SELECTING FROM AN ENSEMBLE OF EXPERTS FOR MACHINE LEARNING

by

Esma Kılıç

BS, in Computer Engineering, Marmara University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor Prof. Ethem Alpaydın for his endless academic support and encouragement, patience and guidance throughout my graduate study and the completion of this research.

I want to thank Prof. Fikret Gürgen and Assoc. Prof. Haluk Topçuoğlu for participating in my thesis committee and giving me feedback.

I also thank Asst. Prof. Olcay Taner Yıldız and Mehmet Aydın Ulaş for providing the datasets and the base classifiers used in this thesis.

Without love and support of my friends and family it would be so difficult to complete this thesis. Especially, I thank to my friends, Emel Coşkun and Melike Horasanlı for their invaluable friendship and also Demet Ayvaz and Reyhan Aydoğan for their motivation and enthusiasm during the thesis.

This thesis, like all other studies during my education, would not be completed without endless encouragement and affection of my family. Especially I thank my sister Hülya Kılıç for her endless encouragement in every stages of my life as being my teacher.

ABSTRACT

SELECTING FROM AN ENSEMBLE OF EXPERTS FOR MACHINE LEARNING

Decision combination has recently become popular to improve over single learner systems. The fundamental idea behind an ensemble of classifiers is that the patterns which are misclassified by different classifiers are not necessarily the same and that by suitably combining the decisions of complementary classifiers misclassification error can be reduced.

Classifier selection is different from fusion: In classifier selection, for a given input, only one or a small number of the models in the ensemble are used whereas in fusion, given an input all models give an output which are then combined, for example by averaging, to calculate the overall output. In this study, we review some classifier selection methods in the literature in a comparative manner. We propose some composite systems which are capable of selecting the optimal subset of the base classifiers from the ensemble dynamically when a test instance is given. We focus on the selection units of these systems and their training so that they learn the areas of expertise of each classifier. In the classification phase, given a data instance, the selection unit allows the calculation and use of the most competent classifiers so that only their decisions are taken into account. For this expertise learning task, we try different algorithms such as decision trees, rule based algorithms, and neural networks.

On 40 datasets and 21 base learning algorithms, we see that by using a well trained selection unit, an ensemble of experts is capable of selecting the experts successfully and improve the overall accuracy. This improvement is significant especially in cases when none of the base classifiers have high accuracy.

ÖZET

YAPAY ÖĞRENME İÇİN BİR UZMANLAR KÜMESİNDEN SEÇİM YAPMA

Karar birleştirme tek öğrenicili sistemlerin başarımını artırmak için son yıllarda yaygınlaşmıştır. Bir sınıflandırıcı kümesinin kullanımındaki temel düşünce, farklı sınıflandırıcılar tarafından yanlış sınıflandırılmış örüntülerin aynı olmayabileceği ve birbirini tamamlayan sınıflandırıcıların kararlarını uygun şekilde birleştirerek yanlış sınıflandırma hatasının azaltılabileceğidir.

Sınıflandırıcı seçme ile kaynaşım farklıdır: Seçmede verilen bir girdi için bir kümedeki modellerin sadece bir veya birkaçı kullanılır, kaynaşımda ise girdi için tüm modeller bir sonuç verir ve bu sonuçlar genel sonucu hesaplayabilmek için örneğin ortalamaları alınarak birleştirilir. Bu çalışmada, yazındaki bazı sınıflandırıcı seçme metodlarını karşılaştırmalı olarak inceliyoruz. Bir deneme verisi verildiğinde temel sınıflandırıcılar kümesinden en uygun alt kümeyi dinamik olarak seçebilen bütünleşik sistemler öneriyoruz. Bu sistemlerin seçme birimlerine odaklanarak, onları her sınıflandırıcının uzmanlık alanlarını öğrenecek şekilde eğitiyoruz. Sınıflandırma aşamasında, verilen bir girdi için seçme birimi en yetenekli sınıflandırıcıların hesaplanması ve kullanılmasına olanak sağlar ve böylece sadece onların kararları hesaba alınır. Bu uzmanlık öğrenimi probleminde, karar ağaçları, kural tabanlı algoritmalar ve yapay sinir ağları gibi değişik algoritmalar deniyoruz.

40 veri kümesi ve 21 temel sınıflandırıcı üzerinde, uzmanlar kümesinin iyi eğitilmiş bir seçme birimi kullanarak başarılı bir şekilde uzmanları seçebildiğini ve toplam başarıyı geliştirdiğini görüyoruz. Bu gelişme özellikle hiçbir temel sınıflandırıcının yüksek başarı gösteremediği durumlarda anlamlı olmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

C_i	i^{th} class in a classification problem
d	Number of attributes of a data instance
D	A classifier ensemble
D_j	j^{th} classifier in a classifier ensemble
D^*	The set of optimal classifiers for a classification problem
k	Number of nearest instances to a specified instance
K	Number of classes in a classification problem
L	Number of classifiers in a classifier ensemble
n	Number of classifiers in a set of optimal classifiers
$P_j(C_i\|x)$	Posterior probability for class C_i estimated by classifier D_j
$p_j(x)$	The weight of classifier D_j to label the input x
r	True class label of a particular x
$R_j(x)$	Output of jth rule based referee for the input x
s	The estimated class label for a particular x

C4.5	A basic decision tree algorithm
EM	Expectation Maximization
HMoE	Hierarchical Mixtures of Experts
Irep	Incremental Reduced Error Pruning
k-nn	k-Nearest Neighbor
LP	Linear Perceptron
MLP	Multilayer Perceptron
MoE	Mixtures of Experts
Ripper	Repeated Incremental Pruning to Produce Error Reduction

1. INTRODUCTION

1.1. Machine Learning

Research on systems which can adapt themselves to a given problem and learn from experience has application in many fields, including computer science, engineering, mathematics, and cognitive science.

Machine learning is programming computers to optimize a performance criterion using example data or past experience [1]. The output of the computer program may be a numeric value in the case of regression, or a group label in the case of classification.

There are many classification algorithms each based on different assumptions and it is very difficult to say which one works best for a given dataset. Moreover, a single classifier that seems to be properly working in general on a classification problem may not cover the whole of the input space efficiently, and it is possible that the patterns which are misclassified by different classifiers may lie in different regions of the input space. Therefore, combining multiple base classifiers that complement each other promises to show better performance than using a single classifier. In the past decades, different aspects of multiple classifiers have been studied and various combination methods have been developed [2]. In some methods, the aim is to find the best way of combining the decisions of a given set of base classifiers while in some others, the aim is to construct a good set of base classifiers.

1.2. Multiple Classifier Systems

The No Free Lunch theorem [3] states that there does not exist one universal machine learning algorithm that performs best on all types of classification tasks. Learning from a finite sample is an ill-posed problem, and each learning algorithm, depending on its inductive bias, fits a slightly different model to the same training data. If these different models make errors on different cases, they can complement each other and

an ensemble scheme can outperform the individual classifiers.

Dietterich [4] states that there are three reasons, statistical, combinational and representational, to justify model combination. Since a learning algorithm tries to find a proper hypothesis in the space, it can find different hypotheses without being trained with (statistically) sufficient data. By constructing an ensemble out of these accurate classifiers, by taking their averages for example, we reduce the risk of choosing the wrong classifier. Another reason is that many learning algorithms like neural networks may get stuck in local optima. For such a case, an ensemble constructed by running many classifiers with different starting points may provide a better approximation. The third reason is valid when the representational assumption of a single classifier does not fit the actual data. When we try to use a linear model for a nonlinear dataset for example, combining multiple linear classifiers can improve the representation ability of the classifier.

The idea of combining multiple classifiers creates new questions to be answered such as which classifiers should be combined and what will be the combining strategy. In the literature, there exist various terms for the same methods since much research on combining classifiers was done in parallel independently. One of the widely used terms is *ensemble classifiers*. Generally, this term is used to state that more than one classifier exists in the system, but more specifically it means that the classification algorithms work independently during training and their predictions are combined during test. The individual classifiers are generally called as *base classifiers*. The systems where different base classifiers are trained with same training set in parallel and all have effects on the output of the system directly are also called *multi-expert systems*.

In *multi-stage systems*, different classifiers are combined serially to train successive classifiers with modified datasets. In the case of cascading, the successive classifiers are trained with the subset of the original dataset on which the previous classifier is erroneous or not confident. The basic idea is that a dataset can be explained with a small number of general rules (that are implemented by the first classifier) and with an additional set of complex rules (in the subsequent classifiers) [5, 6]. In some multi-stage

systems, the output of one classifier is used during training of another [7, 8]. There are also hybrid techniques in which the classifiers are modified by somehow merging one into another [9] or unifying [10].

In the study of Rahman and Fairhurst [11], decision fusion techniques are divided in three groups as vertical, horizontal and hybrid combination schemes. In vertical combination scheme, each classifier receives information from one and supplies information to another. Classifiers in a horizontal scheme work independently and generally are connected to a single element that manipulates the responses of the various classifiers to produce the final solution. The third case is a combination of vertical and horizontal schemes.

Kuncheva [2] divides combination techniques into two groups as decision fusion and classifier selection. In *decision fusion*, each base classifier has knowledge of the whole feature space, as in an ensemble. In *classifier selection*, each base classifier is a local expert, that is, it knows a part of the feature space, and one classifier is selected to label the input. Kuncheva points out that classifier selection approaches, which is the topic of this thesis, are not studied as much as fusion techniques.

1.3. Classifier Selection

The use of multiple classifiers is supported by the fact that the patterns that are misclassified by different classifiers are not necessarily the same [12, 13]. Therefore, a multiple classifier system can be more successful than the individual best classifier in the ensemble if it uses some other classifiers for the instances which are misclassified by the best one.

Dividing the input space into certain regions and searching for the competent classifiers for each region is the simplest methodology used in classifier selection. Different strategies can be used to determine the expertise regions of different classifiers in the input space. These regions may be determined during learning by measuring how accurate each classifier is in each region or during classification dynamically. Moreover,

the regions may be separated from each other with hyperplanes or with more complex, for example, nonlinear boundaries.

1.4. Objectives and Approach

In this study, we are interested in developing a composite system where each classifier's area of expertise is learned and the optimal subset of base classifiers is chosen dynamically for a given test instance.

In the methodology we follow, an ensemble of base classifiers $D_1, \ldots, D_L$ is trained first. Then, given an input, the main aim is to select $n \ll L$ competent classifier(s) from the ensemble so that the accuracy using these n classifiers is maximized and n is minimized. To achieve this, we have to find the regions of competences of the base classifiers, estimate their competences in there, and choose a selection strategy.

1.5. Thesis Organization

The outline of this thesis is as follows. In Chapter 2, different approaches for multiple classifier systems are examined and classifier selection techniques in the literature are categorized. In Chapter 3, we describe multiple classifier systems existing in the literature. The proposed methods on multiple classifier systems are explained in Chapter 4. In Chapter 5, the methods are analyzed according to different aspects. The experiments and the results using 21 learning algorithm on 40 datasets are given in Chapter 6. The conclusions and possible future work are stated in Chapter 7.

2. CLASSIFIER SELECTION METHODOLOGIES

2.1. Selecting Competent Classifiers

There is not as much work on classifier selection as there is on decision fusion [2, 14, 15]. Thus, the selection of competent classifiers among many base classifiers remains an important research problem. In classifier selection, improving the accuracy is just one of the goals. It is also possible to decrease the cost of classifier system by using complex classification algorithms only when they are really needed and using simpler algorithms elsewhere in the input space. Using small ensembles, preferably consisting of a single base classifier, also makes classifier selection more interpretable than decision fusion.

2.2. Systems for Determining Optimality

The idea of using different classifiers for different inputs was suggested by Dasarathy and Sheela in 1978 [16]. In their study, the base classifiers are designated as a linear classifier and a k-nearest neighbor (k-nn) classifier. The composite classifier identifies a conflict domain in the input space and uses k-nn in that domain while the linear classifier is used elsewhere. In their study [16], the application region of each classifier is separated with hyperplanes and the problem of finding these hyperplanes is defined as an optimization problem. The optimal classifier system is achieved when the conflict domain bounded with hyperplanes is small enough that only a small amount of data is classified with the costly k-nn classifier, and it is large enough that data outside the conflict region is linearly separable. When a linear classifier is used, searching for the optimal hyperplane in the learning phase can be useful since the classification phase becomes simpler. However, in multi-class problems, it may be costly to optimize all conflict regions where instances of different classes occur together.

Dasarathy and Sheela [16] define the base classifiers before the learning phase starts. However, it is also common in multiple classifier systems that one or more

classifiers are selected from an ensemble during the learning phase or even during the classification phase dynamically for a particular input. There are various approaches to select n classifiers from an ensemble of L base classifiers (where $n \leq L$) as follows:

a) Calculating competence of the classifiers

b) Using a classifier selector

c) Using a prior ordering of the classifiers

2.2.1. Local Competence Based Systems

Competence calculation is based on estimating the accuracy of base classifiers in particular input regions. Once the region of a test instance is determined, we can find the subset of base classifiers that are competent in there.

Instead of defining the application regions of base classifiers during learning [16], one may find the most efficient classifier for a particular instance by calculating the competence of classifiers *dynamically* during test. Ho *et al.* [17] propose the concept of dynamic classifier selection as an alternative to combination in multiple classifier systems. In the system they propose, rankings of classes are used instead of unique class choices or numerical scores. The proposed selection method is based on partitioning the training set according to the state of agreement of the classifiers on the top choices. A different combination function based on logistic regression is then used for each partition.

Woods *et al.* [18] propose a method for dynamic classifier selection, which is based on local classifier accuracy estimates. For each classifier, an estimate of accuracy in a local region is computed where local regions are defined in terms of the k nearest training neighbors of a test instance. (We are going to discuss Woods *et al.*'s system in more detail in Section 3.6.) In the study of Giacinto and Roli [19], local accuracies are estimated by using the class posterior probabilities, while in the system of Woods *et al.* [18] simply the labels assigned to the training samples are used. Moreover, the distances between the test instance and its neighbors are used within the calculation of local accuracies. This helps us with the problem of choosing k, i.e. the size of local

region, and provides more robust estimates [20]. The system can also *reject* a test instance if none of the local accuracies is higher than a predefined threshold. However, both methods become too costly in terms of calculations for large datasets since various pairwise distances need to be calculated to define the local region.

In order to define the input regions during learning, one way is to use hyperplanes to divide the input space in distinct parts [16]. Another way can be to find data groups and then determine which classifier should be responsible for each group. Kuncheva [21] proposes a method that first clusters and then searches for the most accurate classifier in each cluster. In this approach, classification phase consists of just finding the nearest cluster of a test instance and using the classifier responsible for that cluster. Clustering is an easy and proper way of defining where we should focus on the input space. However, it is risky in that the clusters in the input space may not match with the patterns of success of the base classifiers we have. Frosyniotis *et al.* [22] also mention the benefit gained by clustering data instances before the classifiers are trained. Their multi-net classification system starts by dividing the input space into clusters which may overlap and continues with training multilayer perceptrons for each cluster. In this study, the outputs of base classifiers are combined using different decision fusion schemes. This learning method also offers the advantages of the divide-and-conquer framework, i.e. smaller networks may be employed by training them in parallel in different regions of the input space.

Competence based strategies are generally used to find only one most competent classifier. However, it is very probable that in certain input regions using more than one classifier can give more accurate outputs. To improve overall classification accuracy, Kuncheva [23] suggests using fusion in regions where there is not a single dominating base classifier. Woods *et al.* [18] and Giacinto and Roli [20] also point out that a classifier should be selected as the most competent classifier only if that classifier is significantly better than the second best. Otherwise, a scheme involving more than one classifier, such as randomly selection between best classifiers, or fusion their outputs, for example by taking a vote, should be used.

2.2.2. Systems with a Selector

Another way of determining an optimal set of responsible classifiers for subdomains of the input space is by using an *automated selector* or a *gating system* whose basic task is to decide which classifiers will be used for a given test instance.

Stacked generalization is proposed by Wolpert [24] as a meta-learning method where all base classifiers are trained with the whole training set and a combiner, i.e. a higher level classifier, is trained with the outputs of base classifiers to learn the mapping from the output of base classifiers to the desired output. The role of the higher level classifier is to learn how the base classifiers make mistakes. In some studies, stacking is found to be more accurate than traditional voting strategies [25] whereas the cost of meta-learning sometimes makes it less attractive [26].

Stacking with a linear combiner is similar to the adaptive mixtures of experts (MoE) method which is proposed by Jacobs *et al.* [27]. The base classifiers are named as *experts* since each of them is specialized in a different region of the input space. In the original study, each local expert is a single layer network (a linear perceptron) and there is another neural network named the gating network, used to select one of the experts. The gating network also uses the original input and generates as output a set of coefficients, corresponding to the mixing weights of the experts. The weights sum up to 1 and each weight is interpreted as the probability that the corresponding expert is the most competent expert for the particular input. In a cooperative model, more than one expert can be chosen. In the case of a competitive model, only one of the local experts is chosen. Alpaydın and Jordan [28] compare cooperative and competitive models and conclude that the cooperative model is generally more accurate whereas it learns more slowly. They also emphasize that local linear models are promising even on high dimensional datasets. Furthermore, when compared with multilayer perceptrons, local models learn much faster without increasing the number of parameters much [28].

Hierarchical mixture of experts (HMoE) model is proposed by Jordan and Jacobs [29] which has hierarchically ordered gating networks in a decision-tree like structure.

Leaves are the base classifiers (experts) that will make predictions on the input and each nonterminal node is a gating network. Each gating network takes a weighted sum of the contributions coming from its children and the averaged prediction is propagated up. The root gives the final decision, which is a weighted average of decisions of all leaves. To train the system one can use the standard error backpropagation or the expectation maximization (EM) method, which is observed to be faster than the gradient techniques [30].

The gating probabilities in MoE structure can be used in different ways: In the general case, they are used as the weighting coefficients to the classifier outputs, and this corresponds to weighted voting where votes are input dependent. Alternatively, if they represent expert posterior distribution, then a classifier can be selected by sampling from this distribution (stochastic selection) or simply the maximum can be selected (winner-takes-all). Ortega [31] and Koppel and Engelson [32] independently propose a classification system that is similar to the MoE approach and utilizes winner-take-all method [33]. In their systems, a *referee* for each base classifier is built and an *arbiter* is used to select the most competent classifier based on the outputs of the referees.

It is also possible to optimize properties other than accuracy. Demir and Alpaydın [34] propose dynamic selection algorithms which are based on a utility criterion to maximize the accuracy while minimizing the computational cost. Different models use different input representation and selecting a model corresponds to selecting the best representation for the given instance. Their first algorithm determines a subset of classifiers while the second dynamically selects a single base classifier for each particular instance from the ensemble by calculating utility of the classifiers with simple estimators.

A common point of MoE and HMoE approaches is that both approaches are designed to use simple local models to create a more complex and powerful classifier. Note that during the classification phase, the outputs of different experts are combined (in the case of cooperative model) or one of them is selected (in the case of competitive model) by using an *input dependent* gating network. There are some other data

dependent approaches where the use of dependence provides higher accuracy values. Trained combining methods, in general, are data dependent [35]. This dependence can be either implicit or explicit [36]. Implicit dependencies include methods that train the combiner on the global performance of the data. The weighted average, fuzzy integrals and belief theory approaches can be categorized as implicitly data dependent [2, 36]. The techniques proposed by Woods *et al.* [18] and Giacinto and Roli [19] are examples of explicit data dependent approaches since they perform classifier selection in local subspaces of the input space. Wanas *et al.* [36] propose an input dependent decision fusion scheme where a *detector* module defines weights of classifiers that will be used in the aggregation stage. The detector uses both the input features and the classifier outputs to estimate the weights of classifiers. From this point of view, this approach can be viewed as a mix of the gating network in MoE structure and the upper level classifier in stacked generalization.

2.2.3. Systems Using a Prior Order

In some applications, we have a prior preference over base classifiers, which specify an ordering generally due to complexity, and we use them according to this order in a sequence and do not use the next one unless the preceding ones are not confident.

Pudil *et al.* [37] propose a sequential classifier system with reject option. In the example study, they use Bayes decision rule in both stages of the system. The first classifier works on the data after its dimensionality is reduced while the second classifier works on enlarged features and therefore, has higher cost. It is stated that by utilizing the costlier classifier only for those instances rejected during classification in the first stage, the average decision risk can be decreased.

The studies on sequential classifier systems differ by the strategies they use to train subsequent classifiers. In some systems, since the subsequent classifiers are used for more difficult problems, the first classifier is trained on the original dataset whereas the second classifier is trained with those instances where the first classifier is not accurate or confident. In some other systems, the first classifier works with data which

have a reduced number of dimensions and the subsequent classifiers work with the original high dimensional data. The number of dimensions of the instances used by the subsequent classifiers is increased in some other systems with the output of previous classifiers because of the assumption that the subsequent classifier can make a more accurate decision if it knows the output of previous classifier.

In the study of Chan and Stolfo [38], all base classifiers are trained on the original input space and then a special classifier named *arbiter* is trained with the instances whose class labels cannot be predicted by the base classifiers consistently. Inconsistent predictions are those where base classifiers do not agree on a class label using majority vote. In the classification phase, all base classifiers are used in parallel first; for the inconsistent instances, the arbiter makes the final decision.

AdaBoost is proposed by Freund and Schapire [39], and is also based on training different classifiers iteratively by using misclassified examples more often as training data. However, it is an example of classifier combination method rather than selection. Boosting assigns weights to each instance in the training set that reflects its importance and the misclassified examples are weighted more. After training the last classifier, the decision of all classifiers are aggregated by weighted voting where the weight of each classifier is a function of its accuracy. Wanas *et al.* [36] also try to retrain the ensemble system with a part of training set which mostly includes misclassified examples. They conclude that cooperative retraining of the base classifiers improve the ensemble accuracy.

One of the studies where the original training dataset is used for the first classifier and only a subset of the dataset is used for subsequent classifiers is proposed by Alpaydın and Kaynak [5]. They use the term *cascading* where there is a sequence of classifiers and subsequent classifiers are used only when previous classifiers are not confident [5]. The subsequent classifiers are trained not only with the instances misclassified by the first classifier but also for which the first classifier is not confident. Because the first classifier (a multilayer perceptron) handles a large percentage of the instances using simple and general rules, the exceptions are classified with a nonpara-

metric method (like k-nearest neighbor classifier). The benefit of cascading over a horizontal combining technique such as voting is that in horizontal combination, all base classifiers give output for all test instances whereas in cascading, fewer classifiers are used for most of the data instances.

Egmont-Petersen *et al.* [40] also propose a sequential classifier system where a subsequent base classifier is used when reliable class label cannot be assigned by previous base classifiers. Similar to the study of Pudil *et al.* [37], in their study, the dimensionality of data is reduced. In the learning phase of the system they propose, a classifier (a feed-forward neural network) is built using all irredundant features of the dataset and this classifier becomes the last classifier in the classifier sequence. The remaining features are successively removed one by one using an iterative backward search, so that each classifier uses a smaller subset of features than the classifier in the previous iteration. In the classification phase, the first classifier in the cascade obtains the minimally required set of features as input vector. If the decision of the first classifier is not reliable, the instance is propagated to the next classifier which uses more features to make the class decision.

Gama and Brazdil [8] define *cascade generalization* where base classifiers are generated sequentially. While training the subsequent classifiers, new attributes from the class probability estimations given by the previous classifiers (i.e., decision trees) or by previous decision tree splits are also inserted as new input features to the original dataset. In one of their studies [8] for example, naïve Bayes classifier is trained with the original features and the class conditional probabilities are added to the original features as if these probabilities are just new features. The second classifier (C4.5 decision tree) runs on this modified (enlarged) dataset. It is stated that the performance of the decision tree increases significantly despite the fact that naïve Bayes cannot fit thoroughly on the original problem. This study also shows that high performance can be achieved by combining radically different types of classifiers. Indeed, this study is not based on the classifier selection idea since selecting just one or a few of the base classifiers is not considered, but it is an example of sequential classifier systems. Ting and Witten [41] have a similar system but they use just the probability class

distributions as input to the subsequent classifiers, but do not use the original features.

Ferri *et al.* [42] propose a serial system similar to cascading where the first base classifier is able to state the confidence of its decision and it rejects an instance if the confidence threshold is not achieved. Training the second classifier with fewer and specific examples increases its accuracy. One possibility is that a classifier retains a fixed percentage of the examples. For example, it may be stated that the first classifier retains 60% of the most highly ranked instances, delegating the rest to the second classifier. An alternative decision rule for the delegating classifiers is applying this percentage method on training examples belonging to each class separately and defining separate threshold values for different classes [42, 43]. It is important to handle the trade-off between cost and accuracy carefully. Assume that our first classifier is a simple algorithm and the second classifier is more complex in terms of calculation and therefore costly. When the approach mentioned above is used, the threshold value can be too small on some datasets which means we force the first classifier to make decisions even when it is not confident. On the other hand, if we do not control how much data is left to the second classifier, the first classifier may be doing nothing but consuming time since it makes accurate decisions rarely and leaves almost all data to the second classifier.

In a cascading system, a subsequent classifier is used only if the previous one is not confident. If the last classifier in the system also has the reject option, we should determine a strategy about how to classify rejected data. Alpaydın and Kaynak [5] consider them as exceptions not covered by any rules and keep them in a lookup table and use k-nn. Ferri *et al.* [42] state that if an example is rejected by all classifiers, it should be classified by the *first* classifier rather than the *second* to provide less overfitting (round rebound). Kuncheva [2] emphasizes that some decision fusion strategies such as fixed rules can also be used since the outputs of all base classifiers in the combined scheme are already known.

In decision fusion where base classifiers work in parallel and their final decisions are combined, there is an extra cost during both learning and classification. If the

increase in accuracy gained due to combination is significantly high, the cost may be disregarded. However, improving a single classifier by reducing the overall cost of combination is possible with cascading. Imagine a large input space where k-nn classifier is sufficiently accurate. In such a problem, the cost of both learning, i.e. storing large amounts of data, and classification, i.e. calculation of pairwise distances, are undesirably high. If we can find and use a less costly but sufficiently accurate classifier, such as a linear classifier, on some part of the input space and use k-nn rarely, then the overall cost will decrease without any significant degradation in accuracy.

3. PREVIOUS WORK ON CLASSIFIER SELECTION

3.1. Terminology

We use the following notation for both the classifier selection strategies existing in the literature given in this chapter and the proposed methods:

- It is assumed that we have L base classifiers in a classifier ensemble: $D = [D_1, \ldots, D_L]$.
- All base classifiers D_j, $j = 1, \ldots, L$, are trained with the training dataset X_{tra}.
- Output of any base classifier D_j is the estimated posterior probabilities $P_j(C_i|x)$, $i = 1, \ldots, K$.
- A selection procedure is constructed by using the validation dataset X_{val}.
- The aim of any selection procedure is to find an optimal subset $D^* \subseteq D$ that includes $n \leq L$ of the classifiers for a given test instance x.
- The decisions of the selected classifiers are combined in the decision aggregation (DA) unit of the system. If a single classifier is selected, the final decision of D^* is the decision of the selected classifier. However, it is possible to select more than one classifier and when $n > 1$, voting is applied over the posterior probabilities of the selected classifiers to generate the final decision:

$$P_{D^*}(C_i|x) = \frac{\sum_{j=1}^{L} P_j(C_i|x) 1(D_j \in D^*)}{n}, \; i = 1, \ldots, K \tag{3.1}$$

  We use 0/1 loss and $1(D_j \in D^*)$ is 1 if D_j is one of the chosen classifiers: $n = \sum_{j=1}^{L} 1(D_j \in D^*)$.
- The final output is the class label s that has the highest posterior probability:

$$s = \arg\max_i P_{D^*}(C_i|x) \tag{3.2}$$

3.2. Voting over Posterior Probabilities

Voting corresponds to $D^* = D$ and the output of all base classifiers are combined (Figure 3.1).

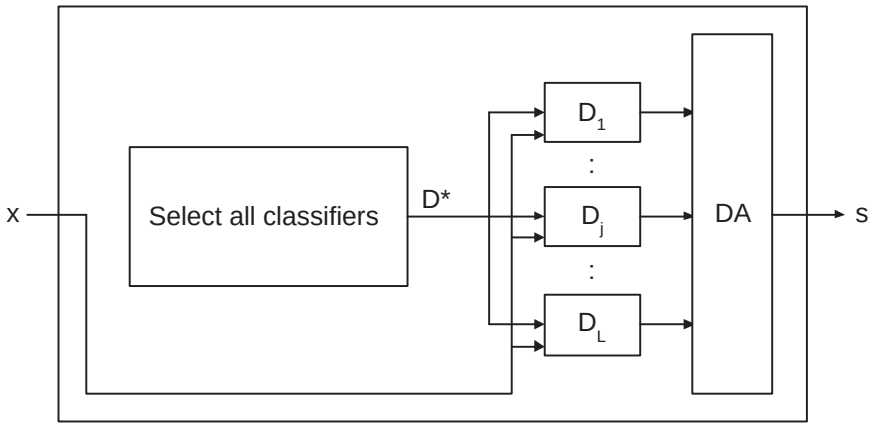

Figure 3.1. Schema of the voting method

3.3. Selecting the Best Classifier Statically

This classifier selection technique is based on the idea that the classifier with the lowest validation error will probably have the lowest error on test instances, too. According to this assumption, it is a good idea to find the classifier which has the highest accuracy on the validation data, and then use only this classifier for all test instances. Note that in a resampling strategy when we have multiple validation/test folds, the classifier selected with this strategy is not unique, since we have multiple folds and the best classifier can vary on each fold.

Given that r is the desired output for the particular x, the misclassification error is calculated as follows (Figure 3.2):

$$E_j(x) = \begin{cases} 1 & \text{if } \max_i P_j(C_i|x) \neq r \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

$$D_w \text{ is the best classifier if } w = \arg\min_j \sum_{\forall x \in X_{val}} E_j(x) \tag{3.4}$$

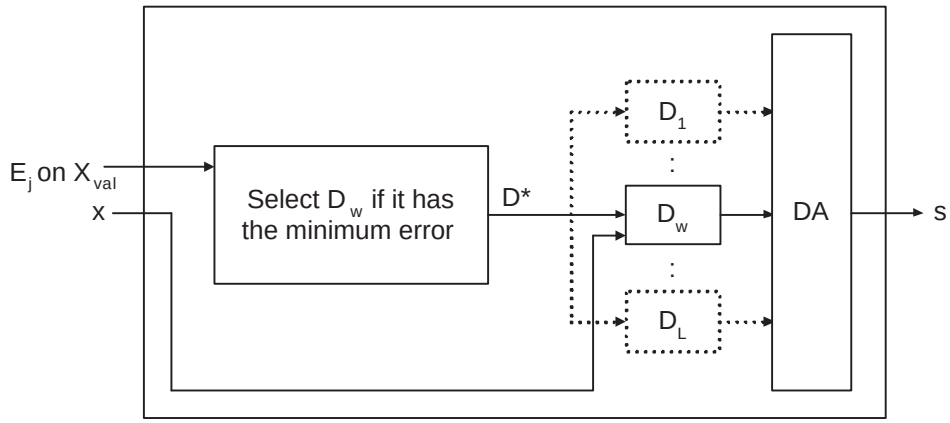

Figure 3.2. Schema of the method that selects the best classifier statically

It is never guaranteed that the validation data represents the whole of the problem space sufficiently, but still it is an important statistic to be used to predict unseen situations.

It is also possible to choose $n > 1$ base classifiers instead of a single one by sorting base classifiers in terms of validation accuracy and choosing the best n. In such a case, voting is used to calculate the final output. The pseudocode is given in Figure 3.3.

1. Find the total error E_j for each classifier D_j on X_{val}.
2. Select n classifier(s) that has the minimum error.
3. For any given x, get the estimates of the selected classifiers.
4. Apply voting over the estimates (Equation (3.1)).
5. Predict the class label (Equation (3.2)).

Figure 3.3. Selecting the best classifier statically

In test phase of this system, all the classifiers except the best are disregarded and only the best classifier is used. However, this system is still able to choose different classifiers for different datasets. Therefore, it makes the analysis of relationship between the datasets and their corresponding successful classifiers possible.

3.4. Selecting the Best Classifier Dynamically

To select one of the base classifiers, one simple strategy is to select a classifier which is confident of the prediction it has made for the particular test instance. The highest posterior probability that is predicted by a base classifier D_j is considered as its confidence:

$$\text{Confidence of } D_j = \max_i P_j(C_i|x) \tag{3.5}$$

We name such a confident classifier as the dynamic best classifier since the more confident a classifier is the higher probability is there that it will estimate the true class.

$$D_w \text{ is the dynamic best classifier if } w = \arg\max_j \max_i P_j(C_i|x) \tag{3.6}$$

This system checks all base classifiers for each test instance dynamically in classification phase and the selection is based on the particular test instance rather than the validation data (Figure 3.4). That is, as opposed to static selection, for different test instances, different base classifiers can be selected.

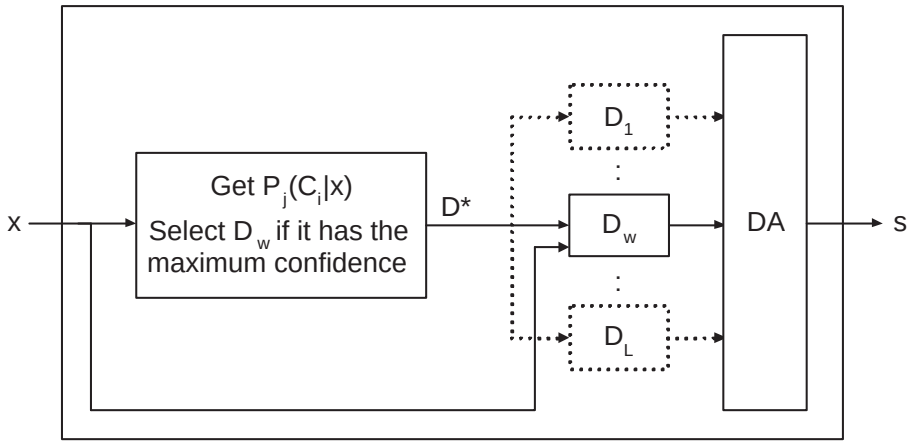

Figure 3.4. Schema of the method that selects the best classifier dynamically

As before, it is possible to select $n > 1$ classifiers and take a vote over them instead of selecting just one single best classifier. The pseudocode is given in Figure 3.5.

1. Given x, find confidence of all classifiers (Equation (3.5)).
2. Select n classifier(s) that has the maximum confidence (Equation (3.6)).
3. Apply voting over the estimates of the selected classifiers (Equation (3.1)).
4. Predict the class label (Equation (3.2)).

Figure 3.5. Selecting the best classifier dynamically

The requirement for this system to work efficiently is that all base classifiers should give outputs in the same scale. This is clearly the case if the outputs are posterior probabilities, otherwise there is the need for scaling and normalization.

3.5. Selecting Classifiers Randomly

To see what improvement static or dynamic selection brings, we also implement random selection, where a single, or $n > 1$ base classifiers are chosen at random uniformly (Figure 3.6).

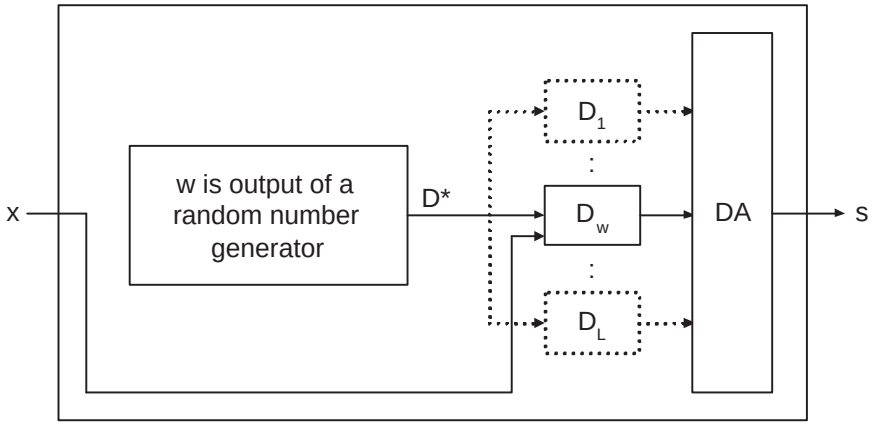

Figure 3.6. Schema of the method that selects classifiers randomly

The accuracy of the random selection will act as a benchmark to indicate the improvement of taking validation error or confidence into account. The pseudocode is given in Figure 3.7.

1. Given x, select n of L base classifiers randomly.
2. Get the estimates of all selected classifiers.
3. Apply voting over the estimates (Equation (3.1)).
4. Predict the class label (Equation (3.2)).

Figure 3.7. Selecting classifiers randomly

3.6. Selecting Classifiers Using Local Competence

We now review the classifier selection method proposed by Woods *et al.* [18] in more detail. As we have previously discussed in Section 2.2.1, this method is based on the calculation of local competences of classifiers, and the aim is to select classifiers from a classifier ensemble *dynamically*.

Local competence of a classifier is defined in two different ways: In the *class independent* method, the competence is defined as the local accuracy and the local region is defined as the area where k nearest neighbors of a particular test instance are located. In the *class conscious* method, the competence is defined as the local class accuracy, where we consider the k nearest neighbors whose estimated class labels are same with the estimated class label of the test instance. In both cases, the accuracy of a particular classifier on the k instances is estimated by dividing the number of correctly classified neighbors by k. The value of the parameter k is determined before classification (it is taken as 10).

$$\text{Local competence of } D_j = \frac{\sum_{\forall x \in x_n} 1(\max_i P_j(C_i|x) = r)}{k} \tag{3.7}$$

where

$$x_n = \begin{cases} k \text{ nearest neighbors} & \text{if class independent} \\ k \text{ nearest neighbors, } \max_i P_j(C_i|x_n) = \max_i P_j(C_i|x) & \text{if class dependent} \end{cases} \tag{3.8}$$

Once the competences are calculated, the selection procedure selects the classifier that has the highest local competence (Figure 3.8).

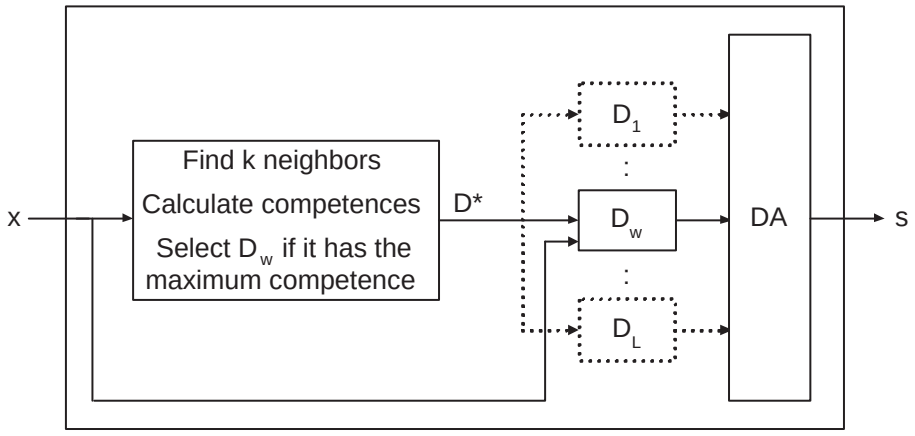

Figure 3.8. Schema of the method that selects the most locally competent classifiers

The pseudocode of the algorithm for the class independent case is given in Figure 3.9. As mentioned before, only the definition of the neighbors changes in the class dependent case.

1. Given x, find its k nearest neighbors.
2. Get the estimates of L classifiers for them.
3. Calculate local accuracies of the classifiers on the neighbors.
4. Select the classifier with the highest accuracy and predict the class label. If a tie occurs, use the tie breaking procedure given in Figure 3.10.

Figure 3.9. Selecting classifiers using local competence

We implement both versions, class independent and class dependent cases namely, with two minor differences: The original algorithm starts with getting class predictions of all base classifiers for given x. If all classifiers agree on the class label, then this label is returned as the final decision. Classifier selection procedure is applied whenever a disagreement occurs on the class label. In our implementation, it is possible to use the selection procedure before getting the outputs of all classifiers so that the system always chooses n classifiers.

Furthermore, when the competences of all classifiers are compared, a tie breaking

- Compare the competences of all classifiers. If there is a unique winner, use that classifier to guess the class label of x and return. Otherwise, if there is a classifier tie, obtain the class labels guessed for x by all winners.
- If a unique class label is guessed by all winners or by plurality of them, then assign this label to x and return. Otherwise, if there is a class label tie, the classifier with the next highest local competence is identified to break the tie.
- If there is no classifier left to break the tie, pick a random class label among the tied labels and return. If there is a unique winner class label in the second competence contest, and it can resolve the tie, then use this winning label for x and return.
- If none of the clauses in the previous steps applies, pick a random class label among the tied labels and return.

Figure 3.10. Tie breaking procedure of local competence based system

procedure is applied if there is no single winner. In the original paper [18], class label ties are handled by choosing the label that is selected most often among the classifiers with the next highest local competence. However, there is no guarantee that a class label tie will not occur when all base classifiers are used, and therefore, randomly choosing one of the tied labels is unavoidable. That is why we prefer to implement tie breaking procedure with the minor difference proposed by Kuncheva [2, 23].

Note that when more than one most competent classifier is found, i.e. there is a classifier tie, our system is allowed to use up to n of these classifiers.

4. PROPOSED METHODS FOR CLASSIFIER SELECTION

4.1. Parameters of Selection Systems

Performance of a classifier selection system depends on various parameters such as the type of base classifiers used, the difficulty of the classification problem, the correlation between the classifiers, and so on. Most of the classifier selection methods are based on the assumption that different base classifiers are accurate in different parts of the input space. Therefore, the most important stage of a classifier selection system is to divide the input space into regions properly. Thus, in one part of our research, we discuss methods which focus on the definition of the regions and the competences of classifiers in there. In our work, we construct methods that learn the regions automatically for each base classifier so that they tell the selection system which base classifier should be used for a particular test instance.

Besides defining regions of expertise of the base classifiers, the system should have a proper strategy to combine decisions of the selected base classifiers. In the second part of the research, we focus on a centralized gating selector that decides on partitioning of the regions and the weights of the base classifiers.

4.2. Systems with Referees

We start with L base classifiers in a classifier ensemble $D = [D_1, \ldots, D_L]$ and the goal is to find the optimal subset D^* that includes n classifiers for any test instance, where $n \leq L$. The optimal subset is the one that has the highest probability of being accurate and the smallest cardinality, that is, we would like to maximize accuracy and minimize the subset size and so the complexity.

In our approach, all of the base classifiers have already been trained. Then, for each base classifier, a *referee* structure is trained. The task of a referee is to

estimate in which parts of the input space the corresponding base classifier is expected to produce correct classification. For any given instance, the subset will contain those base classifiers whose referees return positive. The task of the learning expertise areas of a base classifier can be seen as a new learning problem and different machine learning algorithms can be used to learn these expertise domains.

Figure 4.1 illustrates the training of the referees assuming that the base classifiers D_j have already been trained on X_{tra}. To evaluate the success of classifiers and to train the referee structures, we use the validation set X_{val}. We assume that the output of any base classifier D_j for any data instance x is a vector of class posterior probabilities $P_j(C_i|x)$, where $i = 1, \ldots, K$ denotes class indices, rather than a single class label or a list of ranked class labels. Each referee takes the same input in X_{val} as well as the output vector of the corresponding classifier and is trained either as a binary classifier (the base classifier is correct/wrong) or as a regressor (the posterior probability estimate of the base classifier for the correct class).

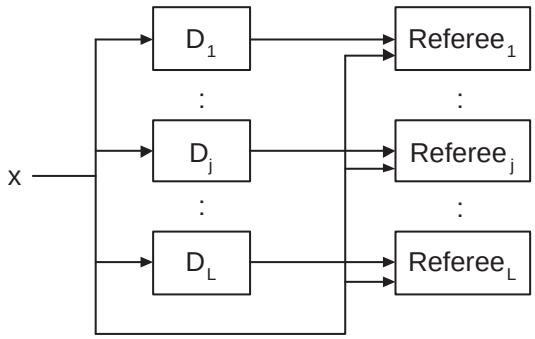

Figure 4.1. Training phase of the referee based system

In the testing phase, a test instance x is sent to the referees first, and the optimal set of classifiers which will be used for the instance depends on the estimations of the referees. Then, only the outputs of the classifiers in the optimal set are used, for example, by taking a vote, as illustrated in Figure 4.2.

Due to the difficulty in learning the expertise areas of the base classifiers as we have seen before, additional features can also be used to train the referees. Ortega [31] and Koppel and Engelson [32] propose to use important attributes of a base classifier and the class predictions as well as the original attributes. The drawback of their

system is that to learn each referee, one has to explore the features of its base classifier and therefore, execute the base classifier before the referee. The system performance will critically depend on the quality of the learned referees as well as the base classifiers. In our work for generality, the referees use only the original attributes although it is emphasized by Ortega *et al.* [33] that this makes finding the solution harder. The reason why we do not use the internal features of base models and the class predictions is that we want to find the optimal set of classifiers for each test instance *without running all the base classifiers first*. Moreover, our proposed method does not select just one best classifier for each region but more than one classifier can be selected.

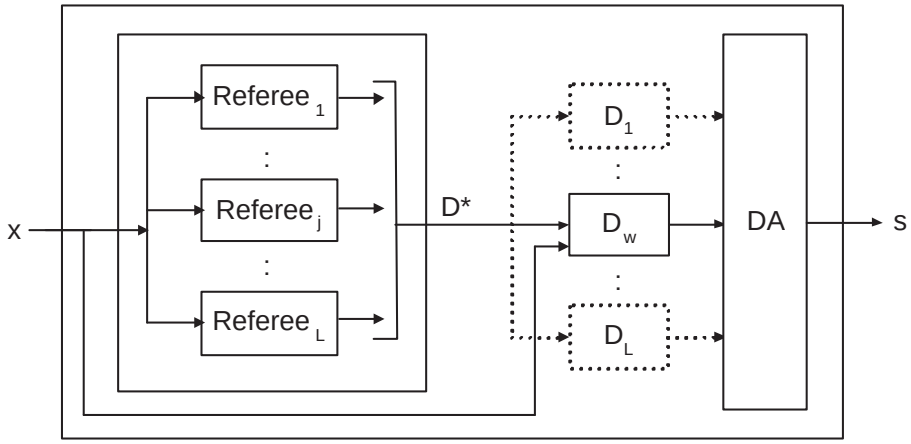

Figure 4.2. Testing phase of the referee based system

To construct the referees we use different machine learning techniques. We use decision trees, rule based systems, as well as neural network based perceptrons. In the following sections, the proposed classifier selection systems are explained with the assumption that we have already trained L different base classifiers. After the construction of referee of each classifier, with the help of the referees, the system will be able to decide which n of the L classifiers should be selected for a particular test instance.

4.2.1. Classification Tree Based Referees

The task of a *referee* is to learn whether an instance would be correctly classified by its base classifier if it were given as input. Before training a referee, each instance of X_{val} is classified with the corresponding base classifier and X_{val} is divided into two

distinct groups as correctly (X_C) and incorrectly classified (X_I) examples. The referee has to learn to map the given x to one of the classes, i.e. X_C or X_I.

To train the referees and solve this new two-class problem, different classification algorithms can be used. However, this problem is most probably much harder than the original classification problem. In a classic two-class problem, we would normally expect the instances of the two classes to be close to each other; on the other hand, areas of success and failure of a base classifier may not have a specific pattern in the input space. If the original classification problem includes K classes to be learned, then we may have many disjoint regions where the base classifier correctly classifies the instances. Thus, the new two-class problem includes more than two groups of instances.

Since *tree induction algorithms* are widely used for various classification problems and they are capable of solving problems without making any parametric assumption [44], we construct each referee by using the C4.5 algorithm [45]. To prevent large trees and reduce the probability of overfitting, we use prepruning, where node creation stops and a leaf node is created when the number of instances covered by a node is less than five percent of the training set size, even if the entropy is still high.

It is possible to define a maximum limit n for the number of base classifiers to be selected. To make this possible, we define another parameter on the leaves of the trees, which indicates the competence of the corresponding base classifier. It can be calculated in two ways; one of which is the confidence level (the average posterior probabilities estimated for true classes) and the other is the ratio of the correctly classified instances to the instances covered by the leaf. In the case of a tie between the ratios, the number of correctly classified examples is taken into account. In the referee of the base classifier D_j, the value of the parameter on the leaf that covers the given x is denoted as $R_j(x)$. In the selection unit of the system, $R_j(x)$ values of all referees, $j = 1, \ldots, L$, are obtained and sorted. Then, n classifiers with the highest $R_j(x)$ values are selected.

After the posterior probability estimates of the selected classifiers are obtained, a decision fusion technique such as simple voting or weighted voting can be used to obtain the final decision. In the case of weighted voting, the weights are usually defined as the confidence values of classifiers for the test instance. Since we have already used these confidences in training the referees, we use simple voting to get the final decision of D^*:

$$P_{D^*}(C_i|x) = \frac{\sum_{j=1}^{L} P_j(C_i|x)1(D_j \in D^*)}{n}, \ i = 1, \ldots, K \tag{4.1}$$

The class label s that has the highest posterior probability is the final output of the system:

$$s = \arg\max_i P_{D^*}(C_i|x) \tag{4.2}$$

Training and testing phases of the proposed algorithm are summarized in Figures 4.3 and 4.4.

1. Train each base classifier D_j separately with X_{tra}.
2. Classify X_{val} with D_j to find X_C and X_I.
3. Construct a classification tree to distinguish between X_C and X_I.
4. Label each leaf as "UseMe" if it covers X_C; otherwise, "dontUseMe".
5. Define R_j parameters on each leaf of the tree which can be calculated as
   - The average posterior probabilities estimated for the true classes, or
   - The ratio of correctly classified examples.

Figure 4.3. Training phase of the rule based referees using hard labels

Since a referee for a base classifier is trained separately from other base classifiers both in our study and the previous referee based studies [31, 32], individual base classifiers can be added and/or removed without affecting other referees.

1. Given x, classify it with all referees.
2. Whenever the label is "UseMe", add the classifier to the list of willing classifiers.
3. If the number of willing classifiers does not exceed n, select all of them.
4. Otherwise, sort them according to $R_j(x)$ and select n classifiers with the highest parameters.
5. Apply voting over the estimates of the selected classifiers (Equation (4.1)).
6. Predict the class label (Equation (4.2)).

Figure 4.4. Testing phase of the rule based referees using hard labels

4.2.2. Regression Tree Based Referees

Above, in training the referees, we use hard labels for data instances: "UseMe" or "dontUseMe". It is possible to increase the reliability of the referee based selection by defining how confidently the instances in an input region are classified or how much a classifier should participate in the final decision. That is, we can use soft labels and consider this as a regression problem.

In general, the values of posterior probabilities estimated by a classifier represent how much the classifier is sure about the decision it has made, as discussed in Section 2.2.3. For example, in a four-class problem, if the posterior probabilities are $[0.28, 0.24, 0.24, 0.24]$ then class label will be C_1, but such a decision is less reliable than the case when the posterior probabilities are $[0.97, 0.01, 0.01, 0.01]$. If we know that real class label is C_1, the second classifier is said to be more confident.

In this variant, each referee estimates how confident is its corresponding base classifier by estimating the posterior probability of the true class, turning this into a regression problem. The training instances are labeled with the posterior probabilities of the true classes and the regression tree algorithm is used to construct each referee. The construction procedure for a regression tree is almost the same as a classification tree except that the impurity measure that is appropriate for the classification (entropy) is replaced with an error measure appropriate for regression (sum of squared error).

The training and testing procedures of the system with regression tree based referees are summarized in Figures 4.5 and 4.6.

1. Train each base classifier D_j separately with X_{tra}.
2. Get the estimates of D_j, i.e. $P_j(C_i|x)$, for all instances in X_{val}.
3. Construct a regression tree to estimate the probabilities of true classes.
4. Label each leaf with R_j that is calculated as the average posterior probability of the covered examples.

Figure 4.5. Training phase of the rule based referees using soft labels

1. Given x, feed it to all referees and obtain $R_j(x)$ values.
2. Sort the classifiers according to $R_j(x)$ and select n classifiers with the highest values.
3. Apply voting over the estimates of the selected classifiers (Equation (4.1)).
4. Predict the class label (Equation (4.2)).

Figure 4.6. Testing phase of the rule based referees using soft labels

4.2.3. Rule Learner Based Referees

We also use Ripper (Repeated Incremental Pruning to Produce Error Reduction) algorithm proposed by Cohen [46], as the classification method to construct the referees of base classifiers. Opposed to a decision tree, which is based on the divide-and-conquer approach, Ripper belongs to the group of rule induction algorithms, called *separate-and-conquer*, or sequential covering algorithms, and is a later version of the Incremental Reduced Error Pruning (Irep) algorithm originally developed by Fürnkranz and Widmer [47].

To construct the referees in this system, the same strategy that is used with the classification tree based referees (Section 4.2.1) is applied. The difference is that we use Ripper instead of C4.5 to distinguish between correctly (X_C) and incorrectly classified examples (X_I).

4.2.4. Perceptron Based Referees

We also use different neural network structures to construct the referees. Because our previous experiments with decision trees show that hard labeling provides higher accuracy than soft labeling, we again consider this as a binary classification problem and use hard labels as correctly or incorrectly classified example.

The referees take input attributes and estimate whether the corresponding base classifier can correctly classify it or not. For this purpose, we train a perceptron whose target output is 1 or 0 to denote correct or incorrect classification, respectively. This problem can be defined as a two-class problem but we do not want to enforce the output of the referee to be 0 or 1. On the contrary, we want to get values between 0 and 1 so that sorting the base classifiers according to these competence estimates will be possible. This allows us to choose $n \leq L$ base classifiers. Towards this aim, we train the network structure by using the error function that is appropriate for regression problems.

The algorithm of training/testing the referees is the same as the algorithm for classification trees and rule learners, so we do not repeat it once more. However, there are some points that make this system different from the others that use referees:
a) In this system, there is only one referee structure, i.e. a neural network, which can be seen as a unified version of the independent referees in the other systems.
b) For the case of a tree or a rule learner, a referee for one base classifier can be simpler or more complex than the referee for another base classifier depending on how accurate the base classifiers are. By using a perceptron however, we use referees of the same complexity for all base classifiers. This may be disadvantageous for the system since the areas of expertise of different base classifiers may have different complexities indeed. On the other hand, by using a unified referee structure and training it for all classifiers together, the competence values generated for the classifiers can be made more reliable.
c) In the neural network structure, once the desired output of the system is defined, we do not have to define an extra measurement to be able to sort the classifiers such

as the confidence values or the ratio of correctly classified examples. The competence values are automatically defined as the outputs of the neural network.
d) Like any neural network structure, the success of the network depends on whether the algorithm converges to a good solution or not. The internal parameters of the system, such as the number of hidden units in the hidden layer of a multi-layer network, the initial values of the weights, and the number of the iterations also affect the final output of the system.

In the simplest case, a linear one layer perceptron can be used as a referee. Such a neural network has the advantages of being simple, fast and easy to train, but the mapping may not be possible with a linear function and therefore, it is possible that a linear network is not a reliable referee. Multilayer perceptrons (MLP) are capable of solving nonlinear recognition problems even by using just one hidden layer, and that is another possibility. To define the number of hidden units of a MLP we try different values and choose the one that has the highest accuracy in the validation set X_{val}.

4.3. Systems with Gating

The mixtures of experts (MoE) model, proposed by Jacobs *et al.* [27], consists of a number of base classifiers, or local experts, and their outputs are combined by using the weights generated by a *gating network*. The gating network also sees the same input as the experts and generates a set of coefficients $p_j(x)$. Typically, the coefficient $p_j(x)$ is interpreted as the probability that expert D_j is the most competent expert to label the particular input x, satisfying $\sum_{j=1}^{L} p_j(x) = 1$.

Gating corresponds to all the referees bundled together. Similar to the perceptron based referee structures, there are a number of differences between gating and the referees such as decision trees and rule sets:
a) In the case of gating, the competences of all base classifiers are trained together.
b) In a referee based system, referees may have different complexities but gating uses identical models for all base classifiers.
c) Gating outputs have two function. They both determine which base classifiers are

to be used, and at the same time, correspond to their weights in a weighted voting scheme.

d) A gating system can be trained together with the base classifiers on the same dataset, whereas referees are trained after the base classifiers and on a different dataset.

Although we construct our system by using a structure that is very similar to the MoE model, there are some differences between the two:

a) In the original MoE structure, both the local experts and the gating function are defined as neural networks trained together by using the same input data. In our proposed system, however, we use various types of base classifiers which have already been trained. Therefore, in our MoE structure the gating network is trained on its own and on a separate dataset.

b) In the original MoE structure, the classifiers are assumed to be *local* experts whereas our base classifiers are general and are trained over the whole input space.

The algorithm for the training phase is given in Figure 4.7. Figure 4.8 illustrates the testing phase of the proposed system. The test instance is given as input to all classifiers and the gating network. The output of the base classifiers are also a vector of posterior probabilities as in the previous referee based methods.

1. Train each base classifier D_j separately with X_{tra}.
2. Construct the MoE structure with L experts and a gating network.
   - The gating network uses the original attributes of data and generates L dimensional output in its last layer.
3. Get $P_j(C_i|x)$ values, i.e. output of each classifier D_j, for all instances in X_{val}.
4. Train the gating network by using X_{val} and $P_j(C_i|x)$ to minimize the error.

Figure 4.7. Training phase of the gating based system

In the decision aggregation unit, a weighted sum of posterior probabilities are calculated for each class where weight of decision of classifier D_j is the $p_j(x)$ value that

is generated by the gating network:

$$o_i = \sum_{j=1}^{L} P_j(C_i|x)p_j(x), \ i = 1, \ldots, K \tag{4.3}$$

Then, the class label with the maximum overall posterior value is chosen. The algorithm for the testing phase is given in Figure 4.9.

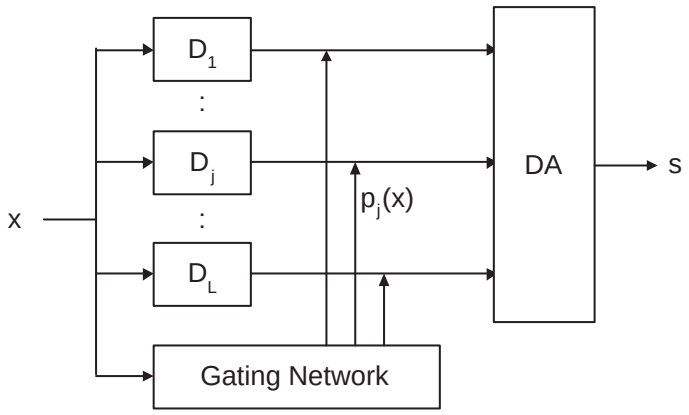

Figure 4.8. Testing phase of the gating based system, adapted from MoE [27]

1. Given x, get $P_j(C_i|x)$ values, i.e. output of each classifier D_j.
2. Given x, get $p_j(x)$ values, i.e. output of the gating network.
3. Calculate the overall class predictions (Equation (4.3)).
4. Predict the class label (Equation (4.2)).

Figure 4.9. Testing phase of the gating based system

To construct the gating network, we use two different network structures, namely linear and multilayer perceptron. For the linear case we have

$$p_j(x) = \frac{\exp\left(v_j x + v_{j0}\right)}{\sum_{l=1}^{L} \exp\left(v_l x + v_{l0}\right)}, j = 1, ..., L \tag{4.4}$$

and for the case of multilayer perceptron, we have

$$p_j(x) = \frac{\exp\left(\sum_h v_{jh} z_h + v_{j0}\right)}{\sum_{l=1}^{L} \exp\left(\sum_h v_{lh} z_h + v_{l0}\right)} \tag{4.5}$$

where z_h denotes hidden units for $h = 1, ..., H$.

5. ANALYSIS OF THE METHODS

5.1. Datasets and Resampling

We use 40 datasets, 38 of which are from the UCI machine learning repository [48] and Delve [49]. Table 5.1 shows the properties of the datasets.

Table 5.1. Properties of the datasets

name	instances	inputs	classes	source
zoo	101	16	7	uci
iris	150	4	3	uci
tae	151	5	3	uci
hepatitis	155	19	2	uci
wine	178	13	3	uci
flags	194	26	8	uci
glass	214	9	6	uci
heart	270	13	2	uci
haberman	306	3	2	uci
artificial	320	5	2	synthetic
flare	323	10	3	uci
ecoli	336	7	8	uci
bupa	345	6	2	uci
ionosphere	351	34	2	uci
dermatology	366	34	6	uci
horse	368	26	2	uci
monks	432	6	2	uci
vote	435	16	2	uci
cylinder	540	45	2	uci
ocr	600	256	10	[50]
balance	625	4	3	uci
australian	690	14	2	uci

Table 5.1. Properties of the datasets (continued)

name	instances	inputs	classes	source
credit	690	15	2	uci
breast	699	9	2	uci
pima	768	8	2	uci
tictactoe	958	9	2	uci
cmc	1473	9	3	uci
yeast	1484	8	10	uci
car	1728	6	4	uci
titanic	2201	3	2	delve
segment	2310	19	7	uci
thyroid	2800	27	4	uci
optdigits	3823	64	10	uci
spambase	4601	57	2	uci
pageblock	5473	10	5	uci
ringnorm	7400	20	2	delve
twonorm	7400	20	2	delve
pendigits	7494	16	10	uci
mushroom	8124	22	2	uci
nursery	12960	8	5	uci

A given dataset is first divided into two parts: 1/3 as the test set and 2/3 as the training set. The training set is then resampled using 5×2 cross-validation (cv) where 2-fold cv is done five times (with stratification) and the roles swapped at each fold to generate ten training and validation folds, tra_i, val_i, $i = 1, \ldots, 10$. tra_i are used to train the base classifiers. val_i are divided into two randomly as *val-*A_i and *val-*B_i, where *val-*A_i are used to train the combiner and *val-*B_i are used for validation. Therefore, we have ten *val-*B_i accuracy results on which we run the 5×2 cv F test to compare accuracies for statistically significant difference. To make the distinction between those two datasets, we name *val-*A_i as X_{val} and *val-*B_i as X_{test} while tra_i are named as X_{tra}.

5.2. Base Classifiers

We use 21 base classifiers that are created by using the following algorithms with different hyperparameters:

- `c45` (C4.5): The standard C4.5 decision tree algorithm.
- `gau` (Gaussian classifier): This is the parametric classifier where each class is represented by a Gaussian and a common covariance matrix is shared by all classes. This is also a linear model.
- `knn` (k-nearest neighbor): With $k = 1, 3, 5, 7$ (`1nn`, `3nn`, `5nn`, `7nn`).
- `ldt`: Linear discriminant tree [51].
- `log` (logistic classifier): This is the linear logistic discriminator trained to minimize cross-entropy by gradient-descent. Discrete features are converted to numeric features by 1-of-n encoding.
- `ml` (multilayer perceptron): There is a single hidden layer where, with d inputs and K classes, the number of hidden units can be d (`ml1`), K (`ml2`), $d+K$ (`ml3`), $(d+K)/2$ (`ml4`), $2(d+K)$ (`ml5`). `ml0` is the linear perceptron.
- `mlt` (multinomial tree): This is the multivariate tree where each decision node is linear [51] and uses all inputs as opposed to the univariate decision tree, C4.5.
- `sm` (selectmax): This simple classifier chooses the class with the highest prior without looking at the input.
- `sv` (support vector machine): LIBSVM 2.82 [52] is used with a linear kernel (`sv1`), radial (Gaussian) kernel (`sv0`), and three polynomial kernels of degree 2, 3, 4. (`sv2`, `sv3`, `sv4`).

5.3. Multiple Classifier Systems

The details of the implemented multiple classifier systems are given in Chapters 3 and 4. The abbreviations we will use are:

- `vote` (voting): All of the classifiers in the ensemble are selected and simple voting is applied (Section 3.2).

- `sbs` (select best statically): The classifier that has the highest accuracy on X_{val} is selected (Section 3.3).
- `sbd` (select best dynamically): The classifier that has the maximum confidence value for the given test instance is selected (Section 3.4).
- `sr` (select randomly): One of the classifiers is selected randomly (Section 3.5).
- `c-in/dp` (competence - independent/dependent): The most competent classifier that has the highest local accuracy (`cin`) or the highest local class accuracy (`cdp`) is selected (Section 3.6).
- `rct` (referee - classification tree): Referees of the classifiers are constructed as classification trees which use the confidence as the competence measurement (Section 4.2.1).
- `rctr` (referee - classification tree - ratio): Referees of the classifiers use the ratio of correctly classified examples as the competence measurement (Section 4.2.1).
- `rrt` (referee - regression tree): Referees are constructed as regression trees with soft labels (Section 4.2.2).
- `rrs` (referee - rule set): Referees are constructed as rule sets with hard labels (Section 4.2.3).
- `r-lp/mlp` (referee - linear/multilayer perceptron): A single referee structure is constructed as a linear (`rlp`) or a multilayer perceptron (`rmlp`), where the number of hidden nodes is set as the optimal one among the five choices (listed in `ml`) (Section 4.2.4).
- `me-l/m` (MoE - linear/multilayer perceptron): Applies weighted voting among the classifiers whose weights are generated by a linear (`mel`) or multilayer (`mem`) gating network, where the number of hidden nodes is set as the optimal one among the five choices (listed in `ml`) (Section 4.3).

Note that given a classifier ensemble with L members, the multiple classifier systems select $n \leq L$ of the base classifiers from the ensemble, except for `vote`, `mel` and `mem` which always use $n = L$ classifiers.

5.4. Case Study: Yeast

Let us assume that we are using the *yeast* dataset and since we do not know which classifier is the best, we decide to use two different base classifiers, one of which is nonparametric and the other is a parametric method. As it is stated, once the individual classifiers are trained with X_{tra}, their success is evaluated and a multiple classifier system is trained by using X_{val}, and tested on X_{test}.

When we use `c45` and `gau` as base classifiers, the accuracy of `gau` is 36.02 while the accuracy of `c45` is 49.00 on X_{val}. If only these values are taken into account, most probably someone will decide to use `c45`. This strategy is very similar to the one used in `sbs`, which also selects the most accurate classifier. On the other hand, simple voting, which is one of the most widely used classifier combination techniques, uses both classifiers while many classifier selection strategies suggest to use one of the classifiers depending on the input.

When simple voting is applied, the accuracy increases to 50.88. The increase in the accuracy indicates that the base classifiers make mistakes on different data instances. In the given example, the increase shows that some of the instances that are misclassified by `c45` are classified correctly with less successful `gau`. The accuracy values on X_{test} also show the same trend, where the accuracy of `gau` is 37.33, the accuracy of `c45` is 50.05 and `vote` achieves 52.46.

It may be thought that selecting the best classifier becomes a more reliable strategy as the number of base classifiers used is increased. A classifier ensemble with $L = 21$ base classifiers seems to be large enough to rely on the best algorithm since the most accurate classifier among 21 different classifiers will most probably classify unseen test instances accurately too. However, by using $L = 21$ base classifiers, the best algorithm achieves an accuracy of 54.22 while the accuracy of a referee based system, `rctr` particularly, can achieve 55.29. Although the difference is not high, it indicates that the single best classifier does not guarantee the highest accuracy value.

We have observed that a common belief about classifiers, that is, the most accurate classifier of an ensemble on validation data will also be the most accurate classifier on any new unseen instance, is true only if the instances that are misclassified by the best classifier is also misclassified by all other classifiers. In other words, if the individual classifiers make errors on different instances of the dataset, then it is possible to increase the accuracy by combining their predictions.

5.4.1. Static, Dynamic and Random Selection

Figure 5.1 helps us to compare three simple strategies: selecting the best classifier statically (`sbs`) and dynamically (`sbd`) as well as random selection (`sr`). The accuracy values are the averages over ten folds drawn with the standard deviation error bars. In all of the methods, the average accuracy increases as the number of selected base classifiers increases, but as more classifiers are selected, the average accuracy stops increasing and even starts to decrease.

When `sbs` is used, determining n in the range of $1 \leq n \leq 6$ shows an increasing trend but it stops at $n = 7$. This experiment indicates that the six best base classifiers complete each other successfully. The seventh one, if added, does not help classify more instances correctly. Although increasing n causes a degradation in the accuracy, as we continue to use more classifiers, the accuracy starts to increase again. We see that by selecting less accurate classifiers, the accuracy value obtained by the six best classifiers can be further improved. This increase in the accuracy can be explained with the diversity of the errors made by the base classifiers. This experiment confirms that an inaccurate classifier may be more successful than the best classifier at classifying the instances in a particular input region.

For small values of n, using `sbd` seems to be disadvantageous in comparison to `sbs`. The main reason is that the base classifiers that return high confidence values, such as `1nn`, are not accurate on *yeast*.

In the simplest case, one may choose one or n of the classifiers randomly. Since

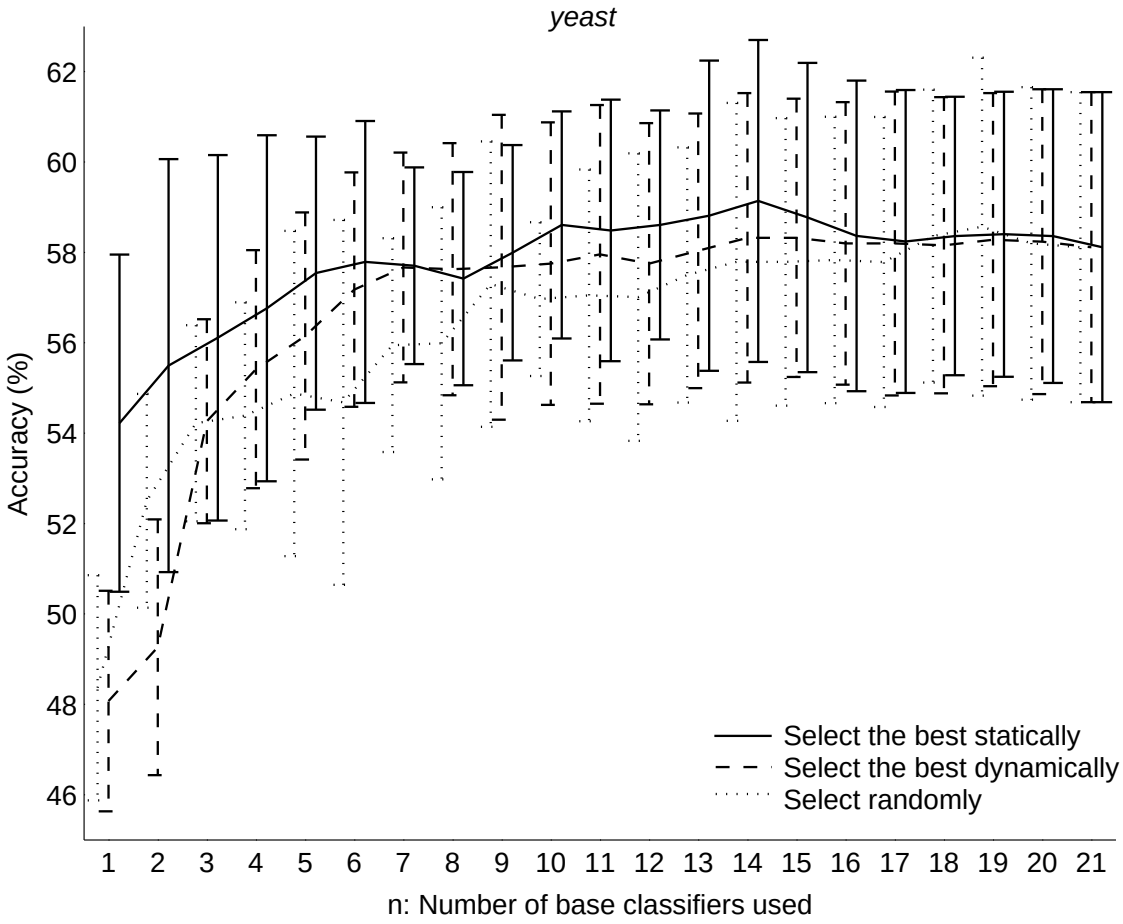

Figure 5.1. Accuracy values of `sbs`, `sbd` and `sr` on *yeast*

we have accurate classifiers in our ensemble more than inaccurate ones, even random selection becomes an efficient technique if n is close to L. Note that none of the methods is allowed to select a base classifier more than once and therefore, as n approaches L, the ensembles used by all three strategies become identical.

5.4.2. The Effect of the Ensemble Size

The performance of the referee based systems depends on both the success of the referees and the accuracy of the selected base classifiers. If the referees can estimate the expected accuracy of the base classifiers and if X_{val} is a good representative of X_{test}, then the overall accuracy will be high. In fact, the overall accuracy will be higher than the accuracies of the individual classifiers if the classifiers make errors in different regions of the input space and these regions are distinguishable by the referees.

In our experiments, we observe that the referee based systems are beneficial especially when the ensemble includes classifiers with different inductive biases, rather than ensembles that include variants of one algorithm with different hyperparameters.

As an example study, we construct a classifier ensemble that includes `c45`, `gau`, `log` and `1nn` as base classifiers. In Figure 5.2, the first four values are the average accuracies of the base classifiers and the others are the average accuracy values of the multiple classifier systems which select only $n = 1$ base classifier. Note that `vote`, `mel` and `mem` use all of the base classifiers. Moreover, the number hidden units in `rmlp` and `mem` structures are optimized on X_{val}.

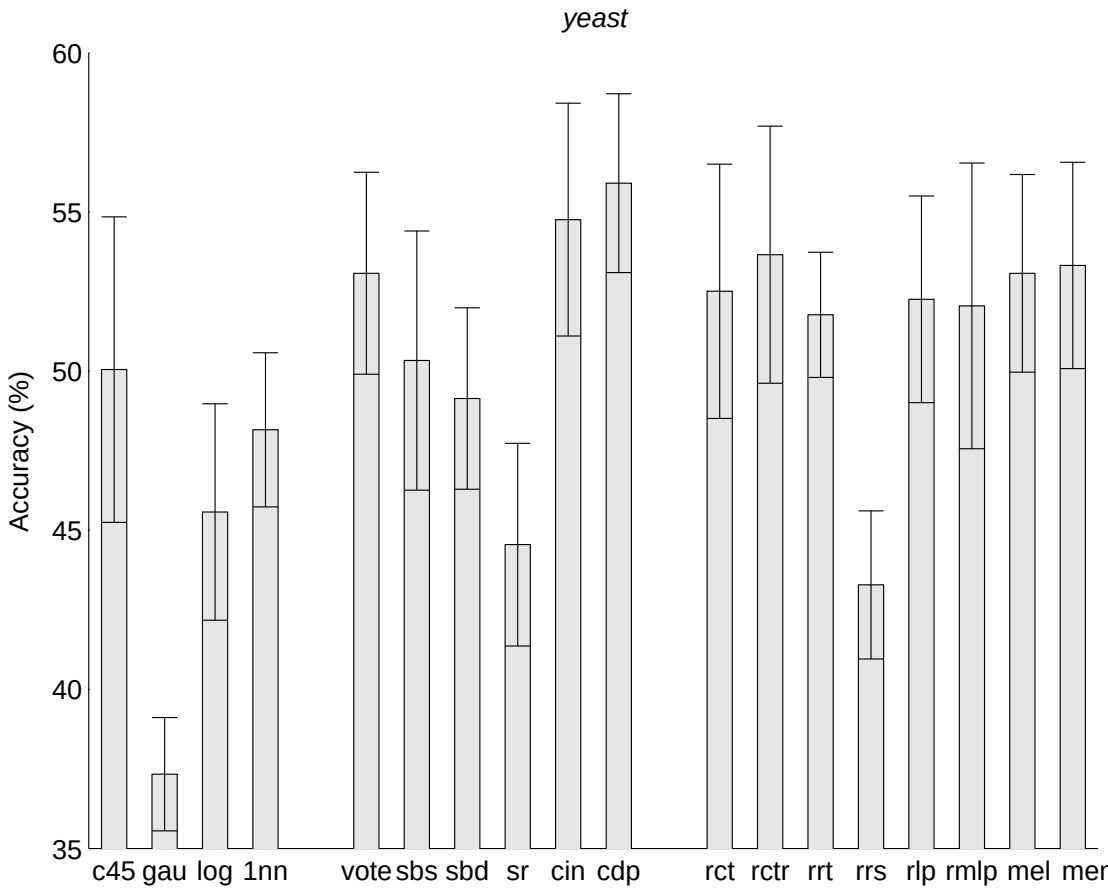

Figure 5.2. Accuracy values of the base classifiers and the multiple classifier systems on *yeast*

As Figure 5.3 shows, different number of hidden units may be found to be optimal in different methods. For *yeast* and the given example ensemble, `rmlp` achieves highest accuracy with $d + K$ hidden units whereas `mem` works well with $2(d + K)$.

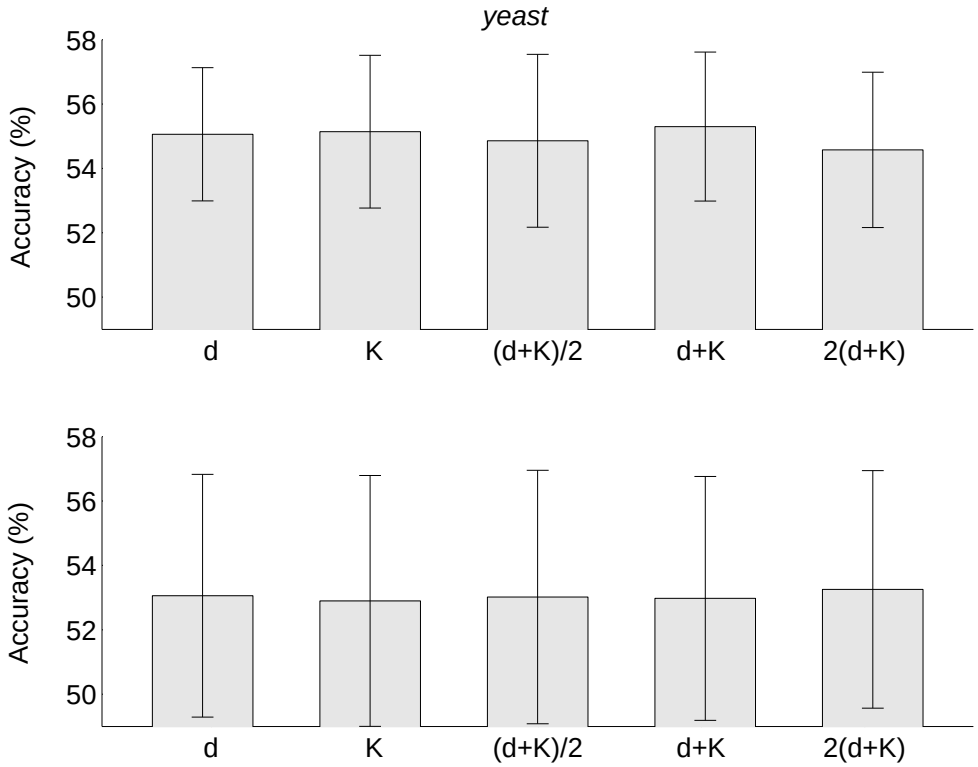

Figure 5.3. Accuracy values of `rmlp` and `mem` with different number of hidden units on *yeast*

When the accuracy values of the base classifiers are considered, it is seen in Figure 5.2 that voting over all base classifiers is beneficial. Note that the accuracy values of `mel` and `vote` are same, and this equality indicates that the weighted voting strategy does not improve simple voting when linear perceptron is used. On the other hand, by using a multilayer perceptron as the gating network, i.e. `mem`, the accuracy is improved slightly.

The accuracy of `sr` is higher than the accuracy of the worst base classifier and much less than the accuracy of the best base classifier, as expected. The accuracy of `sbd` is less than the accuracy of the best base classifier but it is still better than the second best one. It is also seen that `sbd` is less accurate than `sbs`, which may arise from a classifier in the ensemble which produces high posteriors but makes wrong estimations.

As Figure 5.2 illustrates, all referee based systems, except `rrs`, provide higher

accuracy than `sbs` as well as all of the base classifiers. We also see that the system becomes more successful if the referees are constructed by using ratio of the correctly classified instances (static) rather than confidence of the classifiers (dynamic), i.e. `rctr` is better than other rule based referees. As we mentioned, this indicates that there is a classifier which gives higher posterior probabilities for the wrong class. Indeed, `1nn` is the dominant classifier in the ensemble when the posterior probabilities are taken into account. If this classifier were successful on *yeast*, then confidence based systems would yield higher accuracy values. But since this is not the case, using the confidence values becomes disadvantageous.

When `1nn` is eliminated from the ensemble, the accuracy values of the referee based systems that use the confidence values increases slightly, except the one that uses rule sets. Indeed, the reason for the failure of `rrs` is that, during the optimization process in the algorithm, the rule set size is shrank too much since none of the algorithms has good accuracy on *yeast*.

Instead of deleting a base classifier, if we add another successful base classifier to the example ensemble, the accuracy of `sbs` increases. Figure 5.4 illustrates the three cases where the classifier ensemble includes `c45`, `gau`, and `log` initially, and then `1nn` and `sv1` are added. Since `sv1` has higher accuracy than the other base classifiers in the ensemble, it is supposed to increase accuracy in all strategies. After its addition, although they are not as much as with `sbs`, improvements in the accuracy values of the referee based systems also occur. Note that in all cases, the referee based systems, except `rrs`, have higher accuracy than `sbs`.

The figure also shows that when the size of the ensemble is very small, i.e. $L = 3$, selecting one of the classifiers is more beneficial than voting over all of the classifiers. However, after the ensemble is enlarged, i.e. $L = 4$, `vote` shows a dramatic improvement, like `mel` and `mem` do. The experiments also show that `vote`, `mel` and `mem` are more sensitive to the type of the base classifier added to the ensemble with respect to the other selection techniques.

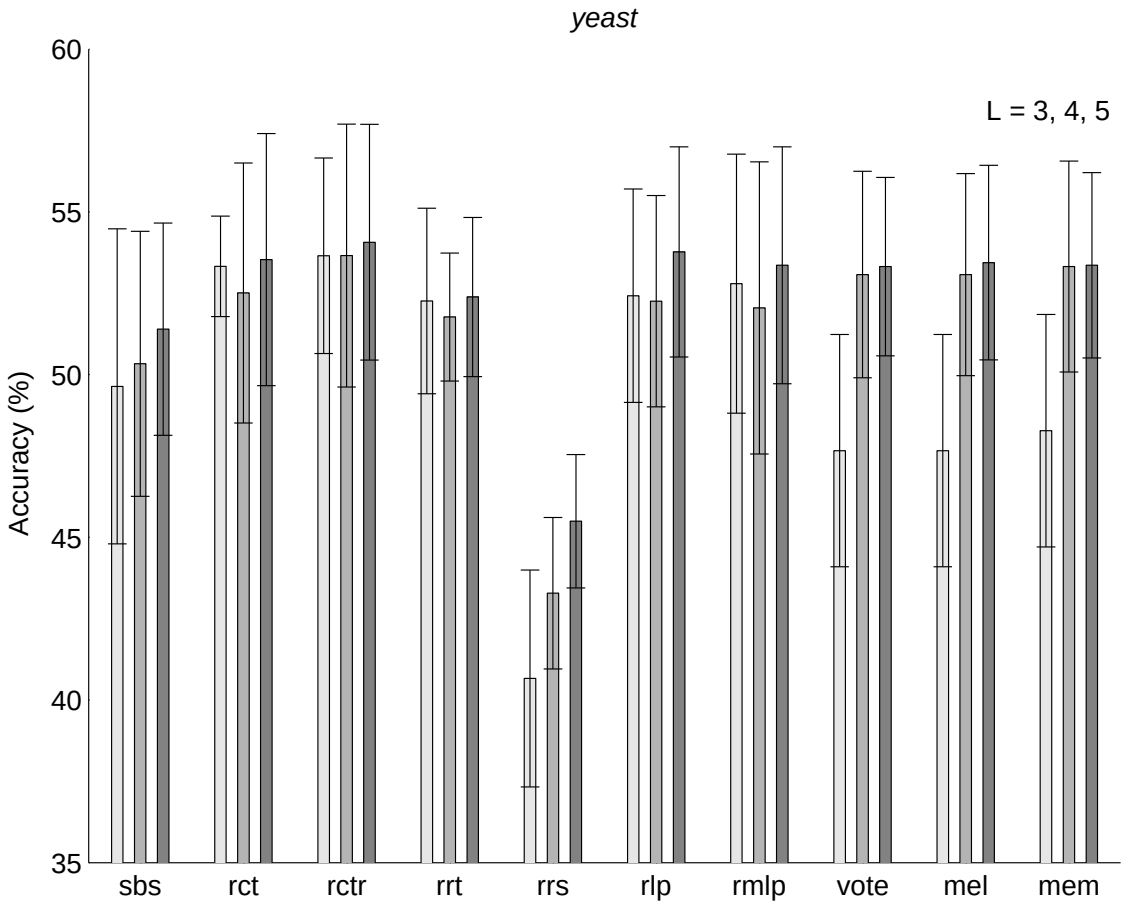

Figure 5.4. Accuracy values of the base classifiers and the multiple classifier systems on *yeast* with different ensembles

5.4.3. The Effect of the Referee on Selecting Experts

Figure 5.5 shows the effect of the referee when the ensemble includes `c45`, `gau`, `log` and `1nn`, and we see that the structure of the referee affects the selected classifier. In particular, we see the percentage of the cases a base classifier is selected and the percentage it is successful. For the classifier systems that use all of the base classifiers, i.e. `vote`, `mel` and `mem`, the weights of the classifiers are taken into account.

First of all, note that on the average the most accurate classifier in the ensemble is `c45` and the least accurate one is `gau`. However, in each fold, a different classifier can be the best one. In Figure 5.5, the four values drawn for `sbs` indicate that `c45` is the best classifier in three of the ten folds whereas `gau` is never the best, `log` is the best in two of the folds and `1nn` is the best in five of the folds. As can be seen, the best

classifier on average, i.e. `c45`, is the most frequently selected classifier in only `rctr` and `mem` methods.

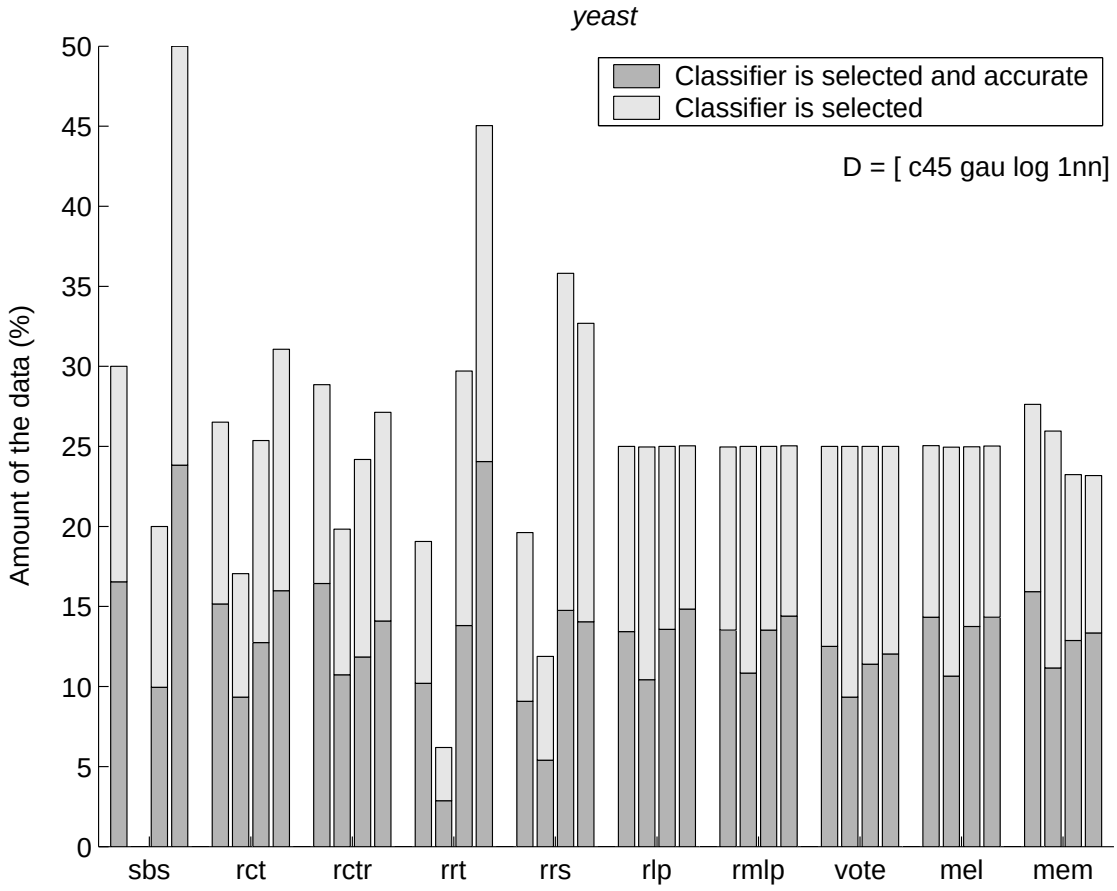

Figure 5.5. Percentage of the usage of the base classifiers on *yeast*

The most frequently selected classifier is `1nn` when the confidence values are taken into account, as is the case in `rrt`. Note that classification tree based referees select `1nn` often if the confidence values are taken into account (`rct`) whereas they select `c45` more often than `1nn` if the ratio of correctly classified examples are considered (`rctr`). Moreover, the rule set based referees generally select the classifiers that have moderate average accuracy, but the amount of the data on which the selected classifier makes error is higher than it is with the other methods. The figure also illustrates that the neural network based referees select classifiers more uniformly than the rule based ones.

5.5. Case Study: Monks

We continue analyzing the multiple classifier systems with different values of n and compare the effect of n. The performance of the individual classifiers can be observed in Figure 5.6. For *monks* dataset, the most accurate classifier on X_{test} is `c45`. Other classifiers can be ordered according to their accuracy values as `sv0`, `mlt`, `ldt`, `ml3`, `3nn`, `1nn`, `ml4`, `ml2`, `ml1`, `7nn`, `5nn`, `gau`, `sv2`, `ml5`, `sv1`, `ml0`, `sv3`, `sv4`, `log`, and `sm`.

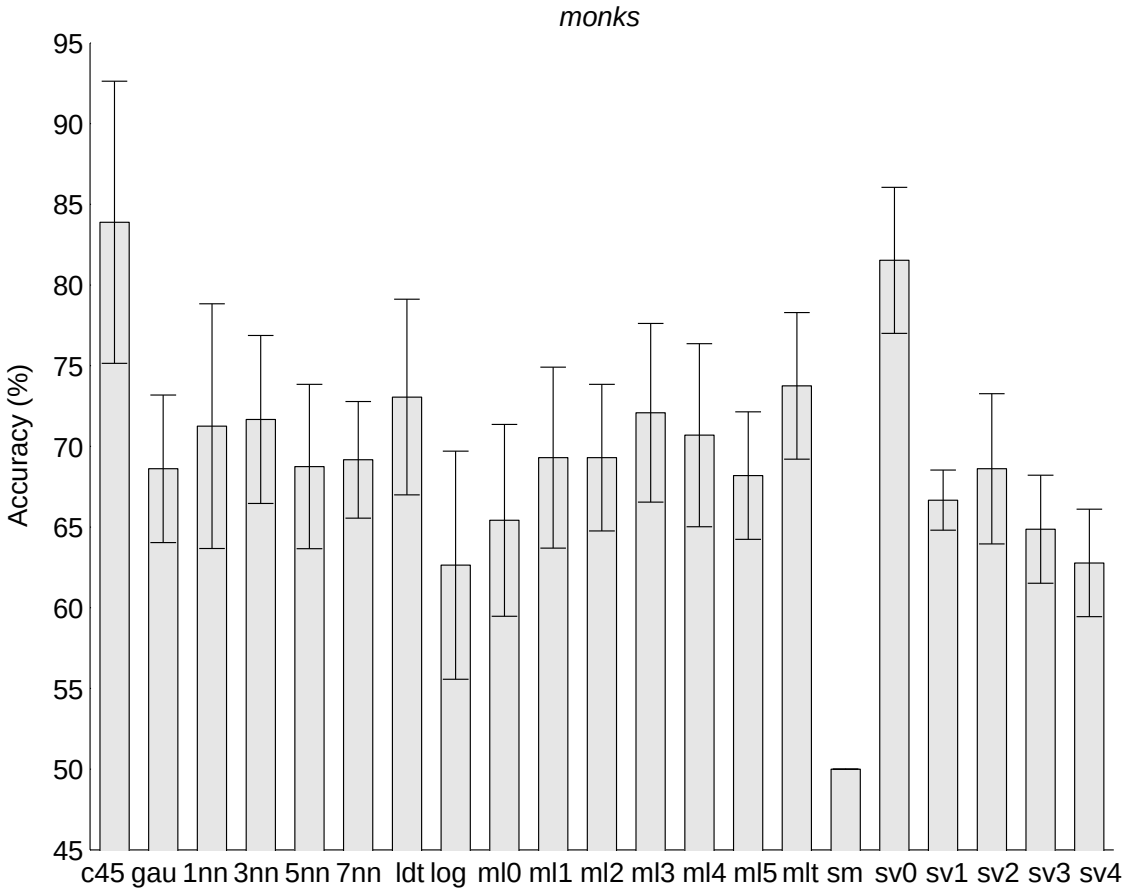

Figure 5.6. Accuracy values of the base classifiers on *monks*

5.5.1. Selecting n Classifiers

As shown in Figure 5.6, `c45` has much higher accuracy than the other base classifiers in the ensemble. Therefore, using only this classifier may be enough to achieve an efficient classifier system. Note that it is also one of the classifiers that produce high confidence values.

As depicted in Figure 5.7 for both `sbs` and `sbd`, the overall accuracy does not increase as the number of selected classifiers increases. However, the random selection strategy, being unaware of the goodness of `c45`, shows an improvement as n increases.

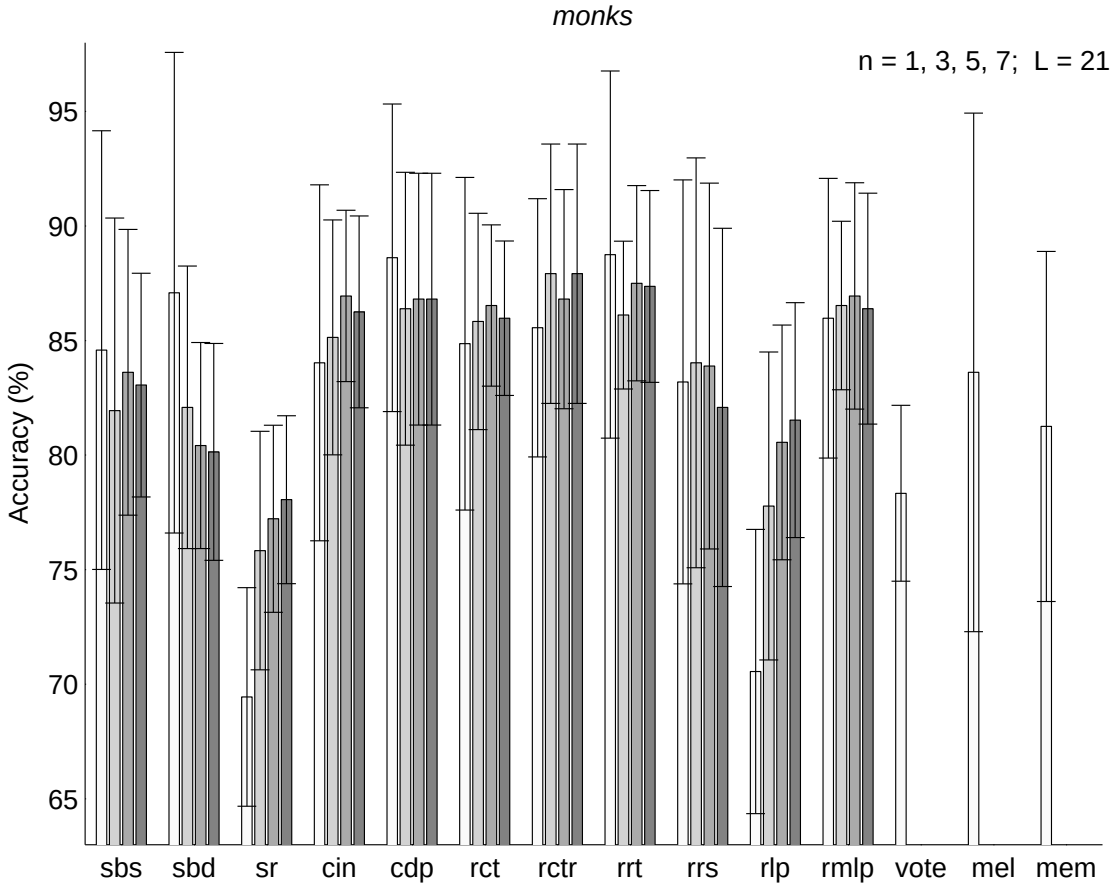

Figure 5.7. Accuracy values of the multiple classifier systems on *monks*

When the change that occurs by increasing n from one to three is analyzed, no common behavior among the referee based systems is observed. However, when $n = 5$, all of them seem to achieve a reasonable accuracy value. In Figure 5.7, it is noticeable that the accuracy of `rrt` is high when $n = 1$, like the dynamic selection method `sbd`. The average accuracy values indicate that `rctr`, `rrt` and `rmlp` are the systems that perform well on *monks* while `rlp` is unsuccessful.

When all of the classifiers are used, i.e. $n = L$, the overall accuracy does not improve in each fold. Therefore, the average accuracy of `vote` is not high. The figure reveals that `mel` provides much higher average accuracy than `vote`, however its performance shows high deviation.

5.5.2. Analysis of the Selected Classifiers

Figure 5.8 illustrates the percentage of the data instances for which the specified base classifier is selected by the rule based referee system. For the case of $n = 1$, `rrt` is more accurate than the others, and the analysis indicates that it selects `c45`, the most accurate base classifier, and `1nn`, the seventh best classifier, mostly. Moreover, it selects `log` frequently although it is the second worst classifier. This indicates that while using an ensemble that includes inaccurate classifiers, using confidence values of the classifiers is beneficial since the confidence value of a classifier warns the system about possible false class predictions and it allows the classifier to be selected only if it is confident.

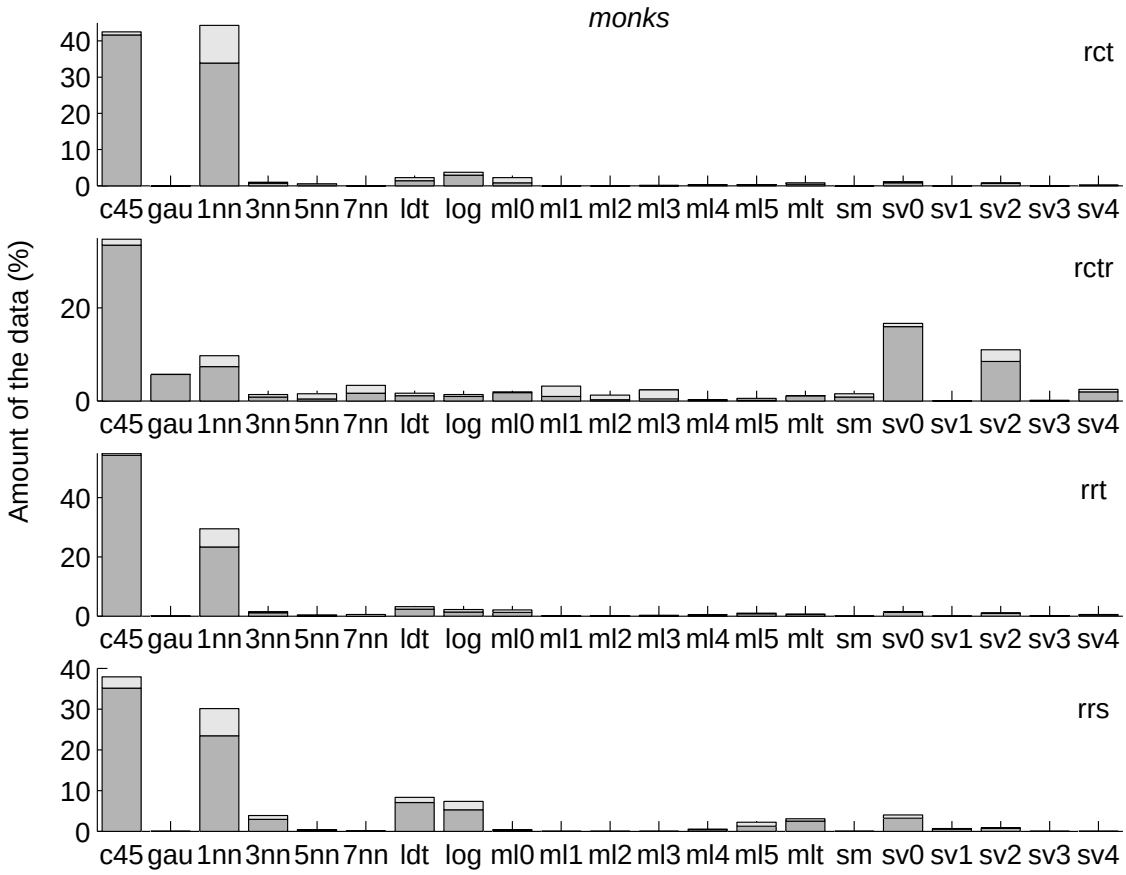

Figure 5.8. Percentage of the usage of the base classifiers on *monks* with the rule based referee systems

Although the accuracy values they provide are similar, `rct` and `rctr` select different classifiers generally. Besides the best base classifier, `rct` uses `1nn` while `rctr` uses

the second best classifier `sv0` and a moderate classifier `sv2`. The `rrs` result reveals that this system also gains benefit from selecting unsuccessful classifiers besides the best ones.

Comparing the referee and gating systems with linear and multilayer perceptrons, we see that the success of `rmlp` results from not only using the best classifier but also using unsuccessful classifiers properly. Indeed, Figure 5.9 shows that even the worst classifier, `sm`, is selected very often by `rmlp` but only for the instances it is accurate.

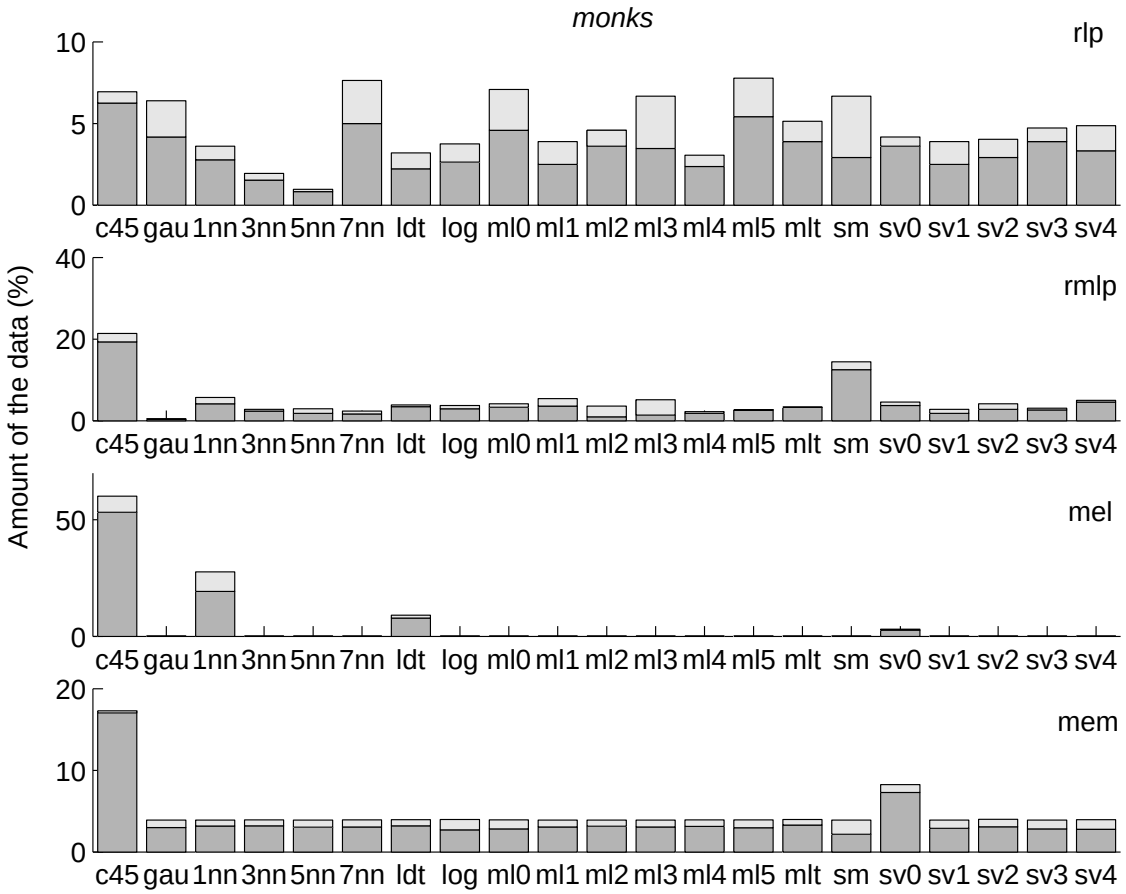

Figure 5.9. Percentage of the usage of the base classifiers on *monks* with the neural network based referee and gating systems

Although `rmlp` has the ability to refer different classifiers cleverly, selections made by `rlp` does not reflect a successful strategy, indicating that linear perceptron is not successful at selecting the best classifier. Furthermore, `mel` selects the best classifiers too often whereas `mem` makes a more uniform selection. Note that while selecting moderate classifiers, `rlp` makes many wrong decisions while the decisions made by `mem`

do not seem to be that worse.

5.6. A Further Analysis of Local Area and Local Competence

With `cin` and `cdp`, we need to define the size of the local area in which the local accuracies of the base classifiers are calculated. Woods *et al.* state [18] that using $k = 10$ neighbors of the given instance seems to result in the best performance. Thinking that the optimal value of this parameter should vary for different datasets, we make some experiments with different values of k on the original algorithm (without defining a limit n). In all experiments, data instances are normalized and Euclidean distance is used to find the nearest neighbors of the particular instance.

Table 5.2 shows the comparison of `cin` and `cdp` with $k = 5$, 10 and 15 on 40 datasets. With different k values, neither `cin` nor `cdp` shows significant difference. For the three cases, although `cin` has significantly higher accuracy values than `cdp` on a few datasets, the number of wins are not significant over 40. Thus, none of these k values is superior to the other.

Table 5.2. Results of `cin` and `cdp` methods with $k = 5$, 10 and 15

	cin-5	cin-10	cin-15	cdp-5	cdp-10	cdp-15
cin-5	0/40/0	0/39/1	0/39/1	5/35/0	2/38/0	1/39/0
cin-10	1/39/0	0/40/0	1/38/1	4/35/1	4/36/0	4/36/0
cin-15	1/39/0	1/38/1	0/40/0	4/36/0	4/36/0	4/36/0
cdp-5	0/35/5	1/35/4	0/36/4	0/40/0	0/38/2	0/38/2
cdp-10	0/38/2	0/36/4	0/36/4	2/38/0	0/40/0	1/37/2
cdp-15	0/39/1	0/36/4	0/36/4	2/38/0	2/37/1	0/40/0

Another possibility is to set $k = N/K$ where N denotes number of training data instances and K is the number of classes. As Table 5.3 shows, this new value provides significantly higher accuracy than $k = 5$ and 10 on 4 and 3 datasets, respectively if `cin` is used whereas we see in Table 5.4 that it may be disadvantageous if `cdp` is used. This parameter becomes too large for datasets where N is large and in the class de-

pendent case, this large number of neighbors defines much larger local regions and the advantage of looking at local accuracy disappears. Another dynamic value defined as k is based on distance calculations. After the matrix of pairwise distances between all training instances is calculated, the maximum values of each row (or each column) of the matrix is summed up. In other words, the sum of maximum pairwise distance for all instances is calculated. Then, the k parameter is defined by dividing the sum by $N.K$. As illustrated in Tables 5.3 and 5.4, although this parameter wins other parameters in some cases, it is not significantly better over 40 datasets.

Table 5.3. Results of `cin` method with different k values

	cin-5	cin-10	cin-15	cin-k/N	cin-dist	cin-3K	cin-5K	cin-10K
cin-k/N	4/35/1	3/36/1	2/36/2	0/40/0	0/40/0	2/37/1	2/38/0	2/37/1
cin-dist	3/36/1	2/38/0	2/37/1	0/40/0	0/40/0	3/37/0	3/37/0	2/38/0
cin-3K	1/39/0	1/38/1	1/37/2	1/37/2	0/37/3	0/40/0	1/37/2	0/39/1
cin-5K	1/39/0	0/39/1	1/39/0	0/38/2	0/37/3	2/37/1	0/40/0	0/39/1
cin-10K	1/38/1	1/38/1	2/37/1	1/37/2	0/38/2	1/39/0	1/39/0	0/40/0

Table 5.4. Results of `cdp` method with different k values

	cdp-5	cdp-10	cdp-15	cdp-k/N	cdp-dist	cdp-3K	cdp-5K	cdp-10K
cdp-k/N	3/34/3	2/33/5	2/35/3	0/40/0	2/36/2	3/33/4	2/35/3	2/38/0
cdp-dist	2/33/5	2/32/6	2/33/5	2/36/2	0/40/0	2/33/5	2/34/4	3/34/3
cdp-3K	2/38/0	1/38/1	2/37/1	4/33/3	5/33/2	0/40/0	0/39/1	2/36/2
cdp-5K	2/36/2	1/37/2	1/38/1	3/35/2	4/34/2	1/39/0	0/40/0	1/37/2
cdp-10K	2/34/4	3/35/2	1/36/3	0/38/2	3/34/3	2/36/2	2/37/1	0/40/0

Some other values used are $k = 3K$, $5K$ and $10K$. These values never become too large values as the previous ones, but the number of wins and losses indicate that they are not significantly different from $k = 10$. Therefore, we follow Woods *et al.*'s suggestion and fix it as $k = 10$.

5.7. A Further Comparison of the Rule Based Referees

The three referee structures, i.e. classification trees, regression trees and rule sets, can also be compared in terms of the construction or the storage cost. We compare the number of conditions created for the referee of each base classifier by taking average condition counts over 40 datasets. It is easily seen in Figure 5.10 that for all base classifiers, the number of conditions created for the classification tree based referees is much less than the number of conditions created for the regression tree based referees. This is due to the impurity condition that is used to stop the creation of nodes in decision trees.

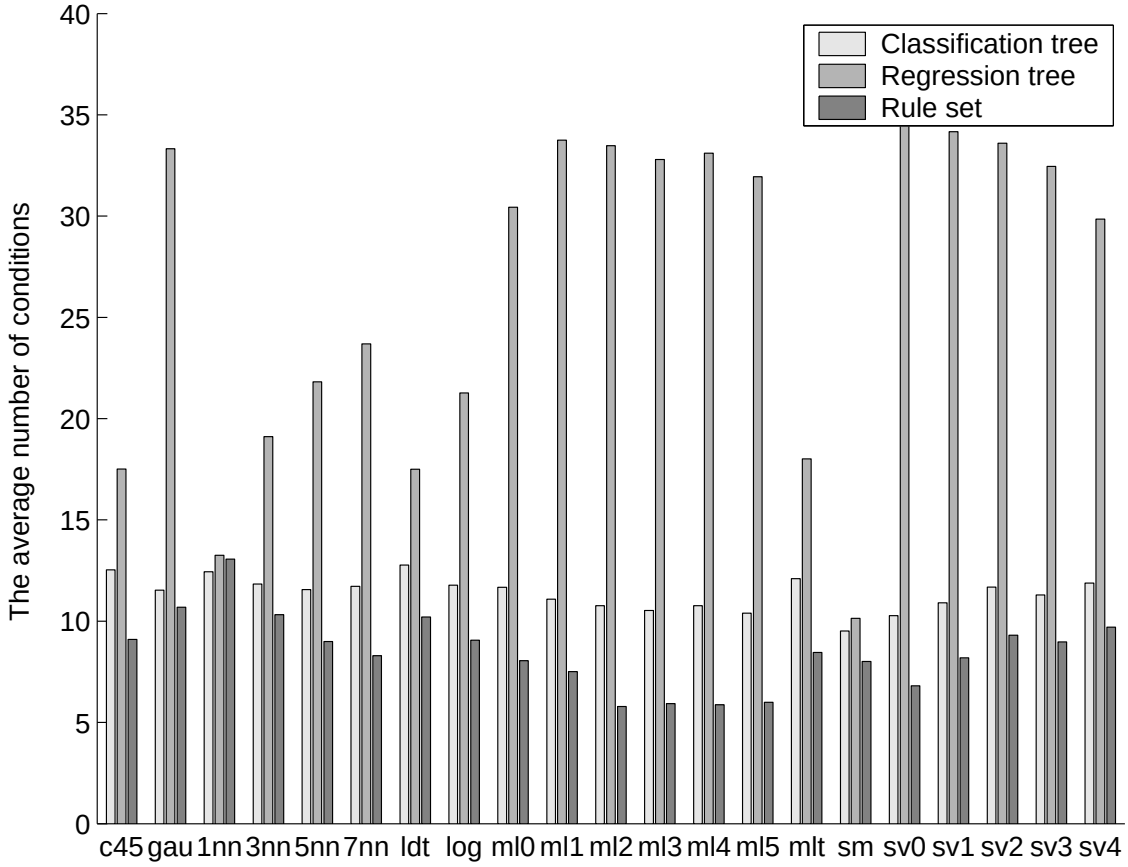

Figure 5.10. Average number of nodes created for the three referee types

The number of conditions is close to each other for both decision trees only in two cases; when a referee is created for `1nn` classifier or `sm` classifier. When the regression tree based referees are used, the referee of `1nn` classifier is not large since the posterior probability values generated by this classifier do not vary too much for different data

instances, and the referee of `sm` classifier is not large because it uses the same prior class probability values for all instances of the same class. Figure 5.10 also illustrates that the rules can be constructed by using less number of conditions when rule sets are used.

6. EXPERIMENTS AND RESULTS

6.1. Comparison Metrics

In all methods compared in this chapter, the aim is to select the n most competent classifier(s) from a classifier ensemble that has L members. The main difference between the methods is their referee units that calculate the competence values.

To be able to have general conclusions, we use 40 datasets and a classifier ensemble that has 21 trained base classifiers. We focus on *win*/*tie*/*loss* matrices as well as the average accuracy values of the systems on the test sets. The matrices indicate the count of significant differences in the accuracy values. We perform 5×2 cross validation and use 5×2 cv F test to check for a statistically significant difference (Appendix A). We also analyze the number of *win*s and *loss*es by using statistical sign test to see whether one algorithm is better or worse than the other frequently and indicate significant results by using italic font.

In the following sections, the term `xxx-n` is used to denote that the method `xxx` selects n classifiers from the ensemble. Due to their competence calculation strategies, the methods are capable of selecting any number of the classifiers as long as $n \leq L$, but we are more interested when D^* contains not more than one thirds of the base classifiers.

6.2. Experiments with the Existing Methods

6.2.1. Selecting the Best Classifier Statically

Results of `sbs` as n is varied are given in Tables 6.1 and 6.2. If we check the number of significant differences in Table 6.1, we see that the selection system with $n = 5$ or 7 wins over $n = 1$; in other words, `sbs` is more accurate when $\frac{L}{4} \leq n \leq \frac{L}{3}$. As a result, we can say that the accuracy of the combination of *a few* best classifiers

is better than the accuracy of the *single* best classifier, and is also better than or as good as `vote` when all base classifiers are used.

Table 6.1. Results of `sbs` method with small n values

	sbs-1	sbs-3	sbs-5	sbs-7	sbs-9	vote
sbs-1	0/40/0	0/40/0	0/38/2	1/35/4	2/37/1	3/37/0
sbs-3	0/40/0	0/40/0	0/39/1	1/39/0	2/38/0	3/37/0
sbs-5	2/38/0	1/39/0	0/40/0	1/39/0	1/39/0	3/37/0
sbs-7	4/35/1	0/39/1	0/39/1	0/40/0	1/39/0	2/37/1
sbs-9	1/37/2	0/38/2	0/39/1	0/39/1	0/40/0	3/37/0
vote	0/37/3	0/37/3	0/37/3	1/37/2	0/37/3	0/40/0

Increasing n more enforces the system to select some classifiers which do not have good accuracy values and the overall accuracy starts not to increase any further and may even decrease. As n becomes very close to L, the set of the selected classifiers starts to include different types of classifiers more often, and in some datasets, voting among them provides an increase in the accuracy again, or at least, the degradation in the accuracy stops. However, as seen in Table 6.2, these increases are not significant and small values of n are still more successful than the large ones.

Table 6.2. Results of `sbs` method with large n values

	sbs-1	sbs-3	sbs-5	sbs-7	sbs-9	vote
sbs-11	1/37/2	0/39/1	0/39/1	0/39/1	0/40/0	2/38/0
sbs-13	2/36/2	0/38/2	0/38/2	0/38/2	1/38/1	1/38/1
sbs-15	0/38/2	0/38/2	0/38/2	0/37/3	0/39/1	3/37/0
sbs-17	0/38/2	0/38/2	0/38/2	0/38/2	0/39/1	0/40/0
sbs-19	1/36/3	0/37/3	0/36/4	1/35/4	0/38/2	0/40/0
vote	0/37/3	0/37/3	0/37/3	1/37/2	0/37/3	0/40/0

Besides the pairwise comparisons given with the matrices, the general behavior of the system can be seen in Figure 6.1 where the number of wins and losses of a system

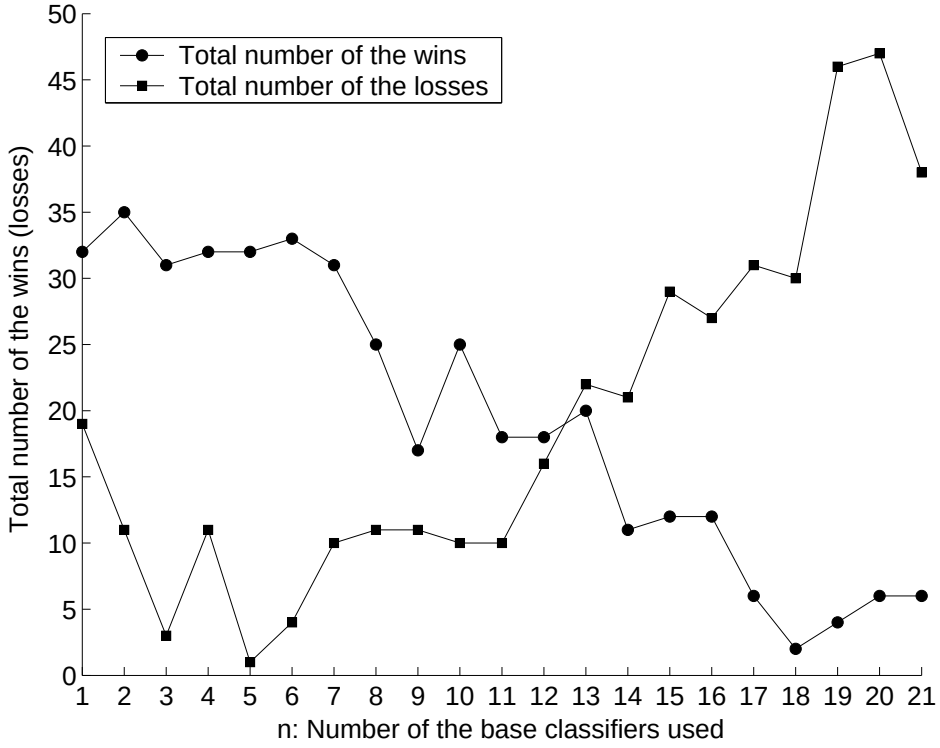

Figure 6.1. Total number of the *wins*/*losses* of `sbs` method over $L = 21$ base classifiers

with particular n are summed up over all other systems with $n = 1..L$. The bow-like graphic also verifies that a system with smaller n value usually wins over the other systems and loses rarely. We can also conclude that this system is affected badly if the classifier ensemble includes inaccurate classifiers.

6.2.2. Selecting the Best Classifier Dynamically

The main reason for using `sbd` is to enforce the system to select a classifier from the ensemble by taking the given test instance into account, i.e. to make a data dependent selection dynamically in the classification phase. The referee unit of this system defines the competence of a classifier as the maximum class posterior probability estimated for the given instance.

Our analysis of the average accuracy values show that when only the best classifier is selected, i.e. $n = 1$, the static selection strategy is more accurate than the dynamic one on 33 datasets. Statistically, the dynamic method is significantly worse than the

static one in more than the quarter of the experiments as seen in Table 6.3. The analysis reveals that the major reason for this failure is the posterior values that mislead the referee unit. Especially, the classifiers such as `c45` and `1nn` pretend to have the maximum confidence value too often and cause misclassification if they are not actually accurate. On the other hand, if those classifiers are accurate on the given dataset, the dynamic method performs well as seen in the case studies given in the previous chapter.

Table 6.3. Results of `sbd` and `sbs` methods

	sbd-1	sbd-3	sbd-5	sbd-7	sbs-1	sbs-3	sbs-5	sbs-7
sbd-1	0/40/0	*0/28/12*	*0/25/15*	*0/24/16*	*0/28/12*	*0/26/14*	*0/24/16*	*0/25/15*
sbd-3	*12/28/0*	0/40/0	0/35/5	0/36/4	0/35/5	*0/34/6*	*0/33/7*	*0/33/7*
sbd-5	*15/25/0*	5/35/0	0/40/0	1/35/4	0/35/5	0/35/5	0/36/4	0/35/5
sbd-7	*16/24/0*	4/36/0	4/35/1	0/40/0	0/36/4	1/36/3	0/37/3	0/35/5

As n is increased, the success of `sbd` increases and the accuracy values of `sbd` and `sbs` become closer. For the case of $n = 5$, `sbs` is significantly more accurate than `sbd` on just a few of the datasets, for example. It may be thought that those two methods start to select almost the same classifiers as n increases and this makes their accuracy values similar. However, our analysis shows that the selected classifiers are very different in `sbd` and `sbs`, and `sbd` selects only those classifiers which produce posterior probability values very close to 1. The reason for the increase in the success of `sbd` is just the benefit we gain by combining the decisions of more than one confident classifiers.

6.2.3. Selecting Classifiers Randomly

The referee unit of `sr` is composed of just a random number generator that does not take into consideration neither X_{val} nor the given test instance. Therefore, as our analysis on many datasets verifies, this system causes lower accuracy values than the other systems we used.

As expected, for small values of n, `sr` is unsuccessful on all datasets because

the probability of selecting an unsuccessful classifier is the same as the probability of selecting a successful one. However, the differences between the accuracy values obtained from the random selection and the other systems are not statistically significant all the time. The major reason is that in our classifier ensemble we have base classifiers each of which is already trained and optimized to be accurate. As it was experimentally illustrated with the case studies in the previous chapter, random selection is much worse than other systems if the classifier ensemble includes base classifiers with different accuracy scores.

As can be seen in Table 6.4, the accuracy of `sr` increases as n increases. Indeed, the system is most beneficial for large n. Using $n = 7$ classifiers provides significantly higher accuracy than $n = 1$ for almost half of the datasets used, for example. Note that the system is not allowed to select a classifier more than once. Therefore, as the system selects more classifiers, the chance of selecting good classifiers increases. When more than half of the classifiers are selected randomly among $L = 21$ base classifiers, the accuracy of the random selection system becomes almost the same as the accuracy of the voting method.

Table 6.4. Results of `sr` and `sbs` methods

	sr-1	sr-3	sr-5	sr-7	sbs-1	sbs-3	sbs-5	sbs-7
sr-1	0/40/0	*0/27/13*	*0/23/17*	*0/21/19*	*0/23/17*	*0/20/20*	*0/18/22*	*0/18/22*
sr-3	*13/27/0*	0/40/0	*0/34/6*	0/36/4	*0/31/9*	*0/30/10*	*0/26/14*	*0/25/15*
sr-5	*17/23/0*	*6/34/0*	0/40/0	0/37/3	*0/33/7*	*0/33/7*	*0/31/9*	*0/31/9*
sr-7	*19/21/0*	4/36/0	3/37/0	0/40/0	0/35/5	0/35/5	*0/34/6*	*0/33/7*

6.2.4. Selecting Classifiers Using Local Competence

The referee unit of this system takes into account both X_{val} and the given test instance. It calculates the local accuracy of a base classifier on X_{val} by defining a local area with the neighbors of the given instance. In the class dependent case, the referee unit checks the class predictions of the classifiers and determines the neighbors whose

estimated class labels are same with the estimated label of the given instance.

Table 6.5 illustrates the comparison between two competence calculation techniques, namely class independent (`cin`) and class dependent (`cdp`) cases. It is again observed that larger values of n are better. However, the benefit is not too high in the class dependent case. The reason can be explained as follows: With `cin`, the competence of each classifier is calculated in the same local region whereas with `cdp`, each classifier uses a local region according to its own estimation, and therefore, the competences are calculated in a larger region of the input space. In our experiments with different sizes of local regions, we have already seen in Section 5.6 that using large areas is disadvantageous for this system.

Table 6.5. Results of `cin` and `cdp` methods

	cin-1	cin-3	cin-5	cin-7	cdp-1	cdp-3	cdp-5	cdp-7
cin-1	0/40/0	*1/31/8*	1/32/7	1/32/7	0/37/3	1/34/5	1/34/5	1/34/5
cin-3	*8/31/1*	0/40/0	1/38/1	1/37/2	4/36/0	1/38/1	1/38/1	1/38/1
cin-5	7/32/1	1/38/1	0/40/0	0/38/2	5/35/0	1/39/0	1/39/0	1/39/0
cin-7	7/32/1	2/37/1	2/38/0	0/40/0	*6/34/0*	3/37/0	1/39/0	1/39/0
cdp-1	3/37/0	0/36/4	0/35/5	*0/34/6*	0/40/0	1/36/3	1/36/3	2/35/3
cdp-3	5/34/1	1/38/1	0/39/1	0/37/3	3/36/1	0/40/0	0/38/2	0/38/2
cdp-5	5/34/1	1/38/1	0/39/1	0/39/1	3/36/1	2/38/0	0/40/0	0/39/1
cdp-7	5/34/1	1/38/1	0/39/1	0/39/1	3/35/2	2/38/0	1/39/0	0/40/0

6.3. Experiments with the Proposed Methods

6.3.1. Rule Based Referee Structures

In `rct`, `rctr`, `rrs` and `rrt`, we have independent referees for each classifier and any referee gives information about the competence of the corresponding base classifier only. The referees in these systems use both X_{val} and the given test instance, like the local competence based systems do.

In all systems, we easily notice that a slight increase in n causes significant increases in the accuracy. When classification trees are used as referees, the increase in the value of n from 1 to 3 creates an increase in 35 datasets used, 8 of which are significant as shown in Table 6.6. If the ratio of correctly classified examples is used instead of the confidence measurement to define the competences, varying n does not affect the accuracy as much. Indeed, for small values of n, `rctr` is superior to `rct` anyway. Moreover, as also shown in the comparison between `sbs` and `sbd`, dynamic methods are affected with n more than static ones.

Table 6.6. Results of `rct` (using the confidence values) and `rctr` (using the ratio of the correctly classified examples) methods

	rct-1	rct-3	rct-5	rct-7	rctr-1	rctr-3	rctr-5	rctr-7
rct-1	0/40/0	*0/32/8*	*0/29/11*	*0/29/11*	0/35/5	*0/32/8*	*0/30/10*	*0/32/8*
rct-3	*8/32/0*	0/40/0	0/37/3	0/37/3	3/36/1	0/38/2	0/37/3	0/38/2
rct-5	*11/29/0*	3/37/0	0/40/0	1/39/0	3/37/0	0/40/0	1/38/1	0/40/0
rct-7	*11/29/0*	3/37/0	0/39/1	0/40/0	3/37/0	0/40/0	0/40/0	0/40/0
rctr-1	5/35/0	1/36/3	0/37/3	0/37/3	0/40/0	0/39/1	0/37/3	0/39/1
rctr-3	*8/32/0*	2/38/0	0/40/0	0/40/0	1/39/0	0/40/0	0/40/0	0/40/0
rctr-5	*10/30/0*	3/37/0	1/38/1	0/40/0	3/37/0	0/40/0	0/40/0	0/40/0
rctr-7	*8/32/0*	2/38/0	0/40/0	0/40/0	1/39/0	0/40/0	0/40/0	0/40/0

The referees based on rule sets have a more stable behavior in terms of the amounts of increases in the accuracy values. The system with $n = 3$ improves the accuracy of almost every dataset, where the improvement is significant in 6 datasets as shown in Table 6.7.

When regression trees are used, the differences in the accuracy values show more variety; in some datasets, the increase is very small whereas in some others it is large. Increasing n from 1 to 3 improves the accuracy in 33 datasets, and as Table 6.8 indicates, 6 of these improvements are significant. The main reason why the systems with $n = 1$ have significant number of losses is that using just one refereed classifier is more risky than using a few classifiers that are refereed independently from each other.

Table 6.7. Results of `rrs` method

	rrs-1	rrs-3	rrs-5	rrs-7
rrs-1	0/40/0	*0/34/6*	*0/32/8*	*0/32/8*
rrs-3	*6/34/0*	0/40/0	0/39/1	1/37/2
rrs-5	*8/32/0*	1/39/0	0/40/0	0/39/1
rrs-7	*8/32/0*	2/37/1	1/39/0	0/40/0

Furthermore, as is shown in all tables, the number of significant differences between the cases $n = 3$ and 5, or $n = 5$ and 7 is much less.

Table 6.8. Results of `rrt` method

	rrt-1	rrt-3	rrt-5	rrt-7
rrt-1	0/40/0	*0/34/6*	*1/31/8*	*1/30/9*
rrt-3	*6/34/0*	0/40/0	1/33/6	2/31/7
rrt-5	*8/31/1*	6/33/1	0/40/0	1/38/1
rrt-7	*9/30/1*	7/31/2	1/38/1	0/40/0

When the systems with different referee structures are compared with each other, it is seen that the systems become more similar to each other as n is increased. As can be seen in Tables 6.9 and 6.10, the number of wins among all the systems with $n = 5$ is lower than the number of wins among the systems with $n = 1$. In general, `rrs` and `rctr` select classifiers from the ensemble more uniformly than `rct` and `rrt` do. But as n is increased, the systems start to select the same classifiers more often and this is the reason of the similarity between their accuracy values. We can conclude that the value of n can be set as $3 \leq n \leq 5$ to get high accuracy values without increasing the cost of classification too much. Note that the values in the proposed range is less than the quarter of L which indicates a significant decrease in complexity when compared with `vote` which uses all L.

Another important part of the analysis is the comparison between performances of different referee structures for a predefined n value. Almost for the half of the 40

datasets, the referees based on classification trees provide higher average accuracy than the referees based on regression trees for $n = 1$, and on the other half, the accuracy is less. Therefore, it is difficult to say that one is better than the other. However, Table 6.9 illustrates that `rct` has significantly higher accuracy than `rrt` in 2 datasets when $n = 1$ and in 3 datasets when $n = 3$ while there is no dataset where it is significantly worse. The results show that `rct` and `rctr` should be preferred to `rrt`. This conclusion also indicates that the confidence values of classifiers may not be reliable enough and they may not measure the competence of a classifier well. Similarly, the results of our experiments on `rct` and `rctr` indicate that using the ratio of positively classified examples is a better strategy than using the confidence level as measurement of competence. When $n = 1$, using the ratio instead of the confidence improves accuracy significantly on 5 datasets, for example. However, Table 6.10 illustrates that as n is increased, `rct` does not lose.

Table 6.9. Results of `rct`, `rctr`, `rrs` and `rrt` methods with $n = 1$ and $n = 3$

$n = 1$	rct	rctr	rrs	rrt	$n = 3$	rct	rctr	rrs	rrt
rct	0/40/0	0/35/5	3/34/3	2/38/0	rct	0/40/0	0/38/2	0/40/0	3/37/0
rctr	5/35/0	0/40/0	*6/34/0*	3/37/0	rctr	2/38/0	0/40/0	1/39/0	2/37/1
rrs	3/34/3	*0/34/6*	0/40/0	3/33/4	rrs	0/40/0	0/39/1	0/40/0	2/37/1
rrt	0/38/2	0/37/3	4/33/3	0/40/0	rrt	0/37/3	1/37/2	1/37/2	0/40/0

Table 6.10. Results of `rct`, `rctr`, `rrs` and `rrt` methods with $n = 5$ and $n = 7$

$n = 5$	rct	rctr	rrs	rrt	$n = 7$	rct	rctr	rrs	rrt
rct	0/40/0	1/38/1	1/39/0	1/39/0	rct	0/40/0	0/40/0	3/37/0	2/37/1
rctr	1/38/1	0/40/0	2/37/1	1/39/0	rctr	0/40/0	0/40/0	1/38/1	1/39/0
rrs	0/39/1	1/37/2	0/40/0	1/39/0	rrs	0/37/3	1/38/1	0/40/0	1/36/3
rrt	0/39/1	0/39/1	0/39/1	0/40/0	rrt	1/37/2	0/39/1	3/36/1	0/40/0

For the comparison of `rct` and `rrs`, it is difficult to make a conclusion based on

win/tie/loss values. The two systems are not significantly different from each other if the number of wins and losses are taken into account, but `rrs` is worse in the case $n = 7$. Our analysis of average accuracies show that the rule sets are more accurate than the classification trees on half of the datasets while the reverse is true on the remaining datasets. Moreover, `rctr` and `rrs` are not significantly different when $n > 1$ although `rctr` is better than `rrs` when $n = 1$.

6.3.2. Perceptron Based Referee Structures

The comparison between linear perceptron (LP) and multilayer perceptron (MLP) based referee structures is illustrated in Table 6.11. As explained in Section 4.2.1, stating whether a classifier can accurately classify a test instance or not and predicting the competence of the classifier are hard tasks. Therefore, a LP may not be successful. As seen in the table, when $n = 1$, significantly higher accuracy values can be obtained by constructing the referee as MLP instead of LP, which indicates that the problem is not linearly separable.

Table 6.11. Results of `rlp` and `rmlp` methods

	rlp-1	rlp-3	rlp-5	rlp-7	rmlp-1	rmlp-3	rmlp-5	rmlp-7
rlp-1	0/40/0	0/36/4	*0/33/7*	*0/31/9*	0/38/2	0/36/4	*0/34/6*	*0/33/7*
rlp-3	4/36/0	0/40/0	0/40/0	0/38/2	2/37/1	1/38/1	0/39/1	0/39/1
rlp-5	*7/33/0*	0/40/0	0/40/0	0/39/1	2/38/0	1/39/0	0/40/0	0/39/1
rlp-7	*9/31/0*	2/38/0	1/39/0	0/40/0	3/37/0	3/37/0	0/40/0	1/39/0
rmlp-1	2/38/0	1/37/2	0/38/2	0/37/3	0/40/0	0/40/0	0/37/3	0/36/4
rmlp-3	4/36/0	1/38/1	0/39/1	0/37/3	0/40/0	0/40/0	0/38/2	0/37/3
rmlp-5	*6/34/0*	1/39/0	0/40/0	0/40/0	3/37/0	2/38/0	0/40/0	0/40/0
rmlp-7	*7/33/0*	1/39/0	1/39/0	0/39/1	4/36/0	3/37/0	0/40/0	0/40/0

It is interesting to see that as n increases, success of `rlp` increases dramatically. This increase indicates that as a referee structure a linear network is capable of distinguishing successful classifiers from unsuccessful ones, but is not able to define the most successful one.

6.3.3. Systems with Gating

This system is also constructed with neural networks where the structure that defines competence of the classifiers (gating network) is trained with X_{val} and gives output based on the given test instance. However, this method has a different classifier selection and decision fusion strategy than the referee based systems. In referee based systems, the output of the referees is used to select the most competent classifiers and simple voting is used to combine the decisions of the selected classifiers. Although the competence of the classifiers is evaluated in terms of the weights like the network based referees, this system uses the weights also in the decision aggregation.

The major unit that affects the accuracy is the gating network. We use two kinds of network structures for gating, a linear perceptron (`mel`) and a multilayer perceptron (`mem`). Note that the number of hidden units in `mem` is different for each dataset since optimal one among the five choices is used as mentioned before. As seen in the previous section, LP is not efficient in comparison to MLP as a referee structure. However, Table 6.12 shows that LP has better performance than MLP when used for gating. Note that `mel` is more accurate than `vote` on 3 datasets. The difference of LP and MLP structures can be explained in terms of underfitting versus overfitting. In a referee based system, LP underfits the problem since defining the accurate areas of a classifier is a hard problem and LP is not capable of explaining complex problems. On the other hand, the gating network is just a unit of a complex composite system and using a complex structure such as a MLP may cause an overfit.

Table 6.12. Results of `mel` and `mem` methods

	mel	mem	vote
mel	0/40/0	4/35/1	3/36/1
mem	1/35/4	0/40/0	1/38/1

6.4. Comparison of Different Methods

We have already compared similar structures with each other. We now will make a further analysis by comparing different methods. At first, to be able to determine their optimal hyperparameters as well as the parameter n, we look at the results of the methods obtained on X_{val}, and to make a more fair comparison we take only the results for $n \leq 7$ into account. When any two methods are found not to be significantly different, the less costly one (having lower n) is preferred. The test results on X_{val} explained below are given in Appendix B.

The results of `sbs` on X_{val} show that `sbs-7` is significantly worse than `sbs-1`, `sbs-3`, and `sbs-5` on a few datasets. Since `sbs-1` wins `sbs-5` and `sbs-7` and is indifferent with `sbs-3`, we use `sbs-1` in the further comparison. The results of `sbd` indicates that `sbd-7` wins over other variants. The random selection method shows the same behavior and we also pick `sr-7`.

The number of wins increases when n increases in both `cin` and `cdp`. However, for the referee based systems, the optimal values of n is found to be different. While the classification tree based referees do not improve as n is changed, `rrs` and `rlp` improve for large values of n, and `rrt` and `rmlp` work efficiently for $n = 1$. Indeed, `rrt` and `rmlp` most probably overfit the data and therefore, the parameters that seem to be best on X_{val} may not work well on test set. Among the rule based referees, `rct` and `rctr` work well, and `rctr` is slightly better than `rct`. Moreover, `rmlp` is better than `rlp` for $n = 1$ whereas `mel` is significantly better than `mem` in 5 datasets.

After setting the optimal parameters defined above, X_{test} is used to compare all of the methods. The results given in Table 6.13 can be summarized as follows:

- Instead of using all of the 21 base classifiers within `vote` method, we can use only the one that is selected by `sbs` or `rctr`. By doing so, the classification cost will decrease significantly. If we have a classifier ensemble that includes classifiers with high classification cost, such as k-nn, it is obviously better to use a selection

Table 6.13. Results of the different methods with predefined parameters

	vote	sbs-1	sbd-7	sr-7	cin-7	rctr-1	rmlp-1	mel
vote	0/40/0	0/37/3	3/37/0	4/36/0	2/37/1	0/39/1	2/37/1	1/36/3
sbs-1	3/37/0	0/40/0	4/36/0	5/35/0	2/38/0	1/39/0	3/35/2	0/39/1
sbd-7	0/37/3	0/36/4	0/40/0	1/37/2	1/38/1	1/36/3	2/36/2	1/36/3
sr-7	0/36/4	0/35/5	2/37/1	0/40/0	1/35/4	1/38/1	1/37/2	1/34/5
cin-7	1/37/2	0/38/2	1/38/1	4/35/1	0/40/0	4/33/3	5/33/2	0/37/3
rctr-1	1/39/0	0/39/1	3/36/1	1/38/1	3/33/4	0/40/0	3/37/0	1/37/2
rmlp-1	1/37/2	2/35/3	2/36/2	2/37/1	2/33/5	0/37/3	0/40/0	0/37/3
mel	3/36/1	1/39/0	3/36/1	5/34/1	3/37/0	2/37/1	3/37/0	0/40/0

unit and then use only the referred classifier rather than all classifiers.

- Instead of simple voting, using a more clever system which makes input dependent weighted voting, like `mel`, may be more accurate.
- Although it is generally worse than other methods, random selection works well if n is sufficiently large.
- The dynamic selection method `sbd` seems to have more losses than wins over other systems when the results on 40 datasets are considered.
- It is hard to say which one is better when `cin` and `rctr` are compared since one is superior to the other in different cases. One may compare their costs in classification phase, where `cin` needs to calculate pairwise distances between the data instances while `rctr` needs to get outputs of the referees first. Note also that `cin` uses 7 classifiers whereas `rctr` uses only 1.
- Although there are some cases that `rmlp` works better than other systems, the comparison with `mel` shows that using `mel` may be more accurate.

These comparisons, however, are just based on the specific case where we use 21 base classifiers and various datasets. It is also possible to foresee the cases where these methods will be successful or fail by considering the results on each dataset individually as well as the case studies in the previous chapter, and much analysis is done during the study. Our conclusions on the different strategies are given below:

The strategy used in `sbs` is reasonable in general and it is easy to apply. However, `sbs` is an efficient method only if the ensemble includes at least one classifier that is accurate, so that it will be selected as the best classifier. If the ensemble includes classifiers with moderate accuracy values all over the input space, as seen in the previous chapter, by choosing one of them and using only it, the accuracy is limited to the individual accuracy of the selected (moderate) classifier. Similarly, when the ensemble includes inaccurate classifiers, `sbs` method will be unsuccessful, too. Note that it is not the quantity but the quality of the base classifiers that affects `sbs`.

As seen in the experiments on various datasets, performance of `sbd` varies too much and therefore one should examine the individual classifiers carefully. If the classifier ensemble includes some classifiers that give higher confidence values than the others, then the system may be misled by them. As the case studies in the previous chapter have shown, if the individual performance of the confident classifier is not good (the case in *yeast*), its harm might be lessened by using more number of classifiers together, whereas `sbd` can be one of the best strategies if that classifier is accurate on the given dataset (the case in *monks*). Due to the data dependent selection, it is possible to obtain higher performance than the individual classifiers, and even higher than `sbs`.

As expected, random selection can be an efficient method only if a high percentage of the classifiers in the ensemble are accurate on the given dataset. Otherwise, as also shown with the experiments, it is not very effective.

The number of data instances in the given dataset is an important factor that affects not only the cost but also the accuracy of the local competence based selection methods. Although the cost of the classification will be high, this method works accurately on datasets having a large or moderate size. Moreover, we observe that the accuracy of `cdp` on different datasets is mostly affected with the changes in k, the neighborhood parameter.

When ensembles that include mostly moderately accurate or inaccurate classifiers are used, especially for small values of n, `rrs` is not preferable, whereas its accuracy on

some datasets is high if the ensemble is large. Its performance is also affected with the dataset itself; it is important to know whether the dataset can be partitioned properly with rule sets or not. The behavior of `rrt` is similar to `sbd` method and is affected by the type of the base classifiers considerably. The two methods, `rct` and `rctr` perform acceptably well. The analysis shows that they generally select bad classifiers besides the best ones from the ensemble which indicates that even the inaccurate classifiers can be selected whenever it seems to be reasonable for the given particular data. Base classifiers which may be considered inaccurate over the whole input domain may be quite accurate on some part of the input space, and the trick is to know where to use them, which is exactly what the referees are doing.

The unified structure `rlp` is not good in most of the cases whereas `rmlp` is comparable to `rctr`. Although `rmlp` seems to use all of the classifiers in the ensemble nearly uniformly whatever their type is, our analysis shows that it generally makes correct decisions by selecting a classifier for the cases it is accurate.

The system that combines the classifiers with (data dependent) weights also gains benefit from using data dependent information. For small ensembles, `mel` is slightly better than `mem` and `vote`. Note that even if the ensemble includes accurate and inaccurate classifiers together, these methods select base classifiers more uniformly than `rct` does.

7. CONCLUSIONS AND FUTURE WORK

Learning systems are used in various fields of engineering, in general and in computer science in particular. There are many different learning algorithms but none is the best on all datasets. Since the cases where one classifier is superior to the others varies, one way to improve classifier systems is to use many classification algorithms together.

There are various strategies used to combine multiple classifiers to construct a composite system. When different classifiers are used together, the risk of choosing wrong classifier decreases while the ability to represent the input space increases. The use of multiple classifiers is also supported by the fact that the patterns that are misclassified by different classifiers are not necessarily the same. Classifier selection methods are based on the idea that each base classifier is a local expert, that is, it knows a part of the input space, and one of the classifiers can be selected to label the given input if it is predicted to be accurate for that input.

First, we examine current methodologies and categorize them. Then, beside the simple methods existing in the literature, we give details about the proposed classifier selection systems each having a trained selection unit. The task of the selection unit is to predict which classifier in the ensemble can classify the given test instance accurately. This is also an example of the data dependent combination techniques which have various successful examples, as mentioned in Section 2.2.2.

We train the selection unit so that it learns the subdomains of the input space where the corresponding base classifier is accurate. Since learning the areas of expertise of the individual classifiers is a hard problem, we construct the selection units of the proposed methods using different learning algorithms based on different assumptions and analyze their success not only in terms of the overall accuracy but also the correctness of their preferences.

First of all, the experiments show that simple methods such as voting and selection of the most accurate classifier (on validation set) perform well only if an appropriate number of successful classifiers exist in the ensemble. In such cases, selecting the best classifier(s) outperforms any other combination technique since the others also use inaccurate classifiers which causes a decrease in the accuracy. However, the major reason for using multiple classifiers is the absence of an accurate classifier on the given problem. As seen in the case studies in Chapter 5, the simple methods do not improve the accuracy as much as the proposed methods if the ensemble includes just unsuccessful classifiers or even moderately accurate and inaccurate classifiers together.

The methods that select different classifiers for different data instances provide higher accuracy than the individual classifiers as well as the simple methods, especially when none of the classifiers in the ensemble is quite accurate. When rule based techniques are used for selection, it is seen that they also select the (static) best classifier most frequently, but the interesting point is that the second or third most frequently selected classifier is the least successful one (with the smallest average accuracy). This is an important indicator that classifier selection systems can gain benefit from even the worst classifier, if it is locally accurate. The reason for preferring the worst one instead of the moderately accurate classifiers may be that the region where the bad classifier is locally accurate is distinguishable in the input space while the areas of expertise of more accurate classifiers do not have a specific pattern.

It is also possible to learn the areas of expertise of the classifiers by training a neural network, such as a MLP. Because we define the learning process as a regression problem, the network produces soft outputs for the classifiers. This may be the reason for their selection strategy which behaves more uniformly than the rule based ones. Although the type of the selected classifiers are not similar, network structures and rule based referees both promise to be successful referees, if the complexity of the network and the measurement of the competence used by the rules are set properly.

As an alternative to the referee based systems, a gating network can be used, which produces input dependent weights to define the participation of the classifiers

in the final decision combination process. The gating network is also trained to learn local experts in the ensemble and we observe that by using a simple gating network, it is possible to achieve accuracy values comparable with voting. This system is superior to voting especially in cases where the ensemble size is small.

The disadvantage of the proposed systems may be the additional cost that the selection unit brings. However, the cost both in training and in testing is unavoidable if we want to have a clever (trained) selection unit and make reasonable selections rather than a static selection that does not take the particular data instance into account. Note also that the method that selects the best classifier dynamically has to obtain the predictions of all classifiers first while the local competence based method has to find neighbors of the given instance first and then calculate the accuracy of each classifier which makes the classification phase too costly on various datasets. If the cost of running all of the classifiers in the ensemble is high, then running a selector unit first and then only the classifiers that are selected may even be more beneficial since the structures we use as selector do not have high classification costs. Moreover, with referee or gating methods, base classifiers which are never selected on X_{val} may be considered redundant and may be pruned. This would decrease the overall accuracy.

In the test results we presented, the dataset X_{test} is used. On X_{test} we have ten results and can check the average behavior over multiple runs, but the disadvantage is that we use resampled data which do not provide independent values. As a future work, the results on an independent test set X_T, unseen during training, can be used to evaluate the generalization of the systems.

The rule based systems have some advantages over the unified neural network based systems: Adding/deleting a base classifier is independent from the other classifiers in the ensemble. Moreover, the complexity of a referee structure, such as a decision tree, can be different from the others since the areas of expertise do not have a common pattern for all base classifiers. However, for a very large ensemble using an independent referee structure for each base classifier may be disadvantageous in terms of cost. Rather than separate structures, one may try to construct a unified referee

structure that is also rule based. However, since the areas of expertise of the classifiers overlap too much, such a training may not be possible. On the other hand, it is possible to implement a cascading system by using a referee structure. If one of the classifiers is thought to be more successful or more preferable to the other classifiers, after a referee (decision tree) for that classifier is constructed, instead of creating new trees for the other classifiers, new subtrees may be added to the nodes where the first classifier is referred as "dontUseMe". It is also possible to decrease the cost of having various decision trees by constructing them as multivariate trees rather than univariate so that their depth may lessen while they divide the input space into regions more successfully.

APPENDIX A: METHODS FOR RESAMPLING AND TESTING

A.1. 5×2 Cross-Validation

To resample a given dataset X using 5×2 cross-validation (cv), it is randomly divided into two parts five times and the halves are swapped each time.

In the beginning, X is randomly divided into two parts as $X_1^{(1)}$ and $X_1^{(2)}$, which gives the first pair of training and validation sets. Note that $X_i^{(j)}$ denotes half j of fold i. Once the first fold is obtained, the role of the two halves are swapped and the second pair for training and validation sets is obtained: $X_1^{(2)}$ and $X_1^{(1)}$.

To get the second fold, X is shuffled randomly and this new fold is also divided into two as $X_2^{(1)}$ and $X_2^{(2)}$. Then these two halves are swapped to get another pair. By doing this process for three more folds, since we get two pairs in each fold, we obtain ten training and validation sets:

$\underline{X_{tra}}$	$\underline{X_{val}}$
$X_1^{(1)}$	$X_1^{(2)}$
$X_1^{(2)}$	$X_1^{(1)}$
$X_2^{(1)}$	$X_2^{(2)}$
$X_2^{(2)}$	$X_2^{(1)}$
$\ldots$	$\ldots$
$X_5^{(1)}$	$X_5^{(2)}$
$X_5^{(2)}$	$X_5^{(1)}$

A.2. 5×2 cv Paired F Test

Once 5×2 cv is performed, to compare two classification algorithms, the differences between the errors rates of the algorithms are calculated for each fold $i = 1, \ldots, 5$ and each half $j = 1, 2$. The average of the error rates $p_i^{(j)}$ on fold i is $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$,

and the estimated variance is $s_i^2 = (p_i^{(1)} - \bar{p_i})^2 + (p_i^{(2)} - \bar{p_i})^2$.

Under the null hypothesis that two algorithms have the same error rate, $p_i^{(j)}$ is the difference of two identically distributed proportions, and it can be treated as approximately normal distributed with 0 mean and unknown variance σ^2 by ignoring the fact that these proportions are not independent. Then $p_i^{(j)}/\sigma$ is approximately unit normal. If $p_i^{(1)}$ and $p_i^{(2)}$ are assumed to be independent normals, then s_i^2/σ^2 has a chi-square distribution with one degree of freedom. If each of the s_i^2 are assumed to be independent, then their sum is chi-square with five degrees of freedom:

$$M = \frac{\sum_i s_i^2}{\sigma^2} \sim \chi_5^2 \tag{A.1}$$

and

$$t = \frac{p_1^{(1)}}{\sqrt{M/5}} = \frac{p_1^{(1)}}{\sqrt{\sum_i s_i^2/5}} \sim t_5 \tag{A.2}$$

giving a t statistic with five degrees of freedom.

If $p_i^{(j)}/\sigma \sim Z$, then $(p_i^{(j)})^2/\sigma^2 \sim \chi_1^2$ and their sum is chi-square with ten degrees of freedom:

$$N = \frac{\sum_i \sum_j (p_i^{(j)})^2}{\sigma^2} \sim \chi_{10}^2 \tag{A.3}$$

Placing this in the numerator of Equation A.2, a statistic that is the ratio of two chi-square distributed random variables is obtained [53]. Two such variables divided by their respective degrees of freedom is F distributed:

$$f = \frac{N/10}{M/5} = \frac{\sum_i \sum_j (p_i^{(j)})^2}{2\sum_i s_i^2} \sim F_{10,5} \tag{A.4}$$

5×2 cv paired F test accepts the hypothesis that the algorithms have the same

error rate at significance level α if this value is less than $F_{\alpha,10,5}$.

A.3. Sign Test

The t-test is the standard test for testing that the difference between population means of two paired samples are equal. If the populations are non-normal, particularly for small samples, then the t-test may not be valid. The sign test is an alternative that can be applied when distributional assumptions are suspect. Sign test is a simple binomial test and a two-sided sign test is used to check whether the data in a vector come from a distribution whose median is zero.

Once 5×2 cv paired F test is applied, the number of cases that two classification algorithms are significantly better/worse than each other, i.e. the number of *win*s and *loss*es, are obtained. Under the null hypothesis both wins and losses follow a binomial distribution with $p = 1/2$. Then, by using tables of the binomial distribution, if the vector of the wins $(+1)$, losses (-1) and ties (0) is found to have zero median, the algorithms are said to be indifferent from each other according to the defined significance level.

APPENDIX B: RESULTS ON THE VALIDATION SET

The results of the classifier selection systems (examined in Chapters 3 and 4) on X_{val} are given in Tables B.1–B.9. Note that X_{val} is the dataset which is used to train the selection units of the systems.

Table B.1. Results of `sbs` method

	sbs-1	sbs-3	sbs-5	sbs-7
sbs-1	0/40/0	0/40/0	1/39/0	3/37/0
sbs-3	0/40/0	0/40/0	0/40/0	2/38/0
sbs-5	0/39/1	0/40/0	0/40/0	1/39/0
sbs-7	0/37/3	0/38/2	0/39/1	0/40/0

Table B.2. Results of `sbd` method

	sbd-1	sbd-3	sbd-5	sbd-7
sbd-1	0/40/0	*0/29/11*	*0/25/15*	*0/23/17*
sbd-3	*11/29/0*	0/40/0	*1/31/8*	0/35/5
sbd-5	*15/25/0*	*8/31/1*	0/40/0	0/37/3
sbd-7	*17/23/0*	5/35/0	3/37/0	0/40/0

Table B.3. Results of `sr` method

	sr-1	sr-3	sr-5	sr-7
sr-1	0/40/0	*0/26/14*	*0/20/20*	*0/22/18*
sr-3	*14/26/0*	0/40/0	0/36/4	0/36/4
sr-5	*20/20/0*	4/36/0	0/40/0	0/38/2
sr-7	*18/22/0*	4/36/0	2/38/0	0/40/0

Table B.4. Results of `cin` and `cdp` methods

	cin-1	cin-3	cin-5	cin-7	cdp-1	cdp-3	cdp-5	cdp-7
cin-1	0/40/0	2/31/7	2/32/6	*2/27/11*	1/36/3	3/30/7	3/30/7	3/30/7
cin-3	7/31/2	0/40/0	1/37/2	0/38/2	2/38/0	2/37/1	3/36/1	3/35/2
cin-5	6/32/2	2/37/1	0/40/0	0/38/2	6/34/0	3/37/0	2/38/0	4/35/1
cin-7	*11/27/2*	2/38/0	2/38/0	0/40/0	*9/31/0*	*6/34/0*	3/37/0	3/37/0
cdp-1	3/36/1	0/38/2	*0/34/6*	*0/31/9*	0/40/0	1/35/4	1/34/5	1/36/3
cdp-3	7/30/3	1/37/2	0/37/3	0/34/6	4/35/1	0/40/0	1/37/2	1/38/1
cdp-5	7/30/3	1/36/3	0/38/2	0/37/3	5/34/1	2/37/1	0/40/0	0/39/1
cdp-7	7/30/3	2/35/3	1/35/4	0/37/3	3/36/1	1/38/1	1/39/0	0/40/0

Table B.5. Results of `rct` and `rctr` methods

	rct-1	rct-3	rct-5	rct-7	rctr-1	rctr-3	rctr-5	rctr-7
rct-1	0/40/0	0/40/0	0/40/0	0/40/0	0/39/1	0/39/1	0/39/1	0/39/1
rct-3	0/40/0	0/40/0	0/40/0	0/40/0	0/39/1	0/39/1	0/39/1	0/39/1
rct-5	0/40/0	0/40/0	0/40/0	0/40/0	0/39/1	0/39/1	0/39/1	0/39/1
rct-7	0/40/0	0/40/0	0/40/0	0/40/0	0/39/1	0/39/1	0/39/1	0/39/1
rctr-1	1/39/0	1/39/0	1/39/0	1/39/0	0/40/0	0/40/0	0/40/0	0/40/0
rctr-3	1/39/0	1/39/0	1/39/0	1/39/0	0/40/0	0/40/0	0/40/0	0/40/0
rctr-5	1/39/0	1/39/0	1/39/0	1/39/0	0/40/0	0/40/0	0/40/0	0/40/0
rctr-7	1/39/0	1/39/0	1/39/0	1/39/0	0/40/0	0/40/0	0/40/0	0/40/0

Table B.6. Results of `rrs` method

	rrs-1	rrs-3	rrs-5	rrs-7
rrs-1	0/40/0	0/37/3	0/38/2	1/37/2
rrs-3	3/37/0	0/40/0	0/40/0	0/38/2
rrs-5	2/38/0	0/40/0	0/40/0	0/40/0
rrs-7	2/37/1	2/38/0	0/40/0	0/40/0

Table B.7. Results of `rrt` method

	rrt-1	rrt-3	rrt-5	rrt-7
rrt-1	0/40/0	5/35/0	*8/32/0*	*11/29/0*
rrt-3	0/35/5	0/40/0	4/36/0	*8/32/0*
rrt-5	*0/32/8*	0/36/4	0/40/0	3/37/0
rrt-7	*0/29/11*	*0/32/8*	0/37/3	0/40/0

Table B.8. Results of `rlp` and `rmlp` methods

	rlp-1	rlp-3	rlp-5	rlp-7	rmlp-1	rmlp-3	rmlp-5	rmlp-7
rlp-1	0/40/0	0/38/2	2/33/5	1/35/4	*0/33/7*	*0/33/7*	*0/34/6*	*0/33/7*
rlp-3	2/38/0	0/40/0	0/39/1	0/36/4	0/35/5	0/36/4	0/36/4	0/37/3
rlp-5	5/33/2	1/39/0	0/40/0	0/38/2	0/36/4	0/38/2	1/37/2	0/38/2
rlp-7	4/35/1	4/36/0	2/38/0	0/40/0	0/37/3	0/37/3	0/36/4	1/37/2
rmlp-1	*7/33/0*	5/35/0	4/36/0	3/37/0	0/40/0	2/38/0	3/37/0	3/37/0
rmlp-3	*7/33/0*	4/36/0	2/38/0	3/37/0	0/38/2	0/40/0	0/39/1	2/37/1
rmlp-5	*6/34/0*	4/36/0	2/37/1	4/36/0	0/37/3	1/39/0	0/40/0	1/39/0
rmlp-7	*7/33/0*	3/37/0	2/38/0	2/37/1	0/37/3	1/37/2	0/39/1	0/40/0

Table B.9. Results of `mel` and `mem` methods

	mel	mem	vote
mel	0/40/0	5/35/0	6/32/2
mem	0/35/5	0/40/0	2/35/3
vote	2/32/6	3/35/2	0/40/0

REFERENCES

1. Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, 2004.

2. Kuncheva, L. I., *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.

3. Wolpert, D. H. and W. G. Macready, "No Free Lunch Theorems for Search", *Technical Report Santa Fe Institute SFI-TR-95-02-010*, 1995.

4. Dietterich, T. G., "Ensemble Methods in Machine Learning", *Lecture Notes in Computer Science*, Vol. 1857, pp. 1–15, 2000.

5. Alpaydın, E. and C. Kaynak, "Cascading Classifiers", *Kybernetika*, Vol. 34, pp. 369–374, 1998.

6. Alpaydın, E., "REx: Learning A Rule and Exceptions", *International Computer Science Institute TR-97-040*, 1997.

7. Gama, J., "Combining Classifiers by Constructive Induction", *10th European Conference on Machine Learning*, pp. 178–189, 1998.

8. Gama, J. and P. Brazdil, "Cascade Generalization", *Machine Learning*, Vol. 41, No. 3, pp. 315–343, 2000.

9. Hendrickx, I., "Local Classification and Global Estimation, Explorations of the k-Nearest Neighbor Algorithm", *PhD Thesis*, Tilburg University, The Netherlands, 2005.

10. Domingos, P., "Unifying Instance-Based and Rule-Based Induction", *Machine Learning*, Vol. 24, No. 2, pp.141–168, 1996.

11. Rahman, A. F. R. and M. C. Fairhurst, "Multiple Classifier Decision Combina-

tion Strategies for Character Recognition: A Review", *International Journal on Document Analysis and Recognition*, Vol. 5, No. 4, pp. 166-194, 2003.

12. Ali, K. M. and M. J. Pazzani, "Error Reduction through Learning Multiple Descriptions", *Machine Learning*, Vol. 24, No. 3, pp. 173–202, 1996.

13. Kittler, J., M. Hatef, R. P. W. Duin, and J. Matas, "On Combining Classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226–239, 1998.

14. Kang, H.-J. and S.-W. Lee, "Experimental Results on the Construction of Multiple Classifiers Recognizing Handwritten Numerals", *6th International Conference on Document Analysis and Recognition*, pp. 1026–1030, 2001.

15. Kang, H.-J. and D. Doermann, "Selection of Classifiers for the Construction of Multiple Classifier Systems", *8th International Conference on Document Analysis and Recognition*, pp. 1194–1198, 2005.

16. Dasarathy, B. V. and B. V. Sheela, "A Composite Classifier System Design: Concepts and Methodology", *Proceedings of the IEEE*, Vol. 67, No. 5, pp. 708–713, 1979.

17. Ho, T. K., J. J. Hull, and S. N. Srihari, "Decision Combination in Multiple Classifier Systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 1, pp. 66–75, 1994.

18. Woods, K., W. P. Kegelmeyer, and K. Bowyer, "Combination of Multiple Classifiers Using Local Accuracy Estimates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 405–410, 1997.

19. Giacinto, G. and F. Roli, "Adaptive Selection of Image Classifiers", *9th International Conference on Image Analysis and Processing*, pp. 38–45, 1997.

20. Giacinto, G. and F. Roli, "Methods for Dynamic Classifier Selection", *10th Inter-*

national Conference on Image Analysis and Processing, Venice, Italy, pp. 659–664, 1999.

21. Kuncheva, L. I., "Clustering and Selection Model for Classifier Combination", *Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, pp. 185–188, 2000.

22. Frosyniotis, D., A. Stafylopatis, and A. Likas, "A Divide-and-Conquer Method for Multi-net Classifiers", *Pattern Analysis and Applications*, Vol. 6, No. 1, pp. 32–40, 2003.

23. Kuncheva, L. I., "Switching Between Selection and Fusion in Combining Classifiers: An Experiment", *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 32, No. 2, pp. 146–156, 2002.

24. Wolpert, D. H., "Stacked Generalization", *Neural Networks*, Vol. 5, pp. 241–259, 1992.

25. Dzeroski, S. and B. Zenko, "Stacking with Multi-Response Model Trees", *3rd International Workshop Multiple Classifier Systems*, pp. 201–211, 2002.

26. Tsoumakas, G., I. Katakis, and I. Vlahavas, "Effective Voting of Heterogeneous Classifiers", *15th European Conference on Machine Learning*, pp. 465–476, 2004.

27. Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts", *Neural Computation*, Vol. 3, pp. 79–87, 1991.

28. Alpaydın, E. and M. I. Jordan, "Local Linear Perceptrons for Classification", *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, pp. 788–792, 1996.

29. Jordan, M. I. and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, Vol. 6, pp. 181–214, 1994.

30. Jordan, M. I. and L. Xu, "Convergence Results for the EM Approach to Mixtures

of Experts Architectures", *Neural Networks*, Vol. 8, pp. 1409–1431, 1995.

31. Ortega, J., "Exploiting Multiple Existing Models and Learning Algorithms", *AAAI96 Workshop in Induction of Multiple Learning Models*, 1995.

32. Koppel, M. and S. P. Engelson, "Integrating Multiple Classifiers by Finding Their Areas of Expertise", *AAAI96 Workshop on Integrating Learned Models*, MIT Press, 1996.

33. Ortega, J., M. Koppel, and S. Argamon, "Arbitrating Among Competing Classifiers Using Learned Referees", *Knowledge and Information Systems*, Vol. 3, pp. 470–490, 2001.

34. Demir, Ç. and E. Alpaydın, "Cost-Conscious Classifier Ensembles", *Pattern Recognition Letters*, Vol. 26, No. 14, pp. 2206–2214, 2005.

35. Duin, R. P. W., "The combining classifier: to Train or Not to Train?", *16th International Conference on Pattern Recognition*, Vol. 2, pp. 765–770, 2002.

36. Wanas, N. M., R. A. Dara, and M. S. Kamel, "Adaptive Fusion and Co-operative Training for Classifier Ensembles", *Pattern Recognition*, Vol. 39, pp. 1781–1794, 2006.

37. Pudil, P., J. Novovicova, S. Blaha, and J. Kittler, "Multistage Pattern Recognition with Reject Option", *11th IAPR International Conference on Pattern Recognition*, Vol. 2, pp. 92–95, 1992.

38. Chan, P. K. and S. J. Stolfo, "A Comparative Evaluation of Voting and Meta-learning on Partitioned Data", *International Conference on Machine Learning*, pp. 90–98, 1995.

39. Freund, Y. and R. E. Schapire, "Experiments with a New Boosting Algorithm", *International Conference on Machine Learning*, pp. 148–156, 1996.

40. Egmont-Petersen, M., W. R. M. Dassen, and J. H. C. Reiber, "Sequential Selection of Discrete Features for Neural Networks - A Bayesian Approach to Building A Cascade", *Pattern Recognition Letters*, Vol. 20, No. 11, pp. 1439–1448, 1999.

41. Ting, K. M. and I. H. Witten, "Stacked Generalization: When Does It Work?", *International Joint Conference on Artificial Intelligence*, 1997.

42. Ferri, C., P. Flach, and J. Herrandez-Orallo, "Delegating Classifiers", *21st International Conference on Machine Learning*, pp. 106–110, 2004.

43. Oliveira, L. S., A. S. Britto, and R. Sabourin, "Improving Cascading Classifiers with Particle Swarm Optimization", *8th International Conference on Document Analysis and Recognition*, pp. 570–574, 2005.

44. Fürnkranz, J., "Separate-and-Conquer Rule Learning", *Artificial Intelligence Review*, Vol. 13, No. 1, pp. 3–54, 1999.

45. Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

46. Cohen, W. W., "Fast Effective Rule Induction", *12th International Conference on Machine Learning*, pp. 115–123, 1995.

47. Fürnkranz, J. and G. Widmer, "Incremental Reduced Error Pruning", *11th International Conference on Machine Learning*, pp. 70–77, 1994.

48. Newman, D. J., S. Hettich, C. L. Blake, and C. J. Merz, "UCI Repository of Machine Learning Databases", http://www.ics.uci.edu/∼mlearn/MLRepository.html

49. Rasmussen, C. E., R. M. Neal, G. E. Hinton, D. van Camp, Z. Ghahramani, R. Kustra, and R. Tibshirani, "The DELVE Manual", http://www.cs.utoronto.ca/∼delve/ 1996.

50. Guyon, I., I. Poujoud, L. Personnaz, G. Dreyfus, J. Denker, and Y. le Cun, "Comparing Different Neural Architectures for Classifying Handwritten Digits", *Interna-*

tional Joint Conference on Neural Networks, Washington, USA, 1989.

51. Yıldız, O. T. and E. Alpaydın, "Linear Discriminant Trees", *International Conference on Machine Learning*, Stanford University, USA, 2000.

52. Chang, C.-C. and Lin C.-J., "LIBSVM A Library for Support Vector Machines", http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

53. Alpaydın E., "Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms", *Neural Computation*, Vol. 11, No. 8, pp. 1885–1892, 1999.