Multi-objective and Semi-supervised Heterogeneous Classifier Ensembles

Shenkai Gu

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

Department of Computer Science Faculty of Engineering and Physical Sciences University of Surrey Guildford, Surrey, GU2 7XH, U.K.

April 2016

© Shenkai Gu 2016
Abstract

In the recent years, many applications in machine learning involve an increasingly large number of features and samples, which poses new challenges to many learning algorithms. To address these challenges, ensemble learning methods, which uses multiple base learners, have been proposed to achieve better predictive performance.

This thesis covers a range of topics in ensemble classification, including multi-objective and semi-supervised heterogeneous classifier ensembles. We first present an empirical study on heterogeneous classifier ensembles, which confirms that heterogeneous ensembles outperform homogeneous ones and single classifiers. Secondly, we present a multi-objective ensemble generation method, which creates a group of members so that the diversity among the base learners could be explicitly maintained. The third topic of this thesis is a feature selection method for data that has a large number of features. By using the modified competitive swarm optimizer as the search algorithm, we are able to considerably reduce the number of features and at the same time improve the classifiers’ generalisation
performance. Finally, we present a novel semi-supervised ensemble learning algorithm, termed Multi-Train, that uses semi-supervised learning algorithms to learn from unlabelled data.
I am heartily thankful to my supervisor, Prof Yaochu Jin, for his constant support and guidance during my study. His academic knowledge and personal characteristics deeply impressed me and kept encouraging me to move forward. I would also like to thank my co-supervisor, Dr Norman Poh, and Dr Saied Sanei, for their gorgeous ideas and invaluable help.

I would also like to extend my gratitude to the Department of Computer Science, Faculty of Engineering and Physical Science (FEPS), University of Surrey for the financial support to make this research possible.

My thanks and gratitude also goes out to academic visitor Prof Xiaoyan Sun and Prof Chaoli Sun, colleague Dr Yun Yang, Dr Hui Wang, Ran Cheng and everyone in the department. Their kind support, discussions, and feedback are hugely appreciated.

Finally, I would like to send my deepest appreciation to all my parents and friends for their endless love and encouragement that have supported me on a more general level throughout my entire academic studies.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Contents</td>
<td>i</td>
</tr>
<tr>
<td>vii</td>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>ix</td>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>xiii</td>
<td>List of Publications</td>
<td>xiii</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Contributions</td>
<td>3</td>
</tr>
<tr>
<td>1.4</td>
<td>Structure of this thesis</td>
<td>5</td>
</tr>
</tbody>
</table>
## Literature Review

2.1 Introduction ................................................................. 7

2.2 Classifier models ............................................................ 8
   2.2.1 Linear discriminant analysis (LDA) ................................. 8
   2.2.2 Support vector machine (SVM) ......................................... 9

2.3 Basics of ensembles .......................................................... 11
   2.3.1 Accuracy measures ....................................................... 12
   2.3.2 Diversity measures ....................................................... 12
   2.3.3 Diversity generation ..................................................... 17
   2.3.4 Model integration ......................................................... 18

2.4 Ensemble member generation ............................................... 21
   2.4.1 Single-objective approach ............................................. 21
   2.4.2 Multi-objective evolutionary algorithms (MOEAs) ............... 22
   2.4.3 Multi-objective approach ............................................. 24

2.5 Member selection for multi-objective ensembles .......................... 28
   2.5.1 Single-objective approaches ......................................... 29
   2.5.2 Multi-objective approaches ......................................... 30

2.6 Semi-supervised learning .................................................... 31
   2.6.1 Self-training ............................................................. 33
   2.6.2 Semi-supervised learning with generative models .................. 33
   2.6.3 Semi-supervised support vector machines (S3VMs) .............. 34
   2.6.4 Semi-supervised learning with graphs .............................. 34
   2.6.5 Semi-supervised learning with committees .......................... 35
3 Heterogeneous Classifier Ensembles for EEG-Based Motor Imaginary Detection

3.1 Introduction .................................................. 37

3.2 Feature extraction ........................................... 39
  3.2.1 Common spatial pattern (CSP) ......................... 39
  3.2.2 Autoregressive model (AR) ......................... 41
  3.2.3 Mean-variance statistics (MV) ....................... 42

3.3 Heterogeneous classifier ensembles .................... 42
  3.3.1 Single classifiers ........................................ 42
  3.3.2 Heterogeneous ensembles with the same feature .... 43
  3.3.3 Heterogeneous ensembles with different features ... 44

3.4 Experiments .................................................. 45
  3.4.1 Datasets and pre-processing ......................... 45
  3.4.2 Experimental results .................................. 46

3.5 Summary ..................................................... 52

4 Generating Diverse and Accurate Classifier Ensembles Using Multi-Objective Optimisation

4.1 Introduction .................................................. 53

4.2 Proposed method ............................................ 55
  4.2.1 Objectives ............................................. 55
  4.2.2 Diversity creation ..................................... 56
  4.2.3 Evolutionary algorithm ............................. 57
4.3 Experiments ......................................................... 59
  4.3.1 Experimental setup ........................................ 59
  4.3.2 Experimental results ...................................... 60
4.4 Summary ............................................................ 66

5 Feature Selection for High Dimensional Classification using A
   Competitive Swarm Optimizer ................................. 69
  5.1 Introduction ....................................................... 69
  5.2 Background ........................................................ 71
    5.2.1 The canonical particle swarm optimisation (PSO) algorithm 71
    5.2.2 The competitive swarm optimiser (CSO) algorithm .......... 72
    5.2.3 Metaheuristics for feature selection ......................... 74
  5.3 Proposed Method ............................................... 75
  5.4 Experimental studies .......................................... 78
    5.4.1 Experimental settings ...................................... 78
    5.4.2 Results ........................................................ 80
  5.5 Summary .......................................................... 88

6 Multi-Train .......................................................... 91
  6.1 Introduction ....................................................... 91
  6.2 Background ........................................................ 93
    6.2.1 Co-training ................................................ 93
    6.2.2 Tri-training ................................................ 95
6.3 Proposed method ........................................... 97
6.4 Experiments ............................................... 100
  6.4.1 Experimental settings ................................. 100
  6.4.2 Results ............................................... 104
6.5 Summary .................................................. 112

7 Conclusion and Future Work ............................. 115
  7.1 Summary and conclusions ............................... 115
  7.2 Future directions ....................................... 118
List of Figures

2.1 Illustration of Pareto optimal solutions of an MOP denoted by black dots, where solution A is known as a knee point. 23
2.2 Illustration of supervised vs semi-supervised learning [111]. 32
3.1 Diagram of an EEG-based motor imaginary detection 43
3.2 Structure of heterogeneous ensembles with same feature 44
3.3 Structure of heterogeneous ensembles with different features 44
4.1 The hypervolume averaged over the six test cases for the six experimental setups. 61
4.2 Comparison of single classifier and ensemble classifiers 63
4.3 Pareto fronts achieved in the 36 different setups. 64
4.4 Zoomed-in version of representative plots in Fig. 4.3 65
5.1 Average fitness of the best particle respect to each generation 82
5.2 Number of new solutions found in each generation. . . . . . . . . 83

5.3 Average number of selected features over generations. . . . . . . . 86

5.4 Graphics showing that the selected features (x-axes) of the best individual over generations (y-axes). The darkness of each pixel represents the frequency of the corresponding feature that has been selected in the respective generation across different experiments - a darker colour represents a higher frequency and vice versa. . . . 87
List of Tables

2.1 Coincident errors between a pair of classifiers . . . . . . . . . . . . 13
2.2 Range and maximum diversity criteria of measures . . . . . . . . 16

3.1 10-fold on training set - dataset aa . . . . . . . . . . . . . . . . 47
3.2 10-fold on training set - dataset al . . . . . . . . . . . . . . . . 47
3.3 10-fold on training set - dataset av . . . . . . . . . . . . . . . . 48
3.4 10-fold on training set - dataset aw . . . . . . . . . . . . . . . . 48
3.5 10-fold on training set - dataset ay . . . . . . . . . . . . . . . . 48
3.6 ‘Pick All’ and ‘Pick 3’ heterogeneous ensemble . . . . . . . . . . . 49
3.7 10-fold on entire set - dataset aa . . . . . . . . . . . . . . . . . . 49
3.8 10-fold on entire set - dataset al . . . . . . . . . . . . . . . . . . 49
3.9 10-fold on entire set - dataset av . . . . . . . . . . . . . . . . . . 50
3.10 10-fold on entire set - dataset aw . . . . . . . . . . . . . . . . . 50
3.11 10-fold on entire set - dataset ay .......................... 50

3.12 T-test statistics for comparing classifiers ensemble and single classifiers ........................................... 51

3.13 Comparison of our result with literature [157] .......................... 51

4.1 Dataset characteristics .............................................. 59

4.2 Experiment parameters ............................................. 60

4.3 List of ensemble methods ........................................... 60

4.4 Test accuracies of single classifier and ensemble classifiers .... 62

5.1 Dataset characteristics .............................................. 78

5.2 Average error rate .................................................. 80

5.3 Average number of selected features ................................. 84

6.1 Dataset characteristics .............................................. 103

6.2 The classification error rate of Multi-Train, tri-training and single classifier with features selected by CSO-based algorithm .... 105

6.3 The classification error rate of Multi-Train, tri-training and single classifier with features are transformed by PCA ............... 106

6.4 The classification error rate of Multi-Train, tri-training and single classifier with all original features ........................... 107

6.5 The classification error rate of Multi-Train, tri-training and single classifier with random tree classifier model .................. 108

6.6 The classification error rate of Multi-Train, tri-training and single classifier with Naive Bayes classifier model .................. 109
6.7 The classification error rate of Multi-Train, tri-training and single classifier with J4.8 decision tree classifier model 110

6.8 The classification error rate of Multi-Train, tri-training and single classifier with $k$NN classifier model 111

6.9 The classification error rate of Multi-Train with different heterogeneous base learner sets 112


Introduction

1.1 Background

In recent years, many real-world machine learning applications have used larger and larger amount of data, which either have an increased number of attributes or samples. However, not all the data are useful in finding the similarities and differences to separate the classes in data mining. They may contain noise, irrelevant information, artefacts, and so on. Even worse, labelling such data becomes relatively more expensive, as human experts are required to label the data manually. Traditional supervised learning algorithms require a certain amount of labelled data to build classifiers. Otherwise, the classifiers may under-fitted and thus have a poor generalisation. These issues give rise to three major concerns in machine learning area, which are: how to deal with noisy data, how to handle a large amount of data, specifically when there are a large number of attributes
to be taken into consideration, and how to treat only partially labelled data.

Quite a few algorithms have been developed in recent years to tackle these concerns, such as classifier ensembles to reduce the impact of noisy data, algorithms to reduce the dimension of features and semi-supervised learning (SSL) algorithms that bring unlabelled data into training process for building more accurate classifiers. Many theories have been studied in these algorithms to quantify and predict the most efficient ways to improve the generalisation performance. However, there are gaps between theories and applications that have not been thoroughly studied. One example is that we all know a good classifier ensemble requires its base learners to be as accurate as possible [1], and at the same time they must not fail on the same samples. It is not a trivial task to generate such base learners, nor straightforward to define an appropriate measure of the diversity among them.

1.2 Objectives

This thesis particularly focuses on extending the theory studies by applying them into applications and attempts to overcome these difficulties, by delivering multi-objective and semi-supervised heterogeneous classifier ensemble algorithms, that improves the generalisation ability of the target data.

The specific objectives of this thesis are summarised as follows:

- Study the effectiveness of using base learners that are trained from heterogeneous sources, where both features are extracted in different ways and the classifier models are different.

- Explore the accuracy and diversity requirements in the theory of classifier ensembles and find the trade-off between them to deliver more stable classifier ensembles.
• Formulate an efficient feature selection method for data with a large number of features so that a smaller yet better subset from the original feature set can be used to train the classifiers better and faster.

• Formulate an SSL algorithm that learns from heterogeneous base learners so that the diversity can be guaranteed. As per theory, the generalisation performance could also be improved.

1.3 Contributions

These objectives have led to several novel contributions, which are peer reviewed, published and listed in this thesis. Each contribution is briefly described as follows:

The first contribution is applied on learning electroencephalography (EEG) signals for motor imaginary detection. Our method benefits from multiple classifier models feeding by different features that extracted by various methods. The results show significant improvement compared with those tests made with single classifier or same feature extraction method. Furthermore, although the classifier models and feature extraction methods used in the proposed algorithm is relatively simpler than those used by other researchers, the results are still better than most of them. This work has been published in [2].

The second contribution is an extensive research on the trade-off between accuracy and diversity measures. The proposed algorithm generates a group of ensemble members at a time so that the explicit diversity can be measured. By using a multi-objective optimisation (MOO) algorithm, i.e., the elitist non-dominated sorting genetic algorithm (NSGA-II), we produced a number of Pareto optimal ensemble groups. After testing the test set on these solutions, we found that the accuracy has superior importance than diversity, which said, those solutions with
higher accuracy but poorer diversity perform worse than those with slightly lower accuracy but better diversity. The verdict of this work is that sorting by the average training accuracy, solutions within the range of top 10% are more likely to outperform others, although the best solution is hard to be found without further information. This work has been published in [3].

The third contribution is to use one of the recent PSO variants, namely, competitive swarm optimiser (CSO), to select a small feature subset from the original large amount of attributes in the dataset. As the original CSO algorithm works in continuous space, a few modifications are made to fit the needs of solving discrete problems in feature selection. Mapping and archiving technologies are proposed for mapping the solutions in continuous space into discrete values and avoiding duplicate evaluations respectively. The results show that this approach is effective and, unlike conventional PSO algorithms, less sensitive to the initialisation criteria, therefore the burden of tuning parameters is relaxed. The selected sub feature sets are also comparatively smaller than other PSO-based methods while the testing accuracies of classifiers trained on the selected features are also higher.

The fourth contribution is a heterogeneous SSL algorithm. Past work showed that one of the keys to build successful SSL algorithms is to train classifiers by different views from data. However, data collected in most machine learning tasks do not come with different views. Inspired by the previous of heterogeneous classifier ensemble, we proposed to simulate the multiple views from single view data by processing features in various ways and use them to train different classifier models. A novel algorithm, namely multi-train, has been developed. The experiments confirmed that the proposed algorithm outperforms existing work.

Apart from those contributions above which propose new findings, we also presented a review [4] of existing multi-objective ensemble generation methods, which covers the topics of accuracy and diversity measures, model integration, MOEAs in data mining, ensemble member and group generation and finally a discussion
on current difficulties and applications.

1.4 Structure of this thesis

The rest of this thesis consists of the following chapters:

Chapter 2: A literature review of existing multi-objective and SSL algorithms, which mainly includes the basis of ensemble learning, ways of generating diverse members, the diversity measures and methods of making the decision from multiple classifiers.

Chapter 3: Presents the early work that proposed to construct heterogeneous classifier ensembles for improving the generalisation ability. A heterogeneous classifier ensemble is that the base classifiers used to form the ensemble use different types of classifier models and input feature. Furthermore, the effectiveness of ensembles with selected base classifiers is also covered in this chapter. By comparing homogeneous with heterogeneous classifier ensembles, the results show that the latter perform better, and the best result can be seen as the ensembles that only contains a few selected best heterogeneous base members.

Chapter 4: Proposes a multi-objective classifier ensemble method that generates a group of members at a time. Unlike traditional methods that generate the members in a multi-objective way, and then selects those members in a second step, this work generates groups of members that those groups are directly used as an ensemble. With the proposed method, we can solve the issue that existing methods are unable to measure the diversity of an ensemble during generating the members. The findings show that although diversity is important in classifier ensembles, the average accuracy of base classifiers seems to influence the ensemble performance more. However, the trade-off between accuracy and diversity plays the most important role as neither those ensembles have highest average accuracy
nor diversity perform better than others.

**Chapter 5:** Solves the feature selection on high dimensional problems by using CSO algorithm. Traditional feature selection methods are inefficient in the dataset that contains a lot of attributes. Previous work attempted to use PSO or other evolutionary algorithms (EAs) to overcome this issue. However, the amount and quality of selected features are under satisfactory. The experiments show that the proposed method can choose a smaller yet better sub feature set with less computational complexity.

**Chapter 6:** A novel modification based on co-training and its successor, tri-training, that brings in heterogeneous base learner in SSL. The original co-training algorithm ensures the diversity by training different classifiers from each view of data. Unfortunately, most datasets do not naturally come with different views. Unlike tri-training that maintains diversity by manipulating samples, this work attempts to create different views by simulating the multiple views by using different feature transformation or extraction method. With this method, we can have more views from the original single-view data, yet ensure the independence among these views. By testing on those dataset used in the tri-training algorithm, the proposed method is proved to have significant improvements than the existing work.

**Chapter 7:** Finally concludes the main contributions in this thesis. Open thoughts and unsolved questions are also discussed in this chapter.
2.1 Introduction

In machine learning research, one of the main issues is the generalisation. Generalisation is the prediction ability of a learning model to predict on unseen data. In recent years, many researchers have tried to enhance the prediction performance of learning models by combining the outputs of a group of learners, which are often known as ensembles. It has been proved, both theoretically [5, 6] and empirically that ensembles can generate more accurate predictions than any of their individual members alone can when certain criteria are met.

Although ensembles often perform better than their individual members do, it is hard to generate the members that meet the theoretical criteria and able to guarantee a better generalisation than a single model alone. It has been shown [7–9]
that any member of a good ensemble must perform better than random guessing (the accuracy requirement) and make errors in different parts of the input spaces (the diversity requirement), and the trade-off between these two criteria is essential.

It has been revealed, however, that diversity among base learners must decrease when they approach the highest level of accuracy [10]. From the optimisation point of view, the objective of enhancing accuracy and increasing diversity are likely to be conflicting with each other. Consequently, it is not trivial to strike a good balance between the two requirements. To address this issue, researchers attempted to aggregate the objective on accuracy with a diversity term, converting the two-objective into one single-objective optimisation (SOO) problem. One difficulty there is that a hyper-parameter needs to be empirically pre-specified and only one optimal solution can be obtained [11]. Many researchers [11–15] took into account the conflicting nature of accuracy and diversity and attempted to evolve a set of base learners using MOO approach, which is a more natural and straightforward machine learning method that can generate a set of solutions (learners) to be selected for constructing an ensemble.

2.2 Classifier models

Before talking about classifier ensembles, we want first to give a brief description of some key classifier models that we have used in the experiments.

2.2.1 Linear discriminant analysis (LDA)

Linear discriminant analysis (LDA) is also known as Fisher’s LDA or FDA [16], which is a binary classifier specified by discriminant functions. The LDA classifier assumes a normal distribution of the data and the covariance matrices, denoted
by $\Sigma$, of both classes, are equal. The separating hyperplane can be estimated by seeking the projection that maximises the distance between the two classes and minimises the interclass variance.

We denote the means of each class by $\mu_1$ and $\mu_2$ respectively, and an arbitrary feature vector by $x$ [17]. Define:

\[
D(x) = \begin{bmatrix} b \, w^\top \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} \quad (2.1)
\]

\[
w = \Sigma^{-1}(\mu_2 - \mu_1) \quad (2.2)
\]

\[
b = -w^\top \mu \quad (2.3)
\]

\[
\mu = \frac{1}{2}(\mu_1 + \mu_2), \quad (2.4)
\]

where $D(x)$ is the distance of the feature vector $x$ from the separating hyperplane which was described by its normal vector $w$ and bias $b$. The observation $x$ is classified as the first class if $D(x)$ is smaller than 0; otherwise as the other class.

### 2.2.2 Support vector machine (SVM)

Support vector machines (SVMs) use a discriminant hyperplane to separate classes [18, 19], where the selected hyperplane is optimised to maximise the distance between the nearest training points [18, 19].

Suppose we have a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i$ are feature vectors and $y_i$ are the labels, $i = 1, 2, \ldots, n$. The decision function can be formed as

\[
f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b\right), \quad (2.5)
\]

where $\alpha_i$ are embedding coefficients and $K(x, x_i)$ is called the kernel function.
The optimal decision function is computed using quadratic programming:

\[
\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]

subject to \( \alpha_i \geq 0, i = 1, \ldots, n \)

\[
\sum_{i=1}^{n} \alpha_i y_i = 0.
\]

(2.6)

SVMs that use a linear discriminant hyperplane are called linear support vector machines (LSVMs), whose kernel function \( K(x, x_i) \) is defined as:

\[
K(x, x_i) = \langle x, x_i \rangle,
\]

(2.7)

where \( \langle x, x_i \rangle \) denotes the dot product.

Other than the linear kernel, SVMs can also use some more complex kernel functions, such as radial basis function (RBF), which is defined as:

\[
K(x, x_i) = \exp \left( \frac{-\|x - x_i\|^2}{2\sigma^2} \right),
\]

(2.8)

where \( \| \cdot \| \) denotes the Euclidean norm on the input space and \( \sigma \) is the width of RBF unit.

Those SVMs that use RBF kernels are called RBF-SVM. As the separation hyperplane is non-linear, RBF-SVM could separate complex data better, but at the same time also may be subject to over-fitting. The risk can be reduced by choosing the width \( \sigma \) properly through trial and error.
2.3 Basics of ensembles

In this section, we are going to present some brief background knowledge of ensembles, which include ways of measuring accuracies and diversities of ensemble members, how to generate diverse members, and methods of integrating members together to improve the generalisation performance against single models.

Some main notations in this section are summarised as follows:

- $N$: the total number of training samples
- $L$: the total number of ensemble members
- $x_i$: the $i$-th training sample, $i \in \{1, \ldots, N\}$
- $y_i^0$: the true label of $x_i$
- $H^*$: the ensemble classifier
- $H_j$: the $j$-th member in $H^*$, $j \in \{1, \ldots, L\}$
- $y_i^j$: the $H_j$'s prediction on $x_i$, $y_i^j = H_j(x_i)$
- $y_i^*$: the ensemble prediction of $i$-th training sample
- $w_j$: the weight of $H_j$ for constructing $H^*$
- $P_j$: the accuracy of $H_j$
- $P^*$: the accuracy of $H^*$
- $P$: the average accuracy of $H_j$, $P = \frac{1}{L} \sum_{j=1}^{L} P_j$
- $o_{ij}$: the oracle output of $x_i$ on $H_j$, $o_{ij} = \begin{cases} 1 & \text{if } y_i^j = y_i^0 \\ -1 & \text{otherwise} \end{cases}$
- $l_i$: the product of $L$ and sum of $w_j$ that $H_j$ classifies $x_i$ incorrectly, $l_i = L \sum_{o_{ij}=-1} w_j$
- $T_j$: the probability that $l_i = j$
2.3.1 Accuracy measures

For any ensemble model, the first and most important requirement is their ability to predict accurately from unseen data, which is defined as follows:

\[ P^j = \frac{1}{N} \sum_{i=1}^{N} d_i, \quad d_i = \begin{cases} 1 & \text{if } y_i^j = y_i^0 \\ 0 & \text{otherwise.} \end{cases} \]  

(2.9)

2.3.2 Diversity measures

For classification problems, diversity is produced when different classifiers predict different labels on the same input pattern [6, 15]. While assessing the accuracy of machine learning models is relatively well-established, measuring the degree of diversity is trickier. There is so far no general accepted definition of diversity [20, 21] and thus researchers tend to use different measures empirically, although Kuncheva et al. [22] demonstrated that some of the existing diversity measures may be misleading.

Diversity can be measured either in the input or output space. However, most work [13, 14, 21, 23, 24] focus on output diversity. Depending on whether the diversity is measured over pairs of learners, or over the training samples, we can divide the diversity measures into two categories, namely, pairwise diversity and non-pairwise diversity.

Pairwise diversity

Pairwise diversity is based on the prediction differences between each pair of ensemble members. It uses coincident errors between a pair of classifiers, as illustrated in Table 2.1, to calculate the degree of difference. The coincident error represents the behaviour of pairs of base learners on patterns and checks if both
learners simultaneously predict a pattern correctly/incorrectly or disagree with each other. The overall diversity is the average of all possible pairs.

Table 2.1: Coincident errors between a pair of classifiers

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{H}_k$ correct</th>
<th>$\mathcal{H}_k$ incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}_i$ correct</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$\mathcal{H}_i$ incorrect</td>
<td>$c$</td>
<td>$d$</td>
</tr>
</tbody>
</table>

Note: By definition, $a + b + c + d = N$

$a$, $b$, $c$ and $d$ in Table 2.1 denotes the count of patterns over training set that predicted correctly or incorrectly by each classifier. Note, since each sample could only be predicted correctly/incorrectly simultaneously by both classifiers, or by either classifier, the sum of $a$, $b$, $c$, and $d$ in this table must be $N$. By using these counts, researchers defined various measures, among which the most widely used ones are Q-statistic [25,26], correlation coefficient [22], disagreement measure [27], double fault [28], $\kappa$-statistic [29] and pairwise failure credit (PFC) [15]:

1. Q-statistic

$$Q_{i,j} = \frac{ad - bc}{ad + bc}$$ (2.10)

2. Correlation coefficient ($\rho$)

$$\rho_{i,j} = \frac{ad - bc}{\sqrt{(a + b)(c + d)(a + c)(b + d)}}$$ (2.11)

3. Disagreement measure

$$DIS_{i,j} = \frac{b + c}{N}$$ (2.12)

4. Double fault

$$DF_{i,j} = \frac{d}{N}$$ (2.13)

5. $\kappa$-statistic ($\kappa$)

$$\kappa_{i,j} = \frac{2(ac - bd)}{(a + b)(c + d) + (a + c)(b + d)}$$ (2.14)
6. Pairwise failure credit (PFC)

\[ PFC_{i,j} = \frac{b + c}{b + c + d} \]  

(2.15)

The overall diversity \( \Delta \) is denoted as follows, which is the average of the pairwise diversity \( \Delta_{i,j} \in \{Q, \rho, DIS, DF, \kappa, PFC\} \):

\[ \Delta = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \Delta_{i,j}. \]  

(2.16)

Non-pairwise diversity

Non-pairwise diversity directly measures a set of classifiers, which is based either on the variance, entropy or on the proportion of base classifiers that fail on randomly selected patterns.

Examples of non-pairwise diversity measures include Kohavi-Wolpert variance [22, 30], interrater agreement [31], difficulty measure [32], generalised diversity [33] and coincident failure diversity [34]. The definition can be seen as follows:

1. Kohavi-Wolpert variance

\[ KW = \frac{1}{NL^2} \sum_{i=1}^{N} (L - l_i)l_i \]  

(2.17)

2. Interrater agreement \( k \)

\[ k = 1 - \frac{\sum_{i=1}^{N} (L - l_i)l_i}{NL(L-1)P(1-P)} \]  

(2.18)
3. Difficulty measure $\theta$

$$\theta = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{L - l_i}{L} - \frac{1}{N} \sum_{i=1}^{N} \frac{L - l_i}{L} \right)^2 \quad (2.19)$$

4. Generational distance (GD)

$$GD = 1 - \frac{\sum_{j=1}^{L} \frac{j(j-1)}{L(L-1)} T_j^j}{\sum_{j=1}^{L} \frac{j}{L} T_j^j} \quad (2.20)$$

5. Coincident failure diversity

$$CFD = \begin{cases} 
0, & T_0 = 1, \\
\frac{1}{1 - T_0} \sum_{i=1}^{L} \frac{L - i}{L - l_i} T_i, & T_0 < 1 
\end{cases} \quad (2.21)$$

**Other diversity measures**

Apart from pairwise and non-pairwise diversity measures, Liu and Yao [35] proposed a different diversity measure in their negative correlation learning (NCL) algorithm, namely correlation penalty function, which measures the diversity of each ensemble member against the entire ensemble. This measure is denoted as:

$$CPF^j = \sum_{i=1}^{N} \left( y_i^j - y_i^* \right) \left[ \sum_{k=1, k \neq j}^{L} (y_k^j - y_k^*) \right] \quad (2.22)$$

Ambiguity [36] is another instance of measure that does not fall into the categories above, which measures the average offset of each ensemble member against the ensemble output.

Different diversity measures have different minimum and maximum values, and
they can focus on either looking for diversity or similarity. Table 2.2 illustrates
the range and maximum diversity criteria of those measures.

Table 2.2: Range and maximum diversity criteria of measures

<table>
<thead>
<tr>
<th></th>
<th>lower bound</th>
<th>upper bound</th>
<th>searching focus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pairwise diversity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q$</td>
<td>$-1$</td>
<td>1</td>
<td>↓</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$-1$</td>
<td>1</td>
<td>↓</td>
</tr>
<tr>
<td>$DIS$</td>
<td>0</td>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td>$DF$</td>
<td>0</td>
<td>1</td>
<td>↓</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>$-2$</td>
<td>2</td>
<td>↓</td>
</tr>
<tr>
<td>$PFC$</td>
<td>0</td>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td><strong>Non-pairwise diversity</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$KW$</td>
<td>0</td>
<td>1/4</td>
<td>↓</td>
</tr>
<tr>
<td>$k$</td>
<td>$-1/(L-1)$</td>
<td>1</td>
<td>↓</td>
</tr>
<tr>
<td>$\theta$</td>
<td>0</td>
<td>$L^2/4$</td>
<td>↓</td>
</tr>
<tr>
<td>$GD$</td>
<td>0</td>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td>$CFD$</td>
<td>0</td>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td><strong>Other diversity measures</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$CPF$</td>
<td>-</td>
<td>-</td>
<td>↓</td>
</tr>
</tbody>
</table>

*↑ denotes those measures looking for diversity thus the
diversity is greater when the value is larger while ↓
denotes the opposite, i.e., those measures looking for
similarity thus the diversity is greater when the value
is smaller.

Although there are various diversity measures proposed, it is difficult to make
comparisons regarding the appropriateness and superiority across them. In some
extreme cases, if the measure is not carefully chosen, increased diversity could
even affect the generalisation ability [37] adversely. It seems that the perfor-
ance of a diversity measure may depend on the context of the use of diversity
and data [38]. Dos Santos et al. [39] tried to use three overfitting control strate-
gies to find the relationship between diversity and performance, and found that
overfitting can be detected in ensemble member selection.
2.3.3 Diversity generation

Sharkey [40] summarised four ways of creating diversity in the neural network (NN) ensembles, such as different initial weights, different training data, different architectures of the networks and different learning algorithms. However, this taxonomy may not be applicable to other ensembles. For this reason, here we propose to categorise various ideas for creating ensemble diversity into functional and structural diversity. By functional diversity, we mean those methods that do not change the structure of base learners. In other words, all base learners use the same type of models and have the same structure. By contrast, structural diversity can be achieved by using different learning models, either the same type of models with various structures, or different types of models.

For example, in NN ensembles, we can create diversity by varying the initial weights of the base learners. Although this method is probably the most straightforward way to generate diversity, it turned out to be a less effective method [41], because the generalisations of the base learners were found to be similar. Fast committee learning [42] is another example, which explicitly modifies the starting point by taking snapshots of NN training progress, which are then used as ensemble members. Boosting [43] (including many variants) and Bagging [44] methods are good examples of data manipulation, which promote diversity by manipulating data. While Boosting and Bagging manipulate data by choosing a different subset of the training data for training different ensemble members, it is also possible to select the subset of features in the training data [45, 46] or a combination of both. Random forest [47] is another example of creating diversity by choosing different subsets of training data or features, which gives competitive results against Boosting and Bagging methods, particularly on classification problems. An improved method, Rotation forest [48], applies principal components analysis (PCA) on each subset to improve the generalisation ability. For models other than NNs, there are also tunable parameters for generating
diversity, such as $k$-nearest-neighbour ($k$NN) and SVMs.

To create structural diversity, we can generate various architectures of the same type of models. For example, for NNs, we can change the number of hidden nodes, activation functions, or neurone connectivities. In addition, various types of models can be employed for ensembles, which are often known as heterogeneous ensembles [49].

### 2.3.4 Model integration

After we generate a series of models, it is difficult to decide which one(s) to choose or how to combine them. Existing approaches for model integration can be categorised into two basic types [50], i.e., selection approaches and fusion approaches. The former approaches only select most locally accurate classifiers to generate the final prediction, by assuming each member is an expert in some local regions of the feature space, while the latter compound all or a selected subset of models to generate the result, by assuming all members make independent errors.

**Selection approaches**

Cross-validation (CV) is one of the most popular selection approaches in ensembles, where the accuracy of each classifier is measured using CV and the one with the highest accuracy is selected. There are two varieties of selection methods, which are either static or dynamic [51]. The static selection proposes to use the single best classifier over the whole dataspace. In contrast, the dynamic selection considers neighbourhood dataspace, where different classifiers are selected on different regions of data.

Winner-takes-all (WTA) is another widely applied method of action selection for intelligent agents. This system works by taking the highest activation as the
prediction. It is commonly seen in NN ensembles, where the outputs are the probabilities of each class the input belongs to.

Dos Santos et al. [50] proposed a three level dynamic overproduce-and-choose strategy to select the classifier ensembles, which apart from generation level, the selection phase is divided into optimisation and selection levels. They use either Bagging or random subspace to generate models and then finds high accuracy models when using SOO, or both high accuracy and diverse models when using MOO. The dynamic selection uses different strategies, which are ambiguity-guided, margin-based, class strength-based or with local accuracy. Their experiments demonstrated better results than static selection. In the fusion of all models methods, ambiguity-guided and margin-based selection outperformed others, and multi-objective methods find better solutions than single-objective methods.

**Fusion approaches**

Further to accuracy and diversity, Shipp and Kuncheva [34] found that fusion procedure may also dramatically influence the performance of ensembles.

For convenience, we denote the degree of support given by member $H^j$ to the hypothesis that a given input (observation) $x_i$ belongs to the $m$-th class $w_m$ as $d_{i,m}^j \in [0, 1]$, which is also called ‘soft label’ and normally an estimate of the posterior probability $P(w_m|x_i)$. Similarly, the ensemble’s ‘soft label’ is denoted as $d_{i,m}^*$. For classification problems, the conversion from ‘soft label’ into ‘crisp label’ $y_i^j$ can be based on the maximum membership rule, which is $y_i^j \leftarrow w_s$ iff $d_{i,s}^j \geq d_{i,m}^j, \forall m = 1, \ldots, c$ for member $H^j$, and similarly $d_{i,s}^* \geq d_{i,m}^*, \forall m = 1, \ldots, c$ for the ensemble $H^*$.

In the following, we very briefly introduce a few most widely used decision-making methods in ensemble generation. For other decision-making methods, the reader is referred to [52–54].
Majority voting, maximum, minimum, average and product operations are grouped together as they do not require any further training after the members are obtained. For majority voting [32], the most represented crisp class label is assigned to the observation, i.e., \( y^*_i \leftarrow w_s \) iff most \( y^j_i \) are \( w_s \). For other methods:

\[
d_{i,m}^* = \mathcal{O}(d_{i,m}^1, \ldots, d_{i,m}^L),
\]

(2.23)

where \( \mathcal{O}(\cdot) \) denotes the respective operation, i.e., maximum, minimum, average or product.

Different from majority voting, weighted voting takes the members’ training accuracy into account as ‘reliability’ of the members’ prediction so that better members will have a higher weight in voting. Assume \( w_m \) are continuous integer numbers:

\[
y^*_i = \text{round}\left(\frac{\sum_j y^j_i \cdot w_j}{\sum w_j}\right).
\]

(2.24)

Naive Bayes assumes that ensemble members are mutually independent, i.e.:

\[
d_{i,m}^* \propto \prod_{j=1}^L \hat{p}(w_m | \mathcal{H}^j(x_i) = y^j_i),
\]

(2.25)

where \( \hat{p}(w_m | \mathcal{H}^j(x_i) = y^j_i) \) are probability estimates that:

\[
\hat{p}(w_m | \mathcal{H}^j(x_i) = y^j_i) = \frac{\# \text{ patterns labelled } y^j_i \text{ by } \mathcal{H}^j \text{ whose true label is } w_m}{\# \text{ patterns labelled } y^j_i \text{ by } \mathcal{H}^j}.
\]

(2.26)

Because of the independence assumption of Naive Bayes approach, it is sensitive to the diversity of ensemble and has been proved that this approach has a high correlation with majority voting. It is also worth to mention that, although most decision-making methods have little correlation with diversity measures, Naive Bayes method shows a stronger correlation with double fault and the measure of
difficulty, which indicates potential benefit if those methods are combined [34]. It is worth noting that selection and fusion approaches do not necessarily conflict with each other. Kuncheva [55] proposed to combine these two approaches together by applying selection in regions that one classifier strongly dominates the others and fusion in rest of regions.

Stacking approaches

Stacking [56], which involves training a learning model to combine the predictions of ensemble members, is another way of integrating models. The idea of this algorithm is to separate the learning process into layers while the first layer gives predictions of different learners and the second layer uses these predictions as input features to find the final prediction.

2.4 Ensemble member generation

2.4.1 Single-objective approach

Early methods for ensemble generation used to aggregate the two objectives, accuracy and diversity into a scalar objective function using a hyper-parameter [11]:

$$J = E + \lambda \Omega,$$

(2.27)

where $E$ represents the accuracy measure and $\Omega$ is a diversity measure. By introducing hyper-parameter $\lambda$, we can control the balance between accuracy and diversity. In this way, traditional machine learning algorithms can be applied to generate ensemble members.

However, single-objective ensemble generation has a few major weaknesses. Firstly, the hyper-parameter must be predefined, which is not trivial, as it is difficult to
achieve an appropriate balance between accuracy and diversity. Secondly, only one single base learner can be found with one run of the algorithm, which means that the single-objective algorithm needs to be run for multiple times to obtain a number of base learners and different hyper-parameters need to be prescribed.

2.4.2 MOEAs

As shown above, ensemble generation needs to consider two conflicting objectives, accuracy and diversity and is a typical multi-objective optimisation problem (MOP). MOPs generally refer to those optimisation problems that involve multiple conflicting objectives:

$$\begin{align*}
\text{min} & \hspace{1em} \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \\
\text{s.t.} & \hspace{1em} \vec{x} \in X, \hspace{1em} \vec{f} \in Y,
\end{align*}$$

where $X \subset \mathbb{R}^n$ is the decision space and $\vec{x} = (x_1, x_2, \ldots, x_n) \in X$ is the decision vector, $Y \subset \mathbb{R}^m$ is the objective space and $\vec{f} \in Y$ is the objective vector, which is composed of $m$ objective functions $f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})$ that map $\vec{x}$ from $X$ to $Y$.

Since the objectives in an MOP conflict with each other, there does not exist one single solution that optimises all the objectives simultaneously. Instead, an MOP has a set of optimal solutions, known as the Pareto optimal solutions, which portray the trade-offs between different objectives. In MOO, one solution is said to dominate another when and only when all its objectives are no worse than the other and at least one objective is better. Solutions that cannot be dominated by any other feasible solution are known as Pareto optimal solutions [57], as illustrated by dots in Fig. 2.1. For most complex real-world optimisation problems, it is hard to know whether the non-dominated solutions approximated by an MOO algorithm are Pareto optimal or not.
Figure 2.1: Illustration of Pareto optimal solutions of an MOP denoted by black dots, where solution $A$ is known as a knee point.

Population-based optimisation algorithms such as EAs have shown to be particularly well suited for MOO since they are able to obtain a set of non-dominated solutions in one single run [58]. There are three classes of MOEAs in general, which include the dominance based methods, decomposition (or weighted aggregation) based methods and indicator based methods.

The first generation of dominance based MOEAs are often known as the non-elitism methods, e.g., genetic algorithm (GA) [59] and the multi-objective genetic algorithm (MOGA) [60]. Afterwards, with the proposal of the NSGA-II [61], the second generation of dominance based MOEAs, known as the elitism dominance based methods, had become prevalent [62–65]. Until today, many MOEAs are still proposed under the elitism framework, although the detail operators inside the framework can be different [66–68], most of which have replaced the genetic operators used in the original NSGA-II by other more advanced reproduction approaches.

The basic idea of decomposition (or weighted aggregation) based MOEAs is to decompose an MOP into a number of single-objective optimisation problems (SOPs) using weighted combinations of different objectives. The weight vectors can be either randomly generated [69, 70] or predefined with a uniform distribution [71]. Among various decomposition based methods, the recently proposed multi-objective evolutionary algorithm based on decomposition (MOEA/D) [72]
has attracted increasing attention due to its promising search performance and computational efficiency.

The idea of the indicator-based MOEAs is fairly straightforward, i.e., using the performance indicators as the selection criteria to control both convergence and distribution of the candidate solutions in an MOEA. The most well-known indicator is the hypervolume indicator [73], and correspondingly various hypervolume based MOEAs have been proposed [74–76].

It is worth noting that although MOEAs have witnessed rapid development during the past two decades, one open issue has been attracting increasing interests most recently, i.e., the many-objective optimisation [77], where the number of objectives is larger than three. For instance, although dominance based MOEAs have shown robustness on bi-objective and three-objective problems, it has been reported that for problems with more than three objectives, the performance of the algorithms will deteriorate rapidly due to the inability of dominance related mechanisms [78,79]. It has also been found that with the increase in the number of objectives, the computational complexity of hypervolume calculation will exponentially increase as well, which makes the time cost of indicator-based methods unfeasibly high [80].

Apart from NSGA-II, a few other MOEAs have been used as optimisation tools for multi-objective learning, such as elitist multi-objective genetic algorithm (EMOGA) [81], guided multi-objective genetic algorithm (G-MOEA) [57] Pareto envelope-based selection algorithm-II (PESA-II) [65], strength Pareto evolutionary algorithm-2 (SPEA2) [63] and niched Pareto genetic algorithm (NPGA) [82].

2.4.3 Multi-objective approach

The members of an ensemble can be generated either in parallel without knowing other members’ performance, or sequentially, i.e., one by one. For example,
Bagging generates members in parallel, as it randomly samples training data into subsets for training base learners, which does not measure the learner’s performance before generating another subset. By contrast, adaptive Boosting (AdaBoost) [83] is designed to enhance the possibility of selecting patterns that the previous base learner fails, thereby improving the performance of the ensemble. Thus, this method falls into the sequential ensemble generation category.

Based on the objectives used, a few categories of ensemble member generation methods in multi-objective perspective are as follows:

**Different error measures**

Inspired by memetic Pareto artificial neural network (MPANN) [84] and NCL [35] algorithms, Abbass [85] proposed two formulations of the ensemble. In the first variant, the training data is divided into two equal subsets using stratified sampling, and then the errors on the two training sets are used as the two objectives. The other method uses two objectives, which are the accuracy of the entire training set and on the same data but with Gaussian noise being injected into the desired output.

Bhowan et al. [86] proposed to use multi-objective genetic programming (MOGP) to evolve the ensemble members, in order to solve unbalanced classification problems. In MOGP, selection methods suggested in NSGA-II and SPEA2 are compared and the results show that the selection scheme in SPEA2 outperforms the non-dominated sorting. The reason might be that non-dominated sorting does not have a strong bias towards any of the objectives while selection in SPEA2 slightly biases to the solutions in the middle region of the Pareto front, which are more likely to have a good trade-off between accuracy and diversity.

The idea of using different error measures as both objectives has also been adopted in class imbalance online learning. Motivated by the finding that undersampling
online bagging (UOB) has better classification performance while oversampling online bagging (OOB) is more robust in imbalance learning. Wang et al. [87] used recalls by OOB and UOB methods as both objectives in order to get a balance of individuals’ accuracy and ensemble robustness. The proposed method creates two set of training patterns, by either oversampling the minority class or undersampling the majority class patterns. The non-dominated solutions are then used to construct an ensemble.

Accuracy vs. diversity

Chandra and Yao proposed the diverse and accurate ensemble (DIVACE) [13,88], which also uses MPANN as the EA, while the objectives are training accuracy and a negative correlation diversity measure proposed in NCL as the second objective. The authors have explored other training algorithms and diversity measures and proposed a new measure called PFC in [15] to replace the negative correlation measure in DIVACE. While negative correlation measures the probability of failures, PFC directly responds to the failures made by each member which makes it less continuous. As a result, the objective of training accuracy would have greater influence, which slightly decreased the test accuracy. However, as each member is more accurate, the WTA decision making performed better than before. Compared to other methods, both variants of the DIVACE algorithm achieved mostly better generalisation performance.

Accuracy vs. complexity

Jin et al. [89] proposed complexity as the second objective in addition to accuracy. The complexity of the NNs is measured by either the sum of the squared weights or the sum of the absolute value of the weights. Other suggested complexity measures include the number of connections in the NNss. The bi-objective op-
timisation that maximises the accuracy and minimises the complexity results in a number of NNs having various structures. Pareto optimal NNs near the knee point [90] are then used to construct NN ensembles.

Oliveira et al. [91] tried to reduce the number of selected features as the complexity objective, besides accuracy as the other objective in their algorithm. The ensemble members are generated and optimised by sampling features on the paradigm of ‘overproduce and choose’, and then selectively used to produce the ensemble by optimising diversity.

Tan et al. [92] used the Pareto archived evolution strategy (PAES) to optimise NNs for dealing with game problems. The game score (accuracy) and the number of hidden neurones (complexity) were used as the two objectives in the multi-objective generation of NNs for constructing ensembles. The WTA was also employed to make the final decision. Their results show significant improvement in performance over single NNs.

Pangilinan and Janssens [93] used the size of the decision tree as the complexity factor, and either NSGA-II or SPEA2 is used to optimise the objectives. By doing so, authors successfully found accurate classifiers that were not too complex and better generalisation was achieved.

Most recently, Tan et al. [94] used three objectives in multi-objective ensemble generation, of which two are accuracy measures, i.e., specificity and sensitivity for imbalanced classification problems, and the third is complexity measure, i.e., the number of input features. Three members having the minimal generational distance (GD) value are selected from the Pareto optimal set for constructing the ensemble.
Other objectives and methods

While most existing multi-objective ensemble approaches involve only two objectives, some researchers have proposed to use more objectives. For instance, Trawiński et al. [95] attempted to consider accuracy, complexity and diversity at the same time. The accuracy is a triplet of training error, error margin, and classification margin; the complexity is the number of classifiers in ensemble; and the diversity is difficulty measure. Their results show that the proposed approach has successfully achieved a high accuracy rate as well as a good accuracy-complexity trade-off on highly complex data.

While above methods measure performance on the training set (labelled data) in supervised learning, they can also be applied on purely unlabelled data in unsupervised learning, or even bring testing set (unlabelled data) into consideration, in SSL. It is possible to find a suitable group of unlabelled data using clustering algorithms, and if partially labelled data is available, each group of unlabelled data can be assigned to a class that matches the labelled data’s characteristics [96]. However, there could be an arbitrary amount of groups on unlabelled data that meets the optimisation criteria. Therefore, specific measures such as cluster deviation and cluster connectedness [97], etc, have to be used as ensemble objectives.

2.5 Member selection for multi-objective ensembles

Most early ensemble generation methods combine all trained base learners for constructing ensembles. However, as suggested in [98,99], it could be beneficial to ensemble a subset of base learners to gain better generalisation. Therefore,
selective ensemble, also known as ensemble pruning may further improve the ensemble performance.

### 2.5.1 Single-objective approaches

In single-objective approaches, a predefined objective is used to select a subset of ensemble members for final fusion. Different objectives have been identified in past researches.

Zhou et al. [100] proposed an approach termed genetic algorithm based selective ensemble (GASEN), where the main idea is to select a subset of members by evolving the weights that represent each member’s contribution to the whole ensemble. GASEN uses a validation set to estimate the generalisation performance and the objective of this algorithm is to minimise the generalisation error on the validation set.

Li et al. [14] proposed a diversity regularised ensemble pruning (DREP) method, which focuses on maximising diversity among pruned ensemble members. They split the training data into two subsets, one for training and the other for validation. Candidate ensemble members are trained with the training set sampled by Bagging, and then pruned with parameters determined on the validation set. The performance is then being evaluated on testing set and the results show significant improvement compared to other pruning methods.

Ruta and Gabrys [101] proposed to use either GA, tabu search, or population-based incremental learning to select ensemble members from a large number of candidates by assigning a members’ weighing to ‘0’ (exclude) or ‘1’ (include). Each member outputs crisp labels and the decision is made by majority voting. The result shows that using evolutionary-based selection could give comparable accuracy to the best found by an exhaustive search, without risking a dramatic loss of generalisation.
Gabry and Ruta [102] further proposed to use multidimensional genetic algorithm (MGA) to optimise the selection of features, classifiers and fusion methods in order to find the best combination of these parameters. Their results show signification improvement over single classifier and soft linear fusion methods on the test databases.

No matter whether the members are generated by SOO, an MOEA can always be applied to selecting a subset of members for ensemble construction. Brochero et al. [103] proposed to use NSGA-II to select a subset of 48 members out of 800 candidate hydrological models on streamflow forecasting problems, where the objectives are ignorance score for penalising bias severely, and reliability diagram that is similar to variance.

2.5.2 Multi-objective approaches

In the multi-objective approach, a set of Pareto optimal base learners will be generated, which are then used to form the final ensemble. Many researchers focused on how to combine members selectively from the set using some criterions rather than using the entire Pareto optimal set. The main purpose of selecting a subset of members is to maximise the generalisation performance of the ensemble.

Bhowan et al. [104] compared two methods of pruning ensemble members from candidate solutions, which are fitness-based or use genetic programming (GP) to evolve a composite voting tree. The former selects \( N \) fittest members that have the best fitness values, where \( N \) is the ensemble size. It is worth mentioning that theoretically all solutions on the Pareto front are of equal quality, therefore, none of them could be considered as better than others. However, as they are handling imbalance learning problems and have used accuracy on both the majority and minority classes as two objectives, the solutions can be ranked according to the average of both objectives. However, as this method uses a linear ordering of the
fittest $N$ solutions for ensemble solution, it cannot guarantee that the overlap of common errors on the selected members is minimal with respect to each other. Therefore, they proposed the latter method that focuses on promoting the diversity among selected members. Their empirical results confirm that the second method outperforms the first one. Their results also show that smaller ensembles may be better than larger ones.

Smith and Jin [105] compared four methods of selecting a subset of Pareto optimal members for ensemble generation. The four methods include selecting all solutions having an error smaller than a given threshold, selecting members based on the training error, selecting solutions near the knee point, and selecting solutions having the largest degree of diversity. They found that selecting a subset of members near the knee point or having the largest degree of diversity might be preferable although the performance of the four compared methods is not considerably different.

### 2.6 Semi-supervised learning

Traditionally, supervised learning uses labelled data to build a model. The amount of labelled training data is generally required to be large to build accurate classifiers, see Fig. 2.2a. However, in many real-world applications, obtaining labelled data is difficult, tedious, expensive, or time-consuming, due to the required human effort. By contrast, in many practical applications such as medical diagnosis [106], image classification [107], speech recognition [108], email categorisation [109], or text document classification [110], there is often a large amount of unlabelled data readily available. SSL addresses this issue by allowing the classification model to integrate part or all unlabelled data in its learning process, see Fig. 2.2b. The aim of using SSL is to maximise the generalisation performance through artificially labelled samples from unlabelled data while min-
imising human effort.

(a) Supervised learning

(b) Semi-supervised learning

**Figure 2.2:** Illustration of supervised vs semi-supervised learning [111].

SSL, together with transductive learning and active learning, are three main paradigms in machine learning literature. To clarify the confusion, the differences among them are: SSL attempts to take advantages of unlabelled data for supervised or unsupervised learning, while transductive learning assumes the unlabelled samples are the test samples, in other words, the test set is known and the goal of transductive learning is to maximize the classification performance on the given test set. Active learning assumes the algorithm has control on selecting input training data so that it asks an expert to be involved in labelling these selective training samples that assumed or known to be important. Here we only focus on SSL algorithm on classification tasks.
There are five categories of semi-supervised classification algorithms, which are:

1. Self-training [112]
2. Semi-supervised learning with generative models [107, 110, 113]
3. Semi-supervised support vector machines (S3VMs) [114–117]
4. Semi-supervised learning with graphs [118–120]
5. Semi-supervised learning with committees [106, 109, 121–124]

2.6.1 Self-training

Self-training initially constructs a single classifier with the labelled data. Most confidently predicted unlabelled data are then iteratively added to the training set, with its predicted label, and used along with previously augmented training set to refine the classifier. This process repeats a given number of iterations or until the convergence criteria are satisfied.

The classification error can be reduced if and only if the classifiers during the training process can predict most unlabelled samples correctly. However, there is no mechanism to avoid mislabelled noises being added to the training set, nor to remove such noise. Practically, the number of mislabelled samples are controlled with more accurate confidence measures with a predefined confidence threshold.

2.6.2 Semi-supervised learning with generative models

Generative models assume that samples with and without labels have the same parametric model where the number of components, prior $p(y)$ and conditional $p(x|y)$ are all correctly known. The unlabelled samples are classified using the learned model’s mixture components that are associated to each class [110, 125].
The drawback on generative models is that, when the size of labelled samples is very small, the model assumption may incorrect. By fitting a large amount of unlabelled data to this incorrect model will result in performance degradation [126]. One way to overcome this issue is to carefully build the generative model [127], e.g., to build multiple Gaussian components for each class.

2.6.3 Semi-supervised support vector machines (S3VMs)

The idea of S3VMs is to use the large amount of unlabelled data to adjust the decision boundary that is built by a small amount of labelled data, such that the boundary goes through the low-density regions. The optimal decision boundary supposes to cross the low-density region so that the training error on both labelled and unlabelled data can be minimised [114,115].

S3VM sometimes also been referred as transductive SVM which assumes a large margin separates the unlabeled data from each class. However, this method does not work if this assumption fails to be fulfilled.

2.6.4 Semi-supervised learning with graphs

In graph-based SSLs, a graph is constructed with nodes and edges, where each node represents a training sample regardless whether been labelled, and edges are weights corresponds to the similarity between the respective samples. The target of this algorithm is to find the minimum cut of the graph that nodes in each connected components have the same label [128]. There are a few variants of this methods, e.g., adding random noise to the edge weights and the unlabelled samples are predicted by majority voting [129], or by using a continuous predictive function [130] rather than original discrete predictive function to assign the possible labels to each unlabelled sample. The assumption of graph-based
methods is that the samples connected by strong edges are more likely to be in the same class and vice versa [127].

2.6.5 Semi-supervised learning with committees

Semi-supervised learning with committees (SSLC) is the largest branch of SSL classifications, which is a committee-based SSL that constructs an ensemble of diverse and accurate classifiers to exploit unlabelled samples and maintain a large diversity among them.

SSLCs can be categorised into two categories, by the number of ‘views’, which are multiple-view SSLCs and single-view SSLCs. A ‘view’ in SSLC describes an independent input space which is a description of an instance. Each ‘view’ is obtained by different physical sources or is derived by different feature extraction method. The ‘views’ give discriminative information about the data that any single ‘view’ should be sufficient to build an accurate classifier.

The origin of SSLCs can be seen by the co-training algorithm proposed by Blum and Mitchell [121] which is a multi-view SSLC. A successful co-training classifier must meet two strong assumptions that two sets of features should be conditionally independent and either of them is sufficient to build an accurate classification model. Subsequent research [131] found it sensitive to the independency requirement. Nevertheless, it has been successfully applied to many applications, such as email classification [109] and visual detectors for surveillance video [132].

However, not many real-world applications come with different ‘views’, therefore, researchers are interested in artificially splitting one ‘view’ into two. Literatures can be seen in [131,133,134].

Due to the difficulties in splitting ‘views’ for co-training like algorithms, many studies [106,122,123,135–138] have investigated the possibilities on a single ‘view’ without splitting it.
3.1 Introduction

A brain-computer interface (BCI) is a communication system that requires no peripheral muscular activity [139]. BCI systems allow subjects to send commands to electronic devices only by the brain activity [140]. The BCI systems are operated by identifying different brain activity patterns from subjects, and then translate them into commands. Most existing BCI systems rely on classification algorithms [141] to identify brain activity patterns.

A BCI system typically consists of five components: signal acquisition, feature extraction, feature translation (classification), the output device and operating protocol [139]. The input to BCI systems are signals recorded from the scalps or
the surface of the brain using either non-invasive (e.g. EEG, functional magnetic resonance imagery (fMRI)) or invasive devices (e.g. intra-cortical) [142]. Before these signals can be used for detecting the brain signals of interest, features must be extracted. The objectives of feature extraction are to reduce the dimension of the signal and to distinguish the differences between classes. Extracted features can be temporal, spatial, or spatio-temporal [141,143]. In addition, other preprocessing techniques such as artefact removal and feature selection (use a subset of the features) are also needed before classification.

A wide range of classification techniques, including LDA, NN, non-linear Bayesian classifier, $k$NN, and SVMs have been adopted for EEG signal classification, among which LDA and SVMs are two most popular ones [144].

It is often difficult to build a good single classifier if the dimensionality of the data is very high and the training set is comparatively small. Normally the classifiers built on a training set smaller than the data dimensionality are biassed and are very likely to have large variances as a result of insufficient estimation of related parameters [145]. These classifiers often perform poorly and unreliably on unseen data. To address this problem, ensemble techniques, which combine some single classifiers, have been developed to improve classification performance [145].

Most ensembles use the same type of base learning models obtained by manipulating the training data [146,147] or including additional objectives in learning to explicitly encourage diversity [88] or complexity [11]. Ensembles of the same type (but can be slightly different in structure) of base models are termed homogeneous ensembles. Recently, heterogeneous ensembles [148], where the base classifiers are different types, have been investigated to enhance the performance of the classifier ensembles.

This work proposes to construct heterogeneous classifier ensembles for EEG signal classification, where the base classifiers use different types of models and different input features as well. We hypothesise that such heterogeneous ensembles can
have a higher degree of diversity as they use completely different input features. This hypothesis has been empirically verified on the EEG datasets studied in this work.

The remainder of the chapter is organised as follows. Section 3.2 depicts the algorithms used for extracting features from the datasets. Section 3.3 describes the base classifiers as well as heterogeneous ensembles using same and different features. Section 3.4 presents the experimental results, and finally, Section 3.5 summarises this chapter.

### 3.2 Feature extraction

In this work, we extract features from original data by two steps, which are using common spatial pattern (CSP) to emphasise the difference between classes and then apply either autoregressive model (AR) or mean-variance statistics (MV) to extract the features from the time-series data.

#### 3.2.1 Common spatial pattern (CSP)

CSP projects multichannel EEG signals into subspace, where the differences between the two classes are highlighted and the similarities are minimised. The aim of CSP is to make the subsequent classification more effective, by designing a spatial filter that transforms the input data into output data with an optimal variance for subsequent discrimination [149]. CSP calculates the normalised spatial covariance $C$ from the input data $X$, which is an $N \times T$ matrix of a single trial, where $N$ is number of channels and $T$ is the number of samples per channel, by means of [150]

$$C = \frac{XX'}{\text{trace}(XX')}$$

(3.1)
where \( \cdot \)' denotes the transpose operation and \( \text{trace}(x) \) is the sum of the diagonal elements of \( x \).

For each class to be separated, the spatial covariance \( \overline{C}_d, d \in \{1, 2\} \) is calculated by averaging over the trials of each class. The composite spatial covariance is given as

\[
C_0 = \overline{C}_1 + \overline{C}_2 = U_0 \Sigma U_0',
\]

where \( U_0 \) is the matrix of eigenvectors and \( \Sigma \) is the diagonal matrix of eigenvalues sorted in descending order. The whitening transformation

\[
P = \sqrt{\Sigma^{-1}} U_0'
\]

equalises the variance in the space spanned by \( U_0 \), i.e., all eigenvalues of \( PC_0 P' \) are equal to one. If \( \overline{C}_1 \) and \( \overline{C}_2 \) are transformed into average covariance matrices as

\[
S_1 = PC_1 P' \quad \text{and} \quad S_2 = PC_2 P',
\]

then \( S_1 \) and \( S_2 \) share common eigenvectors, i.e.,

\[
S_1 = B \Sigma_1 B', \quad S_2 = B \Sigma_2 B', \quad \Sigma_1 + \Sigma_2 = I,
\]

where \( I \) is the identity matrix.

Since the sum of two corresponding eigenvalues is always one, the eigenvector with largest eigenvalue for \( S_1 \) has the smallest eigenvalue for \( S_2 \), and vice versa. This property makes the eigenvector \( B \) useful for classification of two groups. The projection of whitened signal onto the first and last eigenvector in \( B \), i.e., the eigenvector corresponding to the largest \( \Sigma_1 \) and \( \Sigma_2 \), are optimal for separating variance in two groups. The projection matrix \( W \) is denoted as

\[
W = (B' P)'.
\]
With the projection matrix $W$, the decomposition of trial $X$ is given as

$$Z = WX. \quad (3.7)$$

$Z$ can be seen as EEG source components including common and specific components of different classes. The original EEG signal $X$ can be reconstructed by

$$X = W^{-1}Z, \quad (3.8)$$

where $W^{-1}$ is the inverse of matrix $W$.

The columns of $W^{-1}$ are the CSP and can be considered as time-invariant EEG source distribution vectors.

By decomposing the EEG according to (3.7), the features used for classification can be obtained.

### 3.2.2 Autoregressive model (AR)

The autoregressive model (AR) can be denoted as

$$Y_t = \sum_{i=1}^{p} w_{t-i}Y_{t-i} + e, \quad (3.9)$$

where $Y_t$ is the predicted signal at time $t$, $w$ is a vector of $p$ coefficients (AR order) and $e$ is the prediction error [151].

This modelling requires a selection of data duration and a model order. The selection of these parameters varies from study to study. For example, Schlögle et al. [152] used an AR of an order of 3, Ince et al. [153] used AR order of 6. The data duration is the time of EEG epoch used for each spectral estimate. This parameter was also varied considerably seen in the literature [154,155].

41
3.2.3 Mean-variance statistics (MV)

As an alternative to AR feature extraction, we used MV features as different types of features for heterogeneous ensembles. The features are selected by calculating the average and variance of each CSP channel gathered in the steps of Section 3.2.1.

3.3 Heterogeneous classifier ensembles

EEG signal data are often noisy and are likely to have outliers. According to Lotte et al. [144], LDA and SVMs have shown to be effective and successful for EEG signal classification. Hence, we chose LDA, LSVM and RBF-SVM as the base classifiers in this work.

3.3.1 Single classifiers

The workflow of the single classifiers is pretty straightforward. Original data, which contains time-series recordings on all electrodes are used. A subset of the data, which only contains signals from electrodes that are relevant to motor imaginary tasks are used. The data is firstly passed through a bandpass filter to retain waveforms between 3 to 28 Hz, and those filtered signals are transformed by CSP. By applying AR or MV on each epoch of the former result, the features can be extracted. Those extracted features are then used to train the classifier and predict unseen data.

The structure of single classifiers is illustrated in Fig. 3.1.
3.3.2 Heterogeneous ensembles with the same feature

In feature extraction, we have obtained two different types of features, one based on AR and the other, MV. Furthermore, three different types of base classifiers are used to build heterogeneous classifier ensembles, which are LDA, LSVM (Linsvm) and RBF-SVM. However, in this setup, all basic classifiers use the same feature, i.e., either AR or MV, which can be graphically seen in Fig. 3.2. The output of each base classifier is weighted using its training accuracy calculated from CV, as we assume there is a positive correlation between the accuracy on training data (based on CV) and test data. This simple weighting method has shown to work well in our work.

The sum of accuracy of training set $S$ is denoted as

$$S = \sum_{i=1}^{n} acc_i,$$  \hspace{1cm} (3.10)

where $i$ is the $i$-th base classifier, $n$ is the number of base classifiers and $acc_i$ is the accuracy of the training set by the $i$-th classifier based on CV.

The label on the test data is estimated by

$$y_j = \begin{cases} 0 & \text{if} \frac{1}{S} \sum_{i=1}^{n} acc_i y_{i,j} < 0.5 \\ 1 & \text{otherwise} \end{cases},$$  \hspace{1cm} (3.11)

where $y_{i,j}$ denotes the label prediction of $j$-th sample of test set and $y_j$ is the ensemble prediction.
3.3.3 Heterogeneous ensembles with different features

The main difference in this setup from the previous one is that the base classifiers in this setup use two different features. As a result, we can construct an ensemble using all six base classifiers (three different models combined with two different features) or only choose three base classifiers that have the highest accuracies on the training data. The structure of this method can be seen in Fig. 3.3. The dashed arrows denote that the ensemble classifier will use the results of either six or three base classifiers’ depending on the configuration.
3.4 Experiments

3.4.1 Datasets and pre-processing

The datasets we used are EEG recordings collected by Fraunhofer FIRST, Intelligent Data Analysis Group, and Campus Benjamin Franklin of the Charit - University Medicine Berlin, Department of Neurology, Neurophysics Group. The aim was to discriminate trials of different motor imagery tasks. Data were recorded from five healthy subjects who sat in comfortable chairs with arms resting on armrests. Visual cues were indicated on the screen for 3.5 seconds when subjects were asked to perform three motor imagery tasks: (L)eft hand, (R)ight hand or right (F)oot. The presentation of target cues was alternated by periods of 1.75 to 2.25 seconds, in which subject could relax. We used all five datasets for examining the performance of various classifiers.

The data were recorded using a 128-channel Ag/AgCl electrode cap from ECI and amplified using BrainAmp amplifiers. The 118 EEG channels were measured at positions of the extended international 10/20-system. Signals were band-pass filtered between 0.05 and 200 Hz and then digitised at 1000 Hz with 16 bit (0.1 µV) accuracy. All data are down-sampled at 100 Hz (by picking each 10th sample) for analysis. Each dataset collected from five different subjects, which are named a, al, av, aw and ay, respectively, have 280 trials in total, whereas labelled and unlabelled counts are rated 168:112, 224:56, 84:196, 56:224 and 28:252, respectively. True labels of the unlabelled samples are also provided by the group (after the competition), which we used as test data for examining the performance of the classifiers. There are only two different tasks, e.g. (R)ight hand and (F)oot, in each dataset, hence, this work focuses on binary classification.

Since we are focusing on sensorimotor detection, only frontal and parietal chan-

\footnote{Available at http://bbci.de/competition/iii/desc_IVa.html}
nels [150, 156] from the recordings are selected to be used for classification, whose electrodes are named as F3, F1, Fz, F2, F4, FC5, FC3, FC1, FCz, FC2, FC4, FC6, C5, C3, C1, Cz, C2, C4, C6, CP5, CP3, CP1, CPz, CP2, CP4, CP6, O1 and O2 on channel 6 to 20, 33 to 39, 51 to 57, 69 to 75, 112 and 114 in the datasets, respectively. This will reduce the dimension from 118 to 28.

We applied a band-pass filter of 3 to 28 Hz on the selected channels to extract the theta, alpha and beta waves that have been successfully used for motor imaginary detection [157]. After this, we split the filtered recordings into the number of trials epochs on each dataset, and each epoch contains 0 to 3.5 second data as the description of the dataset. Processed data are then projected by CSP method, to enlarge the difference between classes. Finally, AR (with order = 6) or MV are used to extract features from the time-series.

We examined the performance of individual classifiers and both types of heterogeneous ensembles on all these five datasets. The classifier models in this experiment are LDA, LSVM and RBF-SVM where $\sigma = 10$. To ensure a fair comparison of the classifier ensembles, we conducted three different sets of tests on each dataset, which are 10-fold CV on the training set, train the classifiers on the training set then validate on the test set and 10-fold CV on the entire dataset. The reason of using 10-fold CV is because it is the accuracy estimation that was found to perform best in [158].

The results are shown in the following section.

### 3.4.2 Experimental results

The first test, 10-fold CV on the training set, is aimed to prove the improvement on accuracy of classifier ensembles over the individual base classifiers. We ran ten simulations for each setup and recorded the average results in Tables 3.1 to 3.5. We can see that the classification rates of the single base classifiers vary
significantly from subject to subject. However, the classifier ensembles produce results that are more accurate. We can also see that heterogeneous ensemble with the different feature (i.e., ‘Ensemble - Pick All’ and ‘Ensemble - Pick 3’ in the tables) outperform heterogeneous ensembles having the same feature. For further comparison, we found that ensembles composed by three best base classifiers having the highest accuracy on training data exhibit the best performance.

Table 3.1: 10-fold on training set - dataset aa

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA</td>
<td>92.95±0.15</td>
<td>89.85±0.72</td>
</tr>
<tr>
<td>AR-Lin-SVM</td>
<td>92.22±0.14</td>
<td>90.57±0.85</td>
</tr>
<tr>
<td>AR-RBF-SVM</td>
<td>72.67±0.36</td>
<td>68.70±1.00</td>
</tr>
<tr>
<td>MV-LDA</td>
<td>87.16±0.15</td>
<td>85.59±0.38</td>
</tr>
<tr>
<td>MV-Lin-SVM</td>
<td>89.19±0.25</td>
<td>86.94±1.17</td>
</tr>
<tr>
<td>MV-RBF-SVM</td>
<td>85.68±0.22</td>
<td>84.64±0.89</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td>-</td>
<td>90.27±1.03</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td>-</td>
<td>85.59±0.82</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td>-</td>
<td>90.88±1.36</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td>-</td>
<td><strong>93.07±1.09</strong></td>
</tr>
</tbody>
</table>

Difference - +2.18

Table 3.2: 10-fold on training set - dataset al

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA</td>
<td>96.84±0.09</td>
<td>95.89±0.34</td>
</tr>
<tr>
<td>AR-Lin-SVM</td>
<td>96.27±0.08</td>
<td>96.03±0.25</td>
</tr>
<tr>
<td>AR-RBF-SVM</td>
<td>88.91±0.13</td>
<td>88.28±0.44</td>
</tr>
<tr>
<td>MV-LDA</td>
<td>94.18±0.11</td>
<td>93.63±0.36</td>
</tr>
<tr>
<td>MV-Lin-SVM</td>
<td>95.72±0.06</td>
<td>95.31±0.44</td>
</tr>
<tr>
<td>MV-RBF-SVM</td>
<td>92.73±0.09</td>
<td>92.55±0.31</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td>-</td>
<td>95.67±0.20</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td>-</td>
<td>93.36±0.33</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td>-</td>
<td>96.78±0.72</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td>-</td>
<td><strong>97.32±0.52</strong></td>
</tr>
</tbody>
</table>

Difference - +0.53

In the second set of tests, the classifiers are trained on the training set and then validated on the test set. These tests are aimed to compare the difference between ‘Pick All’ and ‘Pick 3’ methods for heterogeneous ensemble classifiers on different features. The results are presented in Table 3.6. We can see that ‘Pick 3’ gives higher classification rates except on dataset al. It is probably because the
classification rate on dataset $al$ is relatively high hence the improvement would not be as obvious as other datasets.

The last test, 10-fold CV on the entire dataset, is aimed to further prove the improvement on classification rate of ensemble classifiers than single classifiers by
Table 3.6: ‘Pick All’ and ‘Pick 3’ heterogeneous ensemble

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pick All</th>
<th>Pick 3</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>78.57%</td>
<td>81.25%</td>
<td>+2.68%</td>
</tr>
<tr>
<td>al</td>
<td>94.64%</td>
<td>94.64%</td>
<td>+0.00%</td>
</tr>
<tr>
<td>av</td>
<td>65.31%</td>
<td>67.35%</td>
<td>+2.04%</td>
</tr>
<tr>
<td>aw</td>
<td>56.25%</td>
<td>60.71%</td>
<td>+4.46%</td>
</tr>
<tr>
<td>ay</td>
<td>84.13%</td>
<td>88.10%</td>
<td>+3.97%</td>
</tr>
</tbody>
</table>

using the same size of the training set on each dataset. As the true labels are publicly accessible, we used the entire datasets with true labels as the training sets, and applied a 10-fold CV on them, ran ten times and recorded the average results in Tables 3.7 to 3.11. The results clearly show heterogeneous ensemble classifier on different features has better classification rates than on same features.

Table 3.7: 10-fold on entire set - dataset aa

<table>
<thead>
<tr>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA 89.25±0.12</td>
<td>87.61±0.73</td>
</tr>
<tr>
<td>AR-Lin-SVM 89.04±0.18</td>
<td>87.54±0.49</td>
</tr>
<tr>
<td>AR-RBF-SVM 78.39±0.13</td>
<td>74.93±1.04</td>
</tr>
<tr>
<td>MV-LDA 82.14±0.16</td>
<td>81.64±0.61</td>
</tr>
<tr>
<td>MV-Lin-SVM 83.63±0.17</td>
<td>82.04±0.56</td>
</tr>
<tr>
<td>MV-RBF-SVM 80.70±0.16</td>
<td>80.11±0.41</td>
</tr>
<tr>
<td>Ensemble - AR -</td>
<td>87.07±0.67</td>
</tr>
<tr>
<td>Ensemble - MV -</td>
<td>81.68±0.58</td>
</tr>
<tr>
<td>Ensemble - Pick All -</td>
<td>87.32±0.72</td>
</tr>
<tr>
<td>Ensemble - Pick 3 -</td>
<td><strong>88.54±0.64</strong></td>
</tr>
<tr>
<td>Difference -</td>
<td>+1.21</td>
</tr>
</tbody>
</table>

Table 3.8: 10-fold on entire set - dataset al

<table>
<thead>
<tr>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA 97.11±0.11</td>
<td>95.96±0.38</td>
</tr>
<tr>
<td>AR-Lin-SVM 96.34±0.10</td>
<td>95.32±0.31</td>
</tr>
<tr>
<td>AR-RBF-SVM 89.15±0.17</td>
<td>88.54±0.31</td>
</tr>
<tr>
<td>MV-LDA 93.43±0.07</td>
<td>93.00±0.25</td>
</tr>
<tr>
<td>MV-Lin-SVM 95.69±0.02</td>
<td>95.68±0.11</td>
</tr>
<tr>
<td>MV-RBF-SVM 93.13±0.05</td>
<td>93.00±0.25</td>
</tr>
<tr>
<td>Ensemble - AR -</td>
<td>95.43±0.50</td>
</tr>
<tr>
<td>Ensemble - MV -</td>
<td>93.29±0.28</td>
</tr>
<tr>
<td>Ensemble - Pick All -</td>
<td>96.93±0.48</td>
</tr>
<tr>
<td>Ensemble - Pick 3 -</td>
<td><strong>97.14±0.34</strong></td>
</tr>
<tr>
<td>Difference -</td>
<td>+0.21</td>
</tr>
</tbody>
</table>

To demonstrate the statistical significance of the performance difference between single classifiers and heterogeneous classifier ensembles, we performed a t-test on
### Table 3.9: 10-fold on entire set - dataset av

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA</td>
<td>74.01±0.18</td>
<td>71.39±1.14</td>
</tr>
<tr>
<td>AR-Lin-SVM</td>
<td>72.80±0.23</td>
<td>71.46±0.64</td>
</tr>
<tr>
<td>AR-RBF-SVM</td>
<td>71.71±0.27</td>
<td>67.21±1.44</td>
</tr>
<tr>
<td>MV-LDA</td>
<td>65.88±0.13</td>
<td>65.04±0.93</td>
</tr>
<tr>
<td>MV-Lin-SVM</td>
<td>67.00±0.30</td>
<td>66.14±0.82</td>
</tr>
<tr>
<td>MV-RBF-SVM</td>
<td>62.64±0.18</td>
<td>61.89±0.41</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td></td>
<td>71.79±0.77</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td></td>
<td>65.68±0.64</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td></td>
<td>71.29±0.63</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td></td>
<td>73.07±0.83</td>
</tr>
<tr>
<td>Difference</td>
<td>-</td>
<td>+1.79</td>
</tr>
</tbody>
</table>

### Table 3.10: 10-fold on entire set - dataset aw

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA</td>
<td>91.73±0.12</td>
<td>90.43±0.65</td>
</tr>
<tr>
<td>AR-Lin-SVM</td>
<td>82.89±0.13</td>
<td>80.96±0.75</td>
</tr>
<tr>
<td>AR-RBF-SVM</td>
<td>72.73±0.13</td>
<td>71.29±0.85</td>
</tr>
<tr>
<td>MV-LDA</td>
<td>81.46±0.10</td>
<td>81.32±0.34</td>
</tr>
<tr>
<td>MV-Lin-SVM</td>
<td>85.55±0.12</td>
<td>85.18±0.56</td>
</tr>
<tr>
<td>MV-RBF-SVM</td>
<td>82.92±0.15</td>
<td>82.21±0.41</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td></td>
<td>82.64±1.05</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td></td>
<td>82.57±0.41</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td></td>
<td>86.75±0.59</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td></td>
<td>89.54±0.61</td>
</tr>
<tr>
<td>Difference</td>
<td>-</td>
<td>+2.79</td>
</tr>
</tbody>
</table>

### Table 3.11: 10-fold on entire set - dataset ay

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA</td>
<td>88.13±0.15</td>
<td>86.11±0.85</td>
</tr>
<tr>
<td>AR-Lin-SVM</td>
<td>84.91±0.19</td>
<td>82.89±0.72</td>
</tr>
<tr>
<td>AR-RBF-SVM</td>
<td>79.60±0.12</td>
<td>77.04±0.65</td>
</tr>
<tr>
<td>MV-LDA</td>
<td>92.13±0.04</td>
<td>92.00±0.25</td>
</tr>
<tr>
<td>MV-Lin-SVM</td>
<td>92.86±0.09</td>
<td>92.36±0.51</td>
</tr>
<tr>
<td>MV-RBF-SVM</td>
<td>92.79±0.07</td>
<td>92.46±0.31</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td></td>
<td>84.71±1.04</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td></td>
<td>92.36±0.30</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td></td>
<td>93.14±0.47</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td></td>
<td>93.21±0.87</td>
</tr>
<tr>
<td>Difference</td>
<td>-</td>
<td>+0.07</td>
</tr>
</tbody>
</table>

The results. The null hypothesis $H_0$ is that classifier ensembles perform worse than single classifiers, i.e., $H_0: \mu_d \leq 0$, and the alternative hypothesis is $H_1: \mu_d > 0$. We used the average classification rates on 10-fold CV on the entire dataset for each dataset. The confidence level is set to 90%.
Table 3.12: T-test statistics for comparing classifiers ensemble and single classifiers

<table>
<thead>
<tr>
<th></th>
<th>SC1</th>
<th>SC2</th>
<th>SC3</th>
<th>SC4</th>
<th>SC5</th>
<th>SC6</th>
<th>EC1</th>
<th>EC2</th>
<th>EC3</th>
<th>EC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR-LDA (SC1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>AR-Lin-SVM (SC2)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>AR-RBF-SVM (SC3)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>MV-LDA (SC4)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>MV-Lin-SVM (SC5)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>MV-RBF-SVM (SC6)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Ensemble - AR</td>
<td>EC1</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Ensemble - MV</td>
<td>EC2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Ensemble - Pick All</td>
<td>EC3</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Ensemble - Pick 3</td>
<td>EC4</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

The results are shown in Table 3.12. ‘Y’ and ‘N’ indicates if the classifier in the row performs statistically more accurate than the classifier in the column. From the table, we can see that all classifier ensembles except the ensemble using AR, statistical outperform the examined single classifiers significantly. We also found that heterogeneous classifier ensembles using different features perform slightly better than those using the same feature. Finally, ensembles consisting of selected base classifiers perform even better than ensembles containing all base classifiers.

Table 3.13: Comparison of our result with literature [157]

<table>
<thead>
<tr>
<th>Contributor</th>
<th>aa</th>
<th>al</th>
<th>av</th>
<th>aw</th>
<th>ay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang</td>
<td>95.5%</td>
<td>100.0%</td>
<td>80.6%</td>
<td>100.0%</td>
<td>97.6%</td>
</tr>
<tr>
<td>Li</td>
<td>89.3%</td>
<td>98.2%</td>
<td>76.5%</td>
<td>92.4%</td>
<td>80.6%</td>
</tr>
<tr>
<td>Liu</td>
<td>82.1%</td>
<td>94.6%</td>
<td>70.4%</td>
<td>87.5%</td>
<td>88.1%</td>
</tr>
<tr>
<td>Zhou</td>
<td>83.9%</td>
<td>100.0%</td>
<td>63.3%</td>
<td>50.9%</td>
<td>88.1%</td>
</tr>
<tr>
<td>Bensch</td>
<td>73.2%</td>
<td>96.4%</td>
<td>70.4%</td>
<td>79.9%</td>
<td>50.8%</td>
</tr>
<tr>
<td>Simon</td>
<td>83.0%</td>
<td>91.1%</td>
<td>50.0%</td>
<td>87.9%</td>
<td>54.4%</td>
</tr>
<tr>
<td>Gysels</td>
<td>69.6%</td>
<td>96.4%</td>
<td>64.3%</td>
<td>69.6%</td>
<td>61.9%</td>
</tr>
<tr>
<td>Viduarre</td>
<td>66.1%</td>
<td>92.9%</td>
<td>67.3%</td>
<td>68.3%</td>
<td>50.4%</td>
</tr>
<tr>
<td>Song</td>
<td>66.1%</td>
<td>100.0%</td>
<td>63.3%</td>
<td>64.3%</td>
<td>54.4%</td>
</tr>
<tr>
<td>Arhabi</td>
<td>70.5%</td>
<td>94.6%</td>
<td>56.1%</td>
<td>55.8%</td>
<td>60.3%</td>
</tr>
<tr>
<td>Shahabi</td>
<td>57.1%</td>
<td>76.8%</td>
<td>57.7%</td>
<td>64.3%</td>
<td>54.0%</td>
</tr>
<tr>
<td>Yang</td>
<td>52.7%</td>
<td>85.7%</td>
<td>61.2%</td>
<td>51.8%</td>
<td>43.7%</td>
</tr>
<tr>
<td>Wang</td>
<td>50.9%</td>
<td>53.6%</td>
<td>54.6%</td>
<td>56.2%</td>
<td>46.0%</td>
</tr>
<tr>
<td>Yoon</td>
<td>50.0%</td>
<td>67.9%</td>
<td>52.6%</td>
<td>52.7%</td>
<td>45.6%</td>
</tr>
<tr>
<td>Proposed</td>
<td>81.3%</td>
<td>94.6%</td>
<td>67.4%</td>
<td>60.7%</td>
<td>88.1%</td>
</tr>
</tbody>
</table>

To further examine the performance of the heterogeneous classifier ensembles proposed in this work, we list all results submitted to the competition [157]. Comparing the changed classification rate testing accuracy listed in Table 3.13, our method performs better than many of the submitted methods on the same
dataset, however, is not the best. One potential reason is that in our work, no specific techniques have been adopted to deal with class imbalance [159] in the data.

### 3.5 Summary

In this chapter, we compared the performance of classifier ensembles with that of single classifiers in EEG signal classification. The results show that by using classifier ensembles, we can achieve a higher classification accuracy and a lower standard deviation on testing data than single classifiers can. However, different types of ensembles have different degrees of improvement than single classifiers. Heterogeneous classifier ensembles are found to outperform homogeneous ones, which could because the increased diversity. Furthermore, if only a few base learners that having the highest training accuracies are used to generate the ensemble, the testing accuracy could be higher than those with all base learners been used. This confirms the finds in [100].

For the particular setup used in this chapter, it is apparent that the smaller ensembles have higher training accuracies but lower diversities than larger ones, and the former one performs better than the latter. This seems not compatible with the findings that heterogeneous ensembles outperform homogeneous ones due to higher diversity in the former. However, previous studies have pointed out that the accuracy and diversity can conflict and influence the generalisation. Therefore, a trade-off is essential for building the best ensemble, which is presented in the next chapter.
CHAPTER 4

Generating Diverse and Accurate Classifier Ensembles Using Multi-Objective Optimisation

4.1 Introduction

It has been shown, both theoretically and empirically, that classifier ensembles, which consist of multiple classifiers can perform better than single classifiers [5,6]. Usually, an ensemble is built in two steps, where the first step is to train a set of classifier members (also known as base classifiers) for a given task, and the second step is to combine these classifier members for the final prediction. In the previous chapter, we found that both the accuracy of each ensemble member and the diversity among ensemble members are important for the overall performance of classifier ensembles, which similar conclusions can be seen in [8,9].

Brown [160] categorised existing methods for creating diverse ensembles into four
groups, namely: supplying different training data, employing different learning algorithms, and initialising learning models with different weights or structures. Depending on whether the diversity is taken into account during ensemble construction, ensemble methods can be generally divided into two types, i.e., explicit and implicit methods [160]. For example, Bagging [44] is categorised as an implicit method, where it randomly samples training data to generate different training sets for each ensemble member, without measuring or ensuring diversity. Boosting [161] is an explicit method, that manipulates the probability of selecting training data from the original training set in order to enhance diversity.

Since accuracy and diversity are very likely to conflict with each other, it is difficult to maximise both objectives at the same time, meaning that creating accurate and diverse ensembles is essentially a MOP [11, 13]. To address this issue, previous work has aggregated multiple objectives into a scalar objective function [35, 162] and solved the problem with single-objective EA. However, this requires the user to specify the hyper-parameter before learning, such that only one single solution can be found. As suggested in [11], using the Pareto-based multi-objective learning approach is the more natural way to solve this problem.

Regardless of how diversity is created, methods for building ensembles can be divided into two steps [14, 15, 89, 105, 163, 164], i.e., first train multiple classifiers and then select a subset of them for constructing the ensemble. Theoretically, diversity should be measured among all members of an ensemble, which cannot be guaranteed if the ensemble members are selected one by one from a group of potential base classifiers. In order to address this issue, we propose to combine the member generation and selection steps, where all members of an ensemble are generated simultaneously using an MOEA, to optimise the accuracies of ensemble members and find groups of them, which have maximum diversity, at the same time. The main benefit of this approach is that diversity of the final solution can be accurately measured and the trade-off between accuracy and diversity of the
whole ensemble can be taken into account during ensemble generation.

The remainder of the chapter is organised as follows. Section 4.2 details the proposed generation method. Section 4.3 presents the experiments and Section 4.4 summarises this chapter.

4.2 Proposed method

The main idea of the proposed method is to generate all members of an ensemble simultaneously by maximising members' accuracy and diversity of the whole ensemble using an MOEA. In the following, we provide a brief introduction to the objectives, including accuracy and diversity measures, the EA, and the approach adopted in this work for creating ensemble diversity.

4.2.1 Objectives

Accuracy

The first concern of building a successful ensemble is to ensure all base classifiers are accurate. The most suitable accuracy measure is to find the proportion of test data that has been correctly predicted. However, as the labels of testing data are naturally unknown during training, there is no way to measure the accuracy in this way. By assuming the data distributions on test and training sets are the same or similar, we can use the classification rate on the training set to measure the accuracy of the ensemble.

We propose to combine the predictions of base classifiers by majority voting. Denote the prediction of $i$-th pattern by $j$-th base classifier as $y_{i,j}$, and the final
prediction of $i$-th pattern $y_i$ that is defined as:

$$y_i = \begin{cases} 
0 & \text{if } \frac{1}{L} \sum_{j=1}^{L} y_{i,j} < 0.5 \\
1 & \text{otherwise}
\end{cases} \quad (4.1)$$

where $L$ denotes the number of base classifiers. We assume $L$ is odd, and that the classes are labelled as (or by transcoding them into) ‘0’$'$s and ‘1’$'$s for both classes.

The ensemble accuracy is calculated by $P_i$, which is denoted as:

$$acc = \frac{\sum_{i=1}^{N} P_i(+) }{N}, \quad (4.2)$$

where $P_i(+) \text{ denotes if } i$-th pattern is correctly classified and $N$ denotes the number of patterns.

**Diversity**

The diversity among ensemble members could be the key to a successful classifier ensemble. In this research, we use three different measures to measure ensemble diversity, which are coincident failure diversity (CFD) [22,23], disagreement (DIS) [22,23] and hamming distance (HD) [165] measures.

Details of these measures are described in Section 2.3.2.

**4.2.2 Diversity creation**

Brown concluded four main categories of methods for creating diverse ensemble members in [160]. In our proposed method, we manipulate the training data to create the diverse members. Inspired by [166,167] which manipulate the feature distribution, and Bagging [161] which manipulates the data distribution, we
create diversity either by selecting a subset of features or a subset of training
patterns. We will empirically evaluate the effectiveness of the two approaches in
the next section.

4.2.3 Evolutionary algorithm

In this work, generation of accurate and diverse ensembles can be formulated as
the following bi-objective optimisation problem:

\[
\begin{align*}
\text{max} & \quad \{f_1, f_2\} \\
 f_1 &= acc \\
 f_2 &= div^*,
\end{align*}
\]

where \(div^*\) denotes any of the diversity measures.

An MOEA can be used to achieve a set of non-dominated optimal solutions. The
non-dominated solution set is known as the Pareto set in the decision space and
forms the Pareto front in the objective space [11]. In this work, we adopted the
elitist non-dominated sorting genetic algorithm (NSGA-II), a popular MOEA to
optimise the two objectives. Details about the NSGA-II algorithm can be found
in [61].

We encode the selection of features or training patterns using a binary string,
where each bit denotes whether the corresponding feature or training pattern
will be selected. Thus, the length of the chromosome is equal to the number of
features or training patterns, and the number of chromosomes in each individual
equals the number of base classifiers in the ensemble.

The EA aims to evolve the chromosomes to find the optimal subset of features
or training data to maximise both objectives.
The main steps of the proposed multi-objective approach to generating accurate and diverse ensembles using NSGA-II are as follows:

**Step 1:** Generate an initial population $P_{t=0}$ of given size $D$ by randomly initialising each individual’s chromosome.

**Step 2:** Evaluate each individual in the population $P_t$.

   2a) Decode the selected training data from the chromosome.

   2b) Train the classifiers with the selected training data.

   2c) Compare the prediction with true labels and calculate both objectives.

**Step 3:** Repeat the following steps until the termination condition is satisfied.

   3a) Use non-dominated sorting to assign a front number to all solutions and calculate the crowding distance for all non-extreme solution.

   3b) Use binary tournament selection, recombination and mutation to generate an offspring population $Q_t$ of the same size $D$ from $P_t$.

   3c) Evaluate $Q_t$ as listed in **Step 2**.

   3d) Combine $P_t \cup Q_t \rightarrow R_t$, therefore, elitism is ensured.

   3e) Sort $R_t$ according to non-dominated sorting method.

   3f) Create new generation $P_{t+1}$ by picking up the first $D$ solutions from $R_t$

   3g) Increment the generation counter $t + 1 \rightarrow t$

**Step 4:** Use non-dominated sorting to find the Pareto optimal solutions of the combined population in the last generation.
4.3 Experiments

4.3.1 Experimental setup

We tested the performance of the proposed method using datasets from the UCI machine learning library [168]: German, Heart, Ionosphere and Monks-1,-2,-3. All datasets were prepared by removing patterns with missing values and normalising values of each feature to $\mu=0$ and $\sigma=1$ before the evaluation. The characteristics of the preprocessed datasets are summarised in Table 4.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Patterns</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>1000</td>
<td>24</td>
</tr>
<tr>
<td>Heart</td>
<td>297</td>
<td>13</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>33</td>
</tr>
<tr>
<td>Monks</td>
<td>432</td>
<td>7</td>
</tr>
</tbody>
</table>

Datasets German, Heart and Ionosphere do not have partition information, which is compulsory for the evaluation. Therefore, we randomly split the dataset into two halves for training and testing. The partition information exists on Monks-1,-2,-3 datasets, therefore we used the original partition to perform the experiment. The numbers of training patterns in Monks-1,-2,-3 are 124, 169 and 122, respectively.

The base classifiers are LSVM models and the diversity is created by learning either from datasets using different sub-feature set or subsets of the training patterns. The ensemble accuracies are compared with the classifier using same LSVM model but trained by the entire training set.

Both NSGA-II and LSVMs we used are provided in the Shark Machine Learning Library [169]. The regularisation parameter $C$ of the LSVM algorithm is pre-tuned per dataset with $n$-fold CV where $n=3$. The same optimal value is used
for single classifier and classifier ensembles training. Parameters of the NSGA-II algorithm are listed in Table 4.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ensemble member</td>
<td>9</td>
</tr>
<tr>
<td>Population size</td>
<td>500</td>
</tr>
<tr>
<td>Number of generations</td>
<td>500</td>
</tr>
<tr>
<td>Crossover points</td>
<td>2</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.6</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>$L_c^{-1}$</td>
</tr>
</tbody>
</table>

* $L_c$ denotes the length of chromosome

In the experiments, three diversity measures, i.e., CFD, DIS and HD, are used. Meanwhile, two diversity creation methods, i.e., using different features or using different training patterns are investigated. The combination of the three diversity measures with two diversity creation methods results in six different setups in total, which is listed in Table 4.3.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Diversity measure</th>
<th>Diversity creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD/SF</td>
<td>CFD</td>
<td>By supplying different sub features</td>
</tr>
<tr>
<td>CFD/SS</td>
<td>CFD</td>
<td>By supplying different sub patterns</td>
</tr>
<tr>
<td>DIS/SF</td>
<td>DIS</td>
<td>By supplying different sub features</td>
</tr>
<tr>
<td>DIS/SS</td>
<td>DIS</td>
<td>By supplying different sub patterns</td>
</tr>
<tr>
<td>HD/SF</td>
<td>HD</td>
<td>By supplying different sub features</td>
</tr>
<tr>
<td>HD/SS</td>
<td>HD</td>
<td>By supplying different sub patterns</td>
</tr>
</tbody>
</table>

4.3.2 Experimental results

Convergence analysis

We used hypervolume [73, 170] as the performance indicator to illustrate the convergence of the NSGA-II for multi-objective ensemble generation. In this
Figure 4.1: The hypervolume averaged over the six test cases for the six experimental setups.

work, as both objectives are normalised between 0 and 1, the reference point is set to $(0, 0)$ in all experiments.

In the experiments, once the hypervolume stops increasing for a certain number of generations, we can conclude that the learning has converged. Note, however, that the absolute accuracy and diversity vary across different datasets. Therefore the absolute hypervolume value cannot be used for comparing the results across different test problems. To address this issue, we linearly scaled the hypervolume from 0 to 1, so that it can indicate the degree of convergence of the learning process. The scaling is shown in (4.6) and the results can be found in Fig. 4.1.

$$HV'_i = (HV_i - HV_0)/(HV_{end} - HV_0), \quad (4.6)$$

where $HV_i$ denotes the hypervolume of $i$-th generation and $HV_{end}$ denotes the
result of the last generation.

Fig. 4.1 shows the mean and standard deviation of the hypervolume averaged over the six test problems. From these plots, we can see that in most scenarios, learning converges within 500 generations, while different diversity measures and creation methods show different converging speeds. It is easy to see that creating diversity by supplying different features (refer to the three plots in the left panels in Fig. 4.1) converges faster than by supplying different training patterns. This is because, in all test cases used in this work, the number of features is much smaller than that of training patterns. Therefore, searching for the optimal feature subsets takes much less time to converge than searching for the optimal training patterns. Furthermore, the convergence speed on the HD diversity measure seems slower than other diversity measures. We investigated this and found that the number of achieved Pareto solutions in this setup is significantly smaller than other diversity measures, which may make it more difficult for the MOEA to explore new solutions. This may also be the reason that caused a jump in the convergence in Fig. 4.1(e).

### Classification performance

**Table 4.4:** Test accuracies of single classifier and ensemble classifiers

<table>
<thead>
<tr>
<th></th>
<th>SC</th>
<th>CFD/SF (Δ%)</th>
<th>CFD/SS (Δ%)</th>
<th>DIS/SF (Δ%)</th>
<th>DIS/SS (Δ%)</th>
<th>HD/SF (Δ%)</th>
<th>HD/SS (Δ%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>0.7620</td>
<td>0.7600 (-0.26)</td>
<td>0.7540 (1.05)</td>
<td>0.7700 (1.05)</td>
<td>0.7540 (-1.05)</td>
<td>0.7500 (-1.57)</td>
<td><strong>0.7780 (2.10)</strong></td>
</tr>
<tr>
<td>Heart</td>
<td>0.7432</td>
<td>0.8416 (-13.64)</td>
<td>0.7635 (2.73)</td>
<td>0.8314 (15.55)</td>
<td>0.7808 (5.45)</td>
<td><strong>0.8716 (17.27)</strong></td>
<td>0.8108 (9.09)</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.8571</td>
<td>0.8800 (2.67)</td>
<td>0.8686 (1.33)</td>
<td>0.8686 (1.33)</td>
<td>0.8800 (2.67)</td>
<td>0.9029 (5.33)</td>
<td><strong>0.9143 (6.67)</strong></td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.7083</td>
<td><strong>0.7917 (11.76)</strong></td>
<td>0.7824 (10.46)</td>
<td>0.7639 (7.84)</td>
<td>0.6620 (-6.54)</td>
<td>0.7361 (3.92)</td>
<td>0.7130 (0.65)</td>
</tr>
<tr>
<td>Monks-2</td>
<td>0.5324</td>
<td>0.6806 (27.83)</td>
<td>0.7106 (31.48)</td>
<td>0.6806 (27.83)</td>
<td>0.6736 (26.52)</td>
<td>0.6676 (25.22)</td>
<td><strong>0.7130 (33.91)</strong></td>
</tr>
<tr>
<td>Monks-3</td>
<td>0.6481</td>
<td>0.8333 (28.57)</td>
<td>0.8310 (28.21)</td>
<td>0.8319 (31.43)</td>
<td>0.8380 (29.29)</td>
<td>0.8333 (28.57)</td>
<td><strong>0.8704 (34.29)</strong></td>
</tr>
</tbody>
</table>

Notations:

- **SC:** Single Classifier. Test accuracy of single classifier alone.
- **Δ:** Difference between ensemble method and single classifier.

The accuracies on the test data of single classifier and ensemble classifiers can be found in Table 4.4. For better visualisation, these results have also been plotted
Figure 4.2: Comparison of single classifier and ensemble classifiers

in Fig. 4.2.

Rows in Table 4.4 represent the results from different datasets while columns present the test accuracies of single classifier or best accuracy of classifier ensembles found in Pareto front of the final generation. The values in brackets show the accuracy improvement compared with single classifiers while negative values mean the performance degradation. The best method is highlighted with bold font.

Each group in Fig. 4.2 represents the results from a particular dataset as indicated at the bottom. The seven coloured bars in each group denote the test accuracy of the single classifier and those of six ensembles created using different methods. The results show that classifier ensembles consistently improve the classification performance on four datasets, namely Heart, Ionosphere, Monk-2 and Monk-3 while mixed results have been obtained on the German and Monk-1. Meanwhile, different diversity creation methods and measures give various accuracy improvements. This indicates that the optimal diversity creation method might be problem dependent.

In our experiments, 31 out of 36 ensemble setups improved classifier accuracy. In some datasets, i.e., Monks-2 and Monks-3, classifier ensembles dramatically improved the ensembles’ accuracies by over 30 per cent higher than the single classifiers. Only one setup, i.e., DIS/SS method on Monks-1 dataset had signif-
icantly degraded the classification performance. We conducted some additional investigations on this dataset and found that a few solutions that performed better than the single classifier were lost in later generations. This may indicate that this dataset is easy to overfit and additional measures for controlling overfitting are needed to ensure that a good performance on the training set will also lead to a good performance on the test set.

By comparing the performance among all experimental setups, we can see that combining the HD diversity measure in the input (feature) space with the use of subsets of the training patterns for creating ensemble members might be the preferred option for creating diverse yet accurate ensembles, as it produces the best result on five of the six test cases.

Figure 4.3: Pareto fronts achieved in the 36 different setups.

The Pareto fronts of the 36 setups are plotted in Fig. 4.3, where the results from
the same test case are plotted in the same row and results from the same ensemble generation method are plotted in the same column. The two axes of each subplot indicate the average training accuracy and the diversity measure, respectively. Each circle denotes a generated ensemble on Pareto front, where black circles indicate the solutions perform worse than the single classifier and the coloured circles indicate those solutions perform better than that. The colour of these circles varies smoothly from yellow to red, where colours closer to red indicate the testing accuracy is higher. Blue crosses in the plots mark the solutions that have highest test accuracies, where the bigger the crosses are, the higher the accuracy.

![Graphs](image)

**Figure 4.4:** Zoomed-in version of representative plots in Fig. 4.3

Fig. 4.3 shows that not all solutions on the Pareto front outperform the single classifier. In addition, most solutions with higher accuracy on the test data are distributed near the right end of the Pareto front, i.e., the solutions have higher classification accuracy on the training data, although the best solution is not necessarily the solution having the highest training accuracy. To make it easier
to read, Fig. 4.4 shows a zoomed-in version of some representative plots in Fig. 4.3. This is quite intuitive as the solutions having very high accuracy on the training data are more likely to overfit. However, these results suggest that if we do not have additional information for selecting solutions from the Pareto front, it is still better to select ensembles having a higher training accuracy. Some additional experiments have been carried out and the results indicate that selecting the top 10 per cent on the training data overall outperform others.

4.4 Summary

In this chapter, we proposed a multi-objective ensemble generation method that uses training accuracy and ensemble diversity as both explicit objective. By testing on three different diversity measures and two methods for creating ensemble diversity, we found a few methods perform better than others and single classifiers. The ensemble that consists diversity created by different training samples and measured on the feature space gave the highest testing accuracy in our experiment.

Although there are many good groups of ensembles can be found in the Pareto front, it is difficult to determine the best ensemble. However, most good ensembles are crowded around the area that has higher training accuracy, while in a few datasets they are near the high diversity end. This suggests that in most cases accuracy is more important while in others diversity dominates diversity, but the behaviour is dataset specific. A safer choice for selecting an ensemble from the Pareto optimal solutions is to pick one in the top 10 per cent of those solutions that having the highest training accuracy.

Another difficulty we found in this work is the computational cost. As EA-based methods involve a large number of fitness evaluations, and the fitness can only be measured by training the classifier models, the computational complexity
increases dramatically when the size of data increases, which seems to be an urge to find a solution that reduces the computational cost. We, therefore, tried to build a feature selection method, which is discussed in the next chapter.
Feature Selection for High Dimensional Classification using A Competitive Swarm Optimizer

5.1 Introduction

In machine learning and data mining, the target concepts of a dataset are usually described by a group of features. To build reliable models for solving problems such as classification, it is expected that the features contain as much useful information as possible, and the number of features can be as small as possible. However, since there is often little prior knowledge on the dataset, it is difficult to distinguish which features are relevant and which are not. As a consequence, there are usually a large number of features to be taken into consideration, including many irrelevant and redundant features. Unfortunately, irrelevant and redundant features will cause a few issues, which are: 1) increase training time, 2) reduce the
training efficiency, 3) negatively influence the performance of machine learning model thus trained with them. These issues are mainly caused by the *curse of dimensionality* [171].

Various feature selection methods have been proposed [172] to eliminate the negative impact of the irrelevant and redundant features. The main target of feature selection is to choose relevant features from a large feature set [173, 174]. From the optimisation point of view, feature selection is a difficult combinatorial optimisation problem [171]. Firstly, since the size of the feature subset is not known *a priori*, the dimensionality of the decision space is non-reducible. Secondly, since the features may have complementary or contradictory interactions with each other, the decision space is non-separable. Thus, given an $m$-dimensional feature set, the number of all possible feature subsets is as large as $2^m$, which makes it very unlikely (if not impossible) to solve it with traditional exhaustive search approaches.

Due to the inefficiency of traditional search approaches in solving complex combinatorial optimisation problems, various metaheuristics have been proposed, such as GAs [175], differential evolution (DE) [176], and PSO [177], among many others [178]. Among various metaheuristics, PSO is well known for its algorithmic simplicity and computational efficiency. Recently, some researchers have proposed to apply PSO to feature selection [179, 180]. However, canonical PSO has many limitations for feature selection [181, 182]. Firstly, PSO was originally proposed for continuous optimisation problems, while feature selection is a combinatorial optimisation problem. Secondly, although PSO shows promising performance on low-dimensional problems, it suffers from the *curse of dimensionality*. Very recently, Cheng and Jin have proposed a PSO variant, known as competitive swarm optimiser (CSO), for large-scale optimisation [183]. In CSO, both the global best position and the personal best position are removed. Instead of learning from the global and personal best positions, the particles in CSO learn from randomly se-
lected competitors. CSO shows promising performance on a variety of continuous test problems of a dimension up to 5000, in comparison with some state-of-the-art algorithms for large-scale optimisation.

In this work, we propose to adopt CSO for high-dimensional feature selection. To this end, CSO is modified to be suited for combinatorial optimisation. The modified CSO variant is then embedded in a wrapper feature selection approach.

The remainder of this chapter is organised as follows. Section 5.2 briefly describes related work, including the canonical PSO algorithm and its relevant variants. Section 5.3 details the proposed method, and the experimental results and discussions are presented in Section 5.4. Finally, we summarise this chapter in Section 5.5.

## 5.2 Background

### 5.2.1 The canonical PSO algorithm

The canonical PSO algorithm was developed by Kennedy and Eberhart in 1995 to solve optimisation problems by emulating social swarm behaviours of animals like birds flocking [184].

In PSO, each particle maintains a position and a velocity in an \( n \)-dimensional search space, representing a candidate solution and direction to a potentially better solution. For searching for the position of the global optimum, each particle is iteratively updated as follows:

\[
\begin{align*}
    v_{t+1}^i &= \omega v_t^i + \phi_1 R_1^t (\hat{g}_t - x_t^i) + \phi_2 R_2^t (\hat{x}_t^i - x_t^i) \\
    x_{t+1}^i &= x_t^i + v_{t+1}^i,
\end{align*}
\]

(5.1)

where \( t \) denotes the generation number, \( x_t^i \) and \( v_t^i \) denote the position and ve-
locity of the $i$-th particle in the $t$-th generation, respectively, $\omega$ is termed inertia weight [185], $\phi_1$ and $\phi_2$ are acceleration coefficients, $R_1^t$ and $R_2^t$ are two randomly generated vectors within the range $[0, 1]^n$, $\hat{g}^t$ is the best solution found by all particles so far, also known as the global best position, and $\hat{x}_i^t$ is the best solution found by $i$-th particle so far, also known as the personal best position.

Algorithm 5.1 The canonical PSO algorithm

1: for all particle $X_i^{t=0} = \langle x_i^t, v_i^t \rangle$ do
2: initialize position $x_i^{t=0}$ and velocity $v_i^{t=0}$
3: end for
4: while stop criteria not met do
5: for all particle $X_i^t$ do
6: set personal best $\hat{x}_i$ as best position found so far by the particle
7: set global best $\hat{g}$ as best position found so far by the whole swarm
8: end for
9: for all particle $X_i^t$ do
10: update velocity using equation
\[ v_i^{t+1} = \omega v_i^t + \phi_1 R_1^t (\hat{g}^t - x_i^t) + \phi_2 R_2^t (\hat{x}_i^t - x_i^t) \]
11: update position using equation
\[ x_i^{t+1} = x_i^t + v_i^{t+1} \]
12: end for
13: end while

Although PSO has witnessed a great success over the past two decades, its performance is still limited when the optimization problem has a high-dimensional and complex search space [186, 187]. In order to enhance the performance of PSO, a number of PSO variants have been proposed, including the parameter adaptation based variants [188, 189], the new topological structure based variants [190, 191], and the hybridisation based variants [192, 193], to name a few.

The detailed algorithm can be seen in Algo. 5.1.

5.2.2 The CSO algorithm

Even though many PSO variants have been proposed, not much work has been done on developing PSO for large-scale optimisation. The performance improv-
ment of most existing PSO variants is at the cost of higher computational complexity and more complex algorithmic implementation. In addition, existing PSO variants attempt to modify the global or personal best positions, resulting in only limited performance improvement for large-scale optimisation.

**Algorithm 5.2** The CSO algorithm

1: \( t \leftarrow 0 \)
2: for all particle \( p^t_i = (x^t_i, v^t_i) \) in swarm \( P^t \) do
3: \( \) initialise position \( x^t_i \) and velocity \( v^t_i \)
4: end for
5: while termination criteria is not met do
6: for all particle \( p^t_i \) do
7: \( \) calculate fitness \( f(x^t_i) \)
8: end for
9: \( P^{t+1} \leftarrow \emptyset \)
10: while \( P^t \neq \emptyset \) do
11: randomly chose two different particles \( p^t_{r1} \) and \( p^t_{r2} \) from \( P^t \)
12: if \( f(x^t_{r1}) \) is better than \( f(x^t_{r2}) \) then
13: \( p^t_w \leftarrow p^t_{r1}, p^t_l \leftarrow p^t_{r2} \)
14: else
15: \( p^t_w \leftarrow p^t_{r2}, p^t_l \leftarrow p^t_{r1} \)
16: end if
17: \( v^t_{l+1} = R_1 v^t_l + R_2 (x^t_w - x^t_l) + \phi R_3 (\bar{x}^t - x^t_l) \)
18: \( x^t_{l+1} = x^t_l + v^t_{l+1} \)
19: \( P^{t+1} \leftarrow P^{t+1} \cup \{p^t_w, p^t_{l+1}\} \)
20: \( P^t \leftarrow P^t \setminus \{p^t_{r1}, p^t_{r2}\} \)
21: end while
22: \( t \leftarrow t + 1 \)
23: end while

The recently proposed CSO [183] has shown to be efficient for large-scale optimisation. In CSO, the particles learn from randomly selected competitors, instead of from the global or the personal best position. In each iteration, the swarm is randomly divided into two groups and pairwise competitions are carried out between the particles from each group. After each competition, the winner particle is directly passed to the next iteration, while the loser particle will update its
position and velocity by learning from the winner particle:

\[
\begin{align*}
  v_{i}^{t+1} &= R_{1}^{t} v_{i}^{t} + R_{2}^{t} (x_{w}^{t} - x_{l}^{t}) + \phi R_{3}^{t} (\bar{x}^{t} - x_{l}^{t}) \\
  x_{i}^{t+1} &= x_{i}^{t} + v_{i}^{t+1},
\end{align*}
\]

(5.2)

where \( t \) is the iteration counter, \( R_{1}^{t}, R_{2}^{t}, R_{3}^{t} \) are three randomly generated vectors within \([0, 1]^{n}\), \( x_{w}^{t} \) and \( x_{l}^{t} \) denote the winner particle and the loser particle, respectively, \( \bar{x}^{t} \) denotes the mean position of the current swarm in iteration \( t \), and \( \phi \) controls the influence of \( \bar{x}^{t} \). The detailed procedure of CSO is summarised in Algo. 5.2.

### 5.2.3 Metaheuristics for feature selection

Traditional feature selection approaches can be roughly categorised into two classes, filter approaches and wrapper approaches. Filter approaches are independent of any specific learning algorithms [194, 195] while wrapper approaches involve learning algorithms as part of the evaluation procedure [196, 197].

Using metaheuristics for feature selection has been popular recently. For example, Zhu et al. have proposed to use a GA combined with local search in a hybrid wrapper and filter feature selection algorithm [198]. Neshatian and Zhang designed a GP based multi-objective algorithm for filter feature selection in [199]. Chen et al. have applied ant colony optimization (ACO) together with a rough set theory for feature selection [200]. In particular, PSO, as a popular metaheuristic, has also been widely adopted for feature selection. Chuang et al. have developed an improved binary PSO algorithm for feature selection using gene expression data [201]. Wang et al. have suggested a filter feature selection approach based on rough set and PSO [202]. Li and Chen have reported a wrapper feature selection algorithm based on PSO and an LDA algorithm known as the PSOLDA [203]. More recently, Xue et al. have presented a multi-objective PSO.
algorithm for feature selection in [181]. However, despite the various metaheuristics applied to feature selection, little work has been done that is dedicated to selecting a subset from a large-scale feature set.

### 5.3 Proposed Method

In our proposed method, feature selection can be formulated as the following minimisation problem:

\[
\begin{align*}
\min_x & \quad f(x) \\
\text{s.t.} & \quad x \in X
\end{align*}
\] (5.3)

where \(X \in \mathbb{R}^N\) denotes the feasible solution set. To represent the selected feature sets, \(x\) is encoded by a number of \(N\) binary bits, where \(N\) is the total number of features in the original feature set. For each bit in \(x\), ‘1’ and ‘0’ denotes that the corresponding feature is or is not selected, respectively. In this way, feature selection becomes a combinatorial optimisation problem where the objective is to find the best feature subset \(x^*\) to minimise the error rate of the classification models thus trained with the selected features, which is represented by the fitness function \(f(x)\).

**Algorithm 5.3** Converting continuous values to discrete (binary) values for feature selection

1. \(S \leftarrow \emptyset\), \(S\) is the selected feature subset;
2. for all \(d_i \in x, i = 1, \ldots, N\) do, \(x\) is the particle, \(d_i\) is the \(i\)-th dimension of \(x\);
3. if \(d_i > \lambda\) then
4. \(S \leftarrow S \cup \{i\}\);
5. end if
6. end for

To solve the high-dimensional feature selection problem presented in (5.3), we employed the CSO algorithm in this work. However, in feature selection, the search landscape is discrete while the original CSO algorithm has been proposed
for continuous optimisation. In order to address this issue, we use a threshold parameter \( \lambda \) to determine whether a feature is selected or not, as shown in Algo. 5.3.

It is worth noting that since training a classifier model with a large amount of data is usually highly time-consuming, fitness evaluation in a high-dimensional feature selection is computationally very expensive. We found empirically that many particles may have similar positions which select the same features when converted to the discrete space. It is possible to avoid some computationally expensive fitness evaluations when the selected feature subsets are the same. For this purpose, an archive is designed to record the historical fitness values of all previously selected feature subset such that we can check if a certain feature selection result has already been evaluated before performing real fitness evaluation for it.

1. Look up the \textit{archive} \( \xi \) if the currently selected feature has been evaluated;

2. If it has been evaluated, extract the fitness value and assign it to the particle;

3. If it has not been evaluated, build the classifier model with selected features, and test its error rate. Assign the fitness, i.e., the average error rate, to the particle and add the selection and fitness into \( \xi \).

Our empirical tests show that this simple strategy has significantly reduced the time consumption in the search, especially when the swarm is converging and many particles have a similar position.

The proposed CSO-based feature selection method together with the above two strategies is summarised in Algo. 5.4, where \( \xi \) denotes the archive that records the fitness values of all the particles in history.
Algorithm 5.4 The proposed algorithm

1: $t \leftarrow 0, \xi \leftarrow \emptyset$
2: for all particle $p^t_i = \langle x^t_i, v^t_i \rangle$ in swarm $P^t$ do
3: initialise position $x^t_i$ and velocity $v^t_i$
4: end for
5: while termination criteria not met do
6: for all particle $p^t_i$ do
7: $S^t_i \leftarrow \emptyset$
8: for all $d^t_a \in x^t_i, a = 1 \ldots N$ do
9: if $d^t_a > \lambda$ then
10: $S^t_i \leftarrow S^t_i \cup a$
11: end if
12: end for
13: if $\langle S^t_i, \ast \rangle \in \xi$ then
14: extract $f(S^t_i)$ from $\xi$
15: else
16: calculate fitness $f(S^t_i)$ by $n$-fold cross validation
17: $\xi \leftarrow \xi \cup \{\langle S^t_i, f(S^t_i) \rangle \}$
18: end if
19: end for
20: $P^{t+1} \leftarrow \emptyset$
21: while $P^t \neq \emptyset$ do
22: randomly chose two different particles $p^t_{r1}$ and $p^t_{r2}$ from $P^t$
23: if $f(x^t_{r1})$ is better than $f(x^t_{r2})$ then
24: $p^t_w \leftarrow p^t_{r1}, p^t_l \leftarrow p^t_{r2}$
25: else
26: $p^t_w \leftarrow p^t_{r2}, p^t_l \leftarrow p^t_{r1}$
27: end if
28: $v^t_{i}^{t+1} = R^t_1 v^t_i + R^t_2 (x^t_w - x^t_i) + \phi R^t_3 (\bar{x}^t - x^t_i)$
29: $x^t_{i}^{t+1} = x^t_i + v^t_{i}^{t+1}$
30: $P^{t+1} \leftarrow P^{t+1} \cup \{p^t_w, p^t_{t+1} \}$
31: $P^t \leftarrow P^t \setminus \{p^t_{r1}, p^t_{r2} \}$
32: end while
33: $t \leftarrow t + 1$
34: end while
5.4 Experimental studies

5.4.1 Experimental settings

To assess the performance of proposed algorithm, we conducted a set of experiments on several datasets from the UCI Machine Learning Repository [168]. The properties of datasets are listed in Table 5.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>movement</td>
<td>90</td>
<td>360</td>
<td>15</td>
</tr>
<tr>
<td>musk</td>
<td>167</td>
<td>6598</td>
<td>2</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>279</td>
<td>452</td>
<td>16</td>
</tr>
<tr>
<td>madelon</td>
<td>500</td>
<td>2600</td>
<td>2</td>
</tr>
<tr>
<td>isolet5</td>
<td>617</td>
<td>1559</td>
<td>26</td>
</tr>
<tr>
<td>InterAd</td>
<td>1588</td>
<td>3279</td>
<td>2</td>
</tr>
</tbody>
</table>

For each dataset, we use 70% of the samples in the dataset as training data, and the rest for testing. The selection of training and test sets is randomised while the original ratio of class distribution is preserved in both sets.

The algorithms are implemented on Java SE 8 (revision 1.8.0_45), using Weka [204] data mining library version 3.7.12 for the base classification algorithm.

In order to test the effectiveness of the feature selection, we chose to use only simple classification models, i.e., the $k$NN classifier [205] with $k = 5$. In order to reduce the risk of overfitting, we use the average error rates of $n$-fold CV (with $n = 10$) on training data as the fitness function. PSO, four variants of PSO proposed in Xue’s paper for feature selection [181] and the PCA are compared with the proposed algorithm, along with classification with all features.

To make comparisons between CSO- and PSO-based feature selection, we accordingly modify the PSO algorithm to be suited for feature selection. The modification consists of the conversion of continuous values into discrete values for
feature selection and the archive strategy. The swarm size is set to 100 for both algorithms and the maximal number of generations is set as 200. Note that, as CSO only updates half of the population in each iteration, it only generates half of new solutions that the PSO-based algorithms generate.

The PSO variants in Xue’s paper are denoted as Xue1-kNN, Xue2-kNN, Xue3-kNN and Xue4-kNN, respectively. The major difference between Xue’s algorithms is the number of features selected in the initial population, while Xue1-kNN uses the traditional initialisation strategy in which about half of the features are selected in each individual, Xue2-kNN uses small initialisation strategy where only about 10% features are selected in each individual, Xue3-kNN uses large initialisation strategy where more than half (we used about 2/3 in the experiment) of the features are selected in each individual, and Xue4-kNN uses a combined initialisation where a majority (we used 2/3 in the experiment) of the individuals are initialised with the small initialisation strategy (about 10% features in the experiment), while the rest are initialized with the large initialisation strategy (about 2/3 features in the experiment). Another main difference between Xue’s algorithms and traditional PSO-based algorithms is that in Xue’s algorithm, the threshold parameter $\lambda$ is set to 0.6 while the traditional algorithm uses 0.5 as the threshold parameter.

Other parameters of the training algorithms are: $w$ in the PSO-kNN is set to 0.7298 and both $c_1$ and $c_2$ are 1.49618; $\phi$ in the CSO-kNN is set to 0.1. The particles in all algorithms are randomly initialised between $[0, 1]$ and the threshold parameter $\lambda = 0.5$ is applied on CSO-kNN and PSO-kNN while $\lambda = 0.6$ is applied on Xue’s algorithms. The variance covered in PCA-based feature selection (PCA-kNN) is 0.95. Each algorithm is run for 15 times independently to obtain statistical results.
5.4.2 Results

Error rate

The ultimate target of classification is to improve generalisation ability, which means a lower error rate on unseen data. Therefore, we firstly examine the average error rate of all compared algorithms. The results are summarised in Table 5.2. We also adopted the Wilcoxon rank sum test to compare the results obtained by the CSO-kNN algorithm and other compared algorithms at a significance level of 0.05. The result is also listed in Table 5.2, where symbol ‘+’ denotes the particular algorithm is significantly outperformed by the CSO-kNN algorithm according to the Wilcoxon rank sum test, while ‘−’ denotes the particular algorithm is significantly better than the CSO-kNN algorithm, and ‘=’ denotes that there is no statistically significant difference between the results obtained by the CSO-kNN algorithm and this particular algorithm.

Table 5.2: Average error rate

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CSO-kNN</th>
<th>PSO-kNN</th>
<th>Xue1-kNN</th>
<th>Xue2-kNN</th>
<th>Xue3-kNN</th>
<th>Xue4-kNN</th>
<th>PCA-kNN</th>
<th>SC-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>movement</td>
<td>0.2330</td>
<td>0.2844+</td>
<td>0.2965+</td>
<td>0.2862+</td>
<td>0.2881+</td>
<td>0.2716+</td>
<td>0.2844</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(±0.0257)</td>
<td>(±0.0359)</td>
<td>(±0.0367)</td>
<td>(±0.0344)</td>
<td>(±0.0329)</td>
<td>(±0.0292)</td>
<td>(±0.0341)</td>
<td>(±0.0306)</td>
</tr>
<tr>
<td>musk</td>
<td>0.0009</td>
<td>0.0035+</td>
<td>0.0031+</td>
<td>0.0016+</td>
<td>0.0035+</td>
<td>0.0019+</td>
<td>0.0034+</td>
<td>0.0147</td>
</tr>
<tr>
<td></td>
<td>(±0.0008)</td>
<td>(±0.0012)</td>
<td>(±0.0015)</td>
<td>(±0.0010)</td>
<td>(±0.0022)</td>
<td>(±0.0010)</td>
<td>(±0.0019)</td>
<td>(±0.0025)</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>0.3196</td>
<td>0.4078+</td>
<td>0.4091+</td>
<td>0.3681+</td>
<td>0.4118+</td>
<td>0.3618+</td>
<td>0.4578+</td>
<td>0.4304</td>
</tr>
<tr>
<td></td>
<td>(±0.0224)</td>
<td>(±0.0192)</td>
<td>(±0.0175)</td>
<td>(±0.0287)</td>
<td>(±0.0245)</td>
<td>(±0.0303)</td>
<td>(±0.0076)</td>
<td>(±0.0149)</td>
</tr>
<tr>
<td>madelon</td>
<td>0.1659</td>
<td>0.4178+</td>
<td>0.4119+</td>
<td>0.2438+</td>
<td>0.4149+</td>
<td>0.3824+</td>
<td>0.4795+</td>
<td>0.4369</td>
</tr>
<tr>
<td></td>
<td>(±0.0411)</td>
<td>(±0.0173)</td>
<td>(±0.0167)</td>
<td>(±0.0845)</td>
<td>(±0.0241)</td>
<td>(±0.0924)</td>
<td>(±0.0179)</td>
<td>(±0.0119)</td>
</tr>
<tr>
<td>isolet5</td>
<td>0.1487</td>
<td>0.1879+</td>
<td>0.1883+</td>
<td>0.1895+</td>
<td>0.1974+</td>
<td>0.2013+</td>
<td>0.4400+</td>
<td>0.2202</td>
</tr>
<tr>
<td></td>
<td>(±0.0141)</td>
<td>(±0.0123)</td>
<td>(±0.0130)</td>
<td>(±0.0164)</td>
<td>(±0.0148)</td>
<td>(±0.0124)</td>
<td>(±0.0245)</td>
<td>(±0.0131)</td>
</tr>
<tr>
<td>InterAd</td>
<td>0.0298</td>
<td>0.0401+</td>
<td>0.0403+</td>
<td>0.0419+</td>
<td>0.0402+</td>
<td>0.0398+</td>
<td>0.0704+</td>
<td>0.0456</td>
</tr>
<tr>
<td></td>
<td>(±0.0049)</td>
<td>(±0.0066)</td>
<td>(±0.0057)</td>
<td>(±0.0071)</td>
<td>(±0.0038)</td>
<td>(±0.0034)</td>
<td>(±0.0053)</td>
<td>(±0.0043)</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>-</td>
<td>6/0/0</td>
<td>6/0/0</td>
<td>6/0/0</td>
<td>6/0/0</td>
<td>6/0/0</td>
<td>6/0/0</td>
<td>-</td>
</tr>
</tbody>
</table>

The experimental results show that the CSO-kNN has achieved a statistically lower error rate than all other compared algorithms on all the six datasets. On some datasets, e.g., the musk dataset and the madelon dataset, the error rate achieved by the CSO-kNN is even smaller than half of that achieved by other algorithms. Compared to kNN, CSO-kNN is always better while all other algorithms have instances when they performed worse than kNN. From these results,
we can also see that PCA-\(k\)NN is the least effective algorithm as most results it has obtained are worse than those of \(k\)NN. This can be attributed to the fact that PCA is sensitive to noises and outliers. In other words, PCA is inefficient in removing class-irrelevant attributes.

**Performance**

In the proposed algorithm, the fitness function is to minimise the average error rate on the training data. The fitness convergence profiles of the algorithms are plotted in Fig. 5.1. In the figure, X-axes and Y-axes are the generations and the fitness values, respectively.

From Fig. 5.1, it can be seen that CSO-\(k\)NN is always able to find better solutions, although it may be a bit slower at the early search stage. PSO-\(k\)NN and Xue’s algorithms tend to be trapped in a premature convergence around generation 20. The only exception is found in the first dataset where the difference between CSO-\(k\)NN and compared algorithms is negligible, which might because the dimension of this dataset is relatively low (only 90) on which the difference in search performance between PSO and CSO is minor.

As fitness selection is a combinatorial optimisation problem, small changes in the particle positions may not result in a change in the selected features. Thus, it is easy to understand that many new solutions are found at the beginning of the evolutionary optimisation. As the evolution proceeds, the number of new solutions that can be found may dramatically reduce. Thus, the number of new solutions found can be used as another indicator for convergence, which is presented in Fig. 5.2.

The plots in Fig. 5.2 visually show that the PSO-based algorithms seem to converge faster than the CSO-\(k\)NN since the PSO-based algorithms have found only few solutions at the later stage of evolution. However, a slower convergence rate of
Figure 5.1: Average fitness of the best particle respect to each generation.

CSO-\(k\)-NN than the PSO variants is expected, which indicates a larger diversity in the population of CSO-\(k\)-NN that leads to a better ability to find better solutions in large-scale optimisation. Comparing these plots with those in Fig. 5.1, we can see that the fitness of the PSO-based algorithms improves much more slowly than that of the CSO-\(k\)-NN. The best fitnesses achieved by the PSO-based algorithms were always worse than those by the CSO-\(k\)-NN, which strongly indicates that
the PSO-based algorithms have got trapped in a local optimum. By contrast, the CSO-\(k\)NN can always find better solutions in terms of the fitness value. The only exception was on the first dataset, i.e., the dataset \textit{movement}. Both PSO-based algorithms and CSO-\(k\)NN have found a small number of new solutions probably because of the low dimensionality of this dataset. The fact that CSO-\(k\)NN continues to find new better feature subsets implies that feature selection is naturally
a multi-modal optimisation problem, i.e., many different combinations of feature subsets may have similar or the same generalisation performance.

It also worth mentioning that, apart from the CSO-\(k\)-NN on the last two datasets, most other experiments have many identical solutions produced during the search, meaning that many fitness evaluations may be redundant. As fitness evaluations are very time-consuming for high-dimensional feature selection, it would have taken significantly more time if we do not use the archive mechanism proposed in the proposed algorithm.

**Selected features**

The second target of feature selection is to remove irrelevant features to enhance classification performance. Therefore, we have also taken a look at the average number of selected features obtained in the PSO-based algorithms and the CSO-\(k\)-NN. The results are presented in Table 5.3.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CSO-(k)-NN</th>
<th>PSO-(k)-NN</th>
<th>Xue1-(k)-NN</th>
<th>Xue2-(k)-NN</th>
<th>Xue3-(k)-NN</th>
<th>Xue4-(k)-NN</th>
<th>PCA-(k)-NN</th>
<th>#Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>movement</td>
<td>46.60 (\pm) 5.51</td>
<td>41.07 (\pm) 4.79</td>
<td>43.87 (\pm) 5.48</td>
<td>23.73 (\pm) 8.04</td>
<td>53.33 (\pm) 8.17</td>
<td>39.40 (\pm) 20.15</td>
<td>9 (+)</td>
<td>90</td>
</tr>
<tr>
<td>musk</td>
<td>17.31 (\pm) 8.13</td>
<td>71.47 (\pm) 4.55</td>
<td>72.07 (\pm) 6.04</td>
<td>15.40 (\pm) 8.26</td>
<td>75.73 (\pm) 10.12</td>
<td>14.13 (\pm) 5.94</td>
<td>122 (+)</td>
<td>167</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>17.20 (\pm) 6.75</td>
<td>131.87 (\pm) 8.58</td>
<td>131.27 (\pm) 11.25</td>
<td>19.60 (\pm) 13.15</td>
<td>160.80 (\pm) 11.10</td>
<td>21.73 (\pm) 13.04</td>
<td>103 (+)</td>
<td>279</td>
</tr>
<tr>
<td>madelon</td>
<td>7.33 (\pm) 2.32</td>
<td>252.80 (\pm) 14.53</td>
<td>259.00 (\pm) 16.18</td>
<td>26.00 (\pm) 51.35</td>
<td>323.33 (\pm) 32.27</td>
<td>274.27 (\pm) 139.75</td>
<td>426 (+)</td>
<td>500</td>
</tr>
<tr>
<td>isolet5</td>
<td>138.33 (\pm) 28.42</td>
<td>304.00 (\pm) 13.83</td>
<td>312.80 (\pm) 17.63</td>
<td>178.07 (\pm) 52.16</td>
<td>364.87 (\pm) 28.93</td>
<td>369.93 (\pm) 63.58</td>
<td>182 (+)</td>
<td>617</td>
</tr>
<tr>
<td>InterAd</td>
<td>352.20 (\pm) 114.04</td>
<td>765.60 (\pm) 20.48</td>
<td>784.73 (\pm) 33.46</td>
<td>324.87 (\pm) 131.89</td>
<td>871.73 (\pm) 73.68</td>
<td>870.00 (\pm) 171.58</td>
<td>300 (\pm)</td>
<td>1558</td>
</tr>
</tbody>
</table>

In this comparison, we can see that CSO-\(k\)-NN selects fewer features than most compared algorithms statistically, except for Xue2-\(k\)-NN, which initialise its population with only 10\% of the total features. We found that the number of features selected by the PSO-based algorithms is proportional to the number of features initialized at the first generation. In other words, if particles are initialised with
a large number of features, the number of features selected in the final population will be larger, and vice versa. Therefore, Xue2-$k$NN is more likely to outperform CSO-$k$NN in terms of the number of selected features. Furthermore, as Xue4-$k$NN has a mixed initialisation, depending on the characteristics of the dataset, the best particle may be evolved from those particles initialised with a small or large number of features. As a result, the best solution in the final population selects either much more or fewer features than CSO-$k$NN does. By contrast, CSO-$k$NN is not sensitive to the initialisation, which can always find the optimal feature subset regardless of the number of features selected during the initialisation.

Further observations regarding the number of features that have been selected during the search procedure by each algorithm can be found in Fig. 5.3. We can see that CSO-$k$NN not only finds smaller feature subsets than the PSO-based algorithms on large-scale problems, but the number of selected features also decreases much faster. The only exception is again the Xue2-$k$NN algorithm due to the same reason that we have discussed above.

From Table 5.2 and Fig. 5.3, we can conclude that CSO-$k$NN has a higher degree of diversity than the PSO-based algorithms, which enables it to explore the search space to find a solution that selects a smaller number of features and better performance.

In addition to the number of features that have been selected, it is also of interest to see what features have exactly been selected. To this end, we have also plotted in Fig. 5.4 the number of selected features over the generations during the evolution.

Figs. 5.4a, 5.4c, 5.4e, 5.4g, 5.4i and 5.4k show the results on the movement dataset, which has the least number of features among the six tested datasets. On this dataset, all algorithms seem to have successfully found the global optimum (refer to Table 5.2 and Fig. 5.3) and a similar number of features is selected,
which is indicated by the similar overall darkness in the plots. However, there are still visible differences between Xue2-$k$NN and Xue3-$k$NN as the initial number of selected features are different. Furthermore, these figures also show that CSO-$k$NN has a much higher degree of diversity as the selected features are rapidly changing in contrast to other compared algorithms.
Figure 5.4: Graphics showing that the selected features (x-axes) of the best individual over generations (y-axes). The darkness of each pixel represents the frequency of the corresponding feature that has been selected in the respective generation across different experiments - a darker colour represents a higher frequency and vice versa.

Figs. 5.4b, 5.4d, 5.4f, 5.4j and 5.4l present the results on the arrhythmia dataset, where the CSO-kNN has selected much fewer features and achieved a considerably lower error rate than the compared algorithms. In general, the darkness of the plot for CSO-kNN is much lighter than that of the PSO-kNN, Xue1-kNN and Xue3-kNN, which implies that the general number of features
selected by CSO-$k$NN is much smaller than these algorithms in all independent runs. It is also seen that the overall darkness among CSO-$k$NN, Xue2-$k$NN and Xue4-$k$NN are similar. However, on close inspection, we can find that the early populations of CSO-$k$NN are much darker than the other two algorithms, and as the population evolves, the overall darkness becomes lighter. By contrast, all PSO-based algorithms have a similar overall darkness across the generations. This difference indicates that the CSO-$k$NN is liable to local optimums and has a higher degree of diversity than the PSO-based algorithms. Nevertheless, a few clearly discernible vertical dark lines can be seen in the plots, especially in the CSO-$k$NN, Xue2-$k$NN and Xue4-$k$NN. These lines represent the commonly selected features of all independent runs while most other lines remain light indicating the scarcely (or even never) selected features. The plots clearly show that the PSO-based algorithms tend to select many different features which are mostly not helpful, if not harmful to improve the generalisation ability.

5.5 Summary

This work proposes an efficient feature selection algorithm. The objective is to select a small feature subset from a large number of features while improving or maintaining the classification performance compared with using all features. We adopted the CSO algorithm, with a few modification to fit the specific problem, to achieve the objective.

Experiments on various datasets that were containing a range number of features were taken to test the proposed algorithm. The results demonstrate that the CSO-based methods select fewer features while having better generalisation than the compared algorithms. The difference is more notable if the number of features is large. The purposed method also found to be insensitive to initialisation, unlike the PSO-based methods. This could due to the number of new solutions found
by the proposed method is considerably larger than that found by the PSO-based methods.

At the beginning of this thesis, we identified two main issues in recent data mining applications, which are large number of features and small amount of labelled data. This chapter mainly focus on the former issue, while the latter will be aimed on the next chapter.
6.1 Introduction

In machine learning and data mining applications, unlabelled training samples can be easily made available, while labelled ones are more difficult and expensive to obtain. Out of this reason, SSL, which able to benefit from unlabelled samples together with labelled ones, has attracted more and more attention in the past years.

Most existing SSL techniques distinguish themselves mainly in the way of labelling unlabelled data. These methods can largely be divided into three main categories, which are graph based algorithms [119, 120, 128, 206], expectation-maximization (EM) methods [107, 110, 113, 207] and ensemble methods [106, 121, 123, 135, 208, 209].
Co-training [121], which trains two classifiers on two different ‘views’ then labels unlabelled data based on the prediction of one classifier to augment the training set of the other, is one of the most significant achievements in SSL. In this work, the ‘views’ are sets of attributes which are sufficient and redundant. In other words, each view must be sufficient to train the classifier while the two views are conditionally independent. Dasgupta et al. [210] have shown that if the conditions are met, co-training could achieve better generalisation by maximising its base classifiers’ agreement over unlabelled samples. In practice, however, these conditions are not easy to be satisfied. In order to address the above issue, Goldman and Zhou [135] attempted to use two different supervised learning algorithms to partition the example space into a set of equivalent classes. Unfortunately, their method entails a time-consuming cross-validation technique to label the unlabelled samples.

Zhou [123] extended the co-training method by proposing a tri-training algorithm. Instead of using two classifiers, tri-training uses three classifiers. Those three classifiers are initially constructed by subsampling the labelled samples. At each training iteration, an unlabelled data is labelled for one classifier if the other two classifiers agree on the labelling, under certain conditions. The tri-training method is attractive as it has successfully lifted the requirement of two conditionally independent views in the original co-training method without undergoing the time-consuming cross-validation process proposed in [135]. One potential weakness of the tri-training algorithm is that, as the initial classifiers are trained by subsampling the labelled data, the diversity among the three classifiers may not be guaranteed.

To benefit from the improved accuracy on ensemble learning [4, 211, 212], techniques of a combination of SSL and ensemble are recently became interest. Shao and Tian [213] proposed a selective SSL ensemble learning based on distance to model (DM). Xiao et al. [214] proposed a SSL ensemble on clustering applications.
In this chapter, we propose a new semi-supervised ensemble learning algorithm, which is called Multi-Train. Multi-Train does not require two views like co-training does, instead, it creates multiple views by manipulating the features in different ways or using different types of learning models. The unlabelled samples are predicted by a simple majority voting of the ensemble members, instead of complex measuring methods like DM, in a hope to efficiently improve the accuracy in predicting the labels of unlabelled samples with minimum overhead cost.

The rest of this chapter is organised as follows: Section 6.2 describes the related work, which including co-training and tri-training algorithms. Section 6.3 presents the proposed Multi-Train algorithm. Section 6.4 gives the experimental settings and empirical results on a set of UCI benchmark datasets [168]. Finally, Section 6.5 summarises this chapter.

6.2 Background

Let $L$ denotes the labelled samples with the size $|L|$ and $U$ the unlabelled samples with the size of $|U|$. In many machine learning problems, $|L|$ is typically small. The key issue is how to label some samples in $U$ and use them for training the classifiers together with the labelled samples so that the ensemble can predict more accurately on unseen data.

6.2.1 Co-training

Co-training is a class of the SSL algorithm, which tries to label unlabelled data by taking two independent feature sets as two “views” that are independent and sufficient for correct classification.

Denote the instance space $X = X_1 \times X_2$, each sample $x = (x_1, x_2)$, the distribution over $X$ as $D$, two target functions $f_1 \in C_1$ and $f_2 \in C_2$ over $X_1$ and $X_2$
Algorithm 6.1 The co-training algorithm

1: \( L \): The labelled samples set
2: \( U \): The unlabelled samples set
3: \( n \): Sample size
4: \( T \): Maximum number of iterations
5: \( C \): Number of classes
6: \( \mathcal{H}(X) \): The learning algorithm
7: \( \{ Pr_c \}_{c=1}^{C} \) \( \leftarrow \) class prior probabilities
8: Class growth rate \( n_c \leftarrow n \times Pr_c, (c = 1, \ldots, C) \)
9: \( h_0^1 \leftarrow \mathcal{H}(L(X_1)) \)
10: \( h_0^2 \leftarrow \mathcal{H}(L(X_2)) \)
11: \( t \leftarrow 1 \)
12: \textbf{repeat}
13: \hspace{1em} \textbf{for} \( v \in \{1, 2\} \) \textbf{do}
14: \hspace{2em} \text{Predict} \ U \text{ using} \ h_t^{v-1} \)
15: \hspace{2em} \( S_v \leftarrow \emptyset \)
16: \hspace{2em} \textbf{for} \( c \in \{1, \ldots, C\} \) \textbf{do}
17: \hspace{3em} \( S_v \leftarrow S_v \cup \{n_c \text{ most confident samples of class } c \text{ in the previous prediction} \} \)
18: \hspace{2em} \( L \leftarrow L \cup S_v \)
19: \hspace{2em} \( U \leftarrow U \setminus S_v \)
20: \hspace{2em} \textbf{end for}
21: \hspace{1em} \textbf{end for}
22: \( h_t^1 \leftarrow \mathcal{H}(L(X_1)) \)
23: \( h_t^2 \leftarrow \mathcal{H}(L(X_2)) \)
24: \textbf{until} \( t = T \) \textbf{or} \( |U| = 0 \)
25: \textbf{return} combination of the predictions of \( h_t^1 \) and \( h_t^2 \)
respectively.

**Sufficiency:** The instance distribution $D$ is assumed to be compatible with the target function $f = (f_1, f_2)$ if for any $x = (x_1, x_2)$ with non-zero probability, $f(x) = f_1(x_1) = f_2(x_2)$. The compatibility of $f$ with $D$:

$$p = 1 - Pr_D[(x_1, x_2) : f_1(x_1) \neq f_2(x_2)].$$

(6.1)

**Independency:** A pair of views $(x_1, x_2)$ satisfy view independency just in case:

$$Pr[X_1 = x_1 | X_2 = x_2, Y = y] = Pr[X_1 = x_1, Y = y]$$
$$Pr[X_2 = x_2 | X_1 = x_1, Y = y] = Pr[X_2 = x_2, Y = y].$$

(6.2)

In the training process, two classifiers are initially trained with $L$, each classifier then label one sample in $U$ based on its prediction, which the labels are then used to retrain the other classifier. This process iteratively refines the classifiers by moving samples in $U$ to $L$ and then train the classifier with the updated $L$, until $k$ iterations.

The pseudocode of this algorithm can be seen in Algo. 6.1.

### 6.2.2 Tri-training

One main difficulty for co-training algorithm is that it requires two independent views, which can hardly be satisfied in most machine learning problems. Furthermore, the estimation of the most confident samples in co-training is done by cross-validation, which is a time-consuming process. In order to overcome these issues, Zhou [123] proposed the tri-training algorithm. Instead of training each classifier using different feature sets in co-training, tri-training subsamples $L$ to create different classifiers.
The idea of tri-training is to train three classifiers from $L$. Each classifier is then refined by the unlabelled samples that other two classifiers agree on their prediction. Therefore, the estimation of confidence is no longer necessary.

The three classifiers are initially trained by data subsampled from $L$ so that diverse ensemble members can be created. In each iteration, three classifiers are refined one by one, guided by the simultaneous error $\mathcal{E}$ on the rest two classifiers. As estimation of classification error on the unlabelled samples is difficult, $\mathcal{E}$ is measured on labelled samples only, based on the assumption that unlabelled samples have the same distribution as labelled ones. $\mathcal{E}$ is defined by the percentage of samples in $L$ are simultaneously misclassified by the rest of the classifiers. The pseudocode for error estimation can be seen in Algo. 6.2.

**Algorithm 6.2** The simultaneous error measuring algorithm

1: $<x, y> \in X$: The samples set with label  
2: $H$: Trained learning algorithm  
3: $err \leftarrow 0$  
4: $count \leftarrow 0$  
5: **for all** $x \in X$ **do**  
6: \hspace{1em} $flag \leftarrow True$  
7: \hspace{1em} **for all** $h_i \in H$ **do**  
8: \hspace{2em} $y_i \leftarrow h_i(x)$  
9: \hspace{2em} **for all** $h_j \in \{H \setminus \{h_i\}\}$ **do**  
10: \hspace{3em} $y_j \leftarrow h_j(x)$  
11: \hspace{3em} **if** $y_i \neq y_j$ **then**  
12: \hspace{4em} $flag \leftarrow False$  
13: \hspace{2em} **end if**  
14: \hspace{1em} **end for**  
15: **if** $flag = True$ **then**  
16: \hspace{1em} $count \leftarrow count + 1$  
17: \hspace{1em} **if** $y_i \neq y$ **then**  
18: \hspace{2em} $err \leftarrow err + 1$  
19: \hspace{2em} **end if**  
20: **end if**  
21: **end for**  
22: **end for**  
23: **return** $err/count$
The training of tri-training progress continues until the error $E$ stops decreasing, which indicates that the maximum generalisation has not been achieved. With certain theoretically proved restrictions, agreed unlabelled samples are gradually added to the labelled data, which are used to refine the corresponding classifier until none of the classifiers changes.

Once the training process finishes, the ensemble can be used to predict unlabelled or unseen data with the label that two or more member classifiers agree on.

The detailed tri-training algorithm is listed in Algo. 6.3.

6.3 Proposed method

While tri-training can be considered as an extension of the co-training framework, this work aims to create even more “views” to enhance the performance of semi-supervised learning. To this end, we resort to different means to create diversity among the ensemble members. These may include the use of different classifier models or different feature manipulation methods, or a combination of both. For instance, there are many machine learning models as well as various supervised learning algorithms, which can be used to create different “views”. It is worth mentioning that in order to create views as independent as possible, the models should be as different as possible. For example, LDA and LSVM have both linear hyperplanes, thus, the “views” they create are less independent. By contrast, LDA and $k$NN are more likely to create different views, as $k$NN has discrete hyperplanes that are different from that in LDA.

Another way of creating different “views” is to apply various feature manipulation methods to create different features, either by selecting a subset of the original features, or by transforming the original features into a different spaces using a dimension reduction method.
Algorithm 6.3 The tri-training algorithm

1: \( L \): The labelled samples set
2: \( U \): The unlabelled samples set
3: \( \mathcal{H}(X) \): The learning algorithm
4: \( \mathcal{B}(X) \): The bootstrap algorithm
5: \( \mathcal{S}(X, n_{out}) \): The subsampling algorithm
6: \( \mathcal{E}(X, h, \ldots) \): The simultaneous error measuring algorithm
7: for \( i \in \{1, \ldots, 3\} \) do
8: \( h_i \leftarrow \mathcal{H}(\mathcal{B}(L)), \quad e'_i \leftarrow 0.5, \quad l'_i \leftarrow 0 \)
9: end for
10: repeat
11: for \( i \in \{1, \ldots, 3\} \) do
12: \( L_i \leftarrow \emptyset \)
13: \( update_i \leftarrow False \)
14: \( e_i \leftarrow \mathcal{E}(L, h_j, h_k), (j, k \neq i) \)
15: if \( e_i < e'_i \) then
16: for all \( x \in U \) do
17: if \( h_j(x) = h_k(x), (j, k \neq i) \) then
18: \( L_i \leftarrow L_i \cup \{ < x, h_j(x) > \} \)
19: end if
20: end for
21: if \( l'_i = 0 \) then
22: \( l'_i \leftarrow \left\lfloor \frac{e_i}{e'_i - e_i} + 1 \right\rfloor \)
23: end if
24: if \( l'_i < |L_i| \) then
25: \( update_i \leftarrow True \)
26: else if \( l'_i > \frac{e_i}{e'_i - e_i} \) then
27: \( L_i \leftarrow \mathcal{S}(L_i, \left\lceil \frac{e'_i l'_i}{e_i} - 1 \right\rceil ) \)
28: \( update_i \leftarrow True \)
29: end if
30: end if
31: end for
32: for \( i \in \{1, \ldots, 3\} \) do
33: if \( update_i = True \) then
34: \( h_i \leftarrow \mathcal{H}(L \cup L_i), \quad e'_i \leftarrow e_i, \quad l'_i \leftarrow |L_i| \)
35: end if
36: end for
37: until non of \( h_i(i \in \{1, \ldots, 3\}) \) changes
38: return \( h(x) \leftarrow \arg \max \sum_{y \in \text{label}} \sum_{i: h_i(x) = y} 1 \)
With the help of the artificially created multiple views, a large number of base classifiers could be generated. Consequently, some modifications must be made to the tri-training algorithm. First, as the number of base classifiers may be large, it is less likely that all the rest classifiers are able to agree on an unlabelled data. A solution to this issue is to introduce a voting mechanism to predict the label. Unlike in tri-training algorithm where a deterministic label is given, the proposed method predicts a probability of a data having a particular label. This probability can then be used to select the most confidently predicted unlabelled data to be added to $L$. This process is listed in Algo. 6.4.

**Algorithm 6.4 The algorithm of prediction on unseen data**

1: $x$: A sample without label  
2: $C$: The classes  
3: $H$: Trained learning algorithms  
4: for all $l \in L$ do  
5: \hspace{1em} $vote_l \leftarrow 0$  
6: \hspace{1em} $prob_l \leftarrow 0$  
7: end for  
8: for all $h_i \in H$ do  
9: \hspace{1em} $<y_i, p_i> \leftarrow h_i(x)$  
10: \hspace{1em} for all $c \in C$ do  
11: \hspace{2em} if $c = y_i$ then  
12: \hspace{3em} $vote_c \leftarrow vote_c + 1$  
13: \hspace{3em} $prob_c \leftarrow prob_c + p_i$  
14: \hspace{2em} end if  
15: \hspace{1em} end for  
16: end for  
17: $c \leftarrow \arg \max_i vote_c$  
18: return $<\frac{vote_c}{|C|}, \frac{prob_c}{|C|}>$

To label an unlabelled sample, a parameter $\sigma$ that defines the minimum confidence level of the ensemble is required. Only samples that have a confidence greater than $\sigma$ can be added to the pool in which data that can be selectively added to $L$. It is easy to understand that $\sigma$ should be in the range of $[0.5, 1]$, where $\sigma = 0.5$ represents a majority voting and $\sigma = 1$ denotes that all the rest classifiers must
agree on the predicted label.

In addition, we also modify the sampling process for selecting unlabelled data in the pool to be added to $L$. As the tri-training algorithm has no confidence indication on the unlabelled samples, it randomly selects a certain number of unlabelled data in the pool to be added to $L$. The proposed algorithm, however, adds a certain number of data to $L$ that have the highest confidence level. This will not add much computational complexity compared to the co-training algorithm does, as the confidence level is calculated based on the confidence output in each base classifier rather than using cross-validation in the co-training algorithm.

Finally, the proposed algorithm also requires the user to pair up the feature manipulation methods and the learning model, each pair representing a base classifier. The corresponding classifier is initially trained with the specified learning algorithm by the pre-specified feature manipulation method. Therefore, a pair of feature manipulation method and a model represents a “view” to the data.

The entire Multi-Train algorithm is presented in Algo. 6.5.

### 6.4 Experiments

#### 6.4.1 Experimental settings

In order to compare the performance of the proposed algorithm with the original tri-training algorithm, we conducted a set of experiments on the same 12 datasets from UCI Machine Learning Repository [168]. The properties of datasets are summarised in Table 6.1.

For each dataset, we use 25% samples in the dataset as test data, and the rest 75% are for training. As we are testing the SSL algorithm, not all training data are used with labels. We artificially set 20% of the data as labelled and the rest
Algorithm 6.5 The Multi-Train algorithm

1: $L$: The labelled samples set
2: $U$: The unlabelled samples set
3: $\sigma$: The voting confidence
4: $\mathcal{P} = \langle \mathcal{F}(x), \mathcal{H}(X) \rangle$: The feature manipulation and learning algorithm pairs
5: $\mathcal{B}(X)$: The bootstrap algorithm
6: $\mathcal{S}(X, n_{out}, R)$: The subsampling algorithm with ranking vector $R$
7: $\mathcal{E}(X, h, \ldots)$: The simultaneous error measuring algorithm
8: $N \leftarrow \text{size}(\mathcal{P})$
9: for $i \in \{1, \ldots, N\}$ do
10: \hspace{1em} $\text{homoFlag}_i \leftarrow \text{False}$
11: end for
12: for $i \in \{1, \ldots, N\}$ do
13: \hspace{1em} for $j \in \{i, \ldots, N\}$ do
14: \hspace{2em} if $\mathcal{F}_i = \mathcal{F}_j$ and $\mathcal{H}_i = \mathcal{H}_j$ then
15: \hspace{3em} $\text{homoFlag}_i = \text{True}$, $\text{homoFlag}_j = \text{True}$
16: \hspace{2em} end if
17: \hspace{1em} end for
18: end for
19: for $i \in \{1, \ldots, N\}$ do
20: \hspace{1em} $\text{tmp}_i \leftarrow \mathcal{F}_i(L)$
21: \hspace{1em} if $\text{homoFlag}_i = \text{True}$ then
22: \hspace{2em} $\text{tmp}_i \leftarrow \mathcal{B}(\text{tmp}_i)$
23: \hspace{1em} end if
24: \hspace{1em} $h_i \leftarrow \mathcal{H}_i(\text{tmp}_i)$, $e_i' \leftarrow 0.5$, $l_i' \leftarrow 0$
25: end for
26: repeat
27: for $i \in \{1, \ldots, N\}$ do
28: \hspace{1em} $L_i \leftarrow \emptyset$, $\text{Rank}_i \leftarrow \emptyset$, $\text{update}_i \leftarrow \text{False}$
29: \hspace{1em} $e_i \leftarrow \mathcal{E}(L, h_\ast), (\ast \in \{1, \ldots, i-1, i+1, \ldots, N\})$
30: \hspace{1em} if $e_i < e_i'$ then
31: \hspace{2em} for all $x \in U$ do
32: \hspace{3em} $<\text{label}, \text{confidence}> \leftarrow \text{predicted class label and confidence}$
33: \hspace{3em} if $\text{confidence} > \sigma$ then
34: \hspace{4em} $L_i \leftarrow L_i \cup \{<x, \text{label}>\}$
35: \hspace{4em} $\text{Rank}_i \leftarrow \text{Rank}_i \cup \{\text{confidence}\}$
36: \hspace{3em} end if
37: \hspace{2em} end for
38: \hspace{1em} if $l_i' = 0$ then
39: \hspace{2em} $l_i' \leftarrow \left\lfloor \frac{e_i}{e_i' - e_i} + 1 \right\rfloor$
40: \hspace{1em} end if
41: end for
41: if \( l_i' < |L_i| \) then
42: \( \text{update}_i \leftarrow \text{True} \)
43: else if \( l_i' > \frac{e_i}{e_i' - e_i} \) then
44: \( L_i \leftarrow S(L_i, \left\lceil \frac{e_i' - 1}{e_i} \right\rceil, \text{Rank}_i) \)
45: \( \text{update}_i \leftarrow \text{True} \)
46: end if
47: end for
48: for \( i \in \{1, \ldots, N\} \) do
49: if \( \text{update}_i = \text{True} \) then
50: \( h_i \leftarrow H_i(\mathcal{F}_i(L \cup L_i)), \quad e_i' \leftarrow e_i, \quad l_i' \leftarrow |L_i| \)
51: end if
52: end for
53: until non of \( h_i(i \in \{1, \ldots, N\}) \) changes
54: return \( h(x) \leftarrow \arg \max_{y \in \text{label}} \sum_{i: h_i(x) = y} 1 \)

80% as unlabelled. For example, assuming we have a dataset containing 1000 instances, 250 instances are used as test data, 750 instances are used as training data, among which 150 out of 750 instances are considered as labelled and the rest 600 out of 750 instances are treated as unlabelled. The selection of training and test sets are randomised while preserving the original ratio of positive and negative classes in all sets.

The algorithms are implemented on Java SE 8 (revision 1.8.0.45), using Weka [204] data mining library version 3.7.12 for base classification algorithms.

We use three methods to create different views from same training data, namely, use of different learning models, use differently manipulated features, or a combination of the above.

As to different learning models, we use random tree [215], Naive Bayes classifier [216], J4.8 decision trees [217] and the \( k\)NN [205] with \( k = 5 \) as three different learning models.

Manipulated features include all original features, subset of features and trans-
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attribute</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>14</td>
<td>690</td>
<td>2</td>
</tr>
<tr>
<td>bupa</td>
<td>6</td>
<td>345</td>
<td>2</td>
</tr>
<tr>
<td>colic</td>
<td>22</td>
<td>368</td>
<td>2</td>
</tr>
<tr>
<td>diabetes</td>
<td>8</td>
<td>768</td>
<td>2</td>
</tr>
<tr>
<td>german</td>
<td>20</td>
<td>1000</td>
<td>2</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>29</td>
<td>3772</td>
<td>2</td>
</tr>
<tr>
<td>ionosphere</td>
<td>34</td>
<td>351</td>
<td>2</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>36</td>
<td>3196</td>
<td>2</td>
</tr>
<tr>
<td>sick</td>
<td>29</td>
<td>3772</td>
<td>2</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>9</td>
<td>958</td>
<td>2</td>
</tr>
<tr>
<td>vote</td>
<td>16</td>
<td>435</td>
<td>2</td>
</tr>
<tr>
<td>wdbc</td>
<td>30</td>
<td>568</td>
<td>2</td>
</tr>
</tbody>
</table>

formed features. We simply use PCA for transforming the features and a variant of the CSO [218], which has been shown to work well for large scale optimization, to select optimized feature subsets.

We use the average error rates of $n$-fold cross validation (with $n = 3$) on labelled data as the fitness function of the CSO to reduce the risk of overfitting in selecting feature subsets. Other parameters in the CSO algorithms are set as follows. The population size is 30, the max number of iterations is 100, $\phi$ is 0.1. In the first iteration, particles are randomly initialised between $[0, 1]$ and the threshold parameter $\lambda$ is 0.5. The variance covered in PCA transformation is 0.95. Finally, the $\sigma$ in the Multi-Train algorithm is set to 0.5. $s$ set to 0.5.

To make the comparisons as fair as possible, we apply feature manipulation prior to building the SSL base learners. In this way, we are able to directly compare the classification error rates of single classifiers, tri-training classifiers and the Multi-Train algorithm.

Each algorithm is run for 25 times independently, and the average results are presented and discussed in the following section.
6.4.2 Results

We break down the comparison into three parts. In the first part, we compare ensembles whose base learners use features generated using the same feature manipulation method, while in the second part, the classifier models are the same. The last part of the comparison compare Multi-Train ensembles using a combination of different features and different models.

Compared of ensembles with different classifier models

We employ three different feature manipulation methods in our tests. The first comparisons aim to demonstrate the benefits of using different classifier models. Therefore, we use a fixed feature manipulation method but different classifier models for comparisons.

In the tables, “MT” denotes Multi-Train and “TT” means tri-training algorithms respectively. If there is no prefix then these are supervised learning algorithms. “CSO”, “PCA”, and “NONE” indicate the feature manipulation methods, which are CSO-based feature selection, PCA feature transformation (dimension reduction), and the original features, respectively. In addition, “RT”, “NB”, “J48”, and “\(k\)NN” denote the learning algorithms, which are random trees, Naive Bayes classifiers, J48 decision trees and the \(k\)NN algorithm. All results are shown in Tables 6.2 to 6.4, respectively.

The first column in each table lists the result of Multi-Train containing four base learners, with each member being trained using features obtained from the same feature manipulation method, while different classifier models are adopted for base learners. The following four columns present results of four setups of the tri-training algorithm. Each setting uses features pre-manipulated the same as the Multi-Train algorithm and classifier models as noted. Other settings are the
Table 6.2: The classification error rate of Multi-Train, tri-training and single classifier with features selected by CSO-based algorithm

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-CSOTT-CSO-RTTT-CSO-NBTT-CSO-J48TT-CSO-kNNCSO-RTCSO-NBCSO-J48CSO-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1285 (0.0181) 0.1946 (0.0504) 0.2079 (0.0595) 0.1759 (0.0424) 0.1622 (0.0376) 0.1622 (0.0445) 0.2019 (0.0630) 0.1514 (0.0334) 0.1422 (0.0221)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3586 (0.0613) 0.4199 (0.0636) 0.4556 (0.0658) 0.4080 (0.0592) 0.3996 (0.0660) 0.4019 (0.0666) 0.4598 (0.0625) 0.3954 (0.0647) 0.3667 (0.0512)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1493 (0.0319) 0.2326 (0.0679) 0.2072 (0.0526) 0.2101 (0.0637) 0.1924 (0.0429) 0.2424 (0.0785) 0.1899 (0.0461) 0.1986 (0.0471) 0.1708 (0.0423)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2446 (0.0315) 0.3111 (0.0409) 0.2589 (0.0297) 0.3023 (0.0425) 0.2950 (0.0388) 0.3241 (0.0355) 0.2486 (0.0256) 0.2917 (0.0562) 0.2778 (0.0384)</td>
</tr>
<tr>
<td>german</td>
<td>0.2729 (0.0251) 0.3288 (0.0312) 0.2956 (0.0283) 0.3319 (0.0322) 0.3151 (0.0334) 0.3417 (0.0358) 0.2864 (0.0243) 0.3219 (0.0373) 0.2988 (0.0262)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.0339 (0.0439) 0.0423 (0.0235) 0.0543 (0.0114) 0.0358 (0.0192) 0.0437 (0.0182) 0.0447 (0.0292) 0.0544 (0.0112) 0.0514 (0.0190) 0.0410 (0.0177)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.0996 (0.0430) 0.1742 (0.0682) 0.1735 (0.0629) 0.1730 (0.0672) 0.1985 (0.0631) 0.1773 (0.0599) 0.1610 (0.0696) 0.1505 (0.0586) 0.1837 (0.0436)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.0436 (0.0017) 0.0525 (0.0109) 0.1065 (0.0260) 0.0475 (0.0094) 0.0786 (0.0096) 0.0563 (0.0076) 0.1093 (0.0218) 0.0934 (0.0080) 0.0761 (0.0056)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0243 (0.0063) 0.0360 (0.0109) 0.0562 (0.0260) 0.0285 (0.0094) 0.0326 (0.0076) 0.0391 (0.0096) 0.0591 (0.0218) 0.0263 (0.0080) 0.0292 (0.0056)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2353 (0.0286) 0.2987 (0.0400) 0.3106 (0.0299) 0.3003 (0.0453) 0.3013 (0.0344) 0.2671 (0.0447) 0.2892 (0.0269) 0.2901 (0.0384) 0.2425 (0.0352)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0343 (0.0154) 0.0554 (0.0284) 0.0523 (0.0217) 0.0520 (0.0330) 0.0596 (0.0269) 0.0532 (0.0285) 0.0591 (0.0245) 0.0526 (0.0242) 0.0529 (0.0250)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0387 (0.0149) 0.0798 (0.0305) 0.0587 (0.0175) 0.0772 (0.0300) 0.0545 (0.0181) 0.0892 (0.0313) 0.0528 (0.0166) 0.0829 (0.0324) 0.0469 (0.0152)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1386 (0.1855) 0.1864 (0.1864) 0.1786 (0.1786) 0.1749 (0.1749) 0.1884 (0.1884) 0.1800 (0.1800) 0.1714 (0.1714) 0.1612 (0.1612)</td>
</tr>
</tbody>
</table>

MT-CSO: Multi-Train with four base learners, which are CSO-RT, CSO-NB, CSO-J48, and CSO-kNN.
TT-CSO-RT: tri-training with learners of CSO-RT.
TT-CSO-NB: tri-training with learners of CSO-NB.
TT-CSO-kNN: tri-training with learners of CSO-kNN.
CSO-RT: single classifier that learning model is RandomTree and features are selected by CSO-based algorithm.
CSO-NB: single classifier that learning model is NaiveBayes and features are selected by CSO-based algorithm.
CSO-J48: single classifier that learning model is J4.8 decision tree and features are selected by CSO-based algorithm.
CSO-kNN: single classifier that learning model is kNN and features are selected by CSO-based algorithm.

same as suggested in the tri-training algorithm. The last four columns show the results from the single classifier models as well.

We used the Wilcoxon rank sum test to verify the significance of the improvement of the proposed algorithm, symbol ‘+’ denotes the particular setup is significantly outperformed by Multi-Train, while ‘-’ denotes the particular setup is significantly better than Multi-Train, and finally ‘=’ denotes that there is no statistically significant difference between the results obtained by Multi-Train and the particular
large errors on the same datasets. It is thus understandable that other models
learning models on these particular datasets, while other models produce much
look, we find that the J48 decision tree alone generalizes much better than other
is when the J48 decision tree is used as the learning model. By taking a closer
statistically outperformed by only in four out of the 288 different settings, which
Our results shown in Tables 6.2 to 6.4 demonstrate that the proposed algorithm is

Table 6.3: The classification error rate of Multi-Train, tri-training and single
classifier with features are transformed by PCA

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-PCATT-PCA</th>
<th>RTT-PCA</th>
<th>NBT-PCA</th>
<th>J48TT-PCA</th>
<th>NNBTT-PCA</th>
<th>RTPCA</th>
<th>NBPCA</th>
<th>J48PCA</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1528 (+)</td>
<td>0.2416 (+)</td>
<td>0.2262 (+)</td>
<td>0.2198 (+)</td>
<td>0.2050 (+)</td>
<td>0.2530 (+)</td>
<td>0.2010 (+)</td>
<td>0.2100 (+)</td>
<td>0.1765 (+)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3939 (+)</td>
<td>0.4678 (+)</td>
<td>0.4544 (+)</td>
<td>0.4494 (+)</td>
<td>0.4425 (+)</td>
<td>0.4544 (+)</td>
<td>0.4425 (+)</td>
<td>0.4280 (+)</td>
<td>0.4314 (+)</td>
</tr>
<tr>
<td>colic</td>
<td>0.2543 (+)</td>
<td>0.3928 (+)</td>
<td>0.3286 (+)</td>
<td>0.3266 (+)</td>
<td>0.3333 (+)</td>
<td>0.3891 (+)</td>
<td>0.3255 (+)</td>
<td>0.3145 (+)</td>
<td>0.3033 (+)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2595 (++)</td>
<td>0.3276 (+)</td>
<td>0.2620 (+)</td>
<td>0.3226 (+)</td>
<td>0.3012 (+)</td>
<td>0.3493 (+)</td>
<td>0.2585 (+)</td>
<td>0.2724 (+)</td>
<td>0.2783 (+)</td>
</tr>
<tr>
<td>german</td>
<td>0.2911 (+)</td>
<td>0.3629 (+)</td>
<td>0.3272 (+)</td>
<td>0.3593 (+)</td>
<td>0.3215 (+)</td>
<td>0.3872 (+)</td>
<td>0.3189 (+)</td>
<td>0.3687 (+)</td>
<td>0.3092 (+)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.0724 (++)</td>
<td>0.0705 (+)</td>
<td>0.2060 (+)</td>
<td>0.0793 (+)</td>
<td>0.0715 (+)</td>
<td>0.1095 (+)</td>
<td>0.2670 (+)</td>
<td>0.0884 (+)</td>
<td>0.0694 (+)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.0807 (+)</td>
<td>0.1705 (+)</td>
<td>0.1260 (+)</td>
<td>0.1807 (+)</td>
<td>0.2670 (+)</td>
<td>0.1939 (+)</td>
<td>0.1148 (+)</td>
<td>0.1670 (+)</td>
<td>0.2492 (+)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.1615 (+)</td>
<td>0.2164 (+)</td>
<td>0.2316 (+)</td>
<td>0.1954 (+)</td>
<td>0.1822 (+)</td>
<td>0.2369 (+)</td>
<td>0.2385 (+)</td>
<td>0.2176 (+)</td>
<td>0.1530 (+)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0519 (++)</td>
<td>0.0607 (+)</td>
<td>0.1677 (+)</td>
<td>0.0580 (+)</td>
<td>0.0559 (+)</td>
<td>0.0730 (+)</td>
<td>0.1480 (+)</td>
<td>0.0686 (+)</td>
<td>0.0536 (+)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2379 (+)</td>
<td>0.3282 (+)</td>
<td>0.3105 (+)</td>
<td>0.3037 (+)</td>
<td>0.2653 (+)</td>
<td>0.3324 (+)</td>
<td>0.2944 (+)</td>
<td>0.3093 (+)</td>
<td>0.2288 (+)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0786 (+)</td>
<td>0.1453 (+)</td>
<td>0.1171 (+)</td>
<td>0.1147 (+)</td>
<td>0.0905 (+)</td>
<td>0.1538 (+)</td>
<td>0.0865 (+)</td>
<td>0.1257 (+)</td>
<td>0.0844 (+)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0526 (+)</td>
<td>0.1035 (+)</td>
<td>0.0824 (+)</td>
<td>0.0876 (+)</td>
<td>0.0862 (+)</td>
<td>0.1188 (+)</td>
<td>0.0697 (+)</td>
<td>0.0878 (+)</td>
<td>0.0876 (+)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1739 (+)</td>
<td>0.2414 (++)</td>
<td>0.2413 (+)</td>
<td>0.2249 (+)</td>
<td>0.2185 (+)</td>
<td>0.2543 (+)</td>
<td>0.2392 (+)</td>
<td>0.2253 (+)</td>
<td>0.2025 (+)</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>12/0/0</td>
<td>11/0/1</td>
<td>12/0/0</td>
<td>11/0/1</td>
<td>12/0/0</td>
<td>10/0/2</td>
<td>12/0/0</td>
<td>7/0/5</td>
<td></td>
</tr>
</tbody>
</table>

MT-PCA: Multi-Train with four base learners, which are PCA-RT, PCA-NB, PCA-J48, and PCA-kNN.
TT-PCA-RT: tri-training with learners of PCA-RT.
TT-PCA-NB: tri-training with learners of PCA-NB.
TT-PCA-kNN: tri-training with learners of PCA-kNN.
PCA-RT: single classifier that learning model is RandomTree and features are transformed by PCA.
PCA-NB: single classifier that learning model is NaiveBayes and features are transformed by PCA.
PCA-J48: single classifier that learning model is J4.8 decision tree and features are transformed by PCA.
PCA-kNN: single classifier that learning model is kNN and features are transformed by PCA.

setup. Those results are also concluded as ‘win/lose/tie’ at the bottom of each
table.

Our results shown in Tables 6.2 to 6.4 demonstrate that the proposed algorithm is
statistically outperformed by only in four out of the 288 different settings, which
is when the J48 decision tree is used as the learning model. By taking a closer
look, we find that the J48 decision tree alone generalizes much better than other
learning models on these particular datasets, while other models produce much
large errors on the same datasets. It is thus understandable that other models
Table 6.4: The classification error rate of Multi-Train, tri-training and single classifier with all original features

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-NONE</th>
<th>NONE-RT</th>
<th>NONE-NB</th>
<th>NONE-J48</th>
<th>NONE-NN</th>
<th>NONE-NB</th>
<th>NONE-J48</th>
<th>NONE-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1277</td>
<td>0.2110</td>
<td>0.2110</td>
<td>0.1699</td>
<td>0.1551</td>
<td>0.2291</td>
<td>0.2081</td>
<td>0.1703</td>
</tr>
<tr>
<td></td>
<td>(0.0205)</td>
<td>(0.0455)</td>
<td>(0.0387)</td>
<td>(0.0368)</td>
<td>(0.0275)</td>
<td>(0.0462)</td>
<td>(0.0316)</td>
<td>(0.0471)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3337</td>
<td>0.4089</td>
<td>0.4368</td>
<td>0.3999</td>
<td>0.4299</td>
<td>0.3847</td>
<td>0.4291</td>
<td>0.3891</td>
</tr>
<tr>
<td></td>
<td>(0.0423)</td>
<td>(0.0629)</td>
<td>(0.0518)</td>
<td>(0.0623)</td>
<td>(0.0564)</td>
<td>(0.0650)</td>
<td>(0.0531)</td>
<td>(0.0568)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1402</td>
<td>0.2804</td>
<td>0.2557</td>
<td>0.2130</td>
<td>0.2156</td>
<td>0.3134</td>
<td>0.2112</td>
<td>0.2047</td>
</tr>
<tr>
<td></td>
<td>(0.0332)</td>
<td>(0.0658)</td>
<td>(0.0475)</td>
<td>(0.0662)</td>
<td>(0.0347)</td>
<td>(0.0699)</td>
<td>(0.0445)</td>
<td>(0.0625)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2394</td>
<td>0.2984</td>
<td>0.2611</td>
<td>0.3071</td>
<td>0.2946</td>
<td>0.3349</td>
<td>0.2502</td>
<td>0.2983</td>
</tr>
<tr>
<td></td>
<td>(0.0291)</td>
<td>(0.0357)</td>
<td>(0.0279)</td>
<td>(0.0382)</td>
<td>(0.0314)</td>
<td>(0.0393)</td>
<td>(0.0308)</td>
<td>(0.0438)</td>
</tr>
<tr>
<td>german</td>
<td>0.2687</td>
<td>0.3341</td>
<td>0.2966</td>
<td>0.3376</td>
<td>0.3211</td>
<td>0.3552</td>
<td>0.2737</td>
<td>0.3594</td>
</tr>
<tr>
<td></td>
<td>(0.0290)</td>
<td>(0.0293)</td>
<td>(0.0284)</td>
<td>(0.0341)</td>
<td>(0.0212)</td>
<td>(0.0288)</td>
<td>(0.0280)</td>
<td>(0.0308)</td>
</tr>
<tr>
<td>hypothryoid</td>
<td>0.0454</td>
<td>0.0423</td>
<td>0.0447</td>
<td>0.0176</td>
<td>0.0806</td>
<td>0.0452</td>
<td>0.0459</td>
<td>0.0179</td>
</tr>
<tr>
<td></td>
<td>(0.0088)</td>
<td>(0.0129)</td>
<td>(0.0092)</td>
<td>(0.0065)</td>
<td>(0.0071)</td>
<td>(0.0183)</td>
<td>(0.0095)</td>
<td>(0.0069)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.0837</td>
<td>0.1621</td>
<td>0.1640</td>
<td>0.1545</td>
<td>0.2383</td>
<td>0.1761</td>
<td>0.1568</td>
<td>0.1477</td>
</tr>
<tr>
<td></td>
<td>(0.0356)</td>
<td>(0.0513)</td>
<td>(0.0551)</td>
<td>(0.0557)</td>
<td>(0.0666)</td>
<td>(0.0516)</td>
<td>(0.0562)</td>
<td>(0.0662)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.0475</td>
<td>0.0924</td>
<td>0.1521</td>
<td>0.0366</td>
<td>0.1358</td>
<td>0.0956</td>
<td>0.1364</td>
<td>0.0298</td>
</tr>
<tr>
<td></td>
<td>(0.0107)</td>
<td>(0.0284)</td>
<td>(0.0253)</td>
<td>(0.0138)</td>
<td>(0.0180)</td>
<td>(0.0359)</td>
<td>(0.0211)</td>
<td>(0.0082)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0274</td>
<td>0.0401</td>
<td>0.0759</td>
<td>0.0247</td>
<td>0.0527</td>
<td>0.0432</td>
<td>0.0802</td>
<td>0.0251</td>
</tr>
<tr>
<td></td>
<td>(0.0054)</td>
<td>(0.0100)</td>
<td>(0.0221)</td>
<td>(0.0066)</td>
<td>(0.0075)</td>
<td>(0.0132)</td>
<td>(0.0225)</td>
<td>(0.0065)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2160</td>
<td>0.3018</td>
<td>0.3188</td>
<td>0.2853</td>
<td>0.2551</td>
<td>0.3089</td>
<td>0.2954</td>
<td>0.2883</td>
</tr>
<tr>
<td></td>
<td>(0.0218)</td>
<td>(0.0358)</td>
<td>(0.0287)</td>
<td>(0.0351)</td>
<td>(0.0265)</td>
<td>(0.0387)</td>
<td>(0.0255)</td>
<td>(0.0341)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0538</td>
<td>0.0841</td>
<td>0.0792</td>
<td>0.0581</td>
<td>0.0728</td>
<td>0.0844</td>
<td>0.0768</td>
<td>0.0615</td>
</tr>
<tr>
<td></td>
<td>(0.0138)</td>
<td>(0.0408)</td>
<td>(0.0168)</td>
<td>(0.0329)</td>
<td>(0.0194)</td>
<td>(0.0330)</td>
<td>(0.0198)</td>
<td>(0.0279)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0399</td>
<td>0.0690</td>
<td>0.0523</td>
<td>0.0744</td>
<td>0.0507</td>
<td>0.0852</td>
<td>0.0469</td>
<td>0.0833</td>
</tr>
<tr>
<td></td>
<td>(0.0116)</td>
<td>(0.0221)</td>
<td>(0.0149)</td>
<td>(0.0284)</td>
<td>(0.0218)</td>
<td>(0.0390)</td>
<td>(0.0135)</td>
<td>(0.0341)</td>
</tr>
</tbody>
</table>

| avg         | 0.1353  | 0.1918  | 0.1932  | 0.1730   | 0.1919  | 0.2047  | 0.1842   | 0.1710  |
| win/lose/tie| 11/0/1  | 11/0/1  | 8/2/2   | 12/0/0   | 10/1/2  | 8/2/2   | 11/0/1   | 8/2/2   |

MT-NONE: Multi-Train with four base learners, which are NONE-RT, NONE-NB, NONE-J48, and NONE-NN.
TT-NONE-RT: tri-training with learners of NONE-RT.
TT-NONE-NB: tri-training with learners of NONE-NB.
TT-NONE-NN: tri-training with learners of NONE-NN.
NONE-RT: single classifier that learning model is RandomTree and all features are used.
NONE-NB: single classifier that learning model is NaiveBayes and all features are used.
NONE-J48: single classifier that learning model is J4.8 decision tree and all features are used.
NONE-NN: single classifier that learning model is kNN and all features are used.

can degrade the overall performance of the ensembles as they give significant more errors. Thus, the proposed algorithm performed worse than setups using J48 decision tree alone. However, as none of these classifier models constantly outperform others, we can still conclude that the proposed algorithm is very competitive with others.

Comparisons of ensembles with different feature manipulation methods

As the original co-training algorithm learns two classifier models from two different “views” of data, and the “views” are actually different sets of features, it might be of interest to examine the influence of different feature manipulation
Table 6.5: The classification error rate of Multi-Train, tri-training and single classifier with random tree classifier model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-RT</th>
<th>TT-CSO-RT</th>
<th>TT-PCA-RT</th>
<th>TT-NONE-RT</th>
<th>CSO-RT</th>
<th>PCA-RT</th>
<th>NONE-RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1426</td>
<td>0.1946+</td>
<td>0.2416+</td>
<td>0.1961+</td>
<td>0.2019+</td>
<td>0.2530+</td>
<td>0.2291+</td>
</tr>
<tr>
<td></td>
<td>(0.0219)</td>
<td>(0.0504)</td>
<td>(0.0492)</td>
<td>(0.0455)</td>
<td>(0.0445)</td>
<td>(0.0485)</td>
<td>(0.0462)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3866</td>
<td>0.4199+</td>
<td>0.4678+</td>
<td>0.4088+</td>
<td>0.4019+</td>
<td>0.4544+</td>
<td>0.3847+</td>
</tr>
<tr>
<td></td>
<td>(0.0528)</td>
<td>(0.0636)</td>
<td>(0.0543)</td>
<td>(0.0629)</td>
<td>(0.0666)</td>
<td>(0.0516)</td>
<td>(0.0650)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1967</td>
<td>0.2326+</td>
<td>0.3276+</td>
<td>0.2984+</td>
<td>0.3241+</td>
<td>0.3493+</td>
<td>0.3349+</td>
</tr>
<tr>
<td></td>
<td>(0.0293)</td>
<td>(0.0409)</td>
<td>(0.0291)</td>
<td>(0.0357)</td>
<td>(0.0355)</td>
<td>(0.0438)</td>
<td>(0.0393)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2583</td>
<td>0.3111+</td>
<td>0.3276+</td>
<td>0.3418+</td>
<td>0.3241+</td>
<td>0.3493+</td>
<td>0.3349+</td>
</tr>
<tr>
<td></td>
<td>(0.0293)</td>
<td>(0.0409)</td>
<td>(0.0291)</td>
<td>(0.0357)</td>
<td>(0.0355)</td>
<td>(0.0438)</td>
<td>(0.0393)</td>
</tr>
<tr>
<td>german</td>
<td>0.1967</td>
<td>0.2326+</td>
<td>0.3276+</td>
<td>0.2984+</td>
<td>0.3241+</td>
<td>0.3493+</td>
<td>0.3349+</td>
</tr>
<tr>
<td></td>
<td>(0.0293)</td>
<td>(0.0409)</td>
<td>(0.0291)</td>
<td>(0.0357)</td>
<td>(0.0355)</td>
<td>(0.0438)</td>
<td>(0.0393)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.2043</td>
<td>0.4237+</td>
<td>0.0795+</td>
<td>0.0423−</td>
<td>0.0447−</td>
<td>0.1095+</td>
<td>0.0452+</td>
</tr>
<tr>
<td></td>
<td>(0.0113)</td>
<td>(0.0235)</td>
<td>(0.0065)</td>
<td>(0.0129)</td>
<td>(0.0292)</td>
<td>(0.0127)</td>
<td>(0.0193)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.0951</td>
<td>0.1742+</td>
<td>0.1705+</td>
<td>0.1621+</td>
<td>0.1773+</td>
<td>0.1939+</td>
<td>0.1761+</td>
</tr>
<tr>
<td></td>
<td>(0.0298)</td>
<td>(0.0682)</td>
<td>(0.0487)</td>
<td>(0.0513)</td>
<td>(0.0599)</td>
<td>(0.0699)</td>
<td>(0.0516)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.2586</td>
<td>0.0525+</td>
<td>0.2164+</td>
<td>0.0924+</td>
<td>0.0563+</td>
<td>0.2369+</td>
<td>0.0956+</td>
</tr>
<tr>
<td></td>
<td>(0.0997)</td>
<td>(0.0140)</td>
<td>(0.0276)</td>
<td>(0.0264)</td>
<td>(0.0142)</td>
<td>(0.0219)</td>
<td>(0.0359)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0358</td>
<td>0.0360+</td>
<td>0.0607+</td>
<td>0.0403+</td>
<td>0.0391+</td>
<td>0.0730+</td>
<td>0.0432+</td>
</tr>
<tr>
<td></td>
<td>(0.0059)</td>
<td>(0.0109)</td>
<td>(0.0075)</td>
<td>(0.0100)</td>
<td>(0.0096)</td>
<td>(0.0108)</td>
<td>(0.0132)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2547</td>
<td>0.2987+</td>
<td>0.3828+</td>
<td>0.3018+</td>
<td>0.2892+</td>
<td>0.3324+</td>
<td>0.3089+</td>
</tr>
<tr>
<td></td>
<td>(0.0329)</td>
<td>(0.0400)</td>
<td>(0.0292)</td>
<td>(0.0358)</td>
<td>(0.0447)</td>
<td>(0.0481)</td>
<td>(0.0387)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0514</td>
<td>0.0554+</td>
<td>0.1453+</td>
<td>0.0841+</td>
<td>0.0532+</td>
<td>0.1538+</td>
<td>0.0844+</td>
</tr>
<tr>
<td></td>
<td>(0.0198)</td>
<td>(0.0284)</td>
<td>(0.0464)</td>
<td>(0.0408)</td>
<td>(0.0254)</td>
<td>(0.0673)</td>
<td>(0.0330)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0502</td>
<td>0.0798+</td>
<td>0.1035+</td>
<td>0.0690+</td>
<td>0.0892+</td>
<td>0.1188+</td>
<td>0.0852+</td>
</tr>
<tr>
<td></td>
<td>(0.0163)</td>
<td>(0.0305)</td>
<td>(0.0314)</td>
<td>(0.0221)</td>
<td>(0.0531)</td>
<td>(0.0571)</td>
<td>(0.0390)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1551</td>
<td>0.1855</td>
<td>0.2414</td>
<td>0.1918</td>
<td>0.1884</td>
<td>0.2543</td>
<td>0.2047</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>8/1/3</td>
<td>12/0/0</td>
<td>9/0/3</td>
<td>8/1/3</td>
<td>12/0/0</td>
<td>10/0/2</td>
<td></td>
</tr>
</tbody>
</table>

MT-RT: Multi-Train with three base learners, which are CSO-RT, PCA-RT and NONE-RT.
TT-CSO-RT: tri-training with learners of CSO-RT.
TT-PCA-RT: tri-training with learners of PCA-RT.
TT-NONE-RT: tri-training with learners of NONE-RT.
CSO-RT: single classifier that learning model is RandomTree and features are selected by CSO-based algorithm.
PCA-RT: single classifier that learning model is RandomTree and features are transformed by PCA.
NONE-RT: single classifier that learning model is RandomTree and all features are used.

method on the performance.

Tables 6.5 to 6.8 show the comparative results obtained by ensembles with different feature manipulation methods. The results show that using different feature manipulation methods can also help enhancing the performance of the proposed method. The proposed algorithm is statistically outperformed by others only in six out of 288 compared settings.

The results in this set of the comparison confirmed the conclusions drawn from the first set of the comparisons.

108
Table 6.6: The classification error rate of Multi-Train, tri-training and single classifier with Naive Bayes classifier model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-NB</th>
<th>TT-CSO-NB</th>
<th>TT-PCA-NB</th>
<th>TT-NONE-NB</th>
<th>CSO-NB</th>
<th>PCA-NB</th>
<th>NONE-NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1719</td>
<td>0.2079+</td>
<td>0.2262+</td>
<td>0.2110+</td>
<td>0.2083+</td>
<td>0.2010+</td>
<td>0.2081+</td>
</tr>
<tr>
<td></td>
<td>(0.0361)</td>
<td>(0.0595)</td>
<td>(0.0550)</td>
<td>(0.0387)</td>
<td>(0.0630)</td>
<td>(0.0516)</td>
<td>(0.0316)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.4429</td>
<td>0.4556**</td>
<td>0.4544**</td>
<td>0.4368**</td>
<td>0.4598**</td>
<td>0.4425**</td>
<td><strong>0.4291</strong></td>
</tr>
<tr>
<td></td>
<td>(0.0523)</td>
<td>(0.0658)</td>
<td>(0.0570)</td>
<td>(0.0518)</td>
<td>(0.0625)</td>
<td>(0.0410)</td>
<td>(0.0531)</td>
</tr>
<tr>
<td>colic</td>
<td><strong>0.1837</strong></td>
<td>0.2072+</td>
<td>0.3286+</td>
<td>0.2257+</td>
<td>0.1899+</td>
<td>0.3225+</td>
<td>0.2112+</td>
</tr>
<tr>
<td></td>
<td>(0.0451)</td>
<td>(0.0526)</td>
<td>(0.0365)</td>
<td>(0.0475)</td>
<td>(0.0641)</td>
<td>(0.0455)</td>
<td>(0.0445)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2330</td>
<td>0.2589+</td>
<td>0.2209+</td>
<td>0.2651+</td>
<td>0.2486+</td>
<td>0.2585+</td>
<td>0.2502+</td>
</tr>
<tr>
<td></td>
<td>(0.0247)</td>
<td>(0.0297)</td>
<td>(0.0315)</td>
<td>(0.0279)</td>
<td>(0.0256)</td>
<td>(0.0314)</td>
<td>(0.0308)</td>
</tr>
<tr>
<td>german</td>
<td>0.2715</td>
<td>0.2956+</td>
<td>0.3272+</td>
<td>0.2916+</td>
<td>0.2864+</td>
<td>0.3189+</td>
<td>0.2737+</td>
</tr>
<tr>
<td></td>
<td>(0.0254)</td>
<td>(0.0283)</td>
<td>(0.0280)</td>
<td>(0.0284)</td>
<td>(0.0243)</td>
<td>(0.0293)</td>
<td>(0.0280)</td>
</tr>
<tr>
<td>hyperthyroid</td>
<td>0.0496</td>
<td>0.0543+</td>
<td>0.2606+</td>
<td><strong>0.0454</strong></td>
<td>0.0554+</td>
<td>0.2670+</td>
<td>0.0459+</td>
</tr>
<tr>
<td></td>
<td>(0.0091)</td>
<td>(0.0114)</td>
<td>(0.1058)</td>
<td>(0.0092)</td>
<td>(0.0112)</td>
<td>(0.1225)</td>
<td>(0.0095)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.1258</td>
<td>0.1735+</td>
<td>0.1269**</td>
<td>0.1640+</td>
<td>0.1610+</td>
<td><strong>0.1148</strong></td>
<td>0.1568+</td>
</tr>
<tr>
<td></td>
<td>(0.0520)</td>
<td>(0.0629)</td>
<td>(0.0483)</td>
<td>(0.0551)</td>
<td>(0.0696)</td>
<td>(0.0378)</td>
<td>(0.0562)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.1127</td>
<td>0.1065**</td>
<td>0.2316+</td>
<td>0.1521+</td>
<td><strong>0.0993</strong></td>
<td>0.2385+</td>
<td>0.1364+</td>
</tr>
<tr>
<td></td>
<td>(0.0219)</td>
<td>(0.0295)</td>
<td>(0.0416)</td>
<td>(0.0253)</td>
<td>(0.0242)</td>
<td>(0.0399)</td>
<td>(0.0211)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0662</td>
<td><strong>0.0562</strong></td>
<td>0.1677+</td>
<td>0.0759+</td>
<td>0.0591+</td>
<td>0.1480+</td>
<td>0.0802+</td>
</tr>
<tr>
<td></td>
<td>(0.0254)</td>
<td>(0.0260)</td>
<td>(0.0698)</td>
<td>(0.0221)</td>
<td>(0.0248)</td>
<td>(0.0698)</td>
<td>(0.0225)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td><strong>0.2733</strong></td>
<td>0.3106+</td>
<td>0.3108+</td>
<td>0.3188+</td>
<td>0.2901+</td>
<td>0.2944+</td>
<td>0.2954+</td>
</tr>
<tr>
<td></td>
<td>(0.0198)</td>
<td>(0.0299)</td>
<td>(0.0260)</td>
<td>(0.0287)</td>
<td>(0.0269)</td>
<td>(0.0270)</td>
<td>(0.0255)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0584</td>
<td>0.0523**</td>
<td>0.1171+</td>
<td>0.0792+</td>
<td><strong>0.0489</strong></td>
<td>0.0865+</td>
<td>0.0768+</td>
</tr>
<tr>
<td></td>
<td>(0.0192)</td>
<td>(0.0217)</td>
<td>(0.0352)</td>
<td>(0.0168)</td>
<td>(0.0245)</td>
<td>(0.0318)</td>
<td>(0.0198)</td>
</tr>
<tr>
<td>wdcb</td>
<td><strong>0.0430</strong></td>
<td>0.0587+</td>
<td>0.0824+</td>
<td><strong>0.0523</strong></td>
<td>0.0528+</td>
<td>0.0697+</td>
<td>0.0469**</td>
</tr>
<tr>
<td></td>
<td>(0.0151)</td>
<td>(0.0175)</td>
<td>(0.0444)</td>
<td>(0.0149)</td>
<td>(0.0166)</td>
<td>(0.0279)</td>
<td>(0.0135)</td>
</tr>
<tr>
<td>avg</td>
<td><strong>0.1693</strong></td>
<td>0.1864</td>
<td>0.2413</td>
<td>0.1932</td>
<td>0.1800</td>
<td>0.2302</td>
<td>0.1842</td>
</tr>
</tbody>
</table>

Comparison of heterogeneous ensembles

In the previous comparisons, we use different classifier models or different feature manipulation methods to create diversity among base learners. The results show that the proposed Multi-Train algorithm has achieved statically better performance than the tri-training algorithm and non-SSL methods. We are interested in investigating whether a larger ensemble containing more base learners is able to further improve the generalisation capability.

The last set of experiments to be made in this work is to compare ensembles gen-
Table 6.7: The classification error rate of Multi-Train, tri-training and single classifier with J4.8 decision tree classifier model

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1329</td>
<td>0.1759+</td>
<td>0.2198+</td>
<td>0.1699+</td>
<td>0.1514+</td>
<td>0.2100+</td>
<td>0.1703+</td>
</tr>
<tr>
<td></td>
<td>(0.0229)</td>
<td>(0.0424)</td>
<td>(0.0437)</td>
<td>(0.0368)</td>
<td>(0.0334)</td>
<td>(0.0468)</td>
<td>(0.0471)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3567</td>
<td>0.4080+</td>
<td>0.4494+</td>
<td>0.3969+</td>
<td>0.3954+</td>
<td>0.4280+</td>
<td>0.3893+</td>
</tr>
<tr>
<td></td>
<td>(0.0526)</td>
<td>(0.0592)</td>
<td>(0.0578)</td>
<td>(0.0623)</td>
<td>(0.0647)</td>
<td>(0.0630)</td>
<td>(0.0568)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1580</td>
<td>0.2101+</td>
<td>0.3226+</td>
<td>0.3071+</td>
<td>0.2917+</td>
<td>0.3274+</td>
<td>0.2983+</td>
</tr>
<tr>
<td></td>
<td>(0.0357)</td>
<td>(0.0637)</td>
<td>(0.0802)</td>
<td>(0.0622)</td>
<td>(0.0564)</td>
<td>(0.0873)</td>
<td>(0.0625)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2566</td>
<td>0.3023+</td>
<td>0.3326+</td>
<td>0.3071+</td>
<td>0.2917+</td>
<td>0.3274+</td>
<td>0.2983+</td>
</tr>
<tr>
<td></td>
<td>(0.0349)</td>
<td>(0.0425)</td>
<td>(0.0393)</td>
<td>(0.0382)</td>
<td>(0.0562)</td>
<td>(0.0380)</td>
<td>(0.0438)</td>
</tr>
<tr>
<td>german</td>
<td>0.2960</td>
<td>0.3319+</td>
<td>0.3593+</td>
<td>0.3376+</td>
<td>0.3219+</td>
<td>0.3687+</td>
<td>0.3359+</td>
</tr>
<tr>
<td></td>
<td>(0.0256)</td>
<td>(0.0322)</td>
<td>(0.0382)</td>
<td>(0.0341)</td>
<td>(0.0373)</td>
<td>(0.0301)</td>
<td>(0.0308)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.0262</td>
<td>0.0358+</td>
<td>0.0793+</td>
<td>0.0176+</td>
<td>0.0331+</td>
<td>0.0884+</td>
<td>0.0179+</td>
</tr>
<tr>
<td></td>
<td>(0.0166)</td>
<td>(0.0192)</td>
<td>(0.0097)</td>
<td>(0.0065)</td>
<td>(0.0190)</td>
<td>(0.0139)</td>
<td>(0.0069)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.1091</td>
<td>0.1739+</td>
<td>0.1807+</td>
<td>0.1545+</td>
<td>0.1595+</td>
<td>0.1670+</td>
<td>0.1477+</td>
</tr>
<tr>
<td></td>
<td>(0.0348)</td>
<td>(0.0672)</td>
<td>(0.0736)</td>
<td>(0.0557)</td>
<td>(0.0584)</td>
<td>(0.0666)</td>
<td>(0.0662)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.0327</td>
<td>0.0475+</td>
<td>0.1954+</td>
<td>0.0366+</td>
<td>0.0451+</td>
<td>0.2176+</td>
<td>0.0298+</td>
</tr>
<tr>
<td></td>
<td>(0.0070)</td>
<td>(0.0123)</td>
<td>(0.0154)</td>
<td>(0.0138)</td>
<td>(0.0124)</td>
<td>(0.0268)</td>
<td>(0.0082)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0215</td>
<td>0.0285+</td>
<td>0.0580+</td>
<td>0.0247+</td>
<td>0.0263+</td>
<td>0.0686+</td>
<td>0.0251+</td>
</tr>
<tr>
<td></td>
<td>(0.0041)</td>
<td>(0.0094)</td>
<td>(0.0072)</td>
<td>(0.0066)</td>
<td>(0.0080)</td>
<td>(0.0101)</td>
<td>(0.0065)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2361</td>
<td>0.3003+</td>
<td>0.3037+</td>
<td>0.2853+</td>
<td>0.2979+</td>
<td>0.3003+</td>
<td>0.2883+</td>
</tr>
<tr>
<td></td>
<td>(0.0304)</td>
<td>(0.0453)</td>
<td>(0.0458)</td>
<td>(0.0351)</td>
<td>(0.0384)</td>
<td>(0.0590)</td>
<td>(0.0431)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0517</td>
<td>0.0520+</td>
<td>0.1147+</td>
<td>0.0581+</td>
<td>0.0526+</td>
<td>0.1257+</td>
<td>0.0615+</td>
</tr>
<tr>
<td></td>
<td>(0.0225)</td>
<td>(0.0330)</td>
<td>(0.0327)</td>
<td>(0.0329)</td>
<td>(0.0242)</td>
<td>(0.0332)</td>
<td>(0.0279)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0542</td>
<td>0.0772+</td>
<td>0.0876+</td>
<td>0.0744+</td>
<td>0.0829+</td>
<td>0.0878+</td>
<td>0.0833+</td>
</tr>
<tr>
<td></td>
<td>(0.0228)</td>
<td>(0.0390)</td>
<td>(0.0268)</td>
<td>(0.0284)</td>
<td>(0.0324)</td>
<td>(0.0343)</td>
<td>(0.0341)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1443</td>
<td>0.1786</td>
<td>0.2249</td>
<td>0.1730</td>
<td>0.1714</td>
<td>0.2253</td>
<td>0.1710</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>11/0/1</td>
<td>12/0/0</td>
<td>8/0/4</td>
<td>11/0/1</td>
<td>12/0/0</td>
<td>9/0/3</td>
<td></td>
</tr>
</tbody>
</table>

MT-J48: Multi-Train with three base learners, which are CSO-J48, PCA-J48 and NONE-J48.
CSO-J48: single classifier that learning model is J4.8 decision tree and features are selected by CSO-based algorithm.
PCA-J48: single classifier that learning model is J4.8 decision tree and features are transformed by PCA.
NONE-J48: single classifier that learning model is J4.8 decision tree and all features are used.

erated using a combination of the settings used in the first two sets of empirical studies. As shown in Table 6.9, eight settings are considered in this comparison, including MT-Hybrid, MT-CSO, MT-PCA, MT-NONE, MT-RT, MT-NB, MT-J48, and MT-kNN. The differences among the settings lie mainly in the base learners as well as the features the base learners use for creating diversity. MT-Hybrid creates diversity by using a combination of three different feature manipulation methods and four different classifier models, resulting in 12 different base learners. MT-CSO, MT-PCA, and MT-NONE have the same settings as those in Section 6.4.2, which create diversity by using four different clasi-
Table 6.8: The classification error rate of Multi-Train, tri-training and single classifier with $k$NN classifier model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-$k$NN</th>
<th>TT-CSO-$k$NN</th>
<th>TT-PCA-$k$NN</th>
<th>TT-NONE-$k$NN</th>
<th>CSO-$k$NN</th>
<th>PCA-$k$NN</th>
<th>NONE-$k$NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1243</td>
<td>0.1622</td>
<td>0.2050</td>
<td>0.1551</td>
<td>0.1422</td>
<td>0.1765</td>
<td>0.1412</td>
</tr>
<tr>
<td></td>
<td>(0.0187)</td>
<td>(0.0376)</td>
<td>(0.0334)</td>
<td>(0.0275)</td>
<td>(0.0221)</td>
<td>(0.0364)</td>
<td>(0.0250)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3663</td>
<td>0.3966</td>
<td>0.4425</td>
<td>0.4299</td>
<td>0.3667</td>
<td>0.4314</td>
<td>0.4138</td>
</tr>
<tr>
<td></td>
<td>(0.0465)</td>
<td>(0.0660)</td>
<td>(0.0593)</td>
<td>(0.0564)</td>
<td>(0.0512)</td>
<td>(0.0609)</td>
<td>(0.0549)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1551</td>
<td>0.1924</td>
<td>0.3333</td>
<td>0.2156</td>
<td>0.1708</td>
<td>0.3033</td>
<td>0.1862</td>
</tr>
<tr>
<td></td>
<td>(0.0307)</td>
<td>(0.0429)</td>
<td>(0.0667)</td>
<td>(0.0347)</td>
<td>(0.0423)</td>
<td>(0.0468)</td>
<td>(0.0343)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2552</td>
<td>0.2950</td>
<td>0.3012</td>
<td>0.2946</td>
<td>0.2778</td>
<td>0.2832</td>
<td>0.2766</td>
</tr>
<tr>
<td></td>
<td>(0.0270)</td>
<td>(0.0388)</td>
<td>(0.0286)</td>
<td>(0.0314)</td>
<td>(0.0384)</td>
<td>(0.0265)</td>
<td>(0.0292)</td>
</tr>
<tr>
<td>german</td>
<td>0.2839</td>
<td>0.3151</td>
<td>0.3215</td>
<td>0.3211</td>
<td>0.2988</td>
<td>0.3092</td>
<td>0.3091</td>
</tr>
<tr>
<td></td>
<td>(0.0211)</td>
<td>(0.0334)</td>
<td>(0.0226)</td>
<td>(0.0314)</td>
<td>(0.0384)</td>
<td>(0.0265)</td>
<td>(0.0238)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.0649</td>
<td>0.0437</td>
<td>0.0715</td>
<td>0.0806</td>
<td>0.0410</td>
<td>0.0694</td>
<td>0.0772</td>
</tr>
<tr>
<td></td>
<td>(0.0034)</td>
<td>(0.0182)</td>
<td>(0.0055)</td>
<td>(0.0071)</td>
<td>(0.0177)</td>
<td>(0.0036)</td>
<td>(0.0047)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.2034</td>
<td>0.1985</td>
<td>0.2670</td>
<td>0.2383</td>
<td>0.1837</td>
<td>0.2492</td>
<td>0.1970</td>
</tr>
<tr>
<td></td>
<td>(0.0551)</td>
<td>(0.0616)</td>
<td>(0.0621)</td>
<td>(0.0666)</td>
<td>(0.0436)</td>
<td>(0.0489)</td>
<td>(0.0577)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.0663</td>
<td>0.0786</td>
<td>0.1822</td>
<td>0.1538</td>
<td>0.0761</td>
<td>0.1530</td>
<td>0.1224</td>
</tr>
<tr>
<td></td>
<td>(0.0128)</td>
<td>(0.0184)</td>
<td>(0.0166)</td>
<td>(0.0180)</td>
<td>(0.0179)</td>
<td>(0.0183)</td>
<td>(0.0146)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0394</td>
<td>0.0326</td>
<td>0.0559</td>
<td>0.0527</td>
<td>0.0292</td>
<td>0.0536</td>
<td>0.0526</td>
</tr>
<tr>
<td></td>
<td>(0.0060)</td>
<td>(0.0076)</td>
<td>(0.0058)</td>
<td>(0.0075)</td>
<td>(0.0056)</td>
<td>(0.0055)</td>
<td>(0.0068)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2108</td>
<td>0.2671</td>
<td>0.2653</td>
<td>0.2551</td>
<td>0.2425</td>
<td>0.2288</td>
<td>0.2390</td>
</tr>
<tr>
<td></td>
<td>(0.0286)</td>
<td>(0.0344)</td>
<td>(0.0322)</td>
<td>(0.0265)</td>
<td>(0.0352)</td>
<td>(0.0275)</td>
<td>(0.0255)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0578</td>
<td>0.0590</td>
<td>0.0905</td>
<td>0.0728</td>
<td>0.0529</td>
<td>0.0844</td>
<td>0.0682</td>
</tr>
<tr>
<td></td>
<td>(0.0158)</td>
<td>(0.0269)</td>
<td>(0.0233)</td>
<td>(0.0194)</td>
<td>(0.0250)</td>
<td>(0.0232)</td>
<td>(0.0198)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0336</td>
<td>0.0545</td>
<td>0.0862</td>
<td>0.0507</td>
<td>0.0469</td>
<td>0.0876</td>
<td>0.0453</td>
</tr>
<tr>
<td></td>
<td>(0.0139)</td>
<td>(0.0181)</td>
<td>(0.0261)</td>
<td>(0.0218)</td>
<td>(0.0152)</td>
<td>(0.0353)</td>
<td>(0.0198)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1551</td>
<td>0.1749</td>
<td>0.2185</td>
<td>0.1919</td>
<td>0.1612</td>
<td>0.2025</td>
<td>0.1774</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>8/2/2</td>
<td>12/0/0</td>
<td>12/0/0</td>
<td>7/2/3</td>
<td>12/0/0</td>
<td>11/0/1</td>
<td></td>
</tr>
</tbody>
</table>

**MT-$k$NN:** Multi-Train with three base learners, which are CSO-$k$NN, PCA-$k$NN and NONE-$k$NN.
**TT-CSO-$k$NN:** tri-training with learners of CSO-$k$NN.
**TT-PCA-$k$NN:** tri-training with learners of PCA-$k$NN.
**TT-NONE-$k$NN:** tri-training with learners of NONE-$k$NN.
**CSO-$k$NN:** single classifier that learning model is $k$NN and features are selected by CSO-based algorithm.
**PCA-$k$NN:** single classifier that learning model is $k$NN and features are transformed by PCA.
**NONE-$k$NN:** single classifier that learning model is $k$NN and all features are used.

The results show that MT-Hybrid has the lowest average error rate. The statistical tests also confirm that MT-Hybrid outperforms other methods, except for one setting using MT-CSO and MT-NONE and two using MT-J48. These findings re-confirm that heterogeneous ensembles have better generalisation ability.
Table 6.9: The classification error rate of Multi-Train with different heterogeneous base learner sets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MT-Hybrid</th>
<th>MT-CSO</th>
<th>MT-PCA</th>
<th>MT-NONE</th>
<th>MT-RT</th>
<th>MT-NB</th>
<th>MT-J48</th>
<th>MT-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>0.1102</td>
<td>0.1285</td>
<td>0.1528</td>
<td>0.1277</td>
<td>0.1426</td>
<td>0.1719</td>
<td>0.1329</td>
<td>0.1243</td>
</tr>
<tr>
<td></td>
<td>(0.0183)</td>
<td>(0.0181)</td>
<td>(0.0353)</td>
<td>(0.0205)</td>
<td>(0.0219)</td>
<td>(0.0361)</td>
<td>(0.0229)</td>
<td>(0.0187)</td>
</tr>
<tr>
<td>bupa</td>
<td>0.3586</td>
<td>0.3586</td>
<td>0.3939</td>
<td>0.3337</td>
<td>0.3866</td>
<td>0.4429</td>
<td>0.3567</td>
<td>0.3663</td>
</tr>
<tr>
<td></td>
<td>(0.0506)</td>
<td>(0.0613)</td>
<td>(0.0420)</td>
<td>(0.0423)</td>
<td>(0.0528)</td>
<td>(0.0523)</td>
<td>(0.0526)</td>
<td>(0.0465)</td>
</tr>
<tr>
<td>colic</td>
<td>0.1228</td>
<td>0.1493</td>
<td>0.2543</td>
<td>0.1402</td>
<td>0.1967</td>
<td>0.1837</td>
<td>0.1580</td>
<td>0.1551</td>
</tr>
<tr>
<td></td>
<td>(0.0260)</td>
<td>(0.0319)</td>
<td>(0.0517)</td>
<td>(0.0332)</td>
<td>(0.0460)</td>
<td>(0.0451)</td>
<td>(0.0357)</td>
<td>(0.0307)</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.2224</td>
<td>0.2466</td>
<td>0.2595</td>
<td>0.2394</td>
<td>0.2583</td>
<td>0.2530</td>
<td>0.2566</td>
<td>0.2552</td>
</tr>
<tr>
<td></td>
<td>(0.0232)</td>
<td>(0.0315)</td>
<td>(0.0261)</td>
<td>(0.0201)</td>
<td>(0.0293)</td>
<td>(0.0247)</td>
<td>(0.0349)</td>
<td>(0.0270)</td>
</tr>
<tr>
<td>german</td>
<td>0.2583</td>
<td>0.2729</td>
<td>0.2911</td>
<td>0.2687</td>
<td>0.2927</td>
<td>0.2715</td>
<td>0.2960</td>
<td>0.2839</td>
</tr>
<tr>
<td></td>
<td>(0.0177)</td>
<td>(0.0251)</td>
<td>(0.0190)</td>
<td>(0.0209)</td>
<td>(0.0235)</td>
<td>(0.0254)</td>
<td>(0.0256)</td>
<td>(0.0211)</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.0538</td>
<td>0.0724</td>
<td>0.0724</td>
<td>0.0454</td>
<td>0.0483</td>
<td>0.0496</td>
<td>0.0262</td>
<td>0.0649</td>
</tr>
<tr>
<td></td>
<td>(0.0067)</td>
<td>(0.0070)</td>
<td>(0.0066)</td>
<td>(0.0088)</td>
<td>(0.0113)</td>
<td>(0.0091)</td>
<td>(0.0166)</td>
<td>(0.0034)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.0583</td>
<td>0.0966</td>
<td>0.0807</td>
<td>0.0837</td>
<td>0.0951</td>
<td>0.1258</td>
<td>0.1091</td>
<td>0.2034</td>
</tr>
<tr>
<td></td>
<td>(0.0233)</td>
<td>(0.0430)</td>
<td>(0.0291)</td>
<td>(0.0356)</td>
<td>(0.0298)</td>
<td>(0.0520)</td>
<td>(0.0348)</td>
<td>(0.0551)</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.0328</td>
<td>0.0436</td>
<td>0.1615</td>
<td>0.0475</td>
<td>0.0486</td>
<td>0.1127</td>
<td>0.0327</td>
<td>0.0663</td>
</tr>
<tr>
<td></td>
<td>(0.0082)</td>
<td>(0.0107)</td>
<td>(0.0246)</td>
<td>(0.0107)</td>
<td>(0.0097)</td>
<td>(0.0219)</td>
<td>(0.0070)</td>
<td>(0.0128)</td>
</tr>
<tr>
<td>sick</td>
<td>0.0268</td>
<td>0.0243</td>
<td>0.0519</td>
<td>0.0274</td>
<td>0.0358</td>
<td>0.0662</td>
<td>0.0215</td>
<td>0.0394</td>
</tr>
<tr>
<td></td>
<td>(0.0047)</td>
<td>(0.0063)</td>
<td>(0.0067)</td>
<td>(0.0054)</td>
<td>(0.0059)</td>
<td>(0.0254)</td>
<td>(0.0041)</td>
<td>(0.0060)</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>0.2157</td>
<td>0.2353</td>
<td>0.2379</td>
<td>0.2160</td>
<td>0.2547</td>
<td>0.2733</td>
<td>0.2361</td>
<td>0.2108</td>
</tr>
<tr>
<td></td>
<td>(0.0217)</td>
<td>(0.0286)</td>
<td>(0.0249)</td>
<td>(0.0218)</td>
<td>(0.0329)</td>
<td>(0.0198)</td>
<td>(0.0304)</td>
<td>(0.0286)</td>
</tr>
<tr>
<td>vote</td>
<td>0.0394</td>
<td>0.0387</td>
<td>0.0526</td>
<td>0.0399</td>
<td>0.0502</td>
<td>0.0430</td>
<td>0.0542</td>
<td>0.0336</td>
</tr>
<tr>
<td></td>
<td>(0.0131)</td>
<td>(0.0140)</td>
<td>(0.0204)</td>
<td>(0.0116)</td>
<td>(0.0163)</td>
<td>(0.0151)</td>
<td>(0.0228)</td>
<td>(0.0139)</td>
</tr>
<tr>
<td>wdbc</td>
<td>0.0300</td>
<td>0.0387</td>
<td>0.0526</td>
<td>0.0399</td>
<td>0.0502</td>
<td>0.0430</td>
<td>0.0542</td>
<td>0.0336</td>
</tr>
<tr>
<td></td>
<td>(0.0131)</td>
<td>(0.0140)</td>
<td>(0.0204)</td>
<td>(0.0116)</td>
<td>(0.0163)</td>
<td>(0.0151)</td>
<td>(0.0228)</td>
<td>(0.0139)</td>
</tr>
<tr>
<td>avg</td>
<td>0.1274</td>
<td>0.1386</td>
<td>0.1739</td>
<td>0.1353</td>
<td>0.1551</td>
<td>0.1693</td>
<td>0.1443</td>
<td>0.1551</td>
</tr>
<tr>
<td>win/lose/tie</td>
<td>8/1/3</td>
<td>12/0/0</td>
<td>12/0/0</td>
<td>11/0/1</td>
<td>11/0/1</td>
<td>8/2/2</td>
<td>9/0/3</td>
<td>11/0/1</td>
</tr>
</tbody>
</table>

MT-Hybrid: Multi-Train with 12 base learners, which are CSO-RT, CSO-NB, CSO-J48, CSO-\text{kNN}, PCA-RT, PCA-NB, PCA-J48, PCA-\text{kNN}, NONE-RT, NONE-NB, NONE-J48 and NONE-\text{kNN}.

MT-CSO: Multi-Train with four base learners, which are CSO-RT, CSO-NB, CSO-J48, and CSO-\text{kNN}.

MT-PCA: Multi-Train with four base learners, which are PCA-RT, PCA-NB, PCA-J48, and PCA-\text{kNN}.

MT-NONE: Multi-Train with four base learners, which are NONE-RT, NONE-NB, NONE-J48, and NONE-\text{kNN}.

MT-RT: Multi-Train with three base learners, which are CSO-RT, PCA-RT and NONE-RT.

MT-NB: Multi-Train with three base learners, which are CSO-NB, PCA-NB and NONE-NB.

MT-J48: Multi-Train with three base learners, which are CSO-J48, PCA-J48 and NONE-J48.

MT-\text{kNN}: Multi-Train with three base learners, which are CSO-\text{kNN}, PCA-\text{kNN} and NONE-\text{kNN}.

6.5 Summary

We propose a new ensemble based SSL algorithm in this chapter, namely Multi-Train. By comparing it with the tri-training algorithm and non-SSL learning models, we show that the proposed Multi-Train algorithm outperforms the compared algorithms. The better performance can be attributed to the multiple views generated using different models as well as different feature manipulation meth-
ods in contrast to the original single-view data. Furthermore, by using ensemble method, the prediction accuracy on the unlabelled data is improved, which therefore is able to reduce the risk of incorrectly labelling the unlabelled data [110,128]. Our results confirm that the heterogeneous ensembles, which consist of different types of based models and use different features, have superior generalisation performance.

As shown in some scenarios, one base learner in the Multi-Train performs significantly better or worse than other base learners. It is therefore of interest to assign a larger weight to those good base learners while prune the poor ones, which can potentially further increase the generalisation ability of Multi-Train.
CHAPTER 7

Conclusion and Future Work

7.1 Summary and conclusions

We presented a review of multi-objective ensemble generation in Chapter 2. After a short introduction to the basics of machine learning ensembles and evolutionary MOO, we focused on the objectives that can be used for generating ensembles and methods for selecting a subset of members from the Pareto optimal set generated by the MOO algorithm.

Although diversity has been considered to be important for enhancing the performance of ensembles, we point out that quantitatively defining diversity is non-trivial and the role of diversity remains to be understood. We also indicate that although the multi-objective approach can remove the burden of predefining the accuracy diversity trade-off before learning, selecting a subset of Pareto optimal members or selecting a Pareto optimal ensemble that can generalise well
on unseen data is non-trivial too. Nevertheless, we must stress that despite all challenges in the multi-objective approach in ensemble generation, it does offer a new perspective in machine learning. On the other hand, many questions remain open for future work.

The performance of classifier ensembles is compared with that of individual base classifiers for EEG signal classification in Chapter 3. The findings confirm that classifier ensembles produce statistically higher accuracies on testing data than single classifiers. Furthermore, by comparing heterogeneous classifier ensembles using the same feature or using different features, the results demonstrate that ensembles using different features give higher classification rate than those using the same feature, which indicates that heterogeneous ensembles can further increase diversity without impairing the accuracy of individual base classifiers. We also tested two ways of ensembling classifiers, in one all trained base classifiers are used and in the other, only a few base classifiers having the best training accuracies are used in the ensemble. According to our experiment results, the latter one gives better classification results, which confirms the findings in [100].

In Chapter 4, we proposed a method that uses the diversity measure as an explicit objective, along with the training accuracy of the ensemble as another objective. Three different diversity measures and two methods for creating ensemble diversity have been tested. Our results indicate that not all methods can reliably result in ensembles that outperform single classifiers. Overall, the diversity creation method that uses different training samples combined with a diversity imposed on the feature space may be most likely to produce the best classification performance.

Although many good solutions could be found in the Pareto front, it is hard to figure out the solution that performs the best on the test dataset. The results suggest that although higher diversity may result in better ensemble performance, the classification accuracies of the ensemble members are still more important
than the diversity of the ensemble, therefore, the trade-off between these two objectives should be the biggest matter. The results of this work also show that the best solutions that have the highest test accuracy are mostly in the top 10 per cent of the Pareto optimal solutions having the highest training accuracy.

Chapter 5 aims to propose an efficient feature selection algorithm to select a small feature subset from a large number of features while maintaining similar or even better classification performance than using all features. The objective has been successfully achieved by adapting the CSO algorithm to feature selection.

Our experimental results demonstrate that the proposed CSO-based method outperforms the conventional PCA-based method, the PSO-based method, and a few recently reported PSO variants based methods on all large-scale datasets with a significant margin. Furthermore, we find that the number of new solutions found by the CSO-based method is considerably larger than that found by the PSO-based methods. Unlike the PSO-based methods whose final optimal results heavily rely on the initialisation in terms of the number of selected features, the proposed method performs consistently well and is insensitive to initialisation.

Finally, we proposed a new SSL ensemble algorithm in Chapter 6, namely Multi-Train. By comparing with the tri-training algorithm and non-SSL learners, Multi-Train shows better generalisation ability than other methods because it generates multiple views from the original single-view data, which is a key criterion of building SSL classifiers. Furthermore, by using ensemble method, the accuracy of prediction of unlabelled data is improved, which is, therefore, able to reduce the risk of labelling the unlabelled data wrongly [110,128]. By solving these two issues in traditional SSL algorithms, the proposed algorithm has better generalisation ability.

Furthermore, we also compared the different ways of generating diversities, including artificially created views by manipulating the feature, using more classifier models or a combination of both. The results show that the combination has
lower classification errors than others. This confirms that heterogeneous ensembles have superior generalisation performance.

This thesis has presented how to use classifier ensembles to improve the generalisation ability in classifier ensembles, especially with partially labelled data with a large number of features. The early work focused on traditional classifier ensemble methods, followed by moving on to multi-objective perspective. By the help of a CSO-based feature selection algorithm on large-scale data, we finally proposed a novel semi-supervised ensemble learning algorithm that outperforms state of art algorithms.

7.2 Future directions

In this thesis, we focused on heterogeneous classifier ensembles in both multi-objective and semi-supervised approaches. However, we could not possibly cover everything in this thesis. Below are a few thoughts that could be considered in future research. For example, as in some scenario, one member in the Multi-Train shows significantly better or worse performance than other members, in the future, we could find some measures to give higher weights to those good members while prune the poor ones. This could potentially further increase the generalisation ability of Multi-Train. Other possible directions are as follows:

How to handle imbalanced classes?

It is not always easy, if not impossible, to obtain data that contain evenly distributed classes. For example, in many medical applications, classes are defined as positive and negative to a certain disease. It is commonly seen that a dataset only contains a few positive samples while the majorities are negative. Even worse, a false negative prediction could be much more severe than a false positive prediction, which the cost of getting false predictions in either class are vastly
different. There has a number of existing accuracy measures that specifically designed for addressing this problem, such as confusion matrix, precision, and recall [219], cost matrix, receiver operating characteristic (ROC) curves [220] and cost curves [221]. Some of those measures give more than one accuracy values, others are seen to fuse the accuracies from each class. This brings up a few thoughts in classifier ensembles, especially in multi-objective approach.

Other than using imbalanced data directly in training, it is also possible to re-sample the original data to balance the classes. Those methods are seen as over-sampling and down-sampling, while the idea is to train with more samples (with replacement) from the minority class and fewer samples from the majority class. Apart from this, generating some synthetic samples in the minority class is another possible solution.

What is the best second objective of the multi-objective ensemble?

There is no doubt that both accuracy and diversity should be taken into account in ensemble generation. It is also evident that accuracy is an indispensable objective. It is, however, much less clear which objective should be used as the second objective. From the literature, diversity seems to be an arguable objective, not only because there is no widely accepted measure of diversity, but also because a larger degree of diversity necessarily means better performance for ensembles. Therefore, it is of great interest for future research to investigate other measures rather than the diversity and can be used as the second objective. One potential measure is complexity, which may be used both for alleviating overfitting and for ensuring structural diversity.

How to select members from the Pareto optimal solutions?

The multi-objective approach to ensemble member generation can result in multiple Pareto optimal members, a subset of which can be used to construct ensembles. By contrast, in the multi-objective ensemble generation approach, multiple
Pareto optimal solutions, each of which is an ensemble, will be generated. In both cases, a subset of the Pareto optimal solutions from the Pareto front needs to be selected. This can also be seen as a model selection problem, although it is hoped that by having a set of Pareto optimal solutions, model selection will become less tricky. Several selection methods have been investigated so far, and it appears that selecting solutions near the knee point or a subset that has the largest diversity is recommendable. Without a clear idea of how to select a subset of the Pareto optimal solutions, the significance of the multi-objective approach will be considerably reduced, and much work remains to be done along this line of research.


135


[148] S. Reid, “A review of heterogeneous ensemble methods,” *Department of

tering of single trial EEG during imagined hand movement,” *IEEE Trans-
actions on Rehabilitation Engineering*, vol. 8, no. 4, pp. 441–446, 2000.

[150] L. F. Nicolas-Alonso and J. Gomez-Gil, “Brain computer interfaces, a re-

computer interface (BCI): model order selection for autoregressive spectral
analysis,” *Multiple Classifier Systems 8th International Workshop, MCS

four-class motor imagery EEG data for the BCI-competition 2005,” *Multi-
ple Classifier Systems 8th International Workshop, MCS 2009, Reykjavik,

imagery EEG recordings with subject adapted non-dyadic arbitrary time–
frequency tilings,” *Multiple Classifier Systems 8th International Work-
shop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings*, vol. 3,
pp. 235–244, July 2006.

of EEG activity into cursor movement by a brain-computer interface


