Chapter 6

Dynamic Combination of Multiple Classifiers System
CHAPTER 6
DYNAMIC COMBINATION OF MULTIPLE CLASSIFIERS SYSTEM

6.1 Introduction

To solve really hard problems, one may use several different representations. It is time to stop arguing over which type of pattern-classification technique is best. Instead, it is possible to work at a higher level of organization and discover how to build managerial systems to exploit the different virtues and evade the different limitations of each of these ways of comparing things.

The ultimate goal of effective mining from data streams in classification point of view is to achieve the best possible classification performance for the task at hand. Estimating classifier accuracy is important in that it allows one to evaluate how accurately a given classifier will label future data. Traditional pattern recognition systems use a single feature descriptor and a particular classification procedure to determine the true class of a given pattern. The design of a single classifier system involves three stages: the selection of features, the construction of decision algorithm and the final performance evaluation [152]. The design process of single classifier system is shown in Figure 6.1.

![Figure 6.1 Design process of a Single Classifier System](image-url)
But, no one classification technique is always superior to the others in terms of classification accuracy. So, the best classification algorithm for the classification task at hand is difficult to identify unless enough prior knowledge is available. The other reason is that for a specific recognition problem, there usually exist numerous types of features that may be too diversified to lump into one single classifier for decision-making.

For problems involving a large number of classes and noisy inputs, perfect solutions are often difficult to achieve. Nowadays, it has been observed that features and classifiers of different types complement one another in classification performance [153]. This has led to a belief that by using features and classifiers of different types simultaneously, classification accuracy could be improved. In recent years some multiple classifier combination techniques were proposed to improve the performance of classifiers [154-156]. However, the combination of potentially conflicting decisions by multiple classifiers remains an unsolved problem. Ideally, the combination function should take advantage of the strengths of the individual classifiers, avoid their weaknesses, and improve classification accuracy.

Multiple classifier combination is a technique of combining the decisions of different classifiers which are trained to solve the same problem, but make different errors. The multiple classifier is a set of single classifiers whose individual predictions are combined in some way to classify new objects. Its construction can be broken into two phases: (1) designing the type of base classifiers (2) deciding how to integrate their predictions.

A proper combination of multiple classifiers should produce more reliable recognition results than any of the individual classifiers. So, multiple independent approaches can be applied to a classification problem, each yielding its own prediction. The results of these techniques can then be combined. Roughly speaking, to design a Multiple Classifier System (MCS) is to combine several classifiers by one combination rule. The design process involves three stages: the construction of classifier ensemble, the construction of combination rule and the performance evaluation [152].
design process of MCS is shown in Figure 6.2.

![Figure 6.2 Design process of an MCS](image)

In both figures 6.1 and 6.2, there is a feedback loop from the last phase to either of the earlier ones. The meaning is obvious in an single classifier system. While in an MCS, it implies that the ensemble or combination rule must be reconstructed if the output performance evaluation does not meet the requirements. So multiple independent approaches can be applied to a classification problem, each yielding its own prediction to achieve high pattern-recognition performances [152]. There are three kinds of topology have been used in MCS [154]. They are

a) **Conditional Topology**
   Once a classifier is unable to classify the output then the following classifier is deployed

b) **Hierarchal Topology**
   The Classifiers are applied in succession with various levels of generalization

c) **Hybrid Topology**
   The choice of the classifier to use is based on the input pattern (selection).
   It is a multiple parallel topology.

Multiple Classifier Systems are involved to integrate base classifiers. Roughly, existing integration techniques can be distinguished into two categories. They are
6.2 Classifier Combination (CC)

In Classifier Combination, for each pattern, the classification process is performed in parallel by different classifiers and the results are then combined according to some decision “fusion” method (e.g., the majority-voting rule). The majority of such combination methods (CC) are based on the assumption that different classifiers make "independent" errors. However, in real pattern recognition applications, it is difficult to design a set of classifiers that should satisfy such an assumption.

Classifiers can be made in different ways to construct classifier ensembles. Different methods for creating diverse classifiers are

- **Varying the set of initializations**: A number of distinct classifiers can be built with different learning parameters, such as the initial weights in an MLP, etc
- **Varying the topology**: Using different topologies, or architectures, for classification can lead to different generalization models
- **Varying the algorithm employed**: Applying different classification algorithms for the same topology may produce diverse classifiers
- **Varying the data**: The mostly used approach to produce classifiers with different generalizations

The most commonly used method to create a set of classifiers for ensembling is by varying the data. Using the same training set, it is possible to make different collection of data in the following ways [155].

- **Sampling Data**: A common approach is to use some sort of sampling technique, such that different classifiers are trained on different subsets of the data.
- **Disjoint Training Sets**: Similar to sampling, however, uses mutually exclusive, or disjoint, training sets. That is we use sampling without replacement to avoid overlap between the training sets.
c. **Boosting and Adaptive Re-sampling:** A series of weak learner can be converted to a strong learner using boosting.

d. **Different Data Sources:** Under the circumstances the data from different input sources (e.g. sensors) are available. It is especially useful when these sources provide different sources of information.

e. **Preprocessing:** Data may be varied by applying different pre-processing methods to each set. Alternatively, data sets may be distorted differently.

Two approaches for constructing classifier ensembles seem to be perceived as classic at present. They are Bagging and Boosting. There are two sampling techniques have been developed. They are

a) Subspace methods

b) Sub-sample Method

### 6.2.1 Subspace method

Instead of sampling the training data, the subspace method samples the feature space. In this method, \( m \) features are randomly selected from the \( n \) dimensional feature vector and thus the \( m \)-dimensional random subspace of the original \( n \)-dimensional space is obtained \((m<n)\). Then classifiers are generated on the training sets, which are constructed by changing each example in the original training set into an \( m \)-dimensional vector. At last, all classifiers are combined by proper combination rule. When the size of training set is relatively small compared with the data dimensionality, subspace method is rather a better choice. The subspace dimensionality is smaller than that in the original feature space, while the number of training examples remains the same. Therefore, the relative size of the training set increases. When data have many redundant features, better classifiers may be obtained in random subspace than in the original feature space. The combined decision of such classifiers may be superior to a single classifier constructed on the original training set in the complete feature space.
6.2.2 Sub-sample Method

In sub-sample method, the samples are Re-sampled according to a given probability distribution. It constructs many different groups of samples by performing bootstrapping iteratively. Each group consists of a set of training samples and is collected by randomly and uniformly re-sampling of the original training set. Then this method applies a classifier to classify each group of samples. Finally, it performs some type of an average of the classifications of each group of sample via voting. Bagging and Boosting are sampling methods belonging to the technique of sub-sample.

They have been found to be accurate, computationally feasible across various data domains, and with no clear dominance between them. It has been shown both theoretically and empirically that an effective classifier ensemble should consist of base learners with high-accuracy as well as high-diversity in predictions. A common practice in building diverse base learners is to inject randomness into the learning algorithms or the training data, such that the underlying base learners can have their own uniqueness (diversity) in classification. Comparing to many mechanisms that help to construct diverse base learners, not much work has been done to improve the accuracy of each base learner. Intuitively, there are two possibilities to improve the accuracy of base learners: selecting a learning algorithm that fits the proposed dataset well or obtaining a training dataset with enhanced data quality. The former can be solved by applying cross-validation to each bootstrap dataset and then choosing a learner which outperforms others; and the latter could be accomplished by applying appropriate data preprocessing techniques. In the proposed work, a threshold based re-sampling technique has been introduced to improve the accuracy of base learners.

6.3 Classifier Selection

Select a single "best" classifier from base classifiers for the final decision, where each base classifier is evaluated with an evaluation set to explore its domain of expertise. When classifying an instance, only the "best" classifier is used to determine the classification of the test instance. In classifier selection, two types of techniques are usually adopted [157, 158].

109
6.3.1 Static Classifier Selection

The selection of the best classifier is specified during a training phase, prior to classifying a test instance.

6.3.2 Dynamic Classifier Selection

The choice of a classifier is made during the classification phase. We call it "dynamic" because the classifier used critically depends on the test instance itself.

6.4 Dynamic Combination of Multiple Classifiers System

Generally, two directions have been developed in the area of pattern recognition for improving classification performance. One is to improve the classification performance of a classifier itself. The other one is to improve a multiple classifier system which consists of a set of classifiers and a decision combination method. Although a number of classifiers are available, none of them is as good as expected. The major difficulty comes from the fact that although a number of features in diversified forms are available, but it is not easy to lump them together into a single classifier. The multiple classifier system is motivated from the assumption that a number of complementary features or classification algorithms can facilitate the classification performance, if they are used simultaneously. This research idea in combining multiple decisions seems to be promising.

In this thesis, the proposed approach to Multiple Classifiers System is a new Hybrid method in which selection and fusion techniques are combined in order to provide the most suitable output to classify the input pattern. The basic idea is to estimate each classifier's accuracy in local regions of feature space surrounding an unknown test sample, and then use the decision of the most locally accurate \( \lceil n/2 \rceil \) classifiers out of 'n' classifiers. In this implementation, "local regions" are defined in terms of the K-nearest neighbors in the training data. The accuracy of the classifier for the local region is estimated by simply calculating the percentage of training samples in the region that are correctly classified.
When constructing individual classifiers, the simple random sampling has been used. Based on the probability of the class label attribute in the entire training set, thresholds have been fixed for all the classes. When the data set have been selected randomly, the probabilities of the classes have been checked against the thresholds. If it does not satisfy the threshold Re-sampling is performed. The random set of samples, which satisfies the threshold is allowed to build the classifier and this classifier has been participated in DCMCS.

Given an unknown sample, the K nearest neighbor has been identified from the training set. The local accuracy is estimated for each classifier, and the decision of the classifier with the highest local accuracy estimate of \( \lfloor n/2 \rfloor \) classifiers out of ‘n’ classifiers are selected. Determining the appropriate size for a local region is part of designing the DCMCS approach. It is clearly seen that to identify a single value of “K” that can work well for all test patterns is very difficult. Accordingly, DCMCS performances were usually greatly affected by the choice of the “K” parameter. It is worth noting that the size of this neighborhood can vary with the test pattern, because it depends on the degree of similarity between the MCB of the test pattern and the ones of the K-nearest neighbors. Accordingly, it is possible that the number of K nearest neighbors is dynamically adapted to the test pattern. The appropriate value of the “K” parameter was decided by experiments. In particular, the “K” value is increased or decreased depending on the degree of similarity between the MCB of the test pattern and those of the neighboring training patterns. The experiments have been carried out for various region sizes ranging from K = 1 to K = 25 using the Euclidean distance metric.

In the proposed work, Multiple classifiers are selected dynamically based on the concepts of classifier’s local accuracy and multiple classifier behavior. For each unknown test pattern \( X^* \), the K nearest neighbors in the training, or validation, data are first identified. Let \( X \) indicate a K nearest neighbor of \( X^* \). Such patterns form the local region used to estimate CLAs. Let \( N(X^*) \) indicate such a local region surrounding test pattern \( X^* \). After the above selection of the patterns forming \( N(X^*) \),
for each classifier, the simplest method to estimate CLA is to compute the ratio between the number of patterns in \( N(X^*) \) that were correctly classified by the classifier \( C_j, j = 1, \ldots, L \), and the number of patterns forming \( N(X^*) \). Based on CLA, the best 2 classifiers out of 3 classifiers are selected to classify a new test instance \( X^* \). In general, dynamically choosing \( \lfloor n/2 \rfloor \) best performance classifiers out of \( n \) classifiers would result in better performance.

Voting is a method of decision making wherein a group of classifiers attempts to gauge its opinion. Majority voting attempts to collect the opinions from the different classifiers and make a decision based on the results of most of the classifiers. Majority voting is a very popular combination scheme both because of its simplicity and its performance on real data [159]. The decision of the selected classifiers for \( X^* \) are combined using majority voting method. The combination can be implemented using a variety of strategies, among which majority vote is by far the simplest, and yet it has been found to be just as effective as more complicated schemes in improving the recognition results.

Algorithm

**Procedure**: Dynamic Combination of multiple classifier system

**Phase 1**: Training Phase

**Input**: Training dataset \( T_r \)

**Output**: A set of \( n \) classifiers

1. Assume that there are \( m \) classes \( C_1, C_2, \ldots, C_m \) and the number of samples in each class is \( n_1, n_2, n_3, \ldots, n_m \) in the training set \( T_r \). The threshold \( T_1, T_2, T_3, \ldots, T_m \) for each class have been computed as

   \[
   T_i = 0.90 \times n_i \quad \text{where} \quad i = 1, \ldots, m
   \]

3. Select a random set of sample \( R \) from \( T_r \).

4. Compute the probability \( P_1, P_2, \ldots, P_m \) for the \( m \) classes in the random set of sample \( R \). If

   \[
   P_1 > T_1 \quad \text{and} \quad P_2 > T_2 \quad \text{and} \quad \ldots \quad \text{and} \quad P_m > T_m
   \]

then the random sample \( R \) is allowed to construct different classifiers.
Phase 2 : Testing Phase

Input : A set of n classifiers
        : A test dataset T_s.

Output : An integer value known as Selected-K which produced the most accurate result for the DCMCS.
        : The highest accuracy (High-Acc).

1. Assume High-Acc=0
2. For K = 1 to 25 {
3. For each sample X in T_s
4. {  
5. Find the K nearest neighbors X* for X in training set T_r.
6. The accuracies for the classifiers M_1, M_2, ..., M_n have been estimated using X*.
7. Select the most accurate n/2 classifiers out of n classifiers.
8. Classify X in each selected Classifier and the results of these classifiers are combined according to majority voting rule
9. The test instance X is assigned by a class label attribute based on the result of the majority voting rule.
10. }
11. Evaluate the accuracy Acc of Test sample T_s.
12. If Acc > High-Acc then
13. {  
14. High-Acc = Acc
15. Selected-K = K;
16. }
17. }

6.5 Experimental Results

6.5.1 Adult database

The individual class accuracy is defined as the percentage of samples of a particular class that were correctly classified by the classifier. The overall accuracy is
defined as the percentage of samples that were correctly classified by the classifier. Table 6.1 shows the accuracy of different classifiers for the individual classes and also the overall accuracy. It shows that the classification performance of DCMCS is superior to that of individual classifiers in class accuracies and also in the overall accuracy.

Table 6.1 Individual class accuracies in different classifiers for the Adult database

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Highest Accuracy Achieved by the random set of samples which doesn't satisfy the threshold</th>
<th>Accuracy Achieved by the random set of samples which satisfy the threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>64.918</td>
<td>79.76</td>
</tr>
<tr>
<td>KNN</td>
<td>56.87</td>
<td>80.45</td>
</tr>
<tr>
<td>Neural Network</td>
<td>67.04</td>
<td>84.53</td>
</tr>
<tr>
<td>DCMCS</td>
<td>69.34</td>
<td>85.17</td>
</tr>
</tbody>
</table>

When considering the individual classifiers, the neural network outperformed the other classifiers in class accuracies for the random samples selected in different iterations. The KNN classifier outperforms the decision tree classifier in class 2 accuracy and also in the overall accuracy. The decision tree produced more accuracy than KNN for class 1. Figure 6.3 compares the overall accuracies of different classifiers for the Adult database. It also shows that the accuracies achieved by the random set of samples which satisfied the threshold outperforms the highest accuracy achieved by the random set of sample which doesn't satisfied the threshold. In individual classifiers, 0.67% to 1.12% of accuracies and in DCMCS 2.41% of accuracies have been improved by the classifiers which have been made up of the random samples those were satisfied the threshold.
Figure 6.3 Comparison of overall accuracies in different classifiers for the adult database

Table 6.2 shows the TP and FP rate of best individual classifiers and the DCMCS algorithm. The number of times each individual classifier is selected by the DCMCS algorithm is also shown. The neural network classifier is selected more number of times by the DCMCS. The decision tree classifier is selected less number of times by DCMCS among the three classifiers. The DCMCS algorithm finds operating points at (0.156, 0.897). It gives higher TP rates and lower FP rates than points obtained by any individual classifier.

<table>
<thead>
<tr>
<th>Method of Classification</th>
<th>TP and FP rate</th>
<th>Overall Accuracy</th>
<th>Number of times selected by DCMCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree</td>
<td>(0.835, 0.182)</td>
<td>78.51</td>
<td>2606</td>
</tr>
<tr>
<td>K- Nearest Neighbor</td>
<td>(0.813, 0.203)</td>
<td>80.14</td>
<td>3476</td>
</tr>
<tr>
<td>Neural Network</td>
<td>(0.167, 0.834)</td>
<td>83.85</td>
<td>4365</td>
</tr>
<tr>
<td>DCMCS</td>
<td>(0.156, 0.897)</td>
<td>85.67</td>
<td>-</td>
</tr>
</tbody>
</table>

The number of times an individual classifier is selected by the DCMCS algorithm seems to be closely correlated to the overall accuracy of the classifier.
Results at other sensitivity levels show the same general trends. In general, since the DCMCS algorithm is attempting to lower the total number of misclassifications, it generates operating points which make the appropriate TP/FP tradeoff in order to drive the overall error rate down.

6.5.2 Earthquake database

Table 6.3 shows the accuracy of different classifiers for the individual classes and also in the overall accuracy. It shows that the DCMCS outperforms individual classifier in class accuracies and also in the overall accuracy. When considering the individual classifiers, the neural network outperformed the other classifiers in class accuracies for any set of random samples. The KNN classifier outperforms the decision tree classifier in class 2 accuracy and also in the overall accuracy. The decision tree produced more accuracy than KNN for class 1.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Highest Accuracy Achieved by the random set of samples which doesn't satisfy the threshold</th>
<th>Accuracy Achieved by the random set of samples which satisfy the threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>86.62</td>
<td>70.12</td>
</tr>
<tr>
<td>K-NN</td>
<td>90.09</td>
<td>57.26</td>
</tr>
<tr>
<td>Neural Network</td>
<td>95.51</td>
<td>50.94</td>
</tr>
<tr>
<td>DCMCS</td>
<td>96.13</td>
<td>72.65</td>
</tr>
</tbody>
</table>

Figure 6.4 compares the overall accuracies of different classifiers for the earthquake database. It also shows that the accuracies achieved by the random set of samples which satisfied the threshold outperforms the highest accuracy achieved by the random set of sample which doesn’t satisfied the threshold. In individual
classifiers, 0.25% to 1.17% of accuracies and in DCMCS 2.47% of accuracies have been improved by the classifiers which were made up of the random samples those satisfied the threshold.

![Diagram showing accuracies of different classifiers](image)

**Figure 6.4 Comparison of overall accuracies in different classifiers for the Earthquake database**

Table 6.4 shows the TP and FP rate of best individual classifiers and the DCMCS algorithm. The number of times each individual classifier was selected by the DCMCS algorithm is also shown. The neural network classifiers have been selected more number of times than the Decision tree classifier and KNN classifier. The decision tree classifiers are selected less number of times among the three classifiers. It shows that if the accuracy of a classifier is high then it has been selected more number of times by DCMCS. DCMCS algorithm finds operating points at (0.156, 0.897). It gives higher TP rates and lower FP rates than points obtained by any individual classifier. The number of times an individual classifier is selected by the DCMCS algorithm again seems to be closely correlated to the overall accuracy of the classifier. Results at other sensitivity levels show the same general trends. In general, since the DCMCS algorithm is attempting to lower the total number of misclassifications, it generates operating points which make the appropriate TP/FP tradeoff in order to drive the overall error rate down.
Table 6.4 Performance of individual classifier for the Earthquake Database

<table>
<thead>
<tr>
<th>Method of Classification</th>
<th>TP and FP rate</th>
<th>Overall Accuracy</th>
<th>Number of times selected by DCMCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree</td>
<td>(0.895, 0.106)</td>
<td>78.12</td>
<td>13147</td>
</tr>
<tr>
<td>K-Nearest Neighbor</td>
<td>(0.898, 0.183)</td>
<td>81.12</td>
<td>17612</td>
</tr>
<tr>
<td>Neural Network</td>
<td>(0.946, 0.047)</td>
<td>85.08</td>
<td>22927</td>
</tr>
<tr>
<td>DCMCS</td>
<td>(0.957, 0.042)</td>
<td>87.39</td>
<td>-</td>
</tr>
</tbody>
</table>

6.6 Conclusion

For this round of experiments, three individual classifiers are used for MCS. They are K Nearest Neighbor with the Euclidean distance metric, a fully connected backpropagation neural network and the Decision tree implementation.

For the MCS approach to be of practical use, it should improve on the best individual classifier, given that the individual classifiers have been reasonably optimized with regards to parameter settings and available feature data. In the proposed work, an earnest effort is made to optimize each individual classifier with respect to selecting "good" values for the parameters which govern its performance. For the KNN classifier, a value of K must be determined. For neural network classifier, the numbers of hidden layers and hidden nodes in a layer must be selected. The parameters for the Decision tree algorithm are selected based on the previous experience with this classifier. All features are used by all classifiers in experiments with these two data sets.

To test the extent to which the MCS results depend on the mix of individual classifiers, the DCMCS algorithm has performed the experiments for all possible combinations of three classifiers on the Adult data set and the earthquake dataset. The results are summarized in table 6.5.
Table 6.5 Accuracy of DCMCS Algorithm for the different number of classifiers

<table>
<thead>
<tr>
<th>Classifiers used as Input to DCMCS</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifiers used as Input to DCMCS</td>
<td></td>
</tr>
<tr>
<td>All 3 classifiers</td>
<td>Adult</td>
</tr>
<tr>
<td>Best two classifiers</td>
<td>85.03</td>
</tr>
<tr>
<td>Best single classifier</td>
<td>85.67</td>
</tr>
<tr>
<td>Best individual classifier</td>
<td>85.43</td>
</tr>
<tr>
<td>Best individual classifier</td>
<td>83.85</td>
</tr>
</tbody>
</table>

The DCMCS algorithm outperforms the best individual classifier in all cases. Interestingly, there exists a combination of two classifiers that is slightly superior than the combination of three classifiers for both the data sets.

Table 6.6 shows the static combination of two classifiers for the MCS. Among the three classifiers, the neural network classifier is identified as the best individual classifier and decision tree is identified as worst classifier. Removing the best individual classifier from the process of dynamic combination, results in a drop in performance in both the databases. Conversely, Removing the worst classifier from the process of dynamic combination, results in better performance in both cases.

Table 6.6 Accuracy of static Combination of two classifiers for the MCS

<table>
<thead>
<tr>
<th>Combination of two classifiers</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree, KNN</td>
<td>Adult</td>
</tr>
<tr>
<td>Decision tree, Neural Network</td>
<td>85.13</td>
</tr>
<tr>
<td>Neural Network, KNN</td>
<td>85.25</td>
</tr>
</tbody>
</table>

The static combination of Decision tree classifier and KNN classifier produces 85.13% accuracy for the Adult dataset and 86.82% for the Earthquake database. For
the Adult database, its accuracy is slightly better than the accuracy of all three classifiers and it is less than the best two classifiers in dynamic combination. But for the Earthquake database its accuracy is less than the dynamic combination of best two and all three classifiers.

In the static combination of Decision tree and the Neural network classifiers, the accuracy for the Adult database is 85.25 and for the Earthquake is 87.13. In this combination, for both the databases the accuracies are slightly superior than the combination of all three classifiers. At the same time, the accuracies for both the databases are less than the dynamic combination of best two classifiers. Likewise in the static combination of Neural Network and KNN classifiers, the accuracies for both the databases are slightly better than the combination of all three classifiers and slightly less than the dynamic combination best two classifiers.

Comparison of tables 6.5 and 6.6 shows that dynamically choosing the best two classifiers out of three classifier yield better performance than any other combinations. In general, dynamically choosing \(|n/2|\) best performance classifiers out of \(n\) classifiers would result in better performance.