The Hong Kong Polytechnic University

Department of Computing

Building a Decision Cluster Classification Model by a Clustering Algorithm to Classify Large High Dimensional Data with Multiple Classes

Yan **LI**

A thesis submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

February 2010

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree of diploma, except where due acknowledgement has been made in the text.

________________________ (Signed)

_____LI Yan_____________ (Name of Student)

Abstract

Clustering and classification are two basic tasks in data mining. As the complexity of data increases, the existing techniques for classification face a lot of challenges, for instance, classifying large high dimensional data with multiple classes. Therefore, new techniques need to be innovated to deal with data in large volume and high dimensions. In this thesis, we aim to propose a possible way to solve this problem by integrating clustering algorithm into classification work.

We propose a new classification framework. This framework consists of three phases: (i) a clustering algorithm is called recursively to build a decision cluster tree, (ii) a classification model is built from this decision cluster tree, (iii) new samples are classified by this classification model. There are many research problems existing in this framework. In this thesis, we describe our methodology for those problems.

In this framework, we propose a new classification method ADCC (Automatic Decision Cluster Classifier) that is designed to use a variable weighting k-means algorithm W-k-means to build a decision cluster tree so that the variable weights of each dimension can be obtained from the training data and used in classification. In partitioning the training data, W-k-means automatically computes the variable weights according to the data distributions so that important variables can get more weights and the noisy variables get less weight. In clustering a data set (i.e., a node), the class variable is removed from the data, so the class variable has no impact on the clustering results. The class variable is used in determining the dominant class for each cluster. To build a better cluster tree, effective methods for selection of the number of clusters and the initial cluster centers at each node are introduced. Furthermore, we use various tests including Anderson-Darling test to determine whether a node can be further partitioned or not. In this way, distribution of the training samples at each node is considered together with the purity and the size of the node. A decision cluster classifier consists of a set of disjoint decision clusters, each labeled with a dominant class that determines the class of new objects falling in the cluster. A series of experiments on both synthetic and real data sets have been conducted. The results show that the new classification method (ADCC) performed better in accuracy and scalability than the existing methods of KNN, decision tree and SVM. It is particularly suitable for large, high dimensional data with many classes.

Sometimes, ADCC method generates some weak decision clusters in which no single class dominates. Existence of weak decision clusters in the model can affect classification performance of the model. In a weak decision cluster, there is no dominant class, so it is difficult to justify the class of the new objects. It has been shown that classification accuracy could be improved after weak decision clusters were avoided from the model. Weak decision clusters occur because objects of different classes are mixed in the clustering process to generate decision clusters. If we assume that objects in the same class have their own cluster distributions, we can separate objects of different classes according to the object class labels and generate a decision cluster tree for each class of objects. Then, we combine the decision clusters of different classes to form the decision cluster classification model. In this way, weak decision clusters can be avoided. We propose a Decision Cluster Forest (DCF) method to build a set of decision cluster trees (decision cluster forest) which form a classification model. Instead of building a single decision cluster tree from the entire training data, we build a set of cluster trees from subsets of the training data set to form a decision cluster forest. Each tree in the forest is built from the subset of objects in the same class. The proposition for this method is that the objects in the same class tend to have their own spatial distributions in the data space. Therefore, decision clusters of objects in the same class are found. The decision clusters in the same tree have the same dominant class. In this way, no weak cluster is created in such decision cluster tree. A decision cluster model can be selected from the set of leaf decision clusters from the decision cluster forest so the model is called a decision cluster forest classification model (DCFC). The decision cluster forest method has advantages of classifying data with multiple classes because the DCFC model is guaranteed to contain decision clusters in all classes. DCFC model is a more intuitive and direct multi-class classification method.

We propose a different classification method based on the tree structure. We propose a Crotch Ensemble classification model for high dimensional data with multiple classes. Generated from a decision cluster tree, a crotch is an inner node of the tree together with its direct children. If the dominant classes of children of a crotch are not all the same, the crotch is defined as a crotch predictor that is a classifier by itself. A crotch ensemble consists of a set of crotch predictors. When classifying a new object, a subset of crotch predictors is selected according to the distances between the object and the crotches. A classification is made on the object as the class predicted by the crotch predictors with the maximum accumulative weights. The experimental results on both synthetic and real data have shown that the Crotch Ensemble model is efficient and effective when classifying new samples.

We propose a special application of our framework in text data classification. A subspace clustering algorithm is integrated to build the decision cluster tree. We adopt cosine distance metric for this application. Experimental results have shown that our framework can integrate different clustering algorithms and other possible methods and can get better classification results for text classification.

Finally, we give the theoretical analysis of error bound of our DCC model. We prove that our Cluster-based classification model (DCC model) is better than the Object-based classification method.

List of Publications

The following technical papers have been published or are currently under review based on the result generated from this work.

1. **Yan Li**, Edward Hung, Korris Chung. A Subspace Decision Cluster Classifier for Text Classification. Submitted to Expert Systems with Applications. Under review (submitted on 10 Feb. 2010) ;

2. **Yan Li**, Zhaocai Sun, Joshua Huang, Yunming Ye, Edward Hung. An ensemble of decision cluster crotches for classification of high dimensional data. Submitted to *Pattern Recognition Letters*. Review result is major revision. (submitted on 1 Aug. 2009);

3. **Yan Li**, Edward Hung, Korris Chung, Joshua Huang. Using A Variable Weighting k-Means Method to Build A Decision Cluster Classification Model. Submitted to *International Journal of Pattern Recognition and Artificial Intelligence*. Under review (submitted on 19 Feb. 2009);

4. **Yan Li**, Edward Hung, "Building A Decision Cluster Forest Model to Classify High Dimensional Data with Multi-classes", Proceedings of *The 1st Asian Conference on Machine Learning (ACML'09)*, LNAI 5828, pp. 263–277, Nanjing, China, November 2-4, 2009. (acceptance rate = 27 / 113 = 23.9%);

5. **Yan Li**, Edward Hung, Korris Chung, Joshua Huang, "Building A Decision Cluster Classification Model for High Dimensional Data by A Variable Weighting k-Means Method", Proceedings of the *Twenty-First Australasian Joint Conference on Artificial Intelligence*, pages 337-347, Auckland, December 1-5, 2008. (acceptance rate = 42/143 = 29%).

6. Yunming Ye, Xutao Li, Biao Wu, **Yan Li**, “Feature Weighting Information-Theoretic Co-clustering for Document Clustering”, Proceedings of the *2nd International Conference on Computer Science and its Applications (CSA 2009)*, pages 2-2, Korea, December 10-12, 2009.

7. Zhaocai Sun, Zhi Liu, **Yan Li**, “Kernal-based Decision Cluster Classifier”, *ICIC express letters*, vol.4 (4): 1223-1229, 2010.

Acknowledgements

I would like to take this opportunity to express my heartful gratitude to everyone who generously gave me their support and kindness.

First of all, I would like to express my gratitude and appreciation to my Supervisor, Dr. Edward Hung, for his continuously patient guidance throughout my research studies. My Ph.D. studies would have never been completed without his encouragement, help, and wonderful suggestions. He is precise for all things including research work, teaching work as well as the details in the daily grind. His attitude and methodology for research and other all things not only contributed to my Ph.D. studies, but also influences all my life.

My special thanks is for my co-supervisor Dr. Korris Chung. He gave me a lot of good comments and suggestions. He encouraged me and help me to do research during my Ph.D. studies. I also want to thank Prof. Joshua Zhexue Huang. He was my teacher during my Master studies. He gave me many good ideas and methodologies for my Ph.D. program. From them I have learned quite a lot. They were always ready to give advice and help.

I would like to thank Mr. Xiaoguang Xu and Dr. Zhaocai Sun for their good comments and kind help. We have cooperated many research works. I also really appreciate the open and friendly environment at Department of Computing. It provides comfortable study environment including soft resource and hard resource. All teachers and students are friendly and accommodating.

At last, I would like to express my deepest thanks to my parents, my husband and my daughter. Their love, constant support and encouragement made me what I am.

Table of Contents

Lists of Figures

Lists of Tables

Chapter 1

Introduction

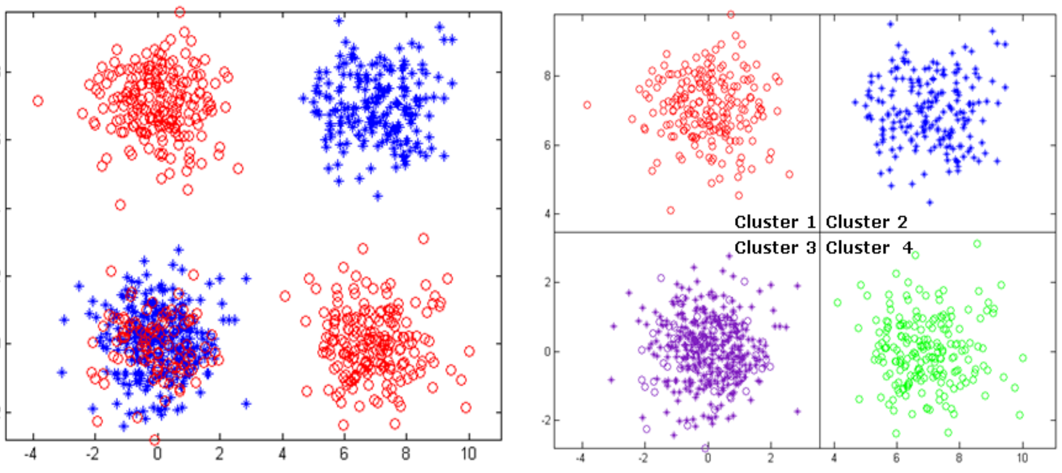
This chapter introduces the research problems and the motivations of this thesis. We also address the originality, contribution and the organization of this thesis.

1.1 Problem Statement

As complexity of data increases, the existing classification techniques face a lot of challenges, for instance, the Grand Challenge data mining problems proposed in the recent KDD Panel Report [21]. Therefore, new techniques need to be innovated to deal with large, high dimensional data with multiple classes. Such data occur in many application domains such as text mining, multimedia mining and bio-informatics.

Clustering and classification are two basic tasks in data mining. Classification is a supervised learning method which builds a model from training data first and then labels unknown data with the model. Clustering is an unsupervised learning method. Clustering is a process of partitioning a set of objects into clusters to make that objects in the same cluster are more similar or closer to each other than objects in different clusters. Classification and clustering have been extensively studied in data mining, machine learning, statistics and pattern recognition, but they are seldom considered together. Clustering methods have some advantages that classification methods do not have. How to integrate and make use of the advantages of clustering and classification is a meaningful research problem. Our work is an attempt to integrate clustering into classification techniques to deal with classification problems.

We will use an example to explain the advantages of our Cluster-based classification which uses decision clusters to classify new objects. This example includes the classic exclusive-Or (XOR) problem as shown in Fig. 1.1(a). There are 900 objects distributed in 4 clusters that are classified into 2 classes marked with '○' in red and '*' in blue respectively. The bottom left cluster is mixed by 200 '*' and 100 '○'. Other three clusters include 200 objects respectively. SVM methods map the data to another feature space by a kernel function so that a linear hyperplane can be found to separate the objects from two classes with highly computation cost. User cannot understand the internal details and working principles of the SVM classifier. Our Cluster-based classification method deals with the problem from a clustering perspective instead of the number of classes in the data set. Our method partitions the whole data set into four clusters without considering two classes. Fig. 1.1(b) shows that our method partitions the data set into four clusters denoted by four different colors respectively. Each cluster has a dominant class as its class label. If the new objects are closer to the cluster 3, they will be classified as cluster 3's dominant class '*'. Any classification algorithm cannot classify cluster 3 with precision higher than 75% due to its own data distribution (its purity is 75%).



(a) the data set with XOR problem (b) space partition by Cluster-based classification

Fig. 1.1 An advantage of Cluster-based classification.

From this example, we can sum up the advantages of our Cluster-based classification method as follows: it is easy to understand; it permits impure decision clusters to save computational cost, so it dose not have overtraining problems and has high robustness; simple partition methods can work under this framework.

1.2 Motivations

Our goal is to find a new classification method which integrates advantages of clustering and classification. Our motivation mainly includes three aspects: clustering can be used for data reduction and data sampling; objects in the same cluster tend to be in the same class; some clustering algorithms are efficient for large high dimensional data.

Figure 1.2 shows the data distribution on three dimensions of the iris data set from UCI repository [23]. The data points are drawn in different color according to their class labels. We can see that, the points in the same class are closer to each other and form a cluster. That is also shown that the objects in the same cluster tend to be in the same class.

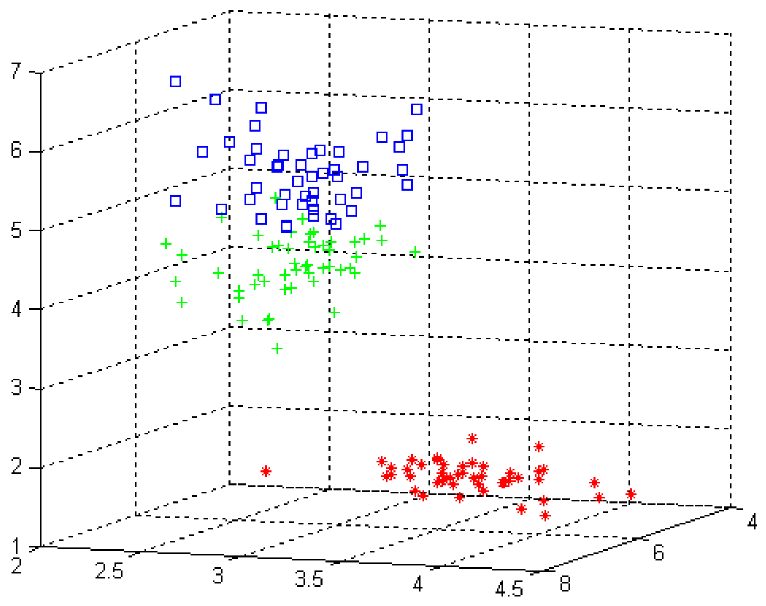


Fig. 1.2 The relationship between cluster and class of iris data set.

In conclusion, our thesis aims at solving some classification problems (e.g. high dimensional sparse problem, XOR problem) from a new angle. Our Cluster-based classification framework integrates the clustering and classification techniques together to combine their advantages.

1.3 Statement of Contributions

Figure 1.3 illustrates the main work of this thesis as well as the corresponding contributions which are claimed to be original as follows:

1. A novel classification framework integrating clustering algorithm into classification is presented. Under this framework, a clustering algorithm is called recursively to build a decision cluster tree or forest. Based on the decision cluster tree or forest, a classification model is specified. This classification framework includes three steps: tree (forest) construction, model selection and classification.
2. An Automatic Decision Cluster Classification (ADCC) method is proposed, where, the weighted k-means clustering algorithm is adopted to build the decision cluster tree because it is efficient for large data sets and it can reduce the influence of noisy attributes by assigning them smaller weights.
3. A Decision Cluster Forest Classification (DCFC) method is developed to deal with the weak decision cluster problem and the multi-classes problem. Instead of building a single decision cluster tree from the entire training data, this method builds a set of decision cluster trees from subsets of the training data set to form a decision cluster forest. Each tree in the forest is built from a subset of objects in the same class.
4. Text data is a typical high dimensional sparse data. Subspace clustering algorithm is efficient for text data. A Subspace Decision Cluster Classification (SDCC) method is designed for text classification.
5. A set of decision clusters are selected from the decision cluster tree or forest plus a specific distance metric as a classification model.
6. Another model selection, named Crotch Ensemble, is introduced. Instead of

considering a set of decision clusters, this model considers all crotch predictors which are inner nodes with their children.

7. A KNN-like classification step is implemented on the first kind of classification model. This kind of Cluster-based classification is proved to be better than Object-based classification.

8. An experimental scheme is designed to demonstrate the performance of this series of classification methods under the decision cluster classification framework.

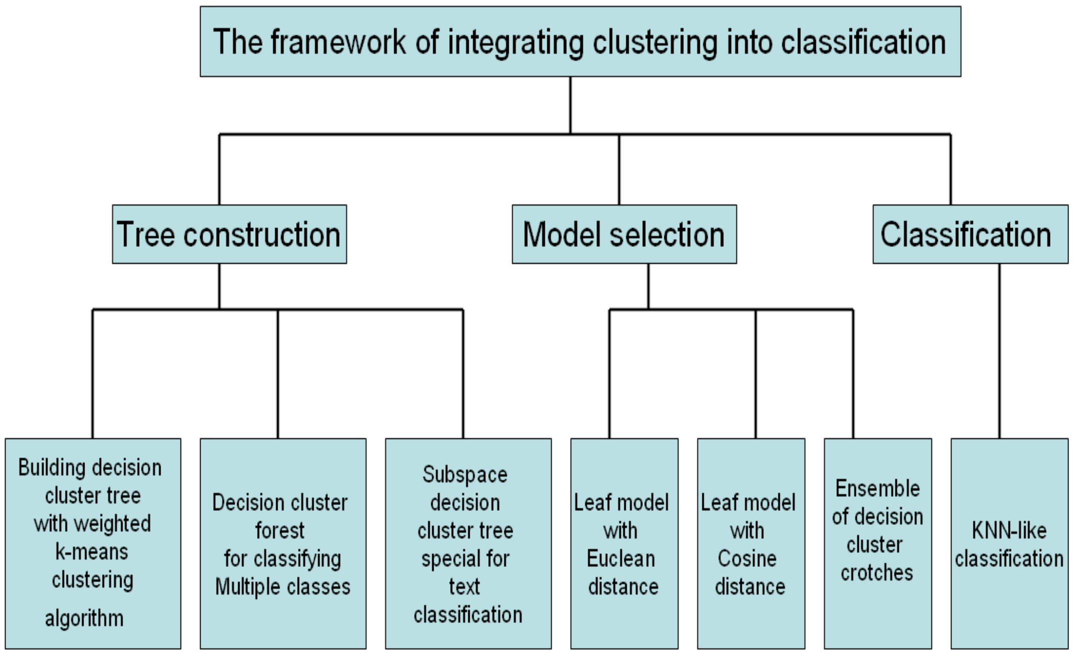

The framework of integrating clustering into classification
Tree construction
Model selection
Classification
Building decision cluster tree with weighted k-means clustering algorithm
Decision cluster forest for classifying Multiple classes
Subspace decision cluster tree special for text classification
Leaf model with Euclean distance
Leaf model with Cosine distance
Ensemble of decision cluster crotches
KNN-like classification

Fig. 1.3 The framework of this thesis.

1.4 Organization

Following the introduction, the thesis proceeds in Chapter 2 with a literature review of classification techniques, clustering algorithms and the integration of clustering and classification. Among clustering algorithms, special attention is given to k-means type algorithms which are used in our new classification methods.

The subsequent chapters (Chapters 3-9) present the research contributions of this thesis. All the chapters correspond to work that has been published or is currently under review.

In Chapter 3, we propose a new classification framework. Under this framework, a clustering algorithm is called recursively to build a decision cluster tree; a classification model is selected from this decision cluster tree; new samples are classified by this classifier with a KNN-like way.

We then propose the first classification model under the new framework in Chapter 4. The new classification method ADCC (Automatic Decision Cluster Classifier) is designed to use a variable weighting k-means algorithm W-k-means [2] to build a decision cluster tree so that the variable weights of each dimension can be obtained from the training data and used in classification. In partitioning the training data, W-k-means automatically computes the variable weights according to the data distributions so that important variables can get larger weights and the noisy variables get smaller weights. In clustering a data set (i.e., a node), the class variable is removed from the data, so the class variable has no impact on the clustering results. The class variable is used in determining the dominant class for each cluster. Effective methods for selection of the number of clusters and the initial cluster centers at each node are introduced to build a better decision cluster tree. Furthermore, we use various tests including Anderson-Darling test [6] to determine whether a node can be further partitioned or not. In this way, distribution of the training samples at each node is considered together with the purity and the size of the node. A decision cluster classifier (DCC) consists of a set of disjoint decision clusters, each labeled with a dominant class that determines the class of new objects falling in the cluster. A series of experiments on both synthetic and real data sets have been conducted to show that the new classification method (ADCC) performs better in accuracy and scalability than the existing methods of KNN, decision tree and SVM. It is particularly suitable for large, high dimensional data with many classes.

In Chapter 5, we propose a Decision Cluster Forest (DCF) method to build a decision cluster forest. We also present the method to construct a classification model from the decision cluster forest. Instead of building a single decision cluster tree from the entire training data, we build a set of decision cluster trees from subsets of the training data set to form a decision cluster forest. Each tree in the forest is built from the subset of objects in the same class. The proposition for this method is that the objects in the same class tend to have their own spatial distributions in the data space. Therefore, decision clusters of objects in the same class are found. The decision clusters in the same tree have the same dominant class. In this way, no weak cluster is created in such decision cluster trees. A decision cluster classification model can be selected from any subset of leaf decision clusters in the decision cluster forest, so the classification model is called a decision cluster forest classification model (DCFC). The decision cluster forest method has advantages of classifying data with multiple classes because the DCFC model is guaranteed to contain decision clusters in all classes. DCFC model is a more intuitive and direct multi-class classification method and easy to use.

In Chapter 6, we propose a Crotch Ensemble classification model for high dimensional data with multiple classes. Generated from a decision cluster tree, a crotch is an inner node of the tree together with its direct children. If the children of a crotch have more than one dominant class, the crotch is defined as a crotch predictor that forms a classifier by itself. A Crotch Ensemble consists of a set of crotch predictors. When classifying a new object, a subset of crotch predictors is selected according to the distances between the object and the crotch predictors. A classification is made on the object as the class predicted by the crotch predictors with the maximum accumulative weights. The experimental results on both synthetic and real data have shown that the Crotch Ensemble model is efficient and effective when classifying new samples.

In Chapter 7, we propose a special application of our framework in text data classification. A subspace clustering algorithm (Entropy Weighting k-Means) is integrated to build a decision cluster tree. We adopt cosine distance metric for this application. Experimental results have shown that our framework can integrate different clustering algorithms and other possible methods and can get better classification results for text classification.

In Chapter 8, we analyze why our DCC model (Cluster-based classification) is better than KNN method (Object-based classification). The error bound is also discussed in this chapter.

Finally, we give the conclusions of our work in Chapter 9, where some suggestions for further work are also discussed.

Chapter 2

Literature Review

In this chapter, we briefly review the research on classification methods, clustering methods and the integration of classification and clustering. In the first part, we review some well-known classification methods. The traditional classification methods face a lot of challenges including how to classify large high dimensional data. We will compare our new approach with some well-known classification methods reviewed in this chapter. The text data is a classic example of high dimensional data. Text classification algorithms are also briefly reviewed. In the second part, we review the clustering methods. In our new approaches, we integrate some of them into our work. The third part reviews the research works which integrate clustering and classification.

2.1 Review of Classification

Classification has been extensively studied in data mining, machine learning, statistics and pattern recognition. Many classification approaches have been investigated in the literature, including KNN, decision tree induction, rule-based classifier, instance-based classifier, artificial neural networks, Support Vector Machine (SVM) etc.

2.1.1 K-nearest Neighbors (KNN)

K-nearest neighbors (KNN) [34] is a simple technique to build classification models. However, it cannot perform classification well on large data sets because of high computational cost. This is because KNN is an instance-based classification method and it uses all of the training objects in classifying new objects. For data set with many classes, it requires a sufficient coverage of cases from all classes in the training data in order to produce accurate classification results. Therefore, such KNN models will be computationally and spatially expensive in classifying new data. Another problem is that when the number of classes is large, it becomes tricky to select the neighborhood parameter k. There is some research work done to solve KNN's time consuming problem [24]. Accelerating the KNN classification process to handle large data through reduction of instances is another research direction [35-38]. A detailed survey can be found in [46, 47].

2.1.2 Decision Tree

A decision tree classifier uses the 'divide and conquer' and greedy strategies to construct an appropriate tree from a given training data set. In a decision tree, each internal node denotes a test on a non-class attribute, where each branch represents an outcome of the test. Each leaf node denotes a class or a class distribution. A path traced from the root to a leaf node represents a classification rule. How to select a test attribute and how to partition a sample set are the key parts of the decision tree construction. Different decision tree algorithms adopt different techniques. For example, C4.5 algorithm adopts gain ratio [25] as the attribute selection standard while CART algorithm adopts Gini index [25].

In dealing with large and complex data sets, decision tree techniques are widely used due to their high efficiency. A majority of work on decision trees in data mining is an extension of Quinlan's ID3 and C4.5 [27]. Tree induction is the core process in decision tree classification. Most existing tree induction methods proceed in a greedy top-down fashion, where they start with an empty tree and the entire data set, and recursively split the data set in internal nodes based on specific split metrics. The models are often based on a small number of dimensions even if the data has many dimensions such as text data. This is because that each partitioning step in building a decision tree model only considers one dimension while the information is usually stored among many dimensions. There are other possible drawbacks of the decision tree classification algorithm [61]: when there are large number of classes, the number of leaves become larger and will result in overlapping problem; the incorrect classification results will accumulate and be passed to deeper levels; it is difficult to design an optimal decision tree for classification.

According to the number of attributes used in split metrics, decision tree induction methods can be classified into two categories: univariate tree induction and multivariate tree induction [27]. The former finds a split according to a single attribute which is recognized as the most useful in discriminating the input data set. In case of multivariate tree induction, finding a split can be seen as finding a composite attribute (combination of existing attributes) that has a good discriminatory capability. In this perspective our ADCC method of building classification trees can be regarded as a multivariate tree induction technique. Multivariate tree induction is not as widely studied as univariate decision trees. Most multivariate decision tree methods consider oblique trees which have tests based on a linear combination of the attributes at some internal nodes. Existing methods include linear discriminant trees [28, 29], hill climbing methods [30, 31], Neural trees [32, 33] etc. However, most methods are not scalable in dealing with large data sets. Our method differentiates from these approaches in that it employs weighting k-means type algorithm as the split function, and only selects leave nodes from the tree to induce the final classification model.

2.1.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) [148] is a new and effective classification method. Since the first paper presented by Vladimir Vapnik in 1992 [149], SVM has been widely used in many applications, such as handwritten digit recognition [150], face recognition [151], text classification [108], gene pattern classification [152] etc. Margin is a key concept in SVM, which measures the separation of two classes. For linearly separable data, the key problem of linear SVM algorithms is to find a separation hyperplane that can lead to maximum margin. The hyperplane with maximum margin is called maximum marginal hyperplane (MMH), which generates an optimal separation of two classes. MMH can be further defined as support vectors so that the problem of searching MMH can be mapped to the problem of finding support vectors. Searching of support vectors is a constrained quadratic optimization problem and can be solved by classical non-linear programming tools and the application of the Karush-Kuhn-Tucker (KKT) Sufficiency Theorem. For non-linear data classification, a feature space transformation step will be required before performing the quadratic optimization process (searching the support vectors in linear classification cases). The basic idea is that the linearly inseparable data can be transformed into higher dimensional space using a nonlinear mapping (such as some kernel functions). The resulted data will be linearly separable in the new dimensional space, where a simple linear SVM algorithm can be employed to learn the classification model. In spite of many successes in various applications, SVM has some intrinsic disadvantages. First, the performance of classification algorithm is sensitive to the selection of the kernel function and its parameters [153], where different data sets will require diverse parameter settings to get good results. This is undesirable in real applications, since searching the best parameter is very difficult if not impossible, due to the high computational complexity of SVM. Secondly, SVM classifiers usually work as a black box and it’s hard for users to understand the internal details. This characteristic limits its applications to some critical areas, such as medical diagnosis, where the interpretable property is essential. Moreover, original SVM can only solve two-class classification problem. For multi-class data, many two-class SVM classifiers will need to be learned by pairwising combination or Directed Acyclic Graph (DAG) mode [154], which will decrease the classification accuracies in many real applications and require more computational cost. Finally, for those data sets with mixed distribution of different class, SVM can not find an appropriate hyperplane to separate the objects of different classes.

2.1.4 Text Classification Methods

In recent years, with the advent of the Web and the enormous growth of digital content in Internet and the availability of more powerful hardware, text classification (TC) has been given more attention and more researches [69]. As the development of hardware, data collection and data storage, a lot of classification techniques to TC have been explored in the literature, such as the Bayesian method [70-72,94,95], k-nearest neighbors (KNN) [73-75,96], decision tree [87-90,98-100], artificial neural networks [78-81,97], support vector machines (SVM) [83-85,101,102,108], centroid-based approaches [93,103-107], and some other algorithms [76,77,82,86,91,92].

Naive Bayes is a common text classification method that is computationally efficient and easy to implement [95]. Rennie and Shih's work [70] discusses several systemic problems of Bayes text classification algorithms and proposes possible solutions to avoid those problems. Naive Bayes algorithms often adopt two event models: the multinomial event model and the multivariate Bernoulli event model. The multinomial event model is frequently referred as multinomial Naive Bayes (MNB) for short [72]. Ref. [72] presents a multinomial Naive Bayes model for text classification and shows that the multinomial Naive Bayes model outperforms the multivariate Bernoulli model. Makoto and Takenobu's work [71] proposes a cluster-based text classification method, where training documents are partitioned into several clusters before classifying new documents. A new document is compared with each cluster rather than with each training document. In this work, Hierarchical Bayesian Clustering (HBC) algorithm is employed to construct a set of clusters for cluster-based classification [71].

KNN method is a case-based learning method which requires large computational cost for calculating similarities between each new testing document and each training document. Section 2.1.1 has already discussed KNN in details. Memory Based Reasoning (MBR) [73] is a typical KNN method. Ref. [73] proposes to use MBR to classify news stories, which does not require manual topic definitions. Expert Network is proposed in [74] to categorize natural language documents automatically. In this method, a set of training documents with expert-assigned classes are used to construct a network that reflects the conditional probabilities of classes assigned to a document. In the expert network, the training documents form the nodes on the intermediate level, the words generated from the training documents form the input nodes, and the classes are the output nodes. A word node (input node) and a document node (intermediate node) are connected if this word occurs in this document. A document node (intermediate node) and a class node (output node) are connected if this class is assigned to this document by experts similarly. The relevance rankings of classes given to a new testing document are evaluated by those connections. A new classifier called the KNN model-based classifier (KNN Model) is proposed in [96]. KNN Model improves the standard KNN by avoiding the critical problems of KNN, including lazy learning problem and selecting appropriate value of k.

Transductive Support Vector Machines for text classification were introduced in [108] which are more suitable for text classification than regular Support Vector Machines. SVM is an effective technique to build classification models from text data. However, its computational complexity prohibits it from being used on very large training data.

Centroid-based text classification methods have the advantages of small computational cost in both training stage and testing stage. However, it is difficult to locate optimal centroids. A fast Class-Feature-Centroid (CFC) classifier for multi-class text classification is designed in [93]. In CFC, centroids are built from inter-class class distribution and inner-class class distribution. This centroids selection method incorporates inter-class and inner-class term distribution and constructs better initial centroids than traditional centroids. Furthermore, CFC adopts a denormalized cosine metrics to calculate the distance between a document and a centroid. Experimental results show that CFC outperforms SVM and other centroid-based text classification methods. CFC is more robust than SVM on sparse data. In [107], Tam et al. have compared centroid-based text classification methods with neighborhood-based and statistical text classification methods.

2.2 Review of Clustering

Clustering is another important problem in data mining. Clustering is a process of partitioning objects into clusters to make that objects in a cluster are more similar than objects in different clusters. There are a lot of clustering algorithms in the literature. In general, clustering algorithms can be classified into the following types: Partitioning algorithms, Hierarchical algorithms, Model-based algorithms, Density-based algorithms and Grid-based algorithms. In recent years, some new types of clustering algorithms have been proposed, such as subspace clustering algorithm, co-clustering algorithm. K-means [134], k-Medoids [133] and k-Prototype[18] are classical Partitioning algorithms. CLARA (Clustering LARge Applications) [133] and its improved version CLARANS (Clustering LArge Applications based upon RANdomized Search) [135] were proposed to deal with the problem that k-medoids algorithms cannot work efficiently for larger data sets. BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) [136], ROCK (RObust Clustering using linKs) [137,138] and CURE (Clustering Using REpresentatives) [139] are three typical hierarchical clustering methods. Density-based clustering algorithms have been proposed to discover arbitrary shape clusters. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [140], OPTICS (Ordering Points To Identify the Clustering Structure) [141] and DENCLUE (DENsity-based CLUstEring) [142] are density-based clustering algorithms. Some typical grid-based clustering algorithms include STING (Statistical Information Grid) [143], WaveCluster [144] and CLIQUE (Clustering In QUEst) [131]. Model-based clustering algorithms try to find the best fit between the given data and some mathematical model. COBWEB [145] is an early model-based clustering algorithm.

Among so many clustering techniques, k-means type algorithms are the most popular methods due to their efficiency and scalability in clustering large data sets. K-prototype [18] algorithm can deal with categorical data. Many new clustering methods, such as weighting-k-means (WKM) [2] and Entropy Weighting k-Means (EWKM) [3], extend the k-means algorithm and get better clustering results. Weighting-k-means (WKM) [2] can find out noisy attributes as well as normal attributes by assigning smaller weights to noisy attributes. It is suitable for high dimensional data with noisy attributes. It is worth to take better use of k-means type algorithm in data mining.

High dimensional and sparse data is very common in real world. Text data is a typical example of this kind of data. In these kinds of data, the clusters always exist in a subset of attributes not in the whole attributes space. Subspace clustering methods are designed to find clusters in different subsets of attributes. Ref. [58] compares many subspace clustering methods. The famous subspace clustering algorithms include PROCLUS [146], FASTDOC [12], HARP [147], CLIQUE [131], ORCLUS [11] and EWKM [3]. An Entropy Weighting k-Means (EWKM) algorithm [3], a kind of soft subspace clustering algorithm, can do subspace clustering on high-dimensional sparse data sets. EWKM clusters data samples in the entire dimensional space but assigns different weights to different dimensions for each cluster during clustering process. The dimensions which are more important for identifying the corresponding cluster will get larger weights. Different dimensions make different contributions to the evaluation of objects in a cluster. WKM is not a subspace algorithm. It just assigns smaller weights for noisy dimensions and larger weights for non-noisy dimensions. All clusters have the same weight distribution of the entire data space in WKM. However, EWKM can find clusters in subspaces by giving different weight distributions for different clusters while WKM fails to do this.

Co-clustering, also called bi-clustering [63], is to partition data matrix to small sub-matrix by clustering rows and columns. General clustering is to find similar rows (objects) using distance metrics computed with columns (attributes). Co-clustering has recently received a lot of attention in several practical applications such as in genes and experimental conditions in bioinformatics [65], text mining [64], recommender systems [66], etc. In [59], a partitioned co-clustering formulation that is driven by the search for a good matrix approximation was introduced. They introduce a new minimum Bregman information (MBI) principle. MBI generalizes the maximum entropy and standard least squares principles simultaneously, and leads to a matrix approximation that is optimal among all generalized additive models in a certain natural parameter space.

None of the existing clustering algorithms can perform well on all kinds of data sets. Different clustering algorithms are conformable for different kinds of data sets. Some research work done in clustering algorithms focused on the problems of large data sets, high dimensions, and large number of classes [8-14,16,17]. Ref. [8] reviews and compares many algorithms for clustering large data sets. In their work, Steinbach et al. [9] analyze the challenges of clustering high dimensional data from two aspects: different clustering tools and different types of data. Cai et al. propose a novel clustering algorithm [10] for high-dimensional document clustering, which aims to cluster the documents into different semantic classes. The document space is always in high dimension. The high-dimensional documents can be projected into a lower dimensional space by using Locality Preserving Indexing (LPI). Projective clustering algorithms [11-14] have been successfully utilized to deal with high-dimensional data clustering problems.

2.3 Integration of Classification and Clustering

Clustering methods can be applied to supervised classification problems. Several clustering-based classification techniques have been explored [5,15,19,22,35]. Mui and

Fu's work [19] was an early example of using an interactive approach to building classification models. The basic procedure was to use a k-means algorithm to partition the training data into k clusters without using the class variable. Then, for each cluster, the percentage of each class was calculated according to the class variable. If one class was found to be dominant in the cluster, say over 90%, this class label was assigned to this cluster as its class and the cluster was taken as a leaf without further partition. If no dominant class was found, the cluster was further partitioned with the k-means algorithm. This process continues until all clusters became leaves. The centers of the leaf clusters together with the dominant classes of these leaf clusters formed a KNN-like classification model. When a new object was classified, the distances between the new case and the cluster centers were computed, the dominant class of the center with the shortest distance to the new object was assigned to the new case. Since the number of leaf clusters was much smaller than the number of training cases in a KNN model, this cluster center based model was more efficient than the KNN model. In their work, Zhang et al. [35] propose a cluster-based tree algorithm for accelerating KNN classification. This cluster-based tree algorithm consists of two steps: tree construction and classification. They evaluate the effectiveness in comparison with the standard KNN and the condensation-based tree algorithm. Kyriakopoulou and Kalamboukis's work [22] is a comparatively new trial to integrate clustering into classification. In this work, they propose a new classification algorithm with clustering which consists three steps: clustering step in which both training and testing data set are clustered; Expansion step in which the data set is augmented with meta-features originated from the clustering step; Classification step in which a classifier is trained based on the expanded data set. This algorithm can learn a classifier from small training data set by combining supervised learning with semi-supervised or unsupervised learning methods. The learned classifier can exploit the distribution information of testing data before classifying them.

The Cluster-based classification model follows a probability mixture model in which each cluster is considered as a distribution of objects of one class in the multi-dimensional data space. As such, the instances in the same cluster tend to have the same class label. Classification can be considered as a high level model that can map one class to more than one cluster. In this way, classification can be viewed as a clustering problem that can be solved with a clustering process.

Zhexue Huang and Michael K.Ng proposed decision clusters classifier in 2000 [1]. This decision cluster classifier model adopted the k-prototype clustering algorithm to construct a decision clusters tree using the interactive Fast Map algorithm. Then they use a KNN-like algorithm to perform classification. Building the model and validating the clusters are based on human beings' judgment which is not automatic. Compared to these interactive approaches, our Automatic Decision Cluster Classifier (ADCC) is different from them: ADCC proposes an automatic hierarchical clustering method which uses some new clustering algorithms; ADCC will consider the class label during the tree construction to specify the number of sub-clusters; ADCC uses termination test methods not human beings' judgment to decide whether the current cluster is a leaf or not. Other clustering algorithms, other distance metrics or other possible methods can be integrated into Decision Cluster Classifier model. In the following chapters, a series of new classification methods are proposed under the Decision Cluster Tree framework.

Chapter 3

Decision Cluster Tree Framework

In this chapter, we present a new classification framework. In this framework, a clustering algorithm is used to build a decision cluster tree. Based on this tree, a decision cluster classification model will be specified. At last, new objects are classified through a KNN-like way.

3.1 Framework

Fig. 3.1 demonstrates the idea of Decision Cluster Tree framework.

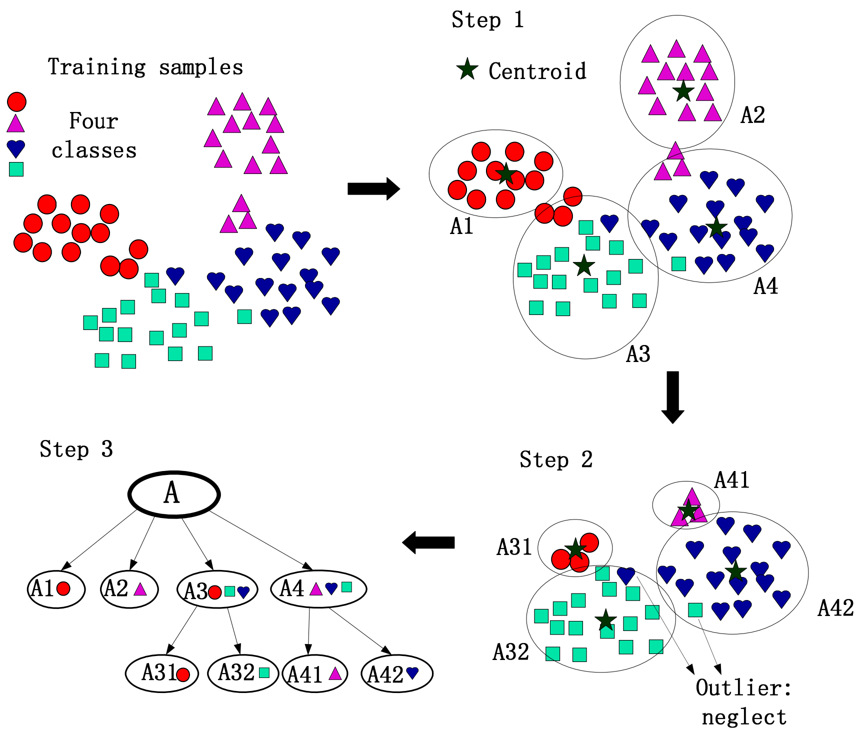


Fig. 3.1 The process of constructing a decision cluster tree.

The details of the process in Fig. 3.1 are as follows. There is a training data set including four classes. Different shapes denote different classes. For the first iteration, we partition the whole training data set into four clusters A_1, A_2, A_3 and A_4. All samples in A_1 belong to the same class, so A_1 does not need to be further partitioned and becomes a leaf node. It is the similar case for A_2.But A_3 and A_4 need to be partitioned in the following iteration. We assume that both A_3 and A_4 should be partitioned into two clusters according to a method which is used to specify the number of sub-clusters. A_3 is then partitioned into A_{31} and A_{32}, and A_4 is partitioned into A_{41} and A_{42}. A_{31}, A_{32}, A_{41} and A_{42} become leaf nodes according to our termination test method. Finally there are six leaf nodes A_1, A_2, A_{31}, A_{32}, A_{41} and A_{42}. The construction of decision cluster tree is completed. Fig. 3.2 presents the classification model generated from decision clusters and the process of classifying a new data.

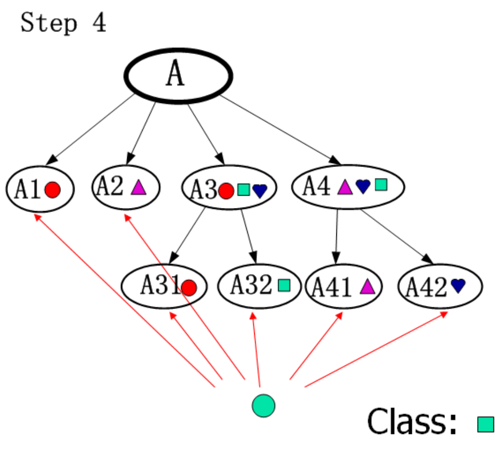


Fig. 3.2 Classify a new object with the leaf nodes.

In Fig. 3.2, we select all leaf nodes from the decision cluster tree as the classifier model. Each leaf is generalized by its center and the most frequent class label. These centers are the training objects for the final KNN-like classification step. For the new object, we compute the distance between it and each center of the node of the classifier model. We assign the new object to the class of the nearest cluster. We call this process KNN-like classification. In the above example, the new object is classified as the dominant class of A_{32}.

In short, this new classification framework mainly includes three steps: tree construction, model selection and classification.

3.2 Preliminary Definition

Before we discuss the details of the decision cluster classifier framework, we describe some preliminary definitions [1] and present our research problems and the goal. These definitions will be used in the rest of this thesis.

A training data set $X = \{x_1, x_2, \cdots, x_n\}$ contains n samples with m attributes and k classes. A cluster $C \subseteq X$ such that the all samples in C belong to X and fulfill certain criteria of similarities.

Definition 1. A clustering or k-clustering of X is a partition of X into k clusters $C_1, C_2, ..., C_k$, which satisfies: $C_i \neq \varnothing$, $i = 1, \cdots, k$, $\bigcup_{i=1}^{k} C_i = X$, and $C_i \cap C_j = \varnothing$, $i \neq j$, $i, j = 1, \cdots, k$. The clustering can be represented as a sequence $X(C_1, C_2, ..., C_K)$.

When generating a clustering from the training data, the class labels of objects are removed first so the objects are clustered without using the label information.

Definition 2. The dominant class in a cluster is the class that the majority of objects are labeled to. A cluster with a dominant class is called a decision cluster. The percentage of the dominant class in the cluster defines the confidence level of the decision cluster.

Definition 3. A decision cluster classification (DCC) model is a subset of decision clusters plus a defined distance function. The distance function is used to compute the distances between an object to be classified and the selected centers of decision clusters.

Definition 4. A clustering S with k clusters is said to be nested in the clustering T, which contains r $(< k)$ clusters, if for any cluster C_i (i = 1,…,k) in S, there is a cluster C_j (j=1,…,r) in T such that $C_i \subseteq C_j$ and there exists at least one cluster C_i in S, which holds $C_i \subset C_j$ and $C_i \neq C_j$.

Definition 5. A Decision Cluster Tree is a sequence of nested clusterings, so that for any two level p, q with p < q (i.e., level p is high than level q) and for any cluster C_j^{q-1} in level q, there is a cluster C_i^{p-1} in level p such that $C_j^{q-1} \subset C_i^{p-1}$.

Figure 3.3 shows an example of a decision cluster tree of four levels. A decision cluster tree consists of a root which is the training data set X at level 0, a number of internal nodes and a number of leaf nodes. It is like decision tree, an internal node is an intermediate outcome; a following decision will be made whether it needs to be further divided and its immediate sub-clusters represent the output of the just decision.

The root X is partitioned into 3 clusters C_1^0, C_2^0, C_3^0. Here, the superscript indicates the level of the node from which the clusters are generated and the subscript is the cluster number in this level. Clusters C_1^0 and C_3^0 are further partitioned into 2 and 3 sub clusters respectively, which form level 2 of the cluster tree. Subsequently, two clusters C_2^1 and C_4^1 are further partitioned into three and two sub clusters respectively, which form level 3 of the cluster tree. This tree can be represented as the sequence of nested clusterings as:

$$X(C_1^0(C_1^1, C_2^1(C_1^2, C_2^2, C_3^2)), C_2^0, C_3^0(C_3^1, C_4^1(C_4^2, C_5^2), C_5^1)).$$

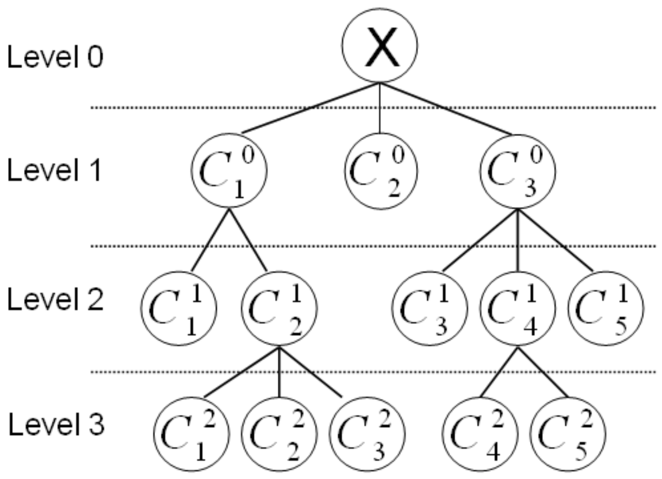


Fig. 3.3 A decision cluster tree.

3.3 Research Problem

In this thesis, we solve the classification problem by building decision cluster tree with clustering algorithm. We call this kind of classification method Cluster-based classification. The idea of Cluster-based classification is shown in Fig. 3.4. It is different from the traditional Object-based classifications which train the classifier from the class labels of training data directly. Cluster-based classification builds many decision areas from clustering training data without class labels recursively. Every decision area has a dominant class. The objects in the same class can be distributed in more than one decision areas.

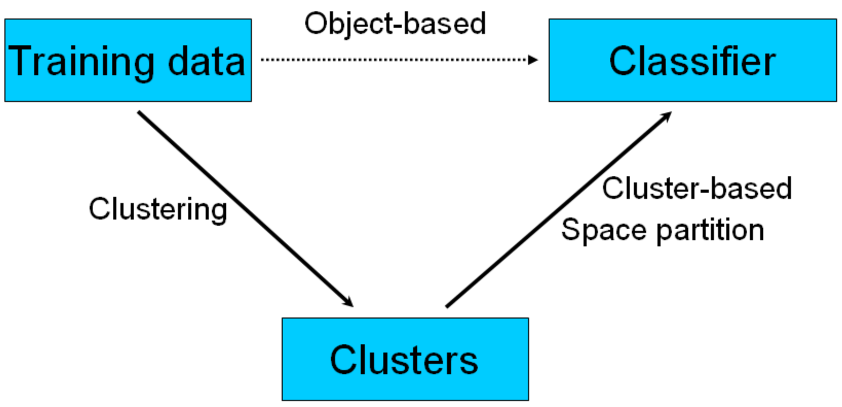


Fig. 3.4 The difference between Cluster-based classification and Object-based classification.

In the kind of Cluster-based classification, the classification problem is considered from another aspect instead of the probabilities. Clusters often exist in the areas with big conditional probabilities. This kind of Cluster-based classification is based on the Bayesian Theory as well as Bayesian Classifiers, but it is based on the clusters not based on estimating conditional probabilities. Under Cluster-based classification framework, we have the following assumptions: objects in the same class tend to have the similar characteristics and be close to each other; objects in the same cluster tend to be in the same class. In our method, we want to integrate the advantages of clustering with the advantages of classification. We formulate our method as follows:

1. Building decision cluster tree(s) T by calling clustering algorithm on the training data S:

$$\Delta : S \mapsto \mathrm{T} \tag{3-1}$$

2. Selecting decision clusters C which are used to construct the classifier from T:

$$\Lambda : \mathrm{T} \mapsto C \tag{3-2}$$

where $C_i \cap C_j = \varnothing$, $\bigcup_{i=1}^{K} C_i = S$, $i \neq j$, K is the number of all decision clusters in classifier.

3. Building decision areas P from C with space partition method to partition the whole data space R^d:

$$\Psi : C \mapsto P \tag{3-3}$$

where $P_i \cap P_j = \varnothing$, $\bigcup_{i=1}^{K} P_i = R^d$, $i \neq j$. In fact, P is corresponding to C.

4. Building classifier M to classify new objects X.

$$\mathrm{M}_{\Delta,\Lambda,\Omega}(x) = \tau(C_i \mid x \in P_i) \tag{3-4}$$

where $\tau(C_i)$ is defined in Equation (3-5).

$$\tau(C_i) = \arg\max_{\omega} \sum_{x_j \in C_i} I(W(x_j) = \omega) \tag{3-5}$$

In Equation (3-5), $W(x_j)$ is the class label of x_j. Any testing object will be classified as the dominant class of the decision area in which the object falls. In this thesis, we present a series of research works related to the above problems.

Definition 6. A Classification Model is either (1) a subset of disjoint decision clusters, or (2) a sub-tree pruned from T, in which it has the same root with T. The first kind of model is named as Sample-based Model, and the second kind of model is named as Tree-based Model. They are all following the principle of Cluster-based classification.

Definition 7. Model Selection is a process of selecting a classification model M from T to maximize the classification accuracy.

Definition 8. Model Complexity is to describe the complexity of a classification model. The model complexity metric adopted in our work is the number of nodes in a classification model.

In principle, any subset of decision clusters or any sub-tree from a decision cluster tree can be treated as a classification model. However, the classification accuracy of a concrete classification model depends on which classification model is used. If the classification accuracies of different classification models are identical, then the simplest one is the best choice. Therefore, the following processes are crucial: (1) generation of a decision cluster tree and (2) selection of a classification model.

So our problem can be described as follows: Given a training data set S and a clustering algorithm f, generating the decision cluster tree T by recursively calling f to partition the nodes and calling termination test method to test whether to end the partition or not. At last, select a best classification model M to execute classification. Table 3.1 shows the research problems under the decision cluster tree framework.

In fact, we want to find an optimal integration of clustering methods, termination test methods, model selection methods, classification methods and other possible solutions for a special problem. Following the Decision Cluster Tree framework, we can build classifiers for different kinds of data sets. In the following chapters, we will discuss the techniques used to solve these problems.

Table 3.1 Research problems.

Step	Research problems
Tree construction	Which clustering algorithms can be implemented?
	How to do the termination test?
	How to specify the number of attributes used in the clustering process?
	How to specify the number of sub-clusters of an internal node in decision cluster tree?
Model selection	How to do the model selection?
	A subset of decision clusters or a sub-tree?
	What other model selection methods can be referenced? （boosting, bagging, random forest）
	Which model is best for the concrete problem?
	Select one or many?
	Interactive or automatically?
Classification	How to classify unknown data?
	Which distance metric can be used?
	KNN-like or decision tree-like?

Chapter 4

Using A Variable Weighting k-Means Method to Build A Decision Cluster Classification Model

In this chapter, a new classification method Automatic Decision Cluster Classifier (ADCC) for high dimensional data is proposed. In this method, a decision cluster classification (DCC) model consists of a set of disjoint decision clusters, each labeled with a dominant class that determines the class of new objects falling in the cluster. A decision cluster tree is first generated from a training data set by recursively calling a variable weighting k-means algorithm. Then, the DCC model is extracted from the decision cluster tree. Various tests including Anderson-Darling test [6] are used to determine the stopping condition of the tree growing. A series of experiments on both synthetic and real data sets have been conducted. Experimental results show that the new classification method (ADCC) performed better in accuracy and scalability than existing methods like KNN, decision tree and SVM. ADCC is particularly suitable for large, high dimensional data with many classes.

4.1 Introduction

In Chapter 3, we have proposed a novel Cluster-based classification framework which integrates clustering into classification. Under this new framework, we propose many classification methodologies. In this chapter, we present the first Decision Cluster Classification model which uses a variable weighting k-means clustering algorithm.

Classification is a basic task in data mining. As complexity of data increases, the existing techniques for classification face a lot of challenges, for instance, solving the Grand Challenge data mining problems proposed in the recent KDD Panel Report [21]. Therefore, new techniques need to be innovated to deal with large, high dimensional data with multiple classes. Such data occur in many application domains such as text mining, multimedia mining and bio-informatics. This chapter proposes an Automatic Decision Cluster Classifier (ADCC) that is designed to achieve that goal.

Clustering methods have been applied to supervised classification problems [5,15,22,35]. An early example of using the k-means clustering method to build a cluster tree classification model was given in [19], where, a binary cluster tree was built by interactively executing the k-means clustering algorithm. At each node, a further partition was determined by the percentage of the dominant class in the cluster node. However, only small numeric data could be classified and every time only two sub-clusters are formed. In 2000, Huang et al. proposed a new interactive approach to build a decision cluster classification model [1]. In this approach, the k-prototypes clustering algorithm was used to partition the training data, and a visual cluster validation method [60] was adopted to verify the partitioning result at each cluster node. The above two interactive methods are not adequate for high dimensional data with noisy attributes because the clustering algorithms used are not able to handle noisy attributes and it is time consuming to involve human judgment.

In this chapter, we propose to use the variable weighting k-means (W-k-means) algorithm [2] to build a Cluster-based classification model automatically. Because W-k-means is able to reduce the impact of noisy attributes by assigning smaller weights to them in clustering. In another word, W-k-means implicitly performs attribute selection in the clustering process. Meanwhile, the weight information can also be used in classification to improve the classification quality. As such, W-k-means is more suitable for high dimensional data with noisy attributes. Another improvement from the previous methods is that in the tree growing process we use various tests including Anderson-Darling test to determine whether a node can be further partitioned or not. In this way, distribution of the training samples at each node is considered together with the percentage of the dominant class used in the previous methods [1]. Anderson-Darling test replaces the visual cluster validation method as in [1] so as to automate the tree building process.

A series of experiments on both synthetic and real data sets were conducted to demonstrate the effectiveness and the accuracy of the ADCC method. Compared with other classification methods, including KNN [34], J48 [27] (a decision tree algorithm) and SMO [124] (one of SVM algorithms), our experimental results show that the ADCC method has performed better than other methods in both classification accuracy and scalability on large high dimensional sparse data sets. Thus the results demonstrate that the ADCC method is more suitable for large, high dimensional data with many classes.

This chapter is accordingly organized as follows. In Section 4.2, we present the details of the ADCC algorithm. In Section 4.3, we show the experimental results on synthetic data sets and real data sets. Concluding remarks are given in Section 4.4.

4.2 Automatic Decision Cluster Classification Method

In this section, we demonstrate the techniques during the construction process of a decision cluster tree. The W-k-means [2] algorithm was adopted to build a decision cluster tree because it is efficient and able to automatically compute the attribute weights from the training data to reduce the effect of noisy attributes. The number of sub-clusters and the initial centers must be specified before executing W-k-means. The ADCC algorithm uses the function $K-Selection(X,\alpha)$ to return the number of sub-clusters for current cluster, where X is the current cluster and α is the threshold. $K-Selection(X,\alpha)$ computes the percentage of each class and return k as the number of classes with a percentage greater than α. Then, we compute the real centers of the k classes as the initial centers. This is called supervised selection which, instead of random selection, is implemented in the ADCC algorithm as the function $C-Selection(k, X)$, where k is the number of sub-clusters generated by $K-Selection(X, \alpha)$.

4.2.1 ADCC Algorithm

Table 4.1 shows the algorithm of automatic construction of decision cluster tree under the above Cluster-based classification framework.

Table 4.1 ADCC Algorithm.

```
Input: A training data set $T$ (with $m$ dimensions and $c$ classes).
Output: A classification model $ADCC-\text{model}$.
1.  initialize a decision cluster tree $DCT$ with root $\{T\}$;
2.  sign the root as internal node;
3.  for each internal node $X$ in $DCT$
4.          if ( $\text{Terminal}-\text{Test}(\text{X})$ ) sign $X$ as leaf node;
5.          $k = K-Selection(X, \alpha)$;
6.          $\text{CENTER}-\text{ARRAY} = C-Selection(k, X)$; ///Compute initial centers
7.          run W-k-means on $X$ with $k$ and $\text{CENTER}-\text{ARRAY}$;
8.          sign $k$ sub-clusters as internal node;
9.          add $k$ sub-clusters into $DCT$;
10.  end for
11. extract all leaf nodes from $DCT$ as classification model $ADCC-\text{model}$
    and represent each node by its center and dominant class;
12. return $ADCC-\text{model}$;
```

4.2.2 Constructing a Decision Cluster Tree with W-k-means

The traditional k-means algorithm treats all attributes equally. It is well-known that a meaningful cluster usually exists in a subset of all attributes. W-k-means can automatically weigh attributes on their importance during the clustering process [2]. W-k-means introduces a new step to the basic k-means algorithm to update the variable weights based on the current partition of data. A weight calculation formula is used to minimize the objective function of clustering given a fixed partition of data. The important attributes can get larger weights while the insignificant attributes will get smaller weights. The effect of noisy attributes can be reduced by smaller weights. For the data set with noisy attributes, W-k-means outperforms the standard k-means algorithm [2].

In our work, the construction of the decision cluster tree is a recursive division process by recursively executing the W-k-means clustering algorithm. To partition a cluster into sub-clusters with W-k-means algorithm, we need to specify a parameter k which is the number of sub-clusters to be generated. We also need to specify the initial centers for each sub-cluster. Here, we take advantage of the class information. We propose some methods to control the iteration and improve the clustering process. These methods include the method of selecting k for W-k-means ($K-Selection(X,\alpha)$, the method of selecting initial centers for W-k-means ($C-Selection(k,X)$) and the termination test method ($\mathrm{Terminal}-\mathrm{Test}(\mathrm{X})$), where X is the current node to be partitioned, α is a threshold and k is the number of sub-clusters. They correspond to solving the three problems at the end of chapter 3, i.e., (1) How many clusters to be generated at each node? (2) How to specify the initial centers? (3) Where to stop at each path of the tree?

The Selection Method for k in W-k-means

Here, k is the number of sub-clusters of the current cluster (node) X in a decision cluster tree. The value of k is very crucial for W-k-means algorithm as it will influence not only the following iteration steps of W-k-means algorithm but also the final classification accuracy. In the following we will discuss the problem of how to determine the value of k. We give an example to explain our method:

Suppose the current cluster X has five classes (C_i, i=1,...,5) as shown in Table 4.2. We do not simply set k to 5, but we set k to 3 since there are mainly only three classes of samples added with some noises from C_4 and C_5. This method can reduce the impact of noisy data.

Table 4.2 The class distribution of X.

	C_1	C_2	C_3	C_4	C_5
Objects	300	200	100	5	5
Percentage	0.49	0.32	0.16	0.008	0.008

We determine the value of k by considering the distribution of classes. We compute the percentage of samples in each class compared with all samples in the current node. Given a threshold α, let k be the number of classes whose percentages are larger than or equal to α. Consider the above example. If α equals to 0.1, C_1, C_2 and C_3 have the percentage values larger than α, so we will set k as 3 instead of 5.

We implemented the selection of k by the function $K-Selection(X,\alpha)$: X is the current sample set and α is the threshold. The function returns the value of k.

The Computation Method for Initial Centers in W-k-means

W-k-means algorithm is a local search approximation algorithm. The final result depends on the initial centers. Therefore, how to specify the initial centers is critical for the clustering accuracy. If we can specify better centers in the beginning, it can reduce the number of iterations and get better clustering result more quickly.

Currently, there are two methods for determination of initial centers: random selection and density-based selection [20]. The random selection method leads to the various and unstable clustering result because the centers are determined randomly. The density-based selection needs to search the whole sample space to compute the density, so it decreases the efficiency vastly.

In this thesis we present a new method called **supervised selection** method. Under this method, we compute the centers of objects of each class using the information of class label. These class centers are used as the initial centers for W-k-means algorithm. We implemented this method by $C-Selection(k,X)$: X is the current cluster to be partitioned and k is the number of sub-clusters generated by $K-Selection(X,\alpha)$. We first compute and store the percentage of each class to the whole data of current cluster. We then select the first k classes (with the largest percentages) to compute the k class centers as the output of the function $C-Selection(k,X)$. Using the class centers can accelerate the step of selecting the initial centers as well as improve the accuracy of determination of initial centers vastly. The efficiency of this new method will be shown in the experiment section.

Termination Test Method

Coming to an intermediate node in decision cluster tree, it is critical to determine whether it is terminal or not. Not terminal means that it should be further partitioned. We call the testing process as termination test or stopping test. This stage is vital for the whole tree construction and will influence the quality of the tree as well as the quality and computing efficiency of the classifier. We use multiple termination conditions: the size, class purity and data distribution to determine whether a node will be further partitioned or not.

Cluster size: If the cluster size is too small, this cluster should be a leaf node and labeled with the dominant class label. This principle can avoid overfitting problem.

Cluster purity: If the cluster size is big enough, the cluster purity should be further estimated. If the percentage of the most frequent class is bigger than the critical value, this cluster node should be a leaf node and labeled with the dominant class label.

Data distribution: Objects in a cluster intend to follow a normal distribution. If we can test how well the objects in a node follow the normal distribution, we can determine how likely they belong to a cluster. The data distribution can be tested by a distribution test method: Anderson Darling (AD) Test [6,7]. AD Test can determine whether a sample comes from a specified distribution. The AD Test was invented by Wilbur Anderson and Donald A. Darling in 1952. It is one of the most powerful statistical methods for detecting the most departures from normality based on the following. Given a hypothesized underlying distribution, the data can be transformed to a uniform distribution. The transformed data can then be tested for uniformity with a distance test [125]. If the testing result satisfies the AD Test criteria, this cluster node should be a leaf node and labeled with the dominant class label, otherwise this cluster node should be further partitioned into sub-clusters.

We implemented a termination test method considering the above three aspects by $\mathrm{Terminal-Test}(\mathrm{X})$ which is shown in Table 4.3. $\mathrm{AD-Test}(\mathrm{X})$ returns true if the distribution of X is almost likely normal distribution. $\mathrm{Terminal-Test}(\mathrm{X})$ needs two parameters, δ and β. δ is the size threshold of a cluster and depends on the size of the smallest class which includes the fewest samples compared with other classes in the whole training data set. β is the purity threshold to judge whether a cluster is pure enough not to be divided. $\mathrm{Terminal-Test}(\mathrm{X})$ judges the size and purity first before doing $\mathrm{AD-Test}(\mathrm{X})$ because $\mathrm{AD-Test}(\mathrm{X})$ needs more time and judging the size and purity can filter some obvious cases. We label the cluster with the dominant class (the most frequent class in a cluster) if it is a leaf node.

Table 4.3 Algorithm of Terminal-Test(X).

```
Input: the node $X$ contains $n$ objects.
Output: A Boolean value Termi which is either (i) TRUE which means "stop" or
        (ii) FALSE, otherwise.
Remarks:
δ: the threshold of the number of samples in $X$ (e.g., 10).
β: the threshold of the frequency of a class in $X$ (e.g., 90\%).
Begin
1.  Termi = FALSE;
2.  if (n <δ OR the frequency of the dominant class >β OR $\mathrm{AD-Test}(X)$
3.          Termi = TRUE and label $X$ with the dominant class label;
4.   return Termi;
End
```

4.2.3 Model Selection and Classification

After a decision cluster tree is built, any subset of disjoint decision clusters makes a DCC model. There are many ways to select classification models from a decision cluster tree. In this work, we select the leaf nodes of the decision cluster tree, because leaf nodes are disjoint with each other and all of them as a whole cover all training samples.

The classification model is used to classify new objects as the following: (1) Define a distance function specific for classification; (2) Compute the distances between a new object and the centers of the decision clusters in the model; (3) Identify the decision cluster with the shortest distance to the object. Assign the label of the decision cluster to the new object as its class.

In this work, we use the weighted Euclidean distance function as follows:

$$d = \sqrt{\sum_{i=1}^{n} (w_i (x_i - y_i))^2} \tag{4-1}$$

In Equation (4-1), w_i is the weight for the i th attribute. The weights are computed when we cluster the training data by the W-k-means clustering algorithm. Since the weight distribution is different when we cluster a different node, here we adopt the weight distribution computed when we cluster the root of the decision cluster tree. The effectiveness of this weighted distance metric will be shown in Section 4.3.2.

4.3 Experiments

In this section, we describe the experiments we have conducted on both synthetic and real data sets. The experimental results demonstrate that ADCC outperforms the original KNN [34] in terms of speed, scalability and classification accuracy vastly. We also conduct experiments to compare ADCC to other classification methods, including decision tree (J48) and SVM (SMO). Weka [126] implementations of these algorithms were used in our comparisons. The synthetic data sets and real data sets used in the experiment will be described below.

We have implemented ADCC system in java and conducted a series of experiments. The resulting ADCC system also integrates the original KNN algorithm for comparison. We executed ADCC and the original KNN on this platform, and executed other classification algorithms on Weka.

4.3.1 Experiment Setup

All experiments were done on Windows XP platform running on an Intel (R) Xeon(R), 1.60 GHz computer with 8GB memory. We conducted experiments on synthetic data sets and real data sets. We adopt the accuracy and execution time as the evaluation methods to compare the classification quality to other classification algorithms. The experiments on synthetic data sets are used to demonstrate that our model is effective and efficient. Meanwhile, we ran experiments on real data sets to compare our methods to other classification methods in speed, scalability and classification quality.

In our experiments, we use default parameters for J48 and SMO in Weka. For J48, the minimus number of instances per leaf is set as 2. The SMOs are trained with a linear kernel where the complexity parameter C is set as 1.0. For KNN, the number of neighbors, k, is equals to 1. For ADCC, we always set α (the parameter in $K - Selection(X, \alpha)$ which was presented in Section 4.2.2) equal to 0.05, δ (a parameter in $\mathrm{Terminal} - \mathrm{Test}(\mathrm{X})$ which is presented in Section 4.2.2) equal to the 10% of the number of samples in smallest class, and β (another parameter in $\mathrm{Terminal} - \mathrm{Test}(\mathrm{X})$ which is presented in Section 4.2.2) equal to 90%.

4.3.2 Experiments on Synthetic Data

We conducted experiments on synthetic data sets to demonstrate the efficiency and scalability of our methods and to compare the classification performance with other classification algorithms (KNN, decision tree (J48) and SVM (SMO)) on spatial data sets.

Through the experiments described here, we want to (1) verify that our method of selecting initial cluster centers is efficient and can get better classification results; (2) demonstrate that ADCC can identify noisy attributes and reduce their influences by using W-k-means as the decision cluster tree construction algorithm; (3) show that the weight information generated during the decision cluster tree construction is useful to improve the classification quality; (4) demonstrate that the KNN-like classification method in ADCC outperforms original KNN in classification speed and scalability. The details will be described in the following parts.

1. Supervised selection of initial cluster centers

The ADCC model includes $K-Selection$ and $C-Selection$ (presented in Section 4.2) algorithms to specify the number of sub-clusters and the initial centers in W-k-means. It has good capabilities of reducing the influence of noisy attributes and noisy samples. We generate the following synthetic data set T, which includes 3 numerical attributes (X, Y and Z), 8 classes and 3,400 samples. The data set are uniformly distributed on the attribute Z, Z is the noisy attribute. The details of T are listed in table 4.4:

Table 4.4 Synthetic data set T.

Class	Center	Variance	Instances
1	(0,4,2)	0.5，0.5，2.309	500
2	(2,3,2)	0.2，0.2，2.309	200
3	(4,4,2)	0.5，0.5，2.309	500
4	(1,2,2)	0.2，0.2，2.309	200
5	(3,2,2)	0.2，0.2，2.309	200
6	(0,0,2)	0.5，0.5，2.309	500
7	(2,1,2)	0.2，0.2，2.309	200
8	(4,0,2)	0.5，0.5，2.039	500
Noisy samples			600

Fig. 4.1 shows the distribution of T on the subspace X and Y, where different classes are represented with different colors. From the Table 4.4 and Fig. 4.1, we found that there are 8 clusters with normally distribution on X and Y. The circle denotes the real centers for each cluster. There are some noisy samples existed on the subspace X and Y. We set the class labels for noisy samples randomly. These noisy samples are used to test the efficiency of the selection method of initial cluster centers.

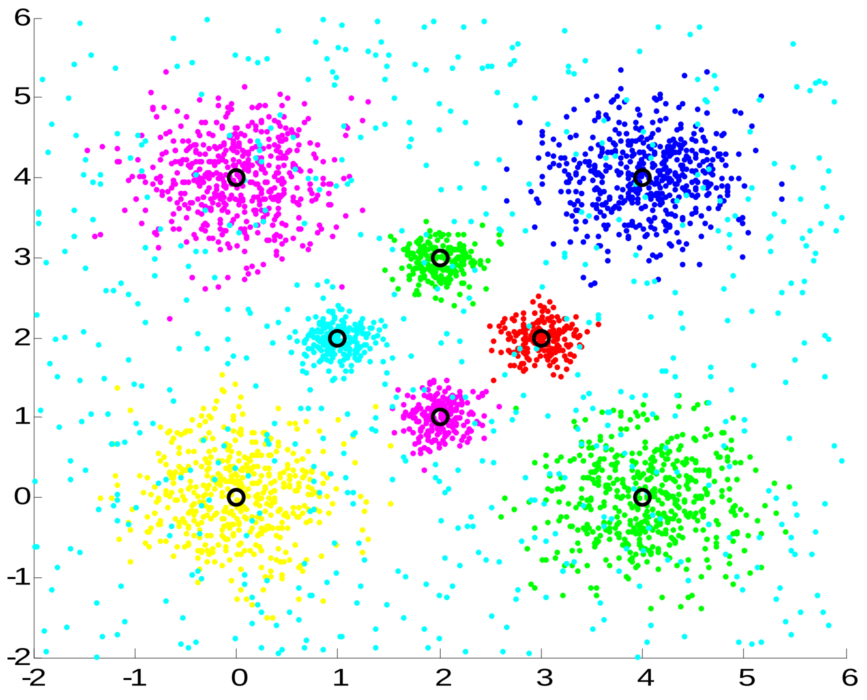
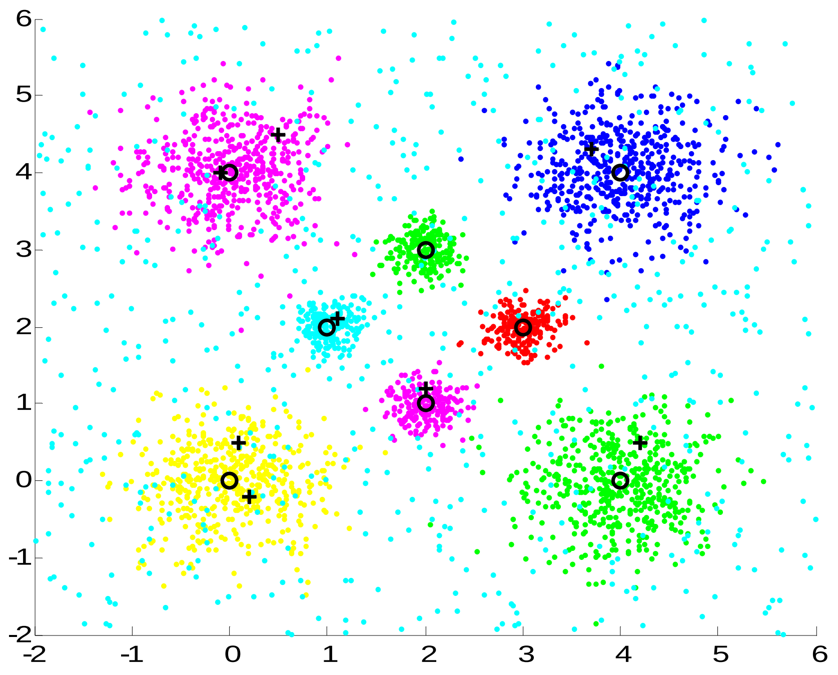
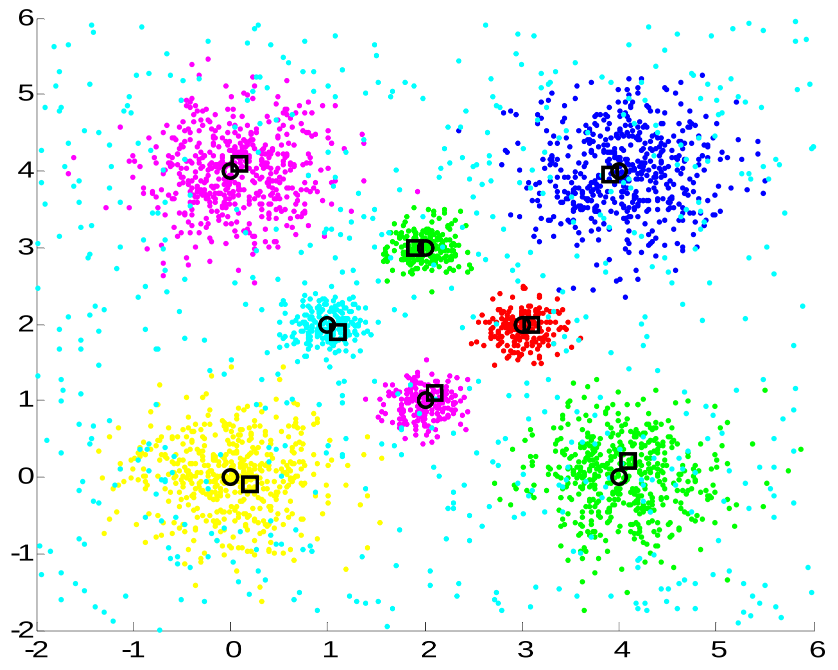


Fig. 4.1 The distribution of the data set T on the dimensions X and Y.

Fig. 4.2 shows the two different methods of setting initial cluster centers. The circles denote the real centers for each cluster. In Fig. 4.2 (a), initial centers are set randomly, denoted by the '+' symbols. The centers randomly selected are usually divergent from the real centers, which makes either no center or more than one center in a cluster. Either case may increase the number of iterations before W-k-means can stop. Fig. 4.2 (b) shows the centers computed by our method $C-Selection$ directly from each class. The centers computed by our method denoted by the rectangle symbols are a bit deviated from real cluster centers due to the noisy samples, but the deviations are insignificant and acceptable.



a）random selection



b）supervised selection

Fig. 4.2 Two methods of selecting the initial cluster centers.

We compare these two initial center selection methods, random selection and supervised selection. Table 4.5 shows the results of 10-fold cross-validation experiments measured in recall [49], precision [48] and F-Measure [50]. We can see that except the precision of class C2, all results produced by the supervised selection were better than the random selection.

Table 4.5 Classification results from random selection and supervised selection of initial clusters centers (R for random selection, P for supervised selection).

		C1	C2	C3	C4	C5	C6	C7	C8
Recall	R	0.907	0.710	0.831	0.719	0.714	0.877	0.818	0.883
	P	0.941	1.00	0.926	1.000	1.000	1.000	1.000	0.982
Precision	R	0.770	0.875	0.817	0.908	0.733	0.841	0.875	0.883
	P	1.000	0.786	1.000	0.941	0.875	1.000	0.985	1.000
F-Measure	R	0.833	0.784	0.824	0.802	0.724	0.859	0.846	0.883
	P	0.970	0.880	0.962	0.970	0.933	1.000	0.979	0.991

2. Identification of noisy attributes and utilization of weight information

W-k-means algorithm can identify noisy attributes by assigning them smaller weights to reduce the influence of noisy attributes. In ADCC model, the weights generated by W-k-means executed in the root node are used to define the weighted distance function when the classifier classifying new objects.

Figure 4.3 plots the synthetic data set T in Table 4.4 in different two-dimensional subspaces. X and Y represent the two attributes that contain eight normally distributed clusters while Z is the noisy attribute on which the samples are uniformly distributed. The clusters cannot be found in the subspaces with Z.

The weights got by executing ADCC five times are shown in Table 4.6. Z gets a smaller weight in every trial since Z has a smaller discrimination capability for clustering.

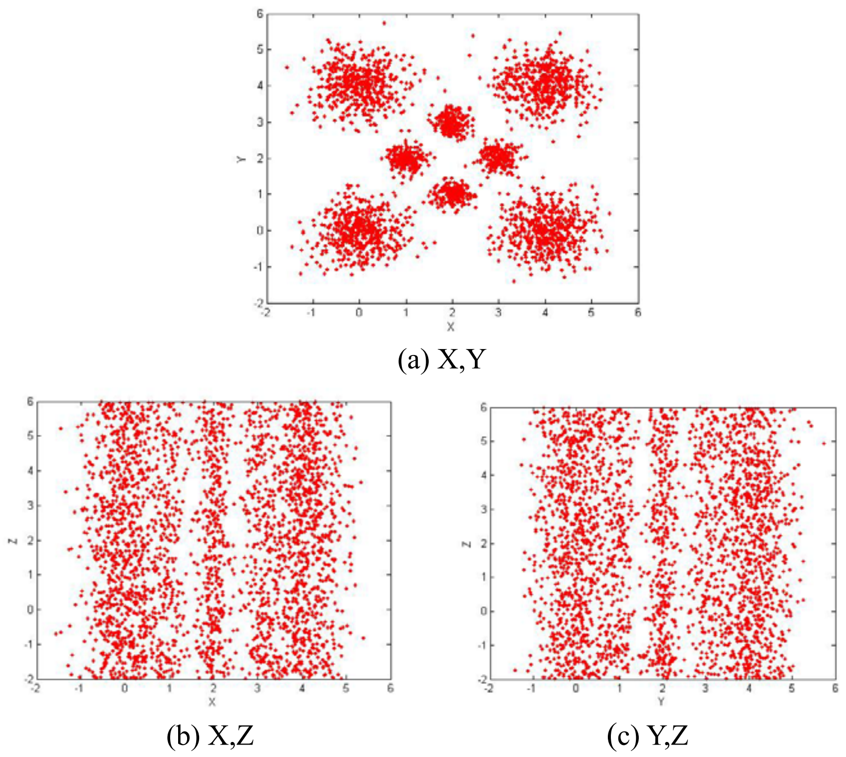


Fig. 4.3 Projection of the data set on different subspaces.

Table 4.6 The weight distribution on the three dimensions.

Times	Weight of X	Weight of Y	Weight of Z
1	0.385	0.399	0.215
2	0.370	0.370	0.257
3	0.344	0.335	0.319
4	0.395	0.392	0.211
5	0.370	0.386	0.242

The following experiments demonstrate that whether the weight information should be used in classification step or not. We adopt the average weights from Table 4.6 as the weights of the three attributes in classification. 10-fold cross-validation is performed in this experiment. The comparison results are demonstrated in Table 4.7. We can see that using weight information (using weighted Euclidean distance function) performs better in most classes. The total average accuracy is 85.46% without weight information, while it can get higher accuracy up to 90.82% by exploiting weight information. In addition to better clustering produced by W-k-means, the classification accuracy is improved by adding the weight information in KNN-like classification step in ADCC. This method reduces the influences of noisy attributes.

Table 4.7 Impact of variable weighting on classification accuracy.

		C1	C2	C3	C4	C5	C6	C7	C8
Recall	Not use	0.938	0.530	0.909	0.650	0.744	0.987	0.762	0.823
	Use	0.968	0.579	0.968	0.690	0.733	0.989	1.000	0.957
Precision	Not use	0.997	0.469	0.998	0.577	0.512	0.993	0.826	0.923
	Use	1.000	0.527	1.000	0.663	0.753	1.000	0.824	1.000
F-Measure	Not use	0.967	0.498	0.952	0.611	0.607	0.990	0.793	0.870
	Use	0.984	0.552	0.984	0.667	0.743	0.955	0.903	0.978

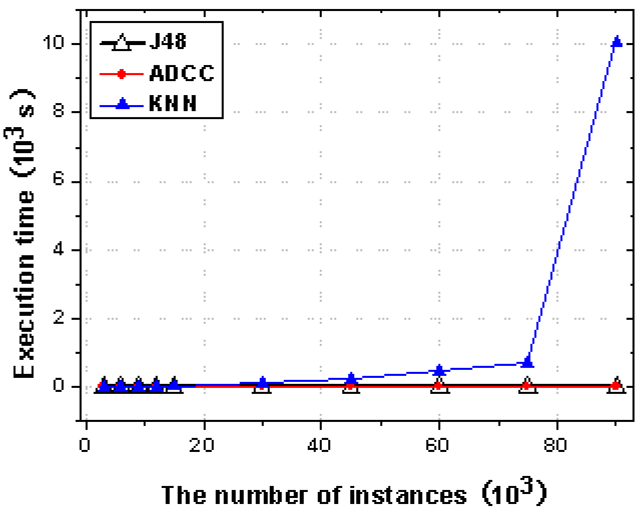
3. Classification Speed and Scalability

We generated two groups of synthetic data sets with different numbers of dimensions and instances (shown in Table 4.8). Each data set contains three clusters randomly generated with normal distributions. In each run, we used 70% of data as training data and the remaining 30% as testing data. We recorded the total execution time and compared the performance of KNN-like classification method in ADCC with original KNN and J48 with different number of dimensions and instances respectively. Table 4.8 shows the details of data sets: data sets A1 to A8 have the number of dimensions varying from 5 to 500, and data sets B1 to B8 have the number of data instances varying from 3,000 to 90,000.

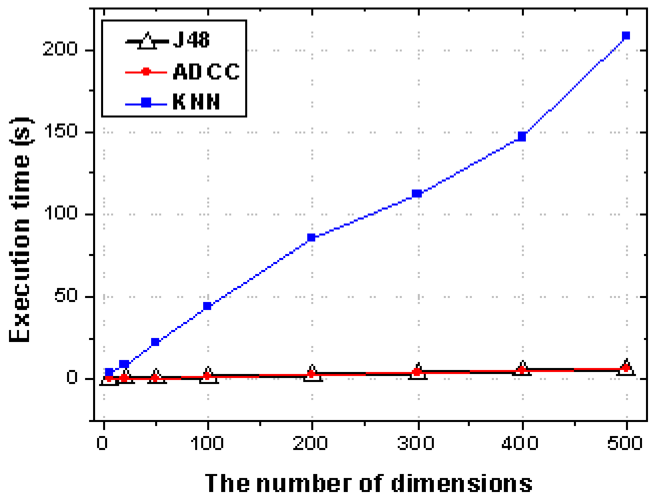
Table 4.8 Two groups of synthetic data sets (each having three classes).

Data sets	Dimensions	Instances	Data sets	Dimensions	Instances
A1	**5**	5,000	B1	4	**3,000**
A2	**20**	5,000	B2	4	**9,000**
A3	**50**	5,000	B3	4	**15,000**
A4	**100**	5,000	B4	4	**30,000**
A5	**200**	5,000	B5	4	**45,000**
A6	**300**	5,000	B6	4	**60,000**
A7	**400**	5,000	B7	4	**75,000**
A8	**500**	5,000	B8	4	**90,000**

The experimental results are shown in Fig. 4.4. Fig. 4.4(a) shows the execution time against the number of instances and Fig. 4.4(b) shows the execution time against the number of dimensions. We can see that the execution time increased linearly for ADCC whereas the execution time for KNN increased rapidly when the number of instanced approached 75000. Although the execution time for KNN increased linearly as the number of dimensions increased, the increase in time was much faster than ADCC. Meanwhile, the execution times for ADCC and decision tree (J48) are comparable to each other and scalable on large data.



(a) Execution time vs. data size



(b) Execution time vs. dimension number

Fig. 4.4 Scalability comparison between ADCC, KNN and decision tree.

4.3.3 Experiments on Spatial Data

We generated five spatial data sets with varying the size from thousands to millions of objects. A series of experiments were conducted on those data sets to show that ADCC can classify spatial data better than other algorithms (e.g. KNN, J48, SMO). Meanwhile the scalability of ADCC was further demonstrated. Table 4.9 gives the information of the five spatial simulation data sets. They have similar data distribution. We randomly choose one of them and show the data distribution in Fig. 4.5.

Table 4.9 Spatial simulation data sets.

Data sets	Classes	Attributes	Size
D1	3	2	13,100
D2	4	2	49,000
D3	5	2	106,087
D4	5	2	524,985
D5	4	2	814,975

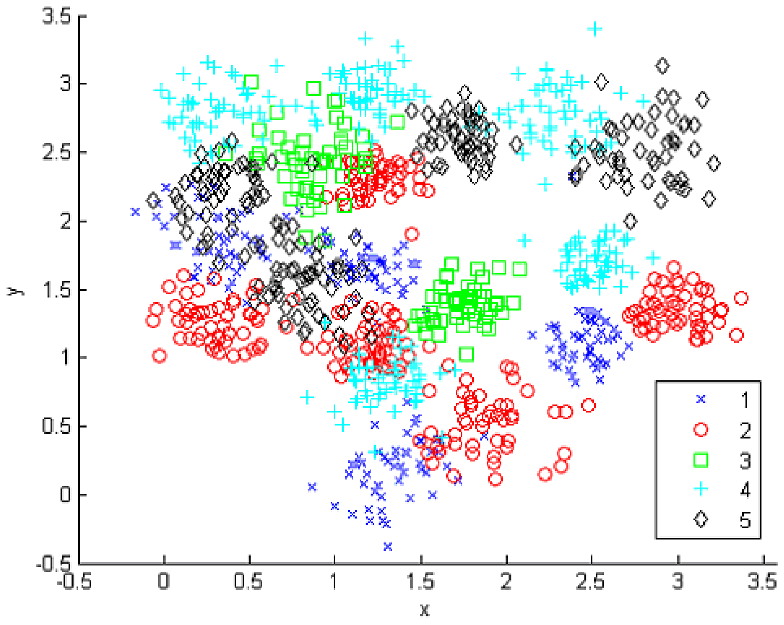
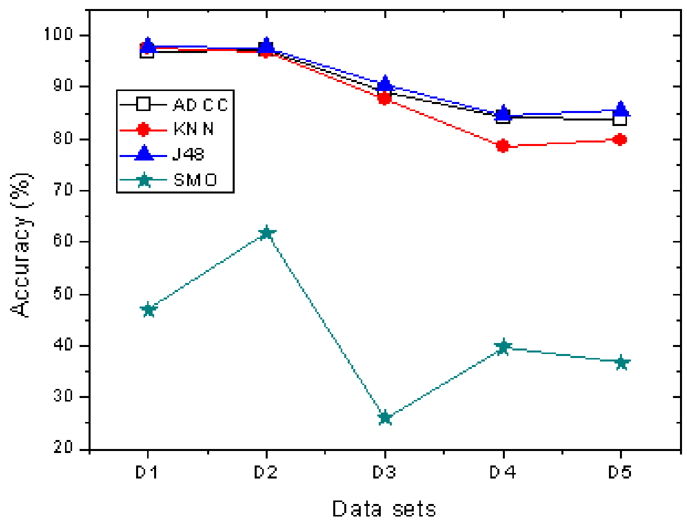
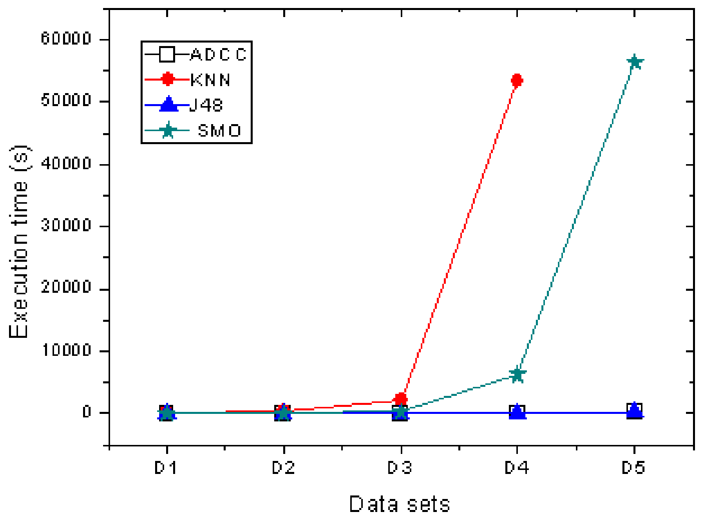


Fig. 4.5 The data distribution of D4.

The comparative results of ADCC and other three classification algorithms are shown in Fig. 4.6. Fig. 4.6(a) shows the classification accuracy on the five data sets and Fig. 4.6(b) demonstrates the classification speed on those five data sets and scalability as the size of data set increased. We can see that ADCC outperforms other three algorithms especially SMO on most cases in terms of accuracy and speed. Meanwhile, the execution time of ADCC and J48 increased linearly whereas the execution time for KNN and SMO increased vastly.



(a) Accuracy



(b) Execution time

Fig. 4.6 Classification results on five data sets.

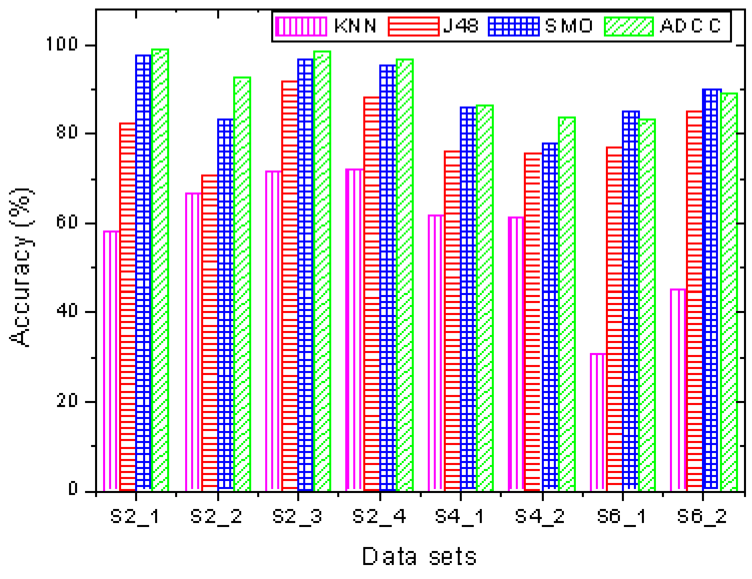
4.3.4 Experiments on Text Data

We show the comparison results of ADCC algorithm and other three classification methods: KNN, decision tree (J48) and SVM (SMO) on the 20-Newsgroups data which is taken from the UCI machine learning data repository [23]. The original text data was first preprocessed to strip the news messages from the special tags and the email headers and eliminate the stem words and stop words. The dimension (word) in each document was weighted by the Term Frequency (TF). Table 4.10 lists eight data sets built from the 20-Newsgroups data. We preprocessed these data sets by deleting some dimensions with smallest TF value. Several thousands words is enough for text data. We also keep all words for Set2_3 and Set2_4 to show our algorithm is efficient on high dimensional data. Different data sets have different cluster properties. Some of them have semantically similar classes, whereas others contain semantically different classes. Some of them have overlapping words (dimensions), while some of them contain the unbalanced number of documents in each class.

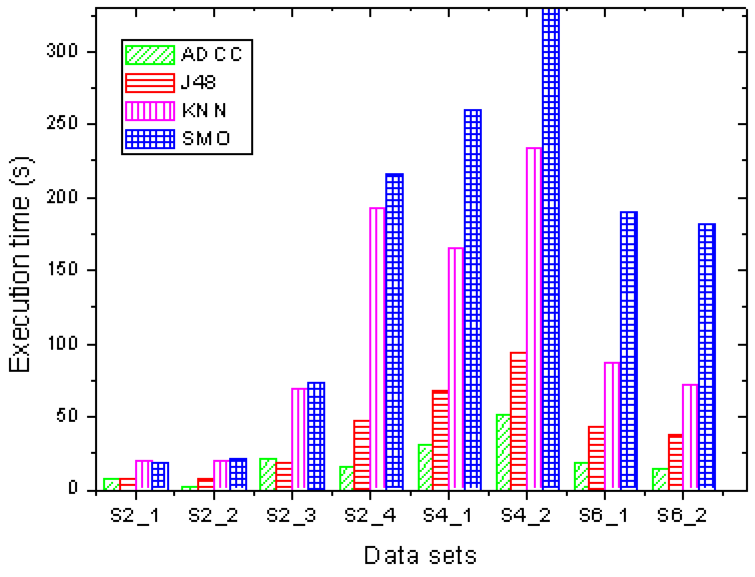
Table 4.10 Text data sets generated from the 20-Newsgroups data (Seti_j denotes the jth data set with i classes).

Data sets	Classes	Dimension	Size
Set2_1	alt.attheism comp.graphics	5201	200 200
Set2_2	comp.sys.imb.pc.hardware comp.sys.mac.hardware	4970	200 200
Set2_3	talk.politics.mideast talk.politics.misc	16411	200 200
Set2_4	rec.autos rec.motocycles	13243	400 400
Set4_1	comp. graphics comp.os.ms.windows.misc rec.autos sci.electronics	7387	400 300 200 100
Set4_2	comp.os.ms.windows.misc comp.sys.imb.pc.hardware comp.sys.mac.hardware comp.windows.x	7302	300 300 300 300
Set6_1	comp.graphics comp.os.ms.windows.misc comp.sys.imb.pc.hardware comp.sys.mac.hardware rec.autos sci.electronics	9425	120 120 120 120 120 120
Set6_2	comp.graphics comp.os.ms.windows.misc comp.sys.imb.pc.hardware comp.sys.mac.hardware comp.windows.x rec.autos	8724	120 120 120 120 100 120

Four classification algorithms, ADCC, KNN, J48 and SMO, were tested on these data sets. Fig. 4.7 shows the classification accuracy and execution time on eight text data sets. We can see that ADCC have the highest accuracy and shortest execution time.



(a) Accuracy



(b) Execution time

Fig. 4.7 Classification results on text data sets.

4.3.5 Experiments on Cancer Data

We also conducted experiments on six gene data sets related to studies of human cancer, which were taken from K-TSP Program Download Page [127]. They have collected 19 publicly available microarray data sets, with sample sizes ranging from 33 to 327 and the number of genes ranging from 2 000 to 16 063. For our decision cluster tree model to be more effective, the training samples should not be too few so that there can be more than a few members in each cluster. Thus, we choose six gene expression data sets since other data sets in K-TSP program are limited in sample size. Table 4.11 lists the characteristics of these six data sets.

Table 4.11 Summary of gene expression data sets.

Data sets	Classes	Genes	Samples	Training samples	Testing samples
GCM	2	16063	280	196	84
Lung	2	12533	181	127	54
Leukemia	2	7129	72	50	22
Prostate1	2	12600	102	72	30
Prostate2	2	12625	88	62	26
Leukemia3	7	12558	327	215	112

We conducted this series of experiments in a similar manner to the experiments on text data sets. Four classification algorithms, ADCC, KNN, J48 and SMO were tested on these data sets. We used 10-fold cross-validation experiment method except Leukemia3. For Leukemia3, we used the training data set and the testing data set originally provided from the original references. Table 4.12 and Table 4.13 show the classification accuracy and execution time on six gene data sets. We can see that, ADCC also have the highest accuracy and the shortest execution time.

Table 4.12 Classification accuracy on gene expression data sets.

Data sets	ADCC	KNN	J48	SMO
GCM	89.29	84.52	75	89.28
Lung	99.5	98	94	98
Leukemia	99	86.3	79.1	98
Prostate1	87.1	90	74	83
Prostate2	80.8	76	55	70
Leukemia3	91.96	75.89	75.89	83.92

Table 4.13 Execution time (seconds) on gene expression data sets.

Data sets	ADCC	KNN	J48	SMO
GCM	26.141	26.187	27.84	29.8
Lung	2.812	8.937	4.2	7.58
Leukemia	1.422	1.953	1.38	1.41
Prostate1	7.078	3.719	2.53	4.76
Prostate2	4.516	9.797	3.59	3.7
Leukemia3	16.031	21.094	9.53	24.38

4.3.6 Experiments on Other Real Data

There are other real-world data sets which are chosen from UCI machine learning data repository [23]. Table 4.14 lists the characteristics of these real data sets.

Table 4.14 Other real data sets.

Data sets	Instances	Dimensions	Classes	Training	Testing
Waveform	5000	40	3	3500	1500
Reuters	9980	337	10	6986	2994

The results of execution time and classification accuracy generated by four classification methods from these two real data sets are listed in Table 4.15. From Table 4.15, we can see that for data set Waveform, the accuracy of ADCC was higher than those of KNN and J48 (decision tree method) and lower than SMO (SVM). ADCC was much faster than KNN but slower than the other two. Comparatively, this data set was simpler with fewer dimensions and instances and a small number of classes. However, for data set Reuters which was more complex with more instances and classes, and much higher dimensions, ADDC outperformed all other algorithms in classification accuracy. It was much faster than KNN and SVM, but only slightly slower than the decision tree implementation. The results further demonstrate that ADDC is more suitable for large, high dimensional data with many classes.

Table 4.15 Classification accuracy on gene expression data sets.

Data sets	Waveform		Reuters	
Metrics	Time(s)	Acc.(%)	Time(s)	Acc.(%)
ADCC	2.516	83.8	74.952	68.97
KNN	15.688	70.8	485.625	57.95
J48	1.2	73.26	35.23	65.29
SMO	1.94	85	392.91	65.86

4.3.7 Parameter Analysis

ADCC needs three parameters, i.e. α, δ and β. α is the threshold in $K-Selection(X,\alpha)$ presented in Section 4.2. δ and β are two parameters of $\mathrm{Terminal-Test}(\mathrm{X})$, which is also presented in Section 4.2. From our experience, α can be set from 0.02 to 0.09. In our previous experiments α was set to 0.05.

In the $\mathrm{Terminal-Test}(\mathrm{X})$ algorithm (see Table 4.3), we first consider the size of the cluster and the purity of the cluster and then consider the result of $\mathrm{AD-Test}(\mathrm{X})$ when we determine whether the current cluster X should be further divided. Since it takes longer for $\mathrm{AD-Test}(\mathrm{X})$ than judging the size and the purity of X, so we want to filter the obvious situations under which the cluster needs not to be further divided. Thus, the size threshold δ and the purity threshold β can be set to the values so that the further clustering of X should be terminated obviously when X does not pass either of these thresholds. δ depends on the size of the smallest class which includes the fewest samples compared with other classes before clustering the whole training samples. δ can be set from 3% to 10% of the number of samples in the smallest class, and β can be changed from 85% to 95%. In our previous experiments we set β to 90%, and δ to 10% of the number of samples in the smallest class.

We use three data sets, i.e. D3, Set4_1 and Prostate1 which were chosen from Table

4.9, Table 4.10, Table 4.11 respectively to demonstrate how these parameters influence the classification result.

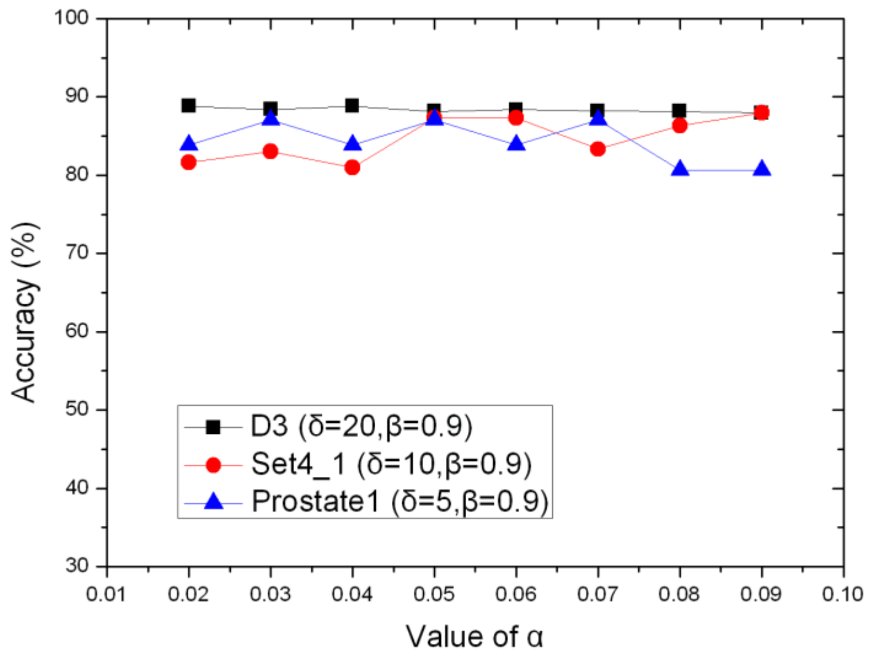


Fig. 4.8 The effect of α on classification accuracy.

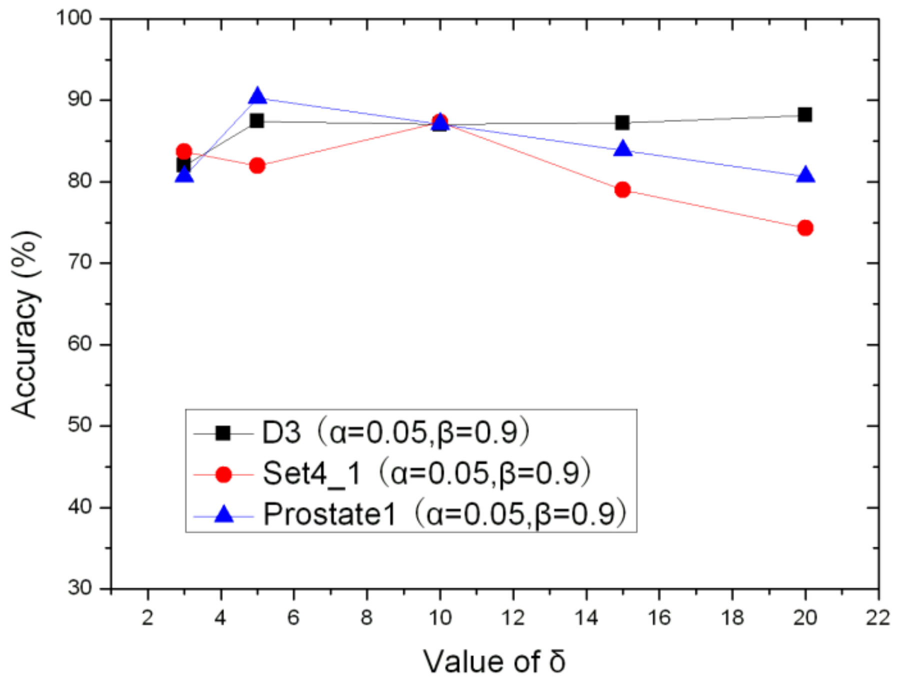


Fig. 4.9 The effect of δ on classification accuracy.

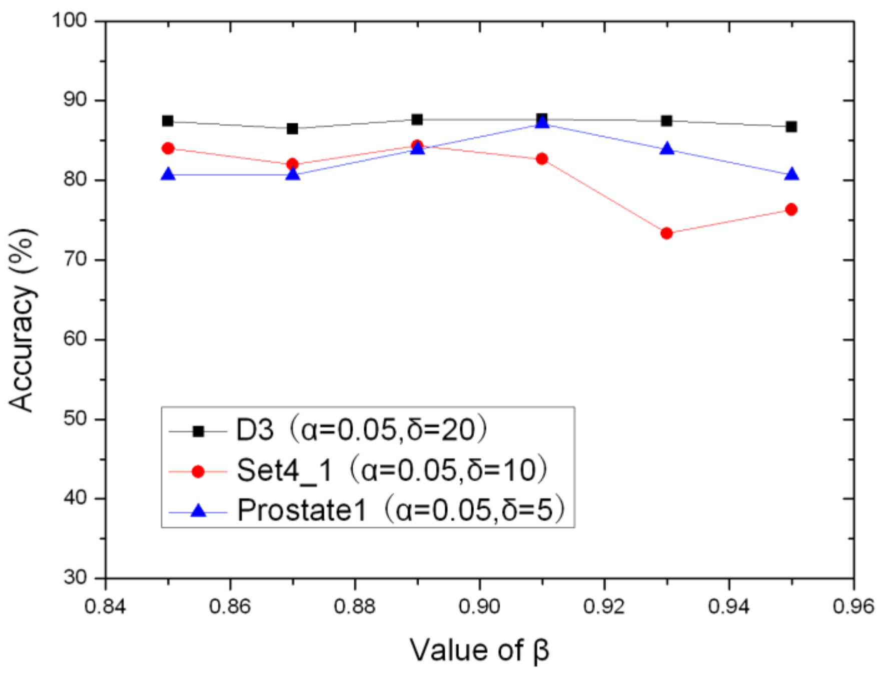


Fig. 4.10 The effect of β on classification accuracy.

Fig. 4.8 shows the classification accuracy against different values of α of ADCC on the three data sets. We can see that the classification accuracy was not sensitive to α when α changed from 0.02 to 0.09. Fig. 4.9 shows the classification accuracy against different values of δ of ADCC on the three data sets. The classification result of ADCC method was robust on the parameter δ when δ changed from 3% to 10% of the number of samples in the smallest class. Fig. 4.10 shows the classification accuracy against different values of β of ADCC on the three data sets. We can see that the classification accuracy was not sensitive to β when β changed from 0.85 to 0.95. These results demonstrated that the classification result of ADCC method was robust on the parameter α, δ and β. In the experiments of the following chapters, if we do not demonstrate especially, the settings of these three parameters are the same as the value in this chapter.

4.4 Conclusion

In this Chapter, we have proposed a new classification method following the

Cluster-based classification framework (Decision Cluster Tree framework). We have presented an automatic algorithm ADCC which uses the variable weighting k-means algorithm W-k-means to build a decision cluster tree from a training data set. In this automatic approach, we have proposed solutions to solve three important problems: (1) selection of the number of sub-clusters at each node, (2) selection of the initial cluster centers, and (3) termination of further clustering at a node.

We have presented experimental results on both synthetic and real world data sets and compared the performance of ADCC with those of other well-known classification methods. The comparison results have shown that ADCC has advantages in classifying large, high dimensional data with multiple classes.

Chapter 5

Building A Decision Cluster Forest Model to Classify High Dimensional Data with Multi-classes

In this chapter, a decision cluster forest classification model is proposed for high dimensional data with multiple classes. A decision cluster forest (DCF) consists of a set of decision cluster trees, in which the leaves of each tree are clusters labeled with the same class that determines the class of new objects falling in the clusters. By recursively calling a variable weighting k-means algorithm, a decision cluster tree can be generated from a subset of the training data that contains the objects in the same class. The set of m decision cluster trees grown from the subsets of m classes constitute the decision cluster forest. Anderson-Darling test is used to determine the stopping condition of tree growing. A DCF Classification (DCFC) model is selected from all leaves of the m decision cluster trees in the forest. A series of experiments on both synthetic and real data sets have shown that the DCFC model performed better in accuracy and scalability than the single decision cluster tree method and the methods of KNN, decision tree and SVM. This new model is particularly suitable for large, high dimensional data with many classes.

5.1 Introduction

One challenge in data mining is classification of high dimensional data with multiple classes [21]. This kind of data may occur in application fields such as text mining, multimedia mining and bio-informatics. To solve this problem, the ADCC method was proposed in [109], which builds a classification model from high dimensional data. Given a training data set, the ADCC algorithm recursively calls the variable weighting k-means algorithm (W-k-means) [2] to generate a decision cluster tree. Each node with its dominant class forms a decision cluster. ADCC uses the leaves of the tree as the classification model, and leaves are labeled with their dominant classes to determine the classes of new objects falling in the clusters. Experimental results have shown that this decision cluster classification model was effective and efficient in classifying high dimensional data [109].

One shortcoming of this ADCC method is that the algorithm generates some weak decision clusters in which no single class dominates. Existence of weak clusters in the model can affect classification performance of the model. It has been shown that classification accuracy could be improved after weak decision clusters were removed from the model [120]. Weak decision clusters occur because objects of different classes are mixed in the clustering process to generate decision clusters. If we assume that objects in the same class have their own cluster distributions, we can separate objects of different classes according to the object class labels and generate decision clusters from objects in each class. Then, we combine the decision clusters of different classes to form the decision cluster classification model. In this way, weak decision clusters can be avoided.

In this chapter, we propose a Decision Cluster Forest method to build a decision cluster classification model from high dimensional data with multiple classes. Instead of building a single decision cluster tree from the entire training data, we build a set of decision cluster trees from subsets of the training data set to form a decision cluster forest. Each tree in the forest is built from the subset of objects in the same class. The proposition for this method is that the objects in the same class tend to have their own spatial distributions in the data space. Therefore, decision clusters of objects in the same class are found. The decision clusters in the same tree have the same dominant class. In this way, no weak decision cluster is created in such decision cluster tree. A decision cluster model can be selected from any subset of leaf decision clusters from the decision cluster forest so the model is called a decision cluster forest classification model (DCFC).

The decision cluster forest method has advantages of classifying data with multiple classes because the DCFC model is guaranteed to contain decision clusters in all classes. In other multi-class classification methods, such as decision trees, the information of small classes is often under represented in the model. The error-correcting output codes method (ECOC) was designed to solve multi-class learning problem by learning multiple binary classification models and matching the classification results with the designed codeword to correct misclassifications [121]. This new method is more like an ensemble method but the challenge is on the design of code word. In contrast, the DCFC model is a more intuitive and direct multi-class classification method and easy to use.

In growing a decision cluster tree from a subset of objects in the same class, we adopt the W-k-means algorithm to reduce the effect of noisy attributes in high dimensional data. We also grow a binary decision cluster tree and use Anderson Darling Test [6,7] as a stopping criterion in tree growing. We have conducted a series of experiments on both synthetic and real data sets to demonstrate the efficiency and accuracy of the DCFC method. Compared with other classification methods, including ADCC, KNN, J48 (a decision tree algorithm) and SMO (one of SVM algorithms), our experimental results have shown that the DCFC method has performed better than those methods in classification accuracy on large high dimensional sparse data sets. Thus the results demonstrate that the DCFC method is more suitable for large, high dimensional data with many classes.

Clustering methods have been explored to solve classification problems [22,35]. An early example of using the k-means clustering algorithm to build a cluster tree classification model can go back to early 80's [19]. In this work, a binary cluster tree was built by interactively executing the k-means clustering algorithm. At each node, a further partition was determined by the percentage of the dominant class in the cluster node. However, only small numeric data could be classified and every time only two sub-clusters are formed. In 2000, Huang et al. proposed a new interactive approach to build a decision cluster classification model [1]. In this approach, the k-prototypes clustering algorithm was used to partition the training data, and a visual cluster validation method [60] was adopted to verify the partitioning result at each cluster node. The above two interactive methods are not adequate for high dimensional data with noise because the clustering algorithms used are not able to handle noisy attributes and it is too time consuming to involve human judgment. The concept clustering tree which is a decision tree where each node as well as each leaf corresponds to a cluster is proposed by Blockeel et al. [122]. The nodes of decision cluster tree proposed in our method are also clusters, but the process of the tree construction including partition method, stopping criteria and the number of sub-clusters are totally different.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the decision cluster forest classification model and the algorithm for model building. In Section 5.3, experimental results and comparisons are reported. In Section 5.4, we conclude this chapter.

5.2 Decision Cluster Forest

In this section, we describe how to construct the decision cluster forest and how to do classification with the decision cluster forest.

5.2.1 Decision Cluster Forest (DCF)

A Decision Cluster Forest (DCF) consists of a set of decision cluster trees. Each tree grows from a subset of the training data that contains objects in the same class. If the training data X has m classes, the decision cluster forest will have m decision cluster trees. Given a decision cluster forest, a Decision Cluster Forest Classification (DCFC) model can be built by simply extracting the leaves of decision cluster trees.

All decision clusters in a DCFC model from the decision cluster forest have strong dominant classes with 100 percentage distribution. This approach follows the proposition that in a large high dimensional data set with multiple classes, the objects in each class tend to occupy a spatial region with its own mixture density distribution. Therefore, the mixture densities can be discovered in a clustering process from the set of objects in the same class.

Let X be a training data set of n objects in m classes. We divide X into m subsets $(X_1, X_2, \ldots X_m)$, each with objects in the same class. For each subset, we use a clustering algorithm to build a decision cluster tree in which all nodes have the same dominant class. The decision cluster forest consists of m decision cluster trees with nodes of m different dominant classes. Fig. 5.1 illustrates a decision cluster forest with different decision cluster trees.

Given a decision cluster forest, we can select any subset of decision clusters from multiple trees to build a DCFC model. Similar to the single decision cluster tree method, the performance of a DCFC model on classification accuracy also depends on the quality of the decision cluster trees and the selection of decision clusters to be included in the model. Therefore, the following two processes are crucial: (1) generation of a set of decision cluster trees and (2) selection of a subset of the decision clusters from the decision cluster forest for the classification model.

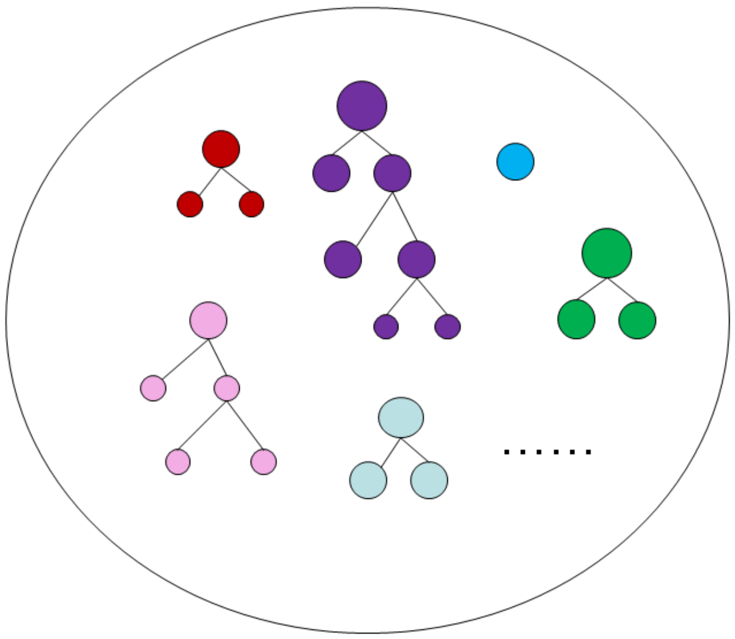


Fig. 5.1 Distribution of decision cluster trees in a decision cluster forest.

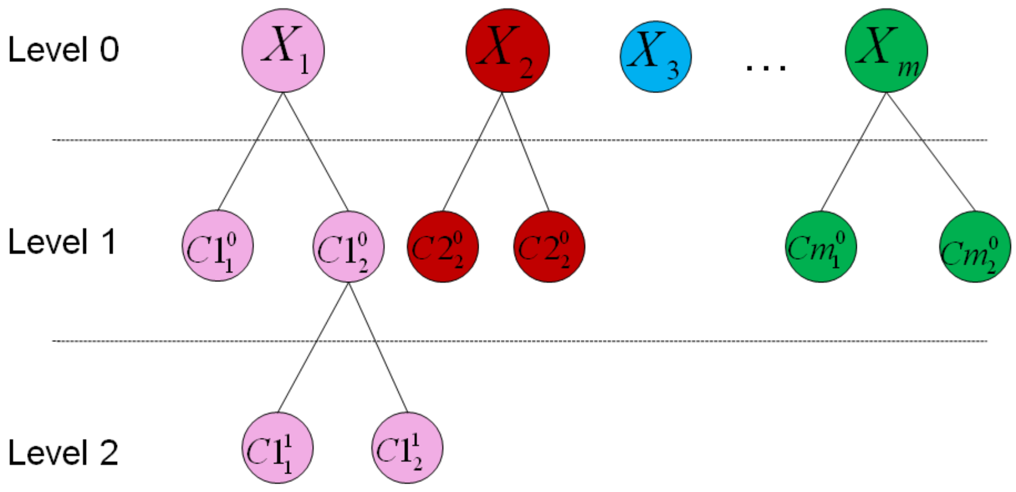


Fig. 5.2 Generation of decision cluster trees for a decision cluster forest.

Given a subset X_i from the training data set X, a decision cluster tree can be built with a process similar to that used in building a single decision cluster tree. However, since all objects in the data set are in the same class, class distribution calculation is not necessary. When executing W-k-means on each inner node, the parameter k (the number of clusters should be divided) of W-k-means is unknown. To avoid decreasing the clustering quality due to a too large k value, we set parameter k to 2 and generate a hierarchical binary cluster tree. That is, we perform a binary partition to result in a binary tree at each inner node. All decision clusters are assigned the same dominant class. A decision cluster tree with the same dominant class is shown in Fig. 5.2. Multiple decision cluster trees are generated from different subsets of training data with different dominant classes marked in different colors.

Take the left tree from Fig. 5.2 as an example. This binary decision cluster tree is generated from the data subset X_1 shown in the root. The root X_1 is first partitioned into 2 clusters $C1_1^0$, $C1_2^0$. Here, the cluster index $C1$ indicates the clusters are generated from the data of class 1. Therefore, all clusters have the same dominant class 1. The superscript indicates the level of the node from which the clusters are generated and the subscript is the cluster number in this level. Cluster $C1_2^0$ is further partitioned into 2 sub-clusters $C1_1^1$, $C1_2^1$, which form level 2 of this decision cluster tree. This binary partition process continues until the stopping criteria are satisfied at the two leaves. Other decision clusters trees marked in different colors are generated from different subsets of the training data in a similar way.

In a real world training data set, the distribution of classes is often unbalanced. Some classes have more objects than others. Therefore, the depths of decision cluster trees are different. In the extreme case such as X_3, a decision cluster tree may have only the root because the number of objects is too small. In this case, we treat X_3 as one decision cluster in the model.

5.2.2 DCF Classification (DCFC) Model

The decision cluster forest in Fig. 5.2 can be represented in a sequence of nested clusterings as follows:

$$\{X1(C1_1^0, C1_2^0(C1_1^1, C1_2^1)); X2(C2_1^0, C2_2^0);...; Xm(Cm_1^0(Cm_1^1, Cm_2^1), Cm_2^0)\}, \quad (5\text{-}1)$$

In this sequence, the decision clusters with the same dominant class are grouped in the top level of X_i for $1 \le i \le m$. Any subset of disjoint decision clusters from the top levels downward can make a DCFC model. There are many ways to select classification models from a decision cluster forest. In this work, we select the leaf nodes of each decision cluster tree in the decision cluster forest to build the DCFC model. Each decision cluster in the model is represented by its center and dominant class.

The DCFC model classifies new objects in a KNN-like way as follows:

1. Compute the distances between a new object and the centers of the decision clusters with a distance function;

2. Assign to the object the dominant class of the decision cluster with the shortest distance to the object.

5.2.3 DCFC Algorithm

Table 5.1 shows the DCFC algorithm to automatically build a decision cluster forest and select a leaf-based DCFC model. The input to the algorithm is a training data set with m classes and the output from the algorithm is a DCFC model. The algorithm first divides the input data X into m subsets. For each subset, it calls the W-k-means algorithm to generate a decision cluster tree. At each node, it calls $\text{Stop}-\text{Test}(\ )$ function to test the stopping criteria to determine whether to call W-k-means to partition the node into two children nodes or turn the node to a leaf node. After all decision cluster trees are generated, a decision cluster forest is obtained. Finally, the set of all leaf decision clusters is returned as the DCFC model.

Table 5.1 DCFC Algorithm.

```
Input: A training data set $X$ with $m$ classes.
Output: A classification model $\mathrm{DCFC-model}$.
Begin
1.  partition $X$ into m training data subsets, $X = \{X_1, X_2, ..., X_m\}$.
2.  for each training data subset $X_i$, $i = 1,...,m$;
3.      initialize a decision cluster tree $DCT_i$ with root $\{X_i\}$;
4.      sign the root as internal node;
5.      for each internal node $C$ in $DCT_i$
6.          if ($\mathrm{Stop-Test}(\mathrm{C})$) sign $C$ as leaf node;
7.          run W-k-means on $C$ to produce two new sub-clusters;
8.          sign the two new sub-clusters as internal node and add them into $DCT_i$;
9.      end for
10. end for
11. include all leaf nodes from all $m$ trees into the classification model
    $\mathrm{DCFC-model}$ and represent each node by its center and its dominant class;
12. return $\mathrm{DCFC-model}$;
End
```

The $\mathrm{Stop-Test}(\mathrm{C})$ algorithm is given in Table 5.2. Since all data objects belong to the same class, the class label information is no longer useful in the tree generation process. In this algorithm, only two stopping criteria are considered: the number of objects in the data subset and the Anderson Darling (AD) Test [6,7]. If a node cluster is too small, it need not be further partitioned. If objects in a cluster follow a normal distribution, it is a good cluster and does not need a further partition. Anderson Darling (AD) Test is a powerful statistical method to determine whether a sample comes from a specified distribution or not. If the testing result satisfies the AD Test criteria, this cluster node is treated as a leaf node. Otherwise, it is further partitioned into sub-clusters.

Table 5.2 Algorithm of Stop-Test(C).

```
Input: the node $C$ which contains n objects.
Output: A Boolean value stop which is either (i) TRUE which means "stop" or
        (ii) FALSE, otherwise.
Remarks:
δ: the threshold of the number of samples in $C$ (e.g., 10).
Begin
1.  stop = FALSE;
2.  if (n <δ OR AD-Test($C$)
3.      stop = TRUE and label C with the dominant class label;
4.   return stop;
End
```

5.3 Experiments

In this section, we present the 10-fold cross-validation experiments we have conducted on both synthetic and real-world data sets. The experimental results demonstrate that

DCFC outperforms some existing classification algorithms, including original KNN, decision tree (J48), SVM (SMO) and ADCC [109] in terms of speed, scalability and classification accuracy. Weka [126] implementations of J48 and SMO were used in our comparisons. DCFC algorithm and ADCC were implemented in java.

All experiments were conducted on an Intel(R) Xeon(R), 1.60 GHz computer with 8GB memory. We compared the accuracy and execution time of these classification algorithms. In our experiments, our setting is similar to [109] where default parameters were used for J48 and SMO in Weka. For J48, the minimal number of instances per leaf is set as 2. The SMOs were trained with a linear kernel where the complexity parameter C was set as 1.0. For KNN, the number of neighbors, k, equals to 1. For ADCC, we use the same parameter settings as in Chapter 4.

DCFC needs one parameter δ presented in $\mathrm{Stop-Test}(\mathrm{C})$, which is presented in Section 5.2.3. In $\mathrm{Stop-Test}(\mathrm{C})$ algorithm (see Table 5.2), we first consider the size of the current node and then consider the result of AD-TEST(C) when we determine whether the current node C should be further divided. Since it takes longer for AD-TEST(C) than judging the size, we want to filter the obvious condition under which the node need not be further divided. Thus, the size threshold δ can be set to values so that the further clustering of C should be stopped obviously when the data size in node C is less than δ. δ depends on the size of the root node and is always set to 5% of the number of samples in the root node. If the size of a node is smaller than 5% of the size of the root, we consider this node as a leaf automatically.

5.3.1 Experiments on Text Data

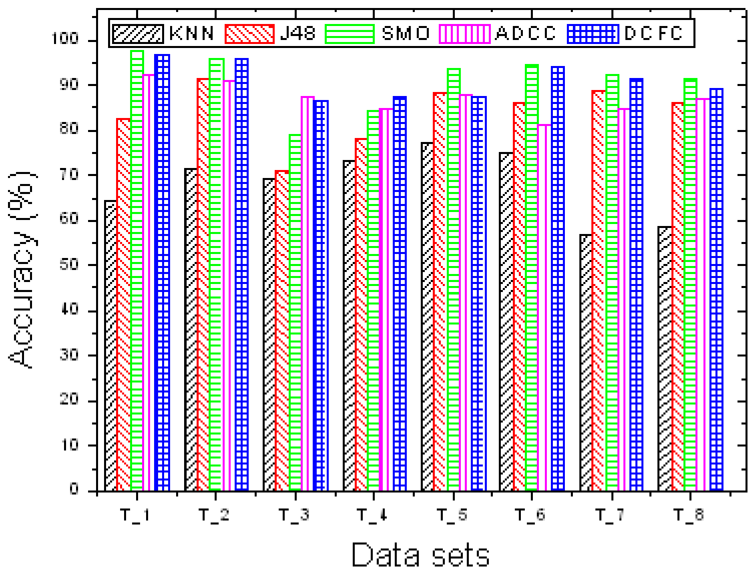
We compared DCFC with other three well-known classification algorithms and ADCC on some high dimensional real data sets. These data sets are taken from the UCI machine learning data repository [23].

Table 5.3 lists eight data sets built from the Twenty Newsgroups data following the similar method in Chapter 4. The original text data was first preprocessed to strip the news messages from the special tags and the email headers and eliminate the stem words and stop words. The dimension (word) in each document was weighted by the Term Frequency (TF). We preprocessed these data sets by deleting some dimensions with smallest TF values. The resulting text data contain several thousand frequent words. Different data sets have different cluster properties. Some of them have semantically similar classes (e.g. T_2), whereas others contain semantically different classes (e.g. T_1). Some of them have overlapping words (dimensions) (e.g. T_5), while some of them contain the unbalanced number of documents in each class (e.g. T_8). This group of text data sets is different from the group in Chapter 4. We generate the different data sets every time to show that our framework is effective on so many different data sets.

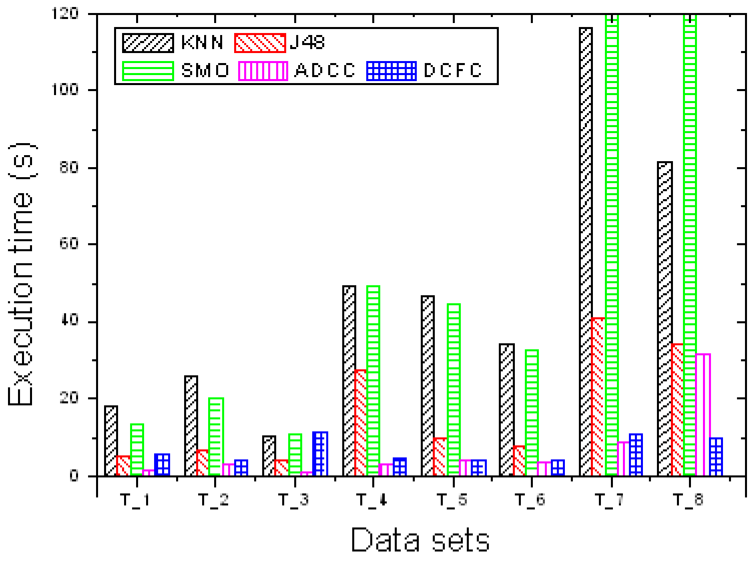
Table 5.3 Text data sets generated from the 20-Newsgroups data .

Data sets	Classes	Dimension	Size
T_1	alt.attheism comp.graphics	3939	200 200
T_2	talk.politics.mideast talk.politics.misc	5795	200 200
T_3	comp.sys.imb.pc.hardware comp.sys.mac.hardware	2558	200 200
T_4	Alt.attheism Talk.religion.misc	5856	300 300
T_5	rec.autos rec.motocycles	3154	400 400
T_6	rec.autos rec.motocycles	3979	200 400
T_7	comp. graphics Rec.sport.baseball Sci.space Talk.politics.mideast	7924	200 200 200 200
T_8	comp. graphics Rec.sport.baseball Sci.space Talk.politics.mideast	8549	300 200 100 50

Five classification algorithms, DCFC, ADCC, KNN, J48 and SMO, were tested on these text data sets. Each classification algorithm was run 10 times on each data set using different 30% of data as testing test (70% as training data). The deviation of results of each algorithm is not large, so we report their average. Fig. 5.3 shows the classification accuracy and execution time on these text data sets. We can see that DCFC achieves higher accuracy than ADCC, KNN and J48 in most cases. DCFC and ADCC are comparably faster than other algorithms. DCFC achieves higher or close accuracy to SMO but much faster than that.



(a) Accuracy



(b) Execution time

Fig. 5.3 Classification results on text data sets.

5.3.2 Experiments on Other Real Data

Table 5.4 lists the other two real data sets which are also taken from UCI machine learning repository [23].

Table 5.4 Other real data sets.

Data sets	Instances	Dimensions	Classes	Training	Testing
Waveform	5000	40	3	3500	1500
reuters	9980	337	10	6986	2994

Table 5.5 lists the results of execution time and classification accuracy generated by five classification methods from these two real data sets. These two data sets are comparably simple and low dimensional. DCFC works as well as ADCC which is more suitable for large, high dimensional data.

Table 5.5 Classification results of data sets in Table 5.4 by five classification methods.

Data sets	Waveform		Reuters	
Metrics	Time(s)	Acc.(%)	Time(s)	Acc.(%)
DCFC	2.5031	84.4	81.922	68
ADCC	2.516	83.8	74.952	68.97
KNN	15.688	70.8	485.625	57.95
J48	1.2	73.26	35.23	65.29
SMO	1.94	85	392.91	65.86

5.3.3 Scalability

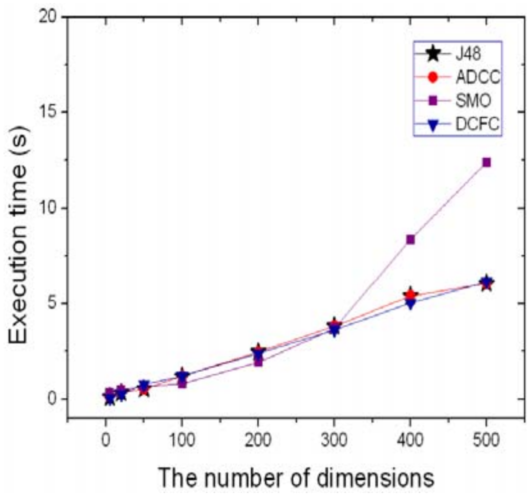
We conducted experiments on synthetic data sets to demonstrate and compare the scalability between our new DCFC algorithm and other classification algorithms. The results show that our method is more scalable when the number of samples and dimensions increase.

We adopted the same two groups of synthetic data sets with different numbers of dimensions and samples (shown in Table 5.6) as the same as in Chapter 4. The two groups of synthetic data sets are listed here again to make the thesis easy to follow. Each data set contains three clusters randomly generated with normal distributions. The class labels are randomly arranged. We recorded the total execution time and compared the performance of DCFC with original KNN, J48 and ADCC with different number of dimensions and samples respectively.

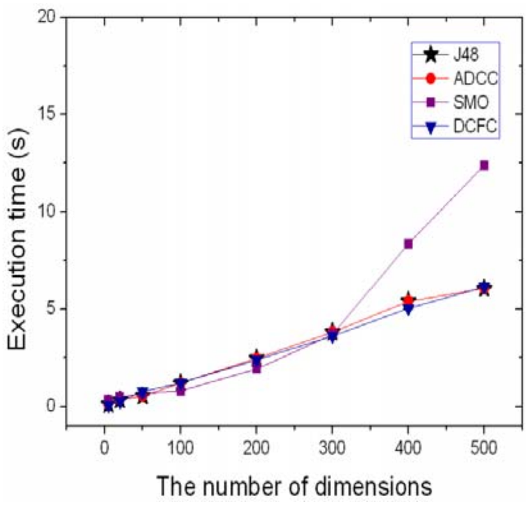
Table 5.6 Two groups of synthetic data sets (each having three classes).

Data sets	Dimensions	Instances	Data sets	Dimensions	Instances
A1	5	5,000	B1	4	3,000
A2	20	5,000	B2	4	9,000
A3	50	5,000	B3	4	15,000
A4	100	5,000	B4	4	30,000
A5	200	5,000	B5	4	45,000
A6	300	5,000	B6	4	60,000
A7	400	5,000	B7	4	75,000
A8	500	5,000	B8	4	90,000

Figure 5.4 shows the execution time against the number of dimensions and Fig. 5.5 shows the execution time against the number of samples. Because the curve of the result of KNN ranges too large to demonstrate the curves of other algorithms clearly, we presented the results in two sub-figures. Sub-figure (a) is the comparison results of DCFC, ADCC, SMO and J48, and Sub-figure (b) is the comparison results adding KNN. We can see that the execution time increased linearly for DCFC, ADCC, SMO and J48 whereas the execution time for KNN increased rapidly when the number of instances approached 90000. Although the execution time for KNN increased linearly as the number of dimensions increased, the increase was much faster than DCFC, SMO, ADCC and J48. Meanwhile, the execution times for DCFC, ADCC, SMO and decision tree (J48) are comparable to each other and scalable on this group of data sets. SMO grows faster over dimensions than DCFC, ADCC and J48.

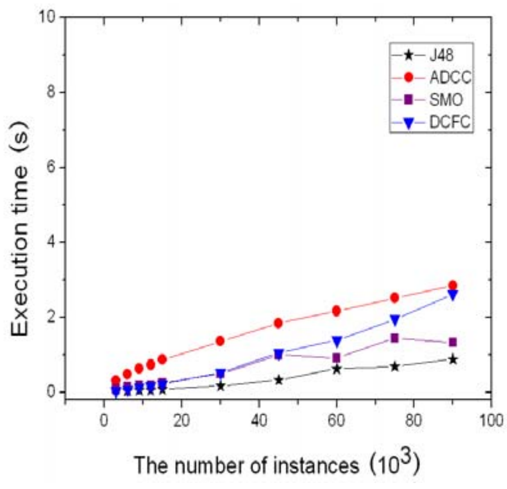


(a) Except KNN

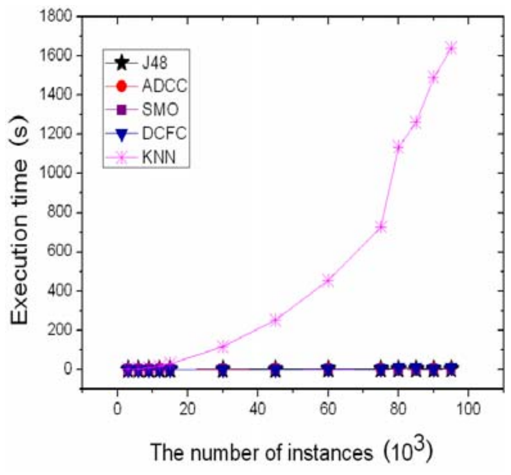


(b) With KNN

Fig. 5.4 Execution time vs. dimension number.



(a) Except KNN



(b) With KNN

Fig. 5.5 Execution time vs. data size.

5.4 Conclusion

In this chapter, we have proposed a new classification method for using a clustering method to build a decision cluster forest. Decision cluster classification models are generated from the decision cluster forest. We have presented an automatic algorithm DCFC which uses the variable weighting k-means algorithm W-k-means to build a decision cluster forest from a training data set. The classifier is constructed by selecting all leaf nodes from the decision cluster forest.

We have presented experimental results on both synthetic and real world data sets and compared the performance of DCFC with those of other well-known classification methods. The comparison results have shown that DCFC has advantages in classifying large, high dimensional data with multiple classes.

Chapter 6

An Ensemble of Decision Cluster Crotches for Classification of High Dimensional Data

In this chapter, we present a Crotch Ensemble classification model for high dimensional data with multiple classes. A Crotch Ensemble is obtained from a decision cluster tree built by calling a clustering algorithm recursively. A Crotch is an inner node of the tree together with its direct children. If the children of a crotch have more than one dominant class, the crotch is defined as a Crotch Predictor that is a classifier by itself. A crotch ensemble consists of a set of crotch predictors. When classifying a new object, a subset of crotch predictors is selected according to the distances between the object and the crotches. A classification is made on the object as the class predicted by the crotch predictors with the maximum accumulative weights. The experimental results on both synthetic and real data have shown that the Crotch Ensemble model can get better classification results on high dimensional data than other classification methods.

6.1 Introduction

We present an ensemble of crotches of decision clusters for classification of high dimensional data with multiple classes in this chapter. A decision cluster tree is built by calling a clustering algorithm recursively. Each node with its dominant class forms a decision cluster [109]. A crotch is an inner decision cluster with its children in the decision cluster tree. This chapter presents a new classification method which uses an ensemble of classification models produced from crotches. An ensemble of classification models gains better performance and wider working space than a single classification model.

A crotch consists of an inner node and its children nodes in a decision cluster tree. When traversing the tree, a crotch whose children have more than one dominant class is regarded as a crotch predictor which itself forms a classification model. These classification models (crotch predictors) construct the Crotch Ensemble. Our task is to generate a more effective classifier based on this crotch ensemble.

Crotch Ensemble is an extension to the decision cluster classification model ADCC proposed in [109]. The ADCC model consists of a set of decision clusters that are taken from all leaves of a decision cluster tree built from a training data set. In this leaf decision cluster model, the domain space can be divided with the Voronoi partition [119] into a set of non-overlapping decision regions by the set of decision cluster centers. The decision about the class of a new object is made by the decision cluster in whose decision region the object falls. Classification errors often occur at the boundary of two decision cluster regions with different dominant classes and in the decision regions of weak decision clusters that contain objects of multiple classes without clearly dominant class. The crotch ensemble classifies a new object with multiple crotch predictors which are built from decision clusters at different levels of the decision cluster tree and the decision clusters in different crotch predictors can overlap. If a new object is misclassified by one crotch predictor, the misclassification can be corrected by other crotch predictors. Therefore, multiple predictor decisions are more robust than the single decision cluster decision. A series of experiments on both synthetic and real data sets have demonstrated that the efficiency and the accuracy of the Crotch Ensemble method is superior to the ADCC model. Compared with other classification methods, including KNN [34], J48 (a decision tree algorithm) [27], Random Forest [118] and ADCC [109], our experimental results also showed that the Crotch Ensemble has performed better than other methods on large, high dimensional data with many classes.

The decision tree has been proved as a useful and powerful tool in data mining and machine learning [27,110,111,112]. However, with the quick development of data acquisition, data transmission and storage technology, traditional decision tree methods face a lot of challenges in classification of high dimensional data with multiple classes [21,113]. A decision tree is often built from a small number of dimensions from high dimension such as text data. This is because each partitioning step in building a decision tree model only considers one dimension while the information is usually stored among many dimensions. When there are a large number of classes, a large number of leaves are generated, which will cause an over-fitting problem [30,114,115,116,117]. To avoid these disadvantages, ADCC algorithm [109] was proposed to take a subset of dimensions at each node to build a decision cluster tree from high dimensional data with a clustering algorithm.

The rest of this chapter is organized as follows. Section 6.2 proposes the Crotch Ensemble algorithm. Discussions and analysis about the crotch ensemble method are given in Section 6.3. Experiments on synthetic and real data sets are presented in Section 6.4. Section 6.5 concludes this work.

6.2 Crotch Ensemble Algorithm

This section proposes Crotch Ensemble algorithm, which uses crotch predictors of a decision cluster tree to construct a classification model. It includes three steps, selecting crotch predictors, constructing Crotch Ensemble classifier and training crotch weight.

Section 6.2.1 presents how to get crotches from a decision cluster tree and select useful crotch predictors among them. Crotch Ensemble classifier is proposed in section 6.2.2. Section 6.2.3 discusses predictor bounding to avoid the influence of outer crotch predictors which are too far away from the new samples. In section 6.2.4, the scheme of training crotch weight to enhance the crotch ensemble classifier is presented.

Fig. 6.1 shows an example of a decision cluster tree of four levels. Level 0 is the root T which is the training data set. The root T is partitioned into three clusters C_1^0, C_2^0, C_3^0. Here, the superscript indicates the level of the node from which the clusters are generated and the subscript is the cluster number in this level. Clusters C_1^0 and C_3^0 are further partitioned into 2 and 3 sub-clusters respectively, which form level 2 of the cluster tree. Subsequently, two clusters C_2^1 and C_4^1 are further partitioned into three and two sub-clusters respectively, which form level 3 of the cluster tree. This tree can be represented as the following sequence of nested clusterings as $T(C_1^0(C_1^1, C_2^1(C_1^2, C_2^2, C_3^2)), C_2^0, C_3^0(C_3^1, C_4^1(C_4^2, C_5^2), C_5^1))$.

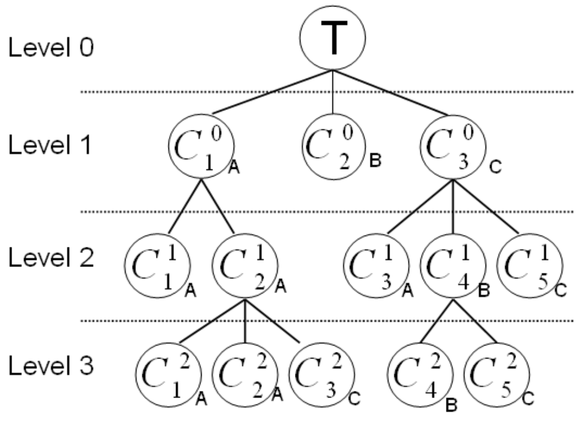


Fig. 6.1 Example of a decision cluster tree (the letter A, B and C beside the nodes are dominant classes).

Based on this decision cluster tree, each node with a dominant class is a decision cluster. ADCC selects all leaf nodes $(C_1^1, C_1^2, C_2^2, C_3^2, C_2^0, C_3^1, C_4^2, C_5^2, C_5^1)$ as the classification model. When classifying a new object t, ADCC computes the distances between t and every leaf, then classify t to the label of the nearest leaf node (cluster).

ADCC outperforms some traditional classification algorithms, such as decision tree, SVM, KNN, on large high dimensional data, but it still has some drawbacks. It only selects leaves to construct the classification model. There is no evidence to prove that leaves are better than inner nodes when classifying new objects. Contrarily, in some instances, inner nodes may even be better than leaves. In a word, it is hard to judge which choice is better. Another shortcoming of ADCC is that all leaves are non-overlapping. The classification errors often occur in the boundary areas of adjacent clusters due to the mixture distribution of objects in different class. In next section, we will introduce our algorithm, which considers both leaves and inner nodes and selects useful nodes automatically.

6.2.1 Crotch Predictor

Definition 1. A *Crotch* is a sub-structure of a decision cluster tree, which is any inner node (decision cluster) with its children.

Crotch is sub-structure of a tree. A crotch only includes a father cluster and its children. In this paper, crotches are represented in a sequential form. For example, in Fig. 6.1, there are totally five crotches: $T(C_1^0, C_2^0, C_3^0)$, $C_1^0(C_1^1, C_2^1)$, $C_2^1(C_1^2, C_2^2, C_3^2)$, $C_3^0(C_3^1, C_4^1, C_5^1)$, $C_4^1(C_4^2, C_5^2)$.

Definition 2. Crotch Predictor is a crotch whose children have more than one dominant class.

Only crotch predictors are useful in classification. For example, in Fig. 6.1, $C_1^0(C_1^1, C_2^1)$ is a crotch but not a crotch predictor because both children have the dominant class A. This crotch cannot be used in classifying new samples.

When using a crotch predictor to classify a given sample, distances between this sample and the centers of children are computed first; then, this sample is classified with the dominant class of the nearest child cluster.

6.2.2 Crotch Ensemble

We can generate many crotch predictors from a decision cluster tree. The crotches on higher level are more general so that they can get lower accuracy when classifying new samples. Whereas, the crotches on lower level are more specific so that they can get higher accuracy in limited local domain. Our new classification method Crotch Ensemble integrates the information on both high level and low level to construct a strong classifier by co-working and inter-restraining.

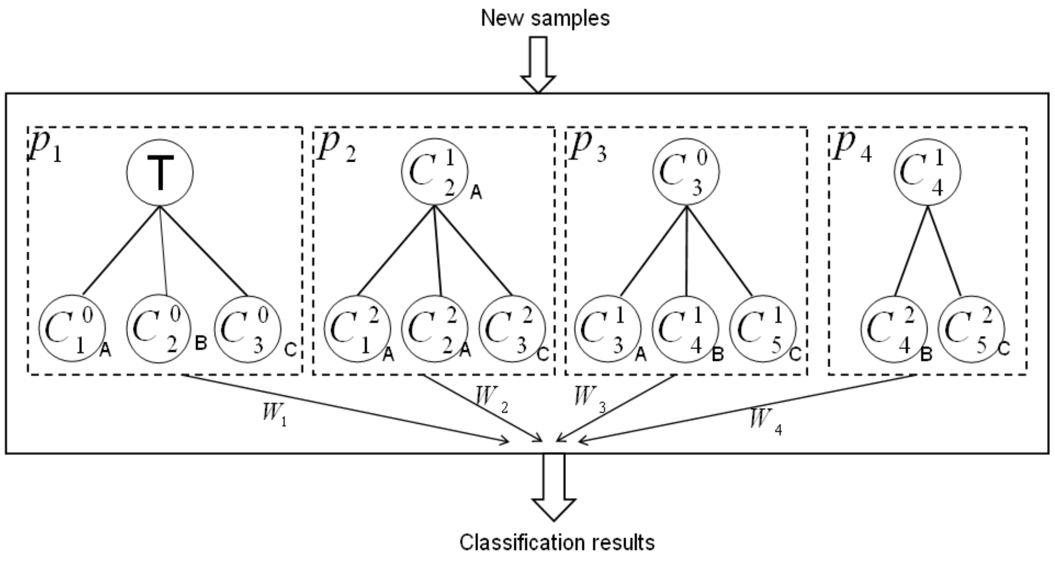


Fig. 6.2 The Crotch Ensemble built from the decision cluster tree in Fig. 6.1.

Definition 3. *Crotch Ensemble* is a set of crotch predictors which are generated from a decision cluster tree. It is also called *Crotch Ensemble Classifier*. The Crotch Ensemble $\mathscr{P}$ includes K crotch predictors: $\mathscr{P} = \{P_1, P_2, ..., P_K\}$. $\mathrm{P}_j(x)$ returns the predicted class label argued to x. *Crotch Ensemble* belongs to the Tree-based Model (defined in Section 3.4).

The crotch predictors are generated from a decision cluster tree and put together to construct a crotch ensemble classifier. For example, Fig. 6.2 is a crotch ensemble classifier built from the decision cluster tree in Fig. 6.1. This crotch ensemble includes all crotch predictors in Fig. 6.1. An objective function of Crotch Ensemble classifier is defined as following:

$$F(x) = \arg\max_{l} \sum_{j=1}^{K} \prod (P_j(x) = l) \tag{6-1}$$

where K is the number of crotch predictors in the crotch ensemble classifier, $P_j(\cdot)$ is the j th crotch predictor. $\prod(\cdot)$ is a logical function, it will return 1 if the condition is TRUE, otherwise it will return 0. The condition is determining whether the classification result of the crotch predictor to classify the sample x is the class l. The function $\arg\max$ returns the class label, l, with which Function (6-1) gets the maximal value. A crotch predictor classifies a sample to the child which is nearest to the sample like the classification method in [109].

K crotch predictors are generated from a decision cluster tree. Their contributions for classifying new samples are different due to each of them getting different classification accuracy when classifying new samples. Crotch predictors which can get better performance are more important than other predictors and should have bigger weights. Considering varying performance of each crotch predictor, a weight factor is added to Function (6-1). Then, the new objective function of Crotch Ensemble classifier is proposed as following:

$$F(x) = \arg\max_{l} \sum_{j=1}^{K} w_j \prod (P_j(x) = l) \tag{6-2}$$

In Function (6-2), W_j is the weight of the j th crotch predictor $P_j(\cdot)$, which satisfies $0 < W_j < 1$ and $\sum_{j=1}^{K} W_j = 1$. The performance of Crotch Ensemble classifier affects the value of weight $W_j\,(j = 1,\ldots,K)$. We propose a training method in Section 6.2.4 to get this weight distribution of all crotch predictors in the Crotch Ensemble.

6.2.3 Crotch Predictor Bounding

Crotch Ensemble uses all crotch predictors to classify new samples by maximizing function (2). However not all crotch predictors are useful for classifying a new sample. Fig. 6.3 shows an example of this problem. There are three predictors P_1, P_2 and P_3. x is a sample to be classified, P_1 and P_3 are near to x, but P_2 is very far from x and should be neglected when classifying x. So, it is necessary to define a predictor boundary to filter the crotch predictors when classifying new samples.

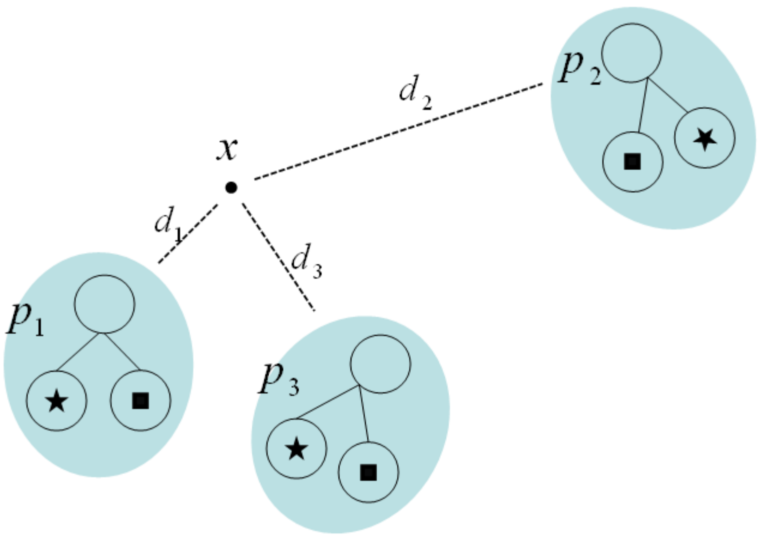


Fig. 6.3 Specify Bounding Predictor for the sample x. (d_1, d_2 and d_3 are distances between x and three predictors, star and rectangle denote the different dominant classes).

Suppose $\mathrm{P} = \{P_1, P_2, ..., P_K\}$ is the Crotch Ensemble, and it includes K crotch predictors. $\mathfrak{T}$ is the metric space. The Definition 5 is used to neglect far away crotch predictors.

Definition 4. The distance between a sample and a crotch predictor $\mathscr{P} = \{P_1, P_2, ..., P_K\}$ Dist(x, P_i) is the distance between this sample and the center of the nearest children node in this crotch.

Definition 5. Given a sample x in $\mathcal{T}$, the *Bounding Predictor* of x, $\mathcal{B}(x)$, is a subset of $\mathcal{P}$ with K' crotch predictors, that is $\mathcal{B}(x) = \{P_{k1}, P_{k2}, ..., P_{kK'}\}$, which satisfies (1) K' <= K, $\mathcal{B}(x) \subseteq \mathcal{P}$, (2) the distances between x_0 and each crotch predictor in $\mathcal{B}$ are smaller than a given distance threshold γ.

The following function (6-3) is used to determine whether the crotch predictor P_i belongs to the bounding predictor of the sample x when classifying x. If it returns 1, the testing predictor P_i belongs to the bounding predictor.

$$\Lambda_x(P_i) = \begin{cases} 1, Dist(x, P_i) < \gamma \\ 0, others \end{cases} \tag{6-3}$$

The value of distance threshold γ is $\theta \cdot \min_{P_i \in \mathcal{P}}(Dist(x, P_i))$ which multiplies the smallest distance between x and all P_i of $\mathcal{P}$ by θ. We call θ bounding predictor factor and it satisfies $\theta \geq 1$. If $\theta = 1$, the bounding predictor only includes the nearest crotch predictor(s).

For example, in Fig. 6.3, for Bounding Predictor of the sample x, is $\mathcal{B}(x) = \{P_1, P_3\}$.

We integrated the function (6-3) into the function (6-2) to get the new objective function of the Crotch Ensemble classifier as follows:

$$F(x) = \arg\max_l \sum_{j=1}^{K} \Lambda_x(P_j(x)) W_j \prod (P_j(x) = l) \tag{6-4}$$

Classifying a new sample with the function (6-4) takes the following three steps: (a) generating Crotch Ensemble from a decision cluster tree; (b) selecting Bounding Predictor for the new sample; (c) classifying the new sample to the class which maximize the function (6-4).

6.2.4 Crotch Weight Training

This section describes how the weights of Crotch Predictors are trained before Crotch Ensemble is used to do classification as discussed in Section 6.2.2.

For a sample x', each crotch predictor in Crotch Ensemble classifies x'. Some of them classify x' correctly and others classify x' wrongly. The crotch predictors who can classify x' correctly are added into $\mathscr{P}_r$, and those classify x' wrongly compose $\mathscr{P}_e$. $\mathscr{P}_r := \{P_j \in \mathscr{P} \mid P_j(x') = l(x')\}$ and $\mathscr{P}_e := \{P_j \in \mathscr{P} \mid P_j(x') \neq l(x')\}$, where $l(x')$ is the correct class label of x'. The corresponding weight sets of predictor sets $\mathscr{P}_r$ and $\mathscr{P}_e$ are $\mathscr{W}_r$ and $\mathscr{W}_e$ respectively. For the Crotch Ensemble $\mathscr{P}$, if $F(x') \neq l(x')$, x' is classified wrongly by this Crotch Ensemble, it must have,

$$\sum_{W_j \in \mathscr{W}_e} W_j > \sum_{W_i \in \mathscr{W}_r} W_i \tag{6-5}$$

Our goal is to train the weights for crotch predictors so that the important crotch predictors have bigger weights and others have smaller weights. A simple strategy is increasing weights for the crotch predictors which tend to classify new samples correctly and reducing weights for the crotch predictors which are more likely to get error results. Table 6.1 is the Crotch Predictor weight training algorithm.

Table 6.1 Crotch Predictor weight training algorithm.

```
Input:    Crotch Predictors Set $\mathscr{P} = \{P_1, P_2, ..., P_K\}$;
          Training Set, $\mathscr{X} = \{<x_1, l(x_1)>, <x_2, l(x_2)>, ..., <x_N, l(x_N)>\}$,
          $x_i (i = 1, 2, ..., N)$ is training sample, $l(x_i)$ is its correct label.
Parameters:    threshold of bounding predictor factor, $\theta$;
               extend coefficient, $\delta$; iteration times, $T$.
Output:    $W = \{W_1, W_2, ..., W_K\}$.
Begin
1.   initialize $W^0 = \{W_1^0, W_2^0, ..., W_K^0\}$ as $W_j^0 = 1/K, (j = 1, 2, ..., K)$;
2.   compute classification error rate $R_{err}(W^0)$ in training set $\mathscr{X}$ using
       classifier $F(x)$ feed with $W^0$;
3.   For $t$ from 1 to $T$
4.       $W^t = W^{t-1}$
5.       For each sample $x_i$ in $\mathscr{X}$ $(i = 1, 2, ..., N)$
6.           If $F(x_i) \neq l(x_i)$
7.               $W^t_{err} = \sum_{j=1}^{K} W^t_j \prod (P_j(x_i) \neq l(x_i))$;
8.               $W^t_{acc} = \sum_{j=1}^{K} W^t_j \prod (P_j(x_i) = l(x_i))$;
9.               $\tau = \prod (P_j(x_i) \neq l(x_i)) \cdot 2 - 1$;
10.              $W^t_j \leftarrow W^t_j \cdot (\frac{W_{acc}}{W_{err}})^{\delta \cdot \tau}$;
11.              re-normalize weights $W$;
12.          End If
13.      End For
14.      If $R_{err}(W^t) > R_{err}(W^{t-1})$
15.          $W = W^{t-1}$;
16.          break;
17.      End If
18.  EndFor
End
```

From Table 6.1, we can see that when Crotch Ensemble classifies the sample x' wrongly, the weights of crotch predictors are adjusted by multiplying coefficients. Them weights of those crotch predictors which classify x' wrongly multiply a coefficient smaller than 1, and the weights of those crotch predictors which classify x' correctly multiply a coefficient bigger than 1. Let $Err = \sum_{W_j \in \mathcal{P}_e} W_j$ and $Acc = \sum_{W_i \in \mathcal{P}_r} W_i$. From 6-5, there is $Err > Acc$ when Crotch Ensemble classify x' wrongly. The coefficient which should be smaller than 1 can be specified as $\frac{Acc}{Err}$, and the other coefficient which should be bigger than 1 can be specified as $\frac{Err}{Acc}$. Thus, the weights are adjusted as following,

$$\begin{cases} W_j \leftarrow W_j \cdot \dfrac{Acc}{Err}, W_j \in \mathcal{W}_e \\ W_i \leftarrow W_i \cdot \dfrac{Err}{Acc}, W_i \in \mathcal{W}_r \end{cases} \tag{6-6}$$

After adjusting of weights, the weights of those crotch predictors which classify x' correctly become bigger than before and the weights of those crotch predictors which classify x' wrongly become smaller than before. If the adjustment have the result

$$\sum_{W_j \in \mathcal{W}_e} W_j < \sum_{W_i \in \mathcal{W}_r} W_i \tag{6-7}$$

x' will be classified correctly after adjusting the weights of crotch predictors. We can control the extent of adjustment by step length parameter δ . That is,

$$W_k \leftarrow W_k \cdot (\frac{Acc}{Err})^{\delta \cdot \tau} \tag{6-8}$$

In expression (6-8), τ is sign coefficient, it is equal to 1 or -1. If $W_k \in \mathcal{P}_r$, τ = -1, and else if $W_k \in \mathcal{P}_e$, τ = 1. δ is extent coefficient. If δ is too small, the weights cannot be adjusted any more and the classification results of the wrongly classified sample cannot be corrected. If δ is too large, the weights are adjusted too much and lead to over-fitting problem.

Weights are adjusted once in every iteration step of Crotch Predictor weight training algorithm. In each iteration step, weights W should be re-normalized to satisfy $\sum W_k = 1$. Weights distribution of crotch predictors is improved in every iteration step.

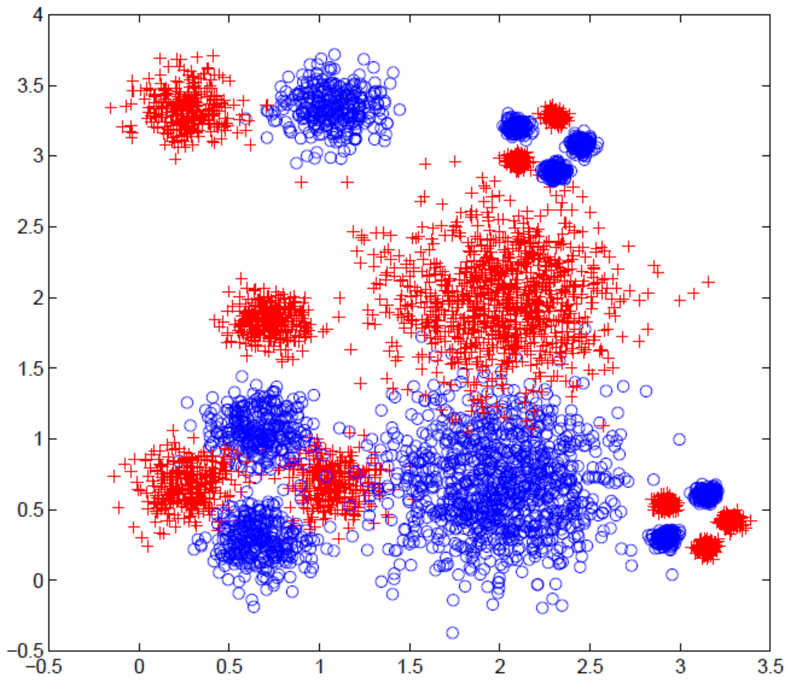


Fig. 6.4 Synthetic data set S.

Figure 6.4 draws a synthetic data set S which includes 5600 samples, 19 clusters, 2 classes in 2 dimensions. We compare ADCC with each step of Crotch Predictor weight training of Crotch Ensemble on this synthetic data set. We take 70% of the data as the training data to build decision cluster tree and to train crotch predictor weights. The remaining 30% data is testing data. When training crotch predictor weights, the Crotch Predictor weights training algorithm (seen in Table 6.1 takes 2 iteration steps to converge and stop. We classify the testing data with Crotch Ensemble based on different three weights distribution (initial weights, weights got from first iteration step, weights got from second iteration step). Table 6.2 lists the error rate of classification under

Crotch Ensemble with those three weights distribution and under ADCC method. We can see that crotch predictor weight training can improve the performance of Crotch Ensemble. Crotch Ensemble outperforms ADCC on this synthetic data set.

Table 6.2 Classification results are improved by crotch predictor weight training.

classifier	Initial weight	Iteration1	Iteration2	ADCC
Error rate	21.6%	4.8	3.7	7.5

6.3 Analysis

In this section, we use a simple example to show why Crotch Ensemble can work better than ADCC. The example data set D is shown in Fig. 6.5. There are 800 points in 4 normally distributed clusters that are classified into 2 classes marked with ‘×’ in red and ‘○’ in green respectively. The 4 stars indicate the 4 cluster centers. We first build an ADCC model to classify this data set and identify the misclassified points. Then, we show how Crotch Ensemble can correct these misclassifications. Finally, we discuss the bounding effect in crotch ensemble.

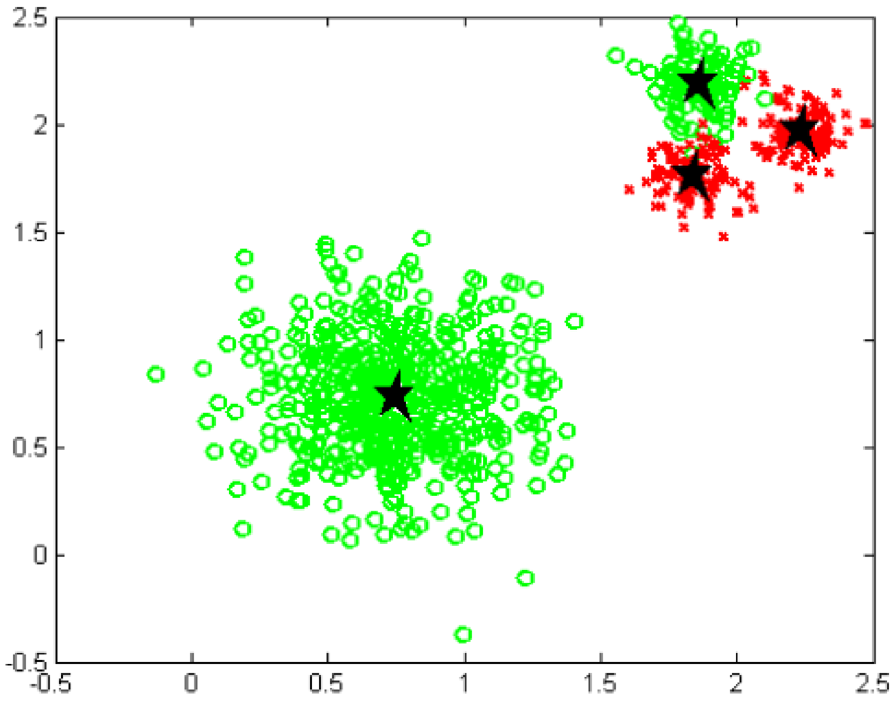


Fig. 6.5 Distribution of the data set D.

6.3.1 Original Decision Cluster Model and Its Shortcoming

Figure 6.6 shows the decision cluster tree built from data set D with the ADCC [109] method. Each decision cluster node is marked with two symbols, one representing the dominant class and the other one representing the cluster center. The legend of these symbols is shown on the left side of the figure. The symbols are used to indicate the dominant classes and cluster centers in the data distribution figures in this section. From this decision cluster tree, we select all leaf nodes to build a leaf-based decision cluster model $ADCC(L_1, L_2, L_3, L_4, L_5)$.

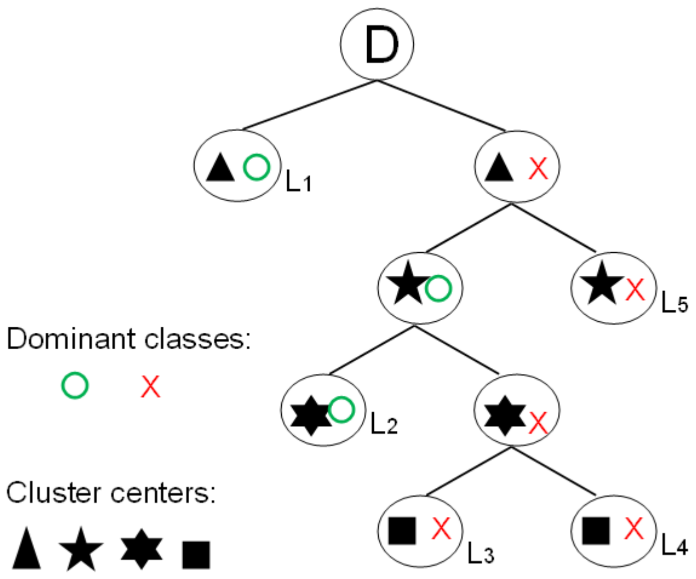


Fig. 6.6 The decision cluster tree built from data set D.

Given the 5 cluster centers in the ADCC model, we can draw a Voronoi diagram [119] in Fig. 6.7 to show how the data space is partitioned by the ADCC model. The 5 sub-regions L_1', L_2', L_3', L_4' and L_5' in Fig. 6.7 indicate the decision areas of the 5 decision clusters L_1, L_2, L_3, L_4 and L_5 in the ADCC model, respectively. A point falling in a decision area is classified as the dominant class of the decision cluster in the model. We can see that decision areas do not overlap in this leaf-based decision cluster model.

In this model, the classification decision is determined by only one decision cluster. The classification performance depends on the purity of the decision clusters, i.e., the percentage of the dominant class in the decision area. If points of different classes overlap in the neighbor decision clusters, classification errors occur. For example in the decision area L_3' of decision cluster L_3 in Fig. 6.7, there are some points in class '○' which would be misclassified as class '×' because the dominant class of L_3' is '×'. These are the inherent classification errors of the decision cluster which cannot be corrected in this model. However, some of these errors may be corrected by incorporating the parent decision clusters and neighbor decision clusters into the classification decision. This is the consideration of the Crotch Ensemble model.

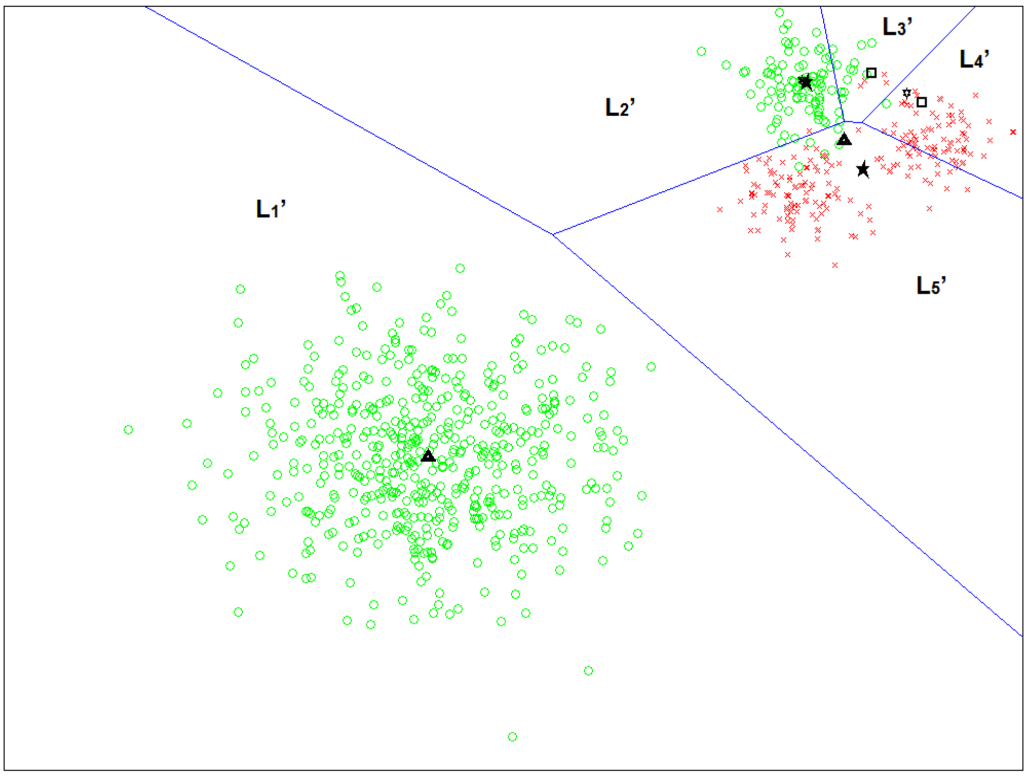


Fig. 6.7 Class decision areas partitioned by the ADCC model.

6.3.2 Correction by Crotch Ensemble

From the decision cluster tree in Fig. 6.6, we can extract four crotches and three of them can be used as crotch predictors. We ignore the crotch that includes leaf nodes L_3 and L_4 because the dominant classes of these leaf nodes are the same.

We plot the crotch ensemble model of 3 crotch predictors in different levels in Fig. 6.8. Predictor 1 is the crotch corresponding to the partitioning of the root node in Fig. 6.6. Predictor 2 is the crotch corresponding to the partitioning of the second child of the root node. Predictor 3 is the crotch corresponding to the partitioning of one child of Predictor 2. The Voronoi partition of the space with a line by each crotch predictor is also shown in Fig. 6.8. The two class labels in the ellipse indicate the dominant classes of the decision areas separated by the line. The bottom frame in Fig. 6.8 shows the overlap projection of three crotch predictors in the crotch ensemble model. Comparing Fig. 6.7 with Fig. 6.8, we can observe that some misclassified points of class '○' by decision cluster L_3 in Fig. 6.7 can be correctly classified by Predictor 2 and Predictor 3.

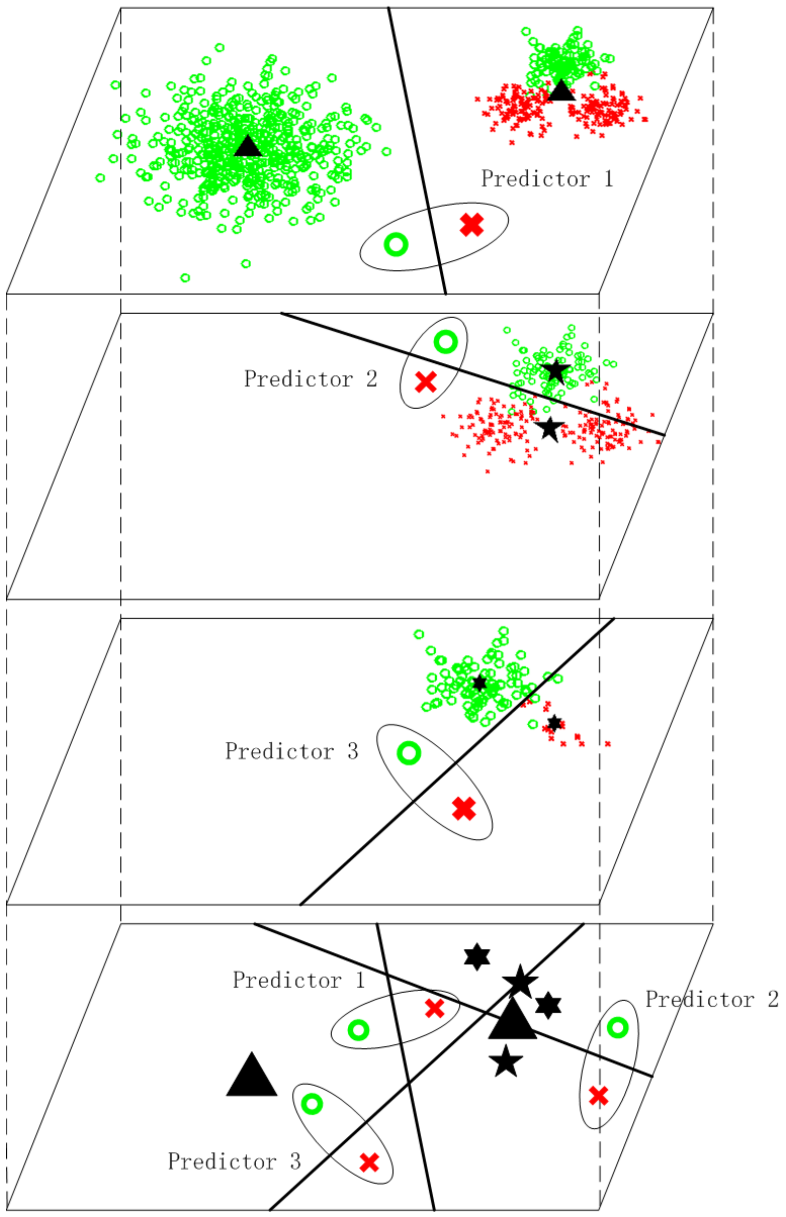


Fig. 6.8 Crotch Ensemble built from the tee in Fig. 6.6.

Since the decision area L_1' in Fig. 6.7 does not contain misclassified points, we remove decision area L_1' and plot the rest decision areas L_2', L_3', L_4' and L_5' in

Fig. 6.9. The misclassified points are represented as '□'. We can see all these points occur in decision area 'L_3' whose dominant class is '×'. However, the true class of these misclassified points is '○'. With the Crotch Ensemble model to classify the points, we can find that Predictor 1 classifies these points as class '×' whereas Predictor 2 and Predictor 3 classify these points as class '○' according to the distances between these points and the Crotch Predictors. The weight W_1 of Predictor 1 is 0.26 and the weights W_2 and W_3 for Predictor 2 and Predictor 3 are 0.25 and 0.49, respectively. Because of $W_1 < W_2 + W_3$, these points are classified as class '○'. Therefore, the misclassifications of some points by the ADCC model are corrected by the Crotch Ensemble model.

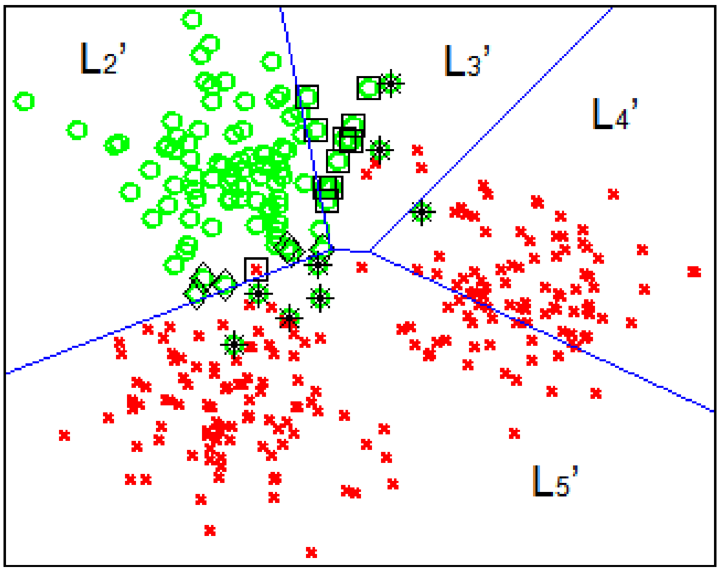


Fig. 6.9 Samples are wrongly classified by ADCC and Crotch Ensemble ('□' denotes the samples which are wrongly classified by ADCC but corrected by Crotch Ensemble, '◇' denotes the samples which wrongly classified by Crotch Ensemble but rightly classified by ADCC, and * denotes the samples which are wrongly classified by both methods).

6.3.3 Bounding Crotch Predictors

In the crotch ensemble model, a new object is classified by the collective decision of multiple crotch predictors, including the primary crotch predictor that is the closest to the project, the crotch predictors that overlap but are at the higher levels of the primary crotch predictor in the decision cluster tree, and the crotch predictors that are the close neighbors to the primary crotch predictor. In a large crotch ensemble model that consists of many crotch predictors, not every crotch predictor makes positive contribution to the right classification decision. Some crotch predictors that are far away from the primary crotch predictor can make negative contribution to the classification. Therefore, we define a distance threshold γ to exclude the crotch predictors whose distances to the object to be classified exceed γ. The crotch predictors who are selected from the crotch ensemble model to classify a new object are called bounding predictors. Depending on the object locations, different sets of bounding predictors are selected to classify different objects.

Figure 6.10 shows an example of the crotch ensemble model built from the decision cluster tree in Fig. 6.8. Given an object X, its distance to the primary crotch predictor Predictor 1 is $Dist(X, P1)$. Let the bounding predictor factor θ > 2, the crotch predictors within the dashed lines will be included as the bounding predictors to classify X. In this case, Predictor 2 is included whereas Predictor 3 is excluded.

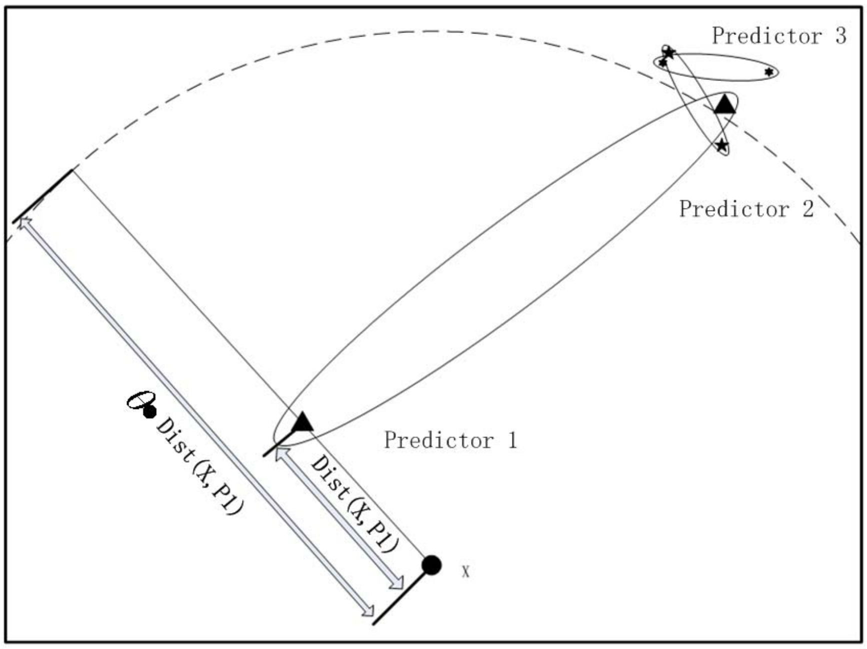


Fig. 6.10 Finding Bounding Predictor.

Figure 6.11 further illustrates how the classification is made by the joint decision of bounding crotch predictors. Given a bounding predictor factor $\theta > 1$, we draw two hyperbolas $\Gamma_{1,2}$ and $\Gamma_{1,3}$ along the axis linking Predictor 1 and Predictor 2, and the axis linking Predictor 1 and Predictor 3, respectively. For the simplification, hyperbolas $\Gamma_{2,3}$ is not drawn in Fig. 6.11. Here we take $\Gamma_{1,2}$ and $\Gamma_{1,3}$ for example. The points on the left side of $\Gamma_{1,2}$ whose Bounding Predictor include Predictor 1 but do not include Predictor 2 and the points on the other side of $\Gamma_{1,2}$ include Predictor 2 but do not include Predictor 1. It is the same way to analyze the points on the two sides of $\Gamma_{1,3}$. The Bounding Predictors of the points in decision area D_1 only include Predictor 1, so the classification decision is made on the points in decision area D_1 by the Predictor 1 with its weight W_1. In the same way, the Bounding Predictors of the points in decision area D_2 and D_3 are $\{P_1, P_2\}$ and $\{P_1, P_3\}$ respectively. The classification decision in the decision area D_2 is made jointly by Predictor 1 with its weight W_1 and

Predictor 2 with its weight W_2 whereas the classification decision in the decision area D_3 is made jointly by Predictor 1 with its weight W_1 and Predictor 3 with its weight W_3. D_2 and D_3 are the boundary areas of the three crotch predictors. In the ADCC model, classification errors often occur in the boundary areas due to the mixture distribution of objects in different classes. Some misclassifications can be corrected by the joint decision in the bounding crotch ensemble model.

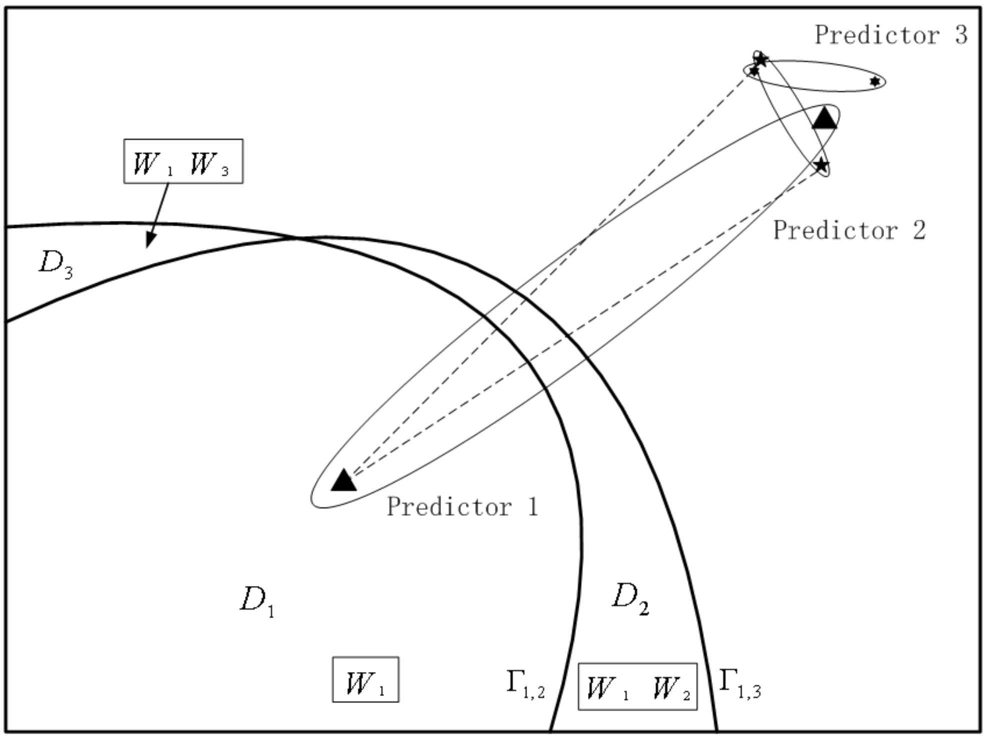


Fig. 6.11 Classification with Bounding Predictor.

The effect of bounding crotch predictors is significant. Taking the data in Fig. 6.4 as an example, 1039 points were misclassified with the crotch ensemble model without bounding. After bounding the crotch predictors, 972 misclassifications were corrected.

6.4 Experiments

In this section, we present the experiments we have conducted on both synthetic and real data sets. The experiments on synthetic data have analyzed the parameters of Crotch Ensemble and compared the performance with other algorithms (including Decision

Tree and ADCC) by increasing the size and the number of dimensions of the data sets. We also compared the classification performance of these classification algorithms on real data sets. All experiments were conducted on an Intel(R) Xeon(R), 1.60 GHz computer with 8GB memory.

6.4.1 Analyzing Parameters

We generated the synthetic data set S_1 whose characteristics are given in Table 6.3. S_1 includes two classes, each with 3 clusters. Each cluster contains 200 samples following a normal distribution. Some clusters are overlapping. The shape and distribution are shown in Fig. 6.12.

Table 6.3 Generation characteristics of S_1 (two orientation variances are 7 and 1).

	Class1			Class2		
	Cluster1	Cluster2	Cluster3	Cluster1	Cluster2	Cluster3
centroid	(10,10)	(0,10)	(-10,10)	(8,12)	(-4,10)	(-10,6)
orientation	$\pi/4$	$13\pi/12$	$2\pi/3$	$\pi/4$	$13\pi/12$	$2\pi/3$

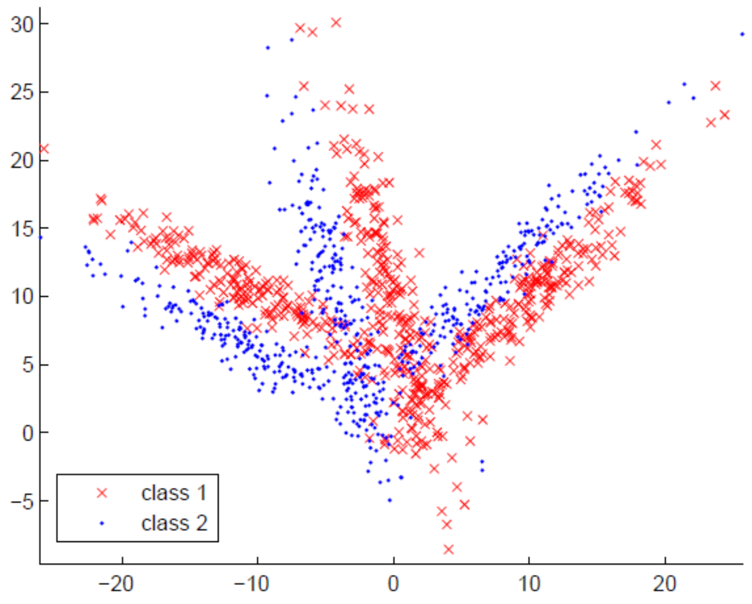


Fig. 6.12 The distribution of data set S_1.

There are two important parameters θ and δ in Crotch Ensemble. Their values will influence the performance of the classification model. $\theta \geq 1$ is used to compute the threshold of crotch predictor bounding (see Section 3.3). It affects the number of the predictors selected to classify a new sample. If $\theta=1$, only the nearest predictor is selected to perform classification. In this condition, the Crotch Ensemble classification method is the same as the classification step of ADCC. The value of θ should not be too big or too small. If θ is too big, some too far away predictors are selected but they are not useful for classifying the current sample. If the value of θ is too small, the useful predictors may be missed. We execute Crotch Ensemble on the data set S_1 to demonstrate how the parameter θ influences the classification result. Fig. 6.13 shows the classification accuracy against different values of θ on Crotch Ensemble. We can see that the classification accuracy is increasing by increasing the value of θ at the beginning but it is decreasing after the value of θ reaches around 8. When the value of θ is bigger than 10, the accuracy is not stable. From our experience the range from 2 to 8 is a good choice of the value of θ. The best choices may be different for different data sets, but the trends of the accuracy lines are the similar.

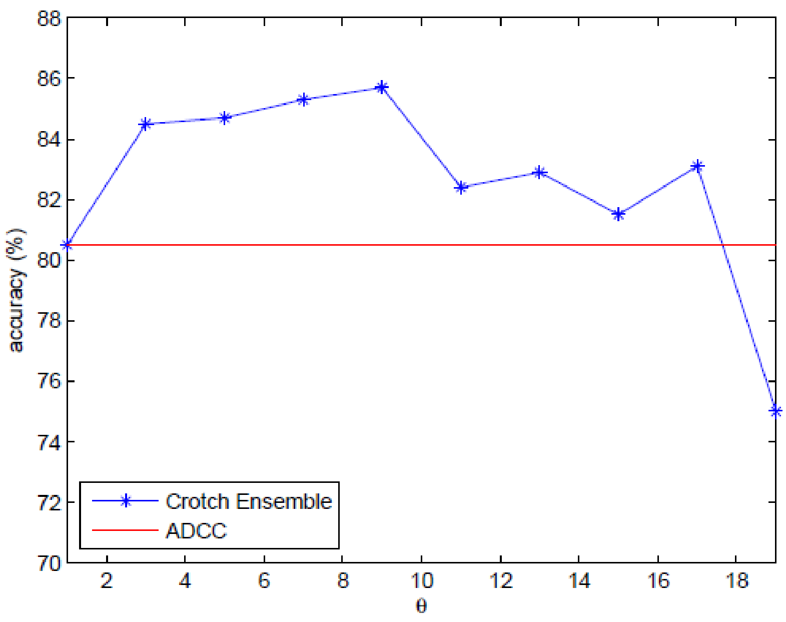


Fig. 6.13 Performance effected by θ.

Another important parameter δ controls the updating range when adjusting the crotch weights (see Section 3.4). It controls the extent of adjusting weights of each crotch predictor in every iteration step. We still execute Crotch Ensemble on the data set S_1 to demonstrate how the parameter δ influences the classification result. Fig. 6.14 shows the classification accuracy against different values of δ on Crotch Ensemble. We can see that the classification accuracy reaches the highest value when δ is around 1. The classification accuracy is increasing by increasing the value of δ at the beginning but it is decreasing after the value of δ reaches around 1. From our experience the range from 0.6 to 1.4 is a good choice of the value of δ.

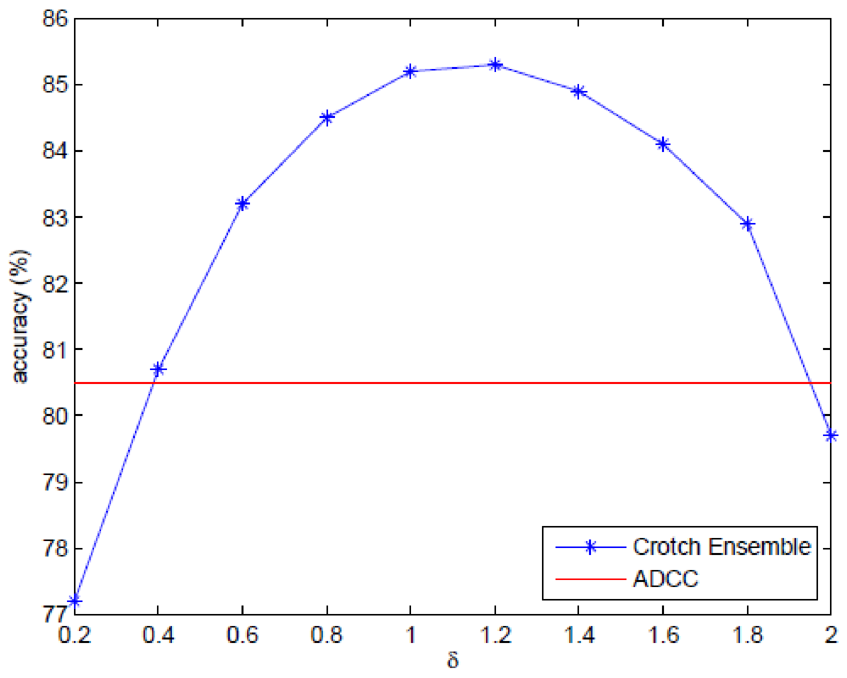


Fig. 6.14 Performance effected by δ.

6.4.2 Scalability

We generated two groups of synthetic data sets with different numbers of dimensions and instances (shown in Table 6.4). Each data set contains ten clusters randomly generated with normal distributions. Each cluster is randomly labeled with one of the three classes. In each run, we used 70% of data as training data and the remaining 30% as testing data. We compared the performance of Crotch Ensemble with ADCC and decision tree algorithm J48 with different number of dimensions and instances respectively. Table 6.4 shows the details of data sets: data sets A1 to A8 have the number of dimensions varying from 10 to 10,000; data sets B1 to B8 have the number of data instances varying from 50 to 100,000.

Table 6.4 Two groups of synthetic data sets (each having three classes).

Data sets	Dimensions	Instances	Data sets	Dimensions	Instances
A1	5	5,000	B1	4	3,000
A2	20	5,000	B2	4	9,000
A3	50	5,000	B3	4	15,000
A4	100	5,000	B4	4	30,000
A5	200	5,000	B5	4	45,000
A6	300	5,000	B6	4	60,000
A7	400	5,000	B7	4	75,000
A8	500	5,000	B8	4	90,000

The experimental results are shown in Fig. 6.15 and Fig. 6.16. Fig. 6.15 shows the classification accuracy against the number of instances. We can see that the classification accuracy of J48 decreases obviously when the number of dimension is increasing. Crotch Ensemble and ADCC perform better than J48 and are stable. Crotch Ensemble and ADCC have similar performance on high dimensional data sets.

Fig. 6.16 shows the classification accuracy against the number of dimensions. Because Crotch Ensemble needs more samples to train weights, it performs worse in small data set. Crotch Ensemble performs better than ADCC and J48 on large data sets.

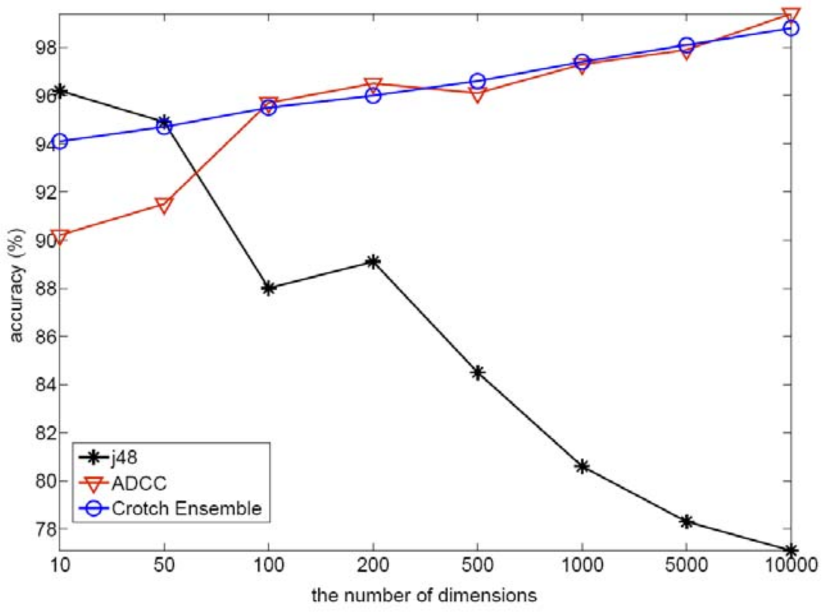


Fig. 6.15 Classification accuracy vs. dimension number.

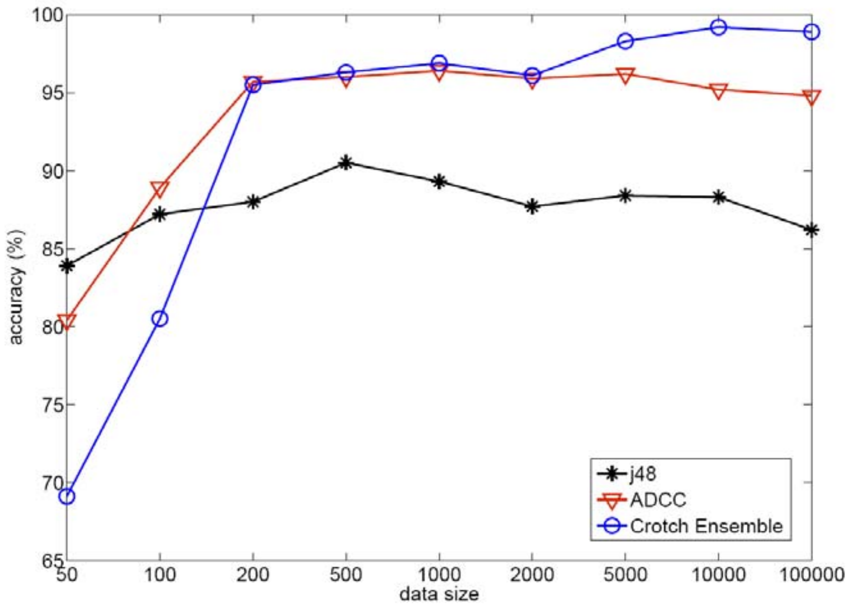


Fig. 6.16 Classification accuracy vs. data size.

6.4.3 Experiments on Real Data

We show the comparison results of Crotch Ensemble algorithm and other four classification methods: decision tree (J48), original KNN, Random Forest and ADCC on five real data sets which are taken from the UCI machine learning data repository [23]. We implemented Crotch Ensemble and ADCC in java. J48, KNN and Random Forest are implemented in Weka [126]. *BreastCancer* is authored by Prognostic Wisconsin Breast Cancer Database. For the *Reuters* data set, the standard document frequency method was used to select relevant attributes from the original feature space. *Madelon* is an artificial data set, which was part of the NIPS 2003 feature selection challenge. *optdigits* is Optical Recognition of Handwritten Digits Data Set. Table 6.5 lists these data sets.

Table 6.5 Four real data sets.

Data Set	Instances	Dimensions	Classes	Training	Testing
Breastcancer	569	32	2	398	171
Madelon	4400	500	2	2000	2400
Reuters	9980	337	10	6986	2994
Optidigits	5620	64	10	3823	1797

The comparative results with 10-fold cross-validation of Crotch Ensemble and other four algorithms are shown in Table 6.6. We can see that Crotch Ensemble outperforms other algorithms in most conditions.

Table 6.6 Classification results on four real data sets.

Algorithm	Breastcancer	Madelon	Reuters	Optidigits
Crotch	**96.7%**	**72.3%**	**71.5%**	91.7%
ADCC	94.9%	72.2%	69.3%	90.9%
Decision	94.2%	67.2%	67.7%	89.7%
KNN	83.9%	63.2%	65.1%	80.5%
Random	95.7%	59.7%	67.8%	**96.8%**

On the same data sets, we record the wrongly classified samples by ADCC and Crotch Ensemble. Table 6.7 shows how much the Crotch Ensemble corrects the ADCC result. Crotch Ensemble is abbreviated as CE.

Table 6.7 Crotch Ensemble correction.

Data set	ADCC wrongly classified but CE rightly classified	ADCC rightly classified but CE wrongly classified
Breastcancer	7	5
Madelon	321	315
Reuters	674	633
Optidigits	75	57

6.5 Conclusion

In this chapter, we propose a new classification method Crotch Ensemble based on the decision cluster tree. We have presented the method of selecting crotch predictor and the algorithm of training the weights for each crotch predictor. We define the Bounding Predictor to filter the crotch predictors when classifying new samples. Those too far away predictors are neglected when classifying a sample using Bounding Predictor of this sample. We analyzed why Crotch Ensemble can correct some misclassified samples by ADCC and how to choose the near crotch predictors.

We have presented experimental results on both synthetic and real world data sets to analyze the parameters and to compare the performance of Crotch Ensemble with those of other well-known classification methods and our previous classification method ADCC. The comparison results have shown that Crotch Ensemble has advantages in classifying large, high dimensional data with multiple classes and performs better than the previous method ADCC. The experimental results have also demonstrated Crotch Ensemble can correct ADCC's results and Crotch Ensemble with Predictor Bounding can filter too far away crotch predictors to improve the classification result.

Chapter 7

A Subspace Decision Cluster Classifier for Text Classification

In this chapter, a new classification method (SDCC) for high dimensional text data with multiple classes is proposed. In this method, a subspace decision cluster classification (SDCC) model consists of a set of disjoint subspace decision clusters, each labeled with a dominant class to determine the class of new objects falling in the cluster. A subspace decision cluster tree is first generated from a training data set by recursively calling a subspace clustering algorithm Entropy Weighting k-means algorithm. Then, the SDCC model is extracted from the subspace decision cluster tree. Various tests including Anderson-Darling test are used to determine the stopping condition of the tree growing. A series of experiments on real text data sets have been conducted. Their results show that the new classification method (SDCC) outperforms the existing methods like decision tree and SVM. SDCC is particularly suitable for large, high dimensional sparse text data with many classes.

7.1 Introduction

Text classification aims at assigning class labels to text records. It is widely extended in many web mining areas, such as Blog documents classification [128], robust classification of rare queries in search engines [103] and hierarchical text classification [101,129]. Text data is a typical example of high dimensional data. Classifying high dimensional data faces many challenges [21]. There are usually thousands or more dimensions, which is the total number of unique words in text data. In a text data set, records (documents) related to a particular topic, for example, politics, usually contain a subset of words (dimensions) which are discriminative for these documents. Those dimensions describing politics are less likely to exist in documents of other topics such as sports. This situation implies that different dimensions have different contributions for documents in different classes. Furthermore, in high dimensional data, the variations of distances between any two data samples becomes less significant [130], so meaningful clusters can only be found in some subspace of the whole high dimensional space [131]. We can find clusters from subspaces of the dimensions instead of the entire dimensions by subspace clustering algorithms. We intend to apply the advantages of subspace clustering into the classification to deal with the sparse high dimensional data. Cluster-based classification model follows a probability mixture model in which each cluster is considered as a distribution of objects of one class in the multidimensional data space [120]. Objects in the same cluster tend to have the same class label. On the other hand, it can be viewed as mapping a class to one or more clusters. In this way, classification can be seen as a clustering problem that can be solved with a clustering process.

In this chapter, we integrate a subspace clustering algorithm Entropy Weighting k-Means (EWKM) [3] into our classification framework. There are three main steps in this work. Fist we build a subspace decision cluster tree by recursively calling the subspace clustering algorithm EWKM. In every partition, the sub-clusters are found in their own subspaces instead of the entire data space. After building the subspace decision cluster tree, we generate a classifier from the tree. Finally, we specify a distance metric for our classifier to classify new samples. In growing the subspace decision cluster tree, we adopt the EWKM to deal with the data sparsity problem which exists in high dimensional data. EWKM extends the k-Means clustering process to calculate a weight for each dimension in each cluster and use the weight distribution to identify the subsets of important dimensions instead of the whole dimensional space. We adopt Anderson Darling Test [6,7] as a stopping criterion in tree growing. Our experimental results on many real text data sets generated from 20-newsgroup corpus demonstrate that our Subspace Decision Cluster Classifier outperforms other classifiers including SVM, decision tree.

The rest of this chapter is organized as follows. In Section 7.2, we briefly review the subspace clustering algorithm Entropy Weighting k-Means (EWKM). In Section 7.3, we introduce the construction method of the subspace decision cluster tree and the methods of model selection and classification. In Section 7.4, experimental results and comparisons are reported. In Section 7.5, we conclude this chapter.

7.2 Entropy Weighting k-Means Algorithm

In this section, we briefly review the subspace clustering algorithm Entropy Weighting k-Means (EWKM) [3] used for clustering high-dimensional sparse data.

High dimensional data is common in real world data mining applications, such as text data mining, bioinformatics data mining and business data mining. Sparsity is a classic problem of high dimensional data. In text data mining, documents related to a particular topic such as culture are characterized by a subset of words. Words appearing in culture documents may not appear in sport documents. The clustering algorithms dealing with this kind of high dimensional sparse data are called subspace clustering algorithms. In the subspace clustering, each cluster contains a set of samples identified by a subset of dimensions.

Entropy Weighting k-Means (EWKM) algorithm is a soft subspace clustering algorithm which clusters data samples in the entire dimensional space but assigns different weights to different dimensions for each cluster during clustering process [3]. The dimensions which are more important for identifying the corresponding cluster will get larger weights. Dimensions make different contributions to the evaluation of objects in a cluster. It is different from Weighting k-means (WKM) which is adopted in the work of Chapter 4. WKM is not a subspace algorithm. It just assigns smaller weights for noisy dimensions and larger weights for non-noisy dimensions. All clusters have the same weight distribution of the entire data space in WKM. However, EWKM can find clusters in subspaces by giving different weight distributions for different clusters.

The objective function is written as follows:

$$F(W,Z,\Lambda)=\sum_{l=1}^{k}[\sum_{j=1}^{n}\sum_{i=1}^{m}\omega_{lj}\lambda_{li}(z_{li}-x_{ji})^2+\gamma\sum_{i=1}^{m}\lambda_{li}\log\lambda_{li}] \tag{7-1}$$

Subject to

$$\begin{cases}\sum_{l=1}^{k}\omega_{lj}=1, 1\le j\le n, 1\le l\le k, \omega_{lj}\in\{0,1\}\\ \sum_{i=1}^{m}\lambda_{li}=1, 1\le l\le k, 1\le i\le m, 0\le\lambda_{li}\le 1\end{cases} \tag{7-2}$$

In Function (7-1), n is the number of objects, k is the number of clusters and m is the number of dimensions. Λ is the weight distribution of dimensions for each cluster, Z is the centroids of the clusters, and W is partition matrix. ω_{lj} is the degree of membership of the j th object belonging to the l th cluster. λ_{li} is the weight for the i th dimension in the l th cluster. x_{ji} is value of the i th dimension the j th object and z_{li} is the value of the i th component of the l th cluster center. The details of the EWKM algorithm can be found in [3].

7.3 Subspace Decision Cluster Tree

In this section we demonstrate the techniques during the construction process of a subspace decision cluster tree. The Entropy Weighting k-Means (EWKM) [3] clustering algorithm is adopted to build a subspace decision cluster tree because it is efficient and able to automatically find clusters from subspace of data instead of the entire data space and compute the attribute weights from the training data to reduce the effect of noisy attributes. Sub-clusters which are generated from the same father cluster are represented in different subsets of attributes.

7.3.1 Definitions

Let $X = \{x_1, x_2, ..., x_n\}$ be a training data set of n classified objects, each described by m attributes and labeled by one of K classes.

Definition 1. The dominant class in a cluster is the class that the majority of objects are labeled. A cluster with a dominant class is called a decision cluster. The percentage of the dominant class in the cluster defines the confidence level of the decision cluster.

Definition 2. A subspace decision cluster (SDC) is a cluster which is generated by a subspace clustering algorithm and exists in the subspace of all dimensions.

Definition 3. A subspace decision cluster classifier (SDCC) consists of a subset of subspace decision clusters generalized from the whole training data set.

In principle, any subset of subspace decision clusters can form a SDCC model. However, the model performance on classification accuracy depends on the subspace decision clusters generated by the clustering process and also depends on which subspace decision clusters are selected to form the classification model. Therefore, the following two processes are crucial: (1) generation of a set of subspace decision clusters and (2) selection of a subset of these clusters for the model. Below, we present a method to generate a set of nested clusters that form a subspace decision cluster tree for classification model selection.

7.3.2 Subspace Decision Cluster Classifier (SDCC) Algorithm

Table 7.1 shows the algorithm of automatic construction of subspace decision cluster tree and the selection of the SDCC model.

Table 7.1 SDCC Algorithm.

```
Input: A training data set $T$ (with $m$ dimensions and $K$ classes).
Output: A classification model $SDCC-\mathrm{mod}\,el$.
Tree construction
1.  initialize a subspace decision cluster tree $SDCT$ with root $\{T\}$;
2.  sign the root as internal node;
3.  for each internal node $X$ in $SDCT$
4.      if ($Ter\min al-Test(X)$)
5.          sign $X$ as $leafnode$;
6.          break;
7.      end if
8.      $k = K-Selection(X,\alpha)$;
9.      $CENTER-ARRAY$ = $C-Selection(k,X)$;//Compute initial centers
10.     run $EWKM$ on $X$ with $k$ and $CENTER-ARRAY$;
11.     sign $k$ sub-clusters as $\mathrm{int}\,ernalnode$;
12.     assign $k$ sub-clusters to $SDCT$;
13. end for
Model selection
14. extract all leaf nodes from $SDCT$ as classification model $SDCC-\mathrm{mod}\,el$;
15. return $SDCC-\mathrm{mod}\,el$;
End
```

7.3.3 Constructing a Subspace Decision Cluster Tree

Subspace clustering algorithms can find clusters from subspace instead of the entire attributes space. EWKM is one of soft subspace clustering algorithms [3] which is to cluster objects in the entire data space but assign different weighting values to the attributes of clusters in the clustering process, based on the importance of the attributes in identifying the corresponding clusters. EWKM can automatically weigh attributes on their importance during the clustering process.

In EWKM algorithm, a new step is introduced to the basic k-means algorithm to update the weight entropy based on the current partition of data. The objective function with this new term can minimize the within-cluster dispersion and maximize the negative weight entropy (the second term in Function (7-1) to stimulate more dimensions to contribute to the identification of clusters simultaneously. For the high dimensional text data sets, EWKM algorithm outperforms the other subspace clustering algorithms [2].

In this chapter, the construction of the subspace decision cluster tree is a recursive division process by recursively executing the EWKM clustering algorithm. To partition a cluster into sub-clusters with EWKM algorithm, we need to specify a parameter k which is the number of sub-clusters to be generated. We also need to specify the initial centers for each sub-cluster. Here, we still take advantage of the class label information as in Chapter 4. We propose some methods to control the iteration and improve the clustering process. These methods include the method of selecting k for EWKM ($K-Selection(X,\alpha)$), the method of selecting initial centers for EWKM ($C-Selection(k,X)$) and the termination test method ($\mathrm{Terminal}-\mathrm{Test}\left(\mathrm{X}\right)$), where X is the current node to be partitioned, α is a threshold and k is the number of sub-clusters. These methods are explained in Chapter 4.

We determine the value of k by considering the distribution of classes. We compute the percentage of samples in each class compared with all samples in the current node. Given a threshold α, let k be the number of classes whose percentages are larger than or equal to α. This paper implemented the selection of k by the function $K-Selection(X,\alpha)$: X is the current sample set and α is the threshold. The function returns the value of k. EWKM algorithm is a local search approximation algorithm. If we can specify better centers at the beginning, it can reduce the number of iterations and get better clustering result more quickly. In this chapter we still use supervised selection method $C-Selection(k,X)$. Using the class centers can accelerate the process of selecting the initial centers as well as improve the accuracy of determination of initial centers vastly. The stopping test stage which determines whether a node should be further divided or not, is vital for the whole tree construction and will influence the quality of the tree as well as the quality and computing efficiency of the classifier. We consider the size, class purity and data distribution together when doing the stopping test. We implemented a termination test method considering the above three aspects by $\mathrm{Terminal}-\mathrm{Test}\left(\mathrm{X}\right)$ method in Chapter 4. The efficiency of these methods have been shown in Chapter 4.

7.3.4 Model Selection and Classification

After a subspace decision cluster tree is built, any subset of disjoint decision clusters makes a SDCC model. In this work, we select the leaf nodes of the subspace decision cluster tree because leaf nodes are disjoint with each other and all of them as a whole cover all training samples.

The classification model is used to classify new objects as the following: (1) Select a distance function specific for classification; (2) Compute the distances between a new object and the centers of the decision clusters in the model; (3) Identify the decision cluster with the shortest distance to the object and assign the label of the decision cluster to the new object as its class.

In this work, we use the cosine distance function which is often used in text mining as follows:

$$Sim(A,B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{7-3}$$

Function (7-3) is a measure of similarity between two documents by finding the cosine of the angle between them. Given two vectors (representing the two documents), A and B, the cosine similarity is represented using their dot product and magnitudes as shown in Function (7-3).

7.4 Experiments

In this section, we present the experiments we have conducted on real text data sets. We compared the classification performance of our classification algorithm and other classification algorithms. All experiments were conducted on in Intel(R) Xeon(R), 1.60 GHz computer with 8GB memory.

7.4.1 Evaluation Method

We compare our SDCC on the Text Classification task with other classification methods such as J48 [27](one of decision tree algorithms), SMO [124] (one of SVM methods in Weka [126]) and libSVM [132]. In this chapter, we focus on classifying multi-class text classification. 10-fold cross-validation has been accomplished for each data set.

To evaluate the classification performance for each class, F1 [50], precision [48] and recall [49] as shown in the Equation (7-4), (7-5) and (7-6) were used. To measure the average performance for all classes of the whole data sets, the macro-averaging F1 and micro-averaging F1 were used. F1 is a combined form for precision (P) and recall (R), which is defined as Equation (7-4).

$$F1 = \frac{2PR}{P+R} \tag{7-4}$$

Table 7.2 The contingency table for class c_i.

Class c_i		Real class	
		Positive	Negative
Classifier results	Positive	TP_i	FP_i
	Negative	FN_i	TN_i

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{7-5}$$

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{7-6}$$

Micro-averaging: P and R are obtained by summing over all individual decisions as shown in Equation (7-7), (7-8) and (7-9), where μ indicates micro-averaging.

$$P^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} (TP_i + FP_i)} \tag{7-7}$$

$$R^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} (TP_i + FN_i)} \tag{7-8}$$

$$F1^{\mu} = \frac{2P^{\mu}R^{\mu}}{P^{\mu} + R^{\mu}} \tag{7-9}$$

Macro-averaging: precision and recall are first evaluated "locally" for each class, and then "globally" by averaging over the results of the different classes as shown in Equation (7-10), (7-11), and (7-12), where M indicates macro-averaging.

$$P^{M} = \frac{\sum_{i=1}^{K} P_i}{K} \tag{7-10}$$

$$R^{M} = \frac{\sum_{i=1}^{K} R_i}{K} \tag{7-11}$$

$$F1^M = \frac{2P^M R^M}{P^M + R^M} \tag{7-12}$$

In the rest of this chapter, we use $\mu - F1$ and $M - F1$ to denote micro-F1 and macro-F1 respectively.

7.4.2 Data sets and Experimental Settings

Our experiments were done on 20-Newsgroups data which is taken from the UCI machine learning data repository [23]. The original text data was first preprocessed to strip the news messages from the special tags and the email headers and eliminate the stem words and stop words. The dimension (word) in each document was weighted by the Term Frequency (TF). Table 7.3 lists eight data sets built from the 20-Newsgroups data. We preprocessed these data sets by deleting some dimensions with smallest TF value. Several thousands words is enough for text data. We also keep all words for T1, T2 and T8 to show our algorithm is efficient on very high dimensional sparse data. Data sets have different cluster properties. Some of them have semantically similar classes (such as T3, T6), whereas others contain semantically different classes (such as T1, T2, T7, T8). Some of them have overlapping words (dimensions) (such as T5, T6, T7), while some of them contain the unbalanced number of documents in each class (such as T2).

We evaluate our SDCC classifier on the real Text Classification task by comparing SDCC's performance with decision tree (J48) [24] and the two SVM methods (SMO [124] and libSVM [132]). Weka [126] implementations of J48 and SMO were used in our comparisons. We adopted Weka LibSVM (WLSVM) [132] which combines the strength of Weka and LibSVM. Weka has a GUI and produces many useful statistics and is easy to use. LibSVM runs much faster than Weka SMO and supports several SVM methods. WLSVM can be viewed as an implementation of the LibSVM running under Weka environment. Our new method SDCC is implemented in java. For the two SVM tools (SMO and libSVM), the linear kernel and the default settings were used which yields the best results in our experiments. For SDCC, we always set α (the parameter in $K-Selection(X,\alpha)$) equal to 0.05, δ (a parameter in $\mathrm{Terminal}-\mathrm{Test}(\mathrm{X})$) equal to the 10% of the number of samples in smallest class, and β (another parameter in $\mathrm{Terminal}-\mathrm{Test}(\mathrm{X})$) equal to 90%.

Table 7.3 Text data sets generated from the 20-Newsgroups data.

Data sets	classes	Dimensions	Size	Data sets	classes	Dimensions	Size
T1	2.comp.graphics	13151	100	T2	2.comp.graphics	10348	120
	10.rec.sport.baseball		100		10.rec.sport.baseball		100
	15.sci.space		100		15.sci.space		59
	18.talk.politics.mideast		100		18.talk.politics.mideast		20
T3	2.comp.graphics	9424	120	T4	2.comp.graphics	13218	120
	3.compcos.ms.windows.misc		120		3.compcos.ms.windows.misc		120
	4.comp.sys.ibm.pc.hardware		120		8.rec.autos		120
	5.comp.sys.mac.hardware		120		13.sci.electronics		120
	8.rec.autos		120		18.talk.politics.mideast		120
	13.sci.electronics		120		19.talk.politics.misc		120
T5	1.alt.attheism	7474	120	T6	2. comp.graphics	7155	120
	2. comp.graphics		120		3.compcos.ms.windows.misc		120
	4.comp.sys.ibm.pc.hardware		120		4.comp.sys.ibm.pc.hardware		120
	5.comp.sys.mac.hardware		120		5.comp.sys.mac.hardware		120
	8.rec.autos		120		8.rec.autos		120
	9.rec.motorcycles		120		13.sci.electronics		120
	19.talk.politics.misc		120		19.talk.politics.misc		120
	20.talk.religion.misc		120		20.talk.religion.misc		120
T7	1.alt.attheism	7939	120	T8	3.compcos.ms.windows.misc	14388	50
	2. comp.graphics		120		5.comp.sys.mac.hardware		50
	4.comp.sys.ibm.pc.hardware		120		8.rec.autos		50
	5.comp.sys.mac.hardware		120		10.rec.sport.baseball		50
	8.rec.autos		120		14.sci.med		50
	9.rec.motorcycles		120		15.sci.space		50
	10.rec.sport.baseball		120		16.soc.religion.christian		50
	12.sci.crypt		120		17.talk.politics.guns		50
	13.sci.electronics		120		18.talk.politics.mideast		50
	14.sci.med		120		20.talk.religion.misc		50

7.4.3 Overall Performance

We compare the overall performance of our method with J48, SMO and libSVM. The comparison results are shown in Table 7.4. We can see that SDCC outperforms the other three classification algorithms obviously on T1, T2 and T8. For the other five text data sets, SDCC outperforms J48 and SMO vastly and its performance is comparable with libSVM.

Table 7.4 Overall performance comparison of different classification methods.

Classifier	Metric	T1	T2	T3	T4	T5	T6	T7	T8
SDCC	$\mu-F1$	0.9425	0.8863	0.9681	0.9722	0.9677	0.9646	0.9583	0.806
	$M-F1$	0.9462	0.8714	0.9691	0.9725	0.9681	0.9659	0.9602	0.8152
J48	$\mu-F1$	0.7725	0.7726	0.8541	0.8958	0.8114	0.7865	0.7316	0.59
	$M-F1$	0.7829	0.7815	0.8545	0.8963	0.8119	0.7875	0.7321	0.6117
SMO	$\mu-F1$	0.8975	0.8127	0.8569	0.893	0.8656	0.8656	0.8475	0.724
	$M-F1$	0.9007	0.8102	0.8577	0.8942	0.8684	0.8686	0.852	0.7445
LibSVM	$\mu-F1$	0.895	0.8528	0.9861	0.9861	0.9802	0.9802	0.9783	0.716
	$M-F1$	0.8977	0.8394	0.9863	0.9862	0.9805	0.9804	0.9785	0.7629

7.4.4 Performance Details

Table 7.5 to Table 7.12 show the performance details on data set T1 to T8. We can see that SDCC and libSVM outperforms the other classification algorithms on each class of those text data sets. SDCC is better than libSVM for several percent on some data sets on which SDCC work better obviously. At the same time, libSVM is better than SDCC for only one or two percent on the other data sets, on which other algorithms also work well.

Table 7.5 F1 value on data set T1.

class#	SDCC	J48	SMO	libSVM
2	0.904	0.701	0.858	0.884
10	0.964	0.779	0.894	0.88
15	0.931	0.791	0.882	0.878
18	0.974	0.835	0.959	0.942

Table 7.6 F1 value on data set T2.

class#	SDCC	J48	SMO	libSVM
2	0.888	0.767	0.842	0.866
10	0.933	0.82	0.83	0.897
15	0.824	0.693	0.696	0.758
18	0.788	0.8	0.788	0.75

Table 7.7 F1 value on data set T3.

class#	SDCC	J48	SMO	libSVM
2	0.992	0.871	0.805	0.996
3	0.987	0.967	1	1
4	0.943	0.856	0.805	0.967
5	0.983	0.798	0.839	0.987
8	0.983	0.858	0.868	0.996
13	0.921	0.774	0.824	0.971

Table 7.8 F1 value on data set T4.

class#	SDCC	J48	SMO	libSVM
2	0.979	0.887	0.789	0.988
3	0.952	0.928	0.983	0.992
8	0.983	0.855	0.874	0.992
13	0.967	0.841	0.877	0.975
18	0.967	0.947	0.987	0.992
19	0.983	0.916	0.851	0.979

Table 7.9 F1 value on data set T5.

class#	SDCC	J48	SMO	libSVM
1	0.975	0.881	0.916	0.992
2	0.959	0.758	0.759	0.983
4	0.929	0.793	0.873	0.975
5	0.975	0.75	0.864	0.988
8	0.983	0.878	0.891	0.992
9	0.979	0.807	0.88	0.96
19	0.958	0.815	0.907	0.975
20	0.979	0.808	0.846	0.979

Table 7.10 F1 value on data set T6.

class#	SDCC	J48	SMO	libSVM
2	0.979	0.755	0.784	0.988
3	0.987	0.963	0.996	1
4	0.919	0.777	0.814	0.959
5	0.987	0.682	0.858	0.987
8	0.988	0.871	0.897	0.996
13	0.911	0.675	0.843	0.951
19	0.967	0.826	0.903	0.987
20	0.979	0.74	0.841	0.975

Table 7.11 F1 value on data set T7.

class#	SDCC	J48	SMO	libSVM
1	0.979	0.77	0.924	0.987
2	0.979	0.696	0.722	0.983
4	0.91	0.714	0.829	0.962
5	0.975	0.748	0.811	0.987
8	0.967	0.763	0.838	0.992
9	0.996	0.795	0.888	0.775
10	0.943	0.733	0.866	0.955
12	0.947	0.721	0.935	0.987
13	0.89	0.589	0.797	0.958
14	1	0.787	0.892	0.996

Table 7.12 F1 value on data set T8.

class#	SDCC	J48	SMO	libSVM
3	0.846	0.68	0.72	0.744
5	0.707	0.505	0.59	0.674
7	0.97	0.674	0.851	0.8
8	0.841	0.653	0.578	0.503
10	0.81	0.731	0.854	0.876
14	0.844	0.667	0.787	0.765
15	0.712	0.423	0.769	0.777
16	0.925	0.783	0.889	0.901
18	0.66	0.322	0.6	0.612
20	0.766	0.598	0.713	0.733

7.5 Conclusion

In this chapter, we propose a subspace decision cluster classifier (SDCC) based on the subspace decision cluster tree. This new method is designed for classifying text data sets. SDCC adopts subspace clustering algorithm EWKM to build a subspace decision cluster tree from a training data set. A classifier includes the cosine distance metric and a set of subspace decision clusters which are selected from the subspace decision cluster tree. SDCC is efficient in classifying text data sets because of the efficiency of the decision cluster tree framework which integrates subspace clustering into classification, and also the subspace clustering and cosine distance metric for text data mining.

We have presented experimental results on real text data sets to compare the performance of subspace decision cluster classification method with those of other well-known classification methods. The comparison results have shown that SDCC has advantages in classifying large, high dimensional sparse text data with multiple classes.

Chapter 8

Theoretical Analysis of Error Bound

In this chapter, we will analyze why our DCC model (Cluster-based classification) is better than KNN method (Object-based classification). The error bound will also be discussed in this chapter.

8.1 Why Cluster-based Is Better than Object-based

The classification step of our Decision Cluster Classification framework is KNN-like classification. The new object is classified to the class of the nearest cluster in classifier. We call this kind of classification Cluster-based classification, while, we call KNN classification Object-based classification. It has been proved that the error rate of KNN algorithm, $P_{NN}(e)$, is less than double of the error rate of Bayesian algorithm $P^{*}(e)$. That is $P_{NN}(e) < 2P^{*}(e)$ [123].

In our DCC framework, we adopt decision clusters instead of objects adopted by KNN algorithm. The objects in the same decision cluster are close to each other and with similar behavior. Decisions based on similar objects are more convincible than decisions based on independent objects. For a cluster $C^{'}$, the dominant class $\Omega^{'}$ is defined as the most frequent class in $C^{'}$ as follows:

$$\Omega^{'} = \arg\max_{\omega} \sum_{x^{'} \in C^{'}} \mathrm{I}(W(x^{'}) = \omega) \tag{8-1}$$

In Equation (8-1), $W(x^{'})$ is the class label of the object $x^{'}$. If $C^{'}$ is the nearest cluster to a new object x, the object x is labeled by the dominant class $\Omega^{'}$ of cluster

$C^{'}$. If the class label of $x^{'}$ is ω, $\mathrm{I}(W(x^{'}) = \omega)$ equals to 1, else $\mathrm{I}(W(x^{'}) = \omega)$ equals to 0. The definition of $\Omega^{'}$ comes from statistical view.

We suppose there are c classes in the cluster $C^{'}$, $\omega_1, \omega_2, ..., \omega_c$, each of them has its own probability to be the dominant class $\Omega^{'}$. We can show the reliability of a cluster by studying the probabilities $\Pr ob(\Omega^{'} = \omega_i)$. The objects in the same cluster $C^{'}$ have similar behaviors, their posterior probabilities for a class ω_i are approximately equal. This condition satisfies the following equation:

$$\Pr ob(\omega_i \mid x_j^{'}, x_j^{'} \in C^{'}) \doteq \Pr ob(\omega_i \mid x_k^{'}, x_k^{'} \in C^{'}), j \neq k\ , 1 \leq j, k \leq c \tag{8-2}$$

So, the posterior probability for each class in the same cluster can be compared by the numbers of objects they include. There exists a class ω_m ($1 \leq m \leq c$) which has larger posterior probability than other classes. That is

$$\Pr ob(\omega_m \mid x^{'}, x^{'} \in C^{'}) = \max_{1 \leq i \leq c} \Pr ob(\omega_i \mid x^{'}, x^{'} \in C^{'}) \tag{8-3}$$

Theorem 1 If there are enough samples with similar behavior in a cluster $C^{'}$, then there is the probability approximately equal to 1 that ω_m is the dominant class $\Omega^{'}$ of $C^{'}$.

Proof

PP_i is the posterior probability of class ω_i, $PP_i = \Pr ob(\omega_i \mid x^{'}, x^{'} \in C^{'})$. According to the large number law, PP_i can be approximately equal to the frequency of samples in class ω_i. So that, we have the following equation:

$$PP_i = \Pr ob(\omega_i \mid x^{'}, x^{'} \in C^{'}) \doteq \frac{\sum_{x^{'} \in C^{'}} \mathrm{I}(W(x^{'}) = \omega_i)}{|C^{'}|} \tag{8-4}$$

In Equation (8-4), $|C^{'}|$ is the number of objects in cluster $C^{'}$. From Equation (8-3) and (8-4), we get the following equation:

$$PP_m = \max_{1 \le i \le c}\{PP_i\} \doteq \max_{1 \le i \le c}\{\frac{\sum_{x^{'} \in C^{'}} \mathrm{I}(W(x^{'}) = \omega_i)}{|C^{'}|}\} \tag{8-5}$$

From the definition of dominate class $\Omega^{'}$ (Equation (8-1)), we get

$$\Pr ob(\Omega^{'} = \omega_m) \doteq 1 \tag{8-6}$$

Theorem 2 The error rate of Decision Cluster Classification model is lower than that of the KNN algorithm.

Proof

In the nearest neighbor algorithm, if there are enough training samples, then the posterior probability of object x for class ω_i is almost equal to that of the nearest $x^{'}$. Equation (8-7) presents this condition.

$$\Pr ob(\omega_i \mid x) \doteq \Pr ob(\omega_i \mid x^{'}) \tag{8-7}$$

Here, x and $x^{'}$ are independent objects. The posterior probability for the pair $(x, x^{'})$ from class pair (ω_i, ω_j) is Equation (8-8).

$$\Pr ob(\omega_i, \omega_j \mid x, x^{'}) = \Pr ob(\omega_i \mid x) \cdot \Pr ob(\omega_j \mid x^{'}) \tag{8-8}$$

The right-classified probability $\mathrm{P}_{NN}(r \mid x)$ of the object x is the probability that x and $x^{'}$ have the same class label,

$$\mathrm{P}_{NN}(r \mid x) = \sum_{i=1}^{c}\sum_{j=1}^{c} \Pr ob(\omega_i, \omega_j \mid x, x^{'}) \mathrm{I}(\omega_i = \omega_j) \tag{8-9}$$

From Equation (8-7), (8-8) and (8-9), we get Equation (8-10).

$$\mathrm{P}_{NN}(r \mid x) = \sum_{i=1}^{c}\sum_{j=1}^{c} \Pr ob(\omega_i \mid x) \cdot \Pr ob(\omega_j \mid x') \cdot \mathrm{I}(\omega_i = \omega_j) = \sum_{i=1}^{c} \Pr ob^2(\omega_i \mid x) \quad (8\text{-}10)$$

Similarly, we can get the right-classification probability of DCC model for classifying x. The probability of right classification is the probability of the real class labels of objects being equal to the dominant classes of decision clusters. Suppose the nearest cluster to x is the cluster C'. In another point of view, x can be treated as a member of C'. x' is any object in C', and it has similar posterior properties with x. That is

$$\Pr ob(\omega_i \mid x', x' \in C') \doteq \Pr ob(\omega_i \mid x) \quad (8\text{-}11)$$

The right-classification probability $\mathrm{P}_{DCC}(r \mid x)$ of DCC model for classifying x can be presented by the following equation:

$$\mathrm{P}_{DCC}(r \mid x) = \sum_{i=1}^{c}\sum_{j=1}^{c} \Pr ob(\omega_i \mid x) \cdot \Pr ob(\omega_j = \Omega') \cdot \mathrm{I}(\omega_i = \omega_j) \quad (8\text{-}12)$$

From **Theorem 1**, we have

$$\Pr ob(\Omega' = \omega_i) = \begin{cases} 1, \omega_i = \omega_m \\ 0, others \end{cases} \quad (8\text{-}13)$$

Equation (8-12) can be simplified by integrating (8-13) as follows:

$$\mathrm{P}_{DCC}(r \mid x) = \sum_{i=1}^{c} \Pr ob(\omega_i \mid x) \cdot \mathrm{I}(\omega_i = \omega_m) = \Pr ob(\omega_m \mid x) \quad (8\text{-}14)$$

From Equation (8-10), (8-14) and $\Pr ob(\omega_m \mid x) \ge \Pr ob(\omega_i \mid x)$, $\sum_{i=1}^{c} \Pr ob(\omega_i \mid x) = 1$, we have the following expression:

$$\begin{aligned} \mathrm{P}_{NN}(r \mid x) &= \sum_{i=1}^{c} \Pr ob^2(\omega_i \mid x) \\ &\le \sum_{i=1}^{c} \Pr ob(\omega_i \mid x) \cdot \Pr ob(\omega_m \mid x) = \Pr ob(\omega_m \mid x) = \mathrm{P}_{DCC}(r \mid x) \end{aligned} \quad (8\text{-}15)$$

From Equation (8-15), the right-classification probability for any point x in the space R^d of the nearest neighbor algorithm is less than or equal to that of the DCC model. At the same time, the error rate $\mathrm{P}_{DCC}(e)$ of all objects in the space R^d of DCC model is smaller than the error rate $\mathrm{P}_{NN}(e)$ of the nearest neighbor algorithm as show in the following expression:

$$\mathrm{P}_{DCC}(e) = 1 - \sum_{x \in R^d} \mathrm{P}_{DCC}(r \mid x) \le 1 - \sum_{x \in R^d} \mathrm{P}_{NN}(r \mid x) = \mathrm{P}_{NN}(e) \tag{8-16}$$

8.2 Error Bound

Suppose the decision clusters in classification model are credible enough to reflect the space partition. The priori probability of the sample x falling into one of the k decision clusters of classifier can be estimated.

$$p_i = \Pr ob(x \in P_i) = \frac{N_i}{N} \tag{8-17}$$

In Equation (8-17), P_i is the ith decision cluster in classifier, N_i is the number of objects of P_i and N is the number of total objects in all decision clusters of the classifier.

The probability of right classification is the probability of the real class labels of objects being equal to the dominant classes of decision clusters. The expectation of being classified correctly by classifier can be expressed by the expectation of being classified correctly by decision cluster as demonstrated in Equation (8-18).

$$E(I(l(x) = \omega)) = \sum p_i E(I(l_i(x) = \omega)) \tag{8-18}$$

We use $Pur(P_i)$ to denote the purity of the ith decision cluster P_i. In general, the bigger of N_i the more reliable of the cluster P_i. We adopt a reliability function $R(n)$ with parameter n to discriminate different decision clusters. $R(n)$ is a monotone increasing function in the scale of $[0,1]$. $R(N_i)$ is the reliability of the ith decision cluster P_i. Because the probability of right classification of a decision cluster cannot be bigger than the purity of this decision cluster and the value of $R(N_i)$ cannot be bigger than 1, we have the following equation:

$$E(I(l_i(x)=\omega)) \geq Pur(P_i)R(N_i) \tag{8-19}$$

From Equation (8-18) and (8-19), we get the following equation:

$$E(I(l(x)=\omega)) \geq Pur(P_i)R(N_i) \tag{8-20}$$

$$E(I(l(x)=\omega)) \geq \sum p_i Pur(P_i)R(N_i) \tag{8-21}$$

We denote $Pur_{\min} = \min\{Pur(P_i)\}$, then we have Equation (8-22). Here, k is the number of decision clusters in classifier.

$$E(I(l(x)=\omega)) \geq \sum_{i=1}^{k} p_i \cdot Pur_{\min} \cdot R(N_i) = \frac{Pur_{\min}}{N}\sum_{i=1}^{k} N_i \cdot R(N_i) \tag{8-22}$$

Let $f(n) = n \cdot R(n)$, we have $f^{'}(n) = R(n) + n \cdot R^{'}(n)$, and $f^{''}(n) = 2R^{'}(n) + n \cdot R^{''}(n)$. $f^{''}(0) = 2R^{'}(0) + 0 \cdot R^{''}(0) = 2R^{'}(0) > 0$. If there is a point N_0 which makes $f^{''}(N_0) = 0$, then $f(n)$ is a convex function in $(0, N_0)$. We have already known that $R(n) > R(N_0)$ in (N_0, ∞). From function (8-21) we have the following function:

$$E(I(l(x)=\omega)) \geq \frac{Pur_{\min}}{N}\left(\sum_{N_i<N_0} f(N_i) + \sum_{N_i \geq N_0} f(N_i)\right) \tag{8-23}$$

Suppose there are k_0 clusters whose sizes (the number of objects) are smaller than N_0. N_{k_0} is the sum of the number of objects of the k_0 clusters. The sizes of the remain $k-k_0$ clusters are bigger than N_0. According to the characteristics of the convex function, we have the following function (8-24) and (8-25),

$$\sum_{N_i<N_0} f(N_i) \geq k_0 \cdot f(\frac{\sum_{N_i<N_0} N_i}{k_0}) = k_0 \cdot f(\frac{N_{k_0}}{k_0}) \tag{8-24}$$

$$\sum_{N_i \geq N_0} f(N_i) \geq (k-k_0) \cdot f(\frac{\sum_{N_i \geq N_0} N_i}{k-k_0}) = (N-N_{k_0}) \cdot R(\frac{N-N_{k_0}}{k-k_0}) \tag{8-25}$$

From Expression (8-23), (8-24) and (8-25) we can get the following

$$E(I(l(x)=\omega)) \geq \frac{Pur_{\min}}{N}(N_{k_0} \cdot R(\frac{N_{k_0}}{k_0}) + (N-N_{k_0})R(\frac{N-N_{k_0}}{k-k_0})) \tag{8-26}$$

The error bound can be estimated now:

$$\begin{aligned} Error &= 1 - E(I(l(x)=\omega)) \\ &\leq 1 - (\frac{Pur_{\min}}{N}(N_{k_0} \cdot R(\frac{N_{k_0}}{k_0}) + (N-N_{k_0})R(\frac{N-N_{k_0}}{k-k_0}))) \end{aligned} \tag{8-27}$$

$$Error \leq 1 - Pur_{\min} R(\frac{N-N_{k_0}}{k-k_0}) + \frac{Pur_{\min} N_k}{N}(R(\frac{N-N_{k_0}}{k-k_0}) - R(\frac{N_{k_0}}{k_0})) \tag{8-28}$$

$$\begin{aligned} Error &\leq 1 - Pur_{\min} R(\frac{N-N_{k_0}}{k-k_0}) + \frac{Pur_{\min} N_k}{N} R(\frac{N-N_{k_0}}{k-k_0}) \\ &= 1 - Pur_{\min}(1 - \frac{N_{k_0}}{N}) R(\frac{N-N_{k_0}}{k-k_0}) \end{aligned} \tag{8-29}$$

We simplify the $R(\frac{N-N_{k_0}}{k-k_0})$ as $R(x_0)$, we can get the final conclusion that

$$Error \leq 1 - Pur_{\min}(1 - \frac{N_{k_0}}{N}) R(x_0) \tag{8-30}$$

We can see that, the precision of DCC model partially depends on the size and purity of decision clusters of classifier. The bigger and more pure decision clusters lead to getting lower error rate.

Chapter 9

Conclusion

We have presented a new classification framework which integrates clustering into classification work. Under this framework, different classifiers are proposed with different characteristics. Main distributions are listed in Section 9.1 and future work is given in Section 9.2.

9.1 Main Contributions

- A novel Cluster-based classification framework which adopts clustering algorithm to solve classification problem is presented. This framework considers clustering and classification together. Under this Cluster-based classification framework, a decision cluster tree or forest is built from partitioning the training data set recursively by calling a clustering algorithm. Then, a classification model is specified from the decision cluster tree or forest. Finally, new objects are classified by this classification model. In a word, this classification framework includes three steps: tree (forest) construction, model selection and classification.
- An Automatic Decision Cluster Classification (ADCC) method is proposed, in which, the weighted k-means (W-k-means) clustering algorithm is used to build a Cluster-based classification model automatically. W-k-means is efficient for large data sets and it can reduce the influence of noisy attributes by assigning them smaller weights. The decision cluster tree is built by executing W-k-means clustering algorithm recursively. In the tree growing process, we use various tests including Anderson-Darling test to determine whether a node can be further

- A Decision Cluster Forest Classification (DCFC) method is developed to deal with weak decision cluster problem and multiple classes problem. This method builds a set of decision cluster trees from subsets of the training data set instead of building a single decision cluster tree. These decision cluster trees form a decision cluster forest. Each tree in the forest is built from the subset of objects in the same class. The decision clusters in the same tree have the same dominant class. In this way, no weak decision cluster in which no single class dominates is created in such decision cluster tree. The decision cluster forest method has advantages of classifying data with multiple classes because the DCFC model is guaranteed to contain decision clusters in all classes. DCFC model is a more intuitive and direct multi-class classification method.

- Text data is a typical high dimensional sparse data. Subspace clustering algorithms are efficient for this kind of high dimensional sparse data. A Subspace Decision Cluster Classification (SDCC) method is designed for text classification. In this work, a subspace decision cluster tree is generated from the training data set by recursively calling a subspace clustering algorithm Entropy Weighting k-Means algorithm. In every partition, the sub-clusters are found in their own subspaces instead of the entire dimension space. After building the subspace decision cluster tree, we generate a classifier form the tree. In the classification step, we use the cosine distance function which is often used in text mining to compute the distances between new objects and the subspace decision clusters.

- A set of decision clusters which are selected from decision cluster tree or forest plus a specific distance metric construct a classification model. In our work, we select leaf nodes to construct the classifier. Euclidean distance function and

Cosine distance function are both integrated into our Cluster-based classification framework. A KNN-like classification step is implemented when classifying new objects.

- Another model selection, named Crotch Ensemble, is introduced. Instead of considering a set of decision clusters, this model considers Crotch Predictors which are inner nodes with their direct children. When classifying a new object, a subset of crotch predictors is selected according to the distances between the object and the crotch predictors. A classification is made on the object as the class predicted by the crotch predictors with the maximum accumulative weights. If a new object is misclassified by one crotch predictor, the misclassification can be corrected by other crotch predictors. Multiple crotch predictor decisions are more robust than the single decision cluster decision.
- An experimental scheme is designed to demonstrate the performance of this series of classification methods under the Decision Cluster Tree framework. We have demonstrated the efficiency and effectiveness of our new methods. We also conducted experiments to compare our new methods with other classification methods.

9.2 Future Work

We now discuss a few directions along which we plan to continue our work.

- More complex model selection methods. In this thesis, the model selection methods are a little simple. We would like to design some other model selection methods. We plan to study more model selection methods and to extract new model selection methods for our Cluster-based classification framework.
- More experiments to compare the DCFC model with the ECOC multi-class classification method. The error-correcting output codes method (ECOC) was

designed to solve multi-class learning problem by learning multiple binary classification models and matching the classification results with the designed codeword to correct misclassifications. We plan to compare our DCFC method with ECOC on multi-class classification problem specially.

- Other subspace algorithms to be implemented in our framework. There are a lot of subspace clustering algorithms in the literature. Besides EWKM, we intend to investigate other subspace clustering algorithms according to their own characteristics.
- Further use of branch information in classification step. In this thesis, we have not made good use of the information of branches in the decision cluster tree. We plan to make use of the information of branches further.
- Further theoretical analysis. We would like to give a complete theoretical system for our Cluster-based classification framework. We plan to analyze why our Cluster-based classification framework integrates clustering methods into classification problem from theoretical view.
- More research work on dealing with other kinds of data sets. We would like to handle other cluster shapes except spherical cluster shape.

Bibliography

[1] J.Z. Huang, M.K. Ng, T. Lin and D. Cheung, “An Interactive Approach to Building Classification Models by Clustering and Cluster Validation,” *IDEAL 2000*, LNCS 1983, pp. 23-28, 2000.

[2] J.Z. Huang, M.K. Ng, H.Q. Rong and Z.C. Li, “Automated Variable Weighting in k-Means Type Clustering,” *IEEE Transactions on Pattern Analysis*. Mach. Intell., Vol. 27, no. 5, pp. 657-668, 2005.

[3] L. Jing, M. Ng and J. Huang, “An Entropy Weighting k-means Algorithm for Subspace Clustering of High-dimensional Sparse Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol.19, pp. 1026-1041, 2007.

[4] S.K. Murthy, “Automatic construction of decision trees from data: a multidisciplinary survey,” *Data Mining Knowl. Disc.*, vol. 2, no. 4, pp. 345–389, 1998.

[5] H.J. Zeng, X.H. Wang, Z. Chen and W.Y. Ma, “CBC: Clustering Based Text Classification Requiring Minimal Labeled Data,” *Proc. of the Third IEEE International Conference on Data Mining (ICDM’03)*, pp. 443-450, 2003.

[6] T.W. Anderson and D.A. Darling, “Asymptotic theory of certain "goodness-of-fit" criteria based on stochastic processes,” *Annals of Mathematical Statistics* vol. 23, pp. 193-212, 1952.

[7] M.A. Stephens, “EDF Statistics for Goodness of Fit and Some Comparisons,” *Journal of the American Statistical Association*, vol. 69, pp. 730-737, 1974.

[8] A.K. Jain, M.N. Murty and P.L. Flynn, “Data Clustering: A Review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[9] M. Steinbach, L. Ertoz and V. Kumar, “The Challenges of Clustering High Dimensional Data,” http://www-users.cs.umn.edu/~ertoz/papers/clustering_chapter.pdf, 2003.

[10] D. Cai, X. He and J. Han, “Document Clustering Using Locality Preserving Indexing,” *IEEE Trans. Knowledge and Data Eng.*,vol. 17, no. 12, Dec. 2005.

[11] C.C. Aggarwal and P.S. Yu, “Finding Generalized Projected Clusters in High Dimensional Spaces,” *Proc. ACM SIGMOD Int’l Conf. Management of Data*, pp. 70-81, 2000.

[12] C.M. Procopiuc, M. Jones, P.K. Agarwal and T.M. Murali, “A Monte Carlo Algorithm for Fast Projective Clustering,” *Proc. ACM SIGMOD Int’l Conf. Management of Data*, pp. 418-427, 2002.

[13] C. Domeniconi, D. Papadopoulos, D. Gunopulos and S. Ma, “Subspace Clustering of High Dimensional Data,” *Proc. SIAM Int’l Conf. Data Mining*, pp. 246-257, 2004.

[14] J.H. Friedman and J.J. Meulman, “Clustering Objects on Subsets of Attributes,” *J. Royal Statistical Soc. B*, vol. 66, no. 4, pp. 815-849, 2004.

[15] N. Ye and X.Y. Li, “A Machine Learning Algorithm Based on Supervised Clustering and Classification,” *AMT 2001*, LNCS 2252, pp. 327–334, 2001.

[16] L. Jing, M.K. Ng, J. Xu and J.Z. Huang, “Subspace Clustering of Text Documents with Feature Weighting k-Means Algorithm,” *Proc. of the Ninth Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp. 802-812, 2005.

[17] L. Parsons, E. Haque and H. Liu, “Subspace Clustering for High Dimensional Data: A Review,” *SIGKDD Explorations*, vol. 6, no. 1, pp. 90-105, 2004.

[18] Z. Huang, “Extensions to the k-Means Algorithms for Clustering Large Data Sets with Categorical Values,” *Data Ming and Knowledge Discovery*, vol. 2, no. 3, pp. 283-304, 1998.

[19] J. Mui and K. Fu, “Automated Classification of Nucleated Blood Cells Using a Binary Tree Classifier,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 5, pp. 429-443, 1980.

[20] Y.M. Ye, J. Huang, X.J. Chen, S.G. Zhou, G. Williams and X.F. Xu, “Neighborhood Density Method for Selecting Initial Cluster Centers in K-Means Clustering,” *PAKDD*, pp. 189-198, 2006.

[21] G. Piatetsky-Shapiro, C. Djeraba, L. Getoor, R. Grossman, R. Feldman and M. Zaki, “What Are The Grand Challenges for Data Mining,” *PAKDD-2006 Panel Report*, *SIGKDD Explorations*, vol. 8, no.2, pp. 70-77, 2006.

[22] A. Kyriakopoulou and T. Kalamboukis, “Text Classification Using Clustering,” *ECML-PKDD Discovery Challenge Workshop Proceedings*, 2006.

[23] C. Blake and C. Merz, UCI Repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA, 1998[http://www.ics.uci.edu/~mlearn/MLRepository.html].

[25] J. Han and M. Kamber: Data Mining Concepts and Techniques, Second Edition. 2006.

[26] C.J. van Rijsbergen, Information Retireval, 2nd ed., London, Butterworths, London, 1979.

[27] Quinlan, John Ross, C4.5: Programs for Machine Learning. Morgan Kaufmann Pub., San Mateo, CA,1993.

[28] K.C. You and King-Sun Fu, “An approach to the design of a linear binary tree classifier,” *Proc. of the Third Symposium on Machine Processing of Remotely Sensed Data*, West Lafayette, IN, 1976. Purdue Univ.

[29] Robust linear discriminant trees. In (AI&Stats-95), pp. 285–291.

[30] Breiman, Leo, Jerome Friedman, Richard Olshen, and Charles Stone, “Classification and Regression Trees,” *Wadsworth Int. Group*, 1984.

[31] S. Murthy, S. Kasif and S. Steven, “A system for induction of oblique decision trees,” *J. of Artificial Intelligence Research*, vol. 2, pp. 1–33, August 1994.

[32] M. Golea, and M. Marchand, “A growth algorithm for neural network decision trees,” *EuroPhysics Letters*, vol. 12, no. 3, pp. 205–210, June 1990.

[33] J.A. Sirat and J.-P. Nadal, “Neural trees: A new tool for classification,” *Network: Computation in Neural Systems*, vol. 1, no. 4, pp. 423–438, October 1990.

[34] T.M. Cover and P.E. Hart, “Nearest Neighbor Pattern Classification,” *IEEE Trans. Information Theory*, vol. 13, no. 1, pp. 21-27, 1967.

[35] B. Zhang, S. N. Srihari, “Fast k-Nearest Neighbor Classification Using Cluster-Based Trees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 4, April 2004.

[36] F. Angiulli, “Fast Nearest Neighbor Condensation for Large Data Sets Classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1450-1464, 2007.

[37] F. Angiulli, “Fast Condensed Nearest Neighbor Rule,” *Proc. of the 22nd Int’l Conf. Machine Learning (ICML ’05)*, pp. 25-32, 2005.

[38] G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour, “An Algorithm for a Selective Nearest Neighbor Decision Rule,” *IEEE Trans. Information Theory*, vol. 21, pp. 665-669, Nov. 1975.

[39] C.L. Chang, “Finding Prototypes for Nearest Neighbor Decision Rule,” *IEEE Trans. Computers*, vol. 23, no. 11, pp. 1179-1184, Nov. 1974.

[40] P.E. Hart, “Condensed Nearest Neighbor Rule,” *IEEE Trans. Information Theory*, vol. 14, pp. 515-516, May 1968.

[41] D.W. Jacobs and D. Weinshall, “Classification with Non-Metric Distances: Image Retrieval and Class Representation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 583-600, June 2000.

[42] A.J. Broder, “Strategies for Efficient Incremental Nearest Neighbor Search,” *Pattern Recognition*, vol. 23, nos. 1/2, pp. 171-178, Nov. 1986.

[43] A. Farago, T. Linder and G. Lugosi, “Fast Nearest-Neighbor Search in Dissimilarity Spaces,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 957-962, Sept. 1993.

[44] J.H. Friedman, F. Baskett and L.J. Shustek, “An Algorithm for Finding Nearest Neighbors,” *IEEE Trans. Computers*, vol. 24, no. 10, pp. 1000-1006, Oct. 1975.

[45] K. Fukunaga and P.M. Narendra, “A Branch and Bound Algorithm for Computing k-Nearest Neighbors,” *IEEE Trans. Computers*, vol. 24, no. 7, pp. 750-753, July 1975.

[46] G. Toussaint, “Proximity Graphs for Nearest Neighbor Decision Rules: Recent Progress,” *Proc. of the 34th Symp. Interface of Computing Science and Statistics (Interface ’02)*, Apr. 2002.

[47] D.R. Wilson and T.R. Martinez, “Reduction Techniques for Instance-Based Learning Algorithms,” *Machine Learning*, vol. 38, no. 3, pp. 257-286, 2000.

[48] R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, "The Second PASCAL Recognising Textual Entailment Challenge," *Proc. of the Second PASCAL ChallengesWorkshop on Recognising Textual Entailment*, Venezia, Italy, 2006.

[49] I. Dagan, O. Glickman and B. Magnini, "The PASCAL Recognising Textual Entailment Challenge," *Proc. of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, UK, pp. 1–8, 2005.

[50] J. Herrera, A. Penas and F. Verdejo, "Question Answering Pilot Task at CLEF 2004," *Working Notes of the CLEF 2004 Workshop*, Bath, United Kingdom, 2004.

[51] R.S. Zemel and G.E. Hinton, "Learning population codes by minimum description length," *Neural Computation,* vol. 7, no. 3, pp. 549–564, 1995.

[52] G.E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Advances in Neural Information Processing Systems* 6, 1994.

[53] Hiroshi Tenmoto, Mineichi Kudo and Masaru Shimbo, "MDL-Based Selection of the Number of Components in Mixture Models for Pattern Classification," *SSPR/SPR*, pp. 831-836, 1998.

[54] H. Bischof, A. Leonardis, and A. Selb, "MDL principle for robust vector quantization," *Pattern Analysis and Application*, vol. 2, no. 1, pp. 59-72, 1999.

[55] Petri Kontkanen, Petri Myllymaki, Wray Buntine, Jorma Rissanen and Henry Tirri, "An MDL framework for data clustering," *Advances in Minimum Description Length: Theory and Applications*, P. Grunwald, I.J. Myung, and M. Pitt, Eds., pp. 323 – 353. MIT Press, Cambridge, 2005.

[56] Petri Kontkanen, Hannes Wettig, and Petri Myllymäki, "NML Computation Algorithms for Tree-Structured Multinomial Bayesian Networks," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, Article ID 90947, 11 pages, 2007. doi:10.1155/2007/90947.

[57] J.I. Myung, D.J.Navarro, Mark A. Pitt, "Model selection by normalized maximum likelihood," *Journal of Mathematical Psychology,* vol. 50, pp. 167–179, 2006.

[58] Anne Patrikainen and Marina Meila, "Comparing Subspace Clusterings," IEEE Trans. Knowledge and Data Engineering, Vol. 18, No. 7, July 2006.

[59] Arindam Banerjee and Inderjit Dhillon, "A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation," *Journal of Machine Learning Research*, vol. 8, 2007.

[60] Z. Huang and T. Lin, "A Visual Method of Cluster Validation with Fastmap," *PAKDD 2000*, LNAI 1805, pp. 153-164, 2000.

[61] S.R. Safavian and D. Landgrebe, "A survey of Decision Tree Classifier Methodology," *IEEE Transactions ON Neural Networks*, Vol. 2, pp. 285-293, 1991.

[62] M.J. Kearns and Y. Mansour, "A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization," *ICML*, pp. 269-277, 1998.

[63] A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123-129, 1972.

[64] B. Gao, T. Liu, X. Zheng, Q. Cheng and W. Ma, "Consistent bipartite graph co-partitioning for starstructured high-order heterogeneous data co-clustering," *Proc. of the 11th International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 41–50, 2005.

[65] H. Cho, I.S. Dhillon, Y. Guan and S. Sra, "Minimum sum-squared residue co-clustering of gene expression data," *Proc. of the 4th SIAM International Conference on Data Mining (SDM)*, pp. 114–125, 2004.

[66] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," *Proc. of the IEEE Conference on Data Mining*, pp. 625–628, 2005.

[67] J. Rissanen, "Modeling by the shortest data description," Automatica 14, pp. 465-471, 1978.

[68] J. Rissanen, "Fisher information and stochastic complexity," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 40-47, 1996.

[69] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.

[70] J. D. M. Rennie, L. Shih, J. Teevan and D. R. Karger, “Tackling the poor assumptions of naive Bayes text classifiers,” *Proc. of the Twentieth International Conference on Machine Learning*, pp. 616－623, 2003.

[71] I. Makoto and T. Takenobu, “Cluster-Based text categorization: A comparison of category search strategies,” *Proc. of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 273–280, 1995.

[72] A. Mccallum and K. Nigam, “A comparison of event models for naive Bayes text classification,” *Proc. of the AAAI-98 Workshop on Learning for Text Categorization*, pp. 41–48, 1998.

[73] B. Masand, G. Lino and D. Waltz, “Classifying news stories using memory based reasoning,” *Proc. of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 59–65, 1992.

[74] Y. M. Yang, “Expert network: Effective and efficient learning from human decisions in text categorization and retrieval,” *Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 13–22, 1994.

[75] Y. M. Yang and X. Liu, “A re-examination of text categorization methods,” *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–49, 1999.

[76] C. Buckley, G. Salton and J. Allan, “The effect of adding relevance information in a relevance feedback environment,” Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 292–300, 1994.

[77] T. Joachims, “A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization,” *Proc. of the 14th International Conference on Machine Learning*, pp. 143–151, 1997.

[78] H. Guo and S.B. Gelfand, “Classification trees with neural network feature extraction,” *IEEE Trans. Neural Netw*, Nol. 3, no. 6, pp. 923–933, 1992.

[79] J.M. Liu, and T. S. Chua, “Building semantic perceptron net for topic spotting,”

Proc. of the 37th Meeting of the Association of Computational Linguistics, pp. 370–377, 2001.

[80] H.T. Ng, W.B. Goh and K.L. Low, “Feature selection, perceptron learning, and a usability case study for text categorization,” *Proc. of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 67–73, 1997.

[81] M.E. Ruiz and P. Srinivasan, “Hierarchical neural networks for text categorization,” *Proc. of the 22nd Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, pp. 81–82, 1999.

[82] H. Schutze, D.A. Hull and J.O. Pedersen, “A comparison of classifier and document representations for the routing problem,” *Proc. of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 229–237, 1995.

[83] C. Cortes and V. Vapnik, “Support vector networks,” *Machine Learning*, Vol. 20, pp. 273–297, 1995.

[84] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” *Proc. of the 10th European Conference on Machine Learning*, pp. 137–142, 1998.

[85] T. Joachims, “Learning to Classify Text Using Support Vector Machines,” *Kluwer Academic*, Hingharn,MA, 2002.

[86] R. Schapire and Y. Singer, “BoosTexter: A boosting-based system for text categorization,” *Machine Learning*, vol. 39, no. 2–3, pp. 135–168, 2000.

[87] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, “Classification and Regression Trees,” Wadsworth, Belmont, CA, 1984.

[88] C.E. Brodley and P.E. Utgoff, “Multivariate decision trees,” *Machine Learning*, vol. 19, no. 1, pp. 45–77, 1995.

[89] J. Quinlan, “C4.5: Programming for Machine Learning,” Morgan Kaumann, SanFransisco, CA, 1993.

[90] P.E. Utgoff and C.E. Brodley, “Linear machine decision trees,” COINS Tech. Rep. 91–10, Dept. of Computer Science, University of Massachusetts, 1991.

[91] L. Denoyer, H. Zaragoza and P. Gallinari, “HMM-Based passage models for document classification and ranking,” *Proc. of the 23rd European Colloquium on Information Retrieval Research*, pp. 126–135, 2001.

[92] D.R.H. Miller, T. Leek and R.M. Schwartz, “A hidden Markov model information retrieval system,” *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 214–221, 1999.

[93] H. Guan, J.Y. Zhou and M.Y. Guo, “A Class-Feature-Centroid Classifier for Text Categorization,” *Proc. 18th International Word Wide Web Conference (WWW2009)*, 2009.

[94] P. Frasconi, G. Soda and A. Vullo, “Text categorization for multi-page documents: a hybrid naïve Bayes HMM approach,” *Proc. of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pp. 11-20, ACM Press New York, NY, USA, 2001.

[95] A.M. Kibriya, E. Frank, B. Pfahringer and G. Holmes, “Multinomial naïve bayes for text categorization revisited,” *Advances in Artificial Intelligence (AI 2004)*, pp. 488-499,2004.

[96] G.D. Guo, H. Wang, D. Bell, Y.X, Bi and K. Greer, “Using kNN model for automatic text categorization,” *Soft Computing*, vol. 10, no. 5, pp. 423-430, 2006.

[97] R.N. Chau, C.S. Yeh and K.A. Smith, “A neural network model for hierarchical multilingual text categorization,” *Advances in Neural Networks*, LNCS, pp. 238-245, 2005.

[98] S. Gao, W. Wu, G.H. Lee and T.S. Chua, “A maximal figure-of-merit (MFoM)-learning approach to robust classifier design for text categorization,” *ACM Transactions on Information Systems*, vol. 42, no. 2, pp. 190-218, 2006.

[99] D. Lewis and J. Catlett, “Heterogeneous uncertainty sampling for supervised learning,” *Proc. of the Eleventh International Conference on Machine Learning*, pp. 148-156, 1994.

[100] S. Weiss, C. Apte, F. Damerau, D. Johnson, F. Oles, T. Goetz and T. Hampp, “Maximizing text-mining performance,” *IEEE Intelligent Systems*, pp. 63-69,

1999.

[101] S. Dumais and H. Chen, "Hierarchical classification of Web content," *Proc. of the 23rd Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 256-263, 2000.

[102] R. Klinkenberg and T. Joachims, "Detecting Concept Drift with Support Vector Machines," *Proc. of the 7th International Conference on Machine Learning*, pp. 487-494, 2000.

[103] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski and T. Zhang, "Robust classification of rare queries using web knowledge," *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 231-238, 2007.

[104] Z. Cataltepe and E. Aygun, "An improvement of centroid-based classification algorithm for text classification," *IEEE 23rd International Conference on Data Engineering Workshop*, pp. 952-956, 2007.

[105] V. Lertnattee and T. Theeramunkong, "Class normalization in centroid-based text categorization," *Information Sciences*, vol. 176, no. 12, pp. 1712-1738, 2006.

[106] S. Tan, "An improved centroid classifier for text categorization," *Expert Systems with Applications*, vol.35, no. 1-2, pp. 279-285, 2008.

[107] V. Tam, A. Santoso and R. Setiono, "A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization," *16th International Conference on Pattern Recognition*, pp. 235-238, 2002.

[108] T. Joachims, "Transductive Inference for Text Classification using Support Vector Machines," *Proc. of the International Conference on Machine Learning (ICML)*, pp. 200-209, 1999.

[109] Yan Li, Edward Hung, Korris Chung, Joshua Huang, "Building A Decision Cluster Classification Model for High Dimensional Data by A Variable Weighting k-Means Method," *Proc. of the Twenty-First Australasian Joint Conference on Artificial Intelligence*, pp. 337-347, 2008.

[110] J.R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, pp. 81-106, 1986.

[111] R. Rastogi and K. Shim, “Public: Adecision tree classifier that integrates building and pruning,” *Proc. of the 24th International Conference on Very Large Data Bases*, pp. 404-415, 1998.

[112] M. Metha, R. Agrawal and J. Riassnen, “SLIQ: A fast scalable classifier for data mining,” *Extending Database Technology*, pp. 18-32, 1996.

[113] P.S. Gregory, D. Chabane, G. Lise, G. Robert, F. Ronen and Z. Mohammed, “Grand challenges spur grand results- Private groups are offering big cash prizes to anyone who can solve a range of daunting problems,” *The Christian Science Monitor,* *www.csmonitor.com/2006/0112/p13s01-stss.html*, 2006.

[114] H. Kim and G. J. Koehler, “An investigation on the conditions of pruning and induced decision tree,” *European Journal of Operation Research*, vol.77, pp. 82-95, 1994.

[115] J. Mingers, “An empirical comparision of pruning methods for decision tree induction,” *Machine Learning*, vol. 2, pp. 227-243, 1989.

[116] J. Furnkranz, “Pruning algorithms for rule learning,” *Machine Learning*, vol. 27, pp. 139-172, 1997.

[117] G. Martinez-Munoz, D.Hernandez-Lobato and A. Suarez, “An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation,” *IEEE Transactions on Pattern Analysis and Machine Inteligence*, vol. 31, no. 2, pp. 245-258, 2009.

[118] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[119] M.G. Voronoi. Nouvelles application des paramtres continus `a la th´eorie des formes quadratiques, deuxi`eme m´emoire, recherche sur le parall ´eloedres primitives. *Journal f¨ur die reine und angewandte Mathematik*, vol. 134, pp. 198- 207, 1908.

[120] L.P. Jing, J. Huang, M. Ng and H. Q. Rong, “A feature weighting approach to building classification models by interactive clustering,” *Modeling decisions for*

artificial intelligence, Springer-Verlag, pp. 284-294, 2004.

[121] T. Dietterich and G. Bakiri, “Solving multiclass learning problems via error-correction output codes,” *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286, 1995.

[122] H. Blockeel, L. Raedt and J. Ramong, “Top-down induction of clustering trees,” *Proc. of the 15th International Conference on Machine Learning*, pp. 55-63, 1998.

[123] Richard O. Duda, Peter E. Hart and David G. Stork, *Pattern classification, Second Edition*, pp. 179-182, 2004.

[124] J. Platt, “ Sequential minimal optimization: A fast algorithm for training support vector machines,” *Microsoft Research,Tech.Rep.:MSR-TR-98-14*, 1998.

[125] S.S. Shapiro and R.S. Francia, “An Approximate Analysis of Variance Test for Normality,” *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 215-216, 1972.

[126] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann and Ian H. Witten (2009), “The WEKA Data Mining Software: An Update,” SGKDD Explorations, vol. 11, no. 1. [http://www.cs.waikato.ac.nz/~ml/weka/].

[127] Aik Choon Tan, Daniel Q. Naiman, Lei Xu, Raimond L. Winslow and Donald Geman, “K-TSP Program Download Page,” 2005[https://jshare.johnshopkins.edu].

[128] X. Ni, G. Xue, X. Ling, Y. Yu, and Q. Yang, “Exploring in the weblog space by detecting informative and affective articles,” In: WWW, Branff, Canada. (2007)

[129] G. Xue, D. Xing, Q. Yang, and Y. Yu, “Deep classification in large-scale text hierarchies,” *Proc. of the 31st Annual International ACM SIGIR Conference*, pp. 627-634, 2008.

[130] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, “When is "nearest neighbor" meaningful?” *Database Theory-ICDT '99, LNCS 1999*, pp.217-235, 1999.

[131] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, “Automatic subspace

clustering of high dimensional data for data mining applications," *SIGMOD Record ACM Special Interest Group on Management of Data*, pp. 94-105, 1998.

[132] Yasser EL-Manzalawy and Vasant Honavar, "LSVM: Integrating LibSVM into Weka Environment," 2005 [http://www.cs.iastate.edu/~yasser/wlsvm].

[133] L. Kaufman and P.J. Rousseeuw, "Finding groups in data: an introduction to cluster analysis," New York: John Wiley & Sons, 1990.

[134] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. of the 5th Berkeley Symp. Math. Statist*, pp. 281-297, 1967.

[135] R. Ng and J. Han, "Efficient and effective clustering method for spatial data mining," *Proc. of the 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pp. 144-155, 1994.

[136] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *Proc. of the 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pp. 103-114, 1996.

[137] S. Guha, R. Rastogi and K. Shim, "Roch: A robust clustering algorithm for categorical attributes," *Proc. of the 1999 Int. Conf. Data Engineering (ICDE'99)*, pp. 512-521, 1999.

[138] G. Karypis, E.H. Han and V. Kumar, "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," *COMPUTER*, pp. 68-75, 1999.

[139] S. Guha, R. Rastogi and K. Shim, "Cure: An efficient clustering algorithm for large databases," *Proc. of the ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pp. 73-84, 1998.

[140] M. Ester, H.P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases," *Proc. of the Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pp. 226-231, 1996.

[141] M. Ankerst, M. Breunig, H.P. Kriegel and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *Proc. of the ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pp. 49-60, 1999.

[142] A. Hinneburg and D.A. Keim, "An efficient approach to clustering in large multimedia databases with noise," *Proc. of the Int. Conf. Knowledge Discovery*

and Data Mining (KDD'98), pp. 58-65, 1998.

[143] W. Wang, J. Yang and R. Muntz, "STING: A statistical information grid approach to special data mining," *Proc. of the Int. Conf. Very Large Data Bases (VLDB'97)*, pp. 186-195, 1997.

[144] G. Sheikholeslami, S. Chatterjee and A. Zhang, "WaveCluster: A multi-resolution clustering approach for very large spatial databases," *Proc. of the Int. Conf. Very Large Data Bases (VLDB'98)*, pp. 428-439, 1998.

[145] D. Fisher, "Improving inference through conceptual clustering," *Proc. of the AAAI Conf.*, pp. 461-465, 1987.

[146] C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu and J. Park, "A framework for finding projected clusters in high dimensional spaces," *Proc. of the ACM SIGMOD Int. Conf. Management of Data*, pp. 193-199, 1999.

[147] K.Y. Yip, D.W. Cheung and M.K. Ng, "HARP: a practical projected clustering algorithm," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 11, pp. 1387-1397, 2004.

[148] Vladimir Vapnikr: The Nature of Statistical Learning Theory, Springer-Verlag, 1995.

[149] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," *the 5th Annual ACM Workshop on COLT*, pp. 144-152, 1992.

[150] Dejan Gorgevik and Dusan Cakmakov, "Combining SVM Classifiers for Handwritten Digit Recognition," *Proc. of the 16th International Conference on Pattern Recognition*, pp. 102-105, 2002.

[151] B. Heisele and P. Ho, T. Poggio, "Face Recognition with Support Vector Machines: Global versus Component-based Approach," *Proc. of the 8th IEEE International Conference on Computer Visio*n, pp. 688-694, 2001.

[152] Y. D. Cai, X. J. Liu, X. B. Xu and G. P. Zhou, "Support Vector Machines for predicting protein structural class," BMC Bioinformatics, vol. 2, no. 3, pp. 1-5, 2001.

[153] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern

Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[154] J. Platt, N. Cristianini and J. Shawe-Taylor, "Large Margin DAGs for Multiclass Classification," *Proc. Neural Information Processing Systems*, pp. 547-553, 2000.