공학박사학위논문

Heterogeneous Ensemble Learning for Multi-class Classification

다중분류 문제를 위한 이질적 앙상블 학습

2015 년 8 월

서울대학교 대학원

산업공학과 데이터마이닝 전공

강 석 호

Heterogeneous Ensemble Learning for Multi-class Classification

다중분류 문제를 위한 이질적 앙상블 학습

지도교수 조 성 준

이 논문을 공학박사학위논문으로 제출함

2015 년 5 월

서울대학교 대학원

산업공학과

강 석 호

강석호의 박사학위논문을 인준함

2015 년 6 월

위 원 장 박 종 헌 (인)

부위원장 조 성 준 (인)

위 원 이 재 욱 (인)

위 원 김 성 범 (인)

위 원 강 필 성 (인)

Abstract

Heterogeneous Ensemble Learning for Multi-class Classification

Seokho Kang
Department of Industrial Engineering
The Graduate School
Seoul National University

In data mining, classification is a type of supervised learning task that involves predicting output variables consisting of a finite number of categories called classes. When the number of classes is larger than two, a classification problem is called a multi-class classification problem. Multi-class classification provides more informative predictions, and is more related to real-world scenarios. In practice, the performance for a multi-class classification problem is typically measured according to the following three perspectives: accurate, reliable, and fast classification. In order to achieve the better performance for the three perspectives, this dissertation proposes to use heterogeneous ensemble learning that exploits multiple classifiers from various classification algorithms, where each classifier plays a different role to accomplish the desired functionality. For accurate multi-class classification, Diversified One-Against-One (DOAO) and Optimally Diversified One-Against-One (ODOAO) are proposed. Their main idea is to decompose the original problem into several binary sub-problems based

on the one-against-one approach. DOAO finds the best classification algorithm for each class pair from the set of heterogeneous base classifiers, thereby makes various classification algorithms to complement each other. Since the best classification algorithm for each class pair is different, DOAO enables better classification accuracy. ODOAO, an extension of DOAO, construct an ensemble where a meta-classifier effectively combines the outputs from all the heterogeneous base classifiers. Heterogeneous Ensemble of One-class Classifiers (HEOC) is also proposed for accurate classification based on decomposition of the original problem into several one-class sub-problems. HEOC constructs an ensemble consisting of one-class classifiers from various one-class classification algorithms. HEOC addresses the normalization of heterogeneous base classifiers via stacking. For reliable multi-class classification, a hybrid reject option is proposed to reject ambiguous instances instead of predicting for all instances. The hybrid reject option constructs a filter classifier and a predictor classifier separately, where the filter decides whether to predict using the predictor based on the confidence for an instance, and the predictor predicts the class of the instance. Each component is trained using the best respective classification algorithm to maximize the capability of its role, thereby improve reject option performance as providing better prediction accuracy for the same degree of rejection. For fast multi-class classification, Neural Network Approximator (NNA) is proposed to reduce computational time in the test phase. NNA approximates a classifier by adopting a multiple-outputs artificial neural network as a function approximator, where each output node corresponds to a decision function in the classifier. This approximator enables fast classification speed without compromising ac-

curacy. The effectiveness of the proposed heterogeneous ensemble methods is demonstrated through experiments on benchmark datasets and real-world applications.

Keywords: Data Mining, Machine Learning, Ensemble, Heterogeneous Ensemble, Multi-class Classification

Student Number: 2011-21163

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Multi-class Classification

Supervised learning is a type of machine learning task of inferring a function of certain variables in order to predict other variables. Classification is a type of supervised learning task that involves predicting output variables consisting of a finite number of categories called classes. In a classification task, a classification algorithm $\mathcal{A}$ defines its hypothesis space $\mathcal{H}_{\mathcal{A}}$. Classifier training is to find the hypothesis $h \in \mathcal{H}_{\mathcal{A}}$ that approximates the true function f given a set of instances called a training dataset, as illustrated in Figure 1.1. Thus, a classifier corresponds to its hypothesis in the hypothesis space. Finding the hypothesis that is closest to the true function f is crucial for obtaining high classification accuracy.

When the number of classes is larger than two, a classification problem is called a multi-class classification problem. Most real-world scenarios, such as handwritten digit recognition, text categorization, and face recognition, correspond to multi-class classification problems. A multi-class classification problem is typically more difficult than a binary-class classification problem. Thus, find-

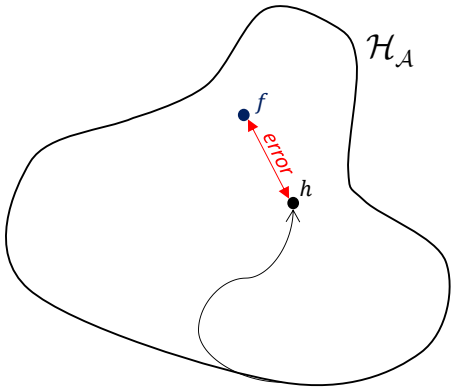
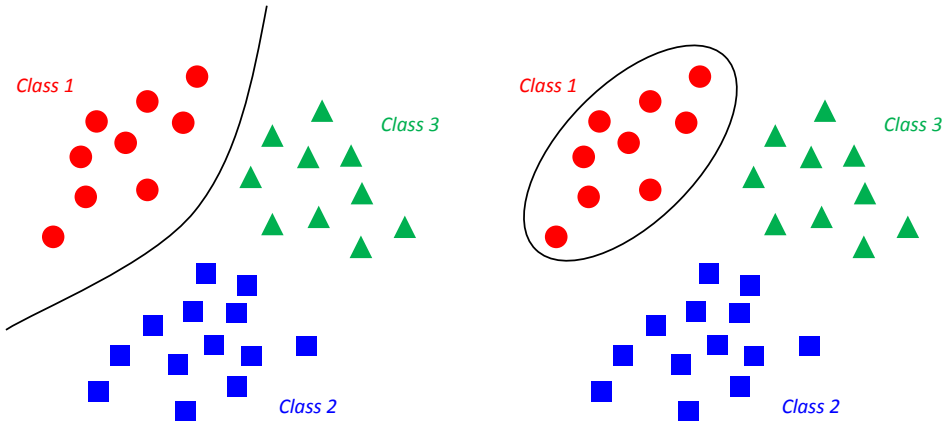


Figure 1.1 Concept of classifier training

ing an appropriate strategy to solve a multi-class classification problem is an important research issue.

In order to solve a multi-class classification problem, three strategies can be considered. The first strategy is to use classification algorithms that are capable of dealing with multi-class classification directly. The second is to construct an ensemble of binary classifiers. The third is to construct an ensemble of one-class classifiers. The latter two can be implemented by decomposing the original multi-class problem into several smaller sub-problems, as illustrated in Figure 1.2.

The decomposition strategies relate to the concept of ensemble learning, thereby treating a multi-class classification problem effectively. Since the decision boundary for multi-class classification problems tends to more complex than it is for one-class or binary classification problems, solving several smaller sub-problems is more preferable (Galar et al., 2011). In addition, heterogeneous ensemble learning can provide better competence because the best classification



(a) A subproblem for the second strategy

(b) A subproblem for the third strategy

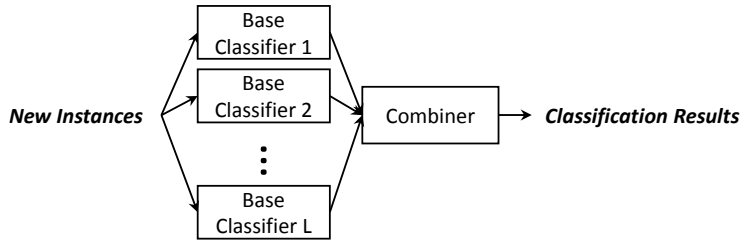
Figure 1.2 Decomposition strategies for multi-class classification

algorithm for each sub-problem is different.

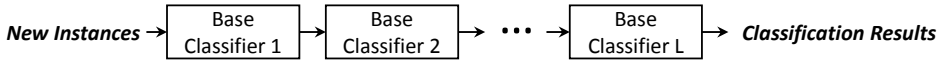
1.2 Ensemble Learning

Ensemble learning aims at combining the outputs from multiple classifiers. It is also known as multiple classifier system, committee learning, and so on. There has been considerable research effort by a wide range of researchers in order to develop ensemble learning methods for different purposes (L. Xu et al., 1992; Ho et al., 1994; Kittler et al., 1998; Rokach, 2010; Woźniak et al., 2014). An ensemble generally outperforms any individual classifiers by exploiting the diversity of different classifier. The diversity of the classifiers can be obtained through a variety of strategies. For example, employing different classification algorithms and manipulating datasets result in different classifiers.

Constructing an ensemble enables dealing with a classification problem more



(a) Parallel ensemble structure



(b) Serial ensemble structure

Figure 1.3 Two basic ensemble structures

efficiently and effectively. Various types of ensembles have been designed to pursue different purposes. The two basic structures are parallel and serial structure (Rokach, 2010), which are illustrated in Figure 1.3. A parallel structure generally involves combination of classifiers with a same functionally in order to obtain more accurate and stable classification performance (Kang, Cho, & Kang, 2015a; Kang & Cho, 2015b; Kang, Cho, & Kang, 2015b). On the other hand, a serial structure exploits classifiers with different functionalities (Kang, Cho, Rhee, & Yu, 2015; Kang & Cho, 2014). This generally combines several classifiers sequentially in order to improve the run-time speed of classification. A hybrid structure is also available to take advantages of these two structures.

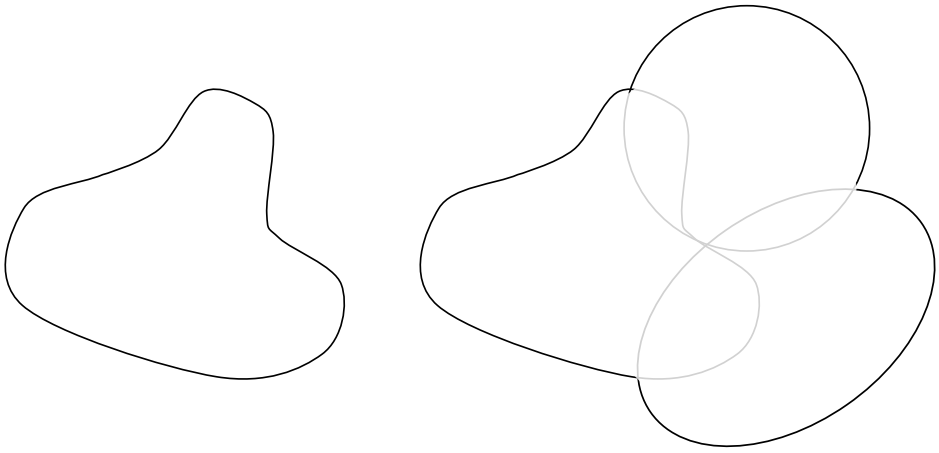
In general, an ensemble offers better classification accuracy and robustness than any individual classifier. Dietterich (2000) explained three fundamental reasons for why an ensemble successfully performs well. The first is a statisti-

cal reason. Given a finite number of training instances, many hypotheses are equally good. Therefore, averaging these hypotheses may result in a more stable approximation of f. The second is a computational reason. Because the hypothesis space is so large, a heuristic search is conducted to find the best hypothesis. However, the search may get stuck at a local optimum. Repeating the search with several random starts provides a better chance of finding the global optimum. The third is a representational reason. The true function f may not be represented by any of the hypotheses in the hypothesis space $\mathcal{H}_{\mathcal{A}}$, but may be better approximated by aggregating several hypotheses.

1.3 Heterogeneous Ensemble Learning

It is well-known that no single algorithm can always perform the best for every classification problem (Sohn, 1999; Lim et al., 2000; Kiang, 2003), which is also known as the no-free-lunch-theorem (Wolpert, 2001). Thus, the heterogeneous ensemble allows us to obtain better classification accuracy by combining the advantages of various algorithms.

Thus, heterogeneous ensemble learning, which employs various classification algorithms to train base classifiers, can be taken into account for further improvement of ensemble learning (Kang, Cho, & Kang, 2015a; Kang & Cho, 2015b; Kang, Cho, & Kang, 2015b; Kang, Cho, Rhee, & Yu, 2015; Kang & Cho, 2014). A heterogeneous ensemble is more likely to obtain a better hypothesis by searching the union of hypothesis spaces defined by different algorithms as shown in Figure 1.4 (Dietterich, 2000), while this generally requires a large



(a) Hypothesis space of an algorithm (b) Extended hypothesis space by various algorithms

Figure 1.4 Hypothesis space of heterogeneous ensemble

computational burden in training and test because it involves numerous base classifiers from various classification algorithms.

Heterogeneous ensemble learning for multi-class classification can be performed by decomposing the original problem into several smaller sub-problems, training base classifiers for the sub-problems using various classification algorithms, and constructing an ensemble of those base classifiers with an appropriate combination method to form a multi-class classifier.

This dissertation proposes to use heterogeneous ensemble learning for improving multi-class classification. A multi-class classification problem can be more accurately solved by constructing a heterogeneous ensemble with a parallel structure of base classifiers, while combining heterogeneous base classifiers is a difficult issue. Thus, we propose methods for accurate classification by dealing

with the heterogeneity effectively when combining the base classifiers. Besides, the reliability and speed of a multi-class classifier are also important for better multi-class classification in practical deployments. Thus, add-on methods based on heterogeneous ensembles with serial structures are proposed for reliable and fast classification. These methods are effective for multi-class classification, but also can be generally applied to any classifier.

1.4 Outlook of this Dissertation

In this dissertation, the following three criteria are addressed as the performance of a multi-class classifier when the classifier is deployed to a multi-class classification problem. The first is how accurately the classifier classifies unseen instances. The second is how reliably the classifier avoids classifying uncertain instances. The third is how fast the classifier classifies a number of instances.

With respect to the three criteria, several heterogeneous ensemble methods are proposed for accurate, reliable, and fast multi-class classification. The heterogeneous ensemble methods utilize multiple classifiers from various classification algorithms by combining their outputs properly in order to pursue the respective purposes. They are compared with homogeneous ensemble methods on benchmark datasets and real-world applications to demonstrate the effectiveness. The methods and applications covered in this dissertation are summarized in Table 1.1.

The rest of this dissertation is organized as follows. In Chapter 2, classification algorithms used in this dissertation are briefly reviewed, and related

Table 1.1 Methods and applications covered in this dissertation

Objective	Chapter	Methods	Applications
Accurate Classification	Chapter 3	• Diversifed One-Against-One • Optimally Diversified One-Against-One	• Benchmark Datasets
	Chapter 4	• Heterogeneous Ensemble of One-class Classifiers	• Benchmark Datasets • Text Categorization
Reliable Classification	Chapter 5	• Hybrid Reject Option	• Anti-diabetic Drug Failrue Prediction
Fast Classification	Chapter 6	• Neural Network Approximator	• Benchmark Datasets • Semiconductor Die Failrue Prediction

work on ensemble learning for multi-class classification is introduced. Chapter 3 and 4 focus on developing heterogeneous ensemble methods for accurate multi-class classification. In Chapter 3, two one-against-one based methods utilizing heterogeneous binary classifiers from various classification algorithms are introduced to persue better classification accuracy. In Chapter 4, a heterogeneous ensemble based on one-class classifier approach is introduced, and the effectiveness of this method is investigated for the text categorization problem. In Chapter 5, a hybrid reject option is proposed to support reliable prediction by improving the trade-off between accuracy and rejection, and is applied to the anti-diabetic drug failure prediction problem. In Chapter 6, a classifier approximation method based on neural networks is proposed to accelerate the run-time speed of a multi-class classifier, and its effectiveness is confirmed on the semi-

conductor die failure prediction problem. Finally, we discuss contributions and future work of this dissertation in Chapter 7.

Chapter 2

Literature Review

2.1 Classification Algorithms

Heterogeneous ensemble learning involves training of diverse classifiers using various classification algorithms. In this section, classification algorithms and one-class classification algorithms used in the dissertation are introduced.

Classification is a supervised learning task of predicting output variables consisting of a finite number of categories called classes. A classifier trained by a classification algorithm predicts the class of new instances according to the posterior probability of each class. In this dissertation, a total of six well-known and widely adopted classification algorithms are used in reference to (Bishop & Nasrabadi, 2006; Wu et al., 2008; Hastie et al., 2009). The following are brief descriptions of the algorithms.

For Artificial Neural Network (ANN), the most general architecture is the Multi-layer Perceptron that consists of three layers of several processing units: the input layer, the hidden layer, and the output layer. The layers are connected with other layers through non-linear functions of linear combinations. ANN seeks to find the best weights to minimize the sum of the squared error on the

validation dataset. Back propagation algorithms are generally used in order to derive the weights.

Decision Tree (DT) induces sets of rules in the hierarchical structure of several nodes via recursive partitioning of the data. Each partitioning is performed by selecting the input variable that mostly separates the classes of the data. This recursive process is continued until the termination condition is met. The most popular DT algorithms are the CART and the C4.5. In this dissertation, the CART is used for training of DTs. This algorithm employs the Gini index as a splitting criterion to divide the data into two regions.

k-Nearest Neighbors (kNN) is an instance-based learning algorithm that does not require any training of models. kNN finds the k instances within the training dataset that are the closest in distance to the test instance. Classification is done by a vote of the selected k instances.

Linear Discriminant Analysis (LDA) is based on the directions that are the best discrimination of the data in different classes. LDA attempts to find a linear combination of input variables that maximizes the ratio of the between-class scatter to the within-class scatter, and classifies based on the linear combination.

Logistic Regression (LR) forms a logistic function of a linear combination of input variables whose output is in the range of $[0, 1]$. The best weights of the combination can be estimated using the maximum likelihood methods.

Support Vector Machine (SVM) is originally designed for binary classification, and seeks to find the maximum margin hyperplane that separates one class from another class. It can deal with non-linear classification problems by

employing kernel functions that map an input space into a high-dimensional feature space. SVM forms a convex optimization problem that can be solved efficiently through sequential minimal optimization.

Among those algorithms, SVM, LR, and LDA are originally designed for binary classification, while kNN, DT, and ANN can deal with multi-class classification directly.

On the other hand, one-class classification is an unsupervised learning task where only one of the classes is utilized and other instances are ignored (Tax, 2001). A one-class classifier, trained by a one-class classification algorithm, determines the score of belonging to the target class for new instances, and the instances whose scores are out of the pre-defined threshold are rejected as outliers. In this dissertation, the eight well-known, widely used one-class classification algorithms are utilized. Brief descriptions for the algorithms are given below.

Density-based algorithms aim to estimate the underlying density function of the data. In these algorithms, the instances that possess a density lower than a given threshold will be rejected as outliers. Gaussian (GAUSS) assumes the distribution of the data as a Gaussian distribution. Mixture of Gaussian (MOG) models the data as a linear combination of several Gaussians. Parzen Window (PARZEN) generates Gaussians for every individual instance in the data and combines them to obtain the final density function.

Boundary-based algorithms aim to obtain the decision boundary in which the data is contained, and the instances outside this boundary will be rejected as outliers. In Nearest Neighbor Data Description (NNDD), a new instance is

classified based on the distance to its nearest neighbor. Support Vector Data Description (SVDD) observes the smallest hypersphere enclosing the data in the feature space, and is capable of dealing with non-linear structures by introducing kernel functions.

Reconstruction-based algorithms aim to encode the structure of the data. The instances having high reconstruction error will be rejected as outliers. k-Means (KMEANS) describes the data by k clusters, and the reconstruction error is defined as the distance to the closest cluster center. Principal Component Analysis (PCA) finds the directions of maximum variance for the data, which is done by eigenvalue decomposition. Auto-encoder Network (AUTOENC) involves training a neural network that reproduces the input layer at the output layer. In the case of the latter two, the reconstruction error is the difference between the original instance and its mapped version.

Note that, it is difficult to state that a classification algorithm is better than another algorithm. The best classification algorithm can be different depending on the conditions or characteristics of a classification problem. Firstly, the best algorithm is different for each dataset (Sohn, 1999; Lim et al., 2000; Kiang, 2003). The best algorithm also varies from instance to instance within a dataset (Woods et al., 1997; Cavalin et al., 2013). In addition, giving variety to a dataset, such as sampling, partitioning, and decomposition, also makes the best algorithm to be different (Rokach, 2010). Thus, the heterogeneous ensemble learning of employing different classification algorithms provides the great opportunity to solve classification problems more effectively.

2.2 Ensemble Learning for Multi-class Classification

2.2.1 Decomposition Strategies

When the number of classes in a classification problem is more than two, the problem is called a multi-class classification problem. To solve a multi-class classification problem, three strategies can be considered. The first is simply to use the classification algorithms that solve the problem directly, such as DT, kNN, and ANN.

The second strategy is to decompose the original problem into several binary sub-problems and to construct an ensemble of binary classifiers for the sub-problems. This strategy permits the use of classification algorithms that were originally designed for binary classification, such as SVM, LR, and LDA. Two common approaches to this strategy are one-against-one and one-against-rest (Rokach, 2010; Lorena et al., 2008). Supposing that a c-class classification problem is given, the one-against-one approach builds $c(c-1)/2$ different binary classifiers for all possible class pairs. Given the same problem, on the other hand, the one-against-rest approach builds c different binary classifiers, where each separates a single class from all the remaining classes. These approaches can be generalized by Error-Correcting Output Coding (ECOC), which is a general framework for decomposing a multi-class problem into several binary problems by exploiting more diverse bipartitions of the classes (Dietterich & Bakiri, 1995).

The third strategy is to decompose the original problem into several one-class sub-problems and to construct an ensemble of one-class classifiers for the

sub-problems (Juszczak & Duin, 2004; Ban & Abe, 2006; Tax & Duin, 2008; Krawczyk et al., 2014; Cyganek, 2012; Sharma et al., 2012). This strategy utilizes one-class classification algorithms, such as PARZEN, SVDD, and PCA. For a c-class classification problem, c one-class classifiers are built, each of which is trained on a single respective class. Each classifier evaluates the degree of belonging to a class independently. Thus, an instance can be classified as belonging to a class simply by selecting the class label with the maximum score value among the classifier.

Among these strategies, the second strategy is most popular and has been widely used. Most studies on this strategy have been conducted by employing SVM as base classifiers because of its superiority in binary classification problems (Lorena et al., 2008; Kang & Cho, 2015a). For SVM, it is known that the one-against-one approach generally performs better than the one-against-rest and other SVM-based multi-class classification algorithms (Galar et al., 2011; Hsu & Lin, 2002; Duan & Keerthi, 2005).

The decomposition strategy has proved successful not only for SVM but also for other classification algorithms. Moreover, this strategy is often effective for classification algorithms that can deal with multi-class classification problems directly. Since the decision boundary for multi-class classification problems tends to more complex than it is for one-class or binary classification problems, solving several smaller sub-problems is more preferable. Galar et al. (2011) reviewed the effectiveness of the decomposition strategy for various classification algorithms, and confirmed that the one-against-one approach yields better classification accuracy compared to the one-against-rest approach in most cases.

Knerr et al. (1990) adopted the one-against-one approach for ANN in order to solve multi-class classification problems. Fürnkranz (2002) and Polat and Güneş (2009) applied the one-against-one and one-against-rest approach, respectively, to DT algorithms.

2.2.2 Combination Strategies

An ensemble is composed of diverse base classifiers by offering diversity to the classifiers. The classification results are different depending on the combination strategy, despite having the same set of base classifiers. Therefore, choosing the most appropriate combination strategy for an ensemble is an important issue. There have been various combination methods proposed (Rokach, 2010; Lorena et al., 2008), and the two basic strategies are known as classifier selection and classifier fusion (Tsoumakas et al., 2005; Kuncheva, 2002).

Classifier selection finds the best classifier from among a set of base classifiers. The assumption in classifier selection is that each classifier is an expert in some conditions. Classifier selection generally works well if some classifiers are superior or inferior to others, particularly with heterogeneous base classifiers that are come from different classification algorithms.

Classifier fusion, by contrast, utilizes the group consensus of the whole base classifiers, and therefore it depends on the comparable success of the base classifiers. Classifier fusion is generally used for combining homogeneous base classifiers. Majority voting is a simple but the most popular method, which finds the largest selected class from the base classifiers. However, this method may fail when the majority of base classifiers provide incorrect classification results

(Kuncheva et al., 2003). Instead, weighted voting puts more weights for more superior base classifiers (Wozniak & Jackowski, 2009).

The abovementioned strategies use linear combination of the outputs from the base classifiers to make final decision. On the other hand, meta-learning enables exploiting non-linear combination of base classifiers. Meta-learning (Vilalta & Drissi, 2002) is to induce which classifiers are reliable and which are not, and is usually employed to combine classifiers from different classification algorithms. The basic idea of meta-learning is to build a meta-classifier that predicts target labels by combining the predictions of base classifiers (Wolpert, 1992). Suppose that a set of base classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_L$ and a set of instances $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^N$ is given, the predictions for the N instances of each base classifiers are $\hat{y}_t^i = \mathcal{C}_i(\boldsymbol{x}_t)$, $t = 1, \ldots, N$, $i = 1, \ldots, L$, and they constitutes a meta-dataset $\mathcal{M} = \{(\hat{y}_t^1, \hat{y}_t^2, \ldots, \hat{y}_t^L), y_t\}_{t=1}^N$ (Džeroski & Ženko, 2004). This meta-dataset is used to train the meta-classifier. During the test phase, a test instance is first classified with the base classifiers, and the meta-classifier then gives the final classification result by combining the predictions from the respective base classifiers.

Note that, the instances used to train the base classifiers should not be used during the training of the meta-classifier in order to avoid overfitting. Partitioning the original dataset into a training dataset and a validation dataset is recommended. This ensures that the base classifiers are only trained exclusively with the training dataset and that the meta-classifier is trained based on the validation dataset (Ting & Witten, 1999; Rokach, 2010).

There have been proposed many studies related to meta-learning, and they

generally aim at combining heterogeneous classifiers. Merz (1999) proposed a method called SCANN based on correspondence analysis and nearest neighbor. Todorovski and Džeroski (2003) used DT to train the meta-classifier. Ting and Witten (1999) introduced stacking with Multi-response Linear Regression (MLR), and Džeroski and Ženko (2004) and Seewald (2002) extended this method. Kim et al. (2003) used an SVM as a meta-classifier to combine the bagging of SVMs. Some researchers exploited stacking to combine binary base classifiers of class pairs that are based on the one-against-one approach. Savicky and Fürnkranz (2003) used Ripper, DT, and nearest neighbor as meta classifiers to combine binary Rippers, Lézoray and Cardot (2008) used DT to combine binary ANNs. Menahem et al. (2009) proposed a three-layer architecture based on LR.

Chapter 3

Heterogeneous Ensemble for Accurate Classification: Binary Classifier Approach

3.1 Binary Classifier Approach for Multi-class Classification

The concept of ensemble learning has been successfully applied to multi-class classification problems. This is typically accomplished by decomposing the original problem into several binary sub-problems. The base classifiers for the sub-problems constitute an ensemble. Regarding this decomposition strategy, the two commonly used approaches are one-against-one and one-against-rest (Rokach, 2010; Lorena et al., 2008). For the one-against-one approach, $c(c-1)/2$ different binary classifiers are built for all possible pairs of classes, whereas the one-against-rest approach builds c different classifiers, each of which distinguishes a single class from all the remaining classes. Once the binary classifiers are built for each approach, various combination methods can be used for aggregating their outputs (Rokach, 2010).

Several experimental studies argued that the one-against-one approach outperforms the one-against-rest approach (Galar et al., 2011; Hsu & Lin, 2002), and that such decomposition strategy is also effective for classification algorithms that are capable of dealing with multi-class classification problems directly (Fürnkranz, 2002; Knerr et al., 1990; Polat & Güneş, 2009).

Focusing on the one-against-one approach, it is essential for each binary classifier of sub-problems to be reasonably well performing; otherwise, non-competent classifiers could negatively affect the entire classification results (Galar et al., 2013; Lorena et al., 2008). Another important point is that the best classification algorithm for each sub-problem can be different, because the sub-problems consist of different instances. Employing a variety of classification algorithms takes the advantages of different inductive biases of the algorithms, thereby yielding better classification accuracy. Such effectiveness can be also explained as the extension of the hypothesis space. A heterogeneous ensemble with various classification algorithms is more likely to obtain a better hypothesis by searching the union of hypothesis spaces defined by different algorithms.

In this respect, the two heterogeneous ensemble methods called Diversified One-Against-One (DOAO) (Kang, Cho, & Kang, 2015a) and Optimally Diversified One-Against-One (ODOAO) (Kang & Cho, 2015b) are proposed in this chapter. In Section 3.2, we propose DOAO method that seeks to find the best classification algorithm for each class pair when applying the one-against-one approach. For a multi-class classification problem, an ensemble is constructed based on the one-against-one approach by using classifiers derived by different classification algorithms. Given a training dataset of a c-class classification

problem, DOAO first builds a number of candidate classifiers for each class pair using candidate classification algorithms. The best candidate classifier for each class pair is chosen based on its validation error rate. As a result, a total of $c(c-1)/2$ classifiers are chosen, and they construct a one-against-one classifier. Through this process, DOAO can yield better classification results compared to other one-against-one classifiers that are based on single classification algorithms.

In Section 3.3, we propose ODOAO, an extension of DOAO, in order to achieve better classification accuracy. ODOAO seeks to find the optimal combination of base classifiers that are built for every class pair and candidate classification algorithm according to the concept of DOAO. To do this, a meta-classifier is trained based on meta-learning, where the input variables are the predicted labels from the base classifiers on the validation dataset, and the output variable is the target label. ODOAO is further enhanced by applying a classification algorithm that can effectively deal with high dimensionality and non-linear relationship between the predictions of the base classifiers when training the meta-classifier. The effectiveness of the proposed methods is investigated through experiments on multi-class benchmark datasets.

3.2 Diversified One-Against-One

The fact that a classification algorithm has the highest classification accuracy for a multi-class classification problem does not mean that it performs best for every sub-problem derived from decomposing the original problem. To address

the problem, DOAO seeks to find the best classification algorithm for each class pair when applying the one-against-one approach to multi-class classification problems.

Suppose that a training dataset of N_1 instances $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$ for a c-class classification problem is given, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector and $y_i \in \{1, \ldots, c\}$ is its target label. DOAO first decomposes the dataset $\mathcal{D}$ into subsets $\mathcal{D}_{ij}$ for every class pair (i, j). For each subset $\mathcal{D}_{ij}$, candidate classifiers $\mathcal{C}_{\mathcal{A}_1,\mathcal{D}_{ij}}, \ldots, \mathcal{C}_{\mathcal{A}_L,\mathcal{D}_{ij}}$ are trained using a pre-defined set of candidate classification algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_L$. Of the candidate classifiers, the best candidate classifier $\mathcal{C}_{\mathcal{A}_{\text{best}},\mathcal{D}_{ij}}$ for each class pair is selected to minimize validation errors. Consequently, a total of $c(c-1)/2$ classifiers are selected, each of which is the most competent for distinguishing its corresponding class pair. Algorithm 1 presents the pseudocode of DOAO. Given these $c(c-1)/2$ classifiers, the classification of a test instance is performed by a majority vote of them.

Figure 3.1 shows an illustrative example of DOAO for a three-class toy dataset. In this figure, circles, rectangles, and triangles represent class 1, 2, and 3, respectively. The decision boundaries of class pairs are represented by bold lines. Figure 3.1(a), 3.1(b), and 3.1(c) depict the decision boundaries obtained using one-against-one classifiers based on single classification algorithms. In this example, the classification algorithms $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ perform the best for the class pair $(1, 2)$, $(2, 3)$, and $(1, 3)$, respectively. DOAO selects the most competent decision boundary for each class pair, as shown in Figure 3.1(d). Therefore, each of the decision boundaries corresponds to a different classification algorithm. Therefore, DOAO can yield better classification accuracy compared to other

Algorithm 1 Diversified One-Against-One (DOAO)

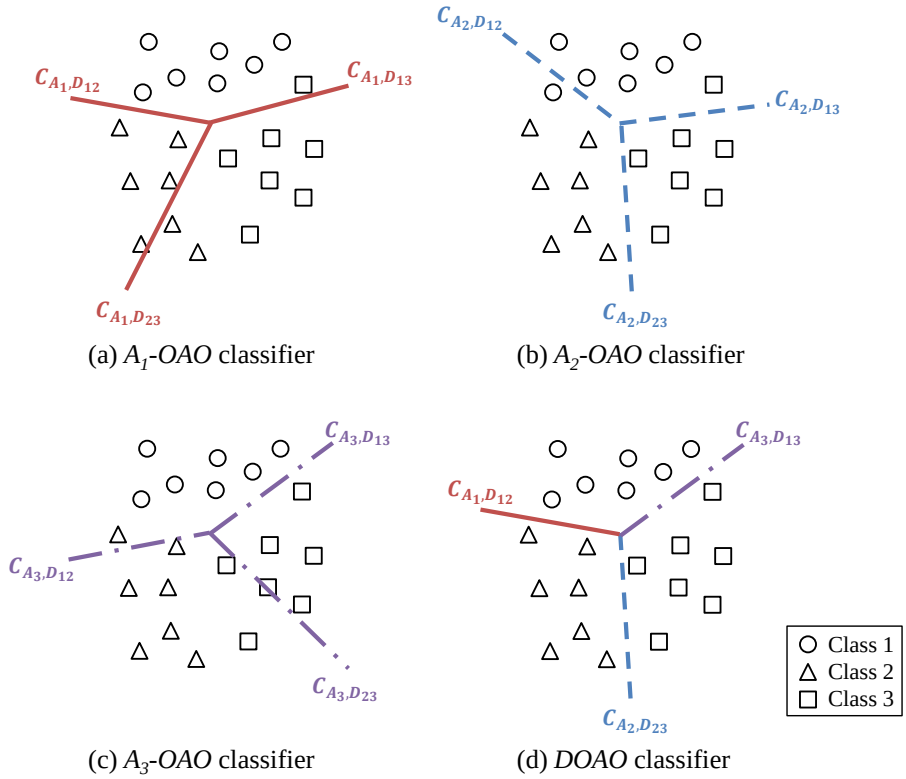
Input: training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$, validation dataset $\mathcal{D}' = \{\boldsymbol{x}_t, y_t\}_{t=N_1+1}^{N_1+N_2}$, $y_t \in \{1, \ldots, c\}$, candidate classification algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_L$

Output: set of base classifiers $\mathbb{C}$

procedure DOAO
  $\mathbb{C} \leftarrow \phi$
  for each class pair (i, j) **do**
    $\mathcal{D}_{ij} \leftarrow \{(\boldsymbol{x}_t, y_t) \in \mathcal{D} | y_t \in \{i, j\}\}$
    $\mathcal{D}'_{ij} \leftarrow \{(\boldsymbol{x}_t, y_t) \in \mathcal{D}' | y_t \in \{i, j\}\}$
    $\mathcal{C}_{\mathcal{A}_k, \mathcal{D}_{ij}} \leftarrow$ candidate classifier trained from $\mathcal{D}_{ij}$ using $\mathcal{A}_k$, $k = 1$ to L
    $\mathcal{A}_{best} \leftarrow \arg\min_{\mathcal{A}_k} \frac{1}{N_2} \sum_{(\boldsymbol{x}_t, y_t) \in \mathcal{D}'_{ij}} \mathbf{1}_{\mathcal{C}_{\mathcal{A}_k, \mathcal{D}_{ij}}(x_t) \neq y_t}$
    $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{C}_{\mathcal{A}_{best}, D_{ij}}\}$
  end for
end procedure

one-against-one classifiers that are based on single classification algorithms.

DOAO makes the various classification algorithms to complement each other. However, this method is differentiated from the vote method, which obtains classification results via a vote of classifiers of various classification algorithms. The vote method aims to achieve a group consensus under the assumption that each classifier has similar competence. Therefore, this method becomes worse when only a few classifiers provide correct classification results and other classifiers do not. On the other hand, DOAO does not encounter this problem because it simply selects the most competent classifier for each sub-problem.



(a) A_1-*OAO* classifier (b) A_2-*OAO* classifier

(c) A_3-*OAO* classifier (d) *DOAO* classifier

Figure 3.1 Construction of a classifier based on DOAO

Moreover, if this method selects the best classifier accurately for every sub-problem, it can unquestionably outperform all the other methods, including the vote method.

The major drawbacks to DOAO are twofold. Firstly, a lower validation error does not always result in a lower test error. This occurs prominently when heterogeneous classifiers are compared. A low-bias, high-variance classifier tends to have a higher test error even when the validation error remains the same. Therefore, classifier selection based on validation errors can lead to overfitting

(Cawley & Talbot, 2010). Secondly, selecting the single best classifier is not always optimal, although it is significantly better than voting. It is known that selective fusion provides better results than either classifier selection or fusion by taking advantage of both (Giacinto & Roli, 2001; Tsoumakas et al., 2005). Considering these two drawbacks, there is room for further improvement. To this end, we propose another method that is an improvement of DOAO in the next section.

3.3 Optimally Diversified One-Against-One

This section introduces ODOAO that aims at the optimal construction of a one-against-one classifier for better classification accuracy. The basic idea in ODOAO is that a meta-classifier discovers the optimal combination of the outputs from a diverse of base classifiers that are trained for every class pair and candidate classification algorithm. This idea is related to meta-learning which is described in Subsection 2.2.2.

Figure 3.2 shows the framework of ODOAO, which consists of two phases. The first phase is to train candidate classifiers from each class pair in the training dataset, which is identical in the original DOAO. This phase results in a set of base classifiers $\mathbb{C}$.

The second phase is to construct a meta-dataset based on the base classifiers and the validation dataset, and to train a meta-classifier using this meta-dataset. Given the classifier set $\mathbb{C}$ consisting of the $L \times c(c-1)/2$ base classifiers from the first phase, the predicted label $\hat{y}_t^i$ is computed for each classifier $\mathcal{C}_i$ and

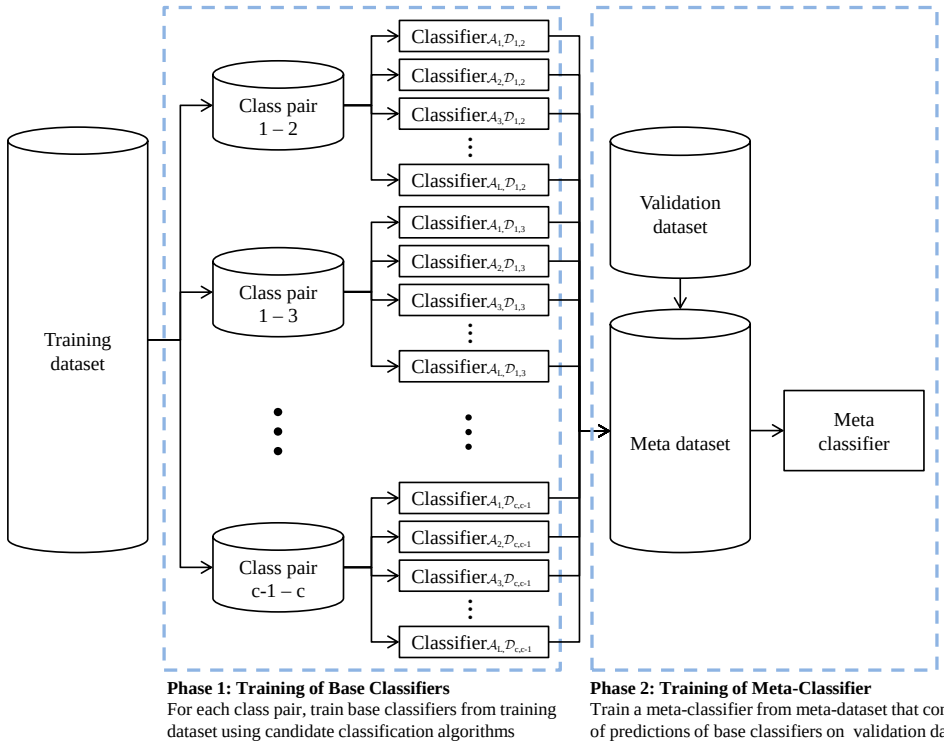


Figure 3.2 Framework of ODOAO

each instance $\boldsymbol{x}_t$ in the validation dataset $\mathcal{D}'$, and the predicted label vector $\mathbf{v}_i$ is constituted as $(\hat{y}^i_{N_1+1}, \hat{y}^i_{N_1+2}, \ldots, \hat{y}^i_{N_1+N_2})$ for each $\mathcal{C}_i$. Then, m clusters $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m$ are generated from the set of vectors $\{\mathbf{v}_i\}_{i|\mathcal{C}_i \in \mathbb{C}}$ using a clustering algorithm. For each clusters, one base classifier is randomly selected. The resulting m base classifiers are used only in order to construct the meta-dataset $\mathcal{M}$. Using the predicted labels of the selected base classifiers as inputs and the target labels as outputs, $\mathcal{M}$ is defined by $\{(\hat{y}^1_t, \hat{y}^2_t, \ldots, \hat{y}^{L\times c(c-1)/2}_t), y_t\}_{(\boldsymbol{x}_t, y_t)\in\mathcal{D}'}$. Finally, the meta-classifier $\mathcal{C}_\mathcal{M}$ is trained with the dataset $\mathcal{M}$. The overall procedure is described in Algorithm 2.

In ODOAO, selecting the base classifiers aims at reducing the redundancies of base classifiers. Therefore, this step should generate clusters where each contains redundant base classifiers. Note that the result of the selection depends on the configuration of a clustering algorithm. Thus, the agglomerative hierarchical clustering algorithm with complete linkage are employed in ODOAO. To evaluate the redundancy, the distance measure between the two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ is defined as $d(\mathbf{v}_i, \mathbf{v}_j) = \min\left(\sum_{(\boldsymbol{x}_t, y_t)\in\mathcal{D}'} \mathbf{1}_{\hat{y}_t^i=a_1 \cap \hat{y}_t^j=b_1}, \sum_{(\boldsymbol{x}_t, y_t)\in\mathcal{D}'} \mathbf{1}_{\hat{y}_t^i=a_1 \cap \hat{y}_t^j=b_2}\right)/N_2$, with assuming that $\hat{y}_t^i \in \{a_1, a_2\}$ and $\hat{y}_t^j \in \{b_1, b_2\}$ for all t. The values of this distance measure range from 0 (most redundant) to 0.5 (least redundant). The distance threshold as stopping criterion of the clustering is set to 0.05.

ODOAO is differentiated from the original DOAO by the second phase. The original DOAO selects the single best classifier for each class pair, and its classification is done by voting on the $c(c-1)/2$ selected classifiers. This method, however, exploits the entire base classifiers trained in the first phase by combining their outputs using the meta-classifier. The meta-classifier decides how to combine the base classifiers. By doing so, ODOAO can overcome the drawbacks in the original DOAO.

In order to ensure that ODOAO performs well, it is important to train the meta-classifier suitably. Two issues should be addressed: high dimensionality and non-linear structure. Regarding the high dimensionality issue, the number of base classifiers obtained in the first phase is $L \times c(c-1)/2$, which is proportional to the square of the number of classes. The dimensionality becomes too large when the number of classes increases, which may lead to degradation of the meta-classifier performance. ODOAO resolves this issue by the base classi-

Algorithm 2 Optimally Diversified One-Against-One (ODOAO)

Input: training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$, validation dataset $\mathcal{D}' = \{\boldsymbol{x}_t, y_t\}_{t=N_1+1}^{N_1+N_2}$, $y_t \in \{1, \ldots, c\}$, candidate classification algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_L$

Output: set of base classifiers $\mathbb{C}$, meta classifier $\mathcal{C}_{\mathcal{M}}$

procedure ODOAO
  ▶ `phase 1`
  $\mathbb{C} \leftarrow \phi$
  for each class pair (i, j) **do**
    $\mathcal{D}_{ij} \leftarrow \{(\boldsymbol{x}_t, y_t) \in \mathcal{D} | y_t \in \{i, j\}\}$
    $\mathcal{C}_{\mathcal{A}_k, \mathcal{D}_{ij}} \leftarrow$ candidate classifier trained from $\mathcal{D}_{ij}$ using $\mathcal{A}_k$, $k = 1$ to L
    $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{C}_{\mathcal{A}_1, D_{ij}}, \ldots, \mathcal{C}_{\mathcal{A}_L, D_{ij}}\}$
  end for
  ▶ `phase 2`
  for each classifier $\mathcal{C}_i \in \mathbb{C}$ **do**
    $\hat{y}_t^i \leftarrow \mathcal{C}_i(x_t)$, $\forall(\boldsymbol{x}_t, y_t) \in \mathcal{D}'$
    $\mathbf{v}_i \leftarrow (\hat{y}_{N_1+1}^i, \hat{y}_{N_1+2}^i, \ldots, \hat{y}_{N_1+N_2}^i)$
  end for
  $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m \leftarrow$ clusters generated from $\{\mathbf{v}_i\}_{i|\mathcal{C}_i \in \mathbb{C}}$
  $s(1), s(2), \ldots, s(m) \leftarrow$ index of a randomly chosen vector in each of $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_m$
  $\mathcal{M} \leftarrow \{(\hat{y}_t^{s(1)}, \hat{y}_t^{s(2)}, \ldots, \hat{y}_t^{s(m)}), y_t\}_{(\boldsymbol{x}_t, y_t) \in \mathcal{D}'}$
  $\mathcal{C}_{\mathcal{M}} \leftarrow$ meta-classifier trained from $\mathcal{M}$
end procedure

Table 3.1 The number of classifiers used for training and test for each method

Method	N. classifiers trained	N. classifiers used for test
Single Algorithm-OAO	$c(c-1)/2$	$c(c-1)/2$
VOTE-OAO	$L \times c(c-1)/2$	$L \times c(c-1)/2$
DOAO	$L \times c(c-1)/2$	$c(c-1)/2$
ODOAO	$L \times c(c-1)/2+1$	$\theta \times L \times c(c-1)/2+1$

fier selection step in the second phase. For the non-linear structure issue, the predicted label of a base classifier for a class pair (i, j) is only i or j because the decomposition in ODOAO is based on the one-against-one approach. Thus, the relationship between the predictions of the base classifiers and the target labels are not linear. To train the meta-classifier, classification algorithms that can deal with non-linear structure effectively, and are also robust to high input dimensionality should be employed.

The training and test time should be taken into account for practical deployment of a classification algorithm (Kang & Cho, 2014). Thus, we analyze the relative time of training and test for ODOAO and other methods in terms of the number of classifiers involved. Table 3.1 shows the number of classifiers trained in the training phase and used for classifying a test instance. Note that θ of ODOAO is the fraction of base classifiers used as inputs of the meta-classifier, and the actual value of θ for each benchmark dataset is reported in Table 3.5.

Regarding the training phase, a single algorithm-OAO involves training of base classifiers for all possible class pairs using a single classification algorithm. VOTE-OAO and DOAO train base classifiers using every candidate classification

algorithm, so $L \times c(c-1)/2$ classifiers are trained. Added to these, ODOAO additionally requires training of one more classifier as a meta-classifier. As a simple comparison, VOTE-OAO, DOAO, and ODOAO involve more classifier training in their training phase than single algorithm-OAOs. However, since we usually have no certain knowledge that which algorithm performs best for a dataset (Sohn, 1999; Lim et al., 2000; Kiang, 2003), we should compare the classification performance of several single algorithm-OAOs that are trained using different classification algorithms and choose the best one. Therefore, the practical difference in the training phase is not so significant.

For the test phase, single algorithm-OAOs and DOAO use $c(c-1)/2$ base classifiers, each of which corresponds to a class pair, to classify a test instance. On the other hand, ODOAO involves $\theta \times L \times c(c-1)/2 + 1$ classifiers for test, which depends on θ but generally larger than other methods. This would be a drawback for real-time applications that requires fast classification speed.

Retraining issue is also a main concern if new data is added continuously and the classifier needs to be updated accordingly. Fast re-training speed is required for such a streaming data environment. We can consider two cases. The first case is that new instances of existing classes are added into the training data. We can cope with this case by using only candidate classification algorithm that can be trained incrementally (Giraud-Carrier, 2000).

The second is about the instances of a novel class. The proposed methods are effective for this case because of the property of the one-against-one approach. The one-against-one approach does not require retraining of any existing classifiers and only requires training of classifiers that correspond to the

novel class, whereas all the existing base classifiers have to be retrained for the one-against-rest approach.

3.4 Performance Evaluation on Benchmark Datasets

3.4.1 Data Description

The effectiveness of the proposed methods was investigated through experiments on the benchmark datasets. The following 15 multi-class benchmark datasets, each with more than two classes, were collected from the UCI machine learning repository (Bache & Lichman, 2014): *Zoo*, *Iris*, *Wine*, *Seed*, *Glass*, *Ecoli*, *Movement*, *Balance*, *Landcover*, *Vehicle*, *Annealing*, *Vowel*, *Yeast*, *CarEvaluation*, and *Segment*. Since the proposed methods have no effect on binary classification problems, the number of classes of every collected dataset is larger than two. A detailed description for each dataset is provided in Table 3.2.

3.4.2 Experimental Settings

In the experiment, the effectiveness of the proposed methods DOAO and ODOAO was investigated. For training of base classifiers, six well-known and widely used classification algorithms that are introduced in Section 2.1 were employed in order to train the base classifiers: ANN, DT, kNN, LDA, LR, and SVM. For meta-classifiers of ODOAO, we used ANN, DT, and SVM to investigate the respective suitability.

Regarding the proposed methods, ODOAO was implemented in three forms

Table 3.2 Data summary

Dataset	N. instances	N. features	N. classes
Zoo	101	16	7
Iris	150	4	3
Wine	178	13	3
Seed	210	7	3
Glass	214	9	6
Ecoli	336	7	8
Movement	360	90	15
Balance	625	4	3
Landcover	675	147	9
Vehicle	846	18	4
Annealing	898	38	6
Vowel	990	10	11
Yeast	1484	8	10
CarEvaluation	1728	6	4
Segment	2310	19	7

(ODOAO$_{\text{ANN}}$, ODOAO$_{\text{DT}}$, and ODOAO$_{\text{SVM}}$), each employed ANN, DT, and SVM, respectively, as the meta-classifier. As one-against-one benchmarks, the six one-against-one classifiers (ANN-OAO, DT-OAO, kNN-OAO, LDA-OAO, LR-OAO, and SVM-OAO) were used, each is based on an individual classification algorithm. They were also compared with a vote-based method (VOTE-OAO), that votes on the candidate classifiers for each class pair, rather than selecting the best classifier. In addition, the two typical homogeneous ensemble methods, ANN-Bagging (Breiman, 1996) and Random Forest (RF) (Breiman,

2001), were employed as benchmark methods. In Bagging, each base classifier is trained with a bootstrap sample drawn randomly from a given training set. RF is an extension of Bagging for DT that randomizes the choice of features when splitting each node of the DT. When training all the classifiers, numeric variables were scaled to be in the range of $[-1, 1]$. All the algorithms used in the experiments were implemented using MATLAB.

To compare the proposed methods with the benchmark methods, the classification performance was evaluated using the misclassification error rate (%) on the test dataset, which is defined by $(1/N)\sum_{t=1}^{N} 1_{y_t \neq \hat{y}_t} \times 100$, where N is the number of test instances, y_t is the target label of the t-th instance, $\hat{y}_t$ is the predicted label of the t-th instance, and $1_{y_t \neq \hat{y}_t}$ is an indicator function that has a value of 1 when $y_t \neq \hat{y}_t$.

A ten-fold cross test procedure was conducted for each method, involved partitioning the original dataset into ten disjointed and equally sized subsets. Then, nine subsets were used as the training dataset, and the test error was calculated for the remaining. This process was independently repeated ten times, using each of the ten subsets exactly once as the test dataset.

In each run, the best parameters of each classification algorithm for a class pair were explored through ten-fold cross validation with a grid search mechanism in the training set. The parameter search spaces used in the experiments are given in Table 3.3. For DOAO, the best classifier for each class pair was selected based on the cross validation error. For ODOAO, the predicted labels for the base classifiers resulting from the cross validation procedure were used to construct the meta-dataset to build meta-classifiers. For ANN-Bagging, the

Table 3.3 Parameter settings for each algorithm

Algorithm	Parameter Setting
ANN	N. hidden nodes=$3, 4, 5, \ldots, 20$
	Max. iterations=300
DT	Min. instances in a leaf node=$1, 2, 3, 5$
	Min. instances in a parent nodes=$5, 10$
	Prune=true
kNN	k=$1, 3, 5, 7, 10, 20, 30$
	Distance type=Euclidean
LDA	No parameter
LR	No parameter
SVM	C=$2^{-3}, \ldots, 2^{10}$
	Kernel type=RBF kernel
	σ=$2^{-5}, \ldots, 2^{5}$

number of base classifiers, the number of hidden nodes, and the bootstrap sample size were set as 10, 10, and 80% of the training set, respectively. For RF, the number of base classifiers, the bootstrap sample size and the minimum number of instances in a leaf node were set as 100, 80% and 1% of the training set, respectively.

3.4.3 Experimental Results

Table 3.4 shows the results from comparing the proposed methods with the benchmark methods in terms of the error rate (%). The numbers in bold indicate the lowest error rate obtained over all the methods tested.

Table 3.4 Error rate (%) comparison results on benchmark datasets

Dataset	Benchmark (typical)		Benchmark (one-against-one)							Proposed			
	ANN-Bagging	RF	ANN-OAO	DT-OAO	kNN-OAO	LDA-OAO	LR-OAO	SVM-OAO	VOTE-OAO	DOAO	$\text{ODOAO}_{\text{ANN}}$	ODOAO_{DT}	$\text{ODOAO}_{\text{SVM}}$
Zoo	7.921	3.960	2.970	9.901	3.960	14.851	3.960	6.931	3.960	**0.990**	2.970	2.970	2.970
Iris	2.667	4.000	3.333	4.667	4.000	2.000	2.000	4.667	4.000	3.333	**1.333**	**1.333**	**1.333**
Wine	2.247	2.247	3.371	5.618	3.371	1.124	2.809	1.124	1.685	1.124	**0.562**	**0.562**	1.124
Seed	6.190	5.714	6.190	6.667	7.619	**3.333**	**3.333**	7.143	5.238	4.762	3.810	**3.333**	**3.333**
Glass	35.981	**19.626**	33.178	23.832	33.178	35.047	35.514	33.178	31.308	32.243	25.234	32.710	26.636
Ecoli	13.690	12.798	13.393	15.179	13.095	12.798	14.583	14.583	12.798	12.500	**11.905**	13.988	12.202
Movement	35.556	20.833	20.556	28.333	16.667	19.167	25.833	15.556	13.611	14.722	**9.444**	20.278	9.722
Balance	9.120	11.200	6.080	22.080	10.240	14.400	10.720	**0.320**	8.000	**0.320**	1.120	1.760	1.440
Landcover	22.667	14.963	16.000	17.481	19.852	30.222	35.852	15.259	15.852	15.259	**12.741**	15.407	13.037
Vehicle	19.976	26.478	21.158	27.541	31.442	19.031	19.031	15.248	18.558	15.248	**14.184**	16.667	15.248
Annealing	1.225	3.341	0.780	1.114	1.225	2.561	**0.223**	1.114	0.557	0.334	**0.223**	0.334	**0.223**
Vowel	18.081	12.424	3.434	16.364	0.909	25.859	19.596	0.606	5.253	**0.505**	1.212	2.929	1.212
Yeast	41.712	38.477	40.431	40.701	40.633	41.577	41.779	40.094	40.162	40.364	39.960	40.499	**38.612**
CarEvaluation	1.563	8.681	0.347	2.894	14.178	7.639	5.903	0.174	1.794	**0.000**	0.116	0.058	**0.000**
Segment	5.108	4.113	3.593	3.030	2.727	5.455	3.983	2.641	2.381	2.381	**1.558**	2.294	1.602

Overall, DOAO and ODOAO yielded better classification results when some single algorithm-OAOs are superior to others, and even outperformed the best of the six single algorithm-OAOs. That is, the classification accuracy can be improved further, even though a sufficiently superior classification algorithm currently exists. Otherwise, VOTE-OAO showed relatively suitable results when the classification accuracy of the six single algorithm-OAOs was similar, but it performed worse than the best of them. DOAO and ODOAO performed better than VOTE-OAO, even though they commonly exploit several candidate classifiers. This is because it is more likely that some classifiers were more competent than others when different classification algorithms are employed to train candidate classifiers. The proposed methods also yielded lower error rate than typical homogeneous ensemble methods, ANN-Bagging and RF, for entire datasets except *Glass* dataset.

Considering DOAO, DOAO was worse than one of the single algorithm-OAOs in some datasets, which indicates that DOAO does not always select the best classifier for each class pair. This is because DOAO simply selects classifiers for each class pair based on the validation error, and a lower validation error does not always leads to a lower test error. The comparison results show that $\text{ODOAO}_{\text{ANN}}$ and $\text{ODOAO}_{\text{SVM}}$ outperform DOAO on most datasets, indicating that the meta-classifiers based on ANN and SVM effectively find better combinations of the base classifiers. However, ODOAO_{DT} perform relatively worse than both DOAO and VOTE-OAO.

Table 3.5 lists the average selection number for each classification algorithm in DOAO. The selection number indicates that DOAO selected the algorithm for

Table 3.5 Average selection number of each candidate classification algorithm for DOAO on benchmark datasets

Dataset	N. classes	N. class pairs	ANN	DT	kNN	LDA	LR	SVM
Zoo	7	21	7.9	2.7	3.9	1.0	2.4	3.1
Iris	3	3	1.2	0.4	0.2	0.2	0.4	0.6
Wine	3	3	1.2	0.0	0.2	0.4	0.3	0.9
Seed	3	3	1.1	0.3	0.3	0.2	0.9	0.2
Glass	6	15	9.0	1.5	1.1	0.6	0.3	2.5
Ecoli	8	28	18.2	1.2	1.9	2.2	1.9	2.6
Movement	15	105	50.7	5.0	17.3	2.9	0.7	28.4
Balance	3	3	0.0	0.0	0.0	0.0	0.0	3.0
Landcover	9	36	25.3	2.6	2.5	0.7	0.0	4.9
Vehicle	4	6	1.8	0.0	0.0	0.0	0.0	4.2
Annealing	6	15	2.9	1.3	0.7	0.3	2.8	2.0
Vowel	11	55	13.7	0.5	14.9	2.0	3.9	20.0
Yeast	10	45	32.9	1.9	3.0	0.5	1.3	5.4
CarEvaluation	4	6	2.3	0.3	0.0	0.0	0.5	2.9
Segment	7	21	7.4	2.7	2.8	1.2	1.6	5.3

the corresponding number of class pairs. As shown in this table, SVM and ANN are dominantly selected, but are not entirely selected on most datasets. Other algorithms are selected instead for particular class pairs, and consequently DOAO yields more desirable classification results. The selection of classifiers is varied by datasets. Some classification algorithms, such as LR and LDA, are not often selected compared to other algorithms, but took a large part of some datasets. In the case of the *Balance* dataset, only SVM is selected for every class pair;

Table 3.6 Average selection number of base classifiers for ODOAO on benchmark datasets

Datasets	N. classes	N. base classifiers total	N. base classifiers selected	θ: Frac. selected (%)
Zoo	7	126	91.2	72.4
Iris	3	18	2.5	13.9
Wine	3	18	9.3	51.7
Seed	3	18	8.0	44.4
Glass	6	90	62.3	69.2
Ecoli	8	168	92.3	54.9
Movement	15	630	575.4	91.3
Balance	3	18	12.2	67.8
Landcover	9	216	186.0	86.1
Vehicle	4	36	33.1	91.9
Annealing	6	60	33.8	56.3
Vowel	11	330	284.2	86.1
Yeast	10	270	136.0	50.4
CarEvaluation	4	36	18.2	50.6
Segment	7	126	77.7	61.7

hence, there is no difference between DOAO and SVM-OAO.

Table 3.6 shows the average number of classifiers selected by the base classifier selection step of ODOAO for each dataset. The results show that input dimensionality of the meta-classifier is reduced by the selection step. On average, 63.3% of the base classifiers are selected across the datasets. In case of the *Iris* dataset, only 13.9% of the base classifiers are used. On the other hand, over 90% of the classifiers are used for the *Movement* and *Vehicle* datasets.

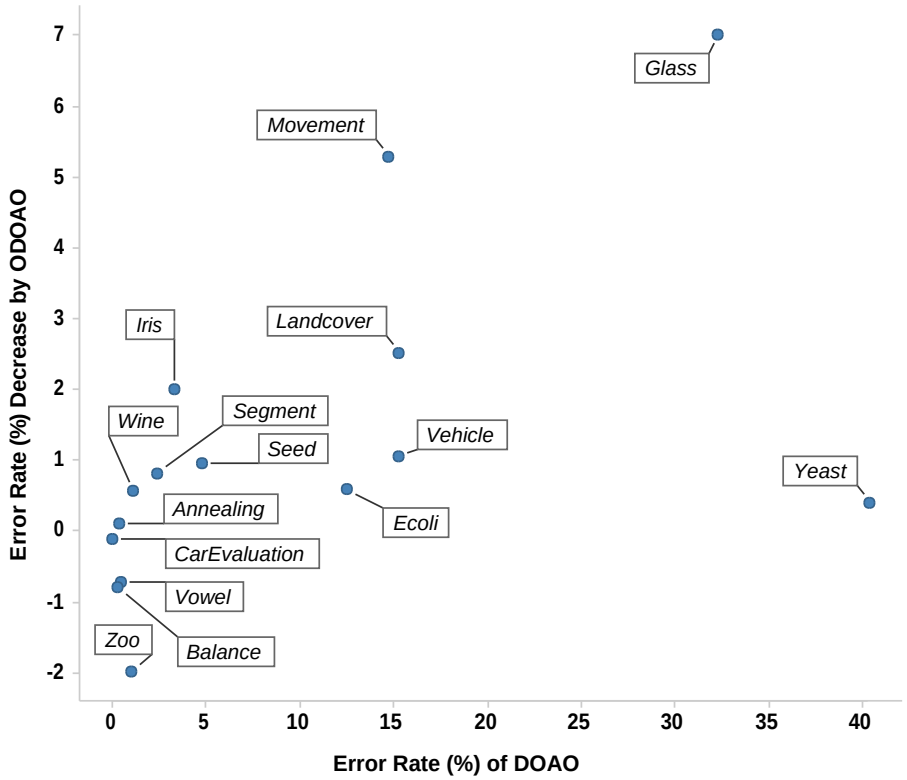


Figure 3.3 The relationship between the error rate (%) of DOAO and that decrease by ODOAO

Considering the relationship between ODOAO and DOAO, the former yields a lower, or at least equal, error rate in 12 out of the 15 datasets. Figure 3.3 plots the error rate difference between $\text{ODOAO}_{\text{ANN}}$ and DOAO against the error rate of DOAO. ODOAO yields a much lower error rate than DOAO for the *Glass, Movement*, and *Landcover* datasets. On the other hand, ODOAO performs worse than DOAO when the error rate is close to 0%, as is the case for the *Zoo, Balance*, and *Vowel* datasets. That is, the meta-classifier of ODOAO itself may lead to

Table 3.7 Results of the Holm post-hoc test

Method	Average Rank	Holm APV	Hypothesis (α=0.05)
ODOAO$_{\text{ANN}}$	2.3	-	-
ODOAO$_{\text{SVM}}$	2.63	0.783	Not rejected
DOAO	3.63	0.542	Not rejected
ODOAO$_{\text{DT}}$	4.67	0.152	Not rejected
SVM-OAO	5.93	0.0123	Rejected
VOTE-OAO	5.97	0.0123	Rejected
ANN-OAO	7.23	2.78×10^{-4}	Rejected
LR-OAO	8.13	1.02×10^{-5}	Rejected
kNN-OAO	8.2	8.85×10^{-6}	Rejected
LDA-OAO	8.3	6.53×10^{-6}	Rejected
DT-OAO	9	3.16×10^{-7}	Rejected

misclassification errors when there is no room for improving the classification accuracy.

We conducted non-parametric statistical tests in order to determine the statistical significance of the comparison results (Demšar, 2006; García & Herrera, 2008; García et al., 2009, 2010). The Friedman test was executed to find out the statistical difference between the one-aganst-one methods used in the experiments. Since the p-value returned by the Friedman test is 8.2838×10^{-11}, the null hypothesis of equivalence between the methods is rejected with a high level of confidence. Therefore it can be concluded that there are significant differences between the methods. Subsequently, the Holm post-hoc test was carried out to compare the best method with the others, and the results from this test are provided in Table 3.7. These results show that ODOAO$_{\text{ANN}}$ sig-

Table 3.8 Results of the Wilcoxon signed-rank test comparing $\mathsf{ODOAO}_{\mathrm{ANN}}$, $\mathsf{ODOAO}_{\mathrm{SVM}}$, and DOAO

Comparison	R^+	R^-	p-value	Hypothesis (α=0.05)
$\mathsf{ODOAO}_{\mathrm{ANN}}$ vs. $\mathsf{ODOAO}_{\mathrm{SVM}}$	47	19	0.107	Not rejected
$\mathsf{ODOAO}_{\mathrm{ANN}}$ vs. DOAO	94	26	0.0267	Rejected
$\mathsf{ODOAO}_{\mathrm{SVM}}$ vs. DOAO	62	16	0.0356	Rejected

nificantly outperforms most other methods, with the exception of $\mathsf{ODOAO}_{\mathrm{SVM}}$ and DOAO.

The Holm test did not provide sufficiently significant differences between $\mathsf{ODOAO}_{\mathrm{ANN}}$, $\mathsf{ODOAO}_{\mathrm{SVM}}$, and DOAO. Therefore, the Wilcoxon signed-rank test was ultimately conducted for a pairwise comparison among these methods. The p-values obtained by the Wilcoxon test are presented in Table 3.8. As shown in Table 3.8, there is no significant difference between $\mathsf{ODOAO}_{\mathrm{ANN}}$ and $\mathsf{ODOAO}_{\mathrm{SVM}}$, whereas DOAO is significantly worse than either method. Consequently, we confirm that ODOAO statistically outperforms DOAO and the other methods. According to the statistical test results, we can conclude that the proposed methods statistically outperform the other methods.

3.5 Summary

In this chapter, we introduced a method called DOAO to solve multi-class classification problems by constructing a multi-class classifier using the one-against-one approach with different classification algorithms. This method aims to ob-

tain improved classification results by selecting the best classification algorithm for each class pair. Thus, it constructs a one-against-one classifier by selecting the best classifiers among several heterogeneous candidate classifiers. Applying DOAO makes various classification algorithms to complement each other. Since the best classification algorithm for each class pair is different, it was founded to yield better classification accuracy than other one-against-one classifiers that are based on individual classification algorithms.

DOAO is not only easy and intuitive, but also very effective for multi-class classification problems. Applying this method to multi-class classification problems provides better classification results. However, there are two major limitations in DOAO. The first is that the minimum validation error does not guarantee the minimum test error. The second is that a selective fusion of the best set of classifiers would better than the single best classifier. Thus, we determined that DOAO can be improved further by addressing these limitations.

Next, we introduced another method, ODOAO, as an improvement to DOAO based on meta-learning. ODOAO constructs a heterogeneous ensemble where a meta-classifier effectively combines the outputs from all the heterogeneous base classifiers that are trained using various classification algorithms for every class pair. Base classifiers are trained according to the one-against-one approach using various candidate classification algorithms, and a meta-classifier is trained to optimally combine the outputs of the base classifiers. In addition, we take into account the issues of high dimensionality and non-linear structure when training the meta-classifier.

Our experimental results showed that the proposed methods outperform

other one-against-one classifiers that are based on single classification algorithms, and also outperforms the vote-based one-against-one classifier, on most benchmark datasets. The experimental results also showed that ODOAO outperforms DOAO and the other one-against-one classifiers on most datasets. The statistical significance of the results was confirmed through non-parametric statistical tests.

Regarding practical deployment of the proposed methods, the training of a number of candidate classifiers can be time consuming. However, such burden does not represent a problem if the training of classifiers is not subjected to a time limit. One legitimate concern is that the heterogeneous ensemble constructed by the proposed methods may require a large amount of computation when classifying new instances because all of the base classifiers as well as the meta-classifier are used for classification. This would be a drawback when fast classification speed is required, as with real-time applications (Kang & Cho, 2014). In Chapter 6, this drawback is addressed by approximating the ensemble with a single model in order to increase the classification speed.

Chapter 4

Heterogeneous Ensemble for Accurate Classification: One-class Classifier Approach

4.1 One-class Classifier Approach for Multi-class Classification

In Chapter 3, heterogeneous ensemble methods involving construction of an ensemble of binary classifiers are proposed. Although the binary classifier approach has received more attention and has been more often used, the strategy that involves construction of an ensemble of one-class classifiers has also been successfully applied to multi-class classification problems (Juszczak & Duin, 2004; Ban & Abe, 2006; Tax & Duin, 2008; D. Lee & Lee, 2007; Hao et al., 2009), because of some merits. This strategy only requires training of one classifier and does not require re-training of any existing classifiers when another class is added into the training data (Juszczak & Duin, 2004). Moreover, this strategy may be effective when class imbalance is severe (H.-j. Lee & Cho, 2006).

In general, this is achieved by decomposing the original multi-class classifi-

cation problem into c one-class sub-problems. For each of the sub-problems, a one-class classifier is trained to determine if an instance belongs to a class. The resulting c base classifiers constitute an ensemble to make predictions for the original problem.

With respect to the third strategy, the existing methods proposed in previous work were based on homogeneous ensembles, which means that they have only used single one-class classification algorithms to construct the ensembles. For further improvement on classification accuracy, heterogeneous ensembles, which employ various classification algorithms to train base classifiers, can be advantageous (Kang, Cho, & Kang, 2015a). The heterogeneous ensemble allows us to obtain better classification accuracy by combining the advantages of various algorithms, while this ensemble generally requires a large computational burden in training and test because it involves many base classifiers from various algorithms.

When constructing a heterogeneous ensemble of one-class base classifiers, the major issue is about the normalization of the prediction scores from the base classifiers. Since each classifier is independently trained, their scores may be on different scales (Bishop & Nasrabadi, 2006). The problem of different scales becomes more serious when base classifiers from different algorithms are considered. Moreover, if some base classifiers are inadequately trained, they may provide incorrect scores as predictions for new instances, and consequently, the classification results provided by an ensemble might be invalid (Galar et al., 2013). Therefore, the scores should be appropriately normalized, and some invalid base classifiers should be filtered out when combining the multiple base

classifiers.

There have been several efforts to address this problem. Some researchers proposed algorithm-specific normalization methods. Ban and Abe (2006) proposed normalization methods for each of the kernel PCA and SVDD algorithms. In particular, there are a number of methods for SVDD (Hao et al., 2009). On the other hand, general normalization methods have also been developed. D. Lee and Lee (2007) utilized Bayesian decision theory for combining base classifiers. Tax and Duin (2008) proposed two normalization methods based on the Bayesian decision theory: outlier normalization (O-norm) and target normalization (T-norm). The methods proposed in previous work focused on combining homogeneous base classifiers on the basis of heuristics. In addition, some methods were only valid for a specific algorithm. On the other hand, we focus on learning-based normalization for combining heterogeneous base classifiers as an improvement over previous work.

To this end, we propose a heterogeneous ensemble method with addressing the aforementioned issues for multi-class classification. The proposed method is based on stacking in order to combine heterogeneous one-class base classifiers trained from various algorithms effectively. MLR (Ting & Witten, 1999) is employed for learning-based normalization. That is, the proposed method builds an MLR model that combines the base classifiers trained by various algorithms. Since an MLR is scale independent, the proposed method can successfully normalize the scores from the heterogeneous base classifiers. In addition, to minimize the risk of invalid base classifiers in the ensemble, we propose selective stacking, which is a variant of stacking, to achieve better classification

accuracy. Selective stacking utilizes variable selection methods when training the meta-classifier in order to exclude invalid and superfluous base classifiers. Experiments were conducted using benchmark datasets to investigate the effectiveness of the proposed method.

4.2 Heterogeneous Ensemble of One-class Classifiers

Multi-class classification problems can be solved by an ensemble of one-class classifiers. The use of various algorithms to train the base classifiers of the ensemble leads to better classification accuracy. However, the normalization of the scores obtained from the heterogeneous base classifiers is a difficult problem because each algorithm has a different type of scores (Tax, 2001). Density-based algorithms provide probability densities as scores. The score type of boundary-based and reconstruction-based algorithms is the distance from the boundary and the reconstruction error, respectively. The scales of the scores may also be different even though the same algorithm is used.

In this section, we propose a multi-class classification method, which combines heterogeneous one-class classifiers based on stacking. In the proposed method, a meta-classifier is trained using stacking with MLR (Ting & Witten, 1999) for learning-based normalization of the scores obtained from the base classifiers. MLR is a multi-class classification algorithm that separates each class from all other classes by linear regression models. Owing to the scale independent characteristic of MLR, the normalization issue can be effectively addressed.

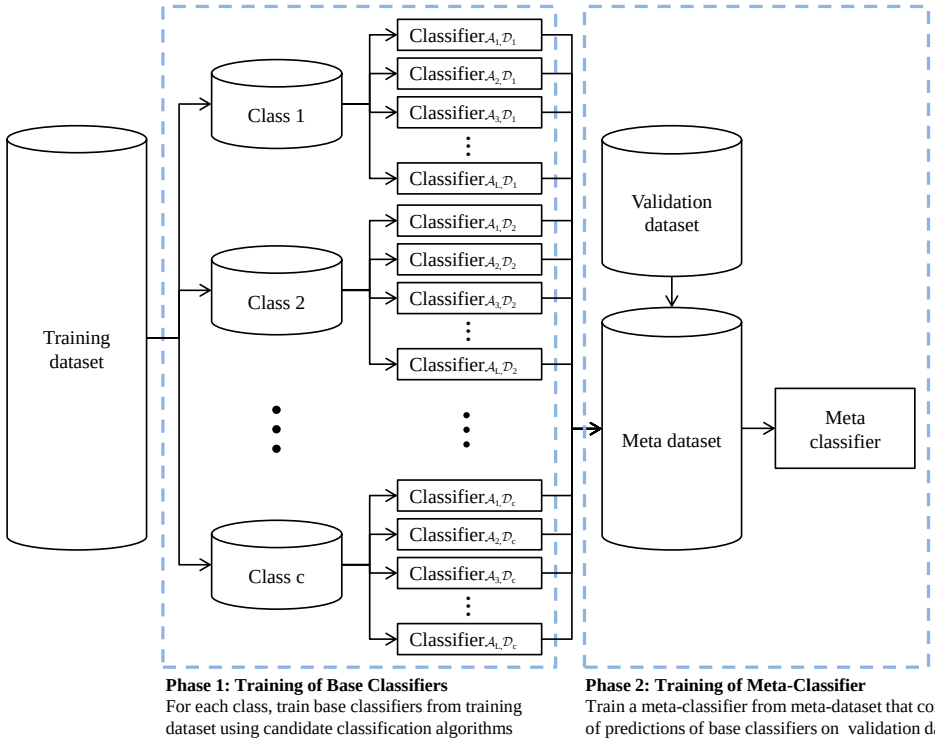


Figure 4.1 Framework of HEOC

MLR has been successfully employed for training of a meta-classifier (Ting & Witten, 1999; Džeroski & Ženko, 2004; Seewald, 2002). MLR is a multi-class classification algorithm that separates each class from the other classes by linear regression models, according to the one-against-rest approach. This algorithm formulates c regression problems for a c-class classification problem. Given a meta-dataset $\mathcal{M} = \{(s_t^1, s_t^2, \ldots, s_t^L), y_t\}_{t=1}^N$, the linear regression model for the i-th class is trained from $\{(s_t^1, s_t^2, \ldots, s_t^L), \mathbf{1}_{y_t=i}\}_{t=1}^N$. After c linear regression models are trained, classification for a new meta instance is done by choosing the class label with the maximum score from the linear regression models.

Figure 4.1 shows the framework of the proposed method that consists of two phases. In the first phase, one-class base classifiers are trained using candidate classification algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_L$. Suppose a training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$ of a c-class classification problem is given, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector, and $y_i \in \{1, \ldots, c\}$ denotes its target label. The dataset $\mathcal{D}$ is partitioned into c subsets $\mathcal{D}_i$ for each class. For every $\mathcal{D}_i$, base classifiers $\mathcal{C}_{\mathcal{A}_1,\mathcal{D}_i}, \mathcal{C}_{\mathcal{A}_2,\mathcal{D}_i}, \ldots, \mathcal{C}_{\mathcal{A}_L,\mathcal{D}_i}$ are trained using the candidate one-class classification algorithms, thereby resulting in $L \times c$ base classifiers.

In the second phase, a meta-classifier is trained using MLR in order to combine the base classifiers. Given a validation dataset $\mathcal{D}' = \{\boldsymbol{x}_t, y_t\}_{t=N_1+1}^{N_1+N_2}$, $y_t \in \{1, \ldots, c\}$, the prediction scores $s_t^{k,i}$ are computed for every validation instance $(\boldsymbol{x}_t, y_t) \in \mathcal{D}'$ using the algorithm $\mathcal{A}_k$. The meta-dataset $\mathcal{M}_i$ of the i-th class is defined as $\{(s_t^{1,1}, \ldots, s_t^{L,1}, s_t^{1,2}, \ldots, s_t^{L,2}, \ldots, s_t^{1,c}, \ldots, s_t^{L,c}), \mathbf{1}_{y_t=i}\}_{(\boldsymbol{x}_t,y_t)\in\mathcal{D}'}$, such that the scores of all base classifiers are input variables and the output variable is a binary variable indicating whether the target label is the i-th class. Finally, a linear regression model $\mathcal{C}_{\mathcal{M}_i}$ is trained from each meta-dataset $\mathcal{M}_i$. The resulting c linear regression models form the meta-classifier. The pseudocode of the proposed method is described in Algorithm 3.

The ensemble that consists of the base classifiers and meta-classifier is used to classify a new instance. Given an instance, the prediction scores are computed using the base classifiers. These scores are input for the meta-classifier, and the meta-classifier provides the final classification result as $\hat{y} = \arg\max_i \mathcal{C}_{\mathcal{M}_i}(\boldsymbol{x})$, by choosing the label with the maximum score among the c linear regression models $\mathcal{C}_{\mathcal{M}_i}$.

Algorithm 3 Heterogeneous Ensemble One-class Classifiers (HEOC)

Input: training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$, validation dataset $\mathcal{D}' = \{\boldsymbol{x}_t, y_t\}_{t=N_1+1}^{N_1+N_2}$, candidate classification algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_L$

Output: base classifiers $\mathcal{C}_{\mathcal{A}_k,\mathcal{D}_i}$, meta classifiers $\mathcal{C}_{\mathcal{M}_i}, \forall k, i$

1: **procedure** HEOC
2: ▶ `phase 1`
3: **for** $i = 1$ to c **do**
4: $\mathcal{D}_i \leftarrow \{(\boldsymbol{x}_t, y_t) \in \mathcal{D} | y_t = i\}\}$
5: $\mathcal{C}_{\mathcal{A}_k,\mathcal{D}_i} \leftarrow$ base classifier trained from $\mathcal{D}_i$ using $\mathcal{A}_k$, $k = 1$ to L
6: $s_t^{k,i} \leftarrow \mathcal{C}_{\mathcal{A}_k,D_i}(x_t)$, $\forall(\boldsymbol{x}_t, y_t) \in \mathcal{D}'$, $k = 1$ to L
7: **end for**
8: ▶ `phase 2`
9: **for** $i = 1$ to c **do**
10: $\mathcal{M}_i \leftarrow \{(\underbrace{s_t^{1,1}, \ldots, s_t^{L,1}}_{\text{class 1}}, \underbrace{s_t^{1,2}, \ldots, s_t^{L,2}}_{\text{class 2}}, \ldots, \underbrace{s_t^{1,c}, \ldots, s_t^{L,c}}_{\text{class } c}), \mathbf{1}_{y_t=i}\}_{(\boldsymbol{x}_t,y_t)\in\mathcal{D}'}$
11: $\mathcal{C}_{\mathcal{M}_i} \leftarrow$ linear regression model trained from $\mathcal{M}_i$
12: **end for**
13: **end procedure**

As an alternative to the standard stacking, stackingC (Seewald, 2002) can be employed for the proposed method. StackingC differs from the standard stacking in the construction of the meta-dataset. For the construction of the meta-dataset $\mathcal{M}_i$, stacking uses the prediction scores of all the base classifiers as the input variables, whereas stackingC uses only those scores that have been computed using base classifiers with the i-th class. Thus, Algorithm 3 can rep-

resent the pseudocode for the proposed method with stackingC if the 10-th line is changed to the following:

$$\mathcal{M}_i \leftarrow \{(\underbrace{s_t^{1,i}, \ldots, s_t^{L,i}}_{\text{class } i}), \mathbf{1}_{y_t=i}\}_{(\boldsymbol{x}_t, y_t)\in\mathcal{D}'}. \tag{4.1}$$

In the proposed method, however, if some base classifiers are invalid or superfluous, the performance of the MLR model may be degraded. Therefore, to prevent performance degradation of the MLR model, only significant base classifiers should be used to train the MLR model. As a further improvement, we propose to use selective stacking, which aids in filtering out invalid base classifiers based on variable selection methods when training the meta-classifier. For each meta-dataset $\mathcal{M}_i$, the linear regression model $\mathcal{C}_{\mathcal{M}_i}$ is trained with a variable selection method. As a result, only a subset of variables among the $L \times c$ input variables of $\mathcal{M}_i$ is selected, which indicates that only selected base classifiers are used as inputs of the meta-classifier. This is differentiated from the standard stacking, because the standard stacking uses all the base classifiers regardless of significance of each base classifier.

For selective stacking, stepwise selection is employed, which is a wrapper method that enters and removes input variables in a stepwise manner, as a variable selection method. In each step of linear regression model training, each input variable is either selected or removed according to the statistical test. The repetition of the steps is terminated when neither the entrance of a new variable or removal of an existing variable improves the model. Consequently, only significant variables are included in the model, which indicates that only the significant base classifiers are used for prediction in the model. The base

classifiers selected for each class may be different because the linear regression models for each class are independently trained.

Note that, the resulting model after stepwise selection does not guarantee a global optimum, because stepwise selection is based on a greedy search. In addition, other variable selection methods can also be employed in selective stacking. Forward selection and backward selection require less time for training the model, but may provide worse performance. Meta-heuristic optimization methods, such as genetic algorithm and particle swarm optimization, are more likely to find the global optimum, but are computationally expensive.

4.3 Performance Evaluation on Benchmark Datasets

4.3.1 Data Description

The effectiveness of the proposed method was investigated through experiments on benchmark datasets. The following 20 datasets from the UCI repository (Bache & Lichman, 2014) are used in the experiments: *Zoo, Iris, Wine, Parkinson, Sonar, Seed, Glass, Heart, Ecoli, Ionosphere, Movement, Balance, Landcover, BreastCancer, Vehicle, Annealing, Vowel, Yeast, CarEvaluation*, and *Segment*. Detailed descriptions for these datasets are listed in Table 4.1.

4.3.2 Experimental Settings

Eight well-known, widely used one-class classification algorithms were employed for the experiments: GAUSS, MOG, PARZEN, NNDD, SVDD, KMEANS,

Table 4.1 Data summary

Dataset	N. instances	N. features	N. classes
Zoo	101	16	7
Iris	150	4	3
Wine	178	13	3
Parkinson	195	22	2
Sonar	208	60	2
Seed	210	7	3
Glass	214	9	6
Heart	303	13	2
Ecoli	336	7	8
Ionosphere	351	34	2
Movement	360	90	15
Balance	625	4	3
Landcover	675	147	9
BreastCancer	683	9	2
Vehicle	846	18	4
Annealing	898	38	6
Vowel	990	10	11
Yeast	1484	8	10
CarEvaluation	1728	6	4
Segment	2310	19	7

PCA, and AUTOENC. All the aforementioned algorithms were implemented in the Data Description Toolbox for MATLAB (Tax, 2014). Brief descriptions for the algorithms are provided in Section 2.1.

Depending on how to construct meta-classifiers, Three versions of the pro-

posed method (Stacking, StackingC, SelectiveStacking) were investigated in the experiments. For SelectiveStacking, stepwise selection was used as a variable selection method, and the p-value threshold to enter and remove variables was set to 0.1 and 0.2, respectively. The proposed method was compared with the eight benchmarks (GAUSS, MOG, PARZEN, NNDD, SVDD, KMEANS, PCA, AUTOENC), each of which constructs an ensemble of one-class classifiers using a single respective algorithm. For each benchmark, T-norm was used for normalizing the base classifiers of the ensembles because it showed better performance than other methods according to the literature (Tax & Duin, 2008). In addition, the majority vote of the classification results obtained from the eight benchmarks (VOTE) was used for comparison. All numeric input variables were scaled to $[-1, 1]$ for the implementation of each method. All experiments were performed on MATLAB.

As a performance measure for the methods, the misclassification error rate on the test dataset was used, which is defined by $(1/N) \sum_{t=1}^{N} 1_{y_t \neq \hat{y}_t} \times 100$, where N is the number of test instances, y_t is the target label of the t-th instance, $\hat{y}_t$ is the predicted label of the t-th instance, and $1_{y_t \neq \hat{y}_t}$ is an indicator function that has the value of 1 when $y_t \neq \hat{y}_t$.

The performance of each method was computed based on the ten-fold cross test procedure. In this procedure, the original dataset is partitioned into ten disjoint, equal-sized subsets, nine of which are used as the training and validation datasets, and the test error is calculated for the remaining set. This process is repeated for ten runs independently such that all the ten subsets are used exactly once for test.

Table 4.2 Parameter settings for each algorithm

Algorithm		Abbreviation	Parameter Setting
Density-based	Gaussian	GAUSS	-
	Mixture of Gaussians	MOG	N. Gaussians=$2, 3, \ldots, 7$
	Parzen Window	PARZEN	Kernel type=Gaussian Kernel width=$2^{-5}, \ldots, 2^{5}$
Boundary-based	Nearest Neighbor	NNDD	Distance type=Euclidean
	Support Vector Data Description	SVDD	Kernel type=Gaussian Kernel width=$2^{-5}, \ldots, 2^{5}$
Reconstruction-based	K-Means	KMEANS	N. clusters=$2, 3, \ldots, 7$
	Principal Component Analysis	PCA	Frac. variance=$0.5, 0.6, \ldots, 0.9$
	Auto-encoder Network	AUTOENC	N. hidden nodes=$2, 4, \ldots, 10$

With regard to training and validation, the ratio of the training to the validation dataset was set as $1:1$. The base classifiers were trained using the training set exclusively, where the parameters of each classifier were chosen such that the classification error on the validation set is minimized. The parameter search spaces used in the experiments are listed in Table 4.2. For the benchmark methods, the fraction of rejection, which is the parameter of T-norm, was explored from $\{0.01, 0.02, 0.05, 0.1\}$. For the proposed method, meta-classifiers were trained based on the predictions of base classifiers on the validation set.

4.3.3 Experimental Results

Table 4.3 lists the comparison results of the benchmark and proposed methods in terms of error rate (%). The results show that the proposed method outperforms the benchmarks in 14 out of 20 datasets. Regarding the benchmark methods, VOTE performs better than the other methods in most cases. In general, Stacking and SelectiveStacking outperform VOTE, whereas StackingC does not. For the *Glass, Landcover*, and *Vowel* datasets, the proposed method outperforms the benchmark methods with at least 5% lower error rate. In contrast, the proposed method is relatively worse for the *BreastCancer, Vehicle*, and *Annealing* datasets.

Table 4.4 shows the average selection number of each candidate algorithm for SelectiveStacking. Although classifiers of SVDD and PARZEN were selected slightly more, classifiers of all the algorithms were evenly selected on most datasets. It means that the heterogeneous ensembles constructed by SelectiveStacking exploits a diverse of classifiers from various one-class classification al-

Table 4.3 Error rate (%) comparison results on benchmark datasets

Dataset	Benchmark									Proposed		
	GAUSS	MOG	PARZEN	NNDD	SVDD	KMEANS	PCA	AUTOENC	VOTE	Stacking	StackingC	SelectiveStacking
Zoo	27.723	39.604	24.752	53.465	34.653	**23.762**	24.752	25.743	**23.762**	33.663	23.762	24.752
Iris	**4.667**	12.000	7.333	16.667	8.000	7.333	6.667	6.000	**4.667**	5.333	5.333	**4.667**
Wine	9.551	38.764	5.618	28.652	6.180	8.989	4.494	4.494	5.618	3.933	3.371	**2.247**
Parkinson	18.974	21.026	16.923	23.077	16.410	16.410	19.487	15.897	14.359	11.282	15.897	**10.256**
Sonar	40.865	49.519	26.923	26.442	23.558	23.558	22.596	27.885	**16.827**	20.673	25.962	19.712
Seed	**6.667**	9.048	10.000	28.095	8.095	8.571	20.000	8.571	**6.667**	7.619	9.524	8.095
Glass	52.336	51.402	42.991	59.346	52.804	46.262	50.000	48.598	46.262	42.056	42.991	**35.514**
Heart	23.490	25.168	18.456	48.322	20.134	19.463	25.839	24.161	20.134	19.463	19.463	**18.456**
Ecoli	24.405	21.131	21.131	37.202	19.643	20.536	31.548	22.024	20.536	18.750	16.667	**15.476**
Ionosphere	31.909	19.088	21.083	21.652	20.228	16.809	8.262	9.402	6.838	**5.128**	12.251	5.698
Movement	64.167	88.333	35.556	70.000	62.500	34.444	35.000	42.222	29.444	32.778	30.000	**26.944**
Balance	9.598	9.923	13.126	21.614	16.798	15.361	12.003	14.231	11.515	**8.166**	10.083	8.318
Landcover	69.037	82.815	28.000	74.963	29.037	27.852	34.815	32.741	27.852	**18.667**	23.407	18.963
BreastCancer	4.255	12.441	**2.788**	17.148	3.518	3.081	6.158	3.960	2.935	3.081	2.935	2.935
Vehicle	**17.612**	22.459	38.771	45.272	50.000	41.608	31.087	23.168	21.631	22.459	31.087	26.123
Annealing	**3.008**	6.683	16.385	23.615	23.049	15.938	14.598	13.818	13.929	3.124	9.466	3.238
Vowel	19.192	25.960	21.515	33.636	47.576	24.444	31.818	22.626	15.051	**6.162**	19.293	8.687
Yeast	51.954	46.294	46.024	67.251	66.173	51.348	55.458	56.469	49.057	42.251	47.372	**41.375**
CarEvaluation	7.347	9.376	13.198	30.214	17.999	13.888	16.956	17.421	11.919	**6.078**	8.162	6.541
Segment	9.481	8.398	11.732	24.848	56.061	11.948	14.459	12.814	7.446	6.277	9.004	**6.190**

Table 4.4 Average selection number of each candidate classification algorithm for SelectiveStacking on benchmark datasets

Dataset	N. classes	GAUSS	MOG	PARZEN	NNDD	SVDD	KMEANS	PCA	AUTOENC	N. classifiers used	N. classifiers total	Ratio used (%)
Zoo	7	1.2	1.1	1.1	1.0	1.0	1.2	0.9	1.2	8.6	56	15.4
Iris	3	0.7	0.9	0.7	0.4	1.2	0.6	0.5	0.8	5.8	24	24.3
Wine	3	0.6	0.2	1.1	0.8	1.1	0.5	0.5	0.5	5.3	24	22.1
Parkinson	2	0.5	0.7	1.2	0.6	1.0	0.6	0.3	0.3	5.2	16	32.5
Sonar	2	0.3	0.1	1.3	1.2	1.0	0.6	0.8	0.6	5.9	16	36.9
Seed	3	0.7	0.4	1.2	0.8	1.0	0.6	1.2	0.7	6.5	24	27.1
Glass	6	0.8	0.9	1.1	0.7	1.0	0.8	0.8	0.9	7.0	48	14.6
Heart	2	0.0	0.0	1.0	0.3	0.6	0.4	0.3	0.2	2.8	16	17.5
Ecoli	8	0.4	0.9	1.8	0.8	0.7	0.8	0.7	0.8	6.9	64	10.8
Ionosphere	2	0.2	0.2	0.8	0.3	1.0	1.6	0.6	0.5	5.2	16	32.5
Movement	15	2.3	1.2	3.3	2.3	2.0	2.1	1.9	2.6	17.8	120	14.8
Balance	3	1.0	1.4	1.0	0.9	1.0	0.8	0.8	0.6	7.6	24	31.5
Landcover	9	2.0	0.0	3.2	1.8	2.5	1.9	2.0	2.1	15.5	72	21.6
BreastCancer	2	0.3	1.3	1.3	0.9	0.8	0.9	0.8	0.6	6.9	16	43.1
Vehicle	4	1.4	1.5	2.2	0.8	1.7	1.1	1.5	1.2	11.3	32	35.2
Annealing	6	1.9	1.1	2.3	1.4	1.3	1.8	2.1	2.0	13.9	48	28.9
Vowel	11	2.7	2.4	5.1	2.7	3.4	3.3	2.9	3.4	25.8	88	29.3
Yeast	10	0.6	1.6	2.6	0.8	1.7	1.7	1.4	1.4	12.0	80	15.0
CarEvaluation	4	1.8	3.7	1.9	1.5	1.1	1.6	1.4	1.1	14.1	32	43.9
Segment	7	3.9	2.4	4.4	2.2	4.3	4.5	4.1	4.0	29.7	56	53.1

Table 4.5 Results of the Holm post-hoc test

Method	Average Rank	Holm APV	Hypothesis (α=0.05)
SelectiveStacking	2.2	-	-
Stacking	2.875	0.554	Not rejected
VOTE	3.95	0.250	Not rejected
StackingC	4.5	0.131	Not rejected
PARZEN	6.675	3.47×10^{-4}	Rejected
KMEANS	6.75	3.30×10^{-4}	Rejected
GAUSS	6.825	2.99×10^{-4}	Rejected
AUTOENC	7.425	3.21×10^{-5}	Rejected
PCA	8.05	2.31×10^{-6}	Rejected
MOG	8.4	4.86×10^{-7}	Rejected
SVDD	8.9	4.20×10^{-8}	Rejected
NNDD	11.45	5.44×10^{-15}	Rejected

gorithms effectively. The ensembles selected averagely about 27.5% of the total classifiers. In the case of *Segment*, *CarEvaluation*, and *BreastCancer*, almost half of total were selected, whereas only a small number of classifiers were selected for some datasets such as *Ecoli*.

The statistical significance of the comparison results was investigated via non-parametric statistical tests (Demšar, 2006; García & Herrera, 2008; García et al., 2009, 2010). Firstly, the Friedman test was performed to find out the statistical difference of the performance among the compared methods. The Friedman test returned the p-value as 1.0263×10^{-10}, which indicates that the null hypothesis of equivalence is rejected with a high confidence. Therefore, the performances of those methods are statistically different. In particular, Selec-

Table 4.6 Results of the Wilcoxon signed-rank test comparing SelectiveStacking, Stacking, and VOTE

Comparison	R^+	R^-	p-value	Hypothesis (α=0.05)
SelectiveStacking vs. Stacking	152	19	0.00189	Rejected
SelectiveStacking vs. VOTE	148	23	0.00325	Rejected
SelectiveStacking vs. StackingC	188	2	9.11×10^{-5}	Rejected
Stacking vs. VOTE	155	55	0.0310	Rejected
Stacking vs. StackingC	136	35	0.0139	Rejected
VOTE vs. StackingC	87	84	0.474	Not rejected

tiveStacking performed best with an average rank of 2.2, followed by Stacking and VOTE. Accordingly, the Holm post-hoc test was conducted to compare the method that was adjudged as having the best performance as per the Friedman test, SelectiveStacking, with the other methods. The results, which are listed in Table 4.5, show that SelectiveStacking statistically outperforms the benchmark methods.

The Holm post-hoc test, however, did not report the difference between SelectiveStacking, Stacking, VOTE, and StackingC. Hence, the Wilcoxon signed-rank test was conducted for pairwise comparison of these methods. The results obtained from the Wilcoxon signed-rank test are listed in Table 4.6. The results show that Selective Stacking outperforms all the other methods, and Stacking outperforms VOTE and StackingC, with a significance level of 0.05. There were no significant differences between VOTE and StackingC. In conclusion, SelectiveStacking shows the best performance with statistical significance, while Stacking also statistically outperforms all the benchmark methods.

Table 4.7 Data summary

Dataset	N. categories	N. terms	N. documents	N. training	N. test
Reuters-21578 (top10)	10	18,993	7,285	5,228	2,057
20Newsgroups	20	26,214	18,846	11,314	7,532

4.4 Application to Text Categorization

4.4.1 Problem Definition

Text categorization is to classify documents into a pre-defined set of categories according to their contents. Each document can be included in single or multiple categories, or not be included in any category. As the number of documents in internet rapidly grows, organizing the documents becomes an important issue. Machine learning enables automatic categorization of the documents by training a classifier with a set of documents that has their category labels and classifying new documents using the classifier. Since the number of categories is usually large, this task is defined as a typical multi-class classification problem. In general, periodic retraining of the classifier is required for this problem because a novel category is often added and the characteristics of categories changes gradually with time. Therefore, one-class classifier approach may be advantageous because of its merits. In this section, the effectiveness of HEOC is compared with other multi-class classification algorithms for text categorization problems.

4.4.2 Data Description

The two most popular datasets, *Reuters-21578* and *20Newsgroups*, were employed. The *Reuters-21578*[1] dataset contains 21,578 news articles of 135 categories collected from the Reuters Newswire in 1987. In this dataset, the distribution of the number of articles in the categories is highly unbalanced. Only the top ten categories were used for this experiment. The *20Newsgroups*[2] dataset is a collection of articles from the Usenet Newsgroup. It includes 19,997 articles that are evenly distributed across 20 categories.

For each dataset, several preprocessing steps such as tokenization, stemming, and stopword removal are required. In the experiments, the preprocessed versions of the datasets used in Cai et al. (2009)'s study were adopted. Table 4.7 shows the detail.

The documents have to be transformed into suitable representations in order to train a classifier. The top 1,000 important terms were selected for each dataset using information gain (Yang & Pedersen, 1997). After selecting the term subset T, each document was represented by the term vector $\mathbf{d}_j$ as below:

$$\mathbf{d}_j = \left(w_{i,j}, w_{2,j}, \ldots, w_{|T|,j}\right) \tag{4.2}$$

where $w_{i,j}$ is the weight of the term i in the document j, and $|T|$ is the cardinality of T. Each weight $w_{i,j}$ was calculated using the normalized *tf-idf*, which is

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2] http://qwone.com/~jason/20Newsgroups/

Table 4.8 Parameter settings for each algorithm

Algorithm	Parameter Setting
SVM	C=$2^{-3}, \ldots, 2^{10}$
	Kernel type=Gaussian kernel
	σ=$2^{-5}, \ldots, 2^{5}$
kNN	k=1, 3, 5, 7, 10, 20, 30
	Distance type=cosine
DT	Min. instances in a leaf node=1, 2, 3, 5
	Min. instances in a parent nodes=5, 10
	Prune=true

generally used for text categorization (Joachims, 2002):

$$w_{i,j} = \frac{tf_{i,j} \times \log \frac{|D|}{df_i}}{\sqrt{\sum_{k=1}^{|T|} \left(tf_{k,j} \times \log \frac{|D|}{df_i} \right)^2}}, \tag{4.3}$$

where $tf_{i,j}$ is the number of occurrences of the term i in the document j, df_i is the number of documents containing the term i, $|D|$ is the cardinality of the document set D.

4.4.3 Experimental Settings

For the proposed method, heterogeneous ensemble of one-class classifiers based on SelectiveStacking is used. The proposed method was compared with the three popular classification algorithms: SVM, kNN, and DT. Parameter settings used for those algorithms are given in Table 4.8.

The effectiveness of the methods was measured using the micro- and macro-

F1. The F1 score is the harmonic mean of recall and precision. The micro-F1 score is the global calculation of F1 score regardless of categories, and the macro-F1 score is the average of F1 scores of all the categories. Other settings were the same as in Section 4.3.

4.4.4 Experimental Results

Table 4.9 shows the mean and standard deviation of classification performance over ten independent run for each method in terms of micro- and macro-F1. The best among homogeneous ensembles of one-class classifiers showed relatively worse performance for text categorization problems, whereas the classification performance of the proposed method, homogeneous ensemble of one-class classifiers, was similar to that of one-against-one SVM. Although one-against-rest SVM yielded the best performance, it is shown that the proposed method achieves comparable performance for text categorization problems, compared with the state-of-the-art multi-class classifiers.

4.5 Summary

Constructing a multi-class classifier based on an ensemble of one-class classifiers has proven to be a promising strategy to solve multi-class classification problems. Previous work on this strategy only used a single one-class classification algorithm to construct the ensemble; however, the classification accuracy of the ensemble can be improved by using various classification algorithms to train base classifiers. Moreover, an appropriate normalization of the prediction

Table 4.9 Micro- and macro-F1 comparison results on text categorization problems

Dataset	Measure	SVM-OAO	SVM-OAR	kNN	DT	Homogeneous Ensemble of OCs (best)	Heterogeneous Ensemble of OCs
Reuters-21578 (top10)	micro-F1	95.727(0.198)	**96.422(0.241)**	94.560(0.286)	89.694(1.046)	94.088(0.675)	95.737(0.230)
	macro-F1	90.134(0.701)	**91.941(0.867)**	87.245(0.675)	79.213(2.010)	86.132(2.277)	90.257(0.792)
20Newsgroups	micro-F1	72.553(0.302)	**74.385(0.609)**	59.765(1.546)	53.968(0.516)	70.762(0.382)	72.570(0.240)
	macro-F1	71.906(0.326)	**73.226(0.540)**	59.142(1.380)	53.458(0.581)	69.744(0.367)	71.133(0.235)

scores obtained from the base classifiers allows improving this strategy further.

This chapter aimed to construct a heterogeneous ensemble of one-class classifiers using various one-class classification algorithms to address the problem of multi-class classification. We proposed a multi-class classification method that utilized stacking with MLR to combine the heterogeneous base classifiers. The proposed method used an MLR model as a meta-classifier to successfully address the normalization issue of the heterogeneous ensemble. In addition, we proposed selective stacking to exclude the base classifiers that degrade the overall performance when training the MLR model. In experimental results, the proposed method yielded higher classification accuracy than other methods with statistical significance.

The main contribution of this chapter is two-fold. First, the proposed method solved a multi-class classification problem using a heterogeneous ensemble of one-class classifiers trained by various classification algorithms. Second, the proposed method successfully normalized the scores from those heterogeneous base classifiers through stacking with MLR. These two points distinguish the proposed method from the existing work. Experimental results confirmed the effectiveness of stacking in combination with heterogeneous one-class classifiers for multi-class classification.

Chapter 5

Heterogeneous Ensemble for Reliable Classification

5.1 Multi-class Classification with a Reject Option

Solving a classification problem involves training a classifier using a set of given instances in order to find the relationship between input and output variables to predict the class of new instances using the trained classifier. In most cases, prediction error is inevitable because of the imperfection of the classifier. As a solution, a reject option can support improving classification reliability by rejecting instances that are difficult to classify (Kang & Cho, 2015a). This is beneficial to various real-world applications that have high misclassification costs and require high prediction accuracy, such as medical prediction, user authentication and equipment fault detection.

The reject option has been actively studied by researchers, mostly focusing on binary classification problems. In particular, many researchers have dealt with the optimal trade-off between accuracy and rejection for a given misclassification cost (Landgrebe et al., 2006; Tortorella, 2005). Embedded reject

options for specific classification algorithms have also been studied (Fumera & Roli, 2002; Simeone et al., 2012; Kang & Cho, 2015a).

Existing research on reject options has mostly focused on binary classification (Chow, 1970; Landgrebe et al., 2006; Tortorella, 2005; Herbei & Wegkamp, 2006), whereas relatively few of these researchers are related to multi-class classification (Cecotti & Vajda, 2013; Tax & Duin, 2008). However, multi-class classification, which predicts that an output variable consists of more than two categories, is advantageous because these prediction results are more informative than those of binary classification. For example on anti-diabetic drug failure prediction, predicting treatment results in the multiple categories of the degree of glycemic control offers better applicability in practice. Glycated hemoglobin ($\mathrm{HbA_{1c}}$) is widely adopted as an indicator of glycemic status to measure efficacy (Bennett et al., 2007; Lu et al., 2010), and the American Diabetes Association (ADA) recommends $\mathrm{HbA_{1c}} < 7.0\%$ as a general glycemic goal and $\mathrm{HbA_{1c}} < 8.0\%$ as a less stringent goal (American Diabetes Association, 2014).

Considering a reject option for multi-class classification, a typical approach is to reject instances that have low posteriors for any class (Cecotti & Vajda, 2013; Tax & Duin, 2008). Suppose that $\boldsymbol{x} \in \mathbb{R}^d$ is an input vector, $y \in \{1, 2, \ldots, c\}$ is the corresponding target class, and $p(y = j|\boldsymbol{x})$ is the posterior of the instance $\boldsymbol{x}$ belonging to the j-th class. This approach can be represented by the following rule:

$$\hat{y} = \begin{cases} \arg\max_j p(y = j|\boldsymbol{x}), & \text{if } \max_j p(y = j|\boldsymbol{x}) \geq \theta \\ \text{reject}, & \text{otherwise.} \end{cases} \tag{5.1}$$

The maximum posterior, $\max_j p(y = j|\boldsymbol{x})$, is used as the confidence of the prediction, and the instance is rejected when the confidence is below the pre-defined threshold θ. Otherwise, classification is performed by choosing the class with the maximum posterior. The rejection rule in Equation 5.1 can be divided into two steps. The first step is to decide whether to classify, and the second step is to classify. The main idea of this chapter is that the best classifier of each step may differ.

In this chapter, we propose a hybrid reject option (Kang, Cho, Rhee, & Yu, 2015) based on heterogeneous ensemble learning to achieve better trade-off between accuracy and rejection for multi-class classification problems. The proposed method is based on a serial ensemble structure, aiming at designing a reject option with a sequence of two classifiers. The proposed method constructs a filter (a classifier for rejection) and a predictor (a classifier for prediction) separately, by employing different classification algorithms. Thus, we can classify with the reject option for a new instance by estimating confidence using the filter and making a prediction using the predictor if the confidence is above a pre-defined threshold.

The proposed method is promising in that it is applicable to solving multi-class classification problems effectively and can be used regardless of the classification algorithms. Moreover, the trade-off between accuracy and rejection can be controlled easily. We examine the effectiveness of the proposed method through experiments on the anti-diabetic drug failure prediction problem using EMR data from the Seoul National University Hospital (SNUH) in the Republic of Korea.

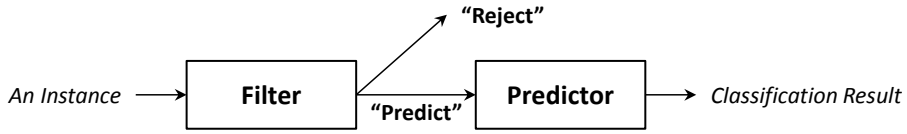


Figure 5.1 Classification with reject option

5.2 Hybrid Reject Option

In this section, we introduce our proposed hybrid reject option. Before beginning, Equation 5.1 can be transformed as Equation 5.2 by dividing the posterior p into p_f and p_p. The p_f is the posterior obtained by the filter classifier that estimates the confidence of an instance for prediction as $\max_j p_f(y = j|\boldsymbol{x}) \geq \theta$, and decides whether to predict using the predictor classifier on the basis of the confidence. The predictor classifier provides the posterior p_p that finally predicts the class of those that the filter does not reject, as $\arg\max_j p_p(y = j|\boldsymbol{x})$. Figure 5.1 represents a reject option based on Equation 5.2, consisting of the filter and predictor.

$$\hat{y} = \begin{cases} \arg\max\limits_j p_p(y = j|\boldsymbol{x}), & \text{if } \max\limits_j p_f(y = j|\boldsymbol{x}) \geq \theta \\ \text{reject}, & \text{otherwise.} \end{cases} \tag{5.2}$$

A classifier generally plays the roles of filter and predictor for a reject option, where $p_f(y = j|\boldsymbol{x}) = p_p(y = j|\boldsymbol{x})$. In contrast, the main idea of this work is the separate use of filter classifier $\mathcal{C}_f$ and predictor classifier $\mathcal{C}_p$, which are used to obtain the respective values of $p_f(y = j|\boldsymbol{x})$ and $p_p(y = j|\boldsymbol{x})$. Each classifier is trained using the best respective classification algorithm to maximize

Algorithm 4 Training phase of the hybrid reject option

Input: training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, classification algorithms $\mathcal{A}_f, \mathcal{A}_p$

Output: filter $\mathcal{C}_f$, predictor $\mathcal{C}_p$

1: **procedure** TRAINING

2: $\mathcal{C}_f \leftarrow$ filter classifier trained from $\mathcal{D}$ using $\mathcal{A}_f$

3: $\mathcal{C}_p \leftarrow$ predictor classifier trained from $\mathcal{D}$ using $\mathcal{A}_p$

4: **end procedure**

the capability of the classifier's role. Thus, we aim to improve reject option performance as the accuracy increases for the same degree of rejection.

The training phase for the proposed method involves training of the filter and predictor. Suppose that a training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector and $y_i \in \{1, 2, \ldots, c\}$ is the corresponding target value, the classification algorithm for filter $\mathcal{A}_f$, and the classification algorithm for predictor $\mathcal{A}_p$, are given. Regarding the classification algorithms, we assume that classifiers from them provides a posterior for each class when classifying an instance. First, the filter classifier $\mathcal{C}_f$ is trained with $\mathcal{D}$ using the classification algorithm $\mathcal{A}_f$. Then, the predictor classifier $\mathcal{C}_p$ is also trained with $\mathcal{D}$ using $\mathcal{A}_p$. The resulting two classifiers, $\mathcal{C}_f$ and $\mathcal{C}_p$, constitute a hybrid classifier system for the proposed method. Algorithm 4 represents the pseudocode for the training phase.

In the test phase, classification for a new instance $\boldsymbol{x}$ is performed as follows. First the posterior $p_f(y = j|\boldsymbol{x})$ is computed using the filter $\mathcal{C}_f$ for each class, and the confidence becomes $\max_j p_f(y = j|\boldsymbol{x})$. When the confidence value

Algorithm 5 Test phase of the hybrid reject option

Input: new instance $\boldsymbol{x}$, filter $\mathcal{C}_f$, predictor $\mathcal{C}_p$, threshold θ

Output: predicted label $\hat{y}$

```
1: procedure TEST
2:     {p_f(y = j|x)}_{j=1}^c ← C_f(x)
3:     confidence(x) ← max_j p_f(y = j|x)
4:     if confidence(x)< θ then
5:         ŷ ←NULL
6:     else
7:         {p_p(y = j|x)}_{j=1}^c ← C_p(x)
8:         ŷ ← arg max_j p_p(y = j|x)
9:     end if
10: end procedure
```

is smaller than the pre-defiend threshold θ, we reject classification for the instance $\boldsymbol{x}$. Otherwise, the posterior $p_p(y = j|\boldsymbol{x})$ is computed using the predictor $\mathcal{C}_p$, and the instance is classified according to the class with the maximum posterior, $\arg\max_j p_p(y = j|\boldsymbol{x})$. The pseudocode of this procedure is described in Algorithm 5.

The merits of using the proposed hybrid reject option are as follows. It can be applied to multi-class classification directly, regardless of classification algorithms. We can easily control the trade-off between accuracy and rejection by changing the rejection threshold. In addition, several classifiers can constitute a hybrid filter by aggregating confidences obtained from the classifiers.

5.3 Application to Anti-diabetic Drug Failure Prediction

5.3.1 Problem Definition

In recent times, diabetes is one of the most prevalent diseases, with the number of patients increasing continuously. Among these patients, type 2 diabetes accounts for 85∼90% (Bennett et al., 2007), and therewith, research on the treatment of those patients has been actively conducted. Most patients with type 2 diabetes are under treatment by ingesting oral hypoglycemic agents and insulin in order to achieve the desired glucose level.

When a patient with type 2 diabetes receives a prescription for treatment, he or she can get check the results after 2∼6 months on the medication. Predicting the prescription results in advance is an important issue. It is much more important to predict the results for those whose treatment fail. However, this is difficult because the treatment results of type 2 diabetes are highly related to multiple factors such as patient characteristics, type of treatment, and presence of complications. Moreover, the efficacy can also be affected by the interaction of various drugs. In order to deal with the complex relationship of these various factors, a machine learning approach based on Electronic Medical Records (EMR) data is being considered to predict the treatment results (Huang et al., 2007; Kang, Kang, et al., 2015).

In particular, this prediction problem can be defined as a typical classification problem in machine learning. For the classification problem, a classifier is trained using a set of instances in which an instance consists of such factors,

with the prescription as input variables and the results as output variables. This classifier predicts results caused by an instance for a current prescription. However, real-world deployment of this approach is limited due to high misclassification costs, especially misclassification for treatment failure, in the medical domain. To cope with this limitation, it is much better for human experts to examine those for whom results are difficult to predict than for the classifier to predict fully.

Therefore, we must consider a reject option, which rejects ambiguous instances instead of predicting for all instances. Thus, the rejected instances can be conveyed to human experts for careful investigation. The reject option supports reliable prediction for a classification problem when the classifier is imperfect because of fundamental reasons, such as noisy data, inductive bias of the classifier, and lack of input variables.

Considering the anti-diabetic drug failure prediction problem, the reject option would be effective because some factors, such as personal efforts and patient life styles, are hard to quantify as input variables despite their significant importance. The desired prediction accuracy can be obtained by using only prediction results with high confidence. By doing so, patients can be managed efficiently by concentrating more on those who result in low confidence or are predicted as treatment failure with high confidence.

5.3.2 Data Description

The effectiveness of the proposed hybrid reject option was demonstrated through experiments on the anti-diabetic drug failure prediction. The EMR data for

patients with type 2 diabetes, which was collected from the SNUH during 2008∼2012, was used in the experiments (Kang, Kang, et al., 2015). Data collection was conducted after the protocol had been approved by the Institutional Review Board of the SNUH. A patient who received the physical examination and the HbA_{1c} test received a prescription within a week. After 2∼6 months, the patient received the test again to check the prescription efficacy. The data were reconstructed so that an instance consists of a prescription record, the patient characteristics at this moment, and the corresponding before/after HbA_{1c} test results. Instances with missing values were excluded. As a result of the preprocessing, we obtained a total of 27,836 instances of 2,995 patients.

The input and output variables used in the experiments are summarized in Table 5.1. The input variables include patient characteristics, prescriptions for diabetes and others, and the HbA_{1c} test record, so that a patient's condition and presence of complications were also considered. Regarding a prescription for diabetes, we used a daily dose of each hypoglycemic agent, except that insulin is binary. Regarding prescriptions for others, we used variables in which each variable indicates whether a drug category is prescribed. All continuous input variables were normalized to be in the scale of $[0, 1]$. The output variable was divided into three categories based on the HbA_{1c} test record after 2∼6 months of the prescription: C1(general goal, < 7.0), C2(less stringent goal, ≥ 7.0 and < 8.0), and C3(failure, ≥ 8.0). Table 5.2 describes the distribution of the output variable. Note that, the ratio of C1 increases, whereas that of C2 and C3 decreases with time.

Table 5.1 Variable description

Category		N. Variables	Variable Names	Variable Type
Input Variables	Patient characteristics	7	Gender, Age, Height, Weight, BMI (Body Mass Index), SBP (Systolic Blood Pressure), DBP (Diastolic Blood Pressure)	Continuous
	Prescription for diabetes	18	Acarbose, Exenatide, Glibenclamide, Gliclazide, Glimepiride, Gliquidone, Insulin (injection), Linagliptin, Metformin, Mitiglinide, Nateglinide, Pioglitazone, Repaglinide, Rosiglitazone, Saxagliptin, Sitagliptin, Vildagliptin, Voglibose	Continuous (daily dose)
	Prescription for others	16	Non-steroidal anti-inflammatory drugs, Endocrine agents, Anesthetics, Immunosuppressant & Electrolyte supplements, Alimentary tract and metabolism, Cardiovascular system, Nervous system, Mineral supplements, Amino acid, Sensory organs, Contrast media, Dermatologicals & Genito-urinary system and sex hormones, Antiinfectives for systemic use, Antineoplastic agents, Antihistamines, Respiratory system	Binary (1: prescribed, 0: not prescribed)
	Test record	1	HbA_1c (within a week before prescription)	Continuous
Output Variable		1	Anti-diabetic drug failure (2-6 months after prescription, C1 when HbA_{1c} <7.0, C2 when ≥7.0 and <8.0, C3 when ≥8.0)	Categorical (C1: general goal, C2: less stringent goal, C3: failure)

Table 5.2 Data summary

Year	N. instances	C1(general goal)	C2(less stringent goal)	C3(failure)
2008	5,009	1,747 (34.9%)	2,121 (42.3%)	1,141 (22.8%)
2009	5,340	1,982 (37.1%)	2,190 (41.0%)	1,168 (21.9%)
2010	5,590	2,306 (41.3%)	2,204 (39.4%)	1,080 (19.3%)
2011	5,829	2,382 (40.9%)	2,314 (39.7%)	1,133 (19.4%)
2012	6,068	2,614 (43.1%)	2,279 (37.6%)	1,175 (19.4%)
Total	27,836	11,031 (39.6%)	11,108 (39.9%)	5,697 (20.5%)

5.3.3 Experimental Settings

For training classifiers as a filter or a predictor, the three classification algorithms evaluated their respective suitability: SVM, RF, and ANN. These algorithms can deal with non-linear relationships and are known to provide high prediction accuracy. RF and ANN provide the posterior of each class for multi-class classification directly, whereas SVM was originally designed for binary classification. Thus, H.-T. Lin et al. (2007)'s method was adopted to estimate the posterior of the SVM based on one-against-rest approach. In addition, we designed two hybrid filters, HYB+ and HYB×, that fuse the confidences of the SVM, RF, and ANN filters. HYB+ and HYB× compute the confidence as summation and multiplication of the confidences from those filters, respectively. All algorithms were implemented in MATLAB.

The sliding window test procedure was conducted to assess each classifier. The basic idea of this procedure is to use past data for training and future data for test; thereby it is appropriate for prediction problems. In our experiments,

Table 5.3 Parameter settings for each algorithm

Algorithm	Parameter Setting
SVM	C=$2^{-3}, \ldots, 2^{10}$
	Kernel type=RBF kernel
	σ=$2^{-5}, \ldots, 2^{5}$
RF	N. Trees=100
	Bootstrap sample size=80%
	Min. instances in a leaf node=$1, 2, 5, 10, 20, 50, 100$
ANN	N. hidden nodes=$3, 4, 5, \ldots, 20$
	Max. iterations=300

the classifiers using the data of year $t-2$ and $t-1$ predicted the data of year t, $t = 2010$, 2011, and 2012, respectively. For each classifier, the best parameters were chosen from the parameter search space in Table 5.3 to maximize validation accuracy. All experiments were performed through ten independent runs by randomizing the index of training and validation, and the results were averaged over those runs for confidence.

In order to evaluate the performance of the reject option, we used the Area under Accuracy-Rejection Curve (AuARC) which plots the accuracy against the rejection rate with a change of the rejection threshold (Nadeem et al., 2010). Thus, it measures the trade-off between accuracy and rejection in a threshold-independent manner.

Table 5.4 AuARC comparison results on anti-diabetic drug failure prediction problem

Year	Predictor	Filter				
(test)		SVM	RF	ANN	HYB+	HYB×
2010	SVM	77.70±0.72	80.83±0.26*	80.81±0.38*	81.16±0.21*	**81.25±0.20***
	RF	78.77±0.38	81.18±0.25	80.86±0.33	81.48±0.16*	**81.54±0.16***
	ANN	78.84±0.46	80.99±0.20	81.03±0.34	81.54±0.15*	**81.60±0.16***
2011	SVM	80.80±0.30	80.94±0.29	81.36±0.26*	**81.82±0.18***	**81.82±0.17***
	RF	80.74±0.39	81.55±0.28	81.46±0.23	82.05±0.20*	**82.06±0.19***
	ANN	80.84±0.30	81.09±0.21	81.45±0.32	81.88±0.17*	**81.89±0.15***
2012	SVM	81.87±0.20	81.69±0.18	82.21±0.30*	82.72±0.19*	**82.73±0.19***
	RF	81.79±0.20	82.07±0.19	82.13±0.24	82.77±0.17*	**82.79±0.17***
	ANN	81.84±0.20	81.81±0.22	82.21±0.29	82.74±0.18*	**82.75±0.18***

5.3.4 Experimental Results

The reject option performance was measured for the 15 combinations of filters and predictors in terms of AuARC. Table 5.4 shows the mean and standard deviation of AuARC values over ten independent runs. A gray-shaded value indicates the baseline reject option, in which the filter is identical to the predictor. The numbers in bold indicate the best AuARC obtained over the filters, and an AuARC marked with an asterisk indicates that the corresponding hybrid reject option is significantly better than the single reject option at the 0.01 level of confidence according to the paired t-test. Overall results show that the hybrid reject options with a hybrid filter outperform the baselines with a statis-

Table 5.5 Accuracy (%) by varying rejection rate for the baseline and hybrid reject options

Year	Method	Filter/Predictor	Rejection Rate (%)					
(test)			0	20	40	60	80	100
2010	Best Baseline	RF/RF	68.12	72.12	76.33	82.37	91.72	100
	Best Hybrid	HYB×/ANN	68.03	72.46	76.62	83.13	92.15	100
2011	Best Baseline	RF/RF	68.92	73.12	77.75	83.21	90.79	100
	Best Hybrid	HYB×/RF	68.92	73.08	77.39	83.32	91.66	100
2012	Best Baseline	ANN/ANN	67.76	72.27	77.51	84.44	93.78	100
	Best Hybrid	HYB×/RF	68.15	72.75	78.04	84.95	94.74	100

tical significance. Regarding the hybrid filter, HYB× was slightly better than HYB+. According to the results, the proposed hybrid reject option improves the trade-off between accuracy and rejection.

We compared the best hybrid reject option with the best baseline for each year with respect to the accuracy corresponding to rejection rate change. Table 5.5 shows the comparison results. The accuracy improved as the rejection rate increased. The hybrid reject option mostly yielded a better accuracy than the baseline for the same rejection rate. In particular, the gap was greater when the rejection rate was large. Overall, rejection of 50% and 80% provided approximately 80% and over 90% of accuracy, respectively. Considering the dataset does not reflect the individual habits and efforts that were not available for this study, this classification accuracy can be said to be a meaningful result.

The comparison was also performed in terms of the miss rate, which is

Table 5.6 Miss rate (%) by varying rejection rate for the baseline and hybrid reject options

Year	Method	Filter/Predictor	Rejection Rate (%)					
(test)			0	20	40	60	80	100
2010	Best Baseline	RF/RF	31.76	26.05	18.40	9.14	1.02	0
	Best Hybrid	HYB×/ANN	26.12	20.96	15.16	6.39	0.70	0
2011	Best Baseline	RF/RF	25.38	19.87	12.10	4.25	0.25	0
	Best Hybrid	HYB×/RF	25.38	18.04	10.26	2.73	0.15	0
2012	Best Baseline	ANN/ANN	21.83	15.85	9.12	1.73	0.48	0
	Best Hybrid	HYB×/RF	26.16	18.25	9.37	1.55	0.22	0

defined as the ratio of C3 instances that are predicted as C1 or C2. Thus, the miss rate looks for misclassification for the failure, which is the major interest for this prediction problem. As shown in Table 5.6, we found that the miss rate decrease when more instances were rejected, and the hybrid reject option is superior to the baseline. The miss rate of the hybrid reject option was reduced by 50% and less than 1% for 40% and 80% rejection, respectively.

5.4 Summary

For medical prediction, such as anti-diabetic drug failure prediction, it is essential to ensure high prediction accuracy, but achieving this is difficult in practice. A reject option can be considered to avoid prediction for uncertain instances to address the problem. In this chapter, we proposed a hybrid reject option for

multi-class classification to achieve the better trade-off between accuracy and rejection.

The proposed method utilizes the two components, filter and predictor, for a reject option. Each component is trained separately using the best respective algorithm, thereby improving the reject option performance. This method can be applied to multi-class classification directly regardless of classification algorithms. The proposed method was applied to the anti-diabetic drug failure prediction problem for type 2 diabetes. We conducted experiments on the EMR data for patients with type 2 diabetes to predict drug failure in the three categories. As a result, the proposed method was found to be effective for achieving a better trade-off of accuracy and rejection. In particular, the hybrid filter, which fuses confidences from different classifiers, yielded the best performance for every case.

The desired prediction accuracy can be obtained by controlling the rejection threshold. By ensuring the desired accuracy in practice, treatment efficacy can be predicted at the moment of prescription, which offers an in-depth analysis to those prescriptions that are predicted as failure or rejected. Furthermore, it is expected to provide clues to discovering additional input variables that are not considered currently, by analyzing prescription cases with results that are difficult to predict. Thus, the prediction can be enhanced by adopting those additional input variables.

Chapter 6

Heterogeneous Ensemble for Fast Classification

6.1 Run-time Speed on Multi-class Classification

The main concern of classification problems is to achieve a high classification accuracy. Besides, there are several issues in deployment of classifiers to real-world problems. One is the training and test speed of a classifier. The majority of studies focus on accelerating the training speed of the classifier. The test speed of the classifier, however, is much more important in real-world deployments. In most cases, training is carried out before applying the classifier, and often not subject to a time limit. On the other hand, the classifier works under time constraints when deployed to make predictions on new instances, hence, fast test speed is demanding. This issue can be interpreted as acceleration of the run-time speed of the classifier.

The issue becomes more serious when solving a multi-class classification problem because it generally involves construction of an ensemble based on decomposition strategies. The ensemble requires a large computational burden

in its test phase because it is constituted by multiple classifiers. Taking into account of SVM, an ensemble have to be constructed to solve a multi-class classification problem because SVM is originally designed for binary classification. In addition, SVM requires more computational burden to classify an instance, while it usually yields better classification accuracy than other classification algorithms. Thus, it is important to reduce the computational burden for practical deployments.

In the test phase, SVM takes $\mathcal{O}(N_{SV})$ of its computational complexity, where N_{SV} is the number of support vectors. By alleviating its complexity, the usability of SVM for real-time applications would be enhanced. To lessen the computational burden of SVM in the test phase, many researchers have focused on reducing the number of support vectors directly (Burges, 1996; Downs et al., 2002; Y. Li et al., 2006; Q. Li et al., 2007; Liang et al., 2013; H.-J. Lin & Yeh, 2009), and approximating kernel functions (Maji et al., 2008, 2013; Vedaldi & Zisserman, 2012). These efforts just tried to modify an SVM partially, whereupon they were limited to the structural characteristics of the SVM. In order to obtain better results, heterogeneous ensemble learning, which put together with different models, can be taken into account.

There are two major types of heterogeneous ensemble methods to increase the test speed for complex algorithms. The first approach is to use a simple, fast model that roughly processes the test instances. Only uncertain or crucial instances are processed by a complex, high-performance model. This approach reduces the number of test instances directly processed by the complex model to lessen its test complexity. Several studies employed an SVM as the complex

model. X. Xu et al. (2005) employed a Rocchio classifier as a simple model for text categorization problems. M. A. Kumar and Gopal (2010) and Ji and Zhao (2013) used DT and kNN, respectively, for binary classification problems. The second approach is to have a simple, fast model that replicates a complex model. This approach can be thought of as a function approximation problem. Schmitz et al. (1999) presented a DT-based approximation of an ANN to yield fast test speed and high interpretability with little prediction loss. Chen and Chen (2004) used multiple simple classifiers to approximate an SVM for classification problems. Applying these approaches would be beneficial to improve the test speed.

In this chapter, we propose a heterogeneous ensemble method called Neural Network Approximator (NNA) (Kang & Cho, 2014) that is an ANN regression-based method for approximation of a complex classifier. The proposed method follows the second approach and seeks to approximate the classifier using an ANN approximator to achieve a high test speed without sacrificing classification accuracy. For a multi-class classification problem, NNA adopts a multiple-outputs ANN whose output nodes correspond to the several decision functions of the classifier. Only a few number of decision function should be approximated regardless of the number of base classifiers involved. Thus, we can yield similar prediction results with less computational burden by employing the ANN as an approximator of the classifier. Several experiments are conducted on benchmark datasets in order to verify the effectiveness of the proposed method for approximation of SVM ensembles.

6.2 Neural Network Approximator

It is well-known that ANN, on its own, shows good fitting performance for regression problems. In addition, the fitting performance of the regression algorithm for a function is better than that for a noisy dataset. Paying attention to this point, we introduce a method, named NNA, to approximate a classifier using an ANN. The proposed method solves multiple regression problems to predict the function values of the decision functions. The decision functions in the classifier are approximated with nodes of a multiple-outputs ANN. By doing so, the classifier plays a role in removing noise from the dataset, and the ANN just approximates the refined dataset generated by the classifier. The aim of the proposed method is to obtain the ANN approximator that has the same classification accuracy and requires less test time compared to the classifiers.

To demonstrate the appromiation performance of the ANN, Figure 6.1 illustrates the functions obtained by SVM, ANN, and NNA, respectively, on the *Motorcycle* dataset (Silverman, 1985). In this figure, an ANN and an SVM are trained with the original training instances, while NNA refers to another ANN trained with the function values of the SVM. As shown, the regression function of NNA almost completely fits that of the SVM, and is different from that of the ANN, which indicates that NNA approximates the SVM with maintaining the similar prediction results.

The approximation process is as follows. Given a set of decision functions in a classifier $\mathbb{F} = \{f_1, \ldots, f_{|\mathbb{F}|}\}$ and the training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N_1}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector, and $y_i \in \{1, \ldots, c\}$ is its corresponding target

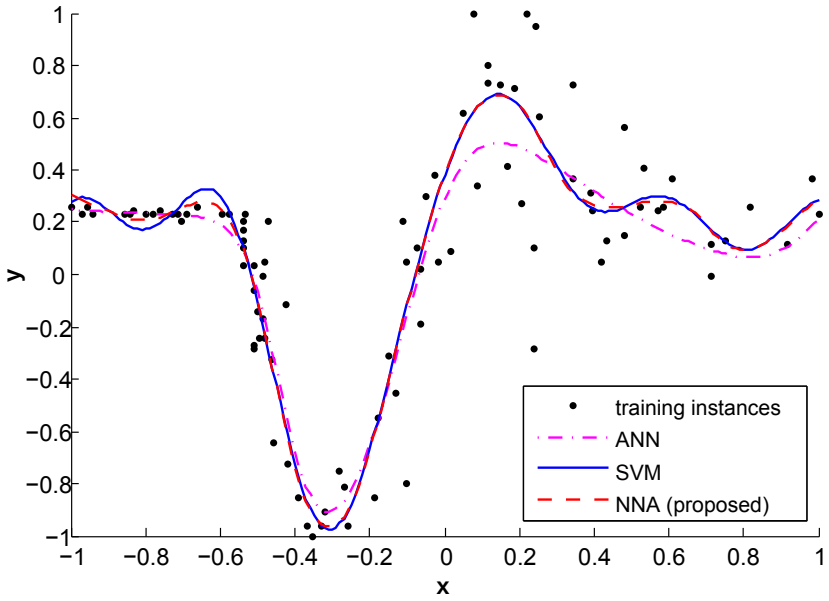


Figure 6.1 Estimated regression function of ANN, SVM and NNA on *Motorcycle* dataset

value, the function values $s_t^i \in \mathbb{R}$ of each decision function f_i, $i = 1, \ldots, |\mathbb{F}|$, are computed for all instances in $\mathcal{D}$. Then, the dataset $\mathcal{D}'$ is constructed using the input vectors and obtained function values as $\{\boldsymbol{x}_t, (s_t^1, s_t^2, \ldots, s_t^{|\mathbb{F}|})\}_{(\boldsymbol{x}_t, y_t) \in \mathcal{D}}$. Finally, an multiple-outputs ANN is trained as a function approximator to fit the function values in the dataset $\mathcal{D}'$. The resulting ANN, as an approximation of the classifier, can be used for testing new instances instead of the classifier. The pseudocode of NNA is described in Algorithm 6.

NNA is more effective for approximation of an ensemble. Although the ensemble generally consists of numerous base classifiers, classification for an instance is performed only by a few number of decision functions. Considering an

Algorithm 6 Neural Network Approximator (NNA)

Input: training dataset $\mathcal{D} = \{\boldsymbol{x}_t, y_t\}_{t=1}^{N}$, set of decision functions in classifier $\mathbb{F} = \{f_1, \ldots, f_{|\mathbb{C}|}\}$

Output: approximator $f_{\mathbb{F}}$

1: **procedure** NNA
2: **for** $i = 1$ to $|\mathbb{F}|$ **do**
3: $s_t^i \leftarrow f_i(x_t), \forall(\boldsymbol{x}_t, y_t) \in \mathcal{D}$
4: **end for**
5: $\mathcal{D}' \leftarrow \{\boldsymbol{x}_t, (s_t^1, s_t^2, \ldots, s_t^{|\mathbb{F}|})\}_{(\boldsymbol{x}_t, y_t) \in \mathcal{D}}$
6: $f_{\mathbb{F}} \leftarrow$ multiple-outputs ANN trained from $\mathcal{D}'$
7: **end procedure**

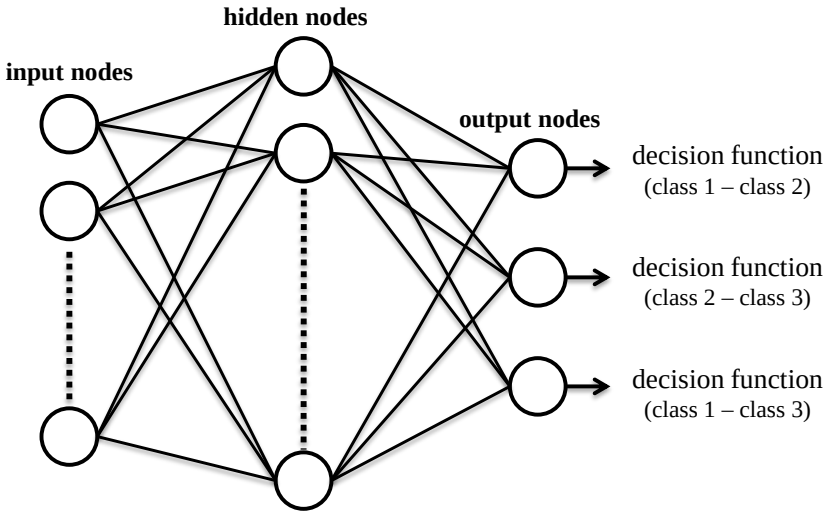


Figure 6.2 Neural network diagram of NNA for 3-class classification problem based on one-against-one approach

ensemble based on the one-against-one approach for a multi-class classification problem, only $c(c-1)/2$ decision functions should be approximated regardless of the number of base classifiers involved in the ensemble. The decision functions in the ensemble are integrated into the ANN. Figure 6.2 shows the example network diagram of NNA for a three-class classification problem based on the one-against-one approach. Each output node in the network corresponds to a decision function in the ensemble.

The proposed method can be applied to most types of ensembles such as of Chapter 3 and 4, while it is more effective for SVM ensembles. The major operations of SVM and ANN in the test phase are the kernel and sigmoidal functions computation, respectively. Considering the RBF kernel as the kernel function for SVM and logistic sigmoid function as the sigmoidal function for ANN, these two computations are similar. Equation 6.1 and 6.2 are the typical form of the RBF kernel function and the logistic sigmoid function, respectively. The decision function of SVM or the regression function of ANN is computed by linear aggregation of the RBF kernel function or the logistic sigmoid function.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right) \tag{6.1}$$

$$h(\boldsymbol{x}) = \frac{1}{1 + \exp\left(c + \mathbf{w}^T \boldsymbol{x}\right)} \tag{6.2}$$

In the test phase, SVM takes $\mathcal{O}(N_{SV})$ of its computational complexity. The support vectors are chosen among training instances in the training process.

For an SVM, a set of support vectors determines the decision boundary in the feature space. Thus, the set of support vectors tends to grow as the size of the dataset increases. In some cases, almost the whole training instances can be the support vectors. The computational complexity of ANN in the test phase is $\mathcal{O}(N_{HN})$, where N_{HN} is the number of hidden nodes. Unlike SVM, the number of hidden nodes of an ANN highly depends on the nature of the dataset. For this reason, the number of support vectors of an SVM is generally larger than that of hidden nodes of an ANN for a dataset, and moreover, more gaps appear as the dataset grows. Consequently, ANN is more advantageous than SVM in terms of the amount of computation. Furthermore, the proposed method is more advantageous for the SVM ensemble. This is because each decision function is associated with a different set of support vectors for an SVM ensemble, whereas the regression functions of the ANN are determined by a common set of hidden nodes with different weights.

6.3 Performance Evaluation on Benchmark Datasets

6.3.1 Data Description

To validate the approximation performance of the proposed method, the following ten benchmark datasets were chosen from the UCI repository (Bache & Lichman, 2014): *Iris*, *Wine*, *Sonar*, *Glass*, *Ionosphere*, *BreastCancer*, *Vechicle*, *Vowel*, *Yeast*, and *Segment*. Table 6.1 shows the descriptions for the datasets. All datasets were partitioned into 70% for the training set and 30% for the test set. Additionally, all numerical features were normalized to be in [-1, 1].

Table 6.1 Data summary

Dataset	N. instances	N. features	N. classes	N. training	N. test
Iris	150	4	3	105	45
Wine	178	13	3	125	53
Sonar	208	60	2	146	62
Glass	214	9	6	150	64
Ionosphere	351	34	2	246	105
BreastCancer	683	9	2	478	205
Vehicle	846	18	4	592	254
Vowel	990	10	11	693	297
Yeast	1484	8	10	1039	445
Segment	2310	19	7	1617	693

6.3.2 Experimental Settings

The effectiveness of the proposed method was evaluated for the SVM ensemble approximation problem. For SVM, the one-against-one approach, which has been reported to show higher accuracy than other approaches (Hsu & Lin, 2002), was adopted to extend SVM for multi-class classification problems. Thus, the decision functions of the binary SVMs for each class pair were replaced with $c(c-1)/2$ output nodes of an ANN for NNA. In addition, we employed ANN trained on training set as a baseline.

The RBF kernel function was adopted as a kernel function for SVM. For ANN, the Levenberg-Marquardt back propagation algorithm, which performs better for regression problems (Hagan & Menhaj, 1994), was employed. The number of hidden layers was set to 1. In addition, the logistic sigmoid function

was used as the sigmoidal function. For all algorithms, the best parameters were found using ten-fold cross validation on the training dataset with a grid search mechanism. The best parameters for the SVM were explored on a two dimensional grid search with $C = \{2^{-3}, 2^{-2}, \ldots, 2^{10}\}$ and $\sigma = \{2^{-5}, 2^{-4}, \ldots, 2^{5}\}$. All experiments were performed in MATLAB.

The classification performance was evaluated using the misclassification error rate (%) which is defined by $(1/N)\sum_{t=1}^{N} 1_{y_t \neq \hat{y}_t} \times 100$, where N is the number of test instances, y_t is actual class of the t-th instance, $\hat{y}_t$ is predicted class of the t-th instance, and $1_{y_t \neq \hat{y}_t}$ is an indicator function that has value 1 when $y_t \neq \hat{y}_t$. In addition, test time and the number of support vectors or hidden nodes were measured to validate the effectiveness of the proposed method.

6.3.3 Experimental Results

Table 6.2 presents the classification results that compare the proposed NNA with a conventional SVM. NNA, the proposed method, has a similar error rate to the SVM, but shows significantly lower test time on every dataset. ANN, the baseline, has worse prediction accuracy than NNA on most datasets. On average, the proposed method achieved approximately 89.6% reduction in test time. Moreover, the test time was further reduced for large-scale datasets. For example for the *Yeast* dataset, the number of support vectors for SVM was 752 while there were only 12 hidden nodes for NNA. Therefore, the test time was reduced by 96.5%.

Figure 6.3 plots the number of support vectors and hidden nodes against dataset size for each dataset. As shown in this figure, as the dataset size grows,

Table 6.2 Approximation results on benchmark datasets

Dataset	N. classes	ANN		SVM Ensemble			NNA			Comparison	
		Error rate (%)	N. hidden nodes	Error rate (%)	Test time (ms)	N. support vectors	Error rate (%)	Test time (ms)	N. hidden nodes	Error rate (%) difference	Test time reduction rate (%)
Iris	3	4.444	24	4.444	0.178	55	4.444	0.027	13	0.000	85.039
Wine	3	1.887	4	3.774	0.273	77	3.774	0.015	4	0.000	94.557
Sonar	2	17.742	8	11.290	0.500	105	12.903	0.040	11	1.613	92.098
Glass	6	39.063	29	35.938	0.386	106	35.938	0.037	12	0.000	90.425
Ionosphere	2	9.524	13	7.619	0.339	54	7.619	0.050	10	0.000	85.327
BreastCancer	2	1.471	13	2.451	0.376	52	2.451	0.071	10	0.000	81.100
Vehicle	4	18.972	13	16.996	1.492	263	16.601	0.179	19	0.395	88.021
Vowel	11	14.478	15	1.347	2.633	422	1.684	0.250	38	0.337	90.513
Yeast	10	43.146	11	41.124	7.881	752	41.573	0.276	12	0.449	96.498
Segment	7	5.916	20	2.742	4.383	287	2.886	0.323	21	0.144	92.629
Average		-	-	-	-	-	-	-	-	0.294	89.621

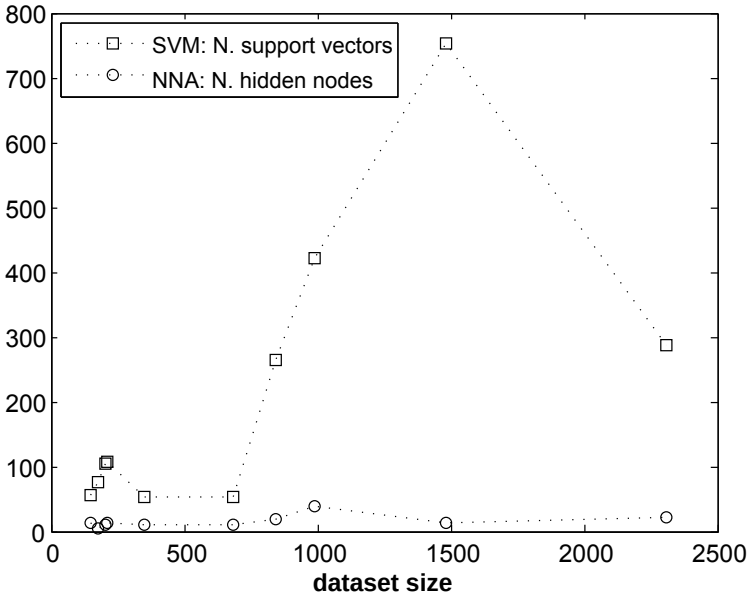


Figure 6.3 Comparison between the number of support vectors for SVM and the number of hidden nodes for NNA

the number of support vectors of SVM greatly increases while the number of hidden nodes of NNA does not.

6.4 Application to Semiconductor Die Failure Prediction

6.4.1 Problem Definition

In recent years, semiconductor manufacturing has become more lengthened and complex owing to technological advances. Accordingly, a large amount of data is being generated in real-time during each step of the manufacturing process. This

data has received considerable attention from researchers for yield management and enhancement (N. Kumar et al., 2006; Shin & Park, 2000; Chien et al., 2007)

The semiconductor manufacturing process can be divided into four basic steps: wafer fabrication, wafer test, assembly, and final test (Uzsoy et al., 1992). The first step is wafer fabrication, which involves building different layers on a wafer with a number of operations in order to produce the required circuitry for the dies of the wafer. After wafer fabrication, wafer test is conducted to analyze and evaluate the electrical properties of the dies in a wafer. Defective dies are filtered out based on the results of this step. Only dies that are not defective proceed to the assembly step in order to obtain packaged chips as final products. The chips are graded as success or failure through a functional test called final test. The final test involves hot test and cold test to evaluate the functionality of a chip in hot and cold environments, respectively.

During wafer test, defective dies with a high probability of failing the final test are filtered out, and only repairable dies proceed to the subsequent steps (Y. Park et al., 2015). However, some faulty dies pass the wafer test, which then ultimately fail the final test (An et al., 2009; S. H. Park et al., 2013). There are a variety of possible reasons to explain this. The wafer test is performed with a finite number of test items due to constraints in time and cost, thus the quality of the dies are underdetermined. Another possibility is that progressive failures may occur after the wafer test. In any case, this problem degrades the final yield and hinders the manufacturing process.

A machine learning approach is worth considering in order to exploit the wafer test data to predict the die-level results of the final test before assembly.

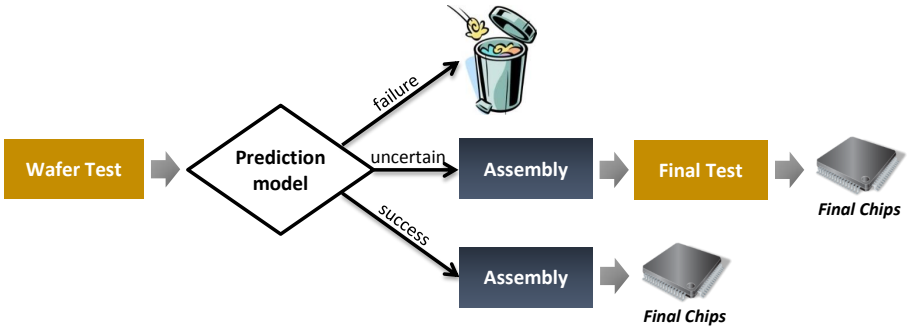


Figure 6.4 Application of a prediction model to final yield management

This approach is data-driven that does not require domain knowledge or assumptions. The final yield could be managed as suggested in Figure 6.4 when it is possible to predict the results of the final test accurately. The final test could be omitted for dies that are certain to pass, whereas no further steps would be required for those certain to fail. Only uncertain dies would proceed to the subsequent steps. Thus, the final test, which leads to time and cost, is unnecessary for dies that are predicted either to pass or fail. Additionally, the final yield can be enhanced by investigating the causes of those predicted to fail the final test, and rejecting them in advance at the wafer test.

Kang, Cho, An, and Rim (2015) proposed to use a RF (Breiman, 2001), which involves training of several randomized decision trees, for this prediction problem. The RF is suitable for our problem because of the following reasons. It achieves accurate and reliable prediction by aggregating the predictions of individual trees, is able to deal with the non-linear relationship between input and output variables, is robust to outliers owing to the property of decision trees, and runs effectively on large scale datasets. Two prediction models were

built separately for each type of failure using the RF, because it is known that the characteristics of hot and cold failures are different. Each prediction model provides scores regarding whether a new die will fail, ranging from 0 to 1, for a failure type. They found the effectiveness of the RF to predict the die-level result in the final test via experiments on a real-world dataset.

An important consideration here is that the prediction must be performed instantly, while the computing resources are limited in the manufacturing process. However, the RF typically has low test speed because of its complexity in the test phase. Thus, in this section, NNA is applied to raise the prediction speed by approximating the RF. Experiments are conducted on a real-world dataset to confirm the effectiveness of NNA for this problem.

6.4.2 Data Description

The data were collected during a week in 2014 from a semiconductor manufacturer in the Republic of Korea. Only repairable dies were used in the experiments. The data originally contained 49 input variables, each of which corresponds to a test item in the wafer test, and a single output variable for the final test result. The final test result belongs to one of the following three categories: success, hot failure (the dies that fail the hot test), and cold failure (the dies that pass the hot test but fail the cold test). The fraction of failure in the data was under 1%, meaning that there is a severe class imbalance.

In addition, five variables were derived to be used as input variables: the distance of the die from the wafer center, previous final yield at the die position (hot and cold failure), wafer test success rate of the adjacent dies, and

abnormalities of the wafer map pattern. We also tried to utilize interaction variables of the proposed variables, but there was no significant improvement in prediction performance. Thus, we did not use these interaction variables for the sake of simplicity in description. The five derived variables were used to predict whether a die pass or fail the final test more accurately, by employing them as input variables for the prediction models. Thus, the prediction models were trained with 54 input variables.

6.4.3 Experimental Settings

Regarding training of the original prediction models, the bootstrap sample size and the minimum number of instances in a leaf node were set as 80% and 1% of the training set, respectively. Other settings were left as default settings of the `TreeBagger` function in MATLAB. The two prediction models, each corresponds to a failure type, were approximated by a single ANN model using NNA. Thus, the ANN model has two output nodes. The number of hidden nodes for the ANN model was set to 10.

Regarding model validation, the sliding window test procedure was conducted. The basic idea of this procedure is that the present is tested by reference to the near past, and the present predicts the near future. This is suitable when the data characteristics change with time, such as in semiconductor manufacturing. In this experiments, the data were partitioned by day. Thus the model that was trained at day t is used to predict the data for day $t+1$, $t = 0, 1, \ldots, 6$. For each day's training set, under-sampled data, including $100,000$ instances of success dies and $1,000$ instances of failure dies for each failure type, were used.

The entire data for a day constitutes the test set.

Classification accuracy, a most popular criterion, is not appropriate for this highly class-imbalanced data because just a simple guessing of predicting all instances as the majority class provides the accuracy above 99%. Thus, the Area under Receiver Operating Characteristics (AuROC) was employed as the criterion for the performance. The AuROC is defined as the area under the ROC curve, which plots the true positive rate against the false positive rate with varying the decision threshold of pass/fail. Thus, the AuROC measures the overall prediction performance in a threshold independent manner. This values were computed for each failure category against success category, as $\mathrm{AuROC}_{\mathrm{HOT}}$ and $\mathrm{AuROC}_{\mathrm{COLD}}$. Note that, a single threshold has to be used in practice, and this threshold can be chosen according to the purpose of the prediction modeling.

6.4.4 Experimental Results

Table 6.3 shows the approximation results on the semiconductor die failure prediction problem. The prediction results of the RF for a day was obtained using the two prediction models, and that of NNA were obtained using a single approximator for the two prediction models. The prediction performance changed with time, where the pattern of the change according to time differed depending on the type of failure. For example, the AuROC of hot failures was the best on the fifth day, whereas the performance of cold failures was the worst on that day.

Overall, the test time was reduced by 99.118%, while the AuROC difference

Table 6.3 Approximation results on semiconductor die failure prediction problem

Day (test)	Random Forest			NNA			Comparison		
	$\text{AuROC}_{\text{HOT}}$	$\text{AuROC}_{\text{COLD}}$	Test time (s)	$\text{AuROC}_{\text{HOT}}$	$\text{AuROC}_{\text{COLD}}$	Test time (s)	$\text{AuROC}_{\text{HOT}}$ difference	$\text{AuROC}_{\text{COLD}}$ difference	Test time reduction rate (%)
1	71.918	68.960	231.620	71.931	68.753	1.948	0.013	0.206	99.159
2	75.599	71.158	120.564	75.276	70.774	1.068	0.323	0.384	99.114
3	74.883	72.046	109.882	74.862	71.726	0.981	0.021	0.319	99.108
4	75.555	69.240	124.056	75.605	69.053	1.122	0.051	0.187	99.096
5	76.449	66.006	130.454	76.623	65.570	1.148	0.173	0.436	99.120
6	71.435	69.721	102.865	71.850	69.690	0.917	0.415	0.032	99.109
Average	-	-	-	-	-	-	0.166	0.261	99.118

for the hot and cold failures were 0.166 and 0.261, on average. This indicates that NNA significantly improves the prediction speed without compromising the prediction accuracy. Thus, the use of NNA provides the desired prediction speed for practical deployment of the prediction models. However, the prediction performance should be improved before deploying the prediction models in practice.

When a prediction model is deployed, the prediction performance of the model may decrease as time goes on due to the gradual change of the data characteristics including the importance of each input variable. Thus, the prediction performance of the model should be monitored and the model has to be updated periodically to reflect the recent change of the data characteristics.

6.5 Summary

The major drawback of a multi-class classifier is its low speed in the test phase, while relatively few studies have been focused on test phase of the classifier in spite of the importance of run-time speed. In this chapter, NNA, an ANN based heterogeneous ensemble method to achieve faster test speed of the classifier was proposed. The proposed method approximates the classifier using a multi-outputs ANN where each node corresponds to the decision function in the classifier. The distinguishing point of the proposed method over existing work is that the approximation problem of the classifier is converted into the regression-based function approximation problem. This can be explained as that the classifier generates noise-filtered functions and the ANN just approximates

these functions. Moreover, it is not strongly restricted by the structure of the classifier. Instead, the structure of the ANN should be considered. Thus, if higher run-time speed is required and some losses of classification accuracy are allowable, we can manually employ the smaller number of hidden nodes rather than choose the best number of hidden nodes through cross validation.

The proposed method was found to be effectively applied to an SVM ensemble for multi-class classification, owing to the characteristics of SVM. The regression functions of the ANN approximated the decision functions of binary SVMs in the ensemble. Therefore, the application of the proposed method can improve the practical usability of SVM, especially in real-time applications. Experimental results on several benchmark datasets showed that the proposed method achieved significant test time reductions compared to conventional SVM without compromising prediction accuracy. Moreover, the proposed method was much more effective for large-scale datasets. We explained this consequences by the characteristics that the computation of the RBF kernel function for SVM and the computation of the sigmoidal function for ANN are similar but the number of hidden nodes of an ANN is generally smaller than the number of support vectors of an SVM, for any dataset.

We examined the effectiveness of NNA to the semiconductor die failure prediction problem using actual data from a semiconductor manufacturer. In this application, the RF models were approximated to obtain faster prediction speed. As a result, the test time of the prediction was reduced by 99% while maintaining the prediction performance. This application is promising in that accurate prediction of die fails in the final test enables the yield to be managed

more effectively. The final test can be skipped for dies that will almost certainly pass or fail, which leads to significant saving in time and cost. In addition, we can conduct further investigation and take preemptive actions for those dies that are predicted as fails, such as filtering out them at the wafer test step or grading the quality of dies in advance. However, the prediction performance should be improved before deploying the prediction models in practice. For example, we can omit the final test for dies from the bottom 50% scores with an acceptable tolerance when over 99% of the fail dies are included in the top 50% scores.

The proposed method can also be applied to various types of ensembles. For meta-learning based ensembles such as ODOAO and HEOC, approximating only a meta-classifier may be sufficient to achieve comparable classification accuracy. In addition, the training speed can be enhanced by using any kinds of regression algorithms that requires smaller test time than ANN. Thus, the proposed method merits further investigation.

Chapter 7

Conculsion

7.1 Contributions

Multi-class classification is a supervised learning task that is closely related to various real-world applications. When deploying a multi-class classifier for a multi-class classification problem, three issues are usually considered depending on the characteristics of the problem. These issues are accurate, fast, and reliable classification, which determine the applicability of the classifier in practice. This dissertation developed methods based on heterogeneous ensemble learning to address the respective issues. These methods construct heterogeneous ensembles utilizing multiple classifiers that are trained using various classification algorithms, where each classifier plays a different role to accomplish the desired functionality.

For accurate classification, we proposed DOAO and ODOAO that construct heterogeneous ensembles of binary classifiers. These methods are based on the one-against-one approach, where base classifiers are trained for class pairs. DOAO selects the best classification algorithm for each class pair as having the minimum validation error, thereby yielded better classification accuracy

than other one-against-one classifiers that are based on individual classification algorithms. Besides, the effectiveness of DOAO is limited because the minimum validation error does not always indicate the minimum test error, especially when comparing heterogeneous classifiers, and proper fuse of the base classifier can outperform the single best classifier. Thus, we proposed ODOAO to address such limitations. ODOAO constructs a heterogeneous ensemble where a meta-classifier combines the outputs of the base classifiers for class pairs considering non-linear relationship and high dimensionality. Through conducting statistical test, we found that these heterogeneous methods achieved more accurate classification than homogeneous ones with statistical significance for benchmark datasets.

In addition, we proposed HEOC utilizing diverse one-class classifiers. In this method, the use of various one-class classification algorithms contributes towards increasing the diversity of the ensemble, while stacking resolves the normalization issues on different scales of outputs obtained from the base classifiers. We also demonstrated the selective utilization of base classifiers by adopting a stepwise variable selection procedure during stacking. The effectiveness of this method was confirmed through experiments on benchmark datasets and text categorization problems.

For fast classification, we proposed NNA to approximate a complex multi-class classifier to reduce computational time in the test time. Since ANN, on its own, has better fitting performance for a function than for a noisy dataset, NNA utilizes ANN as a function approximator of the complex classifier. In detail, a multiple-outputs ANN is employed to approximate the classifier, in which each

output node correspond to a decision function in the classifier, thereby achieve faster run-time speed without compromising prediction accuracy. NNA can be applied effectively to an ensemble consisting of a number of base classifiers, because the ensemble only has a few decision functions. Moreover, NNA performs very well as an SVM approximator because of their structural characteristics. This method was found to be effective for benchmark datasets and semiconductor die failure prediction problem.

For reliable classification, a hybrid reject option was proposed for better trade-off between accuracy and rejection for multi-class classification. This method is useful when high prediction accuracy of a classifier is essential due to high misclassification costs. By applying a reject option, it is much better for human experts to examine those for whom results are difficult to predict than for the classifier to predict fully. The hybrid reject option utilizes the two components, filter and predictor. Each component is trained separately using the best respective algorithm to maximize the capability of its role. We confirmed the effectiveness of applying this method to anti-diabetic drug failure prediction through experiments on real-world EMR data of type 2 diabetes, by showing this method provided better prediction accuracy for the same degree of rejection.

In conclusion, heterogeneous ensemble methods for each of accurate, fast, and reliable multi-class classification were covered in this dissertation. Experimental results showed that heterogeneous ensemble learning is a better way to ensure desired performance for multi-class classification problems in practice. We also described three real-world applications that the proposed meth-

ods are helpful: text categorization, semiconductor die failure prediction, and anti-diabetic drug failure prediction.

Beside the effectiveness, heterogeneous ensemble learning also has its drawback on training time because a heterogeneous ensemble involves training of numerous base classifiers. Thus, more training time is required for the heterogeneous ensemble. When the training phase is subjected to a time limit, a homogeneous ensemble with individual superior algorithms, such as SVM, is more preferable, while this compromises some prediction loss. However, we usually have no certain knowledge that which algorithm performs best for a dataset. In addition, training is carried out before applying the classifier in most cases, therefore often is not subjected to a time limit. Under this circumstance, heterogeneous ensemble learning is a better way to obtain desired performance.

7.2 Future Work

There are some limitations in this dissertation that should be addressed in future work. First, the proposed methods exploit a broader hypothesis space to achieve their goals, while an increase of complexity should be resolved. It is known that the high complexity negatively affects to the classification performance. Second, theoretical foundations of how each method works should be studied. The effectiveness of the proposed methods was mainly confirmed through experiments and statistical tests in this dissertation, while theoretical analysis will support understanding this effectiveness. This includes analysis

of the mechanism, bias-variance decomposition, and lower/upper bound of the performance compared to homogeneous ensemble methods. Third, the effectiveness should also be confirmed for unusual environments, such as very large scale data and online learning with real-time data stream. The importance of these environments is being emphasized as the era of big data arises.

Bibliography

American Diabetes Association. (2014). Standards of medical care in diabetes–2014. *Diabetes Care*, *37*(Supplement 1), S14-S80.

An, D., Ko, H.-H., Gulambar, T., Kim, J., Baek, J.-G., & Kim, S.-S. (2009). A semiconductor yields prediction using stepwise support vector machine. In *Proceedings of the 2009 IEEE International Symposium on Assembly and Manufacturing* (pp. 130–136).

Bache, K., & Lichman, M. (2014). *UCI Machine Learning Repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Ban, T., & Abe, S. (2006). Implementing multi-class classifiers by one-class classification methods. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 327–332).

Bennett, C. M., Guo, M., & Dharmage, S. C. (2007). HbA1c as a screening tool for detection of type 2 diabetes: A systematic review. *Diabetic Medicine*, *24*(4), 333–343.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32.

Burges, C. J. C. (1996). Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning* (pp. 71–77).

Cai, D., Wang, X., & He, X. (2009). Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th International Conference on Machine Learning* (pp. 105–112).

Cavalin, P. R., Sabourin, R., & Suen, C. Y. (2013). Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, *22*(3-4), 673-688.

Cawley, G. C., & Talbot, N. L. C. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, *11*, 2079-2107.

Cecotti, H., & Vajda, S. (2013). Rejection schemes in multi-class classification – Application to handwritten character recognition. In *Proceedings of the 12th International Conference on Document Analysis and Recognition* (pp. 445–449).

Chen, J.-H., & Chen, C.-S. (2004). Reducing SVM classification time using multiple mirror classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part B – Cybernetics*, *34*(2), 1173-1183.

Chien, C.-F., Wang, W.-C., & Cheng, J.-C. (2007). Data mining for yield enhancement in semiconductor manufacturing and an empirical study. *Expert Systems with Applications*, *33*(1), 192-198.

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, *16*(1), 41-46.

Cyganek, B. (2012). One-class support vector ensembles for image segmentation and classification. *Journal of Mathematical Imaging and Vision*, *42*(2-3), 103-117.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1-30.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems, Lecture Notes in Computer Science* (Vol. 1857, pp. 1–15). Springer.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Downs, T., Gates, K. E., & Masters, A. (2002). Exact simplification of support vector solutions. *Journal of Machine Learning Research*, *2*, 293–297.

Duan, K.-B., & Keerthi, S. S. (2005). Which is the best multiclass SVM method? an empirical study. In *Multiple Classifier Systems, Lecture Notes in Computer Science* (Vol. 3541, pp. 278–285). Springer.

Džeroski, S., & Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, *54*(3), 255-273.

Fumera, G., & Roli, F. (2002). Support vector machines with embedded reject option. In *Pattern Recognition with Support Vector Machines, Lecture Notes in Computer Science* (Vol. 2388, pp. 68–82). Springer.

Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research*, *2*, 721-747.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761-1776.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2013). Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. *Pattern Recognition*, *46*(12), 3412-3424.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, *13*(10), 959-977.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*(10), 2044-2064.

García, S., & Herrera, F. (2008). An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677-2694.

Giacinto, G., & Roli, F. (2001). An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, *22*(1), 25-33.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications*, *13*(4), 215-223.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993.

Hao, P.-Y., Chiang, J.-H., & Lin, Y.-H. (2009). A new maximal-margin spherical-structured multi-class support vector machine. *Applied Intelligence*, *30*(2), 98-111.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data mining, Inference, and Prediction.* Springer.

Herbei, R., & Wegkamp, M. H. (2006). Classification with reject option. *Canadian Journal of Statistics*, *34*(4), 709–721.

Ho, T. K., Hull, J. J., & Srihari, S. N. (1994). Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(1), 66-75.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, *13*(2), 415-425.

Huang, Y., McCullagh, P., Black, N., & Harper, R. (2007). Feature selection and classification model construction on type 2 diabetic patients' data. *Artificial Intelligence in Medicine*, *41*(3), 251-262.

Ji, J., & Zhao, Q. (2013). A hybrid SVM based on nearest neighbor rule. *International Journal of Wavelets, Multiresolution and Information Processing*, *11*(6), 1350048.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*. Kluwer Academic Publishers.

Juszczak, P., & Duin, R. P. W. (2004). Combining one-class classifiers to classify missing data. In *Multiple Classifier Systems, Lecture Notes in Computer Science* (Vol. 3077, pp. 92–101). Springer.

Kang, S., & Cho, S. (2014). Approximating support vector machine with artificial neural network for fast prediction. *Expert Systems with Applications*, *41*(10), 4989-4995.

Kang, S., & Cho, S. (2015a). A novel multi-class classification algorithm based on one-class support vector machine. *Intelligent Data Analysis*, *19*(4), in press.

Kang, S., & Cho, S. (2015b). Optimal construction of one-against-one classifier

based on meta-learning. *Neurocomputing*, in press.

Kang, S., Cho, S., An, D., & Rim, J. (2015). Using wafer map features to better predict die-level failures in final test. *IEEE Transactions on Semiconductor Manufacturing*, in press.

Kang, S., Cho, S., & Kang, P. (2015a). Constructing a multi-class classifier using one-against-one approach with different binary classifiers. *Neurocomputing*, *149*, 677-682.

Kang, S., Cho, S., & Kang, P. (2015b). Multi-class classification via heterogeneous ensemble of one-class classifiers. *Engineering Applications of Artificial Intelligence*, *43*, 35-43.

Kang, S., Cho, S., Rhee, S.-j., & Yu, K.-S. (2015). Reliable prediction of anti-diabetic drug failure with a reject option. *IEEE Journal of Biomedical and Health Informatics*, submitted.

Kang, S., Kang, P., Ko, T., Cho, S., Rhee, S.-j., & Yu, K.-S. (2015). An efficient and effective ensemble of support vector machines for anti-diabetic drug failure prediction. *Expert Systems with Applications*, *42*(9), 4265-4273.

Kiang, M. Y. (2003). A comparative assessment of classification methods. *Decision Support Systems*, *35*(4), 441-454.

Kim, H.-C., Pang, S., Je, H.-M., Kim, D., & Bang, S. Y. (2003). Constructing support vector machine ensemble. *Pattern Recognition*, *36*(12), 2757-2767.

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

Knerr, S., Personnaz, L., & Dreyfus, G. (1990). Single-layer learning revisited: A stepwise procedure for building and training a neural network. In *Neurocomputing: Algorithms, Architectures and Applications, NATO ASI Series* (Vol. 68, pp. 41–50). Springer.

Krawczyk, B., Woźniak, M., & Cyganek, B. (2014). Clustering-based ensembles for one-class classification. *Information Sciences*, *264*, 182-195.

Kumar, M. A., & Gopal, M. (2010). A hybrid SVM based decision tree. *Pattern Recognition*, *43*(12), 3977–3987.

Kumar, N., Kennedy, K., Gildersleeve, K., Abelson, R., Mastrangelo, C. M., & Montgomery, D. C. (2006). A review of yield modelling techniques for semiconductor manufacturing. *International Journal of Production Research*, *44*(23), 5019-5036.

Kuncheva, L. I. (2002). Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on Systems, Man, and Cybernetics Part B – Cybernetics*, *32*(2), 146-156.

Kuncheva, L. I., Whitaker, C. J., Shipp, C. A., & Duin, R. P. W. (2003). Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis Applications*, *6*(1), 22-31.

Landgrebe, T. C. W., Tax, D. M. J., Paclík, P., & Duin, R. P. W. (2006). The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, *27*(8), 908-917.

Lee, D., & Lee, J. (2007). Domain described support vector classifier for multi-classification problems. *Pattern Recognition*, *40*(1), 41-51.

Lee, H.-j., & Cho, S. (2006). The novelty detection approach for different degrees of class imbalance. In *Neural Information Processing, Lecture Notes in Computer Science* (Vol. 4233, pp. 21–30). Springer.

Lézoray, O., & Cardot, H. (2008). Comparing combination rules of pairwise neural networks classifiers. *Neural Processing Letters*, *27*(1), 43-56.

Li, Q., Jiao, L., & Hao, Y. (2007). Adaptive simplification of solution for support vector machine. *Pattern Recognition*, *40*(3), 972–980.

Li, Y., Zhang, W., & Lin, C. (2006). Simplify support vector machines by iterative learning. *Neural Information Processing: Letters and Reviews*, *10*, 11–17.

Liang, X., Ma, Y., He, Y., Yu, L., Chen, R.-C., Liu, T., ... Chen, T.-S. (2013). Fast pruning superfluous support vectors in SVMs. *Pattern Recognition Letters*, *34*(10), 1203 - 1209.

Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new clas-

sification algorithms. *Machine Learning*, *40*(3), 203-228.

Lin, H.-J., & Yeh, J. P. (2009). Optimal reduction of solutions for support vector machines. *Applied Mathematics and Computation*, *214*(2), 329–335.

Lin, H.-T., Lin, C.-J., & Weng, R. C. (2007). A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, *68*(3), 267-276.

Lorena, A. C., de Carvalho, A. C. P. L. F., & Gama, J. M. P. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, *30*(1-4), 19-37.

Lu, Z. X., Walker, K. Z., O'Dea, K., Sikaris, K. A., & Shaw, J. E. (2010). A1C for screening and diagnosis of type 2 diabetes in routine clinical practice. *Diabetes Care*, *33*(4), 817–819.

Maji, S., Berg, A. C., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Maji, S., Berg, A. C., & Malik, J. (2013). Efficient classification for additive kernel SVMs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 66-77.

Menahem, E., Rokach, L., & Elovici, Y. (2009). Troika - An improved stacking schema for classification tasks. *Information Sciences*, *179*(24), 4097-4122.

Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Ma-*

chine Learning, *36*(1-2), 33-58.

Nadeem, M. S. A., Zucker, J.-D., & Hanczar, B. (2010). Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology, Journal of Machine Learning Research Workshop and Conference Proceedings* (Vol. 8, pp. 65–81).

Park, S. H., Park, C.-S., Kim, J. S., Kim, S.-S., Baek, J.-G., & An, D. (2013). Data mining approaches for packaging yield prediction in the post-fabrication process. In *Proceedings of the 2013 IEEE International Congress on Big Data* (pp. 363–368).

Park, Y., Kang, S., & Cho, S. (2015). Memory die clustering and matching for optimal voltage window in semiconductor. *IEEE Transactions on Semiconductor Manufacturing*, *28*(2), 180-187.

Polat, K., & Güneş, S. (2009). A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications*, *36*(2), 1587-1592.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1-2), 1-39.

Savicky, P., & Fürnkranz, J. (2003). Combining pairwise classifiers with stacking. In *Advances in Intelligent Data Analysis, Lecture Notes in Computer Science* (Vol. 2810, pp. 219–229). Springer.

Schmitz, G. P. J., Aldrich, C., & Gouws, F. S. (1999). ANN-DT: An algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*, *10*(6), 1392–1401.

Seewald, A. K. (2002). How to make stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the 19th International Conference on Machine Learning* (pp. 554–561).

Sharma, S., Bellinger, C., & Japkowicz, N. (2012). Clustering based one-class classification for compliance verification of the comprehensive nuclear-test-ban treaty. In *Advances in Artificial Intelligence, Lecture Notes in Computer Science* (Vol. 7310, pp. 181–193). Springer.

Shin, C. K., & Park, S. C. (2000). A machine learning approach to yield management in semiconductor manufacturing. *International Journal of Production Research*, *38*(17), 4261-4271.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–52.

Simeone, P., Marrocco, C., & Tortorella, F. (2012). Design of reject rules for ECOC classification systems. *Pattern Recognition*, *45*(2), 863-875.

Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(11), 1137-1144.

Tax, D. M. J. (2001). *One-class Classification: Concept-learning in the Absence of Counter-examples* (Unpublished doctoral dissertation). Delft University of Technology.

Tax, D. M. J. (2014). *DDtools, the Data Description Toolbox for Matlab.* Retrieved from `http://prlab.tudelft.nl/david-tax/dd_tools.html`

Tax, D. M. J., & Duin, R. P. W. (2008). Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, *29*(10), 1565-1570.

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

Todorovski, L., & Džeroski, S. (2003). Combining classifiers with meta decision trees. *Machine Learning*, *50*(3), 223-249.

Tortorella, F. (2005). A ROC-based reject rule for dichotomizers. *Pattern Recognition Letters*, *26*(2), 167-180.

Tsoumakas, G., Angelis, L., & Vlahavas, I. (2005). Selective fusion of heterogeneous classifiers. *Intelligent Data Analysis*, *9*(6), 511-525.

Uzsoy, R., Lee, C.-Y., & Martin-Vega, L. A. (1992). A review of production planning and scheduling models in the semiconductor industry part I: System characteristics, performance evaluation and production planning. *IIE Transactions*, *24*(4), 47-60.

Vedaldi, A., & Zisserman, A. (2012). Efficient additive kernels via explicit

feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(3), 480-492.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77-95.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241-259.

Wolpert, D. H. (2001). The supervised learning no-free-lunch theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications* (pp. 25–42).

Woods, K., Kegelmeyer Jr., W. P., & Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 405-410.

Woźniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3-17.

Wozniak, M., & Jackowski, K. (2009). Some remarks on chosen methods of classifier fusion based on weighted voting. In *Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science* (Vol. 5572, pp. 541–548). Springer.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., ... Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1-37.

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(3), 418-435.

Xu, X., Zhang, B., & Zhong, Q. (2005). Text categorization using SVMs with rocchio ensemble for internet information classification. In *Proceedings of the 3rd International Conference on Networking and Mobile Computing* (pp. 1022–1031).

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 412–420).

국문초록

데이터마이닝에서 분류는 범주 형태의 출력변수를 예측하기 위한 교사학습 방법론이다. 분류문제의 범주가 여러 가지인 경우 우리는 이 문제를 다중분류 문제라고 한다. 다중분류는 이진분류에 비해서 더 많은 정보를 제공하여 활용도가 높으며, 실제 현실문제들과도 더욱 밀접하게 연관되어 있다. 다중분류의 성능은 일반적으로 정확성, 안정성, 그리고 신속성 세가지 측면으로 확인할 수 있다. 이 세가지 측면에 대해서 더 좋은 성능을 얻기 위해서, 본 논문에서는 이질적 앙상블 학습을 통한 방법론들을 제안한다. 이질적 앙상블은 다양한 분류 알고리즘으로부터 얻어진 이질적인 기반 분류기들을 필요한 기능을 갖도록 조합하여 다중분류 문제에 대해서 목표하는 분류 성능을 얻을 수 있다. 분류의 정확성을 위한 이질적 앙상블 방법으로는 Diversified One-Against-One (DOAO)과 Optimally Diversified One-Against-One (DOAO)를 제안하였다. 이들의 기본 원리는 원 다중분류 문제를 일대일 원리에 따라서 여러 개의 이진분류 부분문제로 분해하는 것이다. DOAO는 각 범주 짝에 대해서 최적의 분류 알고리즘을 찾음으로써 여러 알고리즘들을 상호 보완적으로 사용한다. 범주 짝 별로 최적의 알고리즘이 다르다는 점에서, DOAO는 다중분류 문제에 대해서 더 정확한 분류를 할 수 있다. ODOAO는 DOAO를 보완한 방법으로, 메타 분류기를 이용하여 여러 이질적인 기반 분류기들로부터 얻어진 결과들을 효과적으로 조합하여 최종 분류결과를 제공한다. 또한, 분류의 정확성을 위한 Heterogeneous Ensemble of One-class Classifiers (HEOC)도 제안하였다. HEOC는 여러 단일분류 알고리즘들로부터 얻어진 이질적인 기반 분류기들을 이용하여 이질적 앙상블을 구축한다. 이 과정에서 스태킹을 활용하여 여러 이질적 분류기들을 정규화하여 분류기들 간 예측점수 범위의 차이를 조정하여 높은 분류

정확도를 얻는다. 분류의 안정성을 위해서는, 분류가 모호한 경우에 대한 분류를 거절하기 위한 이질적 앙상블 기반 혼합 분류거절 방법을 제안하였다. 이 방법은 새로운 데이터에 대해 분류를 할지를 결정하는 필터와, 분류를 하기로 결정된 데이터에 대해 실제 분류를 수행하는 예측기를 별도로 학습한다. 이 과정에 각각의 성능을 극대화하기 위해 각각에 대해 최적의 분류알고리즘을 적용하여 같은 거절 수준에 대해서 더 나은 정확도를 얻는다. 분류의 신속성을 위해서, 우리는 Neural Network Approximator (NNA)을 제안하였다. NNA는 분류기가 분류를 수행하는 데 걸리는 계산 시간을 줄이기 위해서 다중출력 인공신경망을 분류기 내 결정함수들에 대한 근사기로 활용한다. 이 인공신경망에서 하나의 출력마디는 하나의 결정함수와 연계되며, 따라서 우리는 분류기 대신 근사기만을 이용하여 분류를 수행하여 분류 정확도의 손실을 피하면서 기존 분류기에 비해 더 빠른 분류를 수행한다. 본 논문에서는 다양한 검증용 데이터와 현실 문제에 대한 실험을 통해서 제안하는 이질적 앙상블 기반 방법들의 다중분류에 대한 효과를 확인하였다.

주요어: 데이터마이닝, 기계학습, 앙상블, 이질적 앙상블, 다중분류

학번: 2011-21163