0,127] (i.e., mathematically, such that $t_1 + I_1 < t_2$ and $t_2 + I_2 < 128$). As possible instances for s_1 and s_2, the simple upward and downward steps are defined by the following time function, where I is the length of the pattern:

$$us(t,l) = \begin{cases} -5 & \text{if } 0 \leq t < \frac{l}{2} \\ 5 & \text{if } \frac{l}{2} \leq t < l. \end{cases} \tag{6.2}$$

and

$$ds(t,l) = \begin{cases} 5 & \text{if } 0 \leq t < \frac{l}{2} \\ -5 & \text{if } \frac{l}{2} \leq t < l. \end{cases} \tag{6.3}$$

s_1 and s_2 are independently chosen as either us or ds (each one with the probability 0.5). According to this random choice, the classification of a scenario in one of the four classes is defined as follows

$$c(o) = \begin{cases} uu & \text{if } s_1 = s_2 = us \\ ud & \text{if } s_1 = us \text{ and } s_2 = ds \\ du & \text{if } s_1 = ds \text{ and } s_2 = us \\ dd & \text{if } s_1 = s_2 = ds. \end{cases} \tag{6.4}$$

As a result, the four classes are equiprobable.

Gun Point The gun point data set contains the time series data that are extracted from video of an actor, either aiming a gun or simply pointing to a garget [82]. The data records the movement of the actor's right hand.

Trace The trace data is the simulated data on the PWR 900 MW nuclear plant, corresponding to various occurrences of anomalous rapid load rejection events [77]. The data consists of a normal reference case and three anomalies, including component failures, control function failures, and sensor failures.

Simu The simu data is generated in this research with the underlying model as follows:

$$Class1: \quad y = 0.01sin(t) + 2k \tag{6.5}$$

$$Class2: \quad y = 0.01sin(10t) + 2k + 1,$$

where t =1,2,3,...60; and k = -100, -99,..., 0, ... 99,100. The purpose of the tests on the simu data is to illustrate certain critical issues related to methods such as nearest neighbor classifier and DTW.

6.2.2 Non-Time Series Data Sets

Clouds The clouds data [23] is selected to study the behavior for the heavy intersection of the class distribution for a high degree of nonlinearity of the class boundaries. The data exhibits bi-dimensional distribution. Class 0 is sum of three different Gaussian distributions with $P_{w_0} = 0.5$, and

$$p(x|w_0) = \frac{1}{2}(\frac{p_1(x)}{2} + \frac{p_2(x)}{2} + p_3(x))$$

with

$$p_j(x) = \frac{1}{2\pi}\frac{1}{\sigma_{jx}\sigma_{jy}}e^{-(\frac{(x-m_{jx})^2}{2\sigma_{jx}^2}+\frac{(y-m_{jy})^2}{2\sigma_{jy}^2})},$$

Table 6.2: Characteristics of non-time series data sets

Data Set	Source	Classes	Instances	Dimension
Clouds	[23]	2	987	2
WDBC	[96]	2	569	30
WPBC	[96]	2	198	32
Glass	[96]	7	214	10
Concentric	[23]	2	2500	2
Ionosphere	[96]	2	351	34
BCW	[96]	2	699	10
German	[96]	2	1000	24
XOR	Synthesized	2	500	2

where m_{jx} and m_{jy} are the x and y means of the j^{th} distribution, and σ_{jx} and σ_{jy} are x and y's standard deviations. Class 1 is a single normal distribution with $P_{w_1} = 0.5$ and

$$p(x|w_1) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}.$$

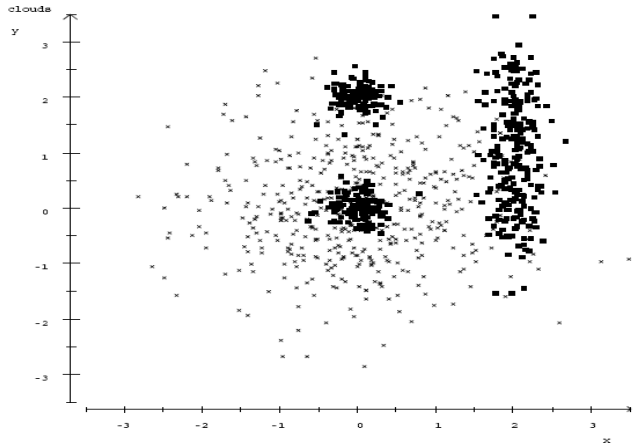

Figure 6.1: Example of clouds data [23]

WDBC The Wisconsin Diagnostic Breast Cancer (WDBC) data is used to identify malignant and benign symptoms. They contains ten real-valued features,

which are computed for each cell nucleus, including radius, texture, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension. The features are computed from a digitized image of a Fine Needle Aspirate (FNA) of a breast mass, and describe the characteristics of the cell nuclei in the image.

WPBC The Wisconsin Prognostic Breast Cancer (WPBC) data is similar to the WDBC data except that the WPBC data has two more features: tumor size and lymph node status. The data are used to predict recurrent and non-recurrent symptoms.

Glass The glass data is used to distinguish seven different glasses by the chemical elements such as aluminum, silicon, calcium, and iron. The class type includes building-windows-float-processed, building-windows-non-float-processed, vehicle-windows-float-processed, vehicle-windows-non-float-processed (none in the database), containers, tableware, and headlamps.

Concentric Concentric data is the simulated data, which has a bi-dimensional uniform concentric circular distribution [23]. The purpose is to study the linear separability of the classifier when some classes are nested in others without overlapping. The database is entirely contained in the square (0,0), (1,1). The points of class 0 are uniformity distributed into a circle of radius 0.3 centered on (0.5,0.5), and the points of class 1 are uniformly distributed into a ring centered on (0.5,0.5) with the internal and external radius equal to 0.3 and 0.5 respectively.

Ionoshpere The ionosphere is the radar data, collected by a system in Goose Bay, Labrador. The targets are free electrons in the ionosphere. *Good* radar returns are those showing evidence of some type of structure in the ionosphere, whereas *Bad* returns are those that do not. The received signals are processed by using an autocorrelation function whose parameters are the time of a pulse and pulse number. There are seventeen pulse numbers for the Goose Bay system. Instances in this database are described by two attributes per pulse number, corresponding to the complex values returned by the function that results from the complex electromagnetic signal [96].

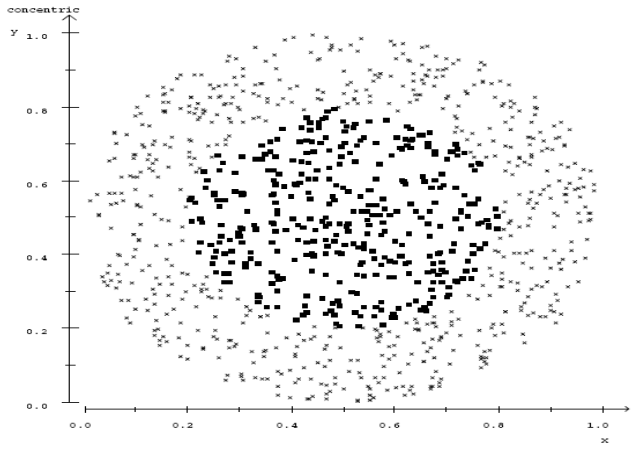

Figure 6.2: Example of concentric data [23]

BCW The Breast-Cancer-Wisconsin (BCW) data is the medical data which are used to recognize benign and malignant symptoms. The data attributes include clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses.

German This data set classifies people who are described by a set of attributes as good or bad credit risks. The original data set contains both qualitative and numeric attributes, including the status of existing checking accounts, duration in months, and credit history. In this thesis, the data set that is used contains only the numeric attributes.

XOR The data distribution of XOR data is depicted in Fig. 6.3. The boundaries for these two categories are *aod* and *boc*. Two linear classifiers, *XLinear* and *Ylinear*, are introduced for this problem. The decision rule for *XLinear* is simple. It projects the data to the X axis and finds the split point along the X axis by minimizing the overall error rate. The decision rule for *Ylinear* is similar. If there is a tie among all the points, the cut point is set to 0.5. For instance, in Fig. 6.3, the decision boundary of *XLinear* and *Ylinear* are *aob* and *cod*, respectively. This problem is denoted as decision XOR problem and used to test the performance of various MCSs.

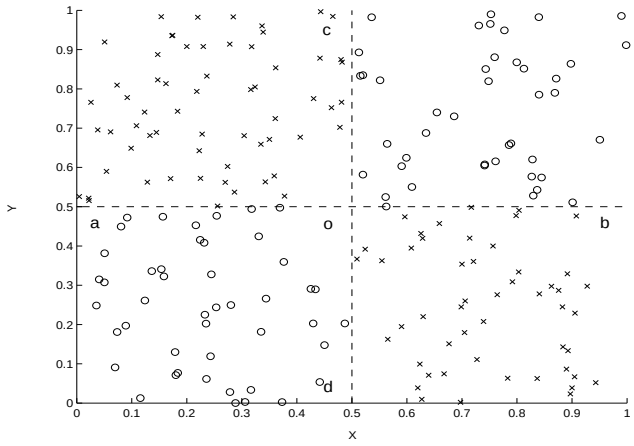

Figure 6.3: Decision XOR problem

6.3 Experiments on GESA

In Chapter 3, GESA is proposed as a robust design of MCSs by dynamically tweaking the trade-off between the goodness of fit and the risk of overfitting. The purpose of this test is to empirically examine the performance of GESA. In the experiment, GESA and the benchmark methods are trained on the data set with the different sizes. If GESA can not control the overfitting, its performance is expected to be significantly worse than that of other methods, when the size of the training data set is small.

6.3.1 Experimental Setup

There are two well-known ways to generate an ensemble: vary the input data and vary the training algorithm and its parameters. The performances of GESA are examined in both situations. Five data sets are adopted for this test, including CBF, clouds, WPBC, WDBC, and glass. Among them, CBF is used for the test in the first case. A set of four homogeneous Probabilistic Neural Networks (PNNs) [91] are employed as the base classifiers. The diverse decisions are generated by varying the representation of the input data and the parameters of the representation methods. One of the base classifiers takes the raw data as the input, and

another takes the coefficient of the Discrete Fourier Transformation (DFT) as the input. The remaining classifiers take the data which is processed by the Double Exponential Smoothing and Differentiation Generation (DES+DG) with different parameters [14]. The remaining data sets, including clouds, WPBC, WDBC, and glass are chosen for the test purpose in the second case. A set of heterogeneous classifiers are employed, including PNN, K-Nearest Neighbor (KNN), Decision Tree, and Perceptron. The parameter setting for each classifier is as follows. For PNN, the Gaussian width is set to 0.1. For KNN, k is set to 1. For the decision tree, the CART (Classification And Regression Tree) model is used, in which the evaluation criteria is set as 'Entropy' and the incorrect ratio is set to 0.1. For Perceptron, the Perceptron-BVI model is chosen. The number of iterations is set to 500 and the convergence rate is set to 0.1. There is no specific reason to set the ensemble size to four. It is understood that the ensemble size can significantly affect the outcome of the methods. However, in this experiment, the focus is on the identified factors, leaving the issue of ensemble size for future research.

In this test, DT, BKS, NBF, and MV are selected as the baseline of comparison, since they have similar properties to those of GESA. For example, all can take multiple representations or heterogenous ensembles. The size of the whole training data set is varied between 45% and 80%. For GESA, DT, BKS, and NBF, since they require that the training data is split to train the base classifiers and the aggregation rule separately, 40% of the data are allocated to train the base classifiers and the rest of the training data are used to train the aggregation rule; that is, the size of the validation data ranges from 5% to 40% of the entire data set. The choice of data partitioning is completely random and class distribution is not considered.

6.3.2 Experimental Results and Discussion

Table 6.3 and 6.4 report the test results from 10 rounds of independent tests which are in the format of the average accuracy and its standard error. MV is one extreme example. It seems that MV does not incur the overfitting problem easily. The evidence is that MV is relatively insensitive to the size of the training data. The average accuracy does not differ much when the size of the training data ranges

Table 6.3: Accuracy (in %)of GESA and other benchmark methods (1)

Data	Size	DT	NBF	BKS	MV	GESA
CBF	45%	90.9±7.60	88.1±9.81	74.6±7.20	88.4±4.23	86.6±6.94
	50%	94.9±2.06	87.6±13.10	82.0±3.75	87.5±2.42	94.0±1.91
	55%	95.1±1.93	94.7±2.34	85.2±4.08	87.0±2.33	93.1±4.23
	60%	94.3±1.94	94.5±2.43	89.1±2.03	86.1±5.66	94.9±2.03
	65%	95.1±2.34	94.7±2.26	90.1±2.22	89.2±4.98	94.9±2.22
	70%	94.8±1.56	95.2±1.62	91.4±2.17	91.9±1.91	94.6±2.17
	75%	94.4±2.06	94.4±2.27	92.1±2.21	90.0±2.79	94.7±2.21
	80%	94.0±2.86	94.5±3.15	92.6±1.69	90.8±2.64	95.8±1.69
Glass	45%	85.3±6.73	82.5±8.31	69.3±10.39	76.1±13.03	86.0±11.6
	50%	90.2±5.36	87.3±7.92	78.7±8.11	73.0±9.31	87.7±10.8
	55%	91.3±4.69	88.3±4.57	82.0±5.03	76.0±6.45	91.6±5.07
	60%	93.1±4.72	90.0±6.71	84.4±4.77	79.1±7.29	94.0±4.50
	65%	94.1±4.78	90.7±4.99	87.8±3.93	77.5±8.91	94.5±4.14
	70%	95.8±3.13	90.2±4.83	89.5±4.17	78.4±5.23	94.6±4.66
	75%	95.7±2.82	90.6±5.63	88.9±5.07	77.9±4.51	94.6±5.23
	80%	96.4±3.59	90.2±4.95	89.8±4.50	78.8±1.03	94.2±4.88
Clouds	45%	85.3±1.53	85.2±1.08	85.3±2.21	82.3±1.96	86.6±1.34
	50%	85.2±1.47	85.2±1.47	86.0±1.21	82.6±2.62	86.1±1.05
	55%	85.1±1.50	85.2±1.36	86.0±1.42	78.9±6.99	86.4±0.89
	60%	85.2±1.35	85.1±1.29	86.3±1.48	80.3±4.64	85.9±0.68
	65%	85.1±1.44	85.1±1.44	86.4±1.53	81.8±1.88	85.8±0.64
	70%	84.8±1.34	84.7±1.42	86.1±1.50	82.7±2.43	85.6±0.70
	75%	85.0±1.36	85.1±1.26	86.4±1.96	81.1±2.77	86.2±1.29
	80%	85.4±1.50	85.5±1.36	86.7±1.76	83.3±2.76	86.1±1.85

Table 6.4: Accuracy (in %)of GESA and other benchmark methods (2)

Data	Size	DT	NBF	BKS	MV	GESA
	45%	70.2±17.3	79.1±15.0	77.5±14.7	74.7±2.80	86.2±3.05
	50%	74.3±12.6	84.0±3.74	82.9±4.11	71.3±3.11	85.6±3.56
	55%	76.3±11.5	84.4±4.34	83.2±4.98	71.9±3.31	85.8±4.79
WPBC	60%	72.4±15.3	84.0±5.24	84.0±5.24	72.4±1.06	85.7±5.31
	65%	72.9±16.1	85.3±4.64	85.3±4.64	71.9±4.23	85.8±4.83
	70%	73.3±17.7	85.2±5.27	85.2±5.27	76.6±2.84	86.5±4.81
	75%	73.2±17.2	83.6±5.56	83.6±5.56	73.9±3.35	85.6±5.95
	80%	72.5±16.7	83.0±6.77	83.0±6.77	77.0±4.87	85.3±7.12
	45%	92.1±1.07	82.2±11.56	92.2±0.99	90.4±0.55	89.5±4.86
	50%	92.1±1.16	82.5±9.45	92.2±1.10	92.0±1.38	88.1±6.04
	55%	92.3±1.52	81.6±9.23	92.4±1.50	91.0±1.35	90.8±1.54
WDBC	60%	92.3±1.95	82.4±9.67	92.4±1.86	91.8±1.71	89.7±5.75
	65%	92.2±1.63	82.2±9.94	92.2±1.56	92.0±1.70	89.5±5.78
	70%	91.9±1.59	79.1±9.37	92.1±1.67	93.6±2.45	90.8±3.06
	75%	91.9±2.00	78.5±8.91	92.1±2.12	92.1±1.04	91.1±3.50
	80%	92.8±1.89	81.7±9.28	92.8±1.89	91.6±2.88	91.3±3.75

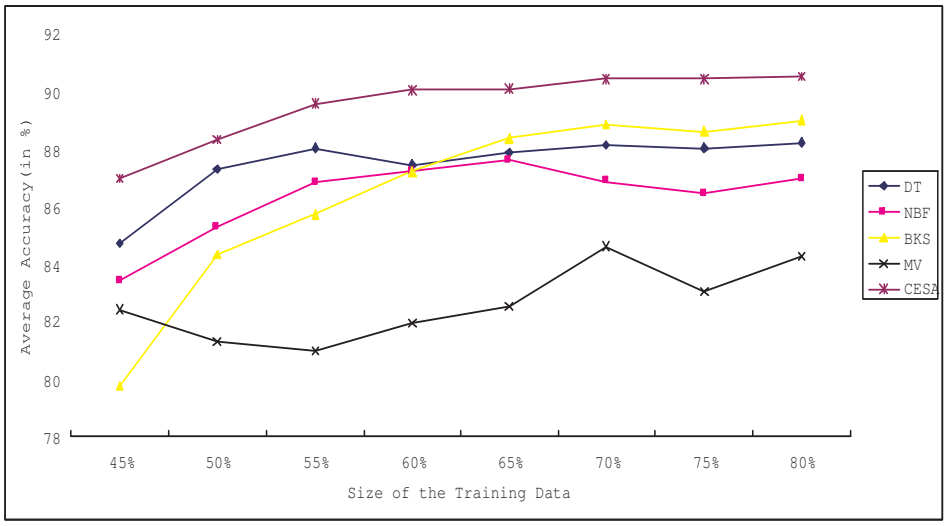

Figure 6.4: Average accuracy of GESA and other benchmark methods

from 40% to 80% of the whole data set. However, MV is not flexible enough to fit the training data. For the glass and WPBC data, MV exhibits a relatively inferior performance due to the inability to learn from the data adequately. In contrast, BKS is another extreme example. It is more flexible to fit the training data, more specifically the validation data. Given that the size of the training data is 80%, the performance of BKS is good for different problems on average. However, the overfitting problem readily occurs. For the CBF and glass data, it is observed that the difference of the average accuracy is as high as 20%, when the sizes are 40% and 80% of the whole data set respectively. It is evident that both MV and BKS are not robust designs of MCSs. DT and NBF are methods which exist between those two extremes as seen in Fig. 6.5. Compared to that of BKS, their performances are not that significantly affected by the size of the training data, which indicates that DT and NBF do not get into the overfitting problem easily. Compared with MV, they are more flexible to fit the data. The evidence suggests that the data are less difficult for them. However, DT and NBF are still not ideal designs of MCSs. Compared with other methods, DT and NBF have difficulty in the WPBC and WDBC data, respectively. GESA is a better design of MCSs. Its performance is robust over different sizes, as well as over different problems. Fig. 6.4 summarizes

the average performances of all the approaches over various data sets. Although GESA might not always be the best for a specific problem, GESA outperforms other methods for different sizes of the training data in terms of the average accuracy.

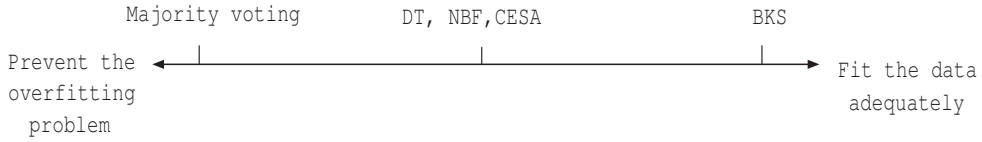

Figure 6.5: Relation between different approaches

6.4 Experiments on the Classification of Time Series Data

6.4.1 Experimental Setup

The purpose of this experiment is to test the performance of MIR-AEGA. Eight time series data sets are chosen for the test purpose, including CBF, CC, WF, WF+DN, GP, TR, TP, and simu. The methods, including adaboost, bagging, RS, DT, NBF , and BKS as well as the individual classifiers are employed as the baseline for the comparison.

Theoretically, MIR-AEGA can work with any base classifier. In this experiment, KNN, DTW, and PNN are selected for the demonstration purpose. For the classification of time series data, PNN gives a relatively weak performance. Therefore, PNN is used as the example of weak classifier. Keogh et al. [55] have reported a relatively good performance for KNN (K=1) and DWT. However, the study in this thesis shows that these techniques might not be robust and can fail in certain situations (e.g., simu data). The purpose is to show that an improved and more reliable performance can be achieved with MIR-AEGA by appropriately selecting the input representations, and aggregating the decisions of the weak classifiers.

The parameter settings for each classifier in this experiment is as follows. For PNN, the Gaussian width is set to 0.1. For KNN, k is 1. In all of the experiments,

the data sets are randomly separated into 60% training and 40% testing. For DT, NBF, BKS, and MIR-AEGA methods, 66.7% of the training data are assigned to train the base classifiers, and the remaining 33.3% are used as a validation set to train the aggregation rule. The representation techniques for the input representation layer are the Raw data (Raw), DFT, Discrete Wavelet Transformation (DWT), DES+DG, Random Feature Selection (RFS), ARMA, PLA, PAA, KL Transformation, LPC, Random Projection (RP), Remove Linear Trend of data (RLT), and Remove Offset Translation (ROT) [14, 43]. In addition, the parameters of these methods are varied to generate 20 different representations in total. PNN and KNN take each of the representations, whereas DTW takes only the raw data as the input. It should be emphasized here that the representation techniques are not limited to the ones previously listed. In this experiment, they are employed only as examples to test the idea of MIR-AEGA. To make a fair comparison with the benchmark classification system, including adaboost and individual classifiers, the data is processed by the various representation methods just discussed, and the processed data are applied to those methods. Due to the limitation of the length of the thesis, only the relatively significant results are reported.

The performances of MIR-AEGA are examined with both the heterogeneous ensemble and homogeneous ensemble. The test results are reported in Table 6.5 - 6.8, which summarize the average accuracy and its standard error (in %) for the different approaches from the 10 rounds of independent tests. The methods' name is signified, as well as the base classifier and the representation of the data for that approach in the form: method(representation, base classifier). For example, Ind(Raw,PNN) denotes the individual PNN classifier which takes the raw data and AdaBoost(DFT,PNN) to denote the adaboost which takes the DFT coefficient as the input and PNN as the base classifier. The results of four versions of MIR-AEGA are reported. MIR-AEGA(All) denotes that MIR-AEGA takes KNN, PNN and DTW as base classifiers, and MIR-AEGA(PNN,KNN) takes PNN and KNN as base classifiers. The reason is that DTW is very time-consuming compared to KNN and PNN. So, the intension is to compare the performance of MIR-AEGA(All) and MIR-AEGA(PNN,KNN) to see the trade-off between the performance and time complexity. In addition, the intension is to examine whether the perfor-

mance of MIR-AEGA heavily depends on the choice of the classifier. Both MIR-AEGA(PNN,100) and MIR-AEGA(PNN,10000) denote MIR-AEGA which takes a set of homogeneous PNN as base classifiers. 100 and 10000 are the value of N_g when $G(p_i^m, \hat{p}_i^m, n_i)$ is estimated in the score function (N_g is set to 10000 in the other tests by default). When N_g is larger, it is anticipated that the estimation is more accurate, but the computation cost is also higher. In this test, the intension is also to examine this conclusion empirically.

6.4.2 Experimental Results and Discussion

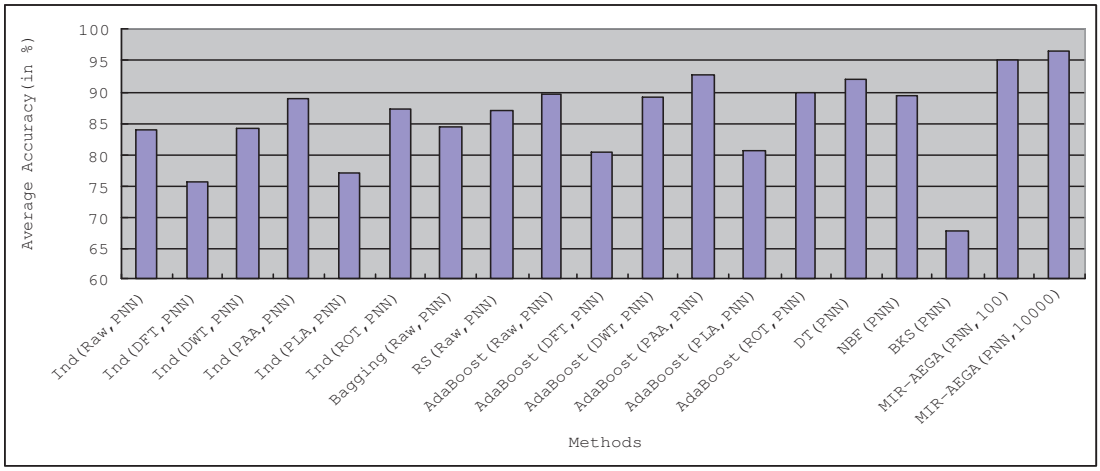

Figure 6.6: Average accuracy of the methods over different data sets excluding simu (1)

Table 6.5 - 6.6 list the results of the MCSs with the homogeneous ensembles. PNN is employed as the base classifier, since it is relatively weak for time series data. The purpose is to examine the effects of the input representations. The experimental results relate that the selection of the input representation is crucial for the classification performance. Take Ind(ARMA,PNN) as an example, ARMA is not good for representing data such as CBF, CC, WF, and WF+DN. The possible

[2]In order to make Adaboost work for this case, the number of iterations is set to 1

Table 6.5: Average test accuracy (in %) for MIR-AEGA and other methods (1)

Methods	Data Sets			
	CBF	CC	WF	WF+DN
Ind(Raw,PNN)	72.5±3.15	81.0±4.14	93.4±0.91	90.2±1.71
Ind(DFT,PNN)	84.2±1.92	74.2±3.96	86.4±1.38	83.9±2.23
Ind(DWT,PNN)	71.7±2.21	80.6±3.77	93.5±0.91	91.1±1.32
Ind(PAA,PNN)	99.5±0.39	82.8±5.42	94.0±0.91	92.6±1.69
Ind(ARMA,PNN)	38.0±3.26	49.4±2.38	47.1±1.91	54.9±2.92
Ind(PLA,PNN)	**100.0±0.00**	76.7±5.28	98.7±0.16	**99.6±0.07**
Ind(LPC,PNN)	41.7±4.79	38.4±1.92	63.8±2.79	69.1±2.22
Ind(RP,PNN)	61.8±9.59	59.0±7.13	82.3±3.97	79.6±4.83
Ind(ROT,PNN)	84.6±1.69	91.4±2.17	93.4±1.32	90.5±1.36
Bagging(Raw,PNN)	72.3±2.74	80.8±4.02	94.5±0.91	90.8±1.37
RS(Raw,PNN)	80.5±3.37	82.2±2.99	96.7±0.98	93.1±1.38
AdaBoost(Raw,PNN)	92.8±2.16	86.1±8.65	93.0±0.93	91.6±1.35
AdaBoost(DFT,PNN)	93.1±0.99	75.1±5.95	84.1±3.21	86.8±2.77
AdaBoost(DWT,PNN)	90.3±1.72	90.1±1.93	92.5±1.33	92.0±1.34
AdaBoost(PAA,PNN)	99.7±0.94	91.8±5.41	98.6±0.42	98.0±0.96
AdaBoost(ARMA,PNN)	35.0±3.55	48.2±1.94	42.5±2.44	52.3±2.78
AdaBoost(LPC,PNN)	46.5±4.07	32.3±2.97	61.3±3.52	70.6±2.15
AdaBoost(PLA,PNN)	99.9±0.10	87.6±4.62	98.2±0.93	99.0±0.99
AdaBoost(RP,PNN)	51.3±6.16	57.4±6.73	75.0±5.22	75.0±4.98
AdaBoost(ROT,PNN)	90.6±1.97	93.7±1.35	92.3±1.33	90.3±1.66
DT(PNN)	98.5±0.77	91.6±1.68	96.5±1.39	93.9±0.90
NBF(PNN)	97.1±2.39	89.0±3.54	96.3±0.97	94.3±0.91
BKS(PNN)	72.8±4.74	60.7±4.94	73.3±4.73	70.8±6.12
MIR-AEGA(PNN,100)	99.8±0.20	90.1±3.09	**99.0±0.38**	99.1±0.44
MIR-AEGA(PNN,10000)	99.3±0.56	**97.4±1.73**	98.8±0.66	99.4±0.63

Table 6.6: Average test accuracy (in %) for MIR-AEGA and other methods (2)

Methods	Data Sets			
	TP	GP	TR	simu
Ind(Raw,PNN)	94.0±0.52	86.4±6.86	70.0±1.95	44.3±1.34
Ind(DFT,PNN)	51.5±2.44	82.4±6.49	67.0±2.18	47.5±1.35
Ind(DWT,PNN)	94.2±0.90	86.5±5.14	70.9±3.78	45.0±3.55
Ind(PAA,PNN)	**97.1±3.39**	86.4±5.93	69.6±5.57	45.5±1.93
Ind(ARMA,PNN)	45.2±3.07	65.8±4.75	77.5±3.23	**100.0±0.00**
Ind(PLA,PNN)	47.8±3.84	62.5±15.0	53.9±9.38	40.5±2.88
Ind(LPC,PNN)	45.4±3.72	50.4±6.10	72.8±7.48	**100.0±0.00**
Ind(RP,PNN)	35.8±2.96	85.0±6.39	65.9±5.59	47.0±2.59
Ind(ROT,PNN)	94.3±0.68	86.9±4.16	70.6±5.81	99.8±0.23
Bagging(Raw,PNN)	94.1±0.99	88.1±6.45	70.5±5.92	45.3±2.37
RS(Raw,PNN)	95.4±0.77	88.1±4.98	72.5±4.04	44.5±1.94
AdaBoost(Raw,PNN)	88.7±0.97	94.3±2.20	80.3±4.74	44.5±2.17
AdaBoost(DFT,PNN)	46.4±0.98	**96.3±1.32**	81.1±3.71	47.0±2.15
AdaBoost(DWT,PNN)	89.0±0.58	94.5±2.76	76.1±7.73	44.6±2.95
AdaBoost(PAA,PNN)	92.9±0.35	94.0±2.61	73.8±10.1	39.3±8.41
AdaBoost(ARMA,PNN)	50.7±2.73	60.4±2.55	80.1±12.9	**100.0±0.00**[2]
AdaBoost(PLA,PNN)	49.9±3.67	84.9±3.92	69.4±3.92	32.5±4.15
AdaBoost(LPC,PNN)	48.2±3.45	47.9±8.07	84.0±5.15	**100.0±0.00**[2]
AdaBoost(RP,PNN)	36.2±3.27	85.6±5.16	74.3±2.72	42.5±9.23
AdaBoost(ROT,PNN)	89.1±0.95	93.9±2.16	80.1±4.40	63.3±4.86
DT(PNN)	90.1±0.87	85.3±6.24	87.1±6.50	94.3±12.70
NBF(PNN)	90.2±0.69	85.3±6.39	73.9±10.1	95.2±11.06
BKS(PNN)	64.0±3.98	64.4±9.30	69.1±8.71	98.8±2.14
MIR-AEGA(PNN,100)	94.3±0.98	85.8±6.91	97.1±2.19	**100.0±0.00**
MIR-AEGA(PNN,10000)	93.9±0.22	88.0±6.03	**98.5±0.56**	**100.0±0.00**

Table 6.7: Average test accuracy (in %) for MIR-AEGA and other methods (3)

Methods	Data Sets			
	CBF	CC	WF	WF+DN
Ind(Raw,KNN)	97.6±0.71	98.2±0.55	98.0±0.77	98.7±0.83
Ind(DFT,KNN)	90.6±1.94	94.1±1.33	96.5±0.77	96.6±0.54
Ind(PAA,KNN)	**100.0±0.00**	97.4±0.38	99.2±0.70	99.3±0.62
Ind(Raw,DTW)	**100.0±0.00**	99.3±0.38	98.1±0.43	98.7±0.97
Bagging(Raw,KNN)	98.6±0.92	98.0±0.25	98.3±0.57	98.6±0.17
RS(Raw,KNN)	99.1±0.67	97.8±0.62	99.8±0.27	**99.6±0.02**
AdaBoost(Raw,KNN)	64.6±4.04	60.4±9.51	62.8±3.39	76.7±5.97
DT(All)	99.9±0.17	99.0±0.21	**100.0±0.00**	99.6±0.39
NBF(All)	**100.0±0.00**	99.1±0.21	99.7±0.20	99.4±0.69
BKS(All)	70.6±2.78	68.9±2.39	81.7±2.78	75.9±1.92
MIR-AEGA(All)	**100.0±0.00**	**99.5±0.23**	99.2±0.29	99.0±0.47
MIR-AEGA(PNN,KNN)	**100.0±0.00**	98.4±1.03	98.6±0.53	98.8±0.79

Table 6.8: Average test accuracy (in %) for MIR-AEGA and other methods (4)

Methods	Data Sets			
	TP	GP	TR	simu
Ind(Raw,KNN)	93.6±0.84	91.9±1.92	83.3±4.84	12.3±1.39
Ind(DFT,KNN)	51.0±0.99	95.9±1.69	86.3±3.25	59.9±4.22
Ind(PAA,KNN)	97.0±0.42	94.3±2.14	83.4±2.44	15.4±2.56
Ind(Raw,DTW)	99.0±0.54	92.2±2.59	**99.9±0.05**	16.1±0.92
Bagging(Raw,KNN)	89.9±0.93	93.3±3.58	83.8±3.50	16.3±4.15
RS(Raw,KNN)	91.6±0.87	95.3±1.66	85.3±4.09	12.8±3.69
AdaBoost(Raw,KNN)	87.1±1.34	91.7±0.63	80.8±3.53	17.3±1.37
DT(All)	92.4±0.27	95.6±2.74	89.6±6.15	95.3±3.44
NBF(All)	93.5±0.35	93.3±2.42	78.5±9.68	83.9±2.60
BKS(All)	79.8±0.99	68.2±6.60	71.2±6.52	91.2±5.14
MIR-AEGA(All)	**99.5±0.44**	**96.5±0.94**	99.7±0.76	**100.0±0.00**
MIR-AEGA(PNN,KNN)	96.6±0.90	96.4±1.69	97.3±1.31	**100.0±0.00**

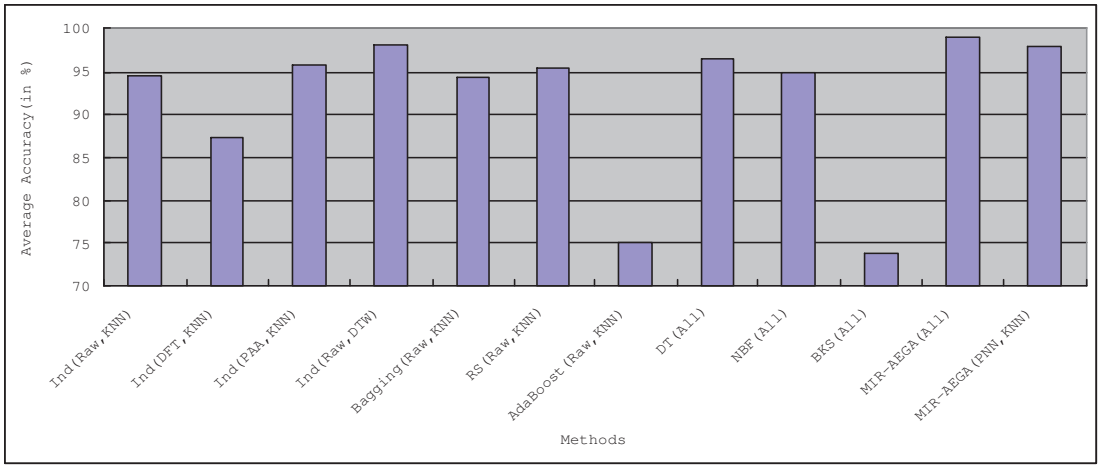

Figure 6.7: Average accuracy of the methods over different data sets excluding simu (2)

reason is that the underlying model of these data sets is neither an autoregressive nor a moving average process. In contrast, PAA representation is very effective for these data sets. However, from the experimental results, no one representation method is consistently dominant. A representation method such as PAA can still have difficulties with data sets such as TR.

Obviously, without prior knowledge, it is difficult to determine the appropriate representation for the given time series data in advance. In addition, the ensemble approaches such as adaboost, bagging, and RS can not help much, since the important information in the data may is already distorted or removed by the inappropriate representations. The experimental results indicate that such methods themselves also strongly depend on the representation of the data. For example, AdaBoost(PAA,PNN) is significantly better than AdaBoost(ARMA,PNN). Although those methods can take *any* input representation, they still face similar problems such as how to dynamically determine the appropriate representations for the given time series data. A straightforward solution is to represent time series data in multiple ways, and dynamically aggregate the decisions of the base classifiers which take different representations. Several MCSs, including DT, NBF and MIR-AEGA

adopt this strategy and achieve relatively good performances on average. In particular, MIR-AEGA gives consideration into the robustness and data-daptiveness in the design of MCSs. Therefore, its performance is on average superior to other MCSs such as DT, NBF, as well as other benchmark methods over different data sets as denoted in Fig. 6.6[3].

Table 6.7 - 6.8 report the results of the MCSs with the heterogeneous ensembles. This test is designed to examine the edge of MIR-AEGA over the claimed *strong* classifiers for time series data such as KNN (K=1) and DTW [55]. The experimental results confirm that DTW is effective for many data sets such as CBF, CC, WF, and WF+DN. However, by exploiting the advantage of multiple representations and heterogeneous classifiers, MIR-AEGA can still significantly outperform DTW in the data set such as GP. Under the significance level $\alpha = 5\%$, the results of the t-test show that both MIR-AEGA(PNN,KNN) and MIR-AEGA(All) outperform other methods significantly. Although MIR-AEGA(All) is not consistently the best one (e.g., on the data sets such as WF, WF+DN, and TR), its performance is very close to the best one. Also the t-test shows that the difference is not significant in most of the cases. The test results demonstrate that MIR-AEGA(All) has better and more reliable performances on average as recorded in Fig. 6.7[3].

In addition, other MCSs, which also have the advantage of multiple input representations and aggregate the decisions of the different classifiers, can achieve good performances on average. Of course, the performance of these MCSs strongly depends on the aggregation strategy. For example, the performance of BKS(All) is not acceptable. In contrast, the ensemble approaches such as RS, bagging, and adaboost, do not use the temporal information in the time series data very effectively. Although these approaches can take *any* input representation and classifier, they can not take *all* of them simultaneously. Therefore, boosting KNN and PNN does not improve the performance significantly nor overcome their difficulty with the simu data. Finally, there is the significant overfitting problem for adaboost, when KNN is chosen as the base classifier for CBF, CC, WF, and WF+DN.

The simu data provides the opportunity to test the robustness of the distance-

[3]Since MIR-AEGA has the advantage in the simu data, which is simulated in this research, it is excluded, when computing the average accuracy, to avoid favoring MIR-AEGA inappropriately.

based classifiers such as KNN and DTW. The experimental results show that KNN and DTW completely fail on this test. This test raises some questions on classifying time series data with KNN and DTW. First, they differ from each other in the similarity measure. KNN adopts the Euclidian distance measure and DTW adopts the edit distance measure. There is an analogy between the effects of the similarity measure and the input representations. The experimental results show that no input representation can be consistently dominant. Similarly, it is not convincing to claim that either Euclidian distance or DTW is the best similarity measure. Although this thesis does not investigate this topic, it is still safe to conclude that individual similarity measure might not be reliable for time series data given its extremely diversified sources. The simu data is an example in such cases. In addition, both DTW and KNN adopt the same learning algorithm, which labels the test data according to the nearest neighbor in the training data. As discussed in Chapter 1, this approach is very local and can easily leads to the overfitting problems, in particular for time series data which are high dimensional.

The design of a MCS requires the determination of the trade-off between the performance and computation cost. Also, there are two such design considerations for MIR-AEGA. One of them is the value of N_g. The larger the value of N_g, the more accurate estimation, improving the classification accuracy. However, the computation cost increases. This point is empirically examined by comparing the performances of MIR-AEGA(PNN,100) and MIR-AEGA(PNN,10000). From the experimental results, MIR-AEGA(PNN,10000) significantly outperforms MIR-AEGA(PNN,100) in CC and GP. As expected, MIR-AEGA(PNN,10000) consumes a much longer time than MIR-AEGA(PNN,100) in the experiments. Another design consideration is components of the knowledge space Ω. For example, DTW is very time-consuming, compared with KNN and PNN although DTW has a higher accuracy. In the experiments, MIR-AEGA(PNN,KNN) is found to be much faster on average than MIR-AEGA(All). Therefore, it has advantages for some time-critical applications. On the other hand, the average accuracy of MIR-AEGA(All) is higher than that of MIR-AEGA(PNN,KNN) over all the test data sets(excluding the simu data), which are 99.0% and 98.0%, respectively. The t-test shows that the difference is not significant under the significance level $\alpha = 5\%$, indicating that

the performance of MIR-AEGA does not heavily depends on DTW. Typically, it is expected that MIR-AEGA has better performances, when there are more representation methods and base classifiers for selection. Accordingly, the computation cost is much higher. These tests demonstrate that MIR-AEGA is flexible, and the users can work out a balance between the accuracy and computation cost for specific problems.

Finally, MIR-AEGA provides some insight into the data; that is, MIR-AEGA does not classify the data in a black-box style. For example, DFT or DWT seem to be the best representations for the simu data, when the KNN classifier is used. However, it turns out that MIR-AEGA actually selects the representations ROT and LPC that give a more superior performance with the KNN classifier. Therefore, besides good and reliable performances, MIR-AEGE automatically identifies the effective input representations and classifiers for the given time series data. This helps in understanding the internal structure of the data better.

6.5 Experiments on the Classification of General Data Type

6.5.1 Experimental Setup

The purpose of this experiment is to test the performances of GAEGA and BAEGA. Several popular methods are selected as the baseline of the comparison including MV, adaboost, RS, DT, NBF, and BKS. For GAEGA/BAEGA, both the addition rule and multiplication rule are used to build the score function, denoted by GAEGA(addition)/BAEGA(addition) and GAEGA(multi)/BAEGA(multi). For GAEGA, k is set to 1. For BAEGA, k is set to 2. There are two well known ways to generate the classifier ensembles: vary the representation of the input data, and vary the learning algorithm and the model parameters. In this experiment, the performances of GAEGA and BAEGA are tested for both of cases.

All the time series data (excluding the simu data set) are employed to test the performances of different approaches in the first case. This provides more options

to vary the input representation of the data. According to the empirical study of Keogh et al. [55], the KNN(K=1) classifier is adopted as the base classifier for speed and accuracy. There are 20 representation techniques which are employed by varying the representation methods and their parameters. The settings are the same as those in the previous experiment. Adaboost, RS, and bagging take the raw data as the input.

The data sets, including the clouds, WDBC, WPBC, phoneme, concentric, inonosphere, BCW, german and XOR, are adopted for the test purpose in the second case. A set of heterogeneous ensembles are employed. They are PNN, KNN, and Decision Tree (CART). In addition, the parameters of these classifiers are varied to generate the knowledge space Ω with 15 members. There is no requirement that GAEGA must take a specific type of classifier. These classifiers are used as an example for examining the performance of GAEGA and BAEGA. For adaboost, RS and bagging, since they can only take a certain type of classifier, CART is chosen as the base classifier, according to the related study in the literature [42]. For the test on the XOR data, MV, DT, NBF, GAEGA, BAEGA, and BKS take the *XLinear* and *YLinear* classifiers. For adaboost, RS and bagging, without loss of generalization, *XLinear* is used as the base classifier.

In all of the experiments, the data sets are randomly separated into training and testing sets. For the trained fusion approach such as DT, BF, BKS, or GAEGA, 66.7% of the training data are used to train the base classifiers, and the remaining 33.3% serve as a validation set to train the aggregation rule.

6.5.2 Experimental Results and Discussion

Table 6.9-6.12 report the average accuracy and standard error of all the methods for each of the 16 data sets. From the results, there are several observations.

The decision XOR is an interesting problem to study the performance of different MCSs. The methods, including the adaboost, bagging, RS, NBF, and DT, all failed to solve this problem. However, the reasons for their failure differ. For the adaboost, bagging, and RS, they can take only one of the classifiers. However, it is obvious that both the *XLinear* and *YLinear* classifiers are needed to solve

Table 6.9: Accuracy (in %) of different methods (1)

Methods	Data Sets			
	CBF	CC	WF	WF+DN
MV	97.3±0.70	96.7±0.69	98.7±0.74	98.9±0.71
AdaBoost	64.4±4.04	60.4±9.51	62.8±3.39	76.7±5.97
Bagging	98.6±0.92	98.0±0.25	98.3±0.57	98.6±0.17
RS	99.1±0.67	97.8±0.62	**99.8±0.27**	**99.6±0.02**
DT	99.4±0.60	98.8±0.76	98.9±0.94	98.9±0.86
NBF	99.6±0.44	98.6±0.71	98.8±0.41	98.8±0.65
BKS	60.8±4.55	60.5±2.10	66.0±5.25	64.0±4.55
GAEGA(multi)	99.8±0.55	98.8±0.63	99.3±0.77	98.9±0.55
GAEGA(addition)	99.4±0.70	95.1±1.43	98.0±1.46	99.2±0.59
BAEGA(multi)	**100.0±0.00**	**99.2±0.51**	98.9±1.24	99.2±0.80
BAEGA(addition)	99.1±2.05	98.9±0.56	98.5±1.00	98.7±1.30

Table 6.10: Accuracy (in %) of different methods (2)

Methods	Data Sets			
	GP	TR	TP	Clouds
MV	90.6±2.66	79.4±3.61	81.0±0.58	82.8±2.37
AdaBoost	91.7±0.63	80.8±3.53	87.1±1.34	88.2±0.94
Bagging	93.3±3.58	83.8±3.50	89.9±0.93	74.8±2.42
RS	95.3±1.66	85.3±4.09	91.6±0.87	75.3±1.11
DT	90.8±3.96	81.5±6.03	86.2±1.01	87.9±0.40
NBF	90.9±4.00	81.6±6.40	86.1±1.17	**89.0±0.54**
BKS	70.4±4.79	61.8±4.83	73.2±1.63	79.2±0.36
GAEGA(multi)	**96.5±2.02**	95.3±4.24	**97.2±0.84**	86.8±1.72
GAEGA(addition)	92.8±1.37	**98.0±1.12**	93.8±1.11	87.7±1.05
BAEGA(multi)	95.8±2.40	97.8±1.05	94.6±1.56	85.9±1.07
BAEGA(addition)	95.0±1.53	96.8±1.68	94.5±1.08	84.7±2.60

Table 6.11: Accuracy (in %) of different methods (3)

Methods	Data Sets			
	WDBC	WPBC	Phoneme	XOR
MV	77.0±2.85	72.2±3.82	75.9±2.10	48.4±7.11
AdaBoost	88.6±2.31	76.4±2.77	80.9±1.57	52.8±2.30
Bagging	75.4±6.75	74.5±3.87	75.5±1.94	51.2±1.65
RS	76.3±7.15	**78.4±2.37**	77.9±2.68	49.5±2.25
DT	93.0±0.79	63.8±4.53	**82.8±1.04**	54.2±6.13
NBF	**93.2±3.43**	77.7±4.06	76.1±1.52	47.7±7.24
BKS	92.5±1.25	77.0±3.94	72.9±0.44	**97.8±4.80**
GAEGA(multi)	92.7±0.67	72.9±3.65	81.9±2.60	97.6±4.82
GAEGA(addition)	92.3±1.86	72.6±6.73	80.9±3.46	97.9±4.13
BAEGA(multi)	92.0±1.01	73.4±6.29	82.1±1.60	97.3±3.34
BAEGA(addition)	91.9±1.90	71.8±3.84	81.1±2.44	97.5±3.79

Table 6.12: Accuracy (in %) of different methods (4)

Methods	Data Sets			
	Concentric	Ionosphere	BCW	German
MV	95.1±1.40	75.6±4.36	95.1±1.30	70.4±1.54
AdaBoost	78.8±2.71	78.7±3.93	95.4±1.04	64.1±2.40
Bagging	67.0±1.37	64.0±3.22	91.2±1.74	70.7±1.41
RS	69.7±1.82	64.6±3.73	92.4±1.90	**70.8±1.39**
DT	95.4±0.72	84.1±7.21	**96.5±1.33**	65.5±2.54
NBF	95.1±0.87	82.1±2.90	95.4±1.03	70.2±2.15
BKS	92.7±2.58	75.7±8.60	92.3±0.97	67.3±1.93
GAEGA(multi)	95.8±0.54	**89.1±2.82**	96.4±1.21	69.8±2.40
GAEGA(addition)	95.4±1.55	88.8±2.53	95.2±1.20	69.4±1.42
BAEGA(multi)	95.7±0.52	86.9±3.70	95.8±2.13	66.9±2.38
BAEGA(addition)	**96.1±1.34**	85.7±1.82	94.2±0.85	68.0±1.27

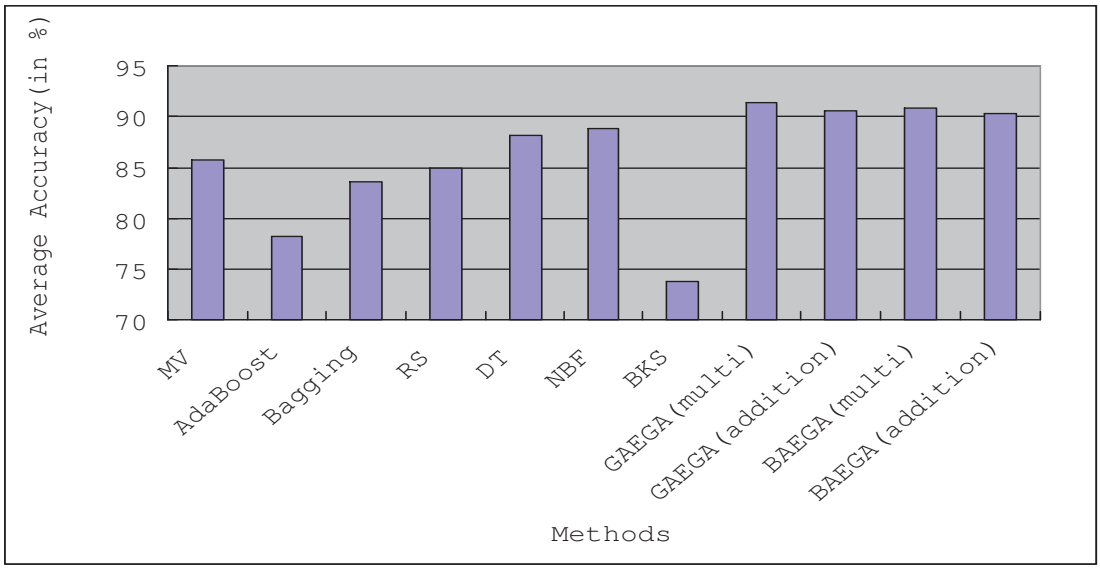

Figure 6.8: Average accuracy of GAEGA and BAEGA over different data sets

this problem. Therefore, the failure of adaboost, bagging, and RS is due to the insufficient representation capability of the ensemble. In contrast, NBF and DT can utilize both of the *XLinear* and *YLinear* classifiers. Their failures come from the ineffectiveness of the aggregation methods for the Decision XOR problem.

Each MCS has its own advantage in certain situations. None is universally better in all the tests. For example, RS outperforms the other methods in the WF, WF+DN, WPBC, and german data sets. NBF achieves the highest accuracy in the clouds and WDBC data sets. However, many methods are not very robust. For instance, MV, RS, and bagging are not good with the WDBC data. Adaboost has a severe overfitting problem on time series data such as CBF, CC, WF, and WF+DN. The performance of BKS is not reliable in many of the tests. The accuracy of DT and NBF is not satisfying in GP, TR and TP data sets. In contrast, the performances of GAEGA and BAEGA prove to be more robust. For each test, their performances are either the best or very close to the best. From Fig. 6.8[4], BAEGA and GAEGA are superior to the other methods over different data sets on

[4]Since the test on XOR data set is special and shows the advantage of GAEGA and BAEGA, this data set is excluded, when computing the average accuracy, to make the comparison as fair

average (XOR data is excluded).

There is no evidence to indicate whether GAEGA or BAEGA is stronger. Since their principles are similar, it is not surprising that their performances are close. In addition, the results show that both GAEGA and BAEGA with the multiplication rule are somewhat stronger than the ones with the addition rule. However, these differences are very small and stronger evidence is needed to support such a conclusion.

6.6 Discussion on the Computation Cost of Different MCSs

Besides the accuracy, the computation cost is another important criterion to evaluate the performance of MCSs. In this section, a rough estimation of the computation cost of different MCSs is conducted. The task for the analysis of the computation cost of MCSs is relatively difficult since the structures of MCSs are complex and it is involved in many other factors such as the computation cost of the base classifier. For simplicity, it is assumed that the computation cost of each base classifier in the ensemble is a constant t. In the analysis, the interest is to explore the relation between the computation cost and the number of used base classifiers h, the number of categories r and the number of training data N. Other computation costs are assumed to be trivial and negligible.

Assume the size of the validation data set is always a fixed portion of the total training data set. For the bootstrap method, B denotes the number of bootstrapped samples. For RS method, K denotes the number of generated subsets of features. Suppose that adaboost specifies the total number of iterations and denote this number with L. For BAEGA, it is assumed that the homogeneous ensemble is adopted. The estimated computation costs for different MCSs in the training and test phases are reported in Table 6.13. The results show that GESA is very time consuming. Its performance is in the exponential order of the number of available classifiers. In contrast, MV has the advantage in achieving the small computation

as possible

Table 6.13: Estimation of the Computation Cost of Different MCSs

Method	Training	Testing
MV	$O(ht)$	$O(h)$
AdaBoost	$O(LNt)$	$O(L)$
Bagging	$O(BNt)$	$O(B)$
RS	$O(KNt)$	$O(K)$
DT	$O(ht + hN)$	$O(hr)$
NBF	$O(ht + hN)$	$O(hr)$
BKS	$O(ht + h^r N)$	$O(h^r)$
GESA	$O(ht + 2^h h^r N)$	$O(2^h h^r)$
GAEGA	$O(ht + h^{2+r} N)$	$O(h^{1+r})$
BAEGA	$O(ht + B^{2+r} N)$	$O(B^{1+r})$

cost. For adaboost, bagging and RS, the computation cost depends on the number of generated classifiers. The computation costs for DT and NBF are modest. In the worst case, the computation cost for BKS could also be very high. Finally, although the average accuracy of GAEGA and BAEGA outperforms other methods, their computation costs in the worst case can be significantly higher than the methods such as MV, NBF, DT and so on. Therefore, the improvements on the average accuracy achieved by GAEGA and GEAGA are not cost-free.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The design of Multiple Classifier Systems (MCSs) is a challenging but promising research topic in machine learning, pattern recognition and data mining. In this thesis, MCSs are studied from the learning point of view. One premise is that a robust design of MCSs should follow some general principles of the supervised learning. One of the fundamental issues in designing the supervised learning algorithm is to fit the training data adequately and reducing the risk of overfitting.

In this thesis, the first objective is to propose the Generalized Exhaustive Search and Aggregation (GESA) approach as a robust design of MCSs. GESA achieves this objective through the development of a score function that is composed of two parts. One measures the goodness of fit of the ensemble on the data to training the aggregation rule. The other part measures the risk of overfitting. By maximizing the score function, a reasonable balance is achieved between these two parts. In the experimental results, the performances of GESA and other measurements are assessed by varying the size of the training data set. The traditional methods such as Behavior Knowledge Space (BKS) easily lead to the overfitting problem, and exhibits an unreliable performance for a small training data set. The learning procedures of other methods such as Majority Voting (MV), Decision Template (DT) and Naive Bayesian Fusion (NBF) are not very adaptive. Therefore, these methods

can not adequately learn the data in some cases. The results are quite similar to the ones of the nearest neighbor classifier and linear classifier. The empirical studies prove that the novel GESA exhibits a robust performance compared with the other methods. GESA achieves the highest average accuracy over all the test data sets, although it might not always provide the best accuracy in individual tests. Also, GESA is readily distinguished from other MCSs by the architecture and the level of the decision-making. It aggregates the decisions of a collection of ensembles, instead of the decisions from an ensemble of base classifiers.

The focus of GESA is on the robustness of the MCSs. GESA does not take into account other design issues, which renders it not to be very practical to solve real problems. Generalized Adaptive Ensemble Generation and Aggregation (GAEGA) and Bootstrapped Adaptive Ensemble Generation and Aggregation (BAEGA) are two extensions of GESA in order to provide a data-adaptive design and to reduce the computation cost. The investigation, described in this thesis, confirms that a data-adaptive design should select only those representations and classifiers that generate the effective decisions. Other representations and classifiers act as the noise ones and tend to generate the noise decisions. It is true that many aggregation methods can dynamically discern the effective decisions from the noise ones at the decision aggregation layer. However, learning such knowledge is not cost-free. Given that the training data set fixed, it is reasonable to assume that the performance of the decision aggregation worsens when there are more noise decisions. The newly developed GAEGA and BAEGA can be readily adopted to identify the ensemble of effective representations and classifiers by maximizing the goodness of fit to the training data. The logic is that the representations and classifiers are regarded as effective if the ensemble, generated by them, fits the data well.

In addition, GAEGA and BAEGA reduce the computation cost by generating and searching the ensemble in a polynomial order of the size of the knowledge space Ω. Therefore, GAEGA and BAEGA tend to be more practical than GESA. Although they are similar in the ensemble generation and decision aggregation, they manipulate the data in a completely different manner. BAEGA does not split the training data to train the base classifier and the aggregation rule separately in a pre-determined way. Instead, BAEGA relies on the bootstrap sample to train the

base classifiers. The idea to use the entire training data for the aggregation rule is partial derived from the *out of bootstrap* method. The empirical studies in Chapter 6 prove that GAEGA and BAEGA are robust and outperform the methods that are tested.

Multiple Input Representation-Adaptive Ensemble Generation and Aggregation (MIR-AEGA) is derived from GAEGA, applying for the classification of time series data. Although the empirical studies of Keogh et al. [55] are extensive, their conclusions can not be generalized. The performances of the nearest neighbor classifier, with either the Eucledian distance or Dynamic Time Warping (DTW), are quite local. These methods can not prevent the overfitting problem effectively. In addition, the claim that a certain similarity measure can adequately model the temporal information in the data is not very convincing. The simu data set is an artificial counter-example to show how both of them can fail in some situations. The experimental results relate that the novel MIR-AEGA can handle the classification of time series data effectively. The success of MIR-AEGA is attributed to the following three reasons. The dynamic information in the time series data is adequately represented in a multiple-representation manner, MIR-AEGA adaptively selects the effective representations and classifiers and filters others as noise, and MIR-AEGA dynamically tweaks the trade-off between the goodness of fit and the risk of overfitting. Therefore, MIR-AEGA is an effective and efficient approach for the classification of time series data.

7.2 Future Work

Although the proposed MCSs approaches are successfully applied to a number of tasks, their performances can be further improved in several ways.

- Develop a More Effective Score Function
  The addition and multiplication rules are employed in this research to build the score function for the Adaptive Aggregation (AA) approach. In fact, the way to balance the goodness of fit to the training data and the risk of overfitting is an open problem. Also, other rules can be employed for the AA

approach in the future study. Moreover, other measures for the goodness of fit and the risk of overfitting can be investigated and employed.

- Generate an Ensemble More Efficiently
  Although both BAEGA and GAEGA significantly reduce the computation cost, compared with GESA, they are still in the polynomial order of the size of the knowledge space Ω. For some time critical applications such as the online classification, a more efficient method is desirable.

- Tune the Parameters of the AA Approach Automatically
  There are a few pre-determined parameters in the AA approach. For example, k controls the weight of $G^k(p_i^m, \hat{p}_i^m, n_i)$, and affects the accuracy of the entire system. N_g controls the accuracy of the numerical estimation, and determines the trade-off between the accuracy and computation cost of the whole system. More advanced algorithms can be designed to adaptively learn the values of the parameters to enhance the performance of the AA approach.

- Learn to Split Training Data Set Adaptively for GAEGA
  GAEGA adopts a fixed split of the training data to train the base classifiers and aggregation rule, separately. It is likely that the base classifiers demand more data in some cases, whereas the aggregation rules needs more data in other cases. Therefore, an adaptive split of the training data might improve the performance of GAEGA.

- Extend MIR-AEGA to Handle Multivariate Time Series Data
  Within the current discussion of MIR-AEGA, the focus is on univariate time series data. In future studies, MIR-AEGA can be extended to handle multivariate time series data.

- Apply the Proposed Approach to Other Applications
  In this thesis, the application of the proposed GAEGA is to classify time series data. GAEGA can also be applied to more specific applications such as speech recognition, text classification, and image recognition in the future research.

List of Publications Resulting from this Thesis

Journal Papers

1. L. Chen, M. Kamel, A generalized adaptive ensemble generation and aggregation approach for multiple classifier systems, submitted to Pattern Recognition.

2. L. Chen, M. Kamel, Multiple input representation-adaptive classifier ensemble generation for the classification of time series data, to be submitted to Pattern Analysis & Application.

3. L. Chen, M. Kamel, A bootstrapped adaptive method for classifier ensemble generation and aggregation, to be submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

Conference Papers

1. L. Chen, M. Kamel, A new design of multiple classifier system and its application to the classification of time series data, SMC2007, Accepted.

2. L. Chen, M. Kamel, Design of multiple classifier systems for time series data, MCS2005 LNCS 3541 (2005) 216-225.

3. L. Chen, M. Kamel, J. Jiang, A modular system for the classification of time series data, MCS2004 LNCS 3077 (2004) 134-143.

Bibliography

[1] R. Alhajj, A. Elnagar, Multiagents to separating handwritten connected digits, IEEE Transactions on Systems, Man, and Cybernetics, Part A, 35:5 (2005) 593-602.

[2] D. Achlioptas, Database-friendly random projections, In Proc. ACM Symp. on the principles of database systems (2001) 274-281.

[3] M. Aksela, J. Laaksonen, Using diversity of errors for selecting members of a committee classifier, Pattern Recognition 39 (2006) 608-623.

[4] M. Aksela, Comparison of classifier selection methods for improving committee performance, MCS2003 LNCS 2709 (2003) 84-93.

[5] C.M. Antunes and A.L. Oliveira, Temporal data mining: an overview, In Workshop on Temporal Data Mining (2001) 1-15.

[6] M.R. Ahmadzadeh, M. Petrou, Use of dempster-shafer theory to combine classifiers which usse different class boundaries, Pattern Analysis & Application 6 (2003) 41-46.

[7] R. Bajaj, S. Chaudhury, Signature verification using multiple neural classifiers, Pattern Recognition 30 (1997) 1-7.

[8] L. Breiman, J.H. Friedman, A. Olshen and C.J. Stone, *Classification and Regression Trees*, Chapman and Hall, New York, 1993.

[9] C.S. Burrus, R.A. Gopinath and H. Guo, *Introduction to Wavelets and Wavelet Transformation: A Primer*, Prentice-Hall, 1998.

[10] L. Breiman, Random forests, Machine Learning 45:1 (2001) 5-32.

[11] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, Knowledge Discovery and Data Mining (2001) 245-250.

[12] K. Chan, A.W. Fu, Efficient time series matching by wavelets, In proceedings of internation conference on data engineering (1999) 126-133.

[13] S.B. Cho, J.H. Kim, Combining multiple neural networks by fuzzy integal and robust classification, IEEE Transatction on Systems Man Cybernet 25 (1995) 380-384.

[14] L. Chen, M. Kamel, J. Jiang, A modular system for classification of time series data, MCS2004 LNCS 3077 (2004) 134-143.

[15] J. Czyz, J. Kittler, L Vandendorpe, Multiple classifier combination for face-based identity verification, Pattern Recognition 37 (2004) 1459-1469.

[16] J.J.R. Diez, C.J.A. Gonzálex, Applying boosting to similarity literals for time series classification, MCS2000 LNCS 1857 (2000) 210-219.

[17] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern classification 2nd Edition*, Wiely-Interscience, 2000.

[18] K. Deng, A.W. Moore, M.C. Nechyba, Learning to recognize time series: combining ARMA models with memory-based Learning, IEEE International Symposium on Computational Intelligence in Robotics and Automation 1 (1997) 246-250.

[19] C. Dietrich, G. Palm, K. Ride, F. Schwenker, Classification of bioacoustic time series based on the combination of global and local decisions, Pattern Recognition 37 (2004) 2293-2305.

[20] C. Dietrich, G. Palm, F. Schwenker, Decision templates for the classification of bioacoustic time series, Proceedings of IEEE Workshop on Neural Networks and Signal Processing (2002) 159-168.

[21] C. Dietrich, F. Schwenker, G. Palm, Classification of time series utilizing temporal and decision fusion, MCS2001 LNCS 2096 (2001) 378-387.

[22] T.G. Dietterich, Ensemble methods in machine learning, MCS2000 LNCS 1857 (2000) 1-15.

[23] ELENA Project: http://www.dice.ucl.ac.be/neuralnets/Research/Projects/ELENA /elena.htm.

[24] E. Bradley, R.J. Tibshirani, *An introduction to the bootstrap*, Chapman and HALL/CRC 1993

[25] J.B. Farison, Y.G. Park, Q. Yu, H. Lu, KL transformation of spatially invariant image sequences, In proceedings of SPIE 3034 (1997) 188-199.

[26] G. Fumera, F. Roli, Analysis of error-reject trade-off in linearly combined multiple classifiers, Pattern Recognition 37 (2004) 1245-1265.

[27] G. Fumera, F. Roli, A theoretical and experimental analysis of linear combiners for multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 27:6 (2005) 942-956 .

[28] M. Gavrilov, D. Anguelov, P. Indyk, R. Motwani, Mining the stock market: which measure is best? KDD (2000) 487-496.

[29] J. Ghosh, S. Beck, C.C. Chu, Evidence combination techniques for robust classification of short-duration oceanic signals. In SPIE Conference on Adaptive and Learning Systems, SPIE Proceeding 1706 (1992) 266-276.

[30] A.K. Ghosh, P. Chaudhuri, C.A. Murthy, On visualization and aggregation of nearest neighbor classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 27:10 (2005) 1592-1602.

[31] C.J.A. Gonzálex, J.R. Diez, Time series classification by boosting interval based literals, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 11 (2000) 2-11.

[32] J. Ghosh, L. Deuser, S. Beck, A neural network based hybrid System for detection, characterization and classification of short-duration oceanic signals, IEEE Journal of Ocean Engineering 17:4 (1992) 351-363.

[33] J. Ghosh, Multiclassifier systems: back to the future, MCS2002 LNCS 2364 (2002) 1-15.

[34] L. Gupta, M. McAvoy, J. Phegley, Classification of temporal sequences via prediction using the simple recurrent neural network, Pattern Recognition 33 (2000) 1759-1770.

[35] P. Geurts, *Contributions to decision tree induction: bias/variance tradeoff and time series classification*, A PhD thesis, Department of Electrical Engineering, University of Liege, Belgium, 2002.

[36] G. Giacinto, F. Roli, Dynamic classifier selection, MCS2000 LNCS 1857 (2000) 177-189.

[37] G. Giancinto, F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognition 34 (2001) 1879-1881.

[38] G. Giacinto, F. Roli, Methods for dynamic classifier selection, ICIAP '99, 10th International Conference on Image Analysis and Processing (1999) 659-664.

[39] K.Y. Hung, R.W.P. Luk, D.S. Yeung, K.F.L. Chung, W. Shu, A multiple classifier approach to detect Chinese character recognition errors, Pattern Recognition 38 (2005) 723-738.

[40] W.H. Hsu, S.R. Ray, Construction of recurrent mixture models for time series classification, In Proceedings of the International Joint Conference on Neural Networks 3 (1999) 1574-1579.

[41] Y.S. Huang, C.Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Transactions on Pattern Analysis and Machine Intelligence 17:1 (1995) 90-94.

[42] T. Hastie, R. Tibshirani, J. Friedman, *Elements of Statistical Learning: Data mining, inference, and prediction*, Published by Springer-Verlag Inc., 2001

[43] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20:8 (1998) 832-844.

[44] M. Jordon, R. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Computing 34 (1994) 181-214.

[45] J. Kittler, F.M. Alkoot, Sum versus vote fusion in multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 25:1 (2003) 110-115.

[46] M. Kaya, R. Alhajj, A novel approach to multiagent reinforcement learning: utilizing OLAP mining in the learning process, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 35:4 (2005) 582-590.

[47] L. Kuncheva, J. Bezdek, R. Duin, Decision templates for multiple classifier fusion: An experimental comparision, Pattern Recognition 34 (2001) 299-314.

[48] H.J. Kang, D. Doermann, Evaluation of the information-theoretic construction of multiple classifier systems, ICDAR (2003) 789-793.

[49] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20:3 (1998) 226-239.

[50] E. Keogh, S. Kasetty, On the need for time series data mining benchmarks: a survey and empirical demonstration, Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (2002).

[51] L. Kuncheva, Switching between selection and fusion in combining classifiers: an experiment, IEEE Transactions on Systems, Man and Cybernetics 32:2 (2002) 146-156.

[52] L. Kuncheva, A theoretical study on six classifier fusioin strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 24:2 (2002) 281-286.

[53] H. Kang, S. Lee, An information-theoretic strategy for constructing multiple classifier systems, In Proceedings of the 15th ICPR 2 (2000) 483-486.

[54] M.W. Kadous, Temporal classification: *extending the classification paradigm to multivariate time series*, A PhD Thesis, University of New South Wales, 2002.

[55] E. Keogh, X. Xi, X., L. Wei, C. A. Ratanamahatana,(2006) "The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/ eamonn/time_series_data/"

[56] E. Keogh, M. Pazzani, An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback, In proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining (1998) 239-241.

[57] L.I. Kuncheva, C.J. Whitaker, C.A. Shipp, R.P.W. Duin, Is independence good for combining classifiers? Proc. of ICPR2000, 15th Int. Conf. on Pattern Recognition 2 (2000) 168-171.

[58] M. Kamel, N. Wanas, Data dependence in combining classifiers, MCS 2003 LNCS 2709 (2003) 1-14.

[59] C.L. Liu, Classifier combination based on confidence transformation, Pattern Recognition 38 (2005) 11-28.

[60] D. Luca, G. Giorgio, F. Roli, G.L. Marcialis, A study on the performances of dynamic classifier selection based on local accuracy estimation, Pattern Recognition 38 (2005) 2188-2191.

[61] L. Lam, Classifier combinations: implementations and theoretical issues, MCS 2000 LNCS 1857 (2000) 77-86.

[62] Y. Lu, C.L. Tan, Combination of multiple classifiers using probabilistic dictionary and its application to postcode recognition, Pattern Recognition 35 (2002) 2823-2832.

[63] X.Z. Liu, L. Zhang, M.J. Li, H.J. Zhang, D.X. Wang, Boosting image classification with LDA-based feature combination for digital photograph management, Pattern Recognition 38 (2005) 887-901.

[64] A. Murua, Upper bounds for error rates of linear combinations of classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 24:5 (2002) 591-602.

[65] D.J. Mashao, M. Skosan, Combining classifier decisions for robust speaker identification, Pattern Recognition 39 (2006) 147-155.

[66] Y. Morinaka, M. Yoshikawa, T. Amagasa, S. Uemura, The L-index: an Indexing structure for efficient subsequence matching in time sequence databases, PAKDD (2001) 51-60.

[67] F. Masulli, G. Valentini, Effectiveness of error correcting output codes in multiclass learning problems, MCS2000 LNCS 1857 (2000) 107-116.

[68] O. Melnik, Y. Vardi, Cun-Hui Zhang, Mixed group ranks: preference and confidence in classifier combination, IEEE Transactions on Pattern Analysis and Machine Intelligence 26:8 (2004) 973-981.

[69] A. Opelt, A. Pinz, M. Fussenegger, P. Auer, Generic object recognition with boosting, IEEE Transactions on Pattern Analysis and Machine Intelligence 28:3 (2006) 416-431.

[70] N.C. Oza, K. Tumer, Input decimation ensembles: decorrelation through dimensionality reduction, MCS2001 LNCS 2096 (2001) 238-247.

[71] R.J. Povinelli, M.T. Johnson, M.T., A.C. Lindgren, Time series classification using Gaussian mixture models of reconstructed phase spaces, IEEE Transaction on Knowledge and Data Engineering 16:6 (2004) 779-783.

[72] S. Prabhakar, A.K. Jain, Decision-level fusion in fingerprint verification, Pattern Recognition 35 (2002) 861-874.

[73] I. Popivanov, R.J. Miller, Similarity search over time series data using wavelets, In proceedings of the 18th international conference on data engineering (2002) 802-813.

[74] O. Pujol, P. Radeva, J. Vitria, Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes, IEEE Transactions on Pattern Analysis and Machine Intelligence 28:6 (2006) 1007-1012.

[75] D. Partridge, W.B. Yates, Engineering multiversion neural-net systems, Neural Computation 8 (1996) 869-893.

[76] D. Refiei, On similarity-based queries for time series data, In proceedings of the 15th IEEE Int'l Conference on Data Engineering (1999) 410-417.

[77] D. Roverso, Neural ensembles for event identification, In Proc. of Safeprocess'2000, The 4th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (2000) 478-483.

[78] G. Rogova, Combining the results of several neural network classifiers, Nerual Networks 7 (1994) 777-781.

[79] D. Ruta, B. Gabrys, Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting, MCS2001 LNCS 2096 (2001) 399-408.

[80] F. Roli, G. Giacinto, G. Vernazza, Methods for designing multiple classifier systems. MCS 2001 LNCS 2096 (2001) 78-87

[81] L. Rabiner, J.B. Hwang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[82] C.A. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints, In proceedings of the Fourth SIAM International Conference on Data Mining (2004).

[83] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence 28:10 (2006) 1619-1630.

[84] F. Roli, S. Raudys, G.L. Marcialis, An experimental comparison of fixed and trained fusion rules for crisp classifier outputs, MCS2002 LNCS 2362 (2002) 232-241.

[85] J.F. Roddick, M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods, IEEE Transactions on Knowledge and Data Engineering 14:4 (2002) 750-767.

[86] J.S. Rao, R. Tibshirani, The out-of-bootstrap method for model averaging and selection, Technical Report, Department of Statistics, University of Toronto, May 1997.

[87] N. Saito, *Local Feature Extraction and Its Applications using a Library of Bases*, A PhD thesis, Department of Mathematics, Yale University, 1994.

[88] A.J.C. Sharkey, Types of multinet system, MCS2002 LNCS 2362 (2002) 108-117.

[89] A. Senior, A combination fingerprint classifier, IEEE Transactions on Pattern Analysis and Machine Intelligence 23:10 (2001) 1165-1174.

[90] Q. Isaac Moro Sancho, Carlos Alonso Gonzlez, Juan Jos Rodrguez Diez, Applying simple combining techniques with artificial neural networks to some standard time series classification problems, Artificial Neural Networks in Pattern Recognition (2001) 43-50.

[91] D.F. Specht, Probabilistic Neural Networks, Neural Networks 3 (1990) 109-118.

[92] A.J.C. Sharkey, N.E. Sharkey, Diversity, selection and ensembles of artificial neural nets, In proceedings of Third International Conference on Neural Networks and their Applications (1997) 205-212.

[93] A.J.C. Sharkey, N.E. Sharkey, U. Gerecke, G.O. Chandroth, The "Test and Select" approach to ensemble combination, MCS2000 LNCS 1857 (2000) 30-44.

[94] M.J. Tax, M. V. Reukelen, R.P.W. Duin, J. Kittler, Combining multiple classifiers by averaging or by multiplying? Pattern Recognition 33 (2000) 1475-1485.

[95] K.M. Ting, I.H. Witten, Issues in stacked generalization, Journal of Artificial Intelligence Research 10 (1999) 271-289.

[96] UCI Machine Learning Repository: http://www.ics.uci.edu/ mlearn/MLRepository.html.

[97] G. Valetini, F. Masulli, Ensembles of learning machines, Neural Nets WIRN Vietri-2002 LNCS 2486 (2002) 3-19.

[98] T. Windeatt, G. Ardeshir, Boosted tree ensembles for solving multiclass problems, MCS2002 LNCS 2362 (2002) 42-51.

[99] N.M. Wanas, R.A. Dara and M.S. Kamel, Adaptive fusion and co-operative training for classifier ensembles, Pattern Recognition 39 (2006) 1781-1794.

[100] N. Wanas, M. Kamel, Weighted combining of neural network ensembles, International Joint Conference on Neural Networks (IJCNN'02) 2 (2002) 1748-1752.

[101] K. Woods, K. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis and Machine Intelligence 19:4 (1997) 405-410.

[102] T. Windeatt, Vote counting measures for ensemble classifiers, Pattern Recognition 36 (2003) 2743-2756.

[103] C. Wang, X.S. Wang, Multilevel filtering for high dimensional nearest neighbor search, In proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2000) 37-43.

[104] L. Xu, A. Krzyzk, C. Suen, Methods of combining multiple classifiers and their Application to handwriting recognition, IEEE Transaction on Systems, Man and Cybernetics 22:3 (1992) 418-435.

[105] B.K. Yi, C. Faloutsos, Fast time sequence indexing for arbitrary L_p norms, VLDB (2000) 385-394.

[106] H. Zouari, L. Heutte, Y. Lecourtier, Controlling the diversity in classifier ensembles through a measure of aggrement, Pattern Recognition 38 (2005) 2195-2199.