Fast and Efficient Non-parametric Classification and Clustering Methods for Large Data Sets

A Thesis

Submitted For the Degree of

Doctor of Philosophy

by

V. Suresh Babu

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati – 781039

JULY 2008

Certificate

This is to certify that the thesis entitled "**Fast and Efficient Non-parametric Classification and Clustering Methods for Large Data Sets**", submitted by **V. Suresh Babu**, a research student in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for the award of the Degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Date:
Guwahati.

Dr. P. Viswanath,
Assitant Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati- 781 039, India.

Acknowledgments

Many thanks to my supervisor, Dr. P. Viswanath, for his guidance, encouragement and effort throughout my PhD duration. Every advice of him motivated me with a new zeal in research. He is a kind hearted person with lot of patience. From him, I learnt many things, both technical and personal.

I am thankful to Prof. M. Narasimha Murty, Indian Institute of Science, Bangalore for his advices which improved the thesis.

I am thankful to Prof. Sukumar Nandi, the previous head of the department of CSE and Dr. G. Sajith, the present head of the department of CSE, for the research facilities made available to me in the department.

Thanks must also go to my doctoral committee, Dr. D. Goswami, Dr. Pinaki Mitra and Dr. G. Sajith for their valuable comments on my thesis work.

Thanks to Dr. Saibaba, Deputy Librarian for his advices on both technical and personal matters.

The following are my colleagues who made my stay at IIT Guwahati a pleasant one and I am thankful to each one of them

- Mr. Tejmani Sinam, who helped me in understanding and solving many technical problems.
- Mr. D. Satyanarayana, a kind person from whom I learnt hardworking.
- Mr. Nityananda Sarma, who helped me in the thesis writing.
- Mr. Sraban Kumar Monhanty, who helped me in many of the occasions.

- Mrs. Mamata, who has great patience.
- Mr. Bidyut, a cool person who helped me with several discussions.
- Mr. Akhilesh Kumar and Mr. Bibhas for their help in understanding some fundamentals of mathematics.
- Mr. Rajendra, Mr. Neminath and Mr. W. Godfrey for their association.
- Ms. Alka, who is an intelligent colleague and who helped me with my teaching assistantship work.

Last, but never the least, I am thank to my family members: Mrs. Sankaramma, my mother, Mr. V. Venkatesulu, my brother, Mr. Mohan Babu, my cousin, Mr. V. Jayachandra Naidu, my cousin, Mrs. Kavitha, my sister-in-law. Their support is crucial throughout my career.

Contents

List of Tables

List of Figures

Abstract

Pattern classification and clustering are two prominent pattern recognition tasks applied in various domains. Non-parametric methods are those which does not assume any model or distribution form for the data. Hence these methods are more general and can give better results provided the data set is a larger one.

Nearest neighbor classifier (NNC) and its variants like k nearest neighbor classifier (k-NNC) are popular non-parametric classifiers. They show good performance and has asymptotic behavior comparable to that of the Bayes classifier. When it comes to clustering methods, DBSCAN (Density Based Spatial Clustering of Applications with Noise) uses density which is found non-parametrically at a point in order to derive density based clusters. DBSCAN can find arbitrary shaped clusters (unlike methods like k-means clustering) along with noisy outliers detection.

Non-parametric methods can work well when the data sets are sufficiently large. But these methods has high computational requirements since they need to store the entire data set and needs to search it repeatedly. Hence these methods requires to scan the data set several times because of which these are not suitable for data mining applications where data set sizes are large.

The thesis proposes hybrid ways of achieving similar results as that of the above mentioned conventional methods but in a very less time. It is proposed to derive prototypes called leaders first from the data set and then to use them in the non-parametric methods instead of the data set. But since only leaders cannot be used to find the density at a point in the feature space, these methods cannot give good performance. To overcome this the thesis proposes to use along with leaders additional information which enables in

finding the density estimates. Each leader is associated with a weight which measures the relative importance of the leader and are used with nearest neighbor based classification. For clustering, each leader is associated with a count value which is the number of patterns for which the the leader is representative. Further, rough-fuzzy membership values are used to enhance their performance.

Some properties of leaders like a lower bound and an upper bound on the number of leaders are formally established. Along with this, an important drawback of using leaders alone for nearest leader based classification is established. That is, under certain conditions, the nearest leader based classifier is nothing but randomly guessing a class-label. One can overcome this severe drawback by using weighted leaders.

The methods proposed in the thesis are (i) the weighted k-nearest leader classifier, (ii) the rough-fuzzy weighted k nearest leader classifier, (iii) rough-DBSCAN and (iv) rough-fuzzy DBSCAN. Experimentally these are compared with the conventional methods using various standard and synthetic data sets and are shown to achieve similar results as achieved by the conventional k nearest neighbor classifier and DBSCAN. But the proposed methods are very fast than their conventional counterparts. Especially the proposed methods are scalable to work with large data sets like those in data mining applications.

Chapter 1

Introduction

Pattern recognition, in general, deals with (i) *pattern classification*, where a query pattern is classified into one of the predefined set of classes based on already classified examples (called *training set*), (ii) *pattern clustering*, where the aim is to find a natural grouping (clustering) among the given set of patterns, and (iii) *dimensionality reduction*, where the objective is to find a subset of features or a subspace of the feature space which captures salient description of the patterns [Jain et al. (2000b)]. There are many methodologies proposed for pattern classification and pattern clustering which are statistical approaches [Jain et al. (2000b)], [Kulkarni et al. (1998)], [Jain et al. (1999)], [Xu and Wunsch (2005)], graph theoretic approaches [Angelova and Weikum (2006)], [Schenker et al. (2003)], [Hammouda and Kamel (2004)], classification tree based approaches [Rokach and Maimon (2005)] *etc,*. These encompasses a wide range of information processing problems of great practical significance, *viz.,* speech recognition [Chibelushi et al. (2002)], classification of handwritten characters [Plamondon and Srihari (2000)], fault detection in machinery [Fenton et al. (2001)], medical diagnosis [Jain et al. (2000a)], spam filtering [Jain et al. (2000a)], document categorization [Tang et al. (1996)], [P. Viswanath et al. (2008)] remote sensing [Gutierrez-Osuna (2002)], image segmentation [Jain et al. (2002)], face recognition [Hjelm et al. (2001)], video tracking & recognition [Jain et al. (2002)], bio-informatics [Altman and Dugan (2003)], data mining [Mitra et al. (2002)], Web mining [Mitra et al. (2002)], forensic sciences [Michael (2002)] *etc.* Often many of these are

the problems which many humans solve in a seemingly effortless fashion and are called *learning* problems. However, their solution using machines has, in many cases, proved to be immensely difficult. The most general, and most natural, framework in which to formulate solutions to pattern recognition problems is a statistical one, which takes into account the probabilistic nature of the problem and its solutions.

1.1 Pattern classification

Pattern classification deals with building classifiers where the objective is to classify a given query pattern q into one of a class which is a member of a set of predefined classes. Here, a training set of the form $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is given where $x_1, x_2, \ldots, x_n$ are patterns and $y_1, y_2, \ldots, y_n$ are their respective class-labels. The problem is to use the training set or a model derived to get a suitable class-label for the given query pattern. To state more formally, let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ be the set of classes, $\mathcal{X}$ be the feature space spanned by d dimensions (where each dimension is called a feature). The classifier is a function $f : \mathcal{X} \rightarrow \Omega$.

The performance of a classifier is measured in terms of error rate (or classification accuracy rate) which measures the degree by which the classifier is wrong (or correct). In most of the cases this is the fraction of patterns that are wrongly classified (or rightly classified) by the classifier when averaged over the entire feature space. Let $P(x, y)$ be the distribution from which the training set is drawn. Then the error rate (or the actual risk) of the classifier is

$$R(f) = \int L(y, f(x)) dP(x, y) \tag{1.1}$$

where y is the class-label assigned to the pattern x by an expert (supervisor, hence this kind of activity is called supervised learning), L is the loss function which measures the disagreement between the expert and the classifier. Even though, both x and y are random variables, traditionally the expert, for a given x associates one y only, also the classifier's output *i.e.,* $f(x)$ is one value only. Hence, in this setting, it may not be possible to build a classifier which can have zero error rate. So the objective is to build a classifier which

has the least possible error rate.

The problem is aggravated by the fact that $P(x, y)$ is unknown to us and also expert may not be able to give a y value for all possible x values. What is given to us is a sample data set drawn (usually assumed to be independently and identically drawn) form the distribution $P(x, y)$. This is divided into training set and test set. The training set is used to build the classifier, whereas the test set is used to measure the performance of the classifier. Training patterns are the patterns which are seen by the classifier and the testing patterns are unseen. The error rate on the training set is called the training error or the empirical risk. If $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is the training set, then

$$R\text{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)). \tag{1.2}$$

It is easy to build a classifier which has zero training error (for example, the nearest neighbor classifier), but this does not guarantee that the actual risk or the test error is small for the classifier. A classifier is said to have good generalization ability if it performs well on the unseen patterns.

Some reasons for poor generalization ability of a classifier are, (i) over training (the classifier is too intensively optimized on the training set), (ii) the number of features could be too large relative to the number of training samples (curse of dimensionality [Jain and Chandrasekharan (1982)]), and (iii) the number of unknown parameters associated with the classifier could be large (for example, a large neural network). If everything else are same, a simple classifier is argued to have better generalization ability than a complex classifier (Occam's Razor [Duda et al. (2000)]). Statistical learning theory [Vapnik (1995)], [Vapnik (1998)] deals with these aspects where the complexity of a classifier is measured in terms of *capacity* or Vapnik-Chervonenkis (VC) dimensionality [Vapnik (1998)].

Classification paradigms can be categorized as those which depends either implicitly or explicitly on probability densities and their estimated values called as the *Bayesian approach*, and the other one where the decision boundary (discriminant function) is directly

estimated (instead of going via the intermediate step of estimating the probability densities) using the given training set. Examples of this second approach are artificial neural networks, support vector machines, etc, and decision trees (rule-based classifiers) which does not require that the patterns are from a metric space. Since the thesis emphasizes on the Bayesian approach, the Bayes classifier is explained in detail below.

1.1.1 The Bayes classifier

Among various classifiers, one of the early and most widely cited classifiers is the Bayes classifier. If we know the distribution $P(x, y)$ from which the data is drawn, then for a given query pattern one can assign the most probable class-label according to the given distribution. When class-conditional distributions for each class is given along with prior probabilities for each class, then the Bayes theory can be used to find the posterior probabilities for each class and can be used in deciding the class-label of the query pattern q. Let $p(x = q \mid y = \omega_i)$ for $i = 1$ to c be the class-conditional densities. Let $P(y = \omega_i)$ be the prior probability for class ω_i. Then the posterior is $P(y = \omega_i \mid x = q) = \frac{p(x=q|y=\omega_i)P(y=\omega_i)}{p(x=q)}$. The class-label chosen is that for which the posterior is maximum. It is easy to combine the Bayes decision theory with any loss function and is shown to achieve the minimum possible risk (or the error-rate).

Once the statistical structure in the form of distributions is given to us, as shown above, it is very easy to build the best classifier, *viz.,* the Bayes classifier. But since, the distributions are not given, if one wants to use the Bayes theory then one has to first estimate the distributions from the given training set. Since these estimation can go wrong, the Bayes classifier which uses these estimated probabilities is no longer an optimal one! Nevertheless, this is a widely used approach, often called *the Bayesian approach.*

Two ways in which the probability distributions (or probability densities) can be estimated are the parametric methods and the non-parametric methods as described below.

1.1.2 Parametric methods

In parametric methods, a parametric form for the distribution (or the density function) is assumed (like Gaussian form) where the parameters (like mean and covariance matrix, in case Gaussian form is assumed) are estimated using methods like maximum-likelihood method, Bayesian parameter estimation method, Expectation maximization method, etc. Even when the data is not from that chosen parametric form, the method gives a distribution of the chosen form. So the estimates can go severely wrong and often the real world data is not from any known parametric form. Non-parametric methods, which does not assume any form are more general and with sufficiently large data sets are more accurate than parametric methods.

1.1.3 Non-parametric methods

Assuming that the patterns are drawn form a metric space, the expected number of patterns that would be present in a small region R is

$$E(k) = nVp(x)$$

where n is the number of patterns in the data set, V is the volume of the region, $p(x)$ is the probability density at a point $x \in R$. Here, the assumption is that the density function $p(\cdot)$ over R is a uniform one. From this, an expected value of $p(x)$ denoted by $\hat{p}(x)$ is

$$\hat{p}(x) = \frac{k/n}{V}$$

where k is the number of patterns falling in R out of the n patterns in the data set. It is shown that, as $V \to 0$, $k \to \infty$, $n \to \infty$ and $k/n \to 0$, $\hat{p}(x)$ will converge to $p(x)$ [Duda et al. (2000)]. For the estimate to be a better one, n should be large and V should be small (V can be specified as a function of n like $1/\sqrt{n}$).

The above approach can be generalized by using a *kernel function* $H(x)$ as

$$H(x) = \begin{cases} 1 & \text{if } x_i \text{ is present in } R, \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

and by using

$$\hat{p}(x) = \frac{1}{nV} \sum_{i=1}^{n} H(x_i) \tag{1.4}$$

where $\{x_1, \ldots, x_n\}$ is the given data set. The *Parzen window* approach, in general, assumes that R is a hyper-cube. If d is the dimensionality of the space, and h is length of an edge of the hyper-cube, then $V = h^d$ and the window function can be defined as

$$H(x_i) = \varphi\left(\frac{x - x_i}{h}\right) \tag{1.5}$$

which is equal to unity if x_i falls within the hyper-cube, and is zero otherwise.

One problem of using a kernel function of the form given in Equation 1.5 is that the density function is not a smooth one and is sensitive to the choice of h (or R, in general). Because of the limited nature of the given data set, it is possible that at certain regions of the space there are no points falling within the hyper-cube. To overcome these problems more general kernel functions (than just a window function) are proposed where contribution from each data element is taken into account. For example, a common choice is a multivariate normal kernel like

$$\frac{1}{V}\varphi\left(\frac{x - x_i}{h}\right) = \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{||x - x_i||^2}{2h^2}\right) \tag{1.6}$$

One can use the above mentioned techniques to find the necessary densities which can be used with the Bayes classifier. But, there is a simple and direct way of deriving a non-parametric classifier called the nearest neighbor classifier (NNC) and its variants like the k-nearest neighbor classifier (k-NNC). Since the thesis deals with non-parametric classifiers NNC and k-NNC are described separately below.

1.1.4 Nearest neighbor based classifiers

Let the region R is chosen so that it is going to encompass the k nearest neighbors in the training set for the given query pattern. Let k_i be the number of neighbors (among the k neighbors) which are from the class ω_i. Then the class-conditional density for the class ω_i at q *i.e.*, $p(x = q \mid y = \omega_i) = \frac{k_i/n_i}{V}$ where n_i is the number of training patterns in class ω_i and V is the volume of R. An estimate of the prior probability $P(y = \omega_i)$ according to the training set is $\frac{n_i}{n}$ where n is the total number of training patterns. Then the posterior probability $P(y = \omega_i \mid x = q)$ will be $\frac{k_i/n_i}{V}\frac{n_i}{n} = \frac{k_i/n}{V}$. Since n and V are independent of the classes, the posterior $P(y = \omega_i \mid x = q)$ is proportionate to k_i. Hence, choosing the class according to the majority voting among the k nearest neighbors is approximately same as doing the Bayes classification. This classifier is called the *k-nearest neighbor classifier (k-NNC)* and when $k = 1$, this is called the *nearest neighbor classifier (NNC)*. NNC and k-NNC are very popular non-parametric classifiers [Dasarathy (1991)], [Duda et al. (2000)]. These are widely used because of their simplicity and good performance. It is shown that NNC with infinite number of training patterns achieves error-rate which is less than twice the Bayes error-rate [Duda et al. (2000)]. It has no design phase, except for choosing the value k which can be found by performing a cross-validation as explained in subsequent sections.

1.1.5 Cross-Validation

In pattern recognition paradigm, validation techniques are generally used to select the optimal parameters of a classifier or a model for a given classification problem. There are one or more free variable(s) that exists for a classifier (E.g., number of neighbors in a k-NNC classifier) and the optimal values for the free variables are to be chosen before using the classifier for the unseen patterns (test patterns). Many methods are described in literature which are random sub-sampling, r-fold cross-validation and leave-one-out cross-validation methods.

Cross-validation methods choose the optimal value for the parameters to which the classifier has minimum average error rate. Random sub-sampling performs r data splits

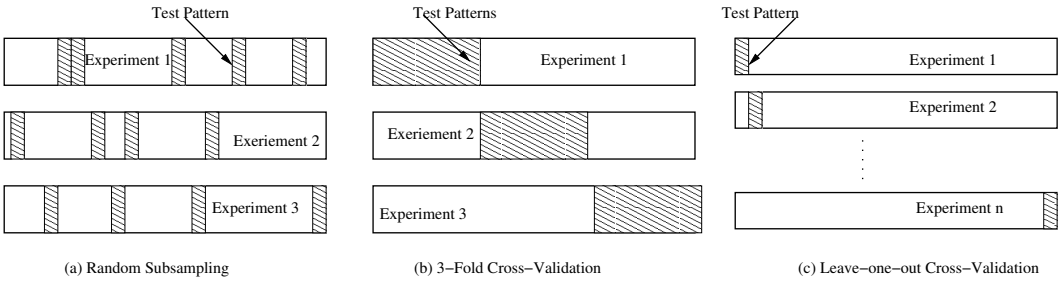


Figure 1.1: Cross-Validation Methods.

of the training set, each split randomly selects a fixed number of patterns without replacement [Kohavi (1995)]. The classifier error rate is estimated for each split by considering different parameter values and the parameter value is chosen for which the average error rate of a classifier is minimum. In case of r-fold cross-validation, the r-disjoint subsets are chosen whose union is equal to the training set and each of the subsets are associated with randomly chosen patterns. For each of r experiments, use the union of $r-1$ subsets for training set and the remaining one subset for testing and the optimal parameters are chosen for which the average error rate of a classifier is minimum. The advantage of r-fold cross-validation is that all the examples in the data set are eventually used for both training and testing. Leave-one-out cross-validation is the special case of r-fold cross-validation in which r is equal to n where n is the number of patterns in the training set. Figure 1.1 explains the above three methods diagrammatically. Generally, three fold cross-validation method is used to estimate the optimal parameters of a classifier.

1.2 Pattern clustering

Apart from pattern classification, the other important pattern recognition task is the pattern clustering. It is about finding *natural* groups or *clusters* in the given data set so that patterns in a cluster are more similar than patterns in two distinct clusters. Here, the data set is unlabeled (that's why this is an unsupervised learning) unlike the classification task where the training set where class-labels of patterns are known is given

to us. Similarity between patterns or clusters is found by a similarity measure or by a distance measure (like Euclidean distance).

The output of a clustering activity is either a hard partition of the data set where each pattern is assigned to exactly one cluster, or a soft partition where a pattern may belong to more than one cluster, possibly with a membership value (which measures the degree by which a pattern belongs to a cluster, like a fuzzy membership value).

Clustering methods can be broadly divided into two, *viz.,* Hierarchical and Partitional. In Hierarchical methods, a hierarchy of partitions is obtained where each level (of the hierarchy) represents a partition of the data set. If π_i and π_{i+1} are two successive levels, then normally, either π_i is a refinement of π_{i+1} or π_{i+1} is a refinement of π_i. Clustering at one level is obtained by either merging few blocks (clusters) of the previous level or by splitting a cluster into a few blocks. Single-link, complete-link, average-link clustering methods are the most widely used methods of this category. Partitional clustering methods initially obtains a partition (may be a random partition) and progressively refines it to minimize a criterion function. A popular method of this category is the k-means clustering method [MacQueen (1967)] which tries to minimize a squared error criterion [Jain et al. (1999)]. Graph-theoretic methods where a minimal spanning tree of the data is constructed and a few largest length edges are removed to generate connected components of the graph which forms a clustering also comes under this category. Apart from this a few others are based on probabilistic models like mixture-resolving and mode-seeking methods where the assumption is that the patterns to be clustered are drawn from one of several distributions, and the goal is to identify the parameters which is mostly done by maximum-likelihood or Expectation Maximization methods [Duda et al. (2000)].

Nonparametric methods for density-based clustering have also been developed Jain et al. (1999) Jain and Dubes (1988) These are inspired by the Parzen-window method to nonparametric density estimation. One of the recent and widely used density based clustering method of this type is the DBSCAN (Density Based Spatial Clustering of Applications with Noise) [Ester et al. (1996)] which groups dense and nearby patterns together to form clusters. This approach can automatically find noisy outliers which is

not achieved in most of the other methods. Since the thesis deals with DBSCAN and its improvements it is described in a separate section below.

1.2.1 DBSCAN

Most partitional clustering methods discover clusters based on the distance (similarity) between patterns. In contrast with this approach, DBSCAN uses a notion of density at a point to perform clustering and continues growing the cluster as long as the patterns are densely connected. Hence it can find an arbitrary shaped clusters along with detection of noisy outliers. It does not require to specify the number of clusters. Density at a point in a feature space is found non-parametrically by counting the number of patterns that fall in a small region around the given point. That is, suppose m number of patterns out of n patterns in a data set are falling in a small region around the given point, then the density at the point is computed as m/nV. Nevertheless, a generalized *kernel function* $H(x)$ can be used to compute the density at the point which is explained in Section 1.1.3. But, DBSCAN requires to specify the *kernel function* $H(x)$ for estimating density. It also requires to specify threshold density value using which it can make sure that the given point is a dense or non-dense. If the density at a point exceeds the threshold value then the point is a dense point else it is a non-dense point.

Suppose for a given region is assumed to be a hyper sphere of radius ϵ at a pattern and hence the threshold density can be specified by a parameter $MinPts$, minimum number of patterns required to be present in the region to make it dense. According to DBSCAN if a pattern x is dense [1] then it is part of a cluster. A non-dense pattern can also be part of a cluster as a border pattern of the cluster if it is at a distance less than or equal to ϵ from a dense pattern, otherwise it is a noisy outlier. Two patterns x_1 and x_2 are in a cluster if there is a sequence of patterns $x_1, y_1, y_2, \ldots, y_m, x_2$ in the data set such that:

1. the distance between successive patterns in the sequence is less than or equal to ϵ,

2. if $m = 0$, then at least one of x_1 and x_2 is dense, and

[1] DBSCAN Ester et al. (1996) calls a dense point as a core point and a non-dense point as a non-core point. This modification is done to bring the material closer to the pattern recognition community

3. if $m > 0$, then the patterns $y_1, y_2, \ldots, y_m$ are dense.

DBSCAN requires $O(n)$ time complexity to find the density at a point and also it requires $O(n^2)$ time complexity to find consistent clusters. Hence, DBSCAN is not suitable for large data sets. One way to overcome this problem is to built an index over the data set like *R-trees* [Guttman (1984)], [*KD-trees*] [Bentley (1975)] *etc.,* which will be useful for finding density at a pattern. But this solution is suitable only when the dimensionality of data is low.

1.3 Dimensionality reduction

Dimensionality reduction is a process that makes the classification task more effective and efficient by using a subspace or by representing the patterns with a subset of features. If the process uses a subspace based on transformations or combinations of the original feature set, then it is called feature extraction. Some of the well known feature extraction methods are principal component analysis (PCA) [Jolliffe (1986)], [Duda et al. (2000)], independent component analysis (ICA) [Comon (1994)], [Cardoso (1998)] , *etc.* A survey of other methods are discussed in [Jain et al. (2000a)], here we describe briefly about PCA and ICA. These two methods differ in the way in which they achieve their objective, while PCA tries to find the direction in feature space that best represent the data in a sum-squared error sense, ICA seeks directions that are most independent from each other.

On the otherhand, if the process uses to represent patterns with a subset of features which has almost same capability as original feature set, then these are called feature selection methods. Of course, an exhaustive search to find minimal set of features which has same capability as that of original features requires exponential time and no non-exhaustive sequential features selection procedure can be guaranteed to produce the optimal subset [Cover and Campenhout (1977)]. This makes to look into some of the heuristic based methods for the problem of feature selection. Some of the well known feature selection methods [Devijver and Kittler (1982)], [Jain and Zongker (1997)], [Guyon and Elisseeff (2003)] are branch-and-bound search [Viswanath et al. (2007)], best individual

features, sequential forward selection, sequential backward selection *etc,*. Branch-and-bound search uses monotonicity property of the criterion function and it is guaranteed to find the optimal subset which is a computationally efficient than exhaustive search. Best individual search selects the best subset of features by evaluating each feature individually whereas sequential forward selection method selects a best single feature then add one feature at a time which in combination with the selected features maximizes the criterion function and sequential backward selection method selects all features and successively delete one feature at a time.

1.4 Problem definition

Nonparametric methods like NNC, k-NNC offers good solution for pattern classification problem, and DBSCAN offers a good solution for pattern clustering. But these methods require that the data set size should be large to get better results. Now a days, in areas like data mining, we have large data sets. The other side of the issue is that, when the data set sizes are large nonparametric methods suffers from huge computational burden and often are not feasible to use. So, the problem addressed in the thesis is *to reduce the computational burden of a few nonparametric methods where the objective is not only to reduce the computational requirements but also to see that the performance is not degraded.* The thesis deals with prototype based nonparametric methods where prototypes are selected using a fast clustering method called *the leaders clustering method.*

1.4.1 Prototypes

Non-parametric methods require that the set of patterns are to be stored to compute density at a point which has huge space and computational burden for large data sets. In order to reduce both space and time requirements, several techniques have been proposed which are called as *training set reduction, training-set condensation, reference set thinning,* and *prototype selection* methods. These methods find a set of representative

patterns which are either a subset or a new set of patterns formed such that noisy, redundant or superfluous patterns are eliminated from the given data set [Angiulli (2007)]. The process of finding representative patterns from a given data set, while the objective is to reduce the size of data set as much as possible with improved or without degraded or with less degraded in performance is called the *prototype selection*. Prototype selection is a well known and well established process in pattern classification.

Several training set condensation algorithms are introduced in the literature starting from a seminal work in [Hart (1968)] which are also known as instance-based [Aha et al. (1991)], [Brighton and Mellish (2002)], [Wilson and Martizen (2000)], lazy [Aha (1997)], memory-based [Stanfill and Waltz (1994)], and case-based learners [Watson and Marir (1994)]. These methods are grouped into three main categories based on the objective that they want to achieve [Brighton and Mellish (2002)], that is, *competence enhancement*, *competence preservation*, and *hybrid approaches*. Noisy patterns degrade the performance of the classifier whereas the redundant patterns are main cause for increase in the computational burden of the classifier. The goal of competence enhancement methods are often possible to improve the performance of a classifier by eliminating noisy or corrupt patterns. On otherhand, the goal of competence preservation methods are to eliminate redundant or superfluous patterns which will not lead to decrease in performance of a classifier while it causes to reduce the computational burden. Hybrid approaches use the advantages of both the approaches which turns out to be better approaches. These methods perform to find the consistent subset based on heuristics and are polynomial time complexity algorithms that require multiple database scans for large data sets. Unfortunately, finding a minimal-cardinality consistent subset for the classifier turns out to be intractable [Wilfong (1992)].

Many training set condensation algorithms are advantageous in-terms of eliminating noisy and redundant patterns. But, these methods require to evaluate the performance of a classifier at each iteration of the elimination process. Hence these methods are not suitable (since evaluation of the performance of a classifier has computational burden) for large data sets. In contrast to these methods there are other kinds of methods to find the

prototypes to alleviate the problem of space and computational burden for non-parametric methods. From a given large data set, to derive an appropriate abstraction (or model) is arduous. If the abstraction is compact (enough having small storage space and access time requirements), accurate (enough to carry out the activity needed), manageable (enough to build and modify easily with respect to the original data set) then data mining process becomes easy. A few representative patterns (prototypes) of the large data set can serve as a good abstraction of the data set. Partitioning the data set based on the similarity of the patterns which can be done by using clustering techniques. A few representatives are used for each partition as prototypes where the performance of classifier is not evaluated unlike in the training set condensation algorithms. Union all the representative patterns from each partition forms an abstraction for the whole data set. The thesis deals with the prototypes which are derived in a single scan of the data set and which are enriched using additional information which enables to get a better performance for non-parametric methods like NNC, DBSCAN.

1.5 Motivation for the thesis

NNC and DBSCAN offers good solutions with large data sets, but has severe shortcomings like huge computational burden and sensitive to noise. These methods implicitly or explicitly use the probability density values. Using a few selected prototypes is a remedy to overcome the computational burden. But the crucial density information that is present in the data set is often lost while deriving the prototypes. For example one can use the k-means clustering [MacQueen (1967)] method with some large value for the parameter k. These k centroids can be used as prototypes. But it is not possible to get the density estimates just by using these centroids. Similarly, just using only leaders [Spath (1980)] as prototypes can be a severely biased way. Along with this the computational requirements to derive the prototypes is also a main concern. k-means clustering method scans the data set several times and hence can be a time consuming process. Hence the thesis argues to use enriched leaders which can be derived in a single database scan. Along with leaders a weight, or count is also derived which are used in the density estimation. The weight

or count depends on the cluster analysis.

Many soft clustering methods are advantageous over hard clustering methods. *Fuzzy set theory*[Zadeh (1965)] and *Rough set theory*[Pawlak (1982)] are two mathematical frameworks which resolve uncertainty and vagueness which are briefly described in the subsequent section. Combined principles of these two theories are prominent than either of them alone [Zadeh (1965)], [Dubois and Prade (1990)], [Y. Y. Yao (1998)], [Nanda and Majumdar (1992)], [Chanas and Kuchta (1992)]. Hence the thesis also proposes to use rough-fuzzy membership values which are not only used to resolve uncertainty in the cluster analysis, but also used in subsequent classification and clustering. Hence the proposed methods are more efficient in terms of performance and also reduces the computational burden significantly.

1.5.1 Fuzzy Set Theory and Rough Set Theory

Rough sets theory [Pawlak (1982)] and fuzzy set theory [Zadeh (1965)] are well known mathematical theories to capture uncertainties associated with the data. These two theories model different types of uncertainty. The rough set theory considers the indiscernibility of elements which is typically characterized by an equivalence relation. Whereas fuzzy set theory deals with the generalization of characteristic function and it does not use indiscernibility of elements.

Let U be the finite non empty set called universe and let $X \subseteq U$ be a set . In many pattern recognition applications like classification, clustering *etc.,* it might be difficult (or impossible) to define X explicitly. For example, clustering patterns is a process of grouping patterns into some sets and defining such sets might be difficult to define explicitly. But X can be roughly defined by two sets called lower and upper approximations of X. That is, $\underline{X} \subseteq X$ is a lower approximation of X and $X \subseteq \overline{X}$ is a upper approximation of X. It might be easy to work with the sets $\underline{X}$ and $\overline{X}$ and it can give lower and upper approximations of the solution. Roughness of the solution depends on the border $\overline{X} - \underline{X}$. If the boarder is empty then the solution is exact. If the boarder is large then the solution is highly rough. The theory can be used in various ways in various applications.

Fuzzy set theory is characterized by a generalized fuzzy membership function. A fuzzy set F is defined by a pair $< U, \mu_F >$ where, $\mu_F : U \rightarrow [0, 1]$. Here $\mu_F(x)$ is the fuzzy membership of x.

Rough set theory, fuzzy set theory and their combined principles are used in many pattern recognition tasks which are feature selection [Li et al. (2006)], [Shiu et al. (2006)], clustering [Lingras and West (2004)], [Peters (2006)] and classification [Shen and Chouchoulas (2002)], *etc,.* These principles are generally used to resolve the uncertainty and vagueness in the cluster analysis and the thesis proposes to use rough-fuzzy set theory to resolve uncertainty of leaders clustering method.

1.5.2 Scope of the Thesis

The scope of the thesis is: (i) to study the properties of leaders as prototypes, (ii) to enrich the leaders with additional information which enables in estimating the probability densities, (iii) to find nearest prototype based classifiers and to compare it with existing classifiers, and finally (iv) to use the enriched leaders in finding density based clusters. The proposed methods are theoretically analyzed using rough and fuzzy set theories and experimentally these are validated using various standard data sets.

1.6 Related recent works

Partitioning the training set using well known clustering methods and representing each partition with an appropriate prototypes is a well known process of prototype selection. It does not require to evaluate the performance of a classifier at each iteration and hence the pre-processing overhead is due to the clustering method alone. A fast clustering method is used for prototype selection to reduces the pre-processing time. Leaders clustering method is a faster one which requires single scan of the data set and hence it can be used for prototype selection. Recently, *an adaptive rough fuzzy* clustering method [Asharaf and Murty (2003)] is proposed to derive the enriched prototypes for classification. Similar to this, *Leaders-Subleaders* method [Vijaya et al. (2004)] is proposed where it uses a hierarchy

of leaders and subleaders in place of a large training data set.

An adaptive rough fuzzy single pass method [Asharaf and Murty (2003)] for clustering large data sets is a variant and enriched version of the leaders method which captures the intrinsic uncertainty involved in the cluster analysis. It employs the rough set theory and hence a leader is associated with two threshold values called a lower threshold (lower approximation) and an upper threshold (upper approximation). It is an incremental clustering method with two sets of leaders (leaders and supporting leaders). It is a two phase single pass clustering method. In the first phase, each pattern finds a fuzzy membership values to each of the leaders and assigns it to either lower approximation of a leader or to upper approximation of one or more leader(s) and it is decided by using lower and upper threshold values. Note that if a pattern is assigned to lower approximation then it only belongs to that leader and if it is in the upper approximation then it belongs to more leaders with some fuzzy membership. It gets added to the supporting leader if a pattern belongs to overlapping leaders which are having equal voting for all available classes. If a pattern belongs to overlapping leaders whose majority voting is not same as leader class label then it is added to the leader set. And also if the pattern is not within the upper threshold of any of the leader in the leader set then it becomes a new leader. Another case can arise if the number of overlapping leaders are below some threshold value where the pattern becomes a new supporting leader. This method adaptively changes the upper threshold of each overlapping leader if the number of overlapping leaders are above the threshold value. At the end of the first phase, adaptive rough fuzzy leaders method has a set of leaders and a set of supporting leaders. In the second phase, the method checks each supporting leader if it is incorrectly classified or not by using the set of leaders, then accordingly the supporting leader is added into the leaders set and hence it improves the quality of the prototype set.

The *Leaders-Subleaders* [Vijaya et al. (2004)] method is an extension of the leaders method which derives a two level hierarchy of prototypes in two database scans. In this method, leaders-subleaders are derived from the whole training set (i.e., the method is not applied for each class of training patterns separately). User defined thresholds τ_1

and τ_2, where $\tau_2 \leq \tau_1$, are used as an input for this method which are chosen based on the maximum and the minimum distances between a pair of patterns in the data set. Initially, leaders clustering algorithm with a threshold τ_1 is applied on the training data set which finds a set of leaders and its followers (set of patterns which belongs to the leader). Later, leaders clustering method is applied for the followers of each leader with threshold tau_2 and hence subleaders are derived within the followers of each leader. In the classification phase, leaders-subleaders are the representatives for the whole training set. The classification method is, for a test pattern it's nearest leader in the leader set is found first and then the nearest subleader in the subleader set of the nearest leader is found. The test pattern is assigned a label based on the nearest of these two.

1.7 Organization of the thesis

Chapter 2 describes properties of leaders which shows a few theoretical results on an upper bound and a lower bound of the number of leaders under a few assumptions. Chapter 3 initially proves the limitation of leaders based classifier and explains weighted k-nearest leader classifier. Chapter 4 explains rough-fuzzy leaders clustering method and its extension which is analyzed using rough set theory. Chapter 5 describes rough-DBSCAN method and establishes a relationship between rough-DBSCAN and DBSCAN. Rough-DBSCAN is further extended to rough-fuzzy DBSCAN and that which uses leaders and subleaders. Finally, chapter 6 summarizes the contributions of the thesis and gives some future research directions.

Chapter 2

Properties of Leaders

2.1 Introduction

Since the thesis is about prototype based non-parametric classification and clustering methods where the prototypes are chosen by employing the leaders clustering method (and hence the prototypes are called *leaders*), this chapter presents a few important properties of leaders which are formally established with the help of a few assumptions. These are basically to show that the number of leaders is small when compared with the data set size. A lower bound on the number of leaders is given which for small data sets depends on both the data set size and the distribution from which the data set is drawn. An upper bound on the number of leaders is also given which is independent of the size of the data set from which the leaders are derived and also is independent of the probability distribution from which the data set is drawn. Experimental studies are done for various data sets which are shown as plots between the number of leaders versus the varying data set sizes.

The chapter is organized as follows. Notation and definitions are given in Section 2.2. Leaders clustering method is described in Section 2.3. A few important properties of leaders are described in Section 2.4. Experimental study for the various data sets are shown in Section 2.5 which shows that, when the threshold value (a parameter used by the leaders clustering method) is appropriately chosen, then asymptotically, the number

of leaders is upper bounded by a constant. Finally, Section 2.6 describes some of the conclusions of this chapter.

2.2 Notation and Definitions

1. *Classes:* There are c classes *viz.*, $\omega_1, \ldots, \omega_c$.

2. *Training set:* The given training set is $\mathcal{D}$. The training set for class ω_i is $\mathcal{D}_i$, for $i = 1$ to c. The number of training patterns in class ω_i is n_i, for $i = 1$ to c. The total number of training patterns is n.

3. *Apriori probabilities:* Apriori probability for class ω_i is $P(\omega_i)$, for $i = 1$ to c. If this is not explicitly given, then $P(\omega_i)$ is taken to be n_i/n, for $i = 1$ to c.

4. *Leaders :* The set of leaders (prototypes) for class ω_i is $\mathcal{L}_i$, for $i = 1$ to c. The set of all leaders is $\mathcal{L} = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_c$. The leaders in the set $\mathcal{L}_i$ is an ordered set. The ordering may be simply the order in which the leaders are derived.

5. *Sub-leaders:* A j^{th} leader in class ω_i has an ordered set of sub-leaders denoted by $\mathcal{SL}_{ij}$.

6. *Saturated region:* With respect to a given data set, and the set of leaders $\mathcal{L}$ derived from it using a threshold distance τ, we say that a region R in the feature space is a *saturated* one, if for each pattern x in R there is a leader l in $\mathcal{L}$ such that $||l - x|| \leq \tau$. In other words, even when a new pattern from R is added to the data set, $\mathcal{L}$ is unaffected.

7. *Overlapping region:* A region R in the feature space is said to be an overlapping region, if for all x such that $x \in R$, we have $p(x|\omega_1) > 0$ and $p(x|\omega_2) > 0$.

8. $B_\tau(x)$: This is $\{y \mid ||x - y|| \leq \tau\}$, *i.e.*, a closed ball of radius τ at x. Assuming that τ is small enough to take the probability distribution over $B_\tau(x)$ as a uniform one.

9. $N_\tau(z; \mathcal{D})$: This is the set of patterns in the data set $\mathcal{D}$ that are present in the region $B_\tau(x)$.

2.3 Leaders clustering method

Leaders method finds a partition of the given data set like most of the clustering methods. Its primary advantage is its running time which is linear in the size of the input data set. To be more precise, it can find the partition in $O(n)$ time where n is the data set size. It needs to read (or scan) the data set only once from the secondary memory and hence is also termed as an *on-line clustering method*. Because of these factors the leaders method is gaining popularity in the fields like *data mining* where the data set sizes are very large.

The leaders method is given in Algorithm 2.1. For a given threshold distance τ, leaders method works as follows. It maintains a set of leaders $\mathcal{L}$, which is initially empty and is incrementally built. For each pattern x in the data set $\mathcal{D}$, if there is a leader $l \in \mathcal{L}$ such that distance[1] between x and l is less than or equal to τ, then x is assigned to the cluster represented by l. In this case, we call x as a *follower* of the leader l. Note that even if there are many such leaders, only one is chosen. If there is no such leader then x itself becomes a new leader and is added to $\mathcal{L}$. The algorithm outputs the set of leaders $\mathcal{L}$. Each leader can be seen as a representative for the cluster of patterns which are grouped with it. As

Algorithm 2.1 Leaders($\mathcal{D}$, τ)

$\mathcal{L} = \emptyset$;
for each $x \in \mathcal{D}$ **do**
  Find a $l \in \mathcal{L}$ such that $||l - x|| \leq \tau$
  if there is no such l or when $\mathcal{L} = \emptyset$ **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
  end if
end for
Output $\mathcal{L}$.

a clustering method the leaders clustering method has the following shortcomings. Let the distance threshold used in the method be τ. The distance between two patterns that are grouped with a leader are guaranteed to be at-most 2τ. In this context, *followers* of a *leader* can be seen as similar. But there is no guarantee that *followers* of two different

[1]For the sake of simplicity, we assume that the patterns are from a Euclidean space and Euclidean distance is used. Whereas the proposed methods are applicable with any distance metric.

leaders are separated by a distance of at-least τ. Indeed, even when we present the same pattern twice, both of these may not be assigned to the same leader. So, patterns in distinct clusters need not be dissimilar. The clusters for two leaders can be as shown in Figure 2.1.

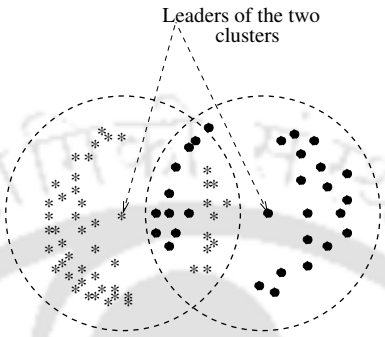


Figure 2.1: Leaders is a bad clustering method.

To some extent these problems can be reduced as follows.

1. Ordering the leaders when they are derived: A pattern x is assigned to a leader l such that $||l-x|| \leq \tau$. If there are more than one such leader then a leader is chosen according to a predefined ordering of the leaders. The ordering could be the same as the ordering in which the leaders are created. The resulting clustering is a set of semi-spheres as shown in Figure 2.2.

2. The clusters can be further refined by first deriving the leaders set *i.e.,* $\mathcal{L}$ and then reassigning each pattern to the closest leader in the set $\mathcal{L}$. This requires one more database scan. This step, we call as the *refinement step.*

2.4 Leaders: some properties

With a few assumptions which simplifies the analysis, as described below, we want to make an important claim about the number of leaders, which is : *there is an upper-bound on the number of leaders which is independent of the size of the data set from which the*

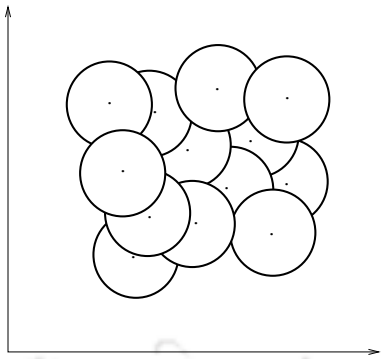


Figure 2.2: Semi-spherical clusters.

leaders are derived and also is independent of the probability distribution from which the data set is drawn. A lower bound on the number of leaders is also given which for small data sets depends on both the data set size and the distribution from which the data set is drawn.

Let the leaders set $\mathcal{L}$ is derived using a threshold distance τ from the data set $\mathcal{D}$. Each pattern x in $\mathcal{D}$ is *i.i.d.* drawn from a probability density function $p(x)$. Since the scanning order of $\mathcal{D}$ influences $\mathcal{L}$, we assume that $\mathcal{D}$ is an ordered data set, the ordering being same as the scanning order. The scanning order is specified by a mapping $m : \{1, 2, \ldots, n\} \rightarrow \mathcal{D}$ where $|\mathcal{D}| = n$. The set of all such mappings is denoted by $\mathcal{O}$. Similarly $\mathcal{L}$ is an ordered set, the ordering being same as the order in which the leaders are added to the initially empty set $\mathcal{L}$ which is incrementally built (see Algorithm 5.2). Let $B_\tau(z)$ be the region which is a closed ball of radius τ at z in the feature space. For a pattern $z \in \mathcal{D}$, let $N_\tau(z; \mathcal{D}) = \{x \mid x \in \mathcal{D}, ||z - x|| \leq \tau\} = \{x_1, x_2, \ldots, x_{r+1}\}$. $N_\tau(z; \mathcal{D})$ is the set of neighbors of z (including z) which are in $B_\tau(z)$. Apart from z there are r patterns in $N_\tau(z; \mathcal{D})$ where r is a non-negative integer.

The following observations and assumptions are used in the subsequent derivations.

1. If l_1 and l_2 are two distinct leaders in $\mathcal{L}$, then $||l_1 - l_2|| > \tau$.

2. For a pattern $z \in \mathcal{D}$, if z is not a leader then there is a leader l such that l is in $N_\tau(z; \mathcal{D}) - \{z\}$ and z is a follower of l.

3. It is assumed that, the scanning order is independent of $|N_\tau(z;\mathcal{D})|$, *i.e.,* the number of training patterns in $B_\tau(z)$.

4. It is assumed that, for a pattern $z \in \mathcal{D}$, a pattern $y \in \mathcal{D} - N_\tau(z;\mathcal{D})$ has no influence on z to be a leader or z to be not a leader. But y can have an indirect influence on z in cases where y is a leader and has a follower which is in $N_\tau(z;\mathcal{D})$. This assumption is made to simplify the analysis.

5. It is assumed that, $\int_{B_\tau(z)} p(x)dx = p(z)V_\tau$, where V_τ is the volume of the closed ball $B_\tau(z)$.

6. It is assumed that, the data set is drawn from a closed and bounded region of the feature space[2]. Theoretically, this need not be true. But, in all most all practical situations, the range of values a feature can take is bounded. Hence the data set is from a bounded region of the feature space is not an unrealistic assumption.

Lemma 1 *Given that a pattern z is in $\mathcal{D}$, provided that $|N_\tau(z;\mathcal{D})| = r+1$, the expected probability by which z can become a leader, averaging over $\mathcal{O}$ (*i.e., *all scanning orders of $\mathcal{D}$), is $E_{\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{1}{r+1}$.*

Proof: Since a pattern $y \in \mathcal{D} - N_\tau(z;\mathcal{D})$ has no influence on z to be a leader or z to be not a leader, $P(z \in \mathcal{L} \mid z \in \mathcal{D})$ is independent of the ordering of patterns in $\mathcal{D} - N_\tau(z;\mathcal{D})$. Because of this, we can restrict our attention to patterns in $N_\tau(z;\mathcal{D})$ alone. z can definitely become a leader when z occurs, according to the scanning order, before all of the patters in $N_\tau(z;\mathcal{D}) - \{z\}$. The total number of orderings of patterns in $N_\tau(z;\mathcal{D})$ is $(r+1)!$. The number of orderings in which z occurs before others is $r!$. So, $E_{\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{r!}{(r+1)!} = \frac{1}{r+1}$, since there are situations where a pattern y in $N_\tau(z;\mathcal{D}) - \{z\}$ even though occurs before z according to the scanning order, might be a follower of some other leader and hence is not in competition with z to become a leader $\square$

[2]In other words, this assumption is same as saying that the feature space is a compact metric space.

Lemma 2 *Let* $|N_\tau(z;\mathcal{D})| = r+1$ *and let* $\mathcal{R} = \{0,1,\ldots,n-1\}$. *Averaging over all possible values of* r (i.e., $\mathcal{R}$) *and all possible scanning orders of* $\mathcal{D}$ (i.e., $\mathcal{O}$), *for a given pattern* z *in* $\mathcal{D}$, *the expected probability by which it can become a leader is*

$$E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{1}{Pn} - \frac{(1-P)^n}{Pn}$$

$$\textit{where} \quad |\mathcal{D}| = n, \quad \textit{and } P = \int_{B_\tau(z)} p(x)dx.$$

Proof: Since it is assumed that the scanning order is independent of $|N_\tau(z;\mathcal{D})|$, $E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] = E_{\mathcal{R}}[E_{\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})]]$

Since it is given that $z \in \mathcal{D}$, from the remaining $(n-1)$ *i.i.d.* patterns, the probability that r patterns can fall in $B_\tau(z)$ is $\binom{n-1}{r}P^r(1-P)^{n-1-r}$. This is the probability by which $(r+1)$ patterns among the n patterns in $\mathcal{D}$ can fall in $B_\tau(z)$. So, $E_{\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{1}{r+1}$ with a probability $\binom{n-1}{r}P^r(1-P)^{n-1-r}$, and $r \in \mathcal{R}$. So, the expected value of $E_{\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})]$ averaging over all possible values of r is,

$$\begin{aligned} E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] &\geq \sum_{r=0}^{n-1} \frac{1}{r+1}\binom{n-1}{r}P^r(1-P)^{n-1-r} \\ &\geq \sum_{r=0}^{n-1} \frac{1}{Pn}\binom{n}{r+1}P^{r+1}(1-P)^{n-(r+1)} \end{aligned}$$

By replacing $r+1$ with j, we get

$$\begin{aligned} E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] &\geq \sum_{j=1}^{n} \frac{1}{Pn}\binom{n}{j}P^j(1-P)^{n-j} \\ &\geq \frac{1}{Pn}\left(\sum_{j=0}^{n}\binom{n}{j}P^j(1-P)^{n-j}\right) - \frac{1}{Pn}\binom{n}{0}P^0(1-P)^n \\ &\geq \frac{1}{Pn}(P+(1-P))^n - \frac{(1-P)^n}{Pn} \\ &\geq \frac{1}{Pn} - \frac{(1-P)^n}{Pn} \quad \square \end{aligned}$$

Since z is given to be in $B_\tau(z)$, we have $P \neq 0$. Lemma 2 is an important one and some situations to understand this better are given below.

1. When $n = 1$, z is the only pattern in $\mathcal{D}$, and hence $E[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{1}{P} - \frac{1-P}{P} = 1$. That is, z must be a leader.

2. When $P = 1$, all n patterns lie in $B_\tau(z)$, hence $E[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{1}{n} - \frac{0}{n} = \frac{1}{n}$.

Theorem 2.4.1 *Let the data set $\mathcal{D}$ is drawn from a closed and bounded region S of the feature space. Then the number of leaders that can be derived from $\mathcal{D}$ using a threshold distance τ, such that $\tau > 0$, is at-least $\frac{V_S}{V_\tau} - \int_S \frac{(1-p(z)V_\tau)^n}{V_\tau}\, dz$, where V_S is the volume of the region S and V_τ is the volume of a hyper-sphere of radius τ.*

Proof: The expected value of $E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})]$ averaging over S is

$$E_{S\times\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] = \int_S E_{\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D}_R)]\, p(z)\, dz$$

$$\geq \int_S \left(\frac{1}{Pn} - \frac{(1-P)^n}{Pn}\right) p(z)\, dz$$

where $P = \int_{B_\tau(z)} p(x)dx$ and $|\mathcal{D}| = n$.

Since it is assumed that $\int_{B_\tau(z)} p(x)dx = p(z)V_\tau$, and hence

$$E_{S\times\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \int_S \frac{1}{np(z)V_\tau} p(z)dz - \int_S \frac{(1-P)^n}{np(z)V_\tau} p(z)dz$$

$$\geq \frac{V_s}{nV_\tau} - \int_S \frac{(1-P)^n}{nV_\tau} dz$$

Now $E_{S\times\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})]$ is the average probability by which a pattern in $\mathcal{D}$ is a leader. Since there are n patterns in $\mathcal{D}$ which are i.i.d drawn from a distribution, the number of leaders k is given by

$$k = nE_{S\times\mathcal{R}\times\mathcal{O}}[P(z \in \mathcal{L} \mid z \in \mathcal{D})] \geq \frac{V_s}{V_\tau} - \int_S \frac{(1-p(z)V_\tau)^n}{V_\tau} dz \quad \square$$

Theorem 2.4.1 gives a lower-bound on the number of leaders. For large values of n, the value of $(1 - p(z)V_\tau)^n$ is very small which can be neglected and hence the lower-bound on the number of leaders can be taken as $\frac{V_S}{V_\tau}$. The following theorem gives an upper-bound on the number of leaders.

Theorem 2.4.2 *Let the data set $\mathcal{D}$ is drawn from a closed and bounded region S of the feature space. Then the number of leaders that can be derived from $\mathcal{D}$ using a threshold distance τ, such that $\tau > 0$, is at most $\frac{V_S}{V_{\tau/2}}$ where V_S is the volume of the region S and $V_{\tau/2}$ is the volume of a hyper-sphere of radius $\tau/2$.*

Proof: Let the leaders set be $\mathcal{L}$ and $|\mathcal{L}| = k$. For two distinct leaders l_1 and l_2 in $\mathcal{L}$, it is guaranteed that $||l_1 - l_2|| > \tau$. Assume that at each leader $l \in \mathcal{L}$ we place a closed ball of radius $\tau/2$, *i.e.*, $B_{\tau/2}(l)$. These closed balls will not intersect with each other. The total volume of these closed balls is $kV_{\tau/2}$. Since, the data set is assumed to be drawn from a bounded region whose volume is V_S, we have $kV_{\tau/2} \leq V_S$. Hence $k \leq \frac{V_S}{V_{\tau/2}}$ □

Let for $n = n_0$, $\int_S \frac{(1-p(z)V_\tau)^n}{V_\tau} dz = 1$. Then for $n \geq n_0$ we have:

$$\int_S \frac{(1 - p(z)V_\tau)^n}{V_\tau} dz \leq 1 \tag{2.1}$$

Corollary 2.4.1 *If $\tau > 0$, for $n \geq n_0$ the following inequalities holds,*

$$\left(\frac{V_s}{V_\tau}\right) - 1 \quad \leq \quad k \quad \leq \quad \left(\frac{V_s}{V_{\tau/2}}\right) \tag{2.2}$$

$$k \leq n \tag{2.3}$$

Proof: From Theorems 2.4.1, 2.4.2, and Inequality (2.1) the Inequality (2.2) directly follows. The set of leaders is a subset of the data set from which the leaders are derived and hence the Inequality (2.3) is true □

Inequality (2.2) states that for a sufficiently large data set (for which $n > n_0$) the number of leaders has lower and upper bounds which are independent of the data set size and also are independent of the distribution function from which the data set is drawn. Inequality (2.3) states that the number of leaders can never exceed the data set size. Hence for large data sets, even when n is infinity, since k is a finite and bounded one, working with leaders has less time requirements when compared to that of working with the entire data set.

The thesis proposes to use leaders in place of the data set. The leaders are further enriched by associating them with a weight or rough-fuzzy weight which measures their relative importance and hence can improve the performance of subsequent classifier or clustering method. So, the proposed methods are hybrid and approximate ways of achieving similar performance as the conventional ones like k-NNC and DBSCAN.

The number of leaders derived depends on the value of the threshold τ, on the scanning order, and on the data set size (if it is not a sufficiently large data set). Theoretical upperbounds and lowerbounds are to guarantee the asymptotic behavior of the methods. To find a practical value for τ, if the memory constraint in the form of number of leaders that can be accommodated in the memory is given, then one can do as follows. By performing a few experiments (as explained in the Section 2.5) with varying values of τ one can obtain their corresponding respective number of leaders. Then for the given number of leaders, an appropriate value of τ can be interpolated (or extrapolated) by using any standard regression method.

The thesis studies how the performance and computational requirements varies with the varying τ values.

2.5 Experimental results

Experimental studies are done for various data sets to compare number of leaders against the data set size. Three synthetic data sets and twelve standard data sets are used in this thesis. One synthetic data set is used for classification purposes and other two synthetic data sets are used for clustering purposes which are described in Chapter 3, Chapter 4

and Chapter 5, respectively. Among the standard data sets six data sets are used for classification purposes and other five data sets are used for clustering purposes. These are available at *http://www.ics.uci.edu/mlearn/MLRepository.html*. The details of the data sets are given in Table 2.1 and in Table 2.2. Experiments are being conducted for all these data sets and plots are drawn with varying data set sizes along the horizontal axis versus number of leaders along the vertical axis. Four different threshold values are chosen for each data set which lie between the minimum and the maximum distance between pair of patterns in the data set. For a given threshold value, the rate in which the number of leaders are growing is found to be reducing as the data set size grows.

2.5.1 Data sets used for classification purpose

For the data sets used for classification purpose, leaders clustering method is applied for each class of patterns separately and the number of leaders is taken as the cumulative sum of the number of leaders of each class. But, the same threshold values is chosen when the leaders clustering method is being applied to each class of patterns. Figure 2.3 shows the plots between the number of leader versus the varying data set sizes for various data sets which are used for classification purpose. *Synthetic* and *Covtype* data sets are large and their plots are clearly showing that the number of leaders are upper bounded by a constant and hence they are approaching parallel to the horizontal axis as the data set size increases. *SenseIT Vehicle (seismic, acoustic and combined)* data sets are also showing similar behavior. Finally *Ijcnn1* is a medium size data set and its plot clearly shows that the number of leaders are smaller when compared with the size of the data set.

2.5.2 Data sets used for clustering purpose

For the data sets used for clustering purpose, leaders clustering method is applied without considering the class labels into account. Figure 2.4 shows the plots between the number of leaders versus the varying data set sizes for various data sets which are used for clustering purpose. *Gaussian Mixture(synthetic)*, *Shuttle*, *A8a* and *A9a* data sets are large and their

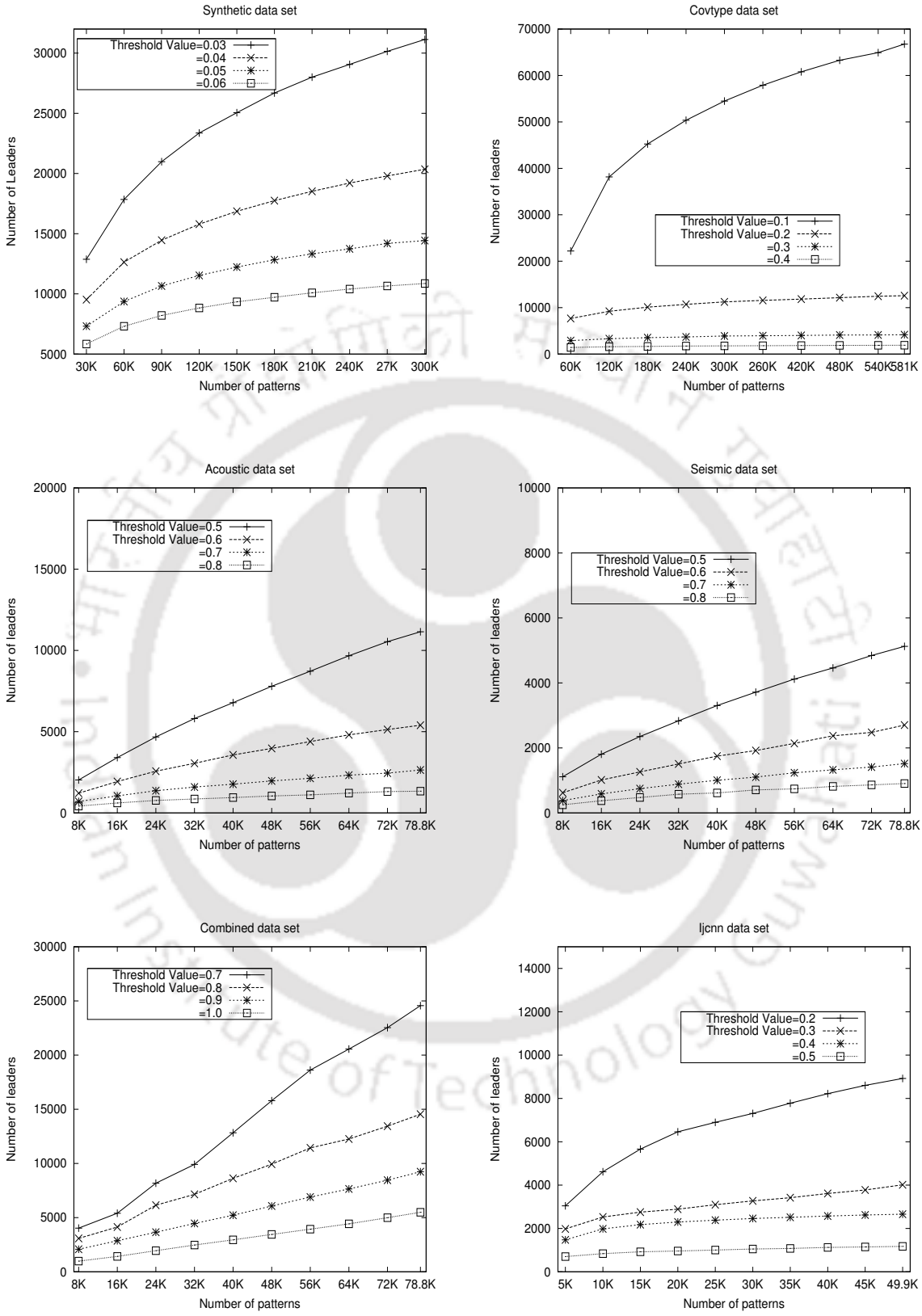


Figure 2.3: Number of leaders Vs. the data set size for data sets used for classification purposes

Table 2.1: Details of data sets used for classification purpose

Data set	Number of Features	Number of Classes	Number of Training patterns	Number of Test patterns
Synthetic	2	2	300000	100000
Covtype.binary	54	2	400000	181012
SensIT vehicle(seismic)	50	3	78823	19705
SensIT vehicle(acoustic)	50	3	78823	19705
Pendigits	16	10	7494	3498
SensIT vehicle(combined)	100	10	78823	19705
Ijcnn1	22	2	49990	91705

Table 2.2: Details of data sets used for clustering purpose

Data set	Number of Features	Number of patterns
Gaussian Mixture (synthetic1)	2	30000
Banana (synthetic2)	2	4000
Shuttle	9	58000
Letter	16	20000
A8a	123	32561
A9a	123	48842
Pendigits	16	10992

plots are clearly showing that the number of leaders are upper bounded by a constant and hence they are approaching parallel to the horizontal axis as the data set size increases. Finally *Letter* and *Pendigits* are small data sets and their plots clearly shows that the number of leaders are smaller when compared with the size of the data sets

2.6 Conclusions

Some important properties which relates the number of leaders with the data set size are formally established. Experimental studies are also done to compare the number of leaders against the data set size. Since the number of leaders are fewer, computationally it is advantageous to work with the leaders set than to work with the entire data set. Further,

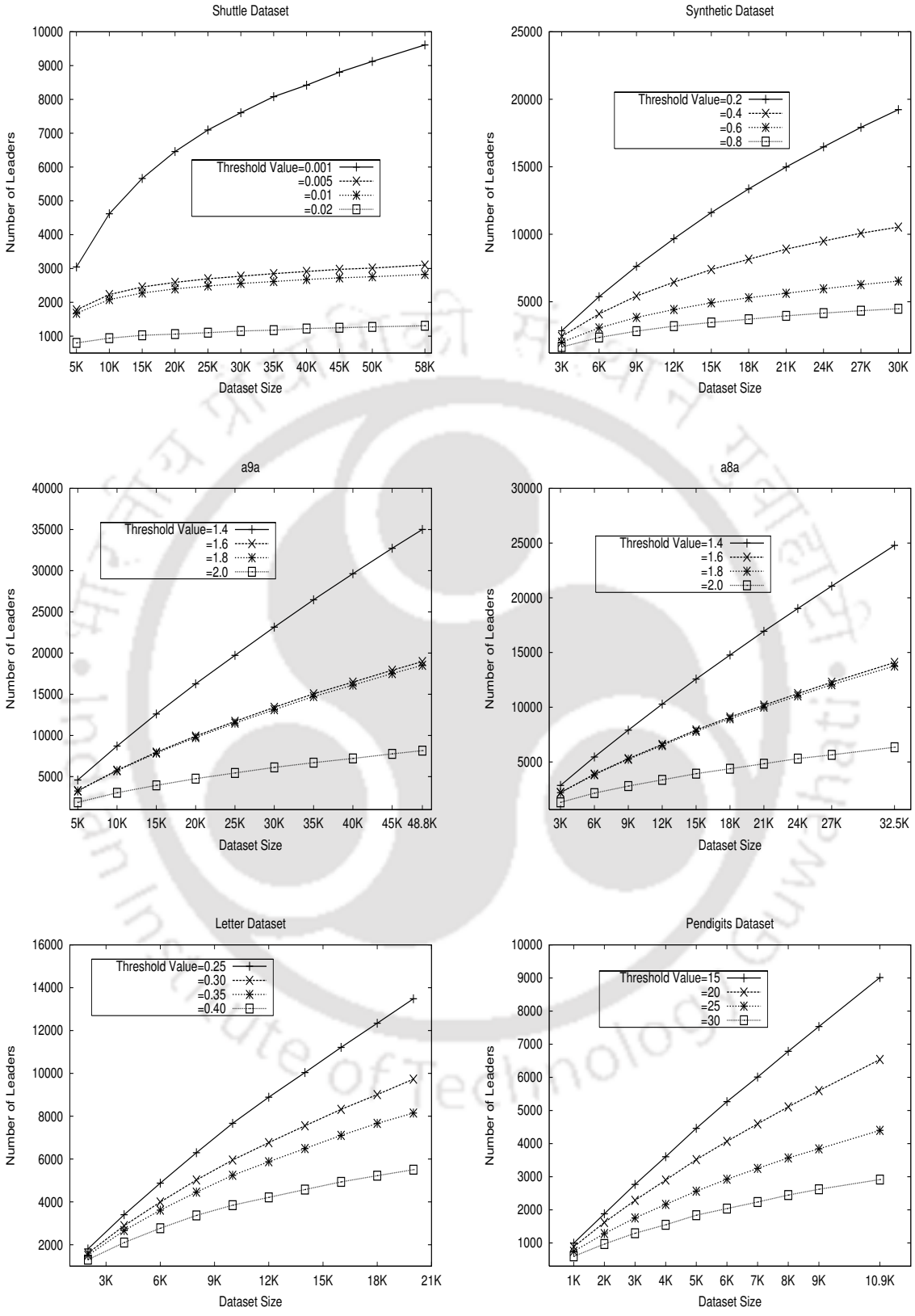


Figure 2.4: Number of leaders Vs. the data set size for data sets used for clustering purposes

leaders can be derived by scanning the data set only once in a linear time, and hence this does not add to the overhead. Experimental results shows that the rate in which the number of leaders grows is smaller when compared with the rate in which the data set size is increased. When the data set sizes are large, like in data mining applications, it is better to use leaders instead of the entire data set.

Chapter 3

Weighted k-nearest leader classifier

3.1 Introduction

Nearest neighbor classifier (NNC) and its variants like k-nearest neighbor classifier (k-NNC) are popular because of their simplicity and good performance [Duda et al. (2000)]. It is shown that asymptotically (with infinite number of training patterns) k-NNC is equivalent to the Bayes classifier and the error of NNC is less than twice the Bayes error [Cover and Hart (1967)], [Duda et al. (2000)]. NNC has no design (training) phase, hence it is highly adoptive to the dynamically varying data sets. It has certain limitations and shortcomings as listed below. (1) It requires to store the entire training set. So the space complexity is $O(n)$ where n is the training set size. (2) It has to search the entire training set in order to classify a given pattern. So the classification time complexity is also $O(n)$. (3) Due to the curse of dimensionality effect its performance can be degraded with a limited training set for a high dimensional data. (4) Presence of noisy patterns in the training set can degrade the performance of the classifier.

Some of the various remedies for the above mentioned problems are as follows. (1) Reduce the training set size by some editing techniques where we eliminate some of the training patterns which are redundant or noisy in some sense [Dasarathy (1991)]. For example, Condensed NNC [Hart (1968)] is of this type. (2) Use only a few selected prototypes from the training set [Vijaya et al. (2004)]. Prototypes can be selected by

partitioning the training set by using some clustering techniques and then taking a few representatives for each block of the partition as the prototypes. Clustering methods like *Leaders* [Spath (1980)], *k-means* [Jain et al. (1987)], *etc.,* can be used to derive the prototypes. (3) Reduce the effect of noisy patterns. Preprocessing the training set and removing the noisy patterns is a known remedy.

With data mining applications where typically the training set sizes are large, the space and time requirement problems are severe than the curse of dimensionality problem. Using only a few selected prototypes can reduce the computational burden of the classifier, but this can result in a poor performance of the classifier. *Leaders-Subleaders* method [Vijaya et al. (2004)] applies the leaders clustering method to derive the prototypes. The classifier is to find the nearest prototype and assign its class label to the test pattern. While the *Leaders-Subleaders* method reduces the classification time when compared with NNC or k-NNC which uses the entire training set, it also reduces the classification accuracy.

This chapter establishes theoretical arguments which explains the nearest leader based classifier is nothing but a random guessing of a class label for a given test pattern which is from a saturated and overlapping region between two or more classes. Later, this chapter attempts at presenting a generalization over the *Leaders-Subleaders* [Vijaya et al. (2004)] method where along with the prototypes we derive its importance also which is used in the classification. Further, we extend the method to k nearest prototypes based classifier [V. Suresh Babu and Viswanath (2007)], [V. Suresh Babu et al. (2008)] instead of 1-nearest prototype based one as done in the *Leaders-Subleaders* method. To improve the performance, noisy prototypes, which is appropriately defined as done in some of the density based clustering methods, are eliminated.

The chapter is organized as follows. Section 3.2 explains training set reduction methods briefly. Section 3.3 explains the limitation of the nearest leader based classifier with an example. Section 3.4 explains weighted leaders clustering method along with a preprocessing method for elimination of noisy prototypes. Section 3.5 explains the weighted k-nearest classifier which is an approximate classifier to the k-NNC classifier. Section 3.6 describes the experimental study for various data sets.

3.2 Training set size reduction methods

This section describes training set size reduction methods for nearest neighbor based classifiers. Condensed nearest neighbor rule (CNN) is an initial work proposed by Hart [Hart (1968)] to reduce the training set size which finds some subset S of a training set that correctly classifies every pattern in the training set by using NNC. That is, if a pattern in the training set is mis-classified using the set S then it must be kept in the set S. Initially S is an empty set and is built incrementally which depends on the scanning order of the training set. Hence, CNN rule is applied with multiple runs on the randomly permuted training set in-order to improve the quality of the consistent subset [Alpaydin (1997)]. Gates [Gates (1972)] proposed the reduced nearest neighbor (RNN) rule which is similar to CNN which builds a consistent set decrementally. That is, initially assuming set S is a training set and the patterns in S which upon deletion from S does not leave any pattern in the training set mis-classified are to eliminated from S. The selective nearest neighbor rule (SNN) is proposed by Ritter *et al.* [Ritter et al. (1975)] which improves on CNN and RNN by attempting to find a minimal consistent subset. Modified condensed nearest neighbor rule (MCNN) [Susheela Devi (2000)] is an order independent rule that tries to find a consistent subset whereas CNN is an order dependent rule. All these rules tries to eliminate redundant or superfluous patterns in-order to reduce the computational burden.

Wilson proposed another rule which is a just opposite of the CNN rule and it attempts to remove noisy patterns [Wilson (1972)]. The rule is all patterns which are mis-classified using nearest neighbor rule are eliminated. Chang's algorithm [Chang (1974)] and Bezdek *et al* [Bezdek et al. (1998)] presented a novel approach to generate a condensed training set by merging pair wise repeatedly. A generalized prototype-based classification is proposed by Mollineda *et al* [Mollineda et al. (2002)] which is an improvement over Chang's algorithm and uses hierarchical clustering technique to select prototypes. All condensed training set methods require to evaluate the performance of classifier at each step and hence these methods are having huge computational burden for large data sets. Clustering training set using fast clustering methods and selecting few representatives from each

cluster is one of the remedy which is used in leaders-subleaders method.

3.3 Limitation of the nearest leader based classifier

Nearest leader based classifier uses a set of leaders in place of the training set to overcome the computation burden of the nearest neighbor classifier for large data sets. A very important property of using leaders as prototypes for a nearest prototype based classifier as done by P.A. Vijaya *et. al.* [Vijaya et al. (2004)] is as follows.

- Considering a two class problem where the classes are denoted by ω_1 and ω_2, if the given query pattern q is from an overlapping region which for both of the classes is a saturated one, the nearest leader for q (which is obtained from the set of leaders) has the class label either ω_1 or ω_2 with equal probability, *i.e.,* with probability 0.5.

An example with theoretical analysis is described below for the property which clearly establishes the limitation of using leaders alone for the nearest prototype based classification. Further it is independent of the probability distributions from which the data sets are drawn. In other words, this is to say that, in some cases, the nearest leader based classifier is nothing but randomly assigning a class-label to the given test pattern (this is irrespective of the posterior probabilities).

3.3.1 An Example

Consider a one dimensional two class problem which has equal prior probabilities. The class conditional densities are uniformly distributed for both the classes as shown in the Figure 3.1. Patterns of class ω_1 are from the regions A and B (see Figure 3.1) and patterns of class ω_2 are from B and C. For a query pattern from the region B, if we employ the Bayes classifier, the misclassification probability will be 0.111, and when we employ a NNC with sufficiently large training set drawn from the same distributions then it will be less than 0.222. Whereas, for a given threshold $\tau > 0$, for sufficiently large training sets where the region B become saturated for both the classes then the misclassification

probability of the nearest leader based classifier will be 0.5. This argument is formally proved in the sub-section 3.3.2.

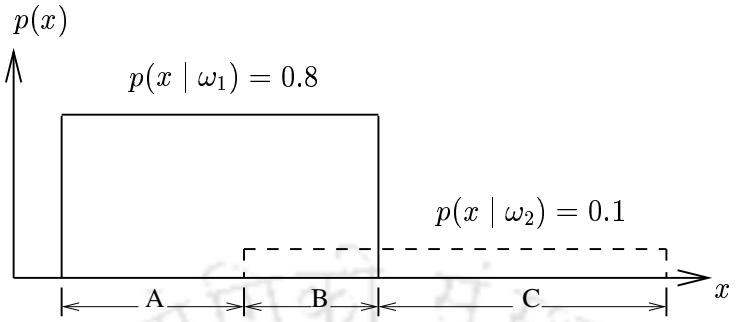


Figure 3.1: Class conditional densities: an example.

3.3.2 Theoretical Analysis

Lemma 3 *For a pattern x let l be the nearest leader. If x is from a saturated region R, assuming that the probability distribution from which the data set is drawn is a continuous and uniform one over $N_\tau(x)$, the expected value of $||l - x||$* i.e., *$\mathcal{E}||l - x||$ is a constant which is independent of the probability distribution.*

Proof:

Since x is from a saturated region, there is a leader l such that $||l - x|| \leq \tau$. Hence, the distribution (probability density function) over $N_\tau(x)$ is a non-zero one. Let this density which is assumed to be uniform over $N_\tau(x)$ be $\bar{p}$. For all y such that $y \in N_\tau(x)$, $p(y \in D \mid y \in N_\tau(x))$ is same. And hence the probability by which y can become the nearest leader of x is same for all y such that $y \in N_\tau(x)$. So the expected distance between x and its nearest leader l is

$$\mathcal{E}||l - x|| = \int_{N_\tau(x)} p(y|y \in N_\tau(x))||y - x||dy$$

We have,

$$p(y|y \in N_\tau(x)) = \frac{p(y \in \mathcal{D} \ \ and \ \ y \in N_\tau(x))}{P(y \in N_\tau(x))} = \frac{\bar{p}}{\bar{p}V} = \frac{1}{V}$$

where V is the volume of the ball $N_\tau(x)$. Hence

$$\mathcal{E}||l - x|| = \frac{1}{V}\int_{N_\tau(x)} ||y - x||dy$$

which depends on τ and is independent of $\bar{p}$. Since τ is the threshold distance used to derive the leaders which is a constant one, $\mathcal{E}||l - x||$ is a constant □

Theorem 3.3.1 *For a given query pattern q, if its nearest leader is from an overlapping region which for both of the classes is a saturated one, then the nearest leader is either from class ω_1 or from class ω_2 with equal probability, i.e., with probability 0.5.*

Proof: Since the leaders set for both of the classes are derived using the same threshold distance, *i.e.,* the τ value, and also since it is assumed that the nearest leader is from a saturated and overlapping region (between the two classes), the expected distance between the query pattern q and its nearest leader from class ω_1 is same as that from class ω_2. Hence the nearest leader is chosen by breaking the tie between these two leaders, which is done by randomly choosing either ω_1 or ω_2. Hence the probability by which the class-label assigned to q is either ω_1 or ω_2 with probability 0.5 □

For sufficiently large data sets, the consequence of Theorem 3.3.1 is a severe one when the class label chosen for the query pattern is the class label of its nearest leader. If the class conditional distributions for both of the classes are of a Gaussian form then the overlapping region is equal to the entire feature space. In this case, as the training set size increases, the saturated regions will grow and for many query patterns the class label chosen is a random one. In this case, with an infinite number of training patterns, the nearest leader classifier randomly assigns a class label for each and every query pattern and hence its asymptotic error rate is 0.5, which is the maximum possible error-rate.

3.4 Weighted Leaders clustering method

The thesis proposes to use along with leaders their relative importance too to overcome the problems that are explained in previous sections.

The leaders method is modified as given below where along with each leader we derive a weight also.

1. For each class of training patterns, the leaders method is applied separately, but the same distance threshold parameter, *i.e.*, τ is used. The set of leaders derived for class ω_i is denoted by L_i, for $i = 1$ to c.

2. Each leader l has an associated weight denoted by $weight(l)$ such that $0 \leq weight(l) \leq 1$.

3. For a training pattern x which belongs to the class ω_i, if there are p leaders that are already derived such that their distance from x is less than τ, then weight of all these p leaders is updated. Let l be a leader among these p leaders, then its new weight is found by $weight(l) = weight(l) + 1/(p \cdot n_i)$. If $p = 0$, then x itself becomes a new leader whose weight is $1/n_i$.

The modified leaders method called *Weighted-Leaders* is given in Algorithm 3.1.

Algorithm 3.1 Weighted-Leaders($\mathcal{D}_i$, τ)

$L_i = \emptyset$;
for each $x \in \mathcal{D}_i$ **do**
  Find the set $P = \{l \mid l \in L_i, ||l - x|| < \tau\}$;
  if $P \neq \emptyset$ **then**
    for each l such that $l \in P$ **do**
      $weight(l) = weight(l) + 1/(\mid P \mid \cdot n_i)$.
    end for
  else
    $L_i = L_i \cup \{x\}$;
    $weight(x) = 1/n_i$;
  end if
end for
Output L_i which is a set of tuples such that each tuple is in the form $< l, weight(l) >$ where l is a leader and $weight(l)$ is its weight.

The leaders and their respective weights derived depends on the order in which the data set is scanned. For example, for a pattern x, there might be a leader l such that $||l - x|| < \tau$ which is created in a later stage (after considering x) and hence the weight of

l is not updated. By doing one more data set scan these kind of mistakes can be avoided. But since the method is devised to work with large data sets, and as the training set size increases the effect these mistakes diminishes, they are ignored.

We assume that the patterns are from a Euclidean space and Euclidean distance is used.

3.4.1 Eliminating noisy prototypes

Since noise (noisy training patterns) can degrade the performance of nearest neighbor based classifiers, we propose to eliminate noisy prototypes in this section.

A leader l which belongs to the class ω_i is defined as a noisy prototype if (1) the class conditional density at l is less than a threshold density and (2) there are no neighbors for l which are dense (*i.e.,* the class conditional density at each of these neighbors is less than the threshold density). This definition is similar to that used in density based clustering methods like DBSCAN [Ester et al. (1996)] and its variants [Viswanath and Pinkesh (2006)].

This process is implemented as follows. For a given ϵ distance, for each leader l (say this belongs to the class ω_i), we find all leaders from class ω_i which are at a distance less than ϵ. Let the set of these leaders be S. Let the cumulative weight of these leaders be W. Then we say that l is a non-dense prototype if $W < \delta$, for a predefined threshold weight δ. If all leaders in the set S are non-dense, then we say l is a noisy prototype and is removed from the respective set of leaders.

Since any two leaders that belongs to a class are separated by distance of at least τ (the threshold used in deriving the leaders), ϵ is normally taken to be larger than τ. Section 5.5 describes about how these parameters are chosen.

The process of eliminating noise can take time at most $O(\mid L \mid^2)$ where L is the set of all leaders. Since $\mid L \mid << n$, where n is the data set size, the noise elimination process will not take much time.

3.5 Weighted k-nearest leaders classifier

This section describes the proposed classifier. Let L_i be the set of leaders obtained after eliminating noisy leaders for class ω_i, for $i = 1$ to c. Let L be the set of all leaders. That is, $L = L_1 \cup \ldots \cup L_c$. For a given query pattern q, the k nearest leaders from L is obtained. For each class of leaders among these k leaders, their respective cumulative weight is found. Let this for class ω_i be W_i, for $i = 1$ to c.

Let the k nearest leaders are from the region R at q. Approximately class conditional density at q for class ω_i is:

$$\hat{p}(q \mid \omega_i) = \frac{m_i}{n_i \cdot V}$$

where m_i is the number of patterns that are present in the region R that belongs to the class ω_i and n_i is the total number of training patterns for the class ω_i, and V is the volume of the region R. Asymptotically as $n_i \to \infty, m_i \to \infty, m_i/n_i \to 0$ and $V \to 0$, it can be shown that $\hat{p}(q \mid \omega_i) \to p(q \mid \omega_i)$ [Duda et al. (2000)].

It is easy to see that

$$W_i \approx \frac{m_i}{n_i}$$

and hence is proportionate to the $\hat{p}(q \mid \omega_i)$. So, $W_i \cdot P(\omega_i)$ is proportionate to the posterior probability $\hat{P}(\omega_i \mid q)$ where $P(\omega_i)$ is the apriori probability for class ω_i.

The classifier chooses the class according to $argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$. If $P(\omega_i)$ is not explicitly given then it is taken to be n_i/n where n_i is the number of training patterns from class ω_i and n is the total number of training patterns.

From the above argument, it is clear that the proposed method is approximately doing the Bayes classification as done by the k-nearest neighbor classifier.

The proposed k nearest leader classifier is given in Algorithm 3.2.

Algorithm 3.2 k-Nearest-Leader(L, q)

{L is the set of all leaders derived from all classes. q is the query pattern to be classified}
Find k nearest leaders of q from L.
Among the k nearest leaders find the cumulative weight of leaders that belongs to each class. Let this be W_i for class ω_i, for $i = 1$ to c.
Class label assigned for $q = argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$.

3.6 Experimental Results

Experimental studies are done with one synthetic data set and seven standard data sets. The details of the standard data sets are given in Table 2.1 and the details of the synthetic data set is given below

A two dimensional *synthetic* data for a two class problem is generated as follows. First class having 60000 patterns were *i.i.d.* drawn from a normal distribution having mean as $(0, 0)^T$ and covariance matrix as $I_{2\times2}$(*i.e.,*identity matrix). Second class also is of 60000 patterns which is also *i.i.d.* drawn from a normal distribution with mean $(2.56, 0)^T$ and covariance matrix $I_{2\times2}$. The Bayes error rate for this synthetic data set is 10%. The data set is divided randomly into two parts consisting of 80000 and 40000 patterns which are used as training and testing sets respectively.

The classifiers chosen for the comparative study are: (1) the nearest neighbor classifier(NNC), (2) the k-nearest neighbor classifier(k-NNC), (3) the nearest leader classifier(NLC), (4) the k-nearest leader classifier(k-NLC) and (5) the weighted k-nearest leader classifier(wk-NLC) which is the proposed one in this paper. NLC and k-NLC are similar to NNC and k-NNC, except that, instead of nearest neighbor(s), nearest leader(s) are taken in to consideration. For wk-NLC experiments are done with noise elimination preprocessing and without it.

The experiments are conducted for various leader's threshold *i.e.,* τ values for each data set. The parameter k value is chosen using three fold cross-validation method. There are two tables given for each data set. First table represents the cross-validation results and the second table represents the comparison of different classifiers. Some data sets the parameter k is large for the classifiers which in-turn makes a larger size cross-validation result table and hence some of the entries of the k values are missing in the table in-order to represent the table in a compact manner. For the elimination of noisy prototypes, the ϵ value is chosen as twice the maximum value among the different threshold values for a given data set and the parameter δ (to decide whether a prototype is a dense or non-dense) used as the weight threshold to eliminate noise which is chosen as 5% of the average weight of the leaders in the respective data sets.

3.6.1 Synthetic Data Set

For *Synthetic* data set the τ values chosen are $0.02, 0.03, 0.04, 0.05, 0.06$. The k values obtained from the cross validation results are shown in Table 3.1. For the k-NNC, the k value is 74 and it is 25 for wk-NLC for each of the threshold values. The comparison of different classifier results for *synthetic* data set is summarized in Table 3.2.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. The classification time and classification accuracy(CA) of the leader based classifiers depends on the threshold τ. As the value τ reduces, the classification time increases, and also the CA increases. With $\tau = 0.03$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 10% . With noise elimination wk-NLC shows some improvement with respect to classification accuracy over wk-NLC without noise elimination. Also with noise elimination wk-NLC can be slightly faster than wk-NLC without noise elimination.

3.6.2 Covtype Data Set

For *Covtype* data set the τ values chosen are $0.1, 0.15, 0.2, 0.25, 0.3$. The k values obtained from the cross validation results are shown in Table 3.3. Since the *Covtype* data set is showing good result with NNC and hence the cross-validation results for k-NNC is not given. The k values for wk-NLC from the cross-validation results are $2, 3, 5, 7$ and 11 for the thresholds $0.1, 0.15, 0.2, 0.25$ and 0.3 respectively. The comparison of different classifier results for *Covtype* data set is summarized in Table 3.4.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. With $\tau = 0.1$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 1% . With $\tau = 0.1$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

Table 3.1: Synthetic data set (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		0.06	0.05	0.04	0.03	0.02
1	85.49(0.05)	70.21(0.21)	72.16(0.19)	73.86(0.14)	76.65(0.14)	79.87(0.05)
10	89.32(0.03)	89.78(0.03)	89.75(0.05)	89.76(0.06)	89.76(0.05)	89.70(0.08)
20	89.71(0.07)	89.95(0.05)	89.95(0.03)	89.93(0.06)	89.93(0.07)	89.85(0.04)
21	89.74(0.04)	89.95(0.02)	89.96(0.03)	89.94(0.06)	89.94(0.06)	89.89(0.05)
22	89.72(0.05)	89.95(0.04)	89.96(0.03)	89.96(0.05)	89.92(0.06)	89.91(0.04)
23	89.74(0.07)	89.96(0.03)	89.94(0.03)	89.96(0.03)	89.93(0.04)	89.90(0.02)
24	89.75(0.07)	89.95(0.03)	89.97(0.02)	89.97(0.04)	89.95(0.03)	89.93(0.01)
25	89.79(0.08)	**90.02(0.04)**	**90.03(0.03)**	**90.01(0.06)**	**89.99(0.02)**	**89.96(0.02)**
26	89.77(0.06)	89.98(0.05)	89.98(0.04)	89.95(0.05)	89.96(0.05)	89.91(0.02)
27	89.80(0.06)	89.98(0.05)	89.97(0.04)	89.94(0.05)	89.97(0.05)	89.91(0.02)
28	89.78(0.05)	89.99(0.05)	89.96(0.06)	89.95(0.03)	89.97(0.03)	89.93(0.03)
29	89.81(0.05)	89.99(0.04)	89.95(0.04)	89.96(0.03)	89.95(0.01)	89.93(0.02)
30	89.82(0.05)	89.99(0.04)	89.97(0.04)	89.95(0.04)	89.97(0.05)	89.92(0.03)
40	89.88(0.06)	89.99(0.02)	89.99(0.04)	89.98(0.05)	89.98(0.04)	89.96(0.04)
50	89.91(0.04)	89.96(0.03)	89.99(0.03)	89.97(0.02)	89.99(0.03)	89.95(0.04)
60	89.93(0.06)	89.95(0.01)	89.97(0.04)	89.95(0.03)	89.97(0.05)	89.96(0.02)
70	89.94(0.04)	89.94(0.02)	89.96(0.05)	89.97(0.02)	89.98(0.04)	89.94(0.03)
71	89.93(0.04)	89.93(0.01)	89.95(0.03)	89.97(0.03)	89.96(0.03)	89.92(0.04)
72	89.92(0.03)	89.93(0.03)	89.96(0.02)	89.96(0.02)	89.95(0.02)	89.91(0.04)
73	89.92(0.03)	89.91(0.01)	89.96(0.05)	89.97(0.02)	89.98(0.04)	89.91(0.05)
74	**89.95(0.03)**	89.89(0.03)	89.93(0.03)	89.96(0.04)	89.96(0.02)	89.92(0.03)
75	89.94(0.03)	89.87(0.04)	89.94(0.04)	89.94(0.02)	89.92(0.05)	89.89(0.02)
76	89.93(0.03)	89.85(0.03)	89.93(0.02)	89.93(0.03)	89.91(0.01)	89.87(0.04)
77	89.94(0.03)	89.86(0.01)	89.92(0.03)	89.91(0.01)	89.92(0.03)	89.88(0.02)
78	89.94(0.03)	89.87(0.04)	89.89(0.03)	89.89(0.05)	89.90(0.04)	89.86(0.02)
79	89.93(0.02)	89.85(0.03)	89.86(0.01)	89.88(0.03)	89.89(0.01)	89.84(0.03)
80	89.91(0.02)	89.84(0.04)	89.86(0.02)	89.85(0.03)	89.86(0.02)	89.83(0.03)

3.6.3 Pendigits Data Set

For *Pendigits* data set the τ values chosen are $20, 30, 40, 50, 60$. The k values obtained from the cross validation results are shown in Table 3.5. The k value for k-NNC is 3 and for wk-NLC from the cross-validation results are $2, 2, 3, 2$ and 2 for the thresholds $20, 30, 40, 50$ and 60 respectively. The comparison of different classifier results is summarized in Table 3.6.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. With $\tau = 30$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 2% . With $\tau = 30$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

Table 3.2: Synthetic Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	4692	85.17
k-NNC	Nil				74	7884	89.61
1-NLC	0.06	7947		4.75	1	36.97	73.93
	0.05	10327		6.81	1	53.74	75.52
	0.04	13949		11.98	1	73.9	77.35
	0.03	20164		22.53	1	105.2	79.80
	0.02	31743		46.41	1	151.15	82.43
k-NLC	0.06	7947		4.75	25	152.57	87.42
	0.05	10327		6.81	25	184.71	87.65
	0.04	13949		11.98	25	250.26	88.48
	0.03	20164		22.53	25	357.73	88.92
	0.02	31743		46.4	25	561.84	89.24
wk-NLC	0.06	7947		4.95	25	182.23	89.55
	0.05	10327		7.35	25	240.16	89.56
	0.04	13949		12.50	25	322.75	89.57
	0.03	20164		23.07	25	444.96	89.60
	0.02	31743		47.33	25	714.25	89.50
wk-NLC with noise elimination	0.06	5147	2800	10.61	25	112.97	89.56
	0.05	7067	3260	19.61	25	159.35	89.59
	0.04	10123	3826	37.99	25	233.45	89.62
	0.03	15789	4375	74.95	25	364.46	89.63
	0.02	26784	4959	163.98	25	607.32	89.56

3.6.4 SensIT Vehicle(acoustic) Data Set

For *acoustic* data set the τ values chosen are $0.4, 0.5, 0.6, 0.7, 0.8$. The k values obtained from the cross validation results are shown in Table 3.7. The k value for k-NNC is 21 and for wk-NLC from the cross-validation results are $23, 23, 20, 19$ and 37 for the thresholds $0.4, 0.5, 0.6, 0.7$ and 0.8 respectively. The comparison of different classifier results is summarized in Table 3.8.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. With $\tau = 0.4$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 0.3% . With $\tau = 0.4$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

Table 3.3: Covtype data set(Cross Validation results)

k	wk-NLC for different threshold values				
	0.3	0.25	0.2	0.15	0.1
1	69.51(0.34)	73.26(0.34)	77.93(0.03)	84.02(0.16)	89.25(0.08)
2	69.09(0.45)	72.51(0.15)	76.91(0.01)	82.59(0.07)	**89.69(0.07)**
3	70.16(0.30)	73.73(0.16)	77.93(0.08)	**83.32(0.10)**	88.74(0.04)
4	70.95(0.26)	74.26(0.14)	78.26(0.21)	83.27(0.02)	88.59(0.10)
5	71.37(0.27)	74.62(0.33)	**78.43(0.13)**	83.20(0.06)	88.34(0.12)
6	71.55(0.34)	74.72(0.32)	78.37(0.14)	83.11(0.07)	88.01(0.06)
7	71.76(0.26)	**74.81(0.34)**	78.38(0.15)	82.89(0.08)	87.77(0.10)
8	71.96(0.20)	74.68(0.34)	78.36(0.13)	82.69(0.04)	87.45(0.07)
9	72.04(0.23)	74.52(0.22)	78.21(0.14)	82.51(0.04)	87.18(0.07)
10	71.96(0.20)	74.42(0.26)	78.09(0.11)	82.36(0.04)	86.94(0.12)
11	**71.97(0.16)**	74.41(0.25)	77.91(0.12)	82.18(0.04)	86.70(0.06)
12	71.85(0.20)	74.33(0.25)	77.80(0.14)	81.96(0.02)	86.49(0.09)
13	71.77(0.23)	74.25(0.28)	77.64(0.15)	81.75(0.01)	86.26(0.06)
14	71.68(0.28)	74.13(0.31)	77.54(0.19)	81.57(0.03)	86.06(0.06)
15	71.62(0.27)	74.02(0.39)	77.38(0.13)	81.39(0.06)	85.89(0.07)

3.6.5 SensIT Vehicle(seismic) Data Set

For *seismic* data set the τ values chosen are $0.3, 0.4, 0.5, 0.6, 0.7$. The k values obtained from the cross validation results are shown in Table 3.9. The k value for k-NNC is 30 and for wk-NLC from the cross-validation results are $42, 45, 47, 47$ and 43 for the thresholds $0.3, 0.4, 0.5, 0.6$ and 0.7 respectively. The comparison of different classifier results is summarized in Table 3.10.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. With $\tau = 0.3$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 0.7% . With $\tau = 0.3$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

3.6.6 SensIT Vehicle(combined) Data Set

For *combined* data set the τ values chosen are $0.6, 0.7, 0.8, 0.9, 1.0$. The k values obtained from the cross validation results are shown in Table 3.11. The k value for k-NNC is 13 and for wk-NLC from the cross-validation results are $24, 35, 42, 42$ and 42 for the

Table 3.4: Covtype Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	1373066	94.89
1-NLC	0.3	4129		90.41	1	538.15	68.81
	0.25	6775		168.65	1	880.93	72.76
	0.2	12385		380.57	1	1577.86	77.36
	0.15	25746		1050.15	1	4137.72	82.78
	0.1	63433		3395.42	1	11927.2	88.72
k-NLC	0.3	4129		90.41	23	765.46	70.28
	0.25	6775		168.65	25	1311.98	73.28
	0.2	12385		380.57	10	2177.86	77.87
	0.15	25746		1050.15	7	4623.61	83.58
	0.1	63433		3395.42	3	12527.4	89.62
wk-NLC	0.3	4129		92.79	11	674.90	71.95
	0.25	6775		171.56	7	1011.11	74.75
	0.2	12385		385.67	5	1744.37	78.49
	0.15	25746		1059.80	3	4453.59	84.06
	0.1	63433		3415.32	2	12154.6	90.46
wk-NLC with noise elimination	0.3	3753	376	103.3	11	627.9	72.02
	0.25	6293	482	235.45	7	965.6	74.95
	0.2	11772	613	565.63	5	1684.28	79.14
	0.15	24844	902	1210.9	3	4086.28	84.98
	0.1	61877	1556	3812.85	2	11657.66	91.65

thresholds $0.6, 0.7, 0.8, 0.9$ and 1.0 respectively. The comparison of different classifier results is summarized in Table 3.12.

From the results it can be seen that the leader based classifies are considerably faster than NNC and k-NNC. With $\tau = 0.6$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 0.7% . With $\tau = 0.6$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

3.6.7 Ijcnn1 Data Set

For *Ijcnn1* data set the τ values chosen are $0.1, 0.2, 0.3, 0.4, 0.5$. The k values obtained from the cross validation results are shown in Table 3.13. The k value for k-NNC is 1 and for wk-NLC from the cross-validation results are $1, 5, 5, 4$ and 7 for the thresholds $0.1, 0.2, 0.3, 0.4$ and 0.5 respectively. The comparison of different classifier results is summarized in Table 3.14.

From the results it can be seen that the leader based classifies are considerably faster

Table 3.5: Pendigits data set (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		60	50	40	30	20
1	99.21(0.05)	94.54(0.46)	96.80(0.22)	97.13(0.23)	98.65(0.13)	99.07(0.07)
2	99.09(0.08)	**94.65(0.45)**	**96.92(0.31)**	97.40(0.21)	**98.95(0.17)**	**99.31(0.08)**
3	**99.31(0.10)**	94.55(0.13)	96.37(0.08)	**97.68(0.13)**	98.61(0.07)	99.15(0.15)
4	99.11(0.14)	94.37(0.07)	95.96(0.06)	97.13(0.21)	98.31(0.28)	98.95(0.11)
5	99.09(0.15)	93.14(0.29)	95.40(0.23)	97.01(0.36)	98.13(0.18)	98.89(0.25)
6	99.08(0.13)	92.09(0.39)	94.81(0.33)	96.78(0.11)	98.00(0.20)	98.77(0.25)
7	98.92(0.18)	91.25(0.35)	94.18(0.36)	96.36(0.11)	97.80(0.15)	98.68(0.29)
8	98.84(0.25)	90.27(0.24)	93.50(0.59)	96.18(0.17)	97.61(0.11)	98.48(0.23)
9	98.76(0.24)	89.19(0.21)	93.18(0.48)	95.84(0.26)	97.34(0.16)	98.33(0.22)
10	98.65(0.30)	88.07(0.36)	92.44(0.68)	95.53(0.27)	97.08(0.18)	98.32(0.28)

than NNC and k-NNC. With $\tau = 0.1$, the CA of wk-NLC is almost similar to the CA of k-NNC but with much reduced classification time. The classification time of wk-NLC when compared with that of k-NNC and NNC are less than 0.9% . With $\tau = 0.1$, wk-NLC with noise elimination shows improvement with respect to classification accuracy over wk-NLC without noise elimination.

3.7 Conclusions

In this paper an improvement over using leaders as prototypes is given. For each of the leaders an associated weight is found which gives its relative importance. These weights are used in the k nearest leader based classifier called weighted k nearest leader classifier. A preprocessing step to eliminate noisy prototypes is also presented. The proposed method is experimentally compared with nearest neighbor classifier(NNC), k nearest neighbor classifier(k-NNC), nearest leader classifier and k nearest leader classifier. With suitable parameters, the proposed method can achieve classification accuracy similar to k-NNC and is superior than the other methods, but is a much fast classifier than k-NNC or NNC. Hence the proposed method is a suitable one to be used with large data sets as in data mining applications.

Table 3.6: Pendigits Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time	k	classification time	CA(%)
1-NNC	Nil				1	302.6	97.74
k-NNC	Nil				3	320.8	97.83
1-NLC	60	385		0.10	1	0.34	91.76
	50	631		0.11	1	0.53	94.19
	40	1165		0.13	1	0.94	95.85
	30	2306		0.19	1	1.78	97.17
	20	4821		0.40	1	4.70	97.48
k-NLC	60	385		0.10	4	0.40	92.28
	50	631		0.11	4	0.63	94.72
	40	1165		0.13	5	1.19	95.11
	30	2306		0.19	3	2.16	96.04
	20	4821		0.40	1	4.70	97.48
wk-NLC	60	385		0.10	2	0.37	93.39
	50	631		0.11	2	0.58	95.48
	40	1165		0.13	3	1.13	95.19
	30	2306		0.19	2	2.20	96.68
	20	4821		0.40	2	5.48	97.57
wk-NLC with noise elimination	60	363	22	0.10	2	0.37	93.37
	50	606	25	0.13	2	0.57	95.43
	40	1133	32	0.21	3	1.09	95.23
	30	2260	46	0.47	2	2.11	96.71
	20	4753	68	1.48	2	5.43	97.57

Table 3.7: SensIT Vehicle(acoustic) (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		0.8	0.7	0.6	0.5	0.4
1	63.78(0.11)	44.17(0.85)	44.90(0.71)	46.92(1.35)	48.45(0.40)	51.45(0.30)
10	70.49(0.15)	61.01(0.45)	61.91(0.36)	62.74(0.42)	63.90(0.30)	65.35(0.25)
15	71.06(0.02)	61.20(0.24)	62.36(0.46)	63.15(0.45)	64.21(0.05)	65.75(0.44)
16	71.12(0.08)	61.19(0.33)	62.36(0.47)	63.11(0.42)	64.20(0.05)	65.74(0.48)
17	71.17(0.04)	61.18(0.34)	62.37(0.46)	63.16(0.50)	64.24(0.07)	65.72(0.43)
18	71.09(0.06)	61.23(0.36)	62.35(0.43)	63.14(0.56)	64.29(0.04)	65.73(0.45)
19	71.30(0.11)	61.28(0.39)	**62.39(0.38)**	63.10(0.59)	64.31(0.02)	65.76(0.38)
20	71.19(0.17)	61.24(0.37)	62.38(0.38)	**63.21(0.61)**	64.26(0.06)	65.79(0.35)
21	**71.33(0.18)**	61.22(0.42)	62.38(0.41)	63.19(0.59)	64.33(0.03)	65.75(0.41)
22	71.29(0.12)	61.20(0.44)	62.33(0.40)	63.15(0.55)	64.30(0.03)	65.84(0.47)
23	71.29(0.08)	61.23(0.43)	62.34(0.46)	63.12(0.52)	**64.34(0.04)**	**65.85(0.40)**
24	71.25(0.08)	61.22(0.43)	62.34(0.42)	63.11(0.51)	64.26(0.05)	65.79(0.33)
25	71.31(0.08)	61.25(0.49)	62.29(0.42)	63.11(0.52)	64.25(0.07)	65.76(0.31)
30	71.21(0.05)	61.41(0.52)	62.19(0.45)	63.04(0.48)	64.21(0.14)	65.57(0.23)
35	71.21(0.02)	61.46(0.51)	62.10(0.45)	63.02(0.52)	64.13(0.12)	65.44(0.15)
36	71.23(0.05)	61.46(0.54)	62.12(0.48)	63.05(0.49)	64.16(0.16)	65.43(0.15)
37	71.24(0.03)	**61.47(0.54)**	62.17(0.51)	63.05(0.48)	64.17(0.10)	65.37(0.17)
38	71.25(0.12)	61.45(0.52)	62.13(0.56)	63.02(0.51)	64.12(0.12)	65.38(0.13)
39	71.20(0.10)	61.43(0.51)	62.07(0.56)	63.04(0.46)	64.09(0.11)	65.34(0.12)
40	71.22(0.14)	61.42(0.47)	62.08(0.55)	63.02(0.45)	64.05(0.18)	65.34(0.14)

Table 3.8: SensIT Vehicle(acoustic) Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time(s)	k	classification time(s)	CA(%)
1-NNC	Nil				1	125935.77	65.4149
k-NNC	Nil				21	158967.30	75.9857
1-NLC	0.8	1365		6.85	1	13.97	57.5082
	0.7	2631		8.01	1	27.59	61.355
	0.6	5397		12.65	1	56.57	63.0506
	0.5	11168		29.72	1	120.42	65.1725
	0.4	20793		76.56	1	255.51	65.524
k-NLC	0.8	1365		6.85	37	32.11	59.6025
	0.7	2631		8.01	19	50.76	61.953
	0.6	5397		12.65	20	105.38	63.8382
	0.5	11168		29.72	23	225.76	66.836
	0.4	20793		76.56	23	403.48	67.283
wk-NLC	0.8	1365		6.96	37	33.11	61.1926
	0.7	2631		8.22	19	52.48	62.7709
	0.6	5397		12.96	20	109.45	64.9056
	0.5	11168		30.82	23	233.88	67.592
	0.4	20793		78.64	23	428.94	69.495
wk-NLC with noise elimination	0.8	1316	49	7.63	37	31.64	61.2078
	0.7	2581	50	10.99	19	50.71	62.7759
	0.6	5346	51	26.91	20	108.52	64.9802
	0.5	11113	55	93.99	23	232.81	67.602
	0.4	20732	61	290.19	23	427.82	69.50

Table 3.9: SensIT Vehicle(seismic) (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		0.7	0.6	0.5	0.4	0.3
1	65.23(0.29)	44.20(0.93)	45.35(1.08)	50.26(0.02)	56.20(0.45)	59.55(0.35)
10	72.00(0.03)	64.87(0.54)	65.56(0.15)	66.52(0.23)	68.17(0.19)	70.06(0.36)
20	72.73(0.17)	66.83(0.34)	67.53(0.67)	68.38(0.41)	69.50(0.19)	70.94(0.18)
25	72.74(0.04)	67.06(0.44)	67.93(0.56)	68.70(0.38)	69.69(0.16)	71.04(0.28)
26	72.74(0.07)	67.10(0.48)	68.02(0.54)	68.71(0.43)	69.65(0.19)	71.15(0.29)
27	72.73(0.05)	67.18(0.45)	68.06(0.55)	68.71(0.39)	69.79(0.17)	71.20(0.27)
28	72.84(0.08)	67.23(0.39)	68.21(0.55)	68.71(0.39)	69.88(0.25)	71.17(0.21)
29	72.87(0.10)	67.28(0.40)	68.25(0.59)	68.78(0.36)	69.88(0.23)	71.24(0.23)
30	**72.90(0.07)**	67.29(0.36)	68.28(0.53)	68.87(0.38)	69.86(0.30)	71.28(0.21)
40	72.79(0.12)	67.56(0.34)	68.39(0.46)	69.14(0.44)	70.19(0.20)	71.29(0.16)
41	72.78(0.15)	67.62(0.38)	68.37(0.50)	69.19(0.48)	70.23(0.21)	71.29(0.21)
42	72.78(0.09)	67.68(0.35)	68.39(0.52)	69.23(0.49)	70.24(0.20)	**71.35(0.17)**
43	72.77(0.14)	**67.78(0.33)**	68.44(0.53)	69.30(0.48)	70.25(0.22)	71.30(0.09)
44	72.78(0.08)	67.78(0.33)	68.46(0.51)	69.39(0.51)	70.25(0.20)	71.31(0.12)
45	72.77(0.22)	67.75(0.33)	68.51(0.45)	69.42(0.48)	**70.29(0.19)**	71.32(0.12)
46	72.76(0.18)	67.76(0.35)	68.52(0.44)	69.41(0.49)	70.28(0.19)	71.33(0.15)
47	72.76(0.22)	67.77(0.32)	**68.57(0.42)**	**69.45(0.47)**	70.27(0.16)	71.32(0.07)
48	72.75(0.14)	67.73(0.28)	68.56(0.43)	69.41(0.44)	70.24(0.22)	71.30(0.08)
49	72.74(0.22)	67.74(0.34)	68.52(0.47)	69.39(0.43)	70.21(0.17)	71.24(0.03)
50	72.75(0.16)	67.76(0.34)	68.54(0.46)	69.36(0.39)	70.24(0.18)	71.26(0.01)

Table 3.10: SensIT Vehicle(seismic) Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time(s)	k	classification time(s)	CA(%)
1-NNC	Nil				1	125935.77	65.4149
k-NNC	Nil				30	171502.65	73.2098
1-NLC	0.7	2643		8.02	1	27.94	61.355
	0.6	5192		12.98	1	58.8	64.450
	0.5	10879		36.87	1	121.28	65.272
	0.4	23494		133.65	1	264.78	65.824
	0.3	42765		369.76	1	402.67.	66.426
k-NLC	0.7	2643		8.02	43	74.36	63.482
	0.6	5192		12.98	47	150.94	65.263
	0.5	10879		36.87	47	311.26	67.924
	0.4	23494		133.65	45	645.78	68.293
	0.3	42765		369.76	42	1125.92	68.956
wk-NLC	0.7	2643		8.18	43	75.55	67.592
	0.6	5192		13.80	47	154.43	68.423
	0.5	10879		38.09	47	317.43	69.931
	0.4	23494		136.75	45	659.41	70.027
	0.3	42765		373.44	42	1146.80	71.306
wk-NLC with noise	0.7	2446	197	12.51	43	68.97	67.561
	0.6	4994	198	32.39	47	141.46	68.942
	0.5	10664	215	120.07	47	302.52	69.982
	0.4	23269	225	520.61	45	629.49	70.146
	0.3	42529	236	1576.76	42	1138.43	71.312

Table 3.11: SenseITvechicle(combined) (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		1.0	0.9	0.8	0.7	0.6
1	75.54(0.08)	57.16(1.05)	60.83(1.21)	63.39(0.18)	67.09(0.24)	70.50(0.18)
10	83.16(0,05)	75.16(0.68)	76.33(0.87)	78.34(0.26)	80.29(0.28)	81.64(0.30)
13	**83.48(0.08)**	76.39(0.79)	77.10(0.76)	79.14(0.32)	80.62(0.39)	81.90(0.32)
21	83.39(0.17)	77.73(0.78)	78.28(0.65)	79.67(0.49)	81.04(0.24)	82.09(0.38)
22	83.38(0.21)	77.73(0.78)	78.37(0.64)	79.78(0.56)	81.06(0.29)	82.11(0.35)
23	83.38(0.09)	77.81(0.79)	78.49(0.68)	79.73(0.61)	81.08(0.34)	82.14(0.38)
24	83.37(0.13)	77.90(0.82)	78.54(0.69)	79.74(0.57)	81.15(0.37)	**82.16(0.36)**
25	83.36(0.04)	77.90(0.84)	78.68(0.70)	79.78(0.51)	81.09(0.35)	82.15(0.38)
30	82.98(0.24)	78.27(0.82)	79.12(0.61)	79.94(0.53)	81.10(0.37)	82.11(0.35)
33	82.59(0.07)	78.35(0.75)	79.22(0.59)	79.90(0.60)	81.16(0.40)	82.08(0.33)
34	82.42(0.21)	78.39(0.77)	79.23(0.62)	79.89(0.60)	81.18(0.41)	82.05(0.37)
35	82.39(0.12)	78.45(0.75)	79.19(0.61)	79.92(0.60)	**81.19(0.36)**	82.07(0.32)
36	82.35(0.13)	78.45(0.71)	79.25(0.59)	79.87(0.58)	81.17(0.37)	82.08(0.30)
37	82.28(0.07)	78.43(0.70)	79.23(0.56)	79.89(0.54)	81.13(0.39)	82.06(0.31)
40	82.19(0.23)	78.41(0.60)	79.22(0.57)	80.01(0.51)	81.09(0.40)	82.08(0.33)
41	81.99(0.02)	78.42(0.57)	79.27(0.57)	80.07(0.53)	81.09(0.37)	82.06(0.38)
42	81.85(0.01)	**78.48(0.57)**	**79.28(0.51)**	**80.09(0.50)**	81.10(0.38)	82.00(0.37)
43	81.79(0.08)	78.45(0.59)	79.23(0.55)	80.06(0.48)	81.08(0.37)	81.99(0.37)
44	81.49(0.17)	78.46(0.58)	79.21(0.52)	80.00(0.49)	81.08(0.40)	82.00(0.35)
45	81.39(0.14)	78.43(0.59)	79.19(0.52)	80.03(0.51)	81.10(0.37)	81.97(0.35)

Table 3.12: SensIT Vehicle(combined) Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time(s)	k	classification time(s)	CA(%)
1-NNC	Nil				1	162658.25	76.23
k-NNC	Nil					194826.56	84.16
1-NLC	1.0	5641		25.98	1	116.85	74.041
	0.9	9591		47.02	1	260.28	74.536
	0.8	16484		103.26	1	488.29	74.983
	0.7	27896		241.48	1	756.46	75.224
	0.6	43687		517.96	1	980.65	75.544
k-NLC	1.0	5641		25.98	42	195.86	74.863
	0.9	9591		47.02	42	338.42	75.392
	0.8	16484		103.26	42	586.78	76.026
	0.7	27896		241.48	35	931.62	76.973
	0.6	43687		517.96	24	1279.48	77.453
wk-NLC	1.0	5641		26.83	42	209.93	79.025
	0.9	9591		48.19	42	352.70	79.523
	0.8	16484		105.22	42	600.28	80.208
	0.7	27896		244.34	35	940.54	81.218
	0.6	43687		522.35	24	1296.64	82.5425
wk-NLC with noise	1.0	5500	141	55.22	42	203.66	79.234
	0.9	9448	143	149.34	42	342.64	79.634
	0.8	16339	145	380.34	42	586.39	80.312
	0.7	27748	148	820.56	35	932.36	81.263
	0.6	43533	154	2142.40	24	1275.76	82.638

Table 3.13: Ijcnn1 (Cross Validation results)

k	k-NNC	wk-NLC for different threshold values				
		0.5	0.4	0.3	0.2	0.1
1	**97.39(0.19)**	82.33(0.57)	84.12(1.49)	85.55(0.67)	92.54(0.14)	**97.20(0.20)**
2	96.90(0.17)	88.86(0.61)	90.62(0.16)	89.29(1.21)	92.79(0.18)	96.38(0.27)
3	97.18(0.08)	90.25(0.11)	91.81(0.43)	91.69(0.48)	93.93(0.36)	96.49(0.15)
4	96.87(0.06)	90.81(0.21)	**91.96(0.45)**	92.30(0.26)	94.00(0.22)	96.02(0.04)
5	96.84(0.11)	91.10(0.15)	91.93(0.31)	**92.44(0.31)**	**94.05(0.22)**	95.81(0.05)
6	96.55(0.11)	91.18(0.17)	91.85(0.25)	92.31(0.24)	93.77(0.25)	95.39(0.14)
7	96.46(0.13)	**91.24(0.21)**	91.71(0.22)	92.20(0.25)	93.59(0.28)	95.10(0.21)
8	96.22(0.18)	91.17(0.17)	91.60(0.21)	92.12(0.23)	93.33(0.25)	94.84(0.15)
9	96.13(0.17)	91.11(0.16)	91.50(0.21)	92.00(0.24)	93.17(0.29)	94.58(0.11)
10	95.98(0.16)	91.04(0.15)	91.37(0.19)	91.95(0.23)	92.95(0.22)	94.44(0.11)

Table 3.14: Ijcnn1 Data Set

Classifier	Threshold	#Leaders	#Noisy leader	Design time(s)	k	classification time(s)	CA(%)
1-NNC	Nil				1	96038.45	97.61
1-NLC	0.5	1869		2.25	1	44.97	89.683
	0.4	3632		4.96	1	102.32	89.852
	0.3	6747		13.98	1	198.24	91.231
	0.2	13073		42.39	1	430.28	92.553
	0.1	27191		135.76	1	818.58	97.076
k-NLC	0.5	1869		2.25	7	68.70	89.937
	0.4	3632		4.96	4	129.68	90.332
	0.3	6747		13.98	5	257.83	90.991
	0.2	13073		42.39	5	485.36	92.965
	0.1	27191		135.76	1	818.58	97.072
wk-NLC	0.5	1869		2.49	7	69.30	91.464
	0.4	3632		5.26	4	131.7	91.785
	0.3	6747		15.01	5	261.61	92.512
	0.2	13073		44.50	5	494.11	93.720
	0.1	27191		137.90	1	828.51	97.076
wk-NLC with noise	0.5	1825	44	3.57	7	68.65	91.471
	0.4	3605	27	9.91	4	128.68	91.824
	0.3	6731	16	34.91	5	258.31	92.563
	0.2	13061	12	123.53	5	490.96	93.853
	0.1	27181	10	496.65	1	817.28	97.169

Chapter 4

Rough-Fuzzy Weighted k-Nearest Leader Classifier

4.1 Introduction

The set of leaders derived (and their weights, in case weighted leaders are derived) depends on the scanning order of the data set. Apart from this, a pattern can be at less than the threshold τ distance from more than one leader, but the leaders clustering method assigns the pattern to only one leader (which is encountered first in the data set). A pattern can belong to more than one leader, but with varying degrees. The chapter proposes to use rough-fuzzy membership values while employing the leaders clustering method. For this the chapter uses rough set theory [Pawlak (1982)] principles and fuzzy set theory [Zadeh (1965)] principles which are related and complementary with each other [Chanas and Kuchta (1992)] [Dubois and Prade (1990)], [Y. Y. Yao (1998)]. Combined principles of rough sets and fuzzy sets are more prominent which resolves impression, uncertainty and vagueness in a better way than either of the theories alone [Dubois and Prade (1990)].

The rough set theory deals with two approximations of an arbitrary subset of a universe called lower and upper approximations. By using these two approximations knowledge hidden in information systems may be explored and expressed in the form of decision rules [Jagielska et al. (1999)], [Kryszkiewicz (1998)], [Pawlak (1991)], [Tsumoto (1998)].

Rough set theory is widely used in feature selection methods where it is possible to find a subset of the original features that are the most informative [Shiu et al. (2006)], [Li et al. (2006)], [Swiniarski and Skowron (2003)]. Rough-fuzzy set theory is applied in many other pattern recognition techniques like decision rules [Greco et al. (2006)], [Greco et al. (2001)], classification rules [Shen and Chouchoulas (2002)], case generation [Pal and Mitra (2004)], document categorization [Singh and Dey (2005a)], [V. Suresh Babu and Viswanath (2006)], information retrieval [Singh and Dey (2005b)] *etc,*. In recent years, rough and fuzzy set principles are applied for soft clustering [Karray and C. De Silva (2004)]. This is in contrast to hard clustering or hard partitioning where a pattern belongs exactly to one cluster only, but in soft clustering a pattern can belong to more than one cluster. Particularly rough set theory is applied to clustering methods in [De (2004)], [Mitra et al. (2006)], [Asharaf et al. (2005)] and rough clustering methods are introduced by Lingras *et al.* [Lingras and West (2004)], do Prado *et al.* [H. A. do Prado et al. (2002)] and Voges *et al.* [Voges et al. (2002)], [Voges et al. (2003)]. Pradipta Maji *et. al.* [Maji and Pal (2007)] proposed rough-fuzzy c-medoids algorithm and it is used for the selection of bio-basis for amino acid sequence analysis. Lingras *et. al.* [Lingras and West (2004)] proposed a rough k-means clustering method and later it is refined by Georg Peters [Peters (2006)] which is applied successfully to web mining. Manish Sarkar [Sarkar (2007)] and H.Bian *et. al.* [Bian and Mazlack (2003)] proposed fuzzy-rough nearest neighbor classification methods which uses fuzzy-rough principles to the conventional and fuzzy k-nearest neighbor classifiers [Keller et al. (1985)] and hence the performance is improved. But, these are not scalable methods for large data sets which considers all training patterns as the neighbors with a different degree (the degree depends on the rough-fuzzy membership values) to classify a test pattern.

Asharaf *et. al.* [Asharaf and Murty (2003)], [Asharaf and Murthy (2004)] proposed the *Adaptive rough fuzzy leaders* method which uses rough-fuzzy set theory in the cluster analysis to resolve the uncertainty. It is applied in classification by deriving enriched prototypes and also applied successfully to clustering (web usage categorization) in a single scan of the data set. It is argued that, since it uses the enriched prototypes in

the classification, the performance of the classifier (NNC which uses these prototypes) is improved. Nevertheless, it does not preserve the density information, and it has $O(n. \mid \mathcal{L} \mid^2)$ time complexity in calculating rough-fuzzy membership values of all patterns where $\mid \mathcal{L} \mid$ is the number of leaders derived using the adaptive rough fuzzy leaders method. Hence, It is a slower one than the one which is proposed in this chapter which has $O(n)$ time complexity.

This chapter attempts at presenting a generalized leaders clustering method for prototype selection. The chapter proposes a scalable and improved rough-fuzzy leaders clustering method which has less computation burden to calculate rough-fuzzy membership values. It proposes to use rough-fuzzy membership values not only to resolve uncertainty while assigning patterns to the clusters but also uses to preserve density information, *i.e.,* rough-fuzzy membership value is used for weight calculation. Further, the chapter proposes two level hierarchical clustering method for prototype selection in a single scan of the data set which is a generalized version of the *Leaders-subleaders* method proposed by P. A. Vijaya *et al.* [Vijaya et al. (2004)]. A k-nearest leader classifier is analyzed using rough set theory and finally the method is empirically verified using some standard data sets and a comparison is drawn with some of the earlier related methods.

The chapter is organized as follows. Section 4.2 explains rough set theory and fuzzy set theory followed by rough-fuzzy leaders clustering method. Section 4.3 explains rough-fuzzy weighted leaders method for prototype selection followed by rough-fuzzy weighted k-nearest leader classifier (RF-wk-NLC). Section 4.4 describes rough-fuzzy weighted leaders-subleaders method followed by RF-wk-NLC which uses rough-fuzzy weighted leaders-subleaders method. Section 4.5 describes the experimental results for various data sets.

4.2 Rough set theory and fuzzy set theory

Rough sets theory [Pawlak (1982)] and fuzzy set theory [Zadeh (1965)] are well known mathematical theories to capture uncertainties associated with the data. These two theories model different types of uncertainty. The rough set theory considers the indiscernibility of elements which is typically characterized by an equivalence relation. Whereas

fuzzy set theory deals with the generalization of characteristic function and it does not use indiscernibility of elements.

Let U be the finite non empty set called universe and let $\mathcal{R}$ be the equivalence relation on U then the notion of approximation space is defined as $< U, \mathcal{R} >$. The relation $\mathcal{R}$ (since it is equivalence relation) partition the universe U into equivalence classes. Suppose $\mathcal{R}$ partitions U into p equivalence classes and are denoted by $U/\mathcal{R}$.

$$U/\mathcal{R} = \{X_1, X_2, \ldots, X_p\} \tag{4.1}$$

where X_i, for $i = 1$ to p are equivalence classes of $\mathcal{R}$. Now, given an arbitrary set $X \in U$, it may be difficult to characterize X explicitly, but, X can be characterized by a pair of lower and upper approximations as

$$\underline{X}_{\mathcal{R}} = \bigcup_{X_i \subseteq X} X_i \text{ and } \overline{X}_{\mathcal{R}} = \bigcup_{X_i \cap X \neq \phi} X_i \tag{4.2}$$

Where fuzzy set theory is characterized by a generalized fuzzy membership function. A fuzzy set F is defined by a pair $< U, \mu_F >$ where, $\mu_F : U \rightarrow [0, 1]$. Here $\mu_F(x)$ is the fuzzy membership of x.

4.2.1 Properties of rough-fuzzy set theory

Some of the important properties of rough set theory and fuzzy set theory are described which are useful to resolve uncertainties that is present in the conventional clustering methods. Figure 4.1 explains the rough-fuzzy set theory used in the clustering methods.

- $\underline{X}_{\mathcal{R}} \subseteq X \subseteq \overline{X}_{\mathcal{R}}$ implies that, for an element $x \in X$, if $x \in \underline{X}_{\mathcal{R}}$ then $x \in \overline{X}_{\mathcal{R}}$.
- If X, Y are two disjoint sets in U then $\underline{X}_{\mathcal{R}}, \underline{Y}_{\mathcal{R}}$ are also disjoint sets and it is not guaranteed that $\overline{X}_{\mathcal{R}}, \overline{Y}_{\mathcal{R}}$ are also disjoint sets. It implies that an element can be a member of at most one lower approximation or it can be a member of one or more upper approximations.
- If $X \in U$ is said to be definable (or exact) if and only if $\underline{X}_{\mathcal{R}} = \overline{X}_{\mathcal{R}}$

- $\mu_F(x)$ is the degree to which extent it is part of the set. If $\mu_F(x) = 1$, then x is definitely in the set. If $\mu_F(x) = 0$, then x is definitely not in the set.

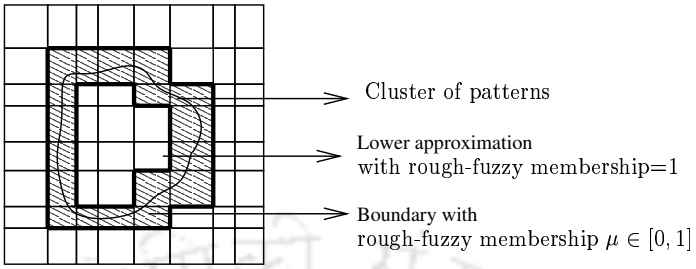


Figure 4.1: Rough-fuzzy set theory for clustering methods

4.2.2 Rough fuzzy principles for leaders clustering method

In order to reduce the computational burden of the *adaptive rough fuzzy leaders* method proposed by Asharaf *et. al.* [Asharaf and Murty (2003)], we present some refinements to use rough fuzzy principles in the assignment of patterns to the prototypes (*i.e.,* leaders). A user defined upper threshold (U_T) and lower threshold (L_T) are used which are said to be the upper and lower approximations of the prototypes such that $L\_T \leq U\_T$. Hence, patterns can be assigned to the prototypes as follows.

- If pattern x falls within the lower threshold of a prototype then it is assigned without any ambiguity to the respective prototype. Note that even if there are more than one such prototype then the pattern x is assigned to any one of them.

- If there is no such prototype (pattern x is not in the lower threshold of any prototype) then the method checks if pattern x is in the upper threshold of one or more prototype(s) then it is assigned to each of these prototype(s) with some rough-fuzzy membership value(s) which will be calculated based on the proximity of the pattern x with respect to these prototype(s).

- If pattern x is not within the upper approximation of a prototype then it does not belongs to that prototype.

Rough-fuzzy membership values of a pattern which is in the boundary of one or more prototypes (if there are more than one such prototype then we call them as *overlapping prototypes*) is calculated based on it's proximity with respect to the overlapping prototypes. Suppose there are r overlapping prototypes for a pattern x, say $l^{(1)}, l^{(2)}, \ldots, l^{(r)}$, then the rough-fuzzy membership of x to each of these overlapping prototypes are

$$\mu_j = \frac{\beta_j}{\sum_{p=1}^{r} \beta_p}, \ j = 1, \ldots, r \tag{4.3}$$

where

$$\beta_j = \left(\sum_{p=1}^{r} \left(\frac{||l_j - x||}{||l_p - x||} \right) \right)^{-1} \tag{4.4}$$

From equation (4.3) we have

$$\sum_{p=1}^{r} \mu_p = 1 \tag{4.5}$$

An example for these kind of assignments is given in Figure 4.2. x_1 belongs to the lower

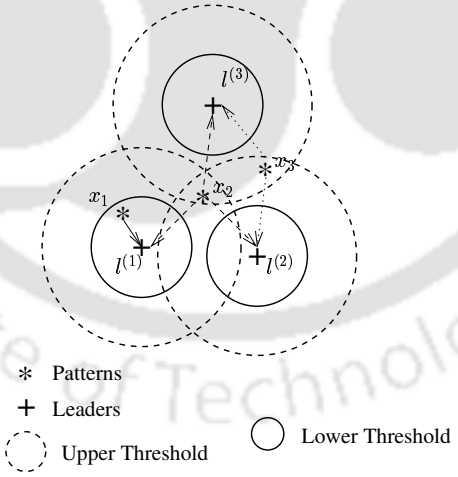


Figure 4.2: Rough fuzzy assignment of patterns to the prototypes

threshold of prototype $l^{(1)}$ then x_1 is assigned to $l^{(1)}$ only. But, x_2 is in the boundary of three prototypes *viz.*, $l^{(1)}$, $l^{(2)}$, and $l^{(3)}$. Hence x_2 is assigned to each of these prototypes with some rough-fuzzy membership values. Similarly x_3 is not in the upper threshold of

$l^{(1)}$ and hence x_3 is not assigned to $l^{(1)}$.

4.3 Rough-fuzzy weighted k-nearest leader classifier

This section describes rough-fuzzy weighted k-nearest leader classifier (RF-wk-NLC) which uses rough fuzzy weighted leaders as prototypes. Firstly, rough-fuzzy weighted leaders for prototype selection is described and is followed by the RF-wk-NLC.

4.3.1 Rough-fuzzy weighted leaders method

A modified leaders clustering method using rough-fuzzy set theory called *Rough-Fuzzy Weighted-Leaders* method is described in Algorithm 4.1.

Algorithm 4.1 Rough-Fuzzy Weighted-Leaders($\mathcal{D}_i$, U_T)

$L_i = \emptyset$;
$L\_T = U\_T/2$;
for each $x \in \mathcal{D}_i$ **do**
  if there is a leader $l \in \mathcal{L}_i$ such that $||l - x|| \leq L\_T$ **then**
    $weight(l) = weight(l) + \frac{1}{n_i}$;
  else
    Find the set $P = \{l \mid l \in L_i, ||l - x|| \leq U\_T\}$;
    if $P \neq \phi$ **then**
      for each l such that $l \in P$ **do**
        { Let μ is a rough-fuzzy membership value of l using rough-fuzzy set theory described in section 4.2.2}
        $weight(l) = weight(l) + \frac{\mu}{n_i}$;
      end for
    else
      $L_i = L_i \cup \{x\}$;
      $weight(x) = \frac{1}{n_i}$;
    end if
  end if
end for
Output L_i which is a set of tuples such that each tuple is in the form $< l,\ weight(l) >$ where l is a leader and $weight(l)$ is its weight.

The method uses a pair of upper (U_T) and lower(L_T) thresholds which are said to be upper and lower approximations of the prototypes. The proposed rough-fuzzy

weighted leaders method uses *L_T=U_T/2*. The reasons being, (i) the method can be specified by only one parameter (*i.e., U_T*), (ii) patterns which belong to lower threshold of two distinct leaders will not intersect their regions (in the feature space). It is an incremental clustering method which requires a single scan of the data set and has $O(n)$ time complexity where n is the data set size. For a given upper threshold (*U_T*), rough-fuzzy weighted leaders clustering method is devised to work for each class of patterns separately, but the same upper threshold parameter *i.e., U_T* is used and the rough-fuzzy weighted leaders method is as follows. It maintains a set of leaders $\mathcal{L}_i$ for each class of patterns $\mathcal{D}_i$ which is initially empty. For each pattern $x \in \mathcal{D}_i$, if there is a leader $l \in \mathcal{L}_i$ such that distance between x and l is less than *L_T* then x is assigned to the lower approximation of l and the weight of l is updated as *weight(l)=weight(l)+*$\frac{1}{n_i}$. If there is no such leader exist then pattern x is verified with the p leaders for which distance between x and l is less than *U_T* and if p is greater than 0 then x is assigned to each of the p leaders with some rough-fuzzy membership value which is described in section 4.2.2. Suppose x is assigned to leader l with rough-fuzzy membership value μ then the weight of leader l is updated as *weight(l)=weight(l)+*$\frac{\mu}{n_i}$. If $p = 0$ then x itself becomes a new leader whose weight is initialized as $\frac{1}{n_i}$. The rough-fuzzy leaders method is described in Algorithm 4.1.

4.3.2 Rough fuzzy weighted k-nearest leader classifier

This section describes the proposed classifier. Let L_i be the set of rough fuzzy weighted leaders obtained for the patterns of the class ω_i, for $i = 1$ to c. Let L be the set of all leaders. That is, $L = L_1 \cup \ldots \cup L_c$. The rough-fuzzy weighted k-nearest classifier is given in Algorithm (4.5). The space and time complexities of *RF-wk-NLC* is $O(|\mathcal{L}|)$ where $|\mathcal{L}|$ is the number of all leaders

Algorithm 4.2 Rough-Fuzzy-weighted-k-Nearest-Leader(L, q)

{L is the set of all leaders derived from all classes. q is the query pattern to be classified}
Find k nearest leaders of q from L.
Among the k nearest leaders find the cumulative weight of leaders that belongs to each class. Let this be W_i for class ω_i, for $i = 1$ to c.
Class label assigned for $q = argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$.

As the τ or U_T value increases, the number of leaders decreases which reduces the classification time and the performance of the classifier. In otherhand, if the τ or U_T value decreases, then the number of leaders increases which causes in increasing the classification time and the performance. In the following section describes a method to tackle this situation and it is analyzed using rough set theory.

4.4 RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method

An extended version of rough-fuzzy weighted leaders method called rough-fuzzy weighted leaders-subleaders method is described which explorate the rough-fuzzy uncertainty present in the leaders-subleaders method proposed by P. A. Vijaya *et.al* [Vijaya et al. (2004)]. Later, *RF-wk-NLC* is described which uses the rough-fuzzy weighted leaders-subleaders method for classification.

4.4.1 Rough-fuzzy weighted leaders-subleaders method

P. A. Vijaya *et. el.* [Vijaya et al. (2004)] proposed a hierarchy of leaders where first level leaders are derived with larger threshold value which are meant for reducing the search-space and later second level leaders with smaller threshold value are derived which are meant for doing the classification. Nevertheless, the method do not preserve the density information and it is a 1-nearest leader based classifier. It derives leaders-subleaders in two scans of the data set.

Rough-fuzzy weighted leaders-subleaders method [V. Suresh Babu and Viswanath] is an incremental method which derives a hierarchy of leaders-subleaders in a single scan of the data set. The proposed method resolves rough-fuzzy uncertainty while assigning patterns to clusters and uses rough-fuzzy membership values to preserve the density information. (*i.e.,* rough-fuzzy membership values are used to update the weights of leaders). Whereas *Leaders-subleaders* method has uncertainty while assigning patterns to clusters (if a pattern within a threshold distance of two or more leaders then the method chooses

any one out of them) and the method looses the density information while deriving prototypes. On the otherhand, *adaptive rough-fuzzy leaders* method uses rough-fuzzy principles to resolve uncertainty in the cluster analysis, but it has huge computational burden and relative importance of prototypes are not used. The proposed method refines rough-fuzzy principles to leaders clustering method and hence it calculates rough-fuzzy membership values for a few of the patterns with respect to a few of the prototypes. Hence, it has reduced computational burden and also it uses leaders-subleaders are representatives along with their relative importance whereas *adaptive rough-fuzzy leaders* method has only leaders as representatives.

Rough-fuzzy weighted leaders-subleaders method requires two threshold values called upper threshold (U_T) and upper subthreshold (U_ST) where upper subthreshold is less than the upper threshold. Similarly it requires lower threshold (L_T) and lower subthreshold (L_ST) which are assumed as exactly half of the upper threshold and upper subthreshold, respectively. Hence the method can be specified by two parameters only. For each pattern $x \in \mathcal{D}_i$, the rough fuzzy weighted leaders-subleaders method works as follows

1. If x is within L_T of any leader then its weight is updated by $\frac{1}{n_i}$ and the same pattern x is verified to see if it is within the L_ST of any sub-leader $sl \in \mathcal{SL}_{ij}$ (suppose x is assigned to j^{th} leader) then the weight of the sub-leader sl is updated by $\frac{1}{n_i}$, else if x is within the U_ST of q sub-leaders then each of the q sub-leaders update their weights as follows. Suppose $s\mu$ is the rough-fuzzy membership of x to one of the sub-leader sl among q sub-leaders then its weight is updated by $\frac{s\mu}{n_i}$, else if $q = 0$ then x becomes a new sub-leader with a weight of $\frac{1}{n_i}$.

2. Else if x is within the U_T of p leaders then each of the p leaders update their weights as follows. Suppose μ is the rough-fuzzy membership of x to one of the leader l among p leaders then its weight is updated by $\frac{\mu}{n_i}$ and it is done for all p leaders. For each of the p leaders, the same pattern x is verified to see if it is within the L_ST of any sub-leader $sl \in \mathcal{SL}_{ij}$ (suppose x is assigned to j^{th} leader with a rough-fuzzy membership μ) then the weight of the sub-leader sl is updated by $\frac{\mu}{n_i}$,

else if x is within the U_ST of q sub-leaders then each of the q sub-leaders update their weights as follows. Suppose $s\mu$ is the rough-fuzzy membership of x to one of the sub-leader sl among q sub-leaders then its weight is updated by $\frac{\mu \times s\mu}{n_i}$, else if $q = 0$ then x becomes a new sub-leader with a weight of $\frac{\mu}{n_i}$.

3. Else if $p = 0$ then x becomes a new leader and a new sub-leader with a weight of $\frac{1}{n_i}$.

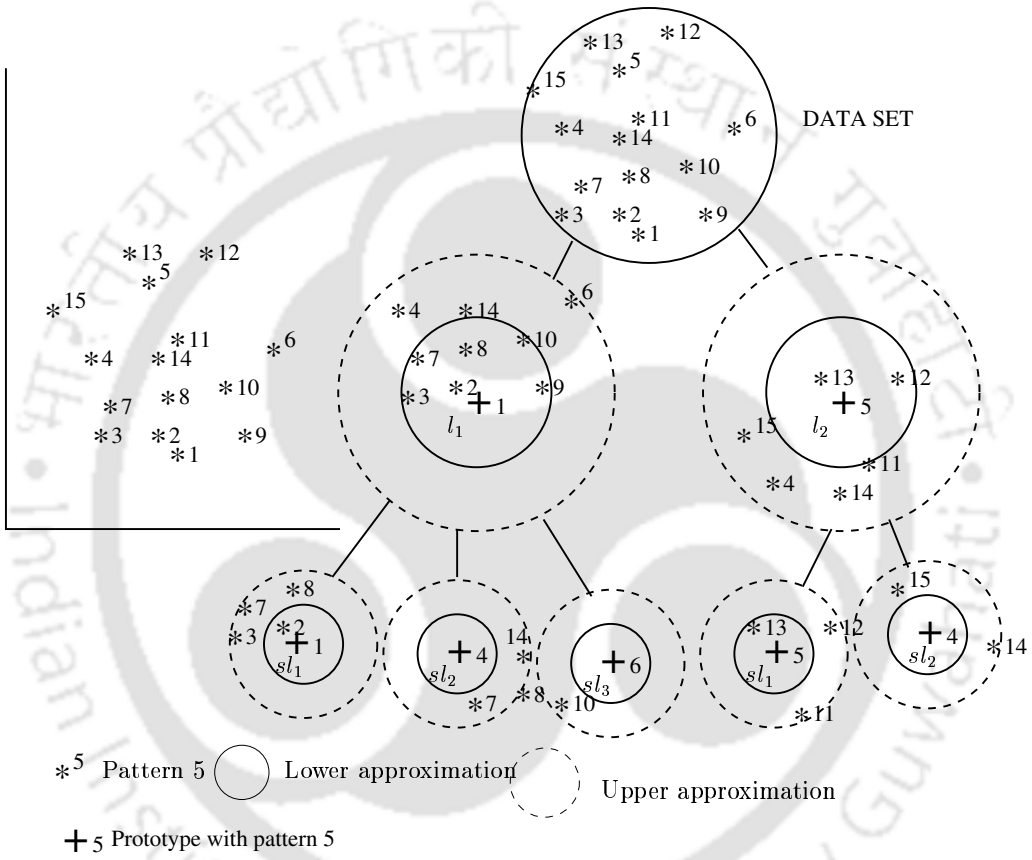


Figure 4.3: Rough fuzzy weighted leaders-subleaders method

Figure 4.3 represents 15 patterns in a class. The scanning order of patterns are as follows $x_1, x_2, x_3, \ldots, x_{15}$. Here, $\{x_1, x_5\}$ is a set of leaders and $SL_1 = \{x_1, x_4, x_6\}$, $SL_2 = \{x_4, x_5\}$ are two sets of subleaders. Patterns are shown in lower and upper approximations of prototypes and patterns in upper approximations are shared by one or more prototypes. Hence, pattern x_4 is a subleader of two different leaders with fuzzy membership values which is not the case in *Leaders-Subleaders* method.

Algorithm 4.3 Rough-fuzzy weighted leaders-subleaders($\mathcal{D}_i$, L_T, U_T)

$L_i = \phi$;
for each $x \in \mathcal{D}_i$ **do**
  if there is $l \in L_i$ such that $||l - x|| \leq L\_T$ **then**
    {suppose x is assigned to a leader j *i.e.*,$l^{(j)}$}
    weight(l)=weight(l)+$\frac{1}{n_i}$;
    SL_{ij} =WeightedSubleaders(x, L_ST, U_ST, $l^{(j)}$, 1);
  else
    if for all $l \in L_i$ such that $||l - x|| \leq U\_T$ **then**
      { Suppose there are r such leaders $l^{(j)}$}
      for $p = 1$ to r **do**
        Calculated the fuzzy membership for each $l^{(p)}$, say μ_p;
        weight($l^{(p)}$)=weight($l^{(p)}$)+$\frac{\mu_p}{n_i}$;
        SL_{ip} =WeightedSubleaders(x, L_ST, U_ST, $l^{(p)}$, μ_p);
      end for
    else
      weight($l^{(j)}$)=$\frac{1}{n_i}$;
      weight(sl)=$\frac{1}{n_i}$;
      $L_i = L_i \cup \{x\}$;
      $SL_{ij} = SL_{ij} \cup \{x\}$;
    end if
  end if
end for
Output L_i which is a set of tuples such that each tuple is of the form $< l^{(j)}, SL_{ij}, weight(l^{(j)}) >$ where $l^{(j)}$ is a j^{th} leader the set of leaders L_i of class ω_i, SL_{ij} is a set of subleaders of leader $l^{(j)}$ and $weight(l^{(j)})$ is its weight.

Since patterns which are within the lower threshold of a prototype are certainly belongs to the prototype and therefore they cannot belong to other prototype. Patterns which are not in the lower threshold of any prototype and which are within the upper threshold of a few prototype have to calculate rough-fuzzy membership values to the respective prototypes. Since the patters of these type are few in number (constant) and also prototypes of which these patterns belong to are also few in number and hence the overall method do not have computational burden. In the second case, at most the method requires $O(p.q)$ updates of the weights of sub-leaders for a pattern x where p and q are constants.

An important property of the rough-fuzzy weighted leaders-subleaders method is that

Algorithm 4.4 WeightedSubleaders(x, L_ST, U_ST, $l^{(j)}$, μ_j)

if there is $sl \in SL_{ij}$ such that $||sl - x|| \leq L\_ST$ **then**
  weight(sl)=weight(sl)+$\frac{\mu_j}{n_i}$;
else
  if for all $sl \in SL_{ij}$ such that $||sl - x|| \leq U\_ST$ **then**
    {Suppose there are m such subleaders}
    for $q = 1$ to m **do**
      Calculated the fuzzy membership $s\mu_q$ for each $sl^{(q)}$;
      weight($sl^{(q)}$)=weight($sl^{(q)}$)+$\frac{s\mu_q \times \mu_j}{n_i}$;
    end for
  else
    weight(sl)=$\frac{\mu_j}{n_i}$;
    $SL_{ij} = SL_{ij} \cup \{x\}$
  end if
end if
Output SL_{ij} which is a set of tuples such that each tuple is of the form $< sl, weight(sl) >$ where sl is a subleader of leader $l^{(j)}$ which is derived from the class ω_i and $weight(sl)$ is its weight.

the weight of a leader is equal to sum of the weights of its subleaders. Since any pattern that belongs to a leader updates its weight as either $\frac{1}{n_i}$ or $\frac{\mu}{n_i}$. Note that μ is the rough-fuzzy membership value of a leader to which the pattern belongs to in case of the pattern is in upper approximation of that leader. Also, the rough-fuzzy membership value is a scaled one (see equation (4.4)). If the weight of a leader is $\frac{1}{n_i}$ then the total weight updated for it's subleaders are either $\frac{1}{n_i}$ in case if pattern is within the lower threshold of subleader or else the sum of the weights updated for subleaders using scaled rough-fuzzy membership values which is once again becomes $\frac{1}{n_i}$. In the second case where weight of a leader is $\frac{\mu}{n_i}$ then sum of the updated weights of it's subleaders also becomes $\frac{\mu}{n_i}$. In case if a pattern becomes a new leader then it also becomes a new subleader with same weight. Hence the property is proved.

For each class of training patterns, the rough-fuzzy weighted leaders-subleaders method is applied separately with the same values of U_T and U_ST and the classifier is described below.

4.4.2 RF-wk-NLC: using rough fuzzy weighted leaders-subleaders method

This section describes the proposed classifier. Let L_i be the set of leaders obtained for class ω_i, for $i = 1$ to c. Let L be the set of all leaders. That is, $L = L_1 \cup \ldots \cup L_c$. For a given test pattern t, the k nearest leaders from L is obtained. Let these leaders be $\{l^{(1)}, l^{(2)}, \ldots, l^{(k)}\}$ such that $||l^{(1)} - t|| \leq ||l^{(2)} - t|| \leq \ldots \leq ||l^{(k)} - t||$. We find a distance from t to leader $l^{(k)}$ and let it be α. The lower and upper approximations of the set of k nearest leaders of t is given by the equations 4.6 and 4.7 respectively.

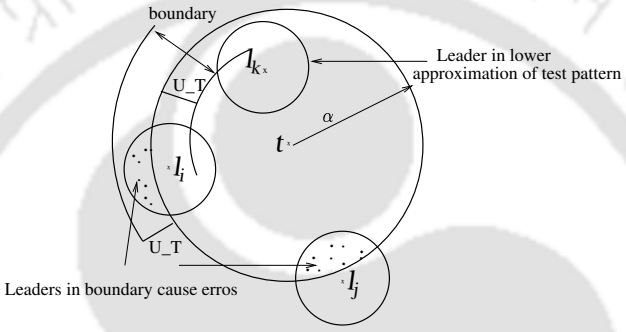


Figure 4.4: Errors in the boundary region of a test pattern

$$\underline{\mathcal{L}_t} = \{l \in \mathcal{L} \mid ||l - t|| \leq \alpha - U\_T\}. \tag{4.6}$$

$$\overline{\mathcal{L}_t} = \{l \in \mathcal{L} \mid ||l - t|| \leq \alpha + U\_T\} \tag{4.7}$$

Boundary of the set of k nearest leaders of t is given by the equation 4.8

$$BN_t = \overline{\mathcal{L}_t} - \underline{\mathcal{L}_t} \tag{4.8}$$

The lower approximation of the set of k nearest leaders of t is the set of leaders which are certainly within the volume of a radius α. The upper approximation of the set of k nearest leaders of t is the set of leaders which are possibly with in the volume of a radius α. The boundary of the set is the set of leaders which may or may not be within the volume of

Algorithm 4.5 Rough Fuzzy weighted k-Nearest-Leader Classifier(L, q)

{L is the set of all leaders derived from all classes. q is the query pattern to be classified}
Find k nearest leaders of q from L.
{Let these k nearest leaders are $\{l_1, \ldots, l_k\}$ such that $||l_1 - t|| \leq \ldots \leq ||l_k - t||$}
Find distance from q to l_k let it be α;
Find the lower and upper approximations of q from L. *i.e.,* $\underline{\mathcal{L}_q}$, $\overline{\mathcal{L}_q}$.
The boundary of a test pattern q (*i.e.,* $BN_q = \overline{\mathcal{L}_q} - \underline{\mathcal{L}_q}$) is calculated.
{let the lower approximation leaders set be Z}
Find the subleaders of every leader in BN_q such that $||sl - q|| \leq \alpha$.
{Let these subleaders set be BZ}
Find the cumulative weight of prototypes (leaders and subleaders) that belongs to each class in $Z \cup BZ$. Let this be W_i for class ω_i, for $i = 1$ to c.
Class label assigned for $t = argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$.

radius α. Hence, the proposed classifier uses the leaders in the lower approximation of t. Subleaders of each boundary leader which are within α distance from t can be used to reduce the errors in the boundary which is shown in the figure 4.4.

Now, the classifier works as follows. It finds leaders $l \in \underline{\mathcal{L}_t}$ and let the set of these leaders be Z. It also finds subleaders of every leader $l \in BN_t$ such that $||sl - t|| \leq \alpha$. Let the set of these subleaders be BZ. For each class of prototypes (leaders and subleaders) in $Z \cup BZ$, their respective cumulative weight is found. Let this for class ω_i be W^i, for $i = 1$ to c.

The classifier chooses the class according to $argmax_{\omega_i}\{W_1 P(\omega_1), \ldots, W_c P(\omega_c)\}$. If $P(\omega_i)$ is not explicitly given then it is taken to be n_i/n where n_i is the number of training patterns from class ω_i and n is the total number of training patterns.

Theorem 4 *As* U_T $\rightarrow$ *0 and* $n \rightarrow \infty$, *the rough fuzzy weighted k-nearest leader classifier is equal to the* Bayes *classifier.*

Proof: Let α *be the distance from the test pattern* t *to the* k^{th} *nearest leader from* L *(set of all leaders). Approximately class conditional density at* t *for class* ω_i *is:*

$$\hat{p}(t \mid \omega_i) = \frac{m_i}{n_i \cdot V}$$

where m_i *is the number of patterns that are present in the hyper-sphere of radius* α *that*

belongs to the class ω_i *and* n_i *is the total number of training patterns for the class* ω_i*, and* V *is the volume of the hyper-sphere. Asymptotically as* $n_i \to \infty, m_i \to \infty, m_i/n_i \to 0$ *and* $V \to 0$*, it can be shown that* $\hat{p}(t \mid \omega_i) \to p(t \mid \omega^i)$ *[Duda et al. (2000)].*

If U_T $\to$ *0, then* L_T $\to$ *0,* U_ST $\to$ *0, and* L_ST $\to$ *0. It is also clear that if* U_T $\to$ *0, then* $\overline{\mathcal{L}_t} = \underline{\mathcal{L}_t}$ *and* $BN_t = \phi$*.*

If U_T $\to$ *0, then the proposed classifier has a rough fuzzy weighted leaders-subleaders in which each pattern is a leader as well as its subleader with a weight of* $\frac{1}{n_i}$*. If* W_i *is the cumulative weight of each class of prototypes from the volume of region* α*, then it is easy to see that*

$$W_i = \frac{m_i}{n_i}$$

and hence is equal to the $\hat{p}(t \mid \omega_i)$*, for* $i = 1$ *to* c*. So,* $W^i \cdot P(\omega^i)$ *is proportionate to the posterior probability* $\hat{P}(\omega^i \mid t)$

Hence, the proposed classifier as U_T $\to$ *0 is equal to the Bayes classifier.*□

By using *RF-wk-NLC*, if *U_T*$\neq$ 0 and $n \to \infty$, then the prototypes (leaders and subleaders) are less in number and these prototypes are within the hyper-sphere of radius α. Since, W_i is the cumulative weight of each class of prototypes in the hyper-sphere of radius α then it is easy to see that

$$W_i \approx \frac{m_i}{n_i}$$

and hence is proportionate to the $\hat{p}(t \mid \omega_i)$, for $i = 1$ to c. So, $W_i \cdot P(\omega_i)$ is proportionate to the posterior probability $\hat{P}(\omega_i \mid t)$ where $P(\omega_i)$ is the apriori probability for class ω_i.

From the above argument, it is clear that the proposed method is approximately doing the *Bayes* classification as done by the k-nearest neighbor classifier. The proposed rough fuzzy weighted k-nearest leader classifier is given in Algorithm 4.5 where rough-fuzzy weighted leaders derived using Algorithm 4.2 is used.

4.5 Experimental Results

Experimental studies are done with one synthetic data set and five standard data sets. The details of the standard data sets are given in Table 2.1 and the details of the synthetic data set is given below.

The classifiers chosen for comparison purposes are: (1) the nearest neighbor classifier (NNC), (2) the k-nearest neighbor classifier (k-NNC), (3) the nearest leader classifier (NLC), (4) the adaptive rough-fuzzy nearest leader classifier (ARFNLC), (4) the nearest leader-subleader classifier (NLSC). The ARFNLC and NLSC are explained in Section 1.6 and, (5) the weighted k-nearest leader classifier which is explained in Chapter 3. Finally, (6) the proposed rough-fuzzy weighted k-nearest leader classifier (RF-wk-NLC).

In the experimental results, two tables and two graphs are given for each data set. First table is used to compare the number of prototypes, value of the free variable (k), design time and classification time of the classifiers for different threshold values. We considered the design time as time taken to derive prototypes and classification time is the time taken for classifying all test patterns in the test set. The value of the free variable k is found by using three fold cross validation method and the results are tabulated in second table. First plot compares the performance of those classifiers which use single level leaders (NLC, wk-NLC, ARFNLC and RF-wk-NLC) and also the performance of conventional classifiers (NNC, k-NNC) with varying threshold values along horizontal axis. Note that NNC, k-NNC do not require threshold value and hence the performance is shown as a horizontal line parallel to x-axis. Second plot compares the performance of all six classifiers with varying threshold values along horizontal axis. For NLC, wk-NLC, ARFNLC and RF-wk-NLC require one threshold value and hence first value of each point on horizontal axis are used where as NLSC and RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders require two threshold values and hence it uses first and second values of each point on horizontal axis. Note that two subthreshold values are chosen for each of the four threshold values and hence there are eight different threshold, subthreshold values.

4.5.1 Synthetic data set

A two dimensional *synthetic* data for a two class problem is generated as follows. First class has 200000 patterns which were *i.i.d.* drawn from a normal distribution having mean as $(0, 0)^T$ and covariance matrix as $I_{2\times2}$(*i.e.,*identity matrix). Second class also is of 200000 patterns which is also *i.i.d.* drawn from a normal distribution with mean $(2.56, 0)^T$ and covariance matrix $I_{2\times2}$. The Bayes error rate for this synthetic data set is 10%. The data set is divided randomly into two parts consisting of 300000 and 100000 patterns which are used as training and testing sets respectively.

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.2. The values of k chosen based on cross-validation results for different threshold values are shown in Table 4.1. From Table 4.1, the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders- subleaders is less than 0.2% when compared with the classification time of NNC and k-NNC and it's performance is almost equal to the performance of k-NNC which is shown in Figure 4.5. From Figure 4.5, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

4.5.2 Covtype data set

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.4. The values of k chosen based on cross-validation results for different threshold values are shown in Table 4.3. NNC has good performance for *Covtype* data set and also ARFNLC is not used since it requires huge computational burden for large data set like *Covtype* data set. From Table 4.3, With *U_T=0.3* and *L_T=0.1* the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders- subleaders is less than 0.3% when compared with the classification time of NNC and k-NNC and it's performance is almost close to the performance of NNC which is shown in Figure 4.6. From Figure 4.6, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

Table 4.1: Synthetic data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	65436
k-NNC	-		-		74	-	73900
NLC	0.6		221		1	0.72	1.52
	0.5		306		1	0.82	2.1
	0.4		455		1	1.04	3.05
	0.3		755		1	1.49	4.93
wk-NLC	0.6		221		25	0.72	7.65
	0.5		306		25	0.82	12.9
	0.4		455		25	1.04	24.59
	0.3		755		25	1.49	41.4
RF-wk-NLC	0.6		221		25	2.16	7.65
	0.5		306		25	2.75	12.9
	0.4		455		25	3.85	24.59
	0.3		755		25	6.10	41.4
ARFNLC	U_T	L_T	#L	#S			
	0.6	0.5	196	25	1	666.97	2.9
	0.5	0.4	280	24	1	1306.67	20.7
	0.4	0.3	467	120	1	3615.46	38.33
	0.3	0.2	945	113	1	1382.32	72.44
NLSC	T	S_T	#L	#SL			
	0.6	0.4	221	650	1	1.52	1.41
	0.5	0.3	306	996	1	1.92	1.95
	0.4	0.2	455	1789	1	2.28	2.78
	0.3	0.1	755	4982	1	3.25	4.57
RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method	U_T	U_ST					
	0.6	0.4	221	1224	13	4.49	16.32
	0.5	0.3	306	1921	12	7.44	23.99
	0.4	0.2	455	2095	17	1.02	34.49
	0.3	0.1	755	10552	8	22.38	48.04

4.5.3 SensIT vehicle(seismic) data set

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.6. The values of k chosen based on cross-validation results for different threshold values are shown in Table 4.5. From Table 4.5, with *U_T*=0.8 and *U_ST=0.5* the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders-subleaders is less than 0.4% when compared with the classification time of NNC and k-NNC and it's performance is almost close to the performance of k-NNC which is shown in Figure 4.7. From Figure 4.7, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

4.5.4 SensIT vehicle(acoustic) data set

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.8. The values of k chosen based on cross-validation results for different

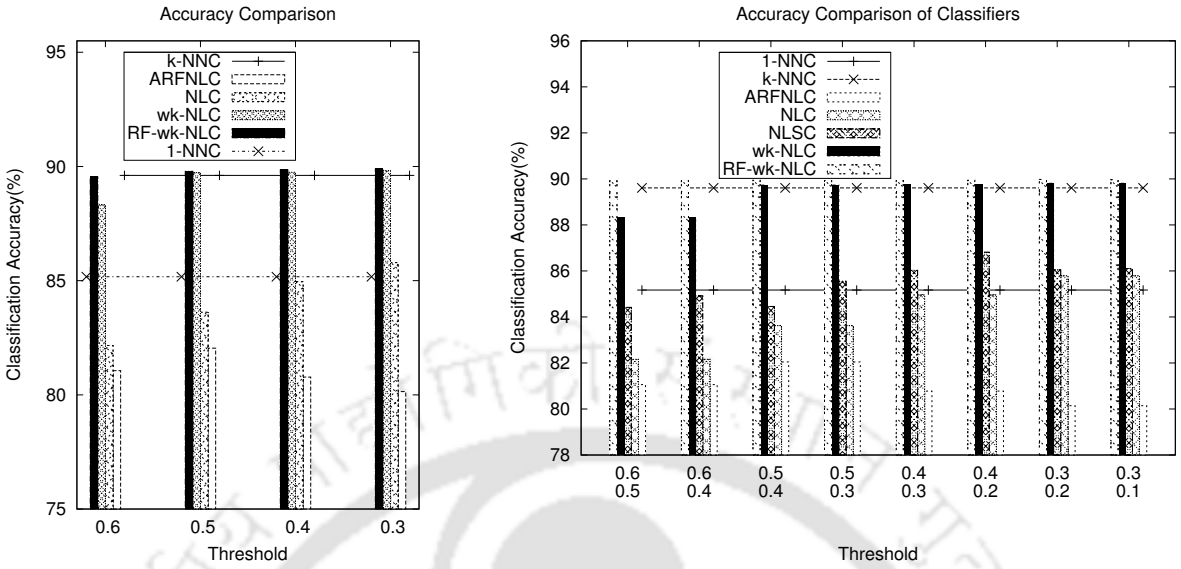


Figure 4.5: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the synthetic data set

threshold values are shown in Table 4.7. From Table 4.7, with *U_T*=0.7 and *U_ST=0.4* the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders-subleaders is less than 0.7% when compared with the classification time of NNC and k-NNC and it's performance is almost close to the performance of k-NNC which is shown in Figure 4.8. From Figure 4.8, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

4.5.5 SensIT vehicle(combined) data set

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.10. The values of k chosen based on cross-validation results for different threshold values are shown in Table 4.9. From Table 4.9, with *U_T*=1.5 and *U_ST=0.8* the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders-subleaders is less than 0.8% when compared with the classification time of NNC and k-NNC and it's performance is almost close to the performance of k-NNC which is shown in Figure 4.9.

Table 4.2: Synthetic data set (Cross-validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	0.6 0.4	0.5 0.3	0.4 0.2	0.3 0.1
1	81.21(0.12)	81.52(0.13)	82.88(0.02)	82.58(0.24)
5	89.95(0.14)	89.92(0.08)	90.05(0.12)	89.96(0.19)
6	89.96(0.21)	89.96(0.23)	90.04(0.05)	90.00(0.14)
7	90.00(0.11)	89.96(0.18)	90.03(0.21)	**90.03(0.09)**
8	89.99(0.06)	89.92(0.16)	90.03(0.07)	90.01(0.05)
9	90.00(0.07)	89.96(0.14)	90.03(0.15)	90.00(0.04)
10	90.00(0.13)	89.95(0.14)	90.04(0.18)	89.99(0.12)
11	90.01(0.01)	89.96(0.09)	90.04(0.13)	89.97(0.17)
12	90.01(0.04)	**89.99(0.10)**	90.06(0.07)	89.98(0.18)
13	**90.03(0.02)**	89.92(0.11)	90.06(0.14)	89.96(0.16)
14	90.00(0.05)	89.94(0.15)	90.05(0.01)	89.95(0.06)
15	90.01(0.14)	89.92(0.13)	90.05(0.01)	89.87(0.07)
16	90.02(0.21)	89.94(0.09)	90.05(0.04)	89.83(0.06)
17	90.01(0.04)	89.97(0.07)	**90.08(0.01)**	89.79(0.12)
18	90.01(0.07)	89.96(0.14)	90.05(0.07)	89.75(0.04)
19	89.99(0.12)	89.95(0.05)	90.06(0.14)	89.78(0.12)
20	90.00(0.21)	89.92(0.02)	90.05(0.09)	89.74(0.14)

From Figure 4.9, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

4.5.6 Ijcnn1 data set

The cross-validation results for RF-wk-NLC: using rough-fuzzy leaders-subleaders method is shown in Table 4.12. The values of k chosen based on cross-validation results for different threshold values are shown in Table 4.11. From Table 4.11, with U_T=0.7 and U_ST=*0.2* the classification time of RF-wk-NLC:using rough-fuzzy weighted leaders-subleaders is less than 0.3% when compared with the classification time of NNC and k-NNC and it's performance is almost close to the performance of k-NNC which is shown in Figure 4.10. From Figure 4.10, the performance of RF-wk-NLC: using rough-fuzzy weighted leaders method is improved when compared with wk-NLC.

Table 4.3: Covtype data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	1373066
NLC	0.5		1020		1	26.56	74.02
	0.4		1887		1	44.77	145.34
	0.3		4129		1	115.56	393.68
	0.2		12385		1	530.82	1238.07
wk-NLC	0.5		1020		7	26.56	129.52
	0.4		1887		19	44.77	340.42
	0.3		4129		11	115.56	765.46
	0.2		12385		5	530.82	2177.86
RF-wk-NLC	0.5		1020		7	26.56	129.52
	0.4		1887		19	44.77	340.42
	0.3		4129		11	115.56	765.46
	0.2		12385		5	530.82	2177.86
	T	S_T	#L	#SL			
NLSC	0.5	0.3	1020	4982	1	55.32	76.6
	0.4	0.2	1887	15580	1	84.5	147.34
	0.3	0.1	4129	73117	1	320.84	433.29
	0.2	0.05	12385	213558	1	230.91	1377.02
	U_T	U_ST					
RF-wk-NLC:	0.5	0.3	1020	11105	2	119.91	229.85
using rough-fuzzy	0.4	0.2	1887	36733	2	239.07	243.70
weighted leaders-	0.3	0.1	4129	139750	1	418.65	1350.76
subleaders method	0.2	0.05	12385	213558	1	395.48	3907.22

4.6 Conclusion

This chapter proposed to use rough-fuzzy membership values along with leaders to achieve better results. To reduce the time requirements further, the chapter proposed a two level hierarchical rough-fuzzy clustering method for prototype selection which are compared with earlier related methods. A rough-fuzzy weighted k-nearest leader classifier is proposed which uses these two levels of prototypes for classification and is called RF-wk-NLC. It is analyzed using rough set theory and showed that under some conditions the proposed classifier is equal to the *Bayes* classifier. The proposed method can achieve classification accuracy which is very similar to that of the k-NNC and is superior than earlier related methods. It is a much faster classifier than k-NNC or NNC.

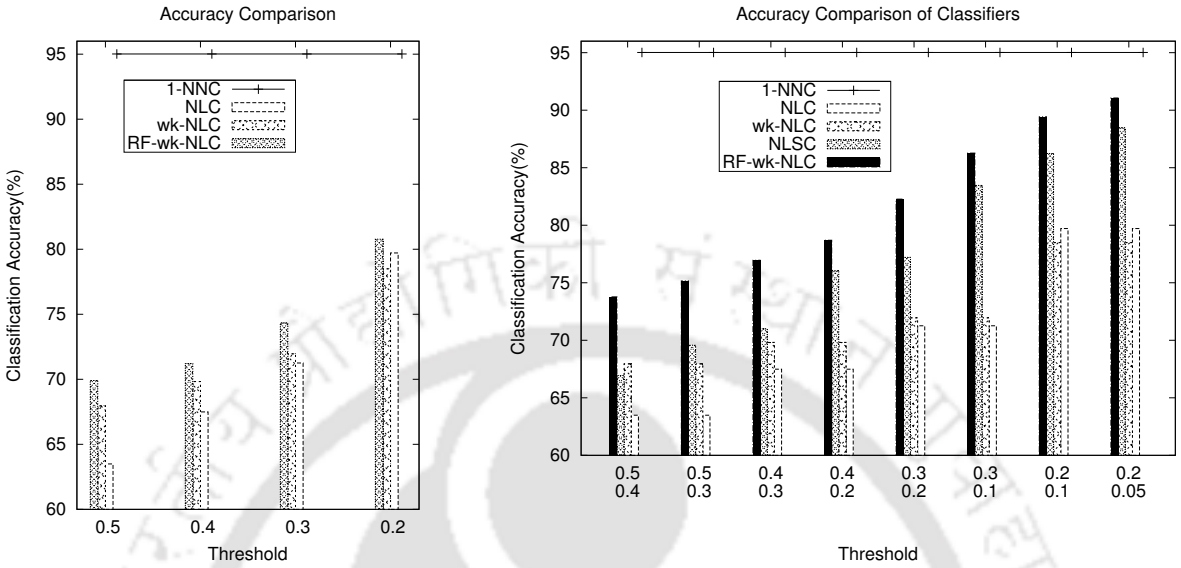


Figure 4.6: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the covtype data set

Table 4.4: Covtype data set (Cross-validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	0.5 0.3	0.4 0.2	0.3 0.1	0.2 0.05
1	73.26(0.22)	75.93(0.13)	**81.98(0.22)**	**88.98(0.24)**
2	**73.47(0.18)**	**76.12(0.19)**	81.21(0.13)	88.38(0.19)
3	72.91(0.13)	76.03(0.24)	80.82(0.24)	87.95(0.10)
4	72.75(0.23)	75.94(0.21)	80.23(0.23)	87.53(0.25)
5	72.32(0.24)	75.87(0.18)	80.05(0.12)	86.76(0.19)
9	70.78(0.17)	73.98(0.14)	79.03(0.15)	84.89(0.14)
10	70.52(0.09)	73.54(0.24)	78.57(0.18)	84.19(0.22)

Table 4.5: SensIT vehicle(seismic) data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	129248.32
k-NNC	-		-		30	-	171502.65
NLC	1.2		271		1	6.22	3.64
	1		600		1	6.36	6.49
	0.9		980		1	6.64	9.21
	0.8		1490		1	7.22	14.73
wk-NLC	1.2		271		15	6.22	6.02
	1		600		16	6.36	11.29
	0.9		980		20	6.64	18.26
	0.8		1490		20	7.22	27.53
RF-wk-NLC	1.2		271		15	6.22	6.02
	1		600		16	6.36	11.29
	0.9		980		20	6.64	18.26
	0.8		1490		20	7.22	27.53
ARFNLC	U_T	L_T	#L	#S			
	0.6	0.5	196	25	1	666.97	2.9
	0.5	0.4	280	24	1	1306.67	20.7
	0.4	0.3	467	120	1	3615.46	38.33
	0.3	0.2	945	113	1	1382.32	72.44
NLSC	T	S_T	#L	#SL			
	1.2	0.9	271	1167	1	12.55	4.54
	1	0.8	600	2130	1	13.74	13.87
	0.9	0.7	980	3525	1	15.00	20.07
	0.8	0.5	1490	12067	1	21.04	34.69
RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method	U_T	U_ST					
	1.2	0.9	271	2921	4	22.66	39.87
	1	0.8	600	4983	6	29.50	98.16
	0.9	0.7	900	8856	5	36.13	200.66
	0.8	0.5	1490	23560	8	359.60	441.11

Table 4.6: SensIT vehicle(seismic) data set (Cross-validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	1.2 0.9	1.0 0.8	0.9 0.7	0.8 0.5
1	62.59	65.50(0.13)	66.50(0.02)	67.24(0.24)
2	68.01(0.18)	70.10(0.19)	70.82(0.23)	70.97(0.12)
3	68.54(0.23)	70.55(0.14)	70.98(0.14)	71.06(0.13)
4	**68.76(0.03)**	70.63(0.21)	71.25(0.23)	71.27(0.25)
5	68.32(0.24)	70.50(0.18)	**71.55(0.12)**	71.33(0.19)
6	68.96(0.21)	**70.87(0.23)**	70.95(0.08)	71.45(0.04)
7	68.14(0.31)	70.69(0.08)	70.68(0.21)	71.38(0.29)
8	68.01(0.16)	70.64(0.26)	70.56(0.07)	**71.67(0.15)**
9	68.05(0.27)	70.55(0.14)	70.68(0.15)	70.95(0.14)
10	68.14(0.13)	70.51(0.24)	70.65(0.18)	70.65(0.22)

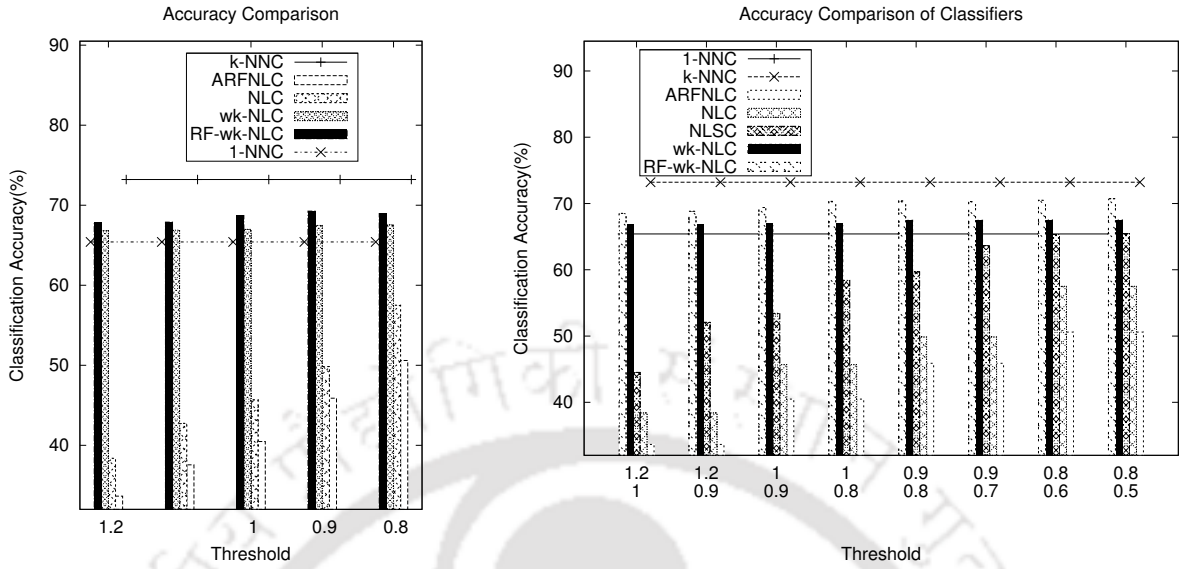


Figure 4.7: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the sensIT vehicle(seismic) data set

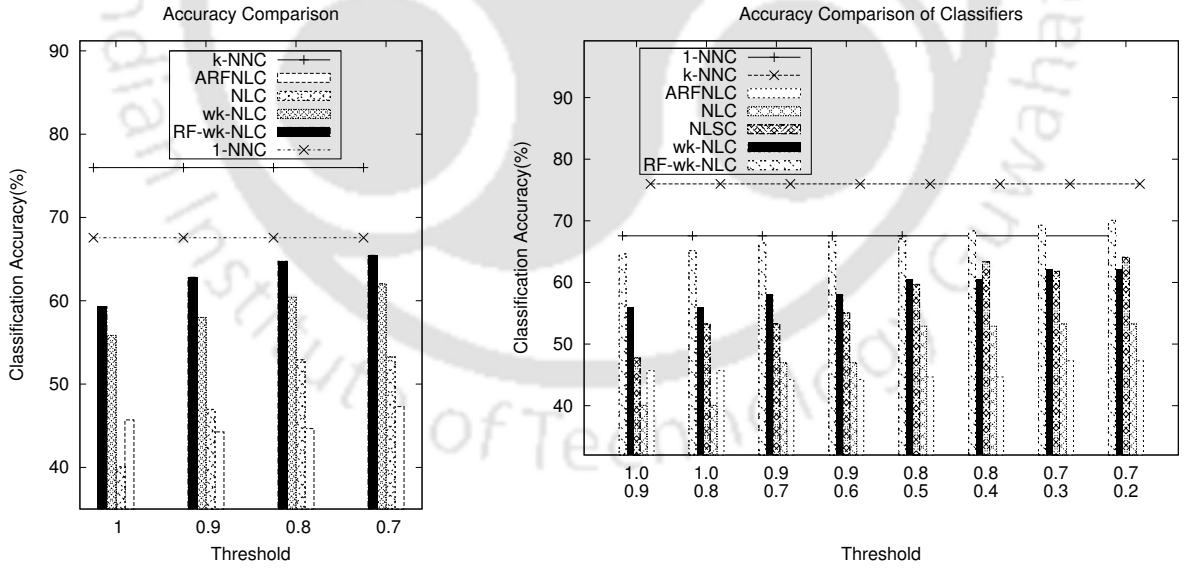


Figure 4.8: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the sensIT vehicle(acoustic) data set

Table 4.7: SeinseIT vechical(acoustic) data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	125935.77
k-NNC	-		-		21	-	158967.30
NLC	1		452		1	6.31	4.56
	0.9		741		1	6.44	6.53
	0.8		1365		1	7.04	10.97
	0.7		2632		1	9.07	22.59
wk-NLC	1		452		16	6.31	8.89
	0.9		741		16	6.44	13.79
	0.8		1365		16	7.04	17.54
	0.7		2632		16	9.07	47.48
RF-wk-NLC	1		452		16	10.21	8.89
	0.9		741		16	12.81	13.79
	0.8		1365		16	17.85	17.54
	0.7		2632		16	29.28	47.48
ARFNLC	U_T	L_T	#L	#S			
	1	0.8	705	280	1	1542.86	11.8
	0.9	0.7	1392	577	1	5420.12	23.62
	0.8	0.6	2960	1118	1	22906.01	55.02
	0.7	0.5	6234	2069	1	94102.66	115.05
NLSC	T	S_T	#L	#SL			
	1	0.8	452	1846	1	12.71	5.05
	0.9	0.7	741	3656	1	14.06	7.33
	0.8	0.5	1365	12522	1	20.53	13.06
	0.7	0.4	2632	22853	1	13.86	50.19
RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method	U_T	U_ST					
	1	0.8	452	5471	3	18.11	110.49
	0.9	0.7	741	11334	7	30.98	260.45
	0.8	0.5	1365	55075	4	266.08	717.84
	0.7	0.4	2632	45105	8	27.14	969.63

Table 4.8: SensIT vehicle(acoustic) data set (Cross-validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	1.0 0.8	0.9 0.7	0.8 0.5	0.7 0.4
1	59.31(0.08)	62.50(0.13)	64.50(0.12)	65.24(0.24)
2	64.05(0.19)	67.70(0.09)	68.52(0.13)	69.27(0.12)
3	**65.56(0.13)**	68.35(0.34)	68.98(0.14)	69.16(0.13)
4	65.27(0.23)	68.63(0.21)	**69.25(0.11)**	69.07(0.25)
5	65.46(0.14)	68.72(0.09)	68.94(0.14)	69.13(0.19)
6	65.51(0.06)	68.87(0.11)	68.48(0.04)	69.35(0.04)
7	65.34(0.31)	**69.39(0.18)**	68.37(0.01)	69.88(0.19)
8	65.35(0.16)	68.94(0.22)	68.16(0.17)	**70.17(0.05)**
9	65.29(0.27)	68.55(0.06)	67.98(0.05)	69.95(0.14)
10	64.99(0.13)	68.11(0.24)	67.65(0.18)	69.25(0.22)

Table 4.9: SeinseIT vechical(combined) data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	162658.25
k-NNC	-		-		13	-	194826.56
NLC	1.8		452		1	7.41	6.59
	1.7		741		1	15.63	8.83
	1.6		1365		1	22.13	11.47
	1.5		2632		1	43.16	25.19
wk-NLC	1.8		452		4	7.51	10.89
	1.7		741		7	15.94	16.79
	1.6		1365		6	22.23	21.54
	1.5		2632		3	43.87	47.48
RF-wk-NLC	1.8		452		4	12.21	10.89
	1.7		741		7	18.81	16.79
	1.6		1365		6	26.85	21.54
	1.5		2632		3	49.28	47.48
ARFNLC	U_T	L_T	#L	#S			
	1.8	1.4	505	280	1	3642.86	8.8
	1.7	1.2	967	577	1	8420.12	15.62
	1.6	1.0	1660	1118	1	26906.01	55.02
	1.5	0.8	2534	2069	1	36102.66	82.05
NLSC	T	S_T	#L	#SL			
	1.8	1.4	357	2478	1	45.42	27.42
	1.7	1.2	452	4256	1	73.56	45.33
	1.6	1.0	1365	9934	1	113.43	85.26
	1.5	0.8	2632	16853	1	189.56	173.24
RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method	U_T	U_ST					
	1.8	1.4	357	4786	3	60.24	250.56
	1.7	1.2	452	10924	2	100.34	480.92
	1.6	1.0	768	21356	3	254.54	780.84
	1.5	0.8	1058	36105	4	380.23	969.63

Table 4.10: SensIT vehicle(combined) data set (Cross-validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	1.8 1.4	1.7 1.2	1.6 1.0	1.5 0.8
1	73.38(0.08)	76.92(0.03)	77.35(0.22)	78.02(0.14)
2	78.75(0.19)	**79.42(0.09)**	80.52(0.09)	81.17(0.02)
3	**79.16(0.13)**	79.21(0.24)	**80.98(0.16)**	81.96(0.23)
4	78.60(0.23)	78.74(0.11)	79.95(0.15)	**82.97(0.15)**
5	78.31(0.14)	78.25(0.09)	79.94(0.24)	81.93(0.09)
6	78.056(0.06)	77.93(0.21)	79.38(0.04)	81.35(0.14)
10	76.92(0.13)	77.19(0.14)	77.75(0.18)	78.25(0.12)

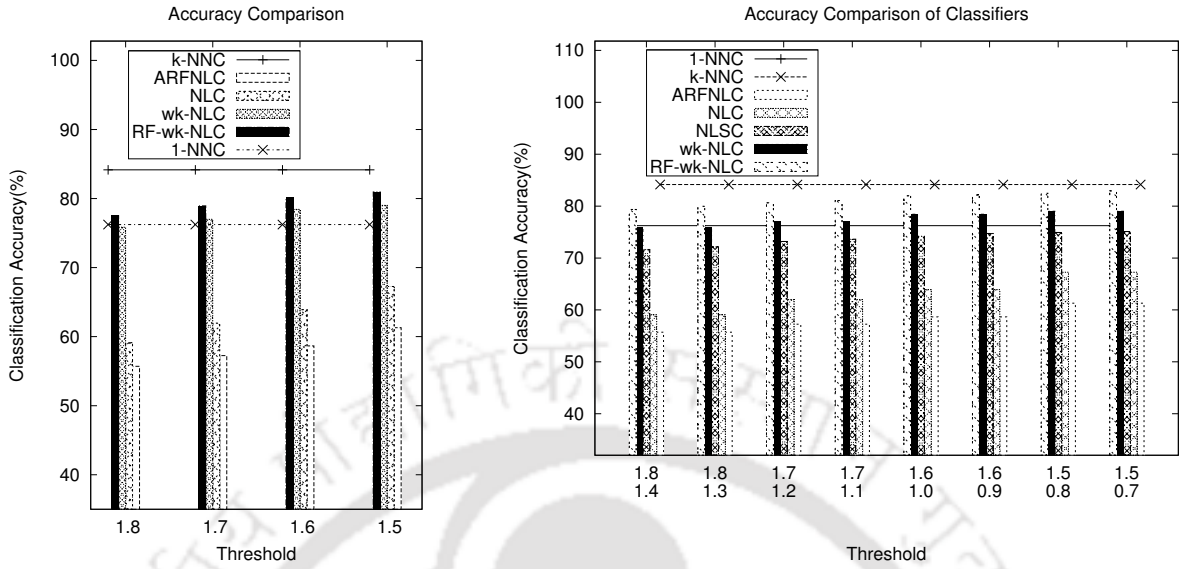


Figure 4.9: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the sensIT vehicle(combined) data set

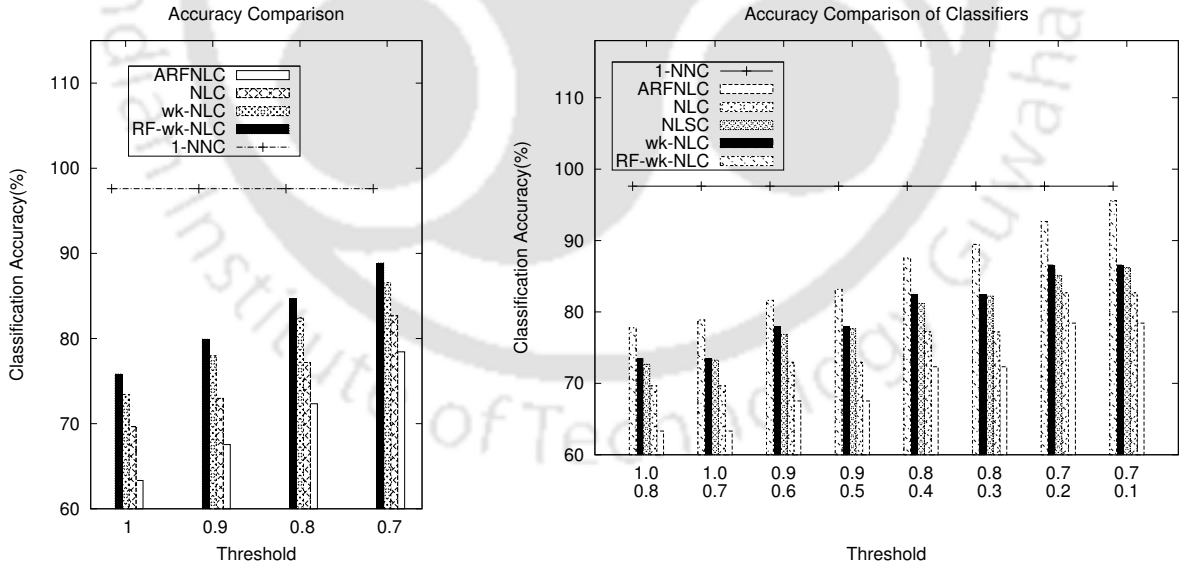


Figure 4.10: Performance comparison of (i) RF-wk-NLC and (ii) Performance comparison of RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method for the ijcnn1 data set

Table 4.11: Ijcnn1 data set results

Classifier	Threshold		Number of prototypes		k	Design time(s)	classification time(s)
NNC	-		-		1	-	96038.45
NLC	1.0		159		1	0.61	8.59
	0.9		245		1	0.93	14.83
	0.8		420		1	1.13	25.47
	0.7		967		1	1.76	36.19
wk-NLC	1.0		159		7	0.61	10.89
	0.9		245		4	0.93	17.79
	0.8		420		5	1.13	28.54
	0.7		967		5	1.76	42.48
RF-wk-NLC	1.0		159		7	1.21	10.89
	0.9		245		4	1.81	17.79
	0.8		420		5	2.85	28.54
	0.7		967		5	3.28	42.48
ARFNLC	U_T	L_T	#L	#S			
	1.0	0.8	505	280	1	3642.86	8.8
	0.9	0.7	967	577	1	8420.12	15.62
	0.8	0.6	1660	1118	1	26906.01	55.02
	0.7	0.5	2534	2069	1	36102.66	82.05
NLSC	T	S_T	#L	#SL			
	1.0	0.8	159	878	1	2.62	13.42
	0.9	0.6	245	1456	1	5.16	28.53
	0.8	0.4	420	3893	1	18.31	42.26
	0.7	0.2	967	14234	1	53.32	82.14
RF-wk-NLC: using rough-fuzzy weighted leaders-subleaders method	U_T	U_ST					
	1.0	0.8	159	985	6	8.24	27.56
	0.9	0.6	245	1765	4	18.34	48.11
	0.8	0.4	420	5325	3	49.54	81.54
	0.7	0.2	967	21562	2	135.23	197.71

Table 4.12: Ijcnn1 data set (Cross Validation results)

k	RF-wk-NLC for different U_T and U_ST values			
	1.0 0.8	0.9 0.6	0.8 0.4	0.7 0.2
1	74.31(0.18)	76.92(0.13)	78.35(0.13)	81.02(0.21)
2	75.85(0.29)	78.51(0.29)	80.82(0.13)	**83.97(0.11)**
3	76.16(0.14)	79.60(0.14)	**81.42(0.12)**	83.63(0.14)
4	76.54(0.13)	**80.57(0.06)**	80.95(0.05)	83.97(0.15)
5	76.82(0.13)	80.25(0.09)	79.94(0.24)	82.91(0.19)
6	**77.16(0.16)**	79.74(0.08)	79.38(0.14)	82.35(0.12)
7	76.91(0.32)	79.07(0.18)	79.37(0.21)	79.88(0.19)
10	76.12(0.23)	77.92(0.12)	77.15(0.10)	78.12(0.22)

Chapter 5

Rough-DBSCAN

Density based clustering techniques like DBSCAN (Density Based Clustering of Applications with Noise) [Ester et al. (1996)] can discover consistent arbitrary shaped clusters along with detection of noisy outliers. This is in contrast with k-means type clustering methods [Duda et al. (2000)] which finds compact and spherical shaped clusters. Single-linkage method [Jain et al. (1999)] can find arbitrary shaped clusters, but apart from its huge time requirements, it is a very sensitive method to noise. Methods like BIRCH [Zhang et al. (1996)] which uses scalable schemes (BIRCH uses CF-tree to represent the hierarchy of clustering) is suitable for spherical shaped compact clusters only. In this perspective, density based methods are attractive. But this also suffers from huge computational requirements. For DBSCAN, the time complexity is $O(n^2)$ where n is size of the data set. So it can take huge amounts of time with large data sets and hence is not suitable for data mining applications. One way to overcome this problem is to build an index over the data set like R-tree [Guttman (1984)] which will be useful for finding neighbors of a pattern based on a distance measure. But this solution is suitable only when the dimensionality of the data is low.

Hybrid clustering methods are recently used [E. Yelow Cheu et al. (2004)], [Lin and Chen (2005)], [Viswanath and Pinkesh (2006)] to overcome the problems with large data sets. The basic technique is to first find suitable prototypes from the large data set and then to apply the clustering method using only the prototypes. These kind of

schemes can be seen as approximate methods where the solution can deviate from that of the original method (which uses the entire data set) based on the representative power (quality) of the prototypes. The thesis proposes to use leaders (the prototypes derived using the leaders clustering method [Spath (1980)]) as prototypes to derive density based clusters quickly. But since as explained in the earlier chapters leaders alone cannot be used to find density at a point in the feature space, the thesis proposes to use additional information along with leaders. In case of classification (see chapters 3 and 4) the thesis proposed to use a weight or rough-fuzzy weight for each leader. Similarly, to obtain a hybrid method that is useful with the DBSCAN the chapter proposes to use a count value (a non-negative integer) with each leader as this is more appropriate than using a weight. The proposed hybrid clustering scheme called rough-DBSCAN since it is a modification of the well known density based clustering method DBSCAN [Ester et al. (1996)]. Further, an improvement over rough-DBSCAN called rough-fuzzy DBSCAN is presented which gives better results than rough-DBSCAN.

Rough set theory [Pawlak (1982)], [Li et al. (2006)], [Shiu et al. (2006)], [Ananthanarayana et al. (2001)], [Asharaf and Murthy (2004)] and fuzzy set theory [Zadeh (1965)], [Chanas and Kuchta (1992)], [Dubois and Prade (1990)], [Y. Y. Yao (1998)] are used to analyze the proposed methods. In literature there exists other rough set based similar clustering methods like rough k-means [Lingras and West (2004)], rough fuzzy leaders [Asharaf and Murty (2003)], etc.

Rest of the chapter is organized as follows. Section 5.1 briefly reviews DBSCAN method while Section 5.2 reviews leaders clustering method and presents a modified leaders method called *counted-leaders*. The proposed clustering method rough-DBSCAN is presented in Section 5.3. Section 5.4 explains further improvements over rough-DBSCAN. Experimental results are given in Section 5.5. Finally, some of the conclusions are given in Section 5.6.

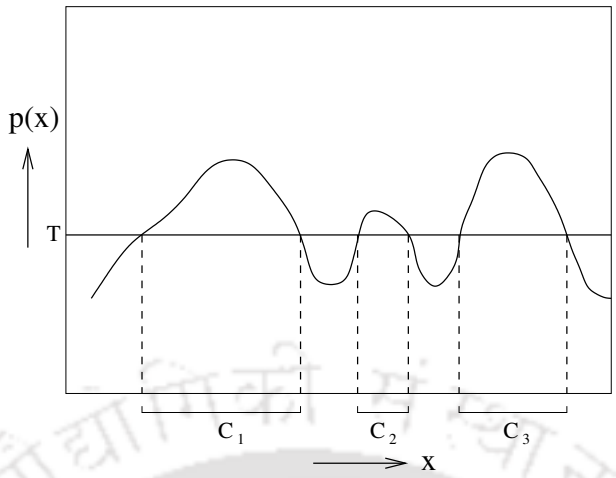


Figure 5.1: Clusters found by DBSCAN.

5.1 DBSCAN

DBSCAN groups the data points which are *dense* and *near-by* into a single cluster. For example, if the underlying probability density function for the data is as given in Figure 5.1, and let the threshold density to distinguish between dense and non-dense points be T, then the clusters found are C_1, C_2 and C_3. Patterns in the data set which do not belong to any of the clusters are called noisy patterns. Details of the method are given below.

Since only a data set is given as input (instead of density function), density at a point is found non-parametrically. It assumes that probability density over a small region is uniformly distributed and the density is given by m/nV where m is the number of points out of n input data points that are falling in the region and V is the volume of the region. The region is assumed to be a hyper sphere of radius ϵ and hence the threshold density can be specified by a parameter *MinPts*, minimum number of points required to be present in the region to make it dense.

According to DBSCAN if a pattern x is dense[1] then it is part of a cluster. A non-dense

[1]DBSCAN given in [Ester et al. (1996)] calls a dense point as a *core* point and a non-dense point as a *non-core* point. This modification is done to bring the material closer to the *pattern recognition* community.

pattern can also be part of a cluster as a border pattern of the cluster if it is at a distance less than or equal to ϵ from a dense pattern, otherwise it is a noisy outlier. Two patterns x_1 and x_2 are in a cluster if there is a sequence of patterns $x_1, y_1, y_2, \ldots, y_m, x_2$ in the data set such that:

1. the distance between successive patterns in the sequence is less than or equal to ϵ,
2. if $m = 0$, then at least one of x_1 and x_2 is dense, and
3. if $m > 0$, then the patterns $y_1, y_2, \ldots, y_m$ are dense.

The method DBSCAN is given in Algorithm 5.1 where $\mathcal{D}$ is the input data set, $N_\epsilon(x; \mathcal{D})$ is the subset of patterns in $\mathcal{D}$ that are present in the hyper-sphere of radius ϵ at x where $x \in \mathcal{D}$. $card(N_\epsilon(x; \mathcal{D}))$ is the cardinality of the set which is nothing but $\mid N_\epsilon(x; \mathcal{D}) \mid$. For given ϵ and *MinPts*, DBSCAN finds a dense point in the input set and expands it by merging neighboring dense regions together. The algorithm marks each pattern of $\mathcal{D}$ with a cluster identifier (*cid*) which gives the cluster to which the pattern belongs or gives a mark *"noise"* indicating that the pattern is a noisy one. One additional mark *"seen"* is used to distinguish between the patterns which are processed from that which are not. Note that a pattern which is initially marked as *"noise"* can later become a border point of a cluster and hence the *"noise"* mark can be deleted.

It can be seen that the time consuming step in DBSCAN is in finding $N_\epsilon(x; \mathcal{D})$ which can take $O(n)$ time where $x \in \mathcal{D}$ and $\mid \mathcal{D} \mid = n$. Hence the time complexity of the method is $O(n^2)$. On the other-hand, if we derive k leaders first by applying the leaders clustering method (see Section 2.3) and subsequently applying DBSCAN only with leaders has total time complexity of $O(n + k^2)$. This is because the leaders clustering method takes only $O(n)$ time to derive the leaders. Since it is shown both theoretically and experimentally that k is considerably smaller than n, especially with large data sets. Hence the hybrid scheme can be a faster one to work with large data sets. But, since leaders alone cannot be used to derive density information at a point in the feature space, the leaders method that is explained in Section 2.3 is modified where each leader is associated with a count value which is the number of followers the leader has. Since a leader is dense if the number

Algorithm 5.1 DBSCAN($\mathcal{D}$, ϵ, *MinPts*)

```
{Each cluster is given an identifier cid }
cid = 0;
for each pattern x in D do
  if x is not marked as "seen" then
    Mark x as "seen";
    Find N_ε(x; D);
    if card(N_ε(x; D)) < MinPts then
      Mark x as "noise";
    else
      Mark x as "seen";
      cid = cid +1;
      Mark each pattern of N_ε(x; D) with cluster identifier cid;
      Add each pattern of N_ε(x; D) which is not marked as "seen" to the list queue(cid).
      while queue(cid) is not empty do
        Take a pattern y from queue(cid) and mark it as "seen";
        if card(N_ε(y; D)) > MinPts then
          Mark each pattern of N_ε(y; D) with cluster identifier cid;
          If any pattern of N_ε(y; D) is marked as "noise" then remove this mark.
          Add each pattern of N_ε(y; D) which is not marked as "seen" to the list
          queue(cid).
        end if
        Remove y from queue(cid).
      end while
    end if
  end if
end for
Output all patterns of D along with their cid or "noise" mark;
```

of patterns surrounded by it within the ϵ radius is more than *MinPts*, usage of count is more appropriate than a weight (as done in classification purposes in previous chapters).

5.2 Counted-Leaders clustering method

The proposed modified leaders clustering method called the *Counted-Leaders* clustering method is given in the Algorithm 5.2. The algorithm outputs a set of triplets $\mathcal{L}^{\star}$ where each one consists of a leader, its followers set, and its count.

Algorithm 5.2 Counted-Leaders($\mathcal{D}$, τ)

$\mathcal{L} = \emptyset$;
for each $x \in \mathcal{D}$ **do**
  Find a leader $l \in \mathcal{L}$ such that $Distance(x, l) < \tau$. If there are many such leaders then choose the first one according to the ordering in which the set $\mathcal{L}$ is created.
  if there is no such l or when $\mathcal{L} = \emptyset$ **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
    $followers(x) = \{x\}$;
    $count(x) = 1$;
  else
    $followers(l) = followers(l) \cup \{x\}$;
    $count(l) = count(l) + 1$;
  end if
end for
Output $\mathcal{L}^\star = \{(l, followers(l), count(l)) \mid l \in \mathcal{L}\}$.

5.3 Rough-DBSCAN

In this section, we present the proposed density based clustering method called the rough-DBSCAN which aims at achieving a similar result as that of the DBSCAN but in a much smaller time requirement. Rough-DBSCAN uses the same values for the parameters ϵ and *MinPts* as that used by the DBSCAN, but instead of partitioning the data set directly, it first partitions the set of leaders derived from the data set which later can be expanded into a partition of the data set by replacing each leader by the set of patterns grouped with it, *i.e.,* its followers set.

Rough-DBSCAN works with $\mathcal{L}^\star = \{ (l, followers(l), count(l)) \mid l \in \mathcal{L}\}$ [2], the output of Algorithm 5.2, but it uses the set of leaders and their respective count values alone. Hence a leader needs to be categorized as either dense or non-dense using only count values of some of its neighboring leaders. According to DBSCAN a leader (which is also a pattern in the given data set $\mathcal{D}$) is dense if $card(N_\epsilon(l; \mathcal{D})) \geq MinPts$, is non-dense otherwise. From the set of leaders using the leader's count values alone it is not possible to find $N_\epsilon(l; \mathcal{D})$ exactly. But it can be roughly obtained by finding a lower and an upper approximations as explained below.

[2] $\mathcal{L}$ is used to denote the set of leaders alone.

Let $\underline{\mathcal{L}_l} = \{l_j \in \mathcal{L} \mid ||l_j - l|| < \epsilon - \tau\}$. $\underline{\mathcal{L}_l}$ is a set of leaders where for each one all its followers are guaranteed to be within the radius ϵ from l. For a leader l a lower approximation of $N_\epsilon(l; \mathcal{D})$ is

$$\underline{N_\epsilon(l; \mathcal{D})} = \bigcup_{l \in \underline{\mathcal{L}_l}} followers(l) \tag{5.1}$$

Let $\overline{\mathcal{L}_l} = \{l_j \in \mathcal{L} \mid ||l_j - l|| \leq \epsilon + \tau\}$. For a leader l an upper approximation of $N_\epsilon(l; \mathcal{D})$ is

$$\overline{N_\epsilon(l; \mathcal{D})} = \bigcup_{l \in \overline{\mathcal{L}_l}} followers(l) \tag{5.2}$$

It should be noted that

$$\underline{N_\epsilon(l; \mathcal{D})} \subseteq N_\epsilon(l; \mathcal{D}) \subseteq \overline{N_\epsilon(l; \mathcal{D})} \tag{5.3}$$

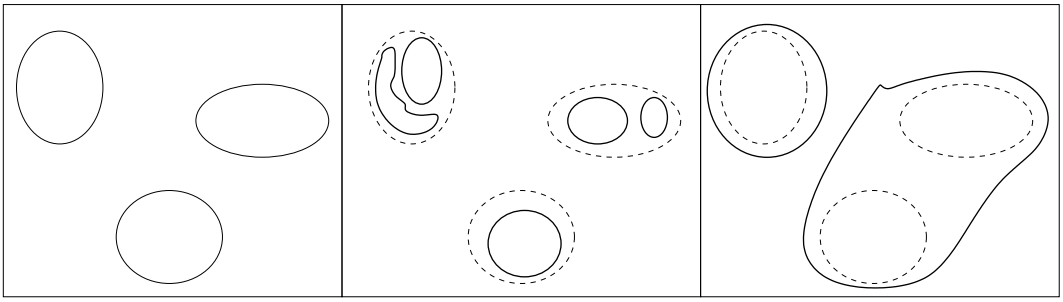
$$\lim_{\tau \to 0} \underline{N_\epsilon(l; \mathcal{D})} = \lim_{\tau \to 0} \overline{N_\epsilon(l; \mathcal{D})} = N_\epsilon(l; \mathcal{D}) \tag{5.4}$$

and

$$\text{card}(\underline{N_\epsilon(l; \mathcal{D})}) = \sum_{l \in \underline{\mathcal{L}_l}} count(l) \tag{5.5}$$

$$\text{card}(\overline{N_\epsilon(l; \mathcal{D})}) = \sum_{l \in \overline{\mathcal{L}_l}} count(l) \tag{5.6}$$

Let DBSCAN, for a given ϵ and *MinPts* finds a partition π of the data set $\mathcal{D}$. If the Algorithm 5.1 (*i.e.,* DBSCAN) is applied with the set of leaders $\mathcal{L}$ where $\text{card}(N_\epsilon(l; \mathcal{D}))$ is replaced by $\text{card}(\underline{N_\epsilon(l; \mathcal{D})})$, it will find a partition of $\mathcal{L}$. Let this is transformed into a partition of $\mathcal{D}$ by replacing each leader by its followers. Let this partition be $\underline{\pi}$. A dense leader according to $\text{card}(N_\epsilon(l; \mathcal{D}))$ can become non-dense according to $\text{card}(\underline{N_\epsilon(l; \mathcal{D})})$ and also it can become a noisy pattern. For example if π is as shown in Figure 5.2(a), then $\underline{\pi}$ could be as shown in Figure 5.2(b) where some of the clusters are further divided into sub-clusters and some of the patterns which are in a cluster can be treated as noisy patterns. Similarly, if instead of $\text{card}(N_\epsilon(l; \mathcal{D}))$ when $\text{card}(\overline{N_\epsilon(l; \mathcal{D})})$ is used we may get a partition $\overline{\pi}$ as shown in Figure 5.2(c). The set of all partitions of $\mathcal{D}$ with distance between two



(a) The partition found by DBSCAN, π

(b) The lower bound partition found by rough–DBSCAN, $\underline{\pi}$

(c) The upper bound partition found by rough–DBSCAN, $\overline{\pi}$

Figure 5.2: Lower and upper approximations of the partition.

partitions π_1 and π_2 defined as $d(\pi_1, \pi_2) = 1 - \textit{Rand-Index}(\pi_1, \pi_2)$ forms a metric space in which as $\tau \to 0$ both $\underline{\pi}$ and $\overline{\pi}$ will approach π. The similarity measure *Rand-Index* [Rand (1971)], [Hubert and Arabie (1985)] is described in Section 5.5.

An interpolated value between card$(\underline{N_\epsilon(l;\mathcal{D})})$ and card$(\overline{N_\epsilon(l;\mathcal{D})})$ can be taken as an approximation of card$(N_\epsilon(l;\mathcal{D}))$ to obtain a partition π' of the data set. For example, the average value of card$(\underline{N_\epsilon(l;\mathcal{D})})$ and card$(\overline{N_\epsilon(l;\mathcal{D})})$ can be used. But, the thesis proposes to use *rough-card*$(N_\epsilon(l;\mathcal{D}))$ as explained below.

Let $\mathcal{L}_l = \{l_j \in \mathcal{L} \mid ||l_j - l|| \leq \epsilon\}$.

$$\textit{rough-card}(N_\epsilon(l;\mathcal{D})) = \sum_{l \in \mathcal{L}_l} \textit{count}(l) \tag{5.7}$$

The leaders in $\overline{\mathcal{L}_l} - \underline{\mathcal{L}_l}$ are leaders like l_i and l_j as shown in Figure 5.3 which lie at the border of the hyper-sphere. Leaders like l_i has some of its followers outside of the hyper-sphere, whereas leaders like l_j which them self are outside of the hyper-sphere can have some of its followers inside the hyper-sphere. On average, since the distribution is assumed to be uniform over a small region like the hyper-sphere of radius ϵ, the number of leaders like l_i is same as the number of leaders like l_j. Hence by taking only leaders like l_i into account can give a better cardinality estimate which is called *rough-card*$(\ N_\epsilon(l;\mathcal{D})\ )$.

The proposed clustering method rough-DBSCAN is given in Algorithm 5.3.

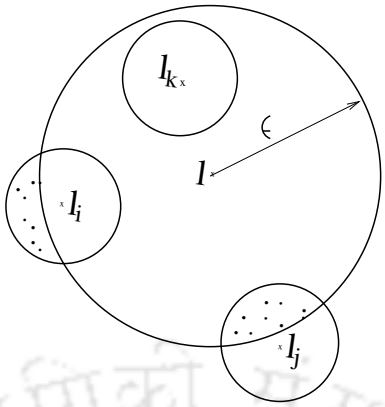


Figure 5.3: Different types of leaders.

Algorithm 5.3 Rough-DBSCAN($\mathcal{L}^\star$,ϵ, *MinPts*)

(i) Let $\mathcal{L}$ be the set of leaders which is extracted from $\mathcal{L}^\star$.
(ii) Do DBSCAN($\mathcal{L}$,ϵ,*MinPts*) by replacing $card(N_\epsilon(l;\mathcal{D}))$ with $rough\text{-}card(N_\epsilon(l;\mathcal{D}))$, where l is a leader in $\mathcal{L}$. Let the partition obtained be $\pi'_{\mathcal{L}}$.
(iii) Obtain π' from $\pi'_{\mathcal{L}}$ by replacing each leader l by $followers(l)$.
(iv) Output π' which is a partition of $\mathcal{D}$.

5.3.1 Rough-DBSCAN: Computational requirements

The space complexity of rough-DBSCAN is $O(n)$ where n is the data set size. This is same as that of DBSCAN. The time requirement of rough-DBSCAN is $O(n+k^2)$ where k is the number of leaders *i.e.,* $|\mathcal{L}|$. But the time requirement of DBSCAN is $O(n^2)$. Since it is guaranteed that $k \leq n$, and assuming that the data set is drawn from a bounded region of the feature-space, there is an upper-bound for k (see Theorem 2.4.2) which is a constant and is independent of n, hence the running time of rough-DBSCAN is $O(n)$. So it is guaranteed that rough-DBSCAN is faster than DBSCAN, and it is especially suitable for large data sets.

5.3.2 Relationship between DBSCAN and Rough-DBSCAN

This section analyzes the conditions required to be satisfied if rough-DBSCAN's result has to be the same as the DBSCAN's result.

For a pattern $x \in \mathcal{D}$, let *leader(x)* be the leader l such that $x \in$ *followers(l)*. Let the clustering output of DBSCAN be the partition of the data set denoted by $\pi_{\mathcal{D}}$, and that of rough-DBSCAN be denoted as $\pi'_{\mathcal{D}}$. In both of these cases, noisy patterns are grouped into a separate cluster so that $\pi_{\mathcal{D}}$ and $\pi'_{\mathcal{D}}$ are partitions of $\mathcal{D}$. The following are the requirements to be satisfied, if $\pi_{\mathcal{D}} = \pi'_{\mathcal{D}}$ has to be true. Let (x_1, x_2) be an arbitrary pair of patterns in $\mathcal{D} \times \mathcal{D}$, then

Requirement 1: If x_1 and x_2 are in distinct clusters according to $\pi_{\mathcal{D}}$ then *leader(x_1)* and *leader(x_2)* are also in distinct clusters according to $\pi'_{\mathcal{D}}$.

Requirement 2: If x_1 and x_2 are in a cluster according to $\pi_{\mathcal{D}}$ then *leader(x_1)* and *leader(x_2)* are also in a cluster according to $\pi'_{\mathcal{D}}$.

There are two types of mistakes made by rough-DBSCAN because of which $\pi'_{\mathcal{D}}$ may not be the same as $\pi_{\mathcal{D}}$. These are:

1. *Density mistake*: A pattern x according to DBSCAN can be dense (or non-dense), but *leader*(x) according to rough-DBSCAN may be non-dense (or dense).

2. *Near-ness mistake*: We say that two patterns p and q are *transitively near* if there is a sequence of patterns $p, y_1, y_2, \ldots, y_m, q$ such that the distance between any two successive patterns is less than or equal to ϵ. If two patterns p and q are transitively near according to DBSCAN but *leader*(p) is not transitively near to *leader*(q) according to rough-DBSCAN (*i.e.,* there is no sequence of leaders *leader*(p),$l_1, l_2, \ldots$,*leader*(q) such that successive leaders are less than or equal to ϵ distance), then we say that there is a near-ness mistake.

If rough-DBSCAN method uses the Equation 5.7 to find the cardinality then according to the argument given in Section 5.3 it is not an unrealistic one to assume that there are no density mistakes when $\tau < \epsilon$.

The *Requirement 1* is satisfied easily as shown by the following Lemma.

Lemma 5 *Assuming that there are no density mistakes, if two arbitrary patterns x_1 and x_2 are in distinct clusters according to DBSCAN then they are in distinct clusters according to rough-DBSCAN also.*

Proof: Suppose that x_1 and x_2 are in distinct clusters, but *leader*(x_1) and *leader*(x_2) are in a same cluster. We show that this leads to a contradiction.

Two patterns p and q are in a cluster according to DBSCAN, if there is a sequence of patterns $s : (p, y_1, y_2, \ldots, y_m, q)$ in $\mathcal{D}$ which satisfies, (i) if $m = 0$ then at least one of p or q is dense and $||p - q|| \leq \epsilon$, (ii) All of the patterns $y_1, \ldots, y_m$ are dense and distance between any two successive patterns in the sequence s is less than or equal to ϵ.

x_1 and x_2 are in a same cluster according to rough-DBSCAN means that there is a sequence $s' : (x_1, l_1, l_2, \ldots, l_m, x_2)$ which satisfies the above mentioned requirements. Then, since $l_1, l_2, \ldots, l_m$ are also patterns in $\mathcal{D}$, according to DBSCAN x_1 and x_2 must be in a cluster. This is the required contradiction. □

The analysis of the *Requirement 2* is a more involved one. We use the term called *Nearness-Index* which is defined below.

Definition 6 Nearness-Index: *For a set of patterns S, we say that δ-nearness is true if for every pair of patterns p and q in S there is a sequence of patterns $p, y_1, y_2, \ldots, y_m, q$ such that (i) the distance between any two successive patterns in the sequence is less than or equal to δ, (ii) if $m = 0$, at least one of p and q is dense, and (iii) if $m > 0$, $y_1, y_2, \ldots, y_m$ are dense. Then the* Nearness-Index *of the set S is the minimum δ value such that δ-nearness is true.*

Lemma 7 *If the distance between two patterns x_1 and x_2 is a then the distance between* leader(x_1) *and* leader(x_2) *is less than or equal to $a + 2\tau$ where τ is the threshold distance used while deriving the leaders.*

Proof: Since the patterns are assumed to be drawn from a metric space, the distance measure satisfies triangle inequality. Let the distance between two patterns x_1 and x_2 is denoted by $dist(x_1, x_2)$. We have $dist(x_1, leader(x_1)) \leq \tau$. So, $dist(x_2, leader(x_1)) \leq a + \tau$ according to the triangle inequality. We have $dist(x_2, leader(x_2)) \leq \tau$. So, $dist(leader(x_1), leader(x_2)) \leq a + 2\tau$. □

Lemma 8 *Let two patterns x and y are in a cluster C according to DBSCAN where C is a cluster which is other than the cluster consisting of the noisy patterns. If the Nearness-Index for C is less than or equal to $\epsilon - 2\tau$, then x and y are also in a cluster according to rough-DBSCAN provided there are no density-mistakes.*

Proof: x and y are in the cluster C and the Nearness-Index for C is less than or equal to $\epsilon - 2\tau$ means that there is a sequence of patterns $x, z_1, z_2, \ldots, z_p, y$ in C satisfying (i) distance between any two successive patterns in the sequence is less than or equal to $\epsilon - 2\tau$, (ii) if $p = 0$, at least one of x and y is dense, and (iii) if $p > 0$, $z_1, z_2, \ldots, z_p$ are dense. Consider the sequence leader$(x), l_1, l_2, \ldots, l_p,$leader$(y)$ where $l_i = leader(z_i)$ for $i \in \{1, 2, \ldots, p\}$. According to Lemma 7 in this sequence successive patterns are at a distance less than or equal to ϵ. Since, we assume that there are no density mistakes leader(x) and leader(y) must be in the same cluster according to rough-DBSCAN. This in turn means that x and y are in the same cluster according to rough-DBSCAN. □

Theorem 5.3.1 Rough-*DBSCAN's result is same as the DBSCAN's result if the following conditions are true.*

1. *There are no density-mistakes.*
2. *Each cluster of patterns has the Nearness-Index of at-most $\epsilon - 2\tau$.*

Proof: Directly follows from Lemma 5 and Lemma 8. □

Theorem 5.3.1 states a sufficient condition for DBSCAN and rough-DBSCAN to give the same result. But this is not a necessary condition. That is, even when these conditions are not met it is possible in certain cases that rough-DBSCAN's result is same as DBSCAN's result.One simple reason for this is that the two mistakes *viz.,* density-mistakes and nearness-mistakes can cancel each other.

For $\epsilon - 2\tau \geq 0$, it is required that $\tau \leq \epsilon/2$.

Nearness-Index is related to the distance between neighboring patterns in a cluster. If the distance between neighboring patterns in a cluster is small then it is likely that the rough-DBSCAN's result is closer to the DBSCAN's result. As the dataset size increases,

the distance between neighboring patterns in a cluster reduces and hence the *Nearness-Index* reduces. Suppose the probability density required to make a cluster dense be t. Let there are n patterns in the data set. Then in a cluster in a hyper-sphere of radius ϵ the expected number of patterns present is $k \geq tnV$ where V is the volume of the hyper-sphere. Assuming that within this hyper-sphere the density is uniform, on average distance between two neighboring patterns in the hyper-sphere is less than or equal to $\epsilon/k = \epsilon/tnV$. This is nothing but the *Nearness-Index* on average. We require *Nearness-Index* $\leq \epsilon - 2\tau$. That is, $\epsilon/tnV \leq \epsilon - 2\tau$. That is, on average, for $n \geq \epsilon/tV(\epsilon - 2\tau)$ the required condition can be satisfied. So, it is more likely that, for large data sets the rough-DBSCAN's result is same as the DBSCAN's result.

5.4 Further improvements to Rough-DBSCAN

While the thesis has a limited scope of analyzing the leaders as prototypes and its usage in a density based clustering scheme, it is worth noting that the time requirement of *Rough*-DBSCAN can be further reduced by: (i) building an index over the set of leaders like *R-tree* index [Guttman (1984)], *KD-tree* index [Bentley (1975)], etc, or (ii) clustering the leaders set $\mathcal{L}$ by using the leaders method with a larger threshold value which gives a coarser set of leaders which can be used to reduce the search time as described in the earlier version of the paper [Viswanath and Pinkesh (2006)]. Whereas the performance (in terms of results obtained by the *Rough*-DBSCAN when compared with that of the DBSCAN) can be improved if *card*($N_\epsilon(l; \mathcal{D})$) is a better estimate. For this the thesis proposes yet another clustering method called rough-fuzzy DBSCAN in Section 5.4.1. To reduce the computational requirements the thesis proposes to use rough-fuzzy leaders and subleaders (as explained in Chapter 4) in Section 5.4.2.

5.4.1 Rough-fuzzy DBSCAN

In rough-fuzzy DBSCAN, each leader is associated with a rough-fuzzy count value which are used in finding a better estimate for *card*($N_\epsilon(l; \mathcal{D})$). A rough fuzzy count values for

the leaders are obtained by exploring the rough-fuzzy uncertainty present in the leaders clustering method. This method is called rough-fuzzy counted leaders method which works similar to the rough-fuzzy weighted leaders method as explained in section 4.3.1. But, it has some differences in calculating count for the leaders which is as follows. If a pattern x is within the lower threshold of a leader then rough fuzzy count of a leader is updated using *count(l)=count(l)*+1, else if x is within the upper threshold of p leaders then the rough-fuzzy membership values for these p leaders are calculated as explained in section 4.2.2 and each of the p leaders rough-fuzzy count is updated. Suppose μ is the rough-fuzzy membership value of x to a leader l among the p leaders then the rough fuzzy count of the leader l is updated using *count(l)=count(l)*+μ. If $p = 0$ then x becomes a new leader with its rough-fuzzy count value set to 1. Hence, the Rough-fuzzy DBSCAN uses rough-fuzzy count values while estimating *card*($N_\epsilon(l; \mathcal{D})$). The space and time complexities to find *card*($N_\epsilon(l; \mathcal{D})$) is $O(|\mathcal{L}|)$ where $|\mathcal{L}|$ is the number of leaders.

5.4.2 Rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method

The computational requirements and the performance of rough-fuzzy DBSCAN mainly depends on the number of leaders. Smaller the number of leaders are in favor of reducing the computational burden whereas in other hand larger the number of leaders leads the solution of rough-fuzzy DBSCAN closer to that of DBSCAN. A method described in this section balances between these two where rough-fuzzy counted leaders-subleaders are used in finding *card*($N_\epsilon(l; \mathcal{D})$). Rough-fuzzy counted leaders-subleaders method works similar to that of rough-fuzzy weighted leaders-subleaders method as explained in section 4.4.1 and also it has some differences in calculating count for the leaders and sub-leaders. There are three possibilities for updating the rough-fuzzy count of the leaders. That is, (1) if a pattern x is within the lower threshold (L_T) of a leader l then it is assigned to the leader l with a rough-fuzzy membership value 1 (*i.e.*, $\mu = 1$) and also the count of the leader l is updated by *count(l)=count(l)*+μ where μ is 1. (2) Else if a pattern x is within the upper threshold (U_T) of p leaders then each of these p leaders updates their rough-fuzzy

count as follows. Suppose $\mu \in [0,1]$ is the rough fuzzy membership value of x to one of the leader l among the p leaders then the rough-fuzzy count of leader l is updated by *count(l)=count(l)+*μ. (3) Else pattern x becomes a leader and a sub-leader with an initial count of 1. Similarly the same pattern x is assigned to its sub-leaders. But, the count of sub-leaders are updated as *count(sl)=count(sl)+*$\mu \times s\mu$ where μ and $s\mu$ are the rough-fuzzy membership values of x to the leader l and the sub-leader sl (for which the pattern x belongs to) respectively.

Rough-fuzzy counted leaders-subleaders are derived in a single scan of the data set. Let $\underline{\mathcal{L}_l} = \{l_j \in \mathcal{L} \mid ||l_j - l|| < \epsilon - U\_T\}$ and $\overline{\mathcal{L}_l} = \{l_j \in \mathcal{L} \mid ||l_j - l|| < \epsilon + U\_T\}$. Now, the method which uses rough-fuzzy counted leaders-subleaders in finding *card*($N_\epsilon(l;\mathcal{D})$) works as follows. It finds the $\underline{\mathcal{L}_l}$ and also finds a set of sub-leaders in $\overline{\mathcal{L}_l} - \underline{\mathcal{L}_l}$ such that $||sl - l|| \leq \epsilon$ and let it be $\mathcal{BZ}$. The *card*($N_\epsilon(l;\mathcal{D})$) is approximately found by using $\sum_{l \in \underline{\mathcal{L}_l}} count(l) + \sum_{sl \in \mathcal{BZ}} count(sl)$. The time complexity of the method to find the *card*($N_\epsilon(l;\mathcal{D})$) is $O(|\mathcal{L}| + max\{|\mathcal{SL}|_1, \ldots, |\mathcal{SL}|_{|\mathcal{L}|}\})$ and the space complexity is $O(|\mathcal{L}| + \sum_{j=1}^{|\mathcal{L}|} |\mathcal{SL}|_j)$ where $|\mathcal{L}|$ is the number of leaders and $|\mathcal{SL}|_j$ is the number of sub-leaders of the j^{th} leader.

5.5 Experimental Results

Experimental studies are done with the following objectives.

1. To compare the clustering result obtained by rough-DBSCAN with that of DBSCAN.

2. To compare the time taken by rough-DBSCAN with that of DBSCAN.

3. To compare the clustering result obtained by using rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: rough-fuzzy counted leaders-subleaders method with that of rough-DBSCAN

4. To compare the time taken by rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: rough-fuzzy counted leaders-subleaders method with that of rough-DBSCAN.

The clustering result of DBSCAN and rough-DBSCAN are compared using the similarity measure *Rand-Index* [Rand (1971)], [Hubert and Arabie (1985)] which is described below.

For a n element data set let π and π' be two portions. Let a be the number of pairs of patterns in the data set which are present in a block of π and also present in a block of π'. That is, these pairs are grouped together according to π and also according to π'. Let b be the number of pairs in the data set which are not grouped in a block according to π and also according to π'. That is, these pairs are such that, each of them are grouped with a different block (cluster) according to π and also according to π'. Then

$$\text{Rand-Index}(\pi, \pi') = \frac{a+b}{\binom{n}{2}}$$

Rand-Index has a value between 0 and 1, with 0 indicating that two sets of partitions do not agree on any pair of patterns and 1 indicating that the two sets of partitions are exactly the same.

Experimental studies are done with two synthetic and five standard data sets. The details of the standard data sets are given in Table 2.1 and the details of the synthetic data sets are given below

The synthetic data sets are: (i) the Gaussian mixture data set, and (ii) the Banana data set. The Gaussian mixture data set is a two dimensional data set having 30000 patterns which is generated from the distribution (density) function $\frac{1}{3}p_1(x)+\frac{1}{3}p_2(x)+\frac{1}{3}p_3(x)$ where $p_i(x) = N(\mu_i, \Sigma_i)$ for $i = 1, 2$ and 3, where $\mu_1 = (0, 0)^T$, $\mu_2 = (6, 6)^T$, $\mu_3 = (6, -6)^T$,

$$\Sigma_1 = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \text{and} \quad \Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

That is, it is a three component mixture where each component is a Gaussian distribution. The other synthetic data set *i.e.,* the banana data set is also a two dimensional data having 4000 patterns consisting of two banana shaped clusters as shown in Figure 5.7.

Experiments are conducted on a PC with an *Intel P4 processor (3.2GHz)* with *512 MB RAM*. The comparisons are shown in two plots for each data set. Each data set has

four plots out of which first two plots describes the comparison of rough-DBSCAN with DBSCAN and next two plots describes the comparison of proposed methods, *i.e.,* rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN:using rough-fuzzy counted leaders-subleaders method.

The first plot shows the *Rand-Index* values along the vertical axis for varying data set sizes along the horizontal axis. The second plot shows the execution time along the vertical axis (log scale is used, since there is too much difference between the values) for varying data set sizes along the horizontal axis. The threshold parameter τ used in the leaders clustering method is also varied. For the DBSCAN and rough-DBSCAN the parameters ϵ and *MinPts* used are same. But, as the data set size is varied the *MinPts* is also varied proportionately (so that the apparent cut-off density threshold that distinguishes dense points from non-dense points remains the same).

The third plot shows the *Rand-Index* values along the vertical axis for various threshold values along the horizontal axis. The fourth plot shows the execution time along the vertical axis for various threshold values along the horizontal axis. Note that rough-DBSCAN and rough-fuzzy DBSCAN use a threshold value which is the second value in the plots. The results are summarized below.

5.5.1 The Gaussian mixture data set

The Figure 5.4 is a pictorial representation of the DBSCAN's result and the rough-DBSCAN's results for 1000 randomly chosen patterns. The rough-DBSCAN's results are shown when the leaders are derived using two different threshold values *viz.,* $\tau = 0.23$ and $\tau = 0.02$. From the Figure 5.4 it can be seen that for a larger τ the DBSCAN's result and the rough-DBSCAN's result deviates considerably, but when the τ value is lowered the difference is also low.

The Figure 5.5 shows the comparison between *Rand-Index* values and execution time(in seconds) for varying data set sizes and τ values. The τ values used (*i.e.,* the leaders threshold value to derive the leaders) are $0.08, 0.06, 0.04, 0.02$, the ϵ is chosen as 0.12. The parameter *MinPts* is 10 when the data set size is 3000, and it is varied proportionately

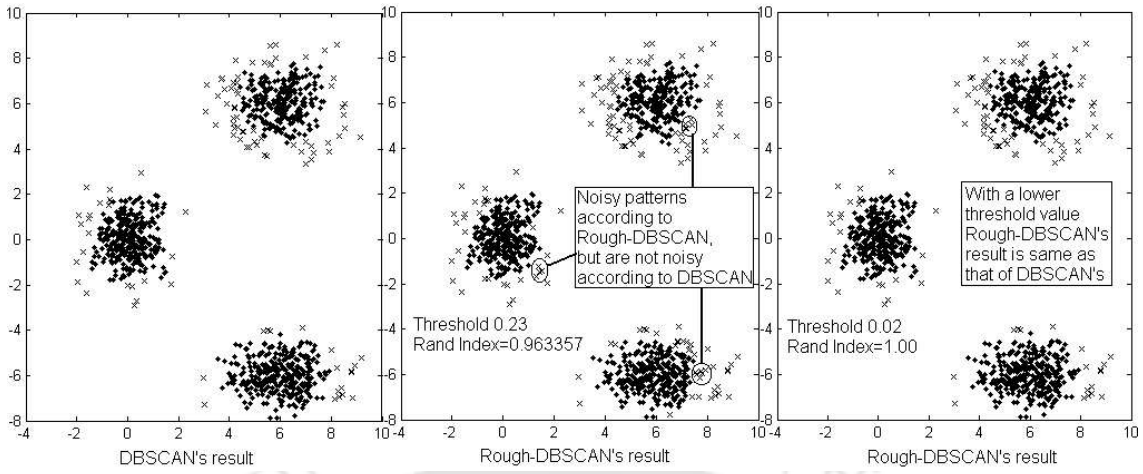


Figure 5.4: Synthetic:Gaussian mixture Data Set Representation for 1000 patterns

with the data set. For the threshold $\tau = 0.02$, the *Rand-Index* value is 0.991642 over the whole data set and the execution time of rough-DBSCAN is less than 3% that of the DBSCAN. It is observed from the plots that when the number of patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

The Figure 5.6 shows rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN because it resolves the rough-fuzzy uncertainty and uses rough-fuzzy count value to estimate density. For upper threshold 0.06 and upper subthreshold 0.02, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders method takes almost half of the time of rough-DBSCAN or rough-fuzzy DBSCAN and also its performance is almost equal to other two methods.

5.5.2 The banana data set

The banana data set which consists of two banana shaped clusters is shown in Figure 5.7. The experimental results are summarized in Table 5.1. For $\tau = 8$ the results of DBSCAN and rough-DBSCAN are considerably different and the Figure 5.7 shows some of the patterns which are seen by DBSCAN as noise but not by rough-DBSCAN. As the τ value is reduced the rough-DBSCAN's result approaches to that of the DBSCAN's and with $\tau = 2$, the rough-DBSCAN gives the same result as that of the DBSCAN, but execution

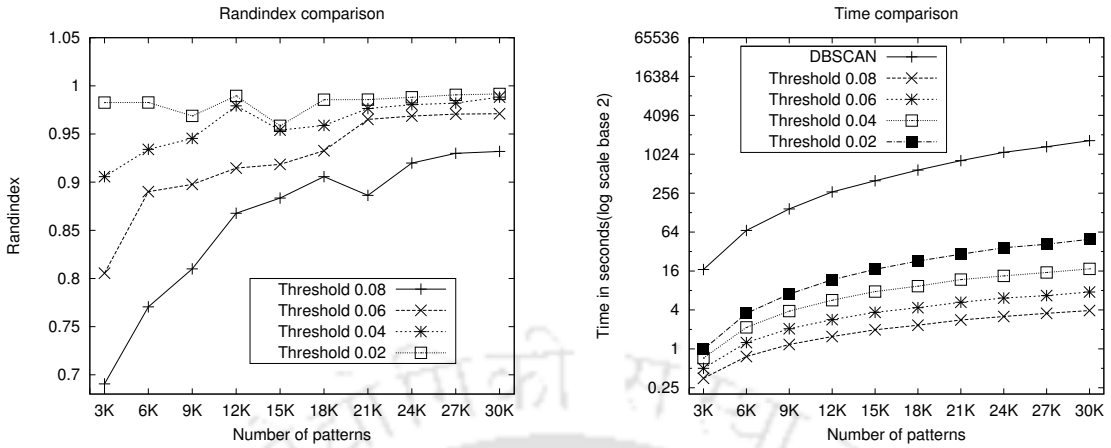


Figure 5.5: Synthetic:Gaussian mixture Data Set

τ	Rand Index	DBSCAN's execution time(s)	*Rough*-DBSCAN's execution time(s)
8	0.981	26.42	0.079
4	0.993	26.42	0.237
2	1	26.42	0.711

Table 5.1: Synthetic:Banana Data Set

time of rough-DBSCAN is less than 3% to that of the DBSCAN.

5.5.3 The Pendigits data set

The Figure 5.8 shows the results obtained for the Pendigits data set. The ϵ value used is 70. The *MinPts* used when the data set size is 1000 is 4. The τ values used are $30, 25, 20, 15$.

With $\tau = 15$, the rough-DBSCAN's result is very close (Rand-Index = 0.999988) to that of the DBSCAN, but has the execution time of less than 4% to that of the DBSCAN, when the whole data set is used. Further, it can be observed that when the number of patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

The Figure 5.9 shows rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN because it resolves the rough-fuzzy uncertainty and uses rough-fuzzy

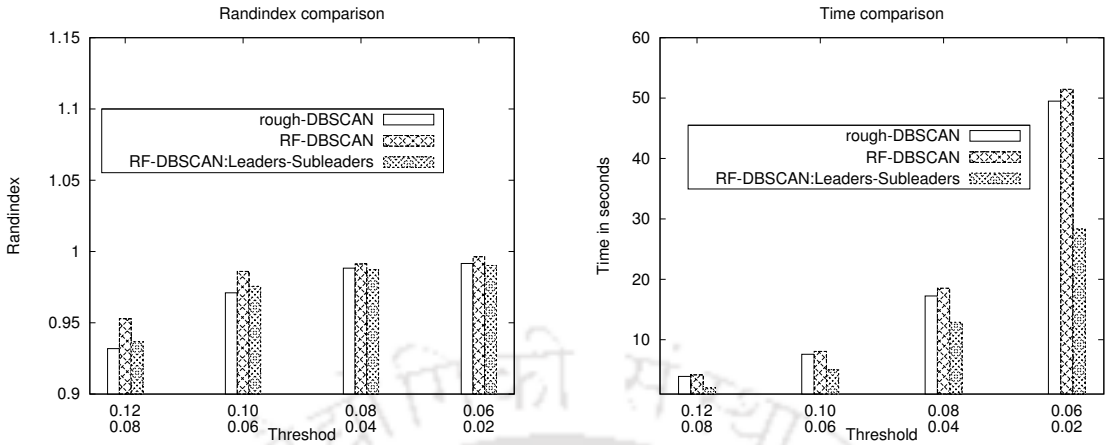


Figure 5.6: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the Synthetic:Gaussian mixture data set

count value to estimate density. For upper threshold 20 and upper subthreshold 15, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders method takes 80% of the time of rough-fuzzy DBSCAN and also its performance is almost equal to other two methods.

5.5.4 The Shuttle data set

The Figure 5.10 summarizes the results obtained for the Shuttle data set. The τ values used are $0.02, 0.01, 0.005, 0.001$. The ϵ value used is 0.06. The parameter *MinPts* when the data set size is 5000 is taken to be 20. It can be seen that when $\tau = 0.001$ both rough-DBSCAN and DBSCAN gives the same result (Rand-Index = 1), but rough-DBSCAN's execution time is less than 0.07% of that of DBSCAN's when the entire data set is used. It is observed from the plots that when the number of patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

For threshold value 0.02, the Figure 5.11 shows the rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN. For upper threshold 0.02 and upper sub-threshold 0.001, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders method takes 70% time of rough-fuzzy DBSCAN and also its performance is almost equal

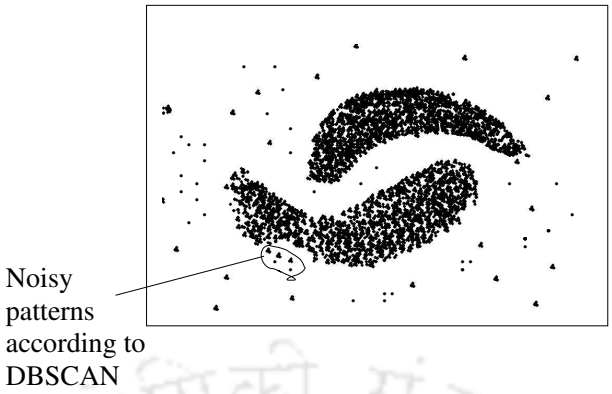


Figure 5.7: Synthetic:Banana Data Set.

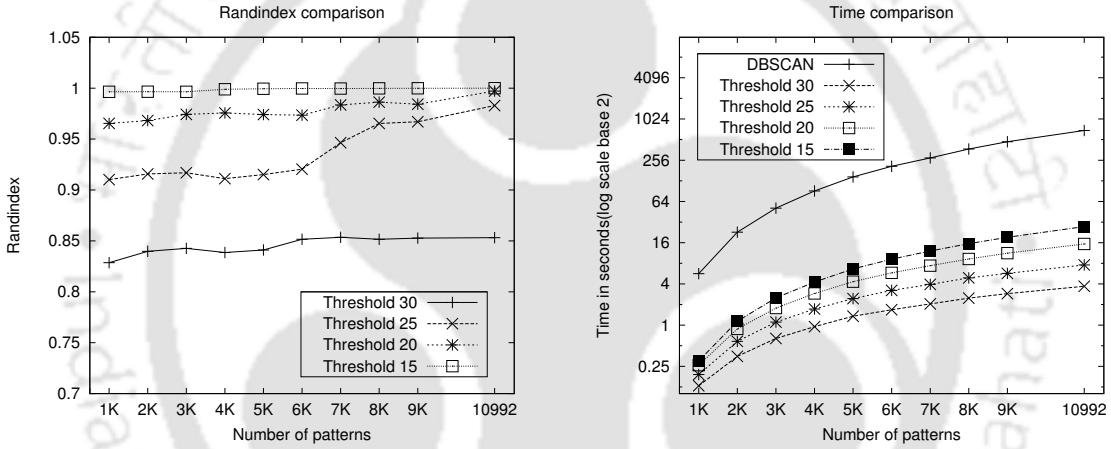


Figure 5.8: Pendigits Data Set

to other two methods.

5.5.5 The Letter data set

The experimental results with the Letter data set is summarized in Figure 5.12. The τ values used are $0.4, 0.35, 0.3, 0.25$. The ϵ value used is 1.2. The parameter *MinPts* when the data set size is 2000 patterns is 8. It can be seen that the Rand-Index is 0.959704 when the entire data set is used with $\tau = 0.25$, but the time taken by rough-DBSCAN is less than 0.6% of that by DBSCAN. Further, it is observed that when the number of

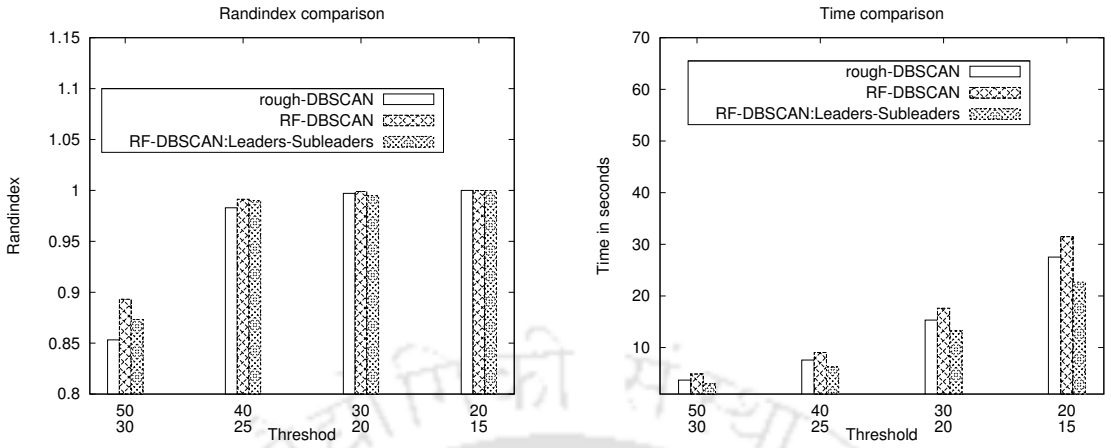


Figure 5.9: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the pendigits Data Set

patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

For threshold values 0.7, 0.6, the Figure 5.13 shows the rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN. For upper threshold 0.4 and upper subthreshold 0.25, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders method takes 60% time of rough-fuzzy DBSCAN and also its performance is almost equal to other two methods.

5.5.6 The A8a Data Set

The experimental results with the a8a set is summarized in Figure 5.14. The τ values used are $2.0, 1.8, 1.6, 1.4$. The ϵ value used is 5.0. The parameter *MinPts* when the data set size is 2000 patterns is 8. It can be seen that the Rand-Index is 0.979421 when the entire data set is used with $\tau = 1.4$, but the time taken by rough-DBSCAN is less than 0.2% of that by DBSCAN. Further, it is observed that when the number of patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

For threshold values 2.0, the Figure 5.15 shows the rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN. For upper threshold 0.4 and upper subthreshold 0.25, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders

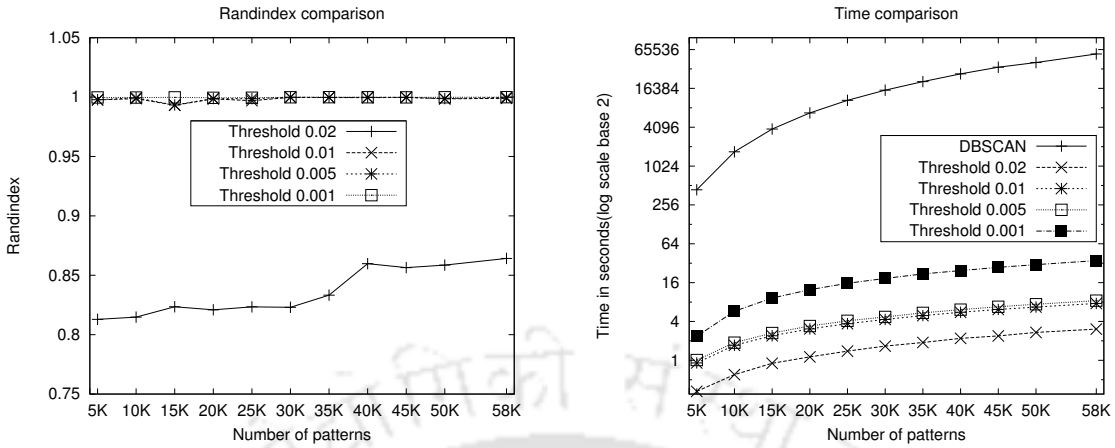


Figure 5.10: Shuttle Data Set

method takes 60% time of rough-fuzzy DBSCAN and also its performance is almost equal to other two methods.

5.5.7 The A9a Data Set

The experimental results with the Letter data set is summarized in Figure 5.16. The τ values used are $2.0, 1.8, 1.6, 1.4$. The ϵ value used is 5.0. The parameter *MinPts* when the data set size is 5000 patterns is 16. It can be seen that the Rand-Index is 0.993421 when the entire data set is used with $\tau = 1.4$, but the time taken by rough-DBSCAN is less than 0.2% of that by DBSCAN. Further, it is observed that when the number of patterns are increasing rough-DBSCAN's result is approaching to the DBSCAN's result.

For threshold values 2.0, the Figure 5.17 shows the rough-fuzzy DBSCAN performs better when compared with rough-DBSCAN. For upper threshold 1.8 and upper sub-threshold 1.4, the rough-fuzzy DBSCAN: using rough-fuzzy counted leaders- subleaders method takes 50% time of rough-fuzzy DBSCAN and also its performance is almost equal to other two methods.

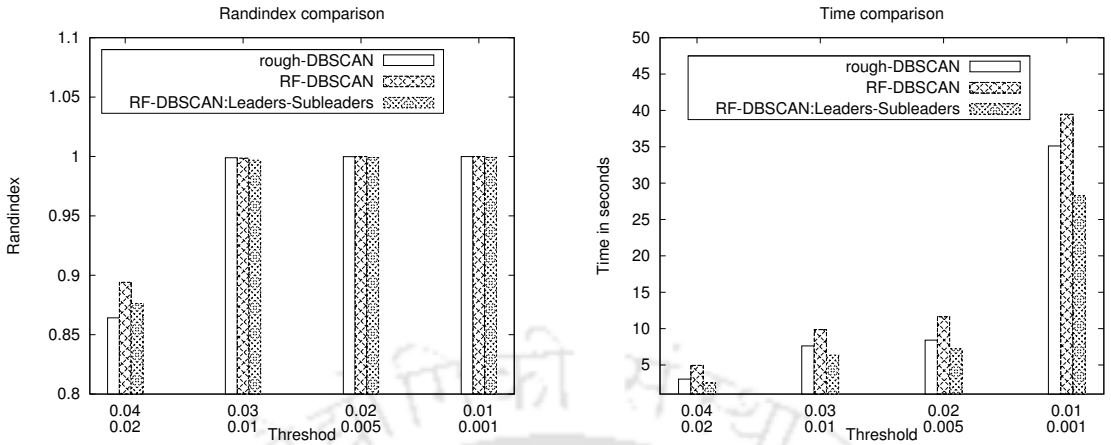


Figure 5.11: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the shuttle data Set

5.6 Conclusions

The chapter presented a novel hybrid clustering scheme to get density based arbitrary shaped clusters. The method is called rough-DBSCAN which first derives the prototypes called leaders by using the leaders clustering method and then uses these prototypes in deriving the density based clusters. The leaders clustering method is modified to store the number of followers each leader has which enables in estimating the densities. The proposed method is analyzed using the rough set theory. Theoretically some properties of the leaders and relationship between DBSCAN and rough-DBSCAN are established. Further, an improvement over *rough*-DBSCAN called rough-fuzzy DBSCAN is proposed which uses rough fuzzy membership values while counting the number of followers a leader has and hence can achieve a better result than rough-DBSCAN. To reduce the computational requirements further rough-fuzzy DBSCAN using rough-fuzzy counted leaders-subleaders method is proposed. Theoretically and experimentally the methods are analyzed and are shown to give good clustering outputs (when compared with that of DBSCAN's output) in a relatively very lesser time than that taken by the method DBSCAN. Asymptotically, under certain assumptions, it is shown that the proposed methods *viz.,* rough-DBSCAN

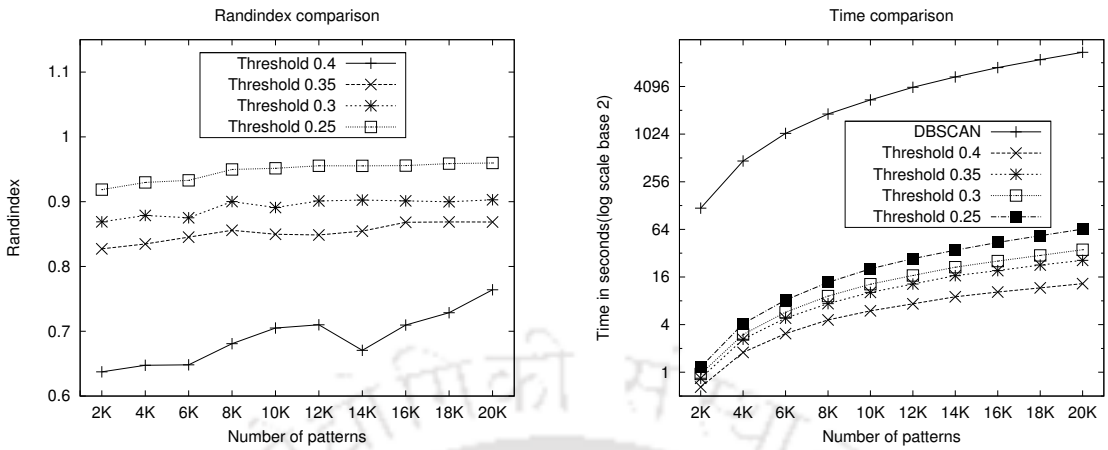


Figure 5.12: Letter Data Set

and rough-fuzzy DBSCAN has a linear time requirement in terms of the number of input patterns (whereas DBSCAN has a quadratic time requirement). Hence the proposed methods are suitable to work with large data sets as in data mining applications.

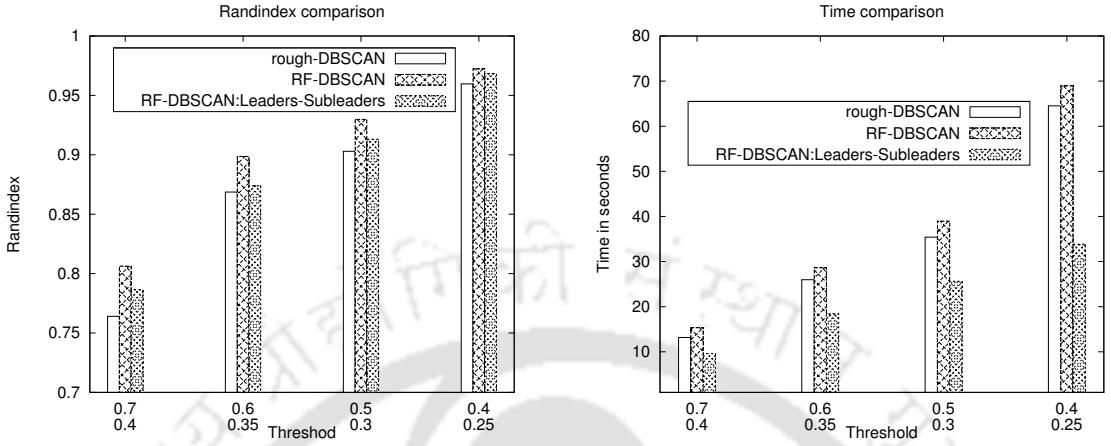


Figure 5.13: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the letter data Set

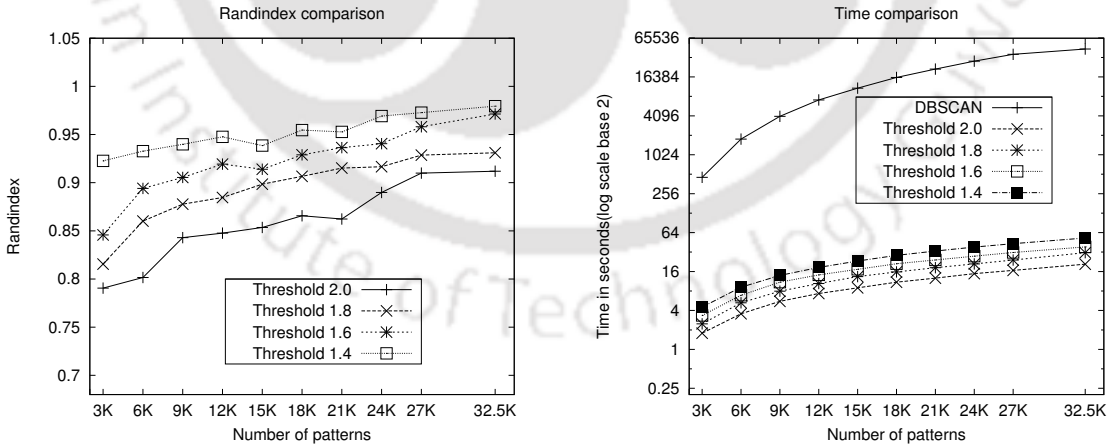


Figure 5.14: A8a Data Set

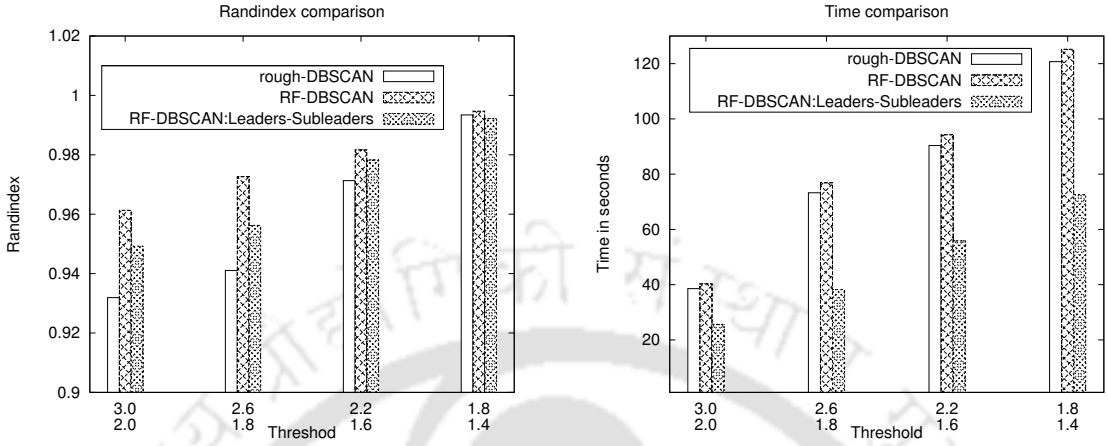


Figure 5.15: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the A8a data Set

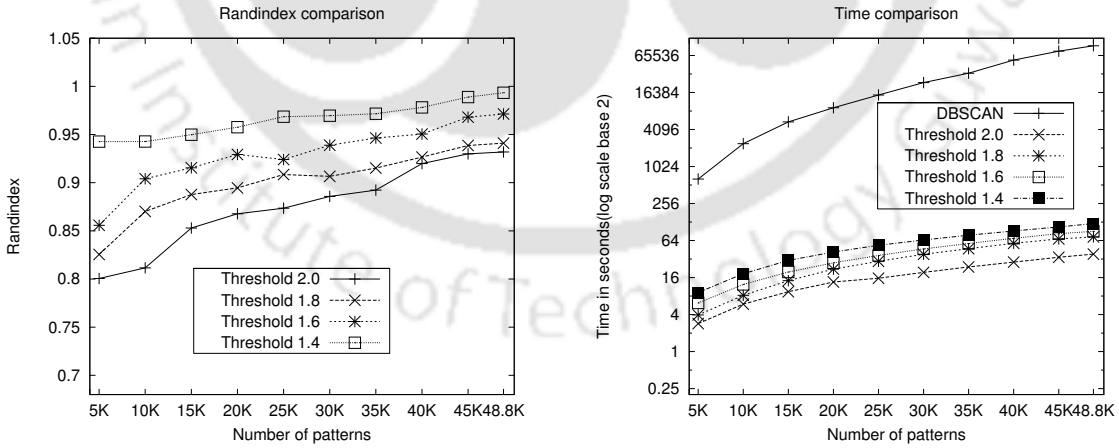


Figure 5.16: A9a Data Set

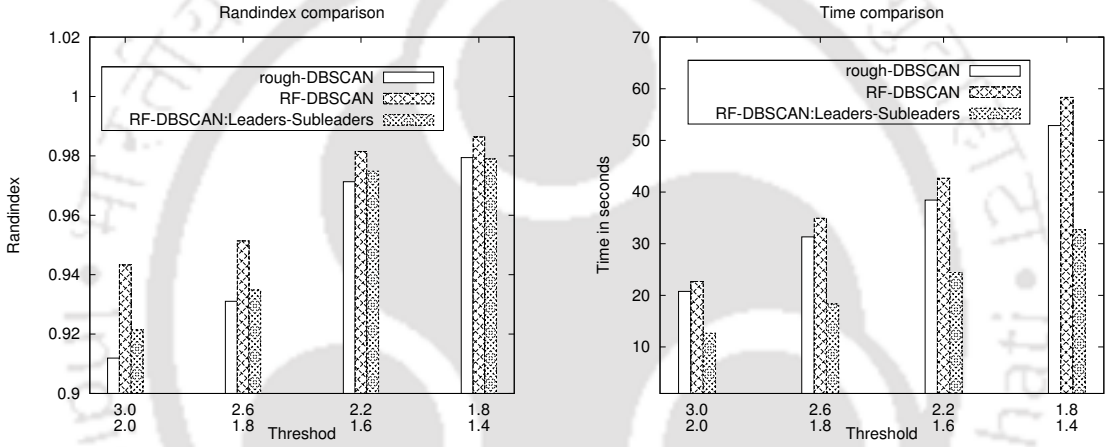


Figure 5.17: (i) Performance comparison, (ii) Execution time comparison of rough-DBSCAN, rough-fuzzy DBSCAN and rough-fuzzy DBSCAN: using rough-fuzzy counted leaders-subleaders method for the A9a data Set

Chapter 6

Summary and Future work

Non-parametric classification methods like nearest neighbor based classifiers and clustering methods like DBSCAN are more general since it does not assume that the data is from a parametric form. Also it shows good performance. Asymptotically, it is shown that the k-nearest neighbor classifier has similar performance as the Bayes classifier and DBSCAN can find consistent arbitrary shaped clusters along with noisy outliers detection. But these methods has high computational requirements since it needs to store the entire data set and search the entire data set. If there are n patterns in the training set then nearest neighbor searching will take $O(n)$ time for each search and the DBSCAN method to find clusters takes $O(n^2)$ time.

The thesis presented hybrid approaches to overcome the computational requirements problem. It is proposed to first derive the prototypes called leaders which can be found quickly in a single scan of the data set, and then to use these leaders in the nearest neighbor based classifier and or to derive density based clusters as done in the DBSCAN. The leaders clustering method is described in detail and some of the properties of leaders like a lower bound and an upperbound on the number of leaders are theoretically established.

But leaders alone cannot be used in getting density information at a point in the feature space. This is because the density information that is present in the data set is lost while deriving the leaders. It is formally established that nearest leader based classifier, some times is nothing but a random classifier (where the class label is randomly

guessed). Hence it is proposed to derive additional information along with leaders which enables to find the density. For classification methods, a weight is assigned to each leader, and for clustering methods a count (a non-negative integer) is associated with each leader which measures relative strength of each leader. To further improve the methods, rough-fuzzy membership values are used with leaders.

The proposed methods are: (i) the weighted k-nearest leader classifier, (ii) the rough-fuzzy weighted k nearest classifier, (iii) rough-DBSCAN and (iv) rough-fuzzy DBSCAN. Experimentally these are compared with the conventional methods using various standard and synthetic data sets and are shown to achieve similar results as achieved by the conventional k nearest neighbor classifier and DBSCAN. But the proposed methods are very fast than their conventional counterparts. Especially the proposed methods are scalable to work with large data sets as those in data mining applications.

Some future directions of research are to build an index like R-tree, KD-tree over the set of leaders which can further reduce the time requirements. Other future direction is to extend the proposed methods to dynamically varying data sets like data streams.

Bibliography

D. W. Aha. Editorial. *Artificial Intelligence Rev.,special issue on lazy learning*, 11: 7–10, 1997.

D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

Ethem Alpaydin. Voting over multiple condensed nearest neighbors. *Artificial Intelligence Review*, 11:115–132, 1997.

R. B. Altman and J. M. Dugan. Defining bioinformatics and structural bioinformatics. *Methods Biochem Anal*, 44:3–14, 2003.

V. S. Ananthanarayana, M. N. Murty, and D. K. Subramanian. An incremental data mining algorithm for compact realization of prototypes. *Pattern Recognition*, 34:2249–2251, 2001.

R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 485–492, New York, NY, USA, 2006. ACM.

Fabrizio Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

S. Asharaf and M. Narasimha Murthy. A rough fuzzy approach to web usage categorization. *Fuzzy sets and Systems*, 148(1):119–129, 2004.

S. Asharaf and M. Narasimha Murty. An adaptive rough fuzzy single pass algorithm for clustering large data sets. *Pattern Recognition*, 36(12):3015–18, 2003.

S. Asharaf, S. K. Shevada, and M. N. Murty. Rough support vector clustering. *Pattern Recognition*, 38:1779–1783, 2005.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

J. C. Bezdek, T. R. Reichherzer, G. S. Lim, and Y. Attikiouzel. Multiple-prototype classifier design. *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):67–79, 1998.

H. Bian and L. Mazlack. Fuzzy-rough nearest-neighbor classification approach. In *22nd International Conference of the North American Fuzzy Information Processing Society*, pages 500–505. NAFIPS, 2003.

H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.

J. F. Cardoso. Blind signal seperation: statistical principles. *Proceedings of IEEE*, 9(10):2009–2025, 1998.

Stefan Chanas and Dorota Kuchta. Further remarks on the relation between rough and fuzzy sets. *Fuzzy Sets and Systems*, 47:391–394, 1992.

C. L. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 23(11):1179–1184, 1974.

C. C. Chibelushi, F. Deravi, and J. S. D. Mason. A review of speech-based bimodal recognition. *IEEE Transactions on Multimedia*, 4(1):23–37, 2002.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36 (3):287–314, 1994.

T. M. Cover and J. M. Van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 7 (9):657–661, 1977.

T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

Belur V. Dasarathy. Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos, California, 1991.

Supriya Kumar De. A rough set theoretic approach to clustering. *Fundamenta Informaticae*, 62(3-4):409–417, 2004.

P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, Englewood Cliffs, 1982.

D. Dubois and H. Prade. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems*, 17(2-3):191–209, 1990.

Richard O. Duda, Peter E.Hart, and David G. Stork. *Pattern Classification*. A Wiley-interscience Publication, John Wiley & Sons, 2 nd edition, 2000.

E. Yelow Cheu, Chee Keong Kwoh, and Zonglin Zhou. On the two-level hybrid clustering algorithm. In *Proceedings of International Conference on Artificial Intelligence in Science and Technology*, pages 138–142, 2004.

M. Ester, H. P. Kriegel, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd ACM SIGKDD*, pages 226–231, Portand, Oregon, 1996.

W. G. Fenton, T. M. McGinnity, and L. P. Maguire. Fault diagnosis of electronic systems using intelligent techniques: a review. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 31(3):269–281, 2001.

G. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.

S. Greco, M. Inuiguchi, and R. Slowinski. Fuzzy rough sets and multiple-premise gradual decision rules. *International Journal of Approximate Reasoning*, 41(2):179–211, 2006.

S. Greco, B. Matarazzo, and R. Slowinski. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 129(1):1–47, 2001.

R. Gutierrez-Osuna. Pattern analysis for machine olfaction: a review. *IEEE Sensors Journal*, 2(3):189–202, 2002.

A. Guttman. R-trees: a dynamic index structure for spatial searching. In *13th ACM SIGMOD Int. Conf. Management Data*, volume 2, pages 47–57, Boston, MA, 1984.

Isabelle Guyon and Andr Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

H. A. do Prado, P. M. Engel, and H. C. Filho. Rough clustering: An alternative to find meaningful clusters by using the reducts from a dataset. In *Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing*, pages 234–238. Springer-Verlag, 2002.

K. M. Hammouda and M. S. Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1279–1296, 2004.

P.E. Hart. The condensed nearest-neighbor rule. *IEEE Transactions on Information Theory*, IT-4:515–516, 1968.

E. Hjelm, A. W. M. Smes, and B. Kee Low. Face detection: a survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

I. Jagielska, C. Matthews, and T. Whitfort. An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems. *Neurocomputing*, 24:37–54, 1999.

A. K. Jain and B. Chandrasekharan. Dimensionality and sample size considerations in pattern recognition practice. In P.R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 835–855, North Holland, 1982.

A. K. Jain, R. C. Dubes, and C. C. Chen. Bootstrap technique for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:628–633, 1987.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

A. K. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs NJ, U.S.A., 1988.

Anil K. Jain, P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1): 4–37, 2000a.

Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000b.

R. Jain, S. Antani, and R. Kasturi. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35(4):945–965, 2002.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

F. O. Karray and C. De Silva. *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications.* Addison-Wesley, 2004.

J. M. Keller, M. R. Gray, and J. A. Givens. Fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems Man Cybernet*, 15(4):580–584, 1985.

Ron Kohavi. A study of crossvalidation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143. Morgan Kaufmann, 1995.

M. Kryszkiewicz. Rough set approach to incomplete information systems. *Information Sciences*, 112:39–49, 1998.

S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh. Learning pattern classification-a survey. *IEEE Transactions on Information Theory*, 44(6):2178–2206, 1998.

Yan Li, Simon C. K. Shiu, and Sankar K. Pal. Combining feature reduction and case selection in building CBR classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):415–429, 2006.

Cheng-Ru Lin and Ming-Syan Chen. Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):145–159, 2005.

P. Lingras and C. West. Interval set clustering of web users with rough k-means. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, 23(1):5–16, 2004.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

P. Maji and S. K. Pal. Rough-fuzzy c-medoids algorithm and selection of bio-basis for amino acid sequence analysis. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):859–872, 2007.

H. Michael. Strategies for the automated recognition of marks in forensic science. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 4709, pages 68–79. The International Society for Optical Engineering, 2002.

S. Mitra, H. Banka, and W. Pedrycz. Roughfuzzy collaborative clustering. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(4):795–805, 2006.

S. Mitra, S. K. Pal, and P. Mitra. Data mining in soft computing framework: a survey. *IEEE Transactions on Neural Networks*, 13(1):3–14, 2002.

R. A. Mollineda, F. J. Ferri, and E. Vidal. An efficient prototype merging strategy for the condensed 1-nn rule through class-conditional hierarchical clustering. *Pattern Recognition*, 35:2771–2782, 2002.

S. Nanda and S. Majumdar. Fuzzy rough sets. *Fuzzy Sets and Systems*, 45(2): 157–160, 1992.

P. Viswanath, Bidyut Kr. Patra, and V. Suresh Babu. *Some Efficient and Fast Approaches to Document Clustering.* To appear in Handbook of Research on Text and Web Mining Technologies,1G1, USA, 2008.

S. K. Pal and P. Mitra. Case generation using rough sets with fuzzy representation. *IEEE Transactions on Knowledge and Data Engineering*, 16(3):293–300, 2004.

Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11:341–356, 1982.

Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*, volume 112. MA: Kluwer, 1991.

G. Peters. Some refinements of rough k-means clustering. *Pattern Recognition*, 39 (8):1481–91, 2006.

R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for the selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

L. Rokach and O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):476–487, 2005.

Manish Sarkar. Fuzzy-rough nearest neighbor algorithms in classification. *Fuzzy Sets Syst.*, 158(19):2134–2152, 2007.

A. Schenker, M. Last, H. Bunke, and A. Kandel. Classification of web documents using a graph model. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 240–244, Washington, DC, USA, 2003. IEEE Computer Society.

Qiang Shen and Alexios Chouchoulas. A rough-fuzzy approach for generating classification rules. *Pattern Recognition*, 35(11):2425–2438, 2002.

S. C. K. Shiu, Y. Li, S. K. Pal, and J. N. K. Liu. A rough set-based case-based reasoner for text categorization. *International Journal of Approximate Reasoning*, 41(2):229–255, 2006.

Shailendra Singh and Lipika Dey. A new customized document categorization scheme using rough membership. *Applied Soft Computing*, 5(4):373–390, 2005a.

Shailendra Singh and Lipika Dey. A rough-fuzzy document grading system for customized text information retrieval. *Information Processing and Management*, 41:195–216, 2005b.

H. Spath. *Cluster Analysis Algorithms for Data Reduction and Classification*. Ellis Horwood, Chichester, UK, 1980.

C. Stanfill and D. Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1994.

V. Susheela Devi. *Optimal Prototype Selection for Efficient Pattern Classifiers*. Ph.D Thesis, Department of Electrical Engineering, IISc, Bangalore, 2000.

R. W. Swiniarski and A. Skowron. Rough set methods in feature selection and recognition. *Pattern Recognition Letters*, 24:833–849, 2003.

Y. Y. Tang, S. W. Lee, and C. Y. Suen. Automatic document processing - a survey. *Pattern Recognition*, 29(12):1931–1952, 1996.

S. Tsumoto. Automated extraction of medical expert system rules from clinical databases based on rough set theory. *Information Sciences*, 112:67–84, 1998.

V. Suresh Babu, P. Viswanath, and M. Narasimha Murty. *Scalable Non-parametric Methods for Large Data Sets*. To appear in Encyclopedia of Data Warehousing and Mining, 2nd Edition, Idea group inc., Montclair State University, USA, 2008.

V. Suresh Babu and P. Viswanath. Rough fuzzy weighted k-nearest leader classifier for large data sets. *To appear in Pattern Recognition*.

V. Suresh Babu and P. Viswanath. Novel document representation scheme using rough membership function for efficient document classification. In *First International Conference on Signal and Image Processing (ICSIP-06)*, volume 2, pages 572–576. IEEE Bangalore Section, 2006.

V. Suresh Babu and P. Viswanath. Weighted k-nearest leader classifier for large data sets. In *2nd International Conference on Pattern Recognition and Machine Intelligence (PReMI-07)*, volume 4815, pages 17–24. Springer Berlin, 2007.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Vladimir N. Vapnik. *Statistical Learning Theory*. A Wiley-interscience Publication, New York, John Wiley & Sons, 1998.

P. A. Vijaya, M. Narasimha Murty, and D.K. Subramanian. Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. *Pattern Recognition Letters*, 25:505–513, 2004.

P. Viswanath, P. Vinay Kumar, V. Suresh Babu, and M. Venkateswara Kumar. Generalized branch and bound algorithm for feature subset selection. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, pages 214–218, Washington, DC, USA, 2007. IEEE Computer Society.

P. Viswanath and Rajwala Pinkesh. l-DBSCAN : A fast hybrid density based clustering method. In *Proceedings of the 18th Intl. Conf. on Pattern Recognition (ICPR-06)*, volume 1, pages 912–915, Hong Kong, 2006. IEEE Computer Society.

K. E. Voges, N. K. Pope, and M. R. Brown. Cluster analysis of marketing data examining on-line shopping orientation: A comparison of k-means and rough clustering approaches. In *Heuristics and Optimization for Knowledge Discovery*, pages 207–224. Idea Group, 2002.

K. E. Voges, N. K. Pope, and M. R. Brown. A rough cluster analysis of shopping orientation data. In *Australian and New Zealand Marketing Academy Conference*, pages 1625–1631, 2003.

I. Watson and F. Marir. Case-based reasoning: A review. *Knowledge Engineering Review*, 9(4), 1994.

G. Wilfong. Nearest neighbor problems. *Computational Geometry and Applications*, 2(4):383–416, 1992.

D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2:408–420, 1972.

D. R. Wilson and T. R. Martizen. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286, 2000.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

Y. Y. Yao. A comparative study of fuzzy sets and rough sets. *Information Sciences*, 109(1-4):227–242, 1998.

L. A. Zadeh. Fuzzy sets. *Information and control*, 8:338–353, 1965.

Tian Zhang, Raghu Ramakrishnan, and Micon Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD International Conference of Management of Data*, pages 103–114, 1996.