Learning the areas of expertise of classifiers in an ensemble

Esma Kilic a *, Ethem Alpaydin a

a Department of Computer Engineering, Bogazici University, 34342 Istanbul, Turkey

Abstract

There are various machine learning algorithms for extracting patterns from data; but recently, decision combination has become popular to improve accuracy over single learner systems. The fundamental idea behind combining the decisions of an ensemble of classifiers is that different classifiers most probably misclassify different patterns and by suitably combining the decisions of complementary classifiers, accuracy can be improved.

In this paper, we investigate two kinds of classifier systems which are capable of estimating how much to weight each base classifier dynamically; during the calculation of the overall output for a given test data instance: (1) In “referee-based system”, a referee is associated with each classifier which learns the area of expertise of its associated classifier and weights it accordingly. (2) However, “gating system” learns to partition the input space among all classifiers. Each referee in referee-based system learns a two-class problem (i.e., whether to use or not to use a classifier) whereas a gating system learns an L-class problem assigning the input to one of L base classifiers.

Our analysis on 20 datasets from different domains and a classifier pool including 21 base learning algorithms reveals that the gating system tends to concentrate on a few of the base classifiers whereas a use of referees leads to a more balanced use of the base classifiers. Moreover, in the case of referees, it is better to use a small subset of base classifiers, instead of a single one.

The study shows that, by using well-trained selection unit (referee or gating), we can get as high accuracy as using all the base classifiers (to combine their decisions) with drastic decrease in the number of base classifiers used, and also improve accuracy. The improvement is significant especially in cases when none of the base classifiers has high accuracy and it indicates that selecting classifiers appears promising as a means of solving hard learning problems.

Keywords: Classifier selection; multiple classifier systems; classifier ensembles

1. Introduction

There are many classification algorithms each based on different assumptions about the data and we can not say a priori which one will work best for a given dataset. The input regions that are misclassified by different base classifiers (learners/experts) are not necessarily the same and combining multiple base classifiers that complement each other promises to achieve higher accuracy than using any single classifier [1].

Base classifiers are made different (uncorrelated/independent/diverse) by training them with different algorithms, by training them on re-sampled datasets (as in bagging/boosting), or by using different input features. In this work, we will be concerned with how the base classifiers are fused rather than how they are constructed. The simplest method for fusing of the outputs is to use a fixed rule, such as averaging or voting [2]. Stacked generalization [3] trains a second level classifier with the outputs of the base classifiers. In both voting and stacking, all the base
classifiers are used for all test inputs. In this work, however, we will be concerned with approaches where we acknowledge that base classifiers may not be accurate in all parts of the input space; our approach then includes estimating how much accurate a classifier is expected to be for a given test instance and then fusing only the classifiers that are predicted to be accurate on this instance. Also, we are not concentrated on the methods for fusing of the outputs of the base classifiers (and use the simple voting method for the fusion) but the behavior and success of the selection systems on partitioning the input space among the base classifiers.

There have been studies which aim choosing a subset from a larger set of classifiers [1]. The selection aims for complexity reduction or higher accuracy, or both. These are static selection methods in that one subset is chosen by looking at accuracy/diversity and the same subset is used for all test instances [4, 5]. In our dynamic selection approach, the selection is done online, during classification, and hence different subsets may be chosen for different test instances. Dynamic selection, by focusing on a different optimal subset for each given instance, promises to be more effective at finding the appropriate subset.

Our strategy is to select $n$ of $L$ base classifiers (such that $n \leq L$) to calculate local competences of the base classifiers. The method proposed by Woods et al. [6] is based on local classifier accuracies estimated for local regions defined in terms of the k-nearest neighbors of the given test instance. In the class-dependent version, only the neighbors that have the same estimated class label with the estimated label of the test instance are taken into account to determine the local area, whereas in the class-independent version all neighbors are taken into account regardless of their class labels. In the study of Giacinto and Roli [7], the system can also reject a test instance if none of the local accuracies is higher than a predefined threshold. However, the calculations in both methods become too costly for large datasets because of the need for calculating the distances to find the k-nearest neighbors.

Kuncheva [8] proposes another method to define the input regions during learning where she first clusters the inputs and then searches for the most accurate classifier in each cluster. Clustering is easy to apply and appropriate for dividing the input space into regions of high density. However, clusters in the input space may not match with the regions of high accuracy of the classifiers.

Competence based strategies are generally used to find only one, the single most competent classifier. To improve the overall classification accuracy, Kuncheva [9] and Woods et al. [6] suggest fusing the outputs of some or all of the base classifiers in the regions where there is not a single dominating base classifier. Similarly, Ko et al. [10] emphasize that by preferring dynamic ensemble selection to dynamic selection of one individual classifier, the risk that is caused with incorrect estimation decreases.

Another way to choose one or a subset of base classifiers dynamically is to use a gating system. In the adaptive mixtures of experts architecture proposed by Jacobs et al. [11], the base classifiers (experts) specialize in different regions of the input space and there is a trained gating network which selects one of the experts (in the competitive model) or assigns weights to experts (in the cooperative model). The experts and the gating network are trained together, in a coupled manner; that is, as the gating network learns to divide the input space, the experts learn the linear discriminant in each localized region.

In the system proposed by Ortega et al. [12], for each base classifier, a separate referee is built and an arbiter is used to select the most competent classifier based on the outputs of the referees. Ortega et al. use referees which are decision trees trained to discriminate the cases where the corresponding classifier is correct from those where it is incorrect. Wanas et al. [13] also propose an input dependent decision fusion scheme where a detector module 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 the mixture of experts and the upper level classifier in stacked generalization.

In this study, we are interested in developing composite systems where each base classifier’s area of expertise is learned and the optimal subset of the classifiers is chosen dynamically from a large pool for a given test instance. In our study, “optimality” refers to accuracy improvement and not using all of the base classifiers’ outputs but a few of them for the final decision fusion.

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 $n < L$ competent classifier(s) from the ensemble so that the overall accuracy using these $n$ classifiers is maximized while $n$ is minimized. To achieve this, we have to find regions of competences of the base classifiers, estimate their levels of competences in there, and decide on a selection/fusion strategy.

We explain our implementation of referees and gating systems in Section 2. The experimental results are
analyzed in Section 3 and we conclude in Section 4.

2. Classifier Selection

In the scenario we are interested in, we have base classifiers $D_j, j = 1, \ldots, L$, already trained on a training set $\text{Tra}$. We also have a validation set $\text{Val}$, composed of $x_t, t = 1, \ldots, N$, unused during the training of the base classifiers on which we will train our referees or gating.

In the general case, we consider a referee as a two-class classifier whose output, in the range $[0, 1]$, indicates the confidence in the accuracy of its corresponding base classifier. Similarly, we consider gating as an $L$-class classifier where $L$ is the number of base classifiers and gating’s output $p_j$ indicates the confidence that the base classifier $D_j$ is the most accurate base classifier for the given input. We use two different models for referees/gating; linear models and decision trees.

Note that for the $L$ base classifiers, we train $L$ independent referees as two-class classifiers and for any input, there may be more than one nonzero referee. On the other hand, gating is a 1-of-$L$ classifier and therefore, considers the base classifiers as mutually exclusive; that is, for any given input, there is a single base classifier which is the best qualified to generate the output among all other base classifiers. Note though that the gating output need not be a hard, 0/1 selection and may be soft values (as we have here) which may be used to weight the outputs of the classifiers.

2.1. Referees

We assume that the base classifiers, $D_j, j = 1, \ldots, L$, have already been trained on the training set $\text{Tra}$ and the associated referees, $R_j$, will be trained on the validation set $\text{Val}$. The output of a linear referee $R_j$ for a given instance $x$ is calculated as

$$p_j(x) = \frac{1}{1 + \exp[-(v_j^T x + v_j^0)]}$$  \hspace{1cm} (1)

where the sigmoid function constrains the outputs to lie between 0 and 1. In this learning problem, the desired output is 1 if the prediction of $D_j$ is correct, and is 0 if it is wrong. During training, the referee parameters, $v_j$, are optimized to minimize the total squared error.

A linear referee assumes that the expertise region of a classifier is linearly separable; that is, there is a hyperplane such that on one side, the classifier is correct for all instances and on the other side, it is wrong. This may be a restrictive assumption and we also try using a decision tree which is more flexible.

Similar to the linear referee, a tree referee $R_j$ is constructed to separate the validation data instances where $D_j$ is correct from those where $D_j$ is incorrect. After the construction of this two-class tree, any leaf is labeled with the average of the posterior probability values estimated for the true class of the instances covered by the specified leaf, that is:

$$p_j(x) = \frac{\sum_{c=1}^{N} d_j(x') l(x' \in \text{Leaf})}{\sum_{c=1}^{N} l(x' \in \text{Leaf})}$$  \hspace{1cm} (2)

where $d_j(x')$ is the posterior probability predicted by $D_j$ for the true class of $x'$ (c), $\text{Leaf}$ is the leaf (node) of the tree that covers input $x$, and $l(a)$ is the Boolean function which is 1 if $a$ is true and 0 otherwise.

In both linear and tree referees, $p_j(x)$, the output of the referee $R_j$, indicates the confidence that the corresponding base classifier $D_j$ will make correct decision for the given $x$.

In testing, to classify a test instance $x$, we calculate all $p_j(x)$ values, $j = 1, \ldots, L$, using linear/tree referees, then sort them in descending order and determine the $n$ classifiers among $L$ that have the highest corresponding $p_j(x)$ which we believe have the highest probability to be accurate on the given $x$. We can therefore write the combined decision as:
where $O_i(x)$ is the overall output for class $i$, $w_j(x)$ is 1 if $D_j$ is among the $n$ selected base classifiers and 0 otherwise. Then, we determine the final output, that is, the class label having the highest overall output $O_i(x)$, $i = 1, \ldots, K$:

$$y = \arg \max_i O_i(x)$$

(4)

Figure 1 illustrates the testing process of the referee-based systems in which we use all referees $R_j$ to obtain $p_j(x)$ values for a given $x$ and select the $n$ best of $L$ to be used in the decision aggregation (DA) unit where a vote is taken to determine the final output $y$.

![Fig. 1. Testing phase of the referee-based systems](image)

Note that learning the expertise region of a base classifier is another learning problem and the classifier we construct for it, i.e., the referee, may also be erroneous—the expertise region of a base classifier may not be separable by a line or a simple decision tree; that is why it is a good idea to set $n > 1$ and fuse a few base classifiers instead of using a single one.

Using a selective approach is advantageous in terms of cost (of running base classifiers): In combining methods like voting and stacking, the outputs of all base classifiers are used, whereas we select one (or few) expert(s) from the ensemble and use (run) only it/them. It may be the case that certain base classifiers may be too costly in terms of disk space or time complexity, or may use input representations that are costly to extract, and in such a case, selection has the advantage that only those which are actually needed will be evaluated. If a referee is not confident, its corresponding classifier need not be evaluated at all. Of course, for this to be meaningful, the referees themselves should be simple and that is why we use simple linear models or decision trees to construct them.

Due to the difficulty in learning the expertise regions of the base classifiers, Ortega et al. [12] propose to use some important attributes of the base classifiers and their predictions as well as the original input attributes. The drawback of using such a system is that, before using the referee, one has to execute the base classifier.

Using independent referees seems more advantageous for construction process than gating since the process of adding or deleting a base classifier and training its referee does not affect the other referees we may already have.

2.2. Gating

Different from the independent referees, in the case of gating, the outputs $p_j(x)$ indicating the competences of all base classifiers for the given $x$ are trained together. The outputs of the gating unit can both determine which base classifiers will be used, and at the same time, correspond to their weights in a weighted voting scheme.

In the case of linear gating, the softmax function is used instead of the sigmoid to force the winner-take-all mechanism among all base classifiers:
The estimated \( p_j(x) \) can then be used in place of \( w_j(x) \) in Eq. 3 to calculate the overall output in the DA unit of the system, as illustrated in Figure 2. The gating parameters are trained on a validation set \( \text{Val} \), different from \( \text{Tra} \), which is the set used to train the base classifiers.

Fig. 2. Testing phase of the gating-based systems, adapted from mixture of experts [11]

Our gating system is similar to that of mixture of experts [11] except:

(a) In the original model, both the local experts and the gating unit are trained together by using the same input data. In our proposed system, the base classifiers have already been trained on the training set \( \text{Tra} \) and the gating network is trained separately from the experts on a different validation set \( \text{Val} \).

(b) In the original mixture of experts, the base classifiers are assumed to be local experts and are all of the same type, whereas our base classifiers are general, they can be different (such as tree, nearest neighbor, etc.) and are trained over the whole input space.

The other alternative for constructing a gating system is using a multi-class decision tree which decides on how much to weight each base classifier for a given input \( x \), which again is trained on a validation set different from the training set of base classifiers. This is a tree where “classes” correspond to classifiers and “class posteriors” correspond to classifier weights in voting:

\[
p_j(x) = \frac{\exp(v_j^T x + v_{j0})}{\sum_{l=1}^L \exp(v_l^T x + v_{l0})}
\]

(5)

Before starting to train the gating tree, “the class of an instance” (the label, in other words) is set to be the index of one of the base classifier whose posterior probability estimate for the correct class is the highest. However, we realized during the experiments that labeling all of the instances with just one of the classifiers that gives the highest posterior probability for the true class of the instance will not be effective since in many cases the posterior probabilities were very close to each other. Therefore, our labeling method depends not only on the estimated posterior probabilities but also the local correctness of the base classifiers on the input space around the instance. The training starts by counting the instances that are correctly classified by base classifiers. The classifier that has the maximum score is used as the label for the instances it classifies correctly and this labeling is used to find a two-class split. This process is repeated at each node where we split the instances of the locally most accurate classifier from others.
3. Experimental Results

3.1. Datasets and Algorithms

We use 20 datasets which are from UCI [14] and Delve [15] machine learning repositories: australian, balance, breast, car, cmc, credit, mushroom, nursery, optdigits, pageblock, pendigits, pima, ringnorm, segment, spambase, thyroid, tic-tactoe, titanic, twonorm, yeast.

A given dataset is first divided into two parts: 1/3 as the test set and 2/3 as the training set. Then, (only) the training set is resampled using $5 \times 2$ cross-validation to generate ten training and validation folds. Tra is used to train the base classifiers. Val is divided into two randomly as val-A and val-B, and val-A is used to train the combiners, whereas val-B is used for model selection (i.e., to finetune $n$, the subset size) in the combiners. Test set, test, is only used to compare accuracies of the all combiners once all training and model selection processes are done.

We use $L = 21$ trained base classifiers:

- c45: The standard C4.5 decision tree algorithm.
- gau: Gaussian classifier where a common covariance matrix is shared by all classes.
- 1nn-7nn: $k$-nearest neighbor with $k = 1, 3, 5, 7$.
- ldt: Linear discriminant tree which uses linear discriminant analysis to find the best split as opposed to c45 which does exhaustive search.
- log: Linear logistic classifier.
- ml0-ml5: Multilayer perceptrons where, with $d$ inputs and $K$ classes, the number of hidden units can be $0$ (a linear model without any hidden layer), $d, K, d + K, (d + K)/2, 2(d + K)$.
- mlt: Multivariate linear discriminant tree that uses all inputs as opposed to the univariate decision trees, c45 and ldt.
- sm: A simple classifier that always chooses the class with the highest prior without looking at the input.
- sv0-sv4: Support vector machines as implemented in LIBSVM 2.82 [16] with radial (sv0), linear (sv1) and three polynomial kernels of degree 2, 3, 4.

The multiple classifier systems that will be compared in our study are two referee and two gating systems we proposed as well as simple (unweighted) voting over all base classifiers without any selection and the local competence based algorithm of Woods et al. [6]:

- rlp: The system with $L$ linear referee networks*.
- rdt: The system with $L$ referee trees*.
- glp: The system with linear gating network*.
- gdt: The system with gating tree*.
- cin: The class-independent version of the local competence based algorithm of Woods et al. [6], with a small modification so that it can be fairly compared with our methods; it uses not only one base classifier but takes a vote over $n \in \{1, 3, 5, 7\}$. (We did not prefer to use the class-dependent algorithm because the class labels of the instances are not taken into account in any other multiple classifier systems we use.)
- vote: Simple voting over $L$ base classifiers.

* Note that, all of these methods finally select $n$ of $L$ base classifiers and fuse the outputs of $n$ base classes.

3.2. Analysis

In this study, we experimented that the success of selection systems depends on various factors. First of all, the accuracies of the base classifiers have a major effect. For example; if all of the base classifiers in the classifier pool is unsuccessful to predict class label of a particular test instance, then any classifier system that selects a subset of the base classifiers cannot result in true prediction.

Secondly, if the selection strategy depends on success areas of the base classifiers on the input space, like referee and gating systems do, complexity of the success areas has a major effect on overall accuracy of the system. For example, optdigits, which is a 10-class problem, turns into a two-class problem for a referee but with a much more difficult decomposition problem. In Figure 3(a), the data instances of optdigits dataset belonging to only 3 classes are shown, after dimensionality reduction using principal components analysis (PCA). In Figure 3(b), the same
instances are labeled as “correctly/incorrectly classified” according to the output of the best base classifier, which for this case is sv2, support vector machine with a polynomial kernel of degree 2. We see that the expertise and the misclassification regions of sv2 overlap and this makes it difficult to learn the success pattern of sv2 by its associated referee.

![Figure 3](image-url)

Fig. 3. (a) optdigits test instances with class values ‘0’, ‘1’ and ‘2’ that are mapped onto 2-dimensional space by PCA, (b) Correctly and incorrectly classified optdigits test instances by sv2

We also analyzed characteristics of the selection systems in terms of selected base classifiers’ own success and frequency of each base classifier that has been selected. In lots of the datasets we see that rlp algorithm tends to select each of the base classifiers with similar frequencies while rdt selects the best and the worst base classifiers very rarely and it selects the classifiers that have medial accuracies frequently. In the following section, we summarize this analysis over 21 base classifiers and 20 datasets by counting how many times the base classifiers have been selected (i.e. find their selecting frequencies) and also indicate how many of the selected base algorithms were successful at guessing true class label of the data instance. Note that even the worst classifier (the base classifier that has the minimum accuracy level among the other classifiers for a particular dataset) may be successful at guessing true class label of the data instance it has been selected for. Therefore, we think that a successful selection system is the one that gives right decisions about which classifier should be selected for each data instance.

3.3. Results

To compare the accuracies of the different selection systems on 20 datasets and to determine statistically significant differences, we use the 5 × 2 cv F test [17]. In Table 1, the results are shown as a win/tie/loss matrix that indicates the number of datasets (out of 20) on which an algorithm has significantly higher/comparable/lower accuracy than another algorithm with at least 95% confidence. Note that, we did not simply display how many times referees and gating systems are resulted in higher accuracy than cin or vote algorithms but we preferred to expose only the “statistically significant” results so that our conclusions and inferences become more reliable. Moreover, we do this comparison two-sided with “wins” and “losses” to see whether an algorithm is significantly worse than another one on specific datasets. In the table, the suffix “n” indicates the subset size of the selected classifiers and it is the one that provides highest accuracy over val-B to the specified algorithm (n ∈ {1, 3, 5, 7}).

We see that rdt and rlp perform better than the other methods generally. Moreover, gdt is more accurate than the simple voting most of the time. When we decrease the significance threshold in the 5×2 cv F test from 0.95 to 0.85 in order to see also smaller accuracy differences, it is observed that the number of losses of vote increases versus all other methods, indicating that, it is more beneficial to use a trained selector mechanism and use only the classifiers suggested by the selector instead of using all.
Table 1. Comparison of the different ensemble methods on 20 datasets

<table>
<thead>
<tr>
<th>algorithm</th>
<th>rlp-n</th>
<th>rdt-n</th>
<th>cin-n</th>
<th>glp</th>
<th>gdt</th>
<th>vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>rlp-n</td>
<td>0/20/0</td>
<td>0/19/1</td>
<td>3/17/0</td>
<td>3/17/0</td>
<td>3/15/2</td>
<td>4/14/2</td>
</tr>
<tr>
<td>rdt-n</td>
<td>1/19/0</td>
<td>0/20/0</td>
<td>4/16/0</td>
<td>3/16/1</td>
<td>3/16/1</td>
<td>2/17/1</td>
</tr>
<tr>
<td>cin-n</td>
<td>0/17/3</td>
<td>0/16/4</td>
<td>0/20/0</td>
<td>0/17/3</td>
<td>0/18/2</td>
<td>2/16/2</td>
</tr>
<tr>
<td>glp</td>
<td>0/17/3</td>
<td>1/16/3</td>
<td>3/17/0</td>
<td>0/20/0</td>
<td>2/15/3</td>
<td>2/16/2</td>
</tr>
<tr>
<td>gdt</td>
<td>2/15/3</td>
<td>1/16/3</td>
<td>2/18/0</td>
<td>3/15/2</td>
<td>0/20/0</td>
<td>3/17/0</td>
</tr>
<tr>
<td>vote</td>
<td>2/14/4</td>
<td>1/17/2</td>
<td>2/16/2</td>
<td>2/16/2</td>
<td>0/17/3</td>
<td>0/20/0</td>
</tr>
</tbody>
</table>

The selection patterns of the proposed systems averaged over 20 datasets given in Table 2 indicate that the referee-based systems cause important differences among the selection frequencies of various base classifiers while the frequencies are more homogeneous for the gating-based systems*. By creating very different selection patterns, glp and gdt are accurate on different datasets.

(*Note that, if all of the base classifiers had been selected at the same frequencies, their selection frequency would be 100/L = 4.76. According to our definition “frequently” means that a base classifier has been selected more frequently than 5.76%, “occasionally”: between 3.76% and 5.76%, and “rarely”: less than 3.76%. Moreover, to form the misclassification levels, first, we evaluate the amount of data instances on which a base classifier has been used and how many of these cases are classified correctly/incorrectly. Then, by normalizing the amounts with the sum of L usage counts and multiplying them with 100, we obtain the usage percentages of each classifier that are comparable to each other. The correctness scales are defined in three levels according to the misclassification rates of the classifiers as: “correctly”: less than 0.5%, “intermediately”: between 0.5% and 1%, and “incorrectly”: more than 1%.)

Table 2. Selection frequency and correctness of the base classifiers averaged over 20 datasets

<table>
<thead>
<tr>
<th>algorithm</th>
<th>frequency</th>
<th>correctly</th>
<th>intermediately</th>
<th>incorrectly</th>
</tr>
</thead>
<tbody>
<tr>
<td>rlp</td>
<td>7 frequently</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8 occasionally</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6 rarely</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rdt</td>
<td>7 frequently</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>7 occasionally</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>7 rarely</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>glp</td>
<td>6 frequently</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>7 occasionally</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8 rarely</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>gdt</td>
<td>0 frequently</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>20 occasionally</td>
<td>4</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1 rarely</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The outputs show that gdt selects almost all base classifier with the same frequency while rdt algorithm tends to select some of them much more frequently than some others. However, as mentioned before, the important parameter to measure success of a selection system should be whether the selected classifiers have been successful at guessing the class label of the data instance that they have been selected for. From this point of view, glp seems more successful than the others (since 13 out of 20 base classifiers have been selected at right time, i.e. for the instances they can classify correctly) but rlp tends to make wrong decisions (4 out of 20 selected base classifiers have not been successful).
4. Conclusions

In this study, we have focused on learning the expertise areas of the base classifiers in an ensemble. The idea is based on estimating dynamically which base classifiers are likely to be accurate on the given input, then fusing only their outputs, and not evaluating those expected to be erroneous. This strategy potentially has the advantage of first, bias reduction because only the base classifiers which are accurate are used, and second, of complexity reduction, because if we can calculate the output using only a few base classifiers rather than the whole ensemble and if we can do this expertise estimation cheaply, we can do the whole recognition process using less computation, especially with respect to the cases where the pool includes costly base classifiers. That is why we use easily trainable structures for referee and gating systems.

Our simulation results show that our systems, both based on referees and gating, use only a few of the base classifiers and become as accurate as using all the base classifiers in the large classifier pool. Moreover, it is seen that by using trainable and input-dependent selector systems, one can benefit from even the least accurate classifiers by using them for the cases when they are predicted to be accurate. The reason for using a bad classifier may be that the region where it is locally accurate may be easily distinguishable in the input space while the expertise areas of more accurate classifiers do not have a simple pattern. As a result, the accuracy value achieved by the overall system can be higher than the one achieved by the single most accurate classifier in the classifier pool.

References