M.Tech. (Computer Science) Dissertation Series

Pattern Synthesis with Active Learning using Classifier Combination

a dissertation submitted in partial-fulfillment of
the requirement for theM.Tech. (Computer Science) degree of
the Indian Statistical Institute

By

Lokesh Kumar S

under the supervision of

Dr. Ashish Ghosh
Machine Intelligence Unit

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta-700 035

Certificate of Approval

This is to certify that this dissertation thesis titled "**Pattern Synthesis with Active Learning using Classifier Combination**" submitted by **Mr. Lokesh Kumar S**, in partial fulfillment of the requirements for the M.Tech. (Computer Science) degree of the Indian Statistical Institute, Kolkata, embodies the work done under my supervision.

A. Ghosh
12.7.05

Dr. Ashish Ghosh
Machine Intelligence Unit
Indian Statistical Institute
Kolkata.

Acknowledgements

I take immense pleasure in thanking Dr. Ashish Ghosh (Associate Professor, MIU, ISI) for providing me an interesting problem and guiding me throughout the dissertation. His pleasent and encouraging words have always kept my spirits up. I thank Mr. B. Uma Shankar (MIU, ISI) and Mr. B.L. Narayan (Research Fellow, MIU, ISI), for patiently hearing to my mathematical and statistical blunders and correcting me wherever necessary. I also thank Dr. Saroj Kumar Meher (Visiting Scientist, MIU, ISI), Anindya Sen, Praveen Tripathi, Pallab Mondal and Bijnan for having discussions, which helped me in understanding the concepts. Last but not the least, I thank my friends who helped me immeasurably in improving the presentation of the thesis.

Lokesh Kumar S

Contents

Abstract

In this dissertation, the problem of Pattern Synthesis in conjunction with Active Learning has been considered. Synthesising new patterns though initially appears to be hardly possible, the research community has been dealing with the inverse problem i.e., the data condensation problem fairly well. Based on this idea a novel pattern synthesis technique mostly mimicing the DBSCAN clustering algorithm has been proposed. The patterns created in this way have been used to the Active Learning process. Moreover, it is well known that in many situations a Classifier Combination, consistently outperforms a single best classifier, and when there are a number of classifiers of different nature, one can use the Query By Committee(QBC) approach for actively selecting the new patterns for further training. In this dissertation Active Learning using Classifier Combination coupled with Pattern Synthesis has been studied. Experimental evaluation also reveals that this is slightly better than the naive Passive Learning approach using Classifier Combination.

Chapter 1

Introduction

In section 1 of this chapter, we give a rather broad formulation of a learning problem, which encompasses many specific problems, the main ones being *pattern classification*, *regression estimation* and *density estimation*. In section 2 the notion of *risk minimization* has been introduced followed by the specialization to Pattern Classification in section 3. In sections 4 and 5, passive and active learning paradigms have been introduced.

1.1 The Learning Problem

The model of learning [V.N. Vapnik 1999] from examples can be described using three components:

1. A generator (G) of random vectors $X \in R^n$, drawn independently from a fixed but unknown distribution function $P(X)$.

2. A supervisor (S) who returns an output vector Y to every input vector X according to a conditional distribution function $P(Y|X)$, also fixed but unknown.

3. A learning machine (LM) capable of implementing a set of functions $f(X, \alpha)$,$\alpha \in \Lambda$, where Λ is a set of parameters.

The learning problem is concerned with choosing from the given set of functions $f(X, \alpha)$, $\alpha \in \Lambda$, the one which predicts the supervisor's response in the best possible way, the selection is based on a training set $\mathcal{T} = \{(X_i, Y_i) \mid \forall i = 1, 2, ..l\}$ of l random independent and identically distributed (i.i.d) observations drawn according to $P(X, Y) = P(X)P(Y|X)$.

1.2 Risk Minimization

In order to choose the best available approximation to the supervisor's response, one measures the loss or discrepancy $\mathcal{L}(Y, f(X, \alpha))$ between the response Y of supervisor to a given input X and the response $f(X, \alpha)$ provided by the learning machine. Consider the expected value of the loss given by the *risk functional*

$$\mathcal{R}(\alpha) = \int \mathcal{L}(Y, f(X, \alpha)) dP(X, Y). \tag{1.1}$$

The goal is to find the function, $f(X, \alpha_0)$, which minimizes the *risk functional* (over the class of functions $f(X, \alpha), \alpha \in \Lambda$) in the situation where the joint probability distribution function $P(X, Y)$ is unknown and the only available information is contained in the training set $\mathcal{T}$.

1.3 Problem of Pattern Classification

Let the supervisor's output Y take values in $\mathcal{O} = \{1, 2, \ldots, M\}$ and let $f(X, \alpha), \alpha \in \Lambda$, be a set of functions with the range $\mathcal{O}$. Consider following loss-function:

$$\mathcal{L}(Y, f(X, \alpha)) = \begin{cases} 0 & \text{if } Y = f(X, \alpha) \\ 1 & \text{if } Y \neq f(X, \alpha). \end{cases} \tag{1.2}$$

For this loss-function, the functional (1.2) provides the probability of classification error (i.e., when the answers given by supervisor and the answers given by function $f(.)$ differ). The problem, therefore, is to find the function which minimizes the probability of classification errors when probability measure is unknown, but the training data $\mathcal{T}$ is given.

1.4 Passive Learning

The learning paradigm in which only the training data $\mathcal{T}$ is used to find the function $f(.)$ that predicts the supervisor's response is called *Passive Learning*, and the algorithms based on this idea are called *Passive Learning Algorithms*

1.5 Active Learning

The learning paradigm in which the samples to be learnt are chosen, *say* biasedly based on the past learning (passive or active), interacting with "the teacher" is called *Active Learning*.

Formally, active learning studies the closed-loop phenomenon of a learner selecting actions or making queries that influence what data are added to its training set [Cohn et al., 1996].
The possible strategies for selecting the samples are

- 1. Use the problem domain knowledge and fix the domain $\mathcal{D}$ from which the patterns are originating.
  2. Randomly pick a sample from this domain $\mathcal{D}$.
  3. Decide whether or not it is useful for further learning and accordingly query for its label.
- 1. Instead of the complete domain consider a finite set(*pool*) of patterns.
  2. Select a sample from this pool, which is most informative/ambiguous, and query for its label.

The latter approach of Active Learning is more practical and commonly known as Pool Based Active Learning [Baram et al., 2004].

1.5.1 Pool Based Active Learning

At this stage let us see the formal definition of pool-based active learning [Baram et al., 2004]. Consider a general setting of a multi-class classification problem and given a pool $\mathcal{U} = \{X_1, X_2, \ldots, X_n\}$ of unlabeled patterns where each X_i is a vector in some space say $\mathcal{X}$, patterns are assumed to be i.i.d. according to some unknown fixed distribution $P(X)$. Each pattern X_i has a label $Y \in \mathcal{O}$ distributed according to some unknown conditional distribution $P(Y|X)$. At the start, lets assume that the learner has a very small repository of randomly selected training set $\mathcal{T}$. At each stage of the active-learning game, let $\mathcal{L}$ be the set of labeled instances already known to the learner. An active learner consists of a classifier learning algorithm $\mathcal{A}$, and a querying function $\mathcal{Q}$, which is a mapping $\mathcal{Q} : \mathcal{L} \times \mathcal{U} \rightarrow \mathcal{U}$. The querying function determines one unlabeled instance/pattern in $\mathcal{U}$ to be labeled by the teacher. On each trial $t = 1, 2, \ldots$ the active learner first applies $\mathcal{Q}$ to choose one unlabeled instance X from $\mathcal{U}$, the label Y of X is revealed and the pair (X, Y) is added to $\mathcal{L}$ and X is removed from $\mathcal{U}$. Then the learner applies $\mathcal{A}$ to induce a new classifier C_t using $\mathcal{L}$ (and possibly $\mathcal{U}$) as a training set and a new trial begins, etc. Thus, the active learner generates a sequence of classifiers $C_1, C_2, \ldots$. Clearly, the maximal number of trials is the initial size of $\mathcal{U}$.

Chapter 2

An Overview of Classifiers

In this chapter let us study few basic classifiers, which will be used later in designing a Classifier Combination that is explained in chapter 5. In section 2 an introduction to Classifier Combination is given.

2.1 Bayes Classifier

Theoretically this is the best known classifier; but the only drawback is that it makes the assumption that for the decision/learning problem at hand all the relevant probability values are known [Duda et al., 2000].

Let $P(\omega_i)$ denote the *a priori probability* of the class i, $\forall\ i = 1, 2, ..., M$ and let $P(X|\omega_i)$ be *the class conditional probability density* function, i.e the probability density function for X given it is in class i

Note that according to Bayes law,

$$P(\omega_i|X) = \frac{P(X|\omega_i)P(\omega_i)}{P(X)} \tag{2.1}$$

where

$$P(X) = \sum_{j=1}^{M} P(X|\omega_j)P(\omega_j). \tag{2.2}$$

The *a posteriori* probability $P(\omega_i|X)$ is the probability that the instance X belongs to class i. To minimize the classification error, the Bayes Decision Rule(BDR)
decides class i if $P(\omega_i|X) > P(\omega_j|X)\ \forall j \neq i$.

In practical situations where we do not have the knowledge about the proabability $P(X|\omega_i)$, we assume it to have a known parametric form eg. Gaussian and estimate the parameters, say using Maximum Likelihood Estimation (MLE). The classifier designed using MLE is called a Maximum

Likelihood Classifier (MLC). It it most common to assume $X|\omega_i \sim N(\mu_i, \Sigma_i)$, to design an MLC. There are other possibilities also.

2.2 k-Nearest Neighbors Classifier

According to this classifier given a test pattern X and the training set $\mathcal{T}$, it classifies X by assigning it the label which is most frequently represented among the k nearest training samples; in other words a decision is made by examining the labels on the k-nearest neighbors in the training set $\mathcal{T}$ and taking the majority vote [Duda et al., 2000]. Sometimes it is possible to have a tie, in such situations it is usually resolved by considering the distance of X with the class means which are in tie.

2.3 Fuzzy k-Nearest Neighbor Classifier

Let X be the test pattern and let $\mathcal{T}^{(X)} = \{X_1, X_2, ...X_k\}$ be the set of k vectors from the reference/training set $\mathcal{T}$ which are closest to X according to some distance metric. Let $\sigma(X_i) \in [0,1]^M, \forall j = 1...k$ be the soft labels/fuzzy memberships of the k nearest neighbors of$X$ in $\mathcal{T}$.

To find the fuzzy memberships of X to various classes is given by the following formula [Kuncheva,2000]

$$\mu_i(X) = \frac{\sum_{j=1}^{k} \sigma_i(X_j)(d_j)^{-2/(m-1)}}{\sum_{j=1}^{k} d_j^{-2/(m-1)}} \quad \forall i = 1, ..., M \tag{2.3}$$

where d_j is the distance between X and X_j and m is the fuzzification parameter. Equation (2.3) is infact coupled with an initial fuzzification of the training set $\mathcal{T}$ as follows:

$$\sigma_i(X) = \begin{cases} 0.51 + 0.49(\frac{k_i}{k}) & \text{if } i \text{ is the true/hard class label of } X \\ 0.49(\frac{k_i}{k}) & \text{otherwise} \end{cases} \tag{2.4}$$

Note that for a fixed $\mathcal{T}$ and k initial fuzzification need to be done only once. Now the defuzzification process is to classify X as belonging to class i where

$$\mu_i(X) > \mu_j(X) \ \forall\ i \neq j. \tag{2.5}$$

2.4 Multi-layer Perceptron(MLP)

Multi-layer perceptrons are feed forward networks with one or more hidden layers of nodes between the input and output layers of nodes. The expressive power of the MLPs stem form the non linearties used with in the nodes.

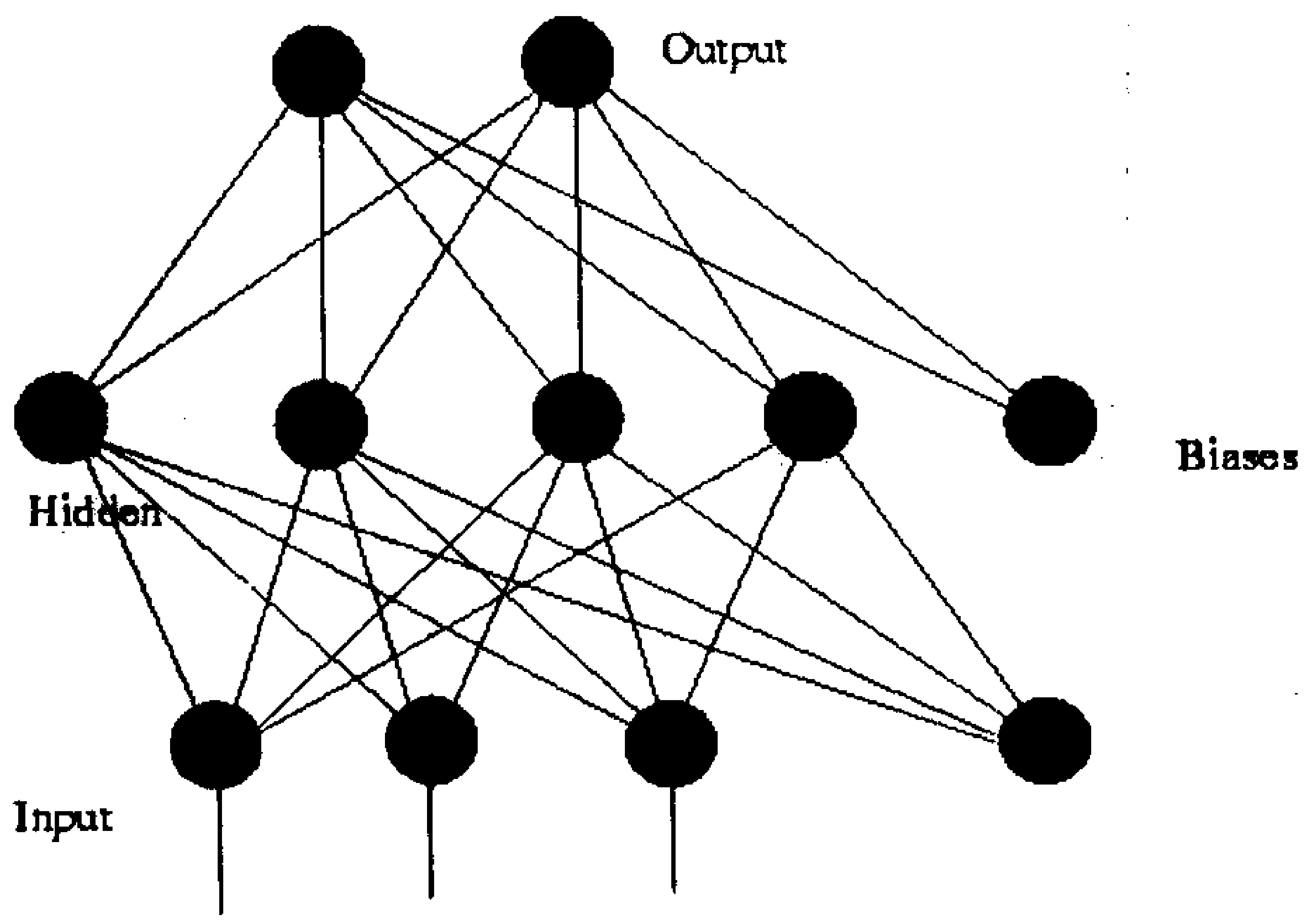

Figure 2.1: The schematic diagram of Multi-layer Perceptron

Figure 2.1 shows a typical multi-layer perceptron neural network structure. As observed in the figure it consists of the following layers:

Input Layer: A layer of neurons that receives information from external sources, and passes this information to the network for processing. These may be either sensory inputs or signals from other systems outside the one being modeled.

Hidden Layer: A layer of neurons that receives information from the input layer and processes them in a hidden way. It has no direct connections to the outside world (inputs or outputs). All connections from the hidden layer are to other layers within the system.

Output Layer: A layer of neurons that receives processed information and sends output signals out of the system.

Bias: Acts on a neuron like an offset. The function of the bias is to provide a threshold for the activation of neurons. The bias input is connected to each of the hidden and output neurons in a network.

The most popular learning algorithm used to train MLP is the backpropagation algorithm, refer [Haykin, 1999].

2.4.1 MLP for pattern classification

In theory, for an M class classification problem in which the union of the M distinct classes forms the entire input space, we need a total of M outputs to represent all possilbe classification decisions, as depicted in Figure 2.2. In this figure the vector X_j denotes one of the training patterns and the vector O_j denotes the output of the network in response to X_j.

Figure 2.2: The block diagram of a Multi-layer Perceptron

Let $F(.)$, a vector-valued function, be the mapping learned by the network trained on $\mathcal{T}$. Clearly any reasonable output decision rule should be based on the knowledge of the vector valued function:

$$F : R^n \ni X \rightarrow O \in R^M \tag{2.6}$$

Suppose now the network is trained with binary target valued vector D_j for each $X_j \in \mathcal{T}$, written as

$$D_{jk} = \begin{cases} 1 & \text{if } X_j \text{ belongs to class } i \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

we can classify the random vector X_j as belonging to class i if

$$O_{ji} > O_{jk} \; \forall \; i \neq k. \tag{2.8}$$

2.5 Classifier Combination

The ultimate goal of designing pattern recognition systems is to achieve the best possible classification performance for the task at hand. This objective traditionally led to the development of different classification schemes for any pattern recognition problem to be solved. The results of an experimental assessment of the different designs would then be the basis for choosing one of the classifiers as a final solution to the problem. It had been observed in such design studies, that although one of the designs would yield the best performance, the sets of patterns misclassified by the different classifiers would not necessarily overlap [Josef Kittler, 1998]. These observations motivated in combining classifiers. The idea is not to rely on a single decision making scheme. Instead, all the designs, or their subset, are used for decision making by combining their individual opinions to derive a consensus decision.

The two main reasons for combining classifiers are efficiency and accuracy [Josef Kittler, 1998]. To increase efficiency one can adopt multistage combination rules whereby objects are classified by a simple classifier using a small set of cheap features in combination of reject option. For the more difficult objects more complex procedures, possibly based on different features, are used.

An important observation in classifier combination is that it is particularly useful if they are of different nature. And an interesting issue in the research concerning classifier combination is the way they are combined. If only labels are available a majority voting is used. If continuous outputs like *a posteriori* probabilities are supplied, an average or some other linear combinations have been suggested.

From the point of view of their analysis, there are basically two classifier combination schemes. In the first scheme, all the classifiers use the same representation of the input pattern. A typical example of this category is a set of k-NN classifiers, each using the same measurement vector, but different classifier parameters (number of nearest neighbors k, or distance metrics used for determining the nearest neighbors). Another example is a set of designs based on a neural network classifier of fixed architecture but having distinct sets of weights which have been obtained by means of different training strategies. In this case, each classifier, for a given input pattern, can be considered to produce an estimate of the, *a posteriori* class probability. In the second scheme, each classifier uses its own representation of the input pattern. In other words, the measurements extracted from the pattern are unique to each classifier. An important application of combining classifiers in this scheme is the possibility to integrate physically different types of measurements/features. In this case, it is no longer possible to consider the computed *a posteriori* probabilities to be estimates of the same functional value, as the classification systems operate in different measurement spaces. In such cases the commonly used classifier combination schemes are product rule, sum rule, min rule, max rule, median rule, and majority voting [Josef Kittler, 1998].

In this dissertation we consider the first scheme in which all the classifers use the same representation of the input pattern and the majority voting strategy is used for combining their decisions.

Chapter 3

Pattern Synthesis

It seems surprising to fix the goal as synthesising new patterns from the existing ones. Nevertheless there are methods to generate new patterns from the old ones, assuming that they all come from the same class. In section 1, we introduce four such verisimilar algorithms proposed by Yoshihiko Hamamoto, Shunji Uchimura, Shingo Tomita in their paper "A Bootstrap Technique for Nearest Neighbour classifier Design," [Yoshihiko Hamamoto et al., 1997]. In section 2, we propose a new algorithm for pattern synthesis considering the whole of training set $\mathcal{T}$, instead of just patterns from one class in isolation.

3.1 Bootstrap techniques for Pattern Synthesis

Let $\mathcal{Z}^i = \{Z_1, Z_2, ...Z_N\}$ be the samples in the original training set which belong to the class i

The following four techniques [Yoshihiko Hamamoto et al., 1997] generates a bootstrap set $Z^i_B = \{Z^b_1, Z^b_2, ...Z^b_N\}$ of size N and are assumed to belong to the class i.

3.1.1 Bootstrapping I

1. Select a sample Z_{k_0} randomly from $\mathcal{Z}^i$
2. Find r nearest neighbors $Z_{k_1}, Z_{k_2}, ..Z_{k_r}$ using Euclidean distance
3. At i^{th} iteration compute the bootstrap sample

$$Z^b_i = \sum_{j=0}^{r} w_j Z_{k_j} \tag{3.1}$$

where w_j is a weight, given by

$$w_j = \frac{\delta_j}{\sum_{c=0}^{r} \delta_c}, 0 \leq j \leq r \tag{3.2}$$

where δ_j is chosen from a uniform distribution on [0,1]

4. Repeat the above steps 1,2 and 3, N times.

The above technique is different from Efron's bootstrapping [B. Efron, 1979](which does random resampling with replacement from $\mathcal{Z}^j$ to create Z_B^i), but in the above algorithm new patterns are being generated by locally combining the original training samples.

3.1.2 Bootstrapping II

1. For each sample Z_{k_0} in $\mathcal{Z}^i$
2. Find r nearest neighbors $Z_{k_1}, Z_{k_2}, ..Z_{k_r}$ using Euclidean distance
3. At i^{th} iteration compute the bootstrap sample

$$Z_i^b = \sum_{j=0}^{r} w_j Z_{k_j} \tag{3.3}$$

where w_j is a weight, given by

$$w_j = \frac{\delta_j}{\sum_{c=0}^{r} \delta_c}, 0 \leq j \leq r \tag{3.4}$$

where δ_j is chosen from a uniform distribution on [0,1]

4. Repeat the above steps 1,2 and 3, N times.

Note that in Bootstrapping II, the samples are chosen such that no sample is selected more than once, where as in Bootstrapping I it is not so.

3.1.3 Bootstrapping III

1. Select a sample Z_{k_0} randomly from $\mathcal{Z}^i$
2. Find r nearest neighbors $Z_{k_1}, Z_{k_2}, ..Z_{k_r}$ using Euclidean distance
3. At i^{th} iteration compute the bootstrap sample

$$Z_i^b = \frac{1}{r+1} \sum_{j=0}^{r} Z_{k_j} \tag{3.5}$$

4. Repeat the above steps 1,2 and 3, N times.

In this method, the bootstrap sample is given as a local sample mean.

3.1.4 Bootstrapping IV

1. For each sample Z_{k_0} in $\mathcal{Z}^i$
2. Find r nearest neighbors $Z_{k_1}, Z_{k_2}, ..Z_{k_r}$ using Euclidean distance
3. At i^{th} iteration compute the bootstrap sample

$$Z_i^b = \frac{1}{r+1} \sum_{j=0}^{r} Z_{k_j} \tag{3.6}$$

4. Repeat the above steps 1,2 and 3, N times.

In this method no sample is selected more than once, and new sample is given as a local sample mean.

Though the authors conclude from their experimental results that there is no clear preference between four Bootstrap techniques [Yoshihiko Hamamoto et. al., 1997], in this work we consider Bootstrap I technique for pattern synthesis, because it maintains the underlying distribution and aribitrarily large number of points can be generated using this technique.

3.2 A DBSCAN based technique for Pattern Synthesis

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a density based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise.

Similar idea has been used in data condensation algorithms. Here we use this idea for pattern synthesis.

Let $\mathcal{T} = \{(Z_i, Y_i | i = 1, 2 \ldots \ldots N\}$ be the training set and M be the total number of classes. The following technique generates a set $\mathcal{T}' = \{(Z_i^d, Y_i^d) | i = 1, 2 \ldots \ldots P\}$ of new samples.

1. Fix P, the total number of points to be generated.
2. Fix $r > 0$, k an integer > 0, and $conf \in [0, 1]$
   (r for radius, $conf$ for confidence)
3. Set $\mathcal{T}' = \phi$
4. For each pattern Z_p in $\mathcal{T}$
   (a) Let Z_p belong to class i
   (b) Find the class frequencies, i.e. total number of patterns from each class say, $k_1, k_2..k_M$ occuring/present in the disc $\mathcal{V}(Z_p, r)$ w.r.t $\mathcal{T}$ (Note that $\mathcal{V}(Z_p, r)$ represents disc with centre Z_p and radius r)
   (c) Define $purity(\mathcal{V}(Z_p, r)) = \frac{k_i}{\sum_{j=0}^{M} k_j}$
5. Consider only the discs with $k_i \geq k$ *and* $purity(\mathcal{V}(Z_p, r)) \geq conf$ as *qualified*
6. Partition the *qualified* discs based on its centre's label
7. In each partition of discs, based on the purity of discs, fix a probability distribution proportional to the purity
8. For each class $c \in \{1, 2 \ldots, M\}$, generate required proportion of points as originally present in training set $\mathcal{T}$ as follows,
   (a) choose a disc randomly from c^{th} partition based on the purity distribution of the *qualified* discs
   (i.e more points are generated from a disc with more purity)
   (b) generate a random point Z_i^d in the chosen disc
   (c) label this point as $Y_i^d = c$
   (d) add (Z_i^d, Y_i^d) to the set $\mathcal{T}'$

Remarks :

1. The above algorithm generates new samples which are *nonlinear* in terms of the existing samples. This is not the case with the Bootstrap Pattern Synthesis methods, which infact is restricted to generating points that are *linear* combinations of existing samples.

2. The above method, also uses the total information in the training set $\mathcal{T}$, and thereby avoids generating large proportion of points in the regions where the classes have high overlap depending on the *conf* parameter.

3. r, k and *conf* have to be chosen carefully such that new patterns are generated from all the classes.

Chapter 4

Active Learning

The goal of active learning is to design and analyze learning algorithms that can effectively choose/query the samples for which they ask the teacher for a label. The incentive for using active learning is mainly to expedite the learning process and reduce the labeling efforts required by the teacher [Baram et al., 2004].

Although this work focuses on pool-based active learning, a number of interesting and relevant ideas appear within other active-learning frameworks. In addition to the pool-based active-learning setting introduced in chapter 1, there are two other settings:

1. Stream-based active learning (see, for example, Freund et al., 1997): the learner is provided with a stream of unlabeled points. On each trial, a new unlabeled point is drawn from the stream and introduced to the learner who must decide whether or not to request its label. Note that the stream-based model can be viewed as an online version of the pool-based model.

2. Active learning with membership queries (Angluin, 1988), also called selective sampling: On each trial the learner "constructs a point" in input space and requests its label. This model can be viewed as a pool-based game where the pool consists of all possible points in the domain.

Within the stream-based setting, [H. S. Seung et al., 1992] present the Query by Committee (QBC) algorithm. This algorithm is based on a seminal theoretical result stating that by halving the version space after each query, the generalization error decreases exponentially (relative to random active learning). Their implementation is as follows:

There exists a source of random inputs, and the committee screens the patterns until one is found which provokes disagreement (as they solve a two class problem, disagreement in this context means that one half of the classifiers assign the pattern to class 1 and the other half assigns class 2). As new patterns are being added at each step an even number of classifiers are induced and the process of screening for new patterns continues. Few main practical difficulties here are, firstly we require a randomized training algorithm in order to design classifiers of different nature. Secondly as number of queries increases, the time required to find a pattern that provokes disagreement diverges.

To overcome the first difficulty, instead of using a committee we use a classifier combination. For tackling the second one we use a pool of unlabeled samples from which we can select a query pattern. Though at this moment we tacitly assumed that a pool is available, let us see in the next chapter how this assumption is relaxed.

Chapter 5

Active Learning With Classifier Combination

In this chapter lets see in detail how active learning has been explored using the classifier combination. After this we present the experimental results, which are analysed in a subsequent section.

Consider that the mixture of classifiers, with possibly multiple instances of the classifiers discussed in chapter 2, ofcourse with different parameters, because there would be not much gain having umpteen similar machines.

Let $C_1, C_2, \ldots, C_q$ be the q classifiers that have been trained on a multi-class classification problem with M classes and a training set $\mathcal{T}$. For an unlabeled pattern X let the labels given by the above classifiers be $L_1, L_2, \ldots, L_q$.

Define Ambiguity/Disagreement(H) among the machines due to X as follows

$$f_i = \textit{total number of labels } (L_r, 1 \leq r \leq q) \textit{that denote class } i \quad (5.1)$$

$$p_i = \frac{f_i}{\sum_{j=1}^{M} f_j} \quad (5.2)$$

$$H = \sum_{i=1}^{M} p_i \log \frac{1}{p_i}. \quad (5.3)$$

Having, at our disposal an unlabeled pool of patterns $\mathcal{U}$ the strategy to choose the most informative pattern is to find the above defined Disagreement among the classifiers for each pattern in $\mathcal{U}$ and choose the pattern which has maximum Disagreement value. Ask the teacher for the correct label of that particular pattern and add this to $\mathcal{T}$.

And the process of training the classifiers on $\mathcal{T}$, finding the most informative pattern from $\mathcal{U}$, disclosing its label, adding this to $\mathcal{T}$ is an iterative process.

So far it is assumed that a pool of unlabeled data is available, but this is actually created using the Pattern Synthesis techniques discussed in chapter 3. Infact for experimental evaluation presented in the next section Bootstrap I and DBSCAN based techniques for pattern synthesis have been used in conjunction with Active Learning Using Classifier Combination.

5.1 Experimental Evaluation

The datasets that has been considered for experimentation are "The Indian Telugu Vowel Data" [S. K. Pal, D. Dutta Majumder, 1977] and "The Diabetes data" from the UCI machine learning repository.

For both these datasets the following three methods have been compared

1. In the first method only passive learning is used, with given percentage of training data to train the classifier combination and tested against the remaining data.
2. In the second method, with the given percentage of training data, passive learning is done, next Bootstrap I technique is applied on this data to generate new patterns(actually the given data is tripled) and active learning is started by collecting 2 or 3 samples in each iteration and adding them to training data, this is repeated until 20% more data(w.r.t initial training data size) is collected.
3. In the third method, with the given percentage of training data, passive learning is done on this data, next DBSCAN based pattern synthesis technique is applied on this data to generate new patterns(here also the given data is tripled) and active learning is started by collecting 2 or 3 samples in each iteration and adding them to training data, this is repeated until 20% more data(w.r.t initial training data size) is collected.

Though theoretically it is best to acquire one sample at each iteration, here at each iteration 2 or 3 samples have been chosen based on the amount of initial training data. Also in each case 25 simulations have been performed and the minimum,maximum and average misclassification percentages have been reported for each data set.

5.1.1 The Indian Telugu Vowel Data

In this data there are in total 871 instances, and 6 overlapping vowel classes and 3 input features(formant frequencies).

This data set is evaluated by taking 8%, 10%, 15%, 20% and 30% of data randomly from 871 instances and the minimum,maximum and average misclassification percentage has been reported in the Table 5.1.

Table 5.1: percentage of misclassification compared for 3 methods on vowel dataset

Percentage of	passive			bootstrap active			dbscan active		
Training Data	min	max	avg	min	max	avg	min	max	avg
8	22.14	35.95	27.10	19.78	30.60	25.34	21.52	28.23	24.35
10	20.36	31.81	24.60	19.47	27.35	22.85	17.68	27.86	22.41
15	18.30	27.32	22.96	18.71	24.90	20.86	16.55	25.17	20.62
20	18.88	23.61	21.26	16.88	22.75	19.67	17.31	22.75	19.83
25	16.79	22.90	19.84	16.49	23.96	20.21	16.03	21.98	19.34

5.1.2 The Diabetes Data

In this data there are in total 768 instances, and 2 classes with 8 input features.

Here also evaluations are done by taking 5%, 8%, 10%, 15%, 20%, 25% and 30% of data randomly from 768 instances with 500 from class 1 and 268 from class 2. The minimum,maximum and average misclassification percentage has been reported in the Table 5.2.

Table 5.2: percentage of misclassification compared for 3 methods on diabetes data

Percentage of	passive			bootstrap active			dbscan active		
Training Data	min	max	avg	min	max	avg	min	max	avg
5	27.67	39.45	32.35	27.26	37.53	31.35	27.12	34.38	30.43
8	25.32	36.49	30.67	27.15	36.63	30.40	26.60	34.08	29.96
10	27.17	33.53	29.91	27.16	36.56	29.98	25.72	33.38	29.83
15	25.11	33.54	29.29	25.11	33.23	28.91	26.18	32.47	28.73
20	25.20	34.80	29.25	24.71	34.47	28.70	26.34	32.52	28.55
30	24.35	32.71	28.26	21.56	29.74	27.51	17.83	31.78	27.15

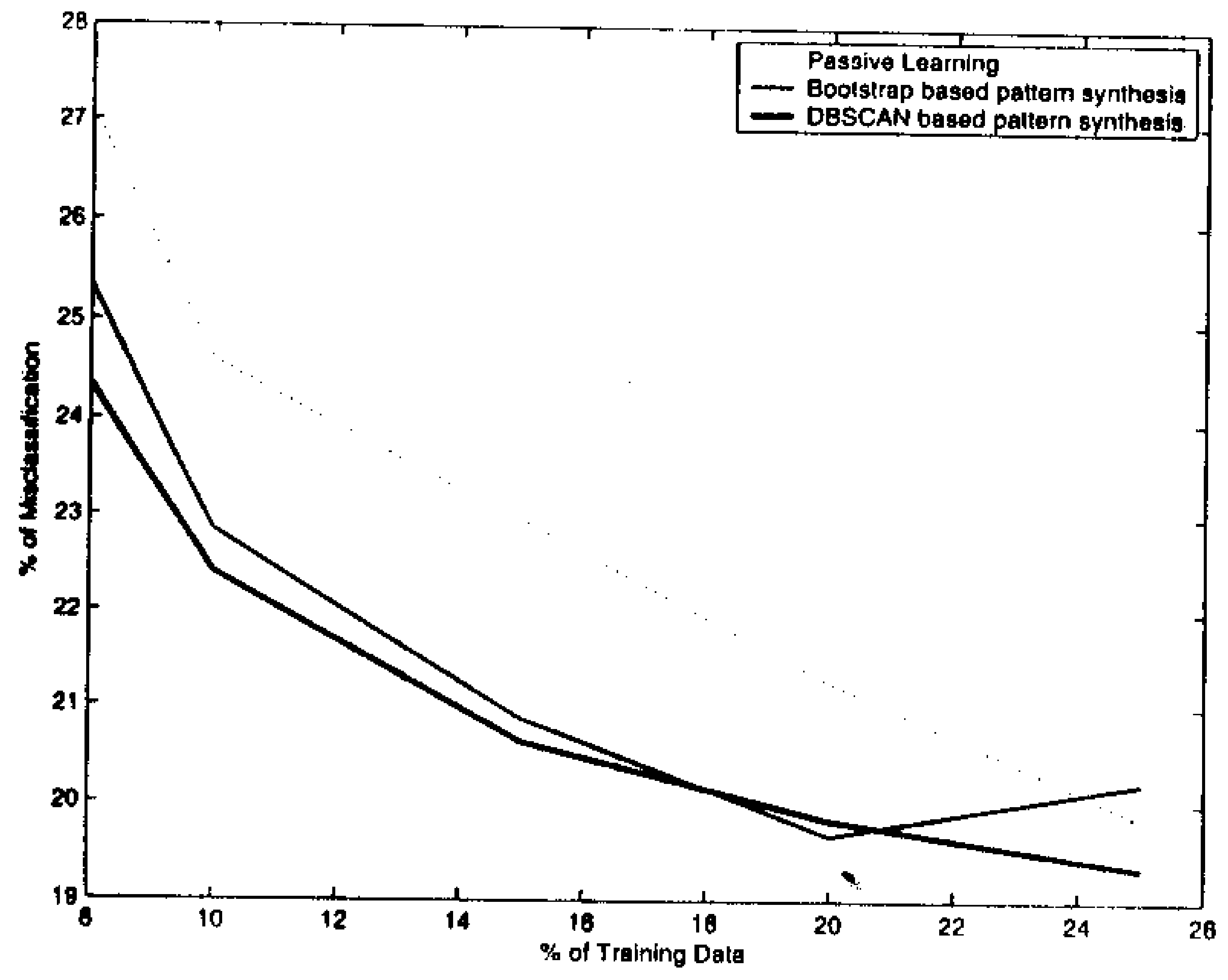

Figure 5.1: percentage training data vs percentage of misclassification for Vowel Dataset

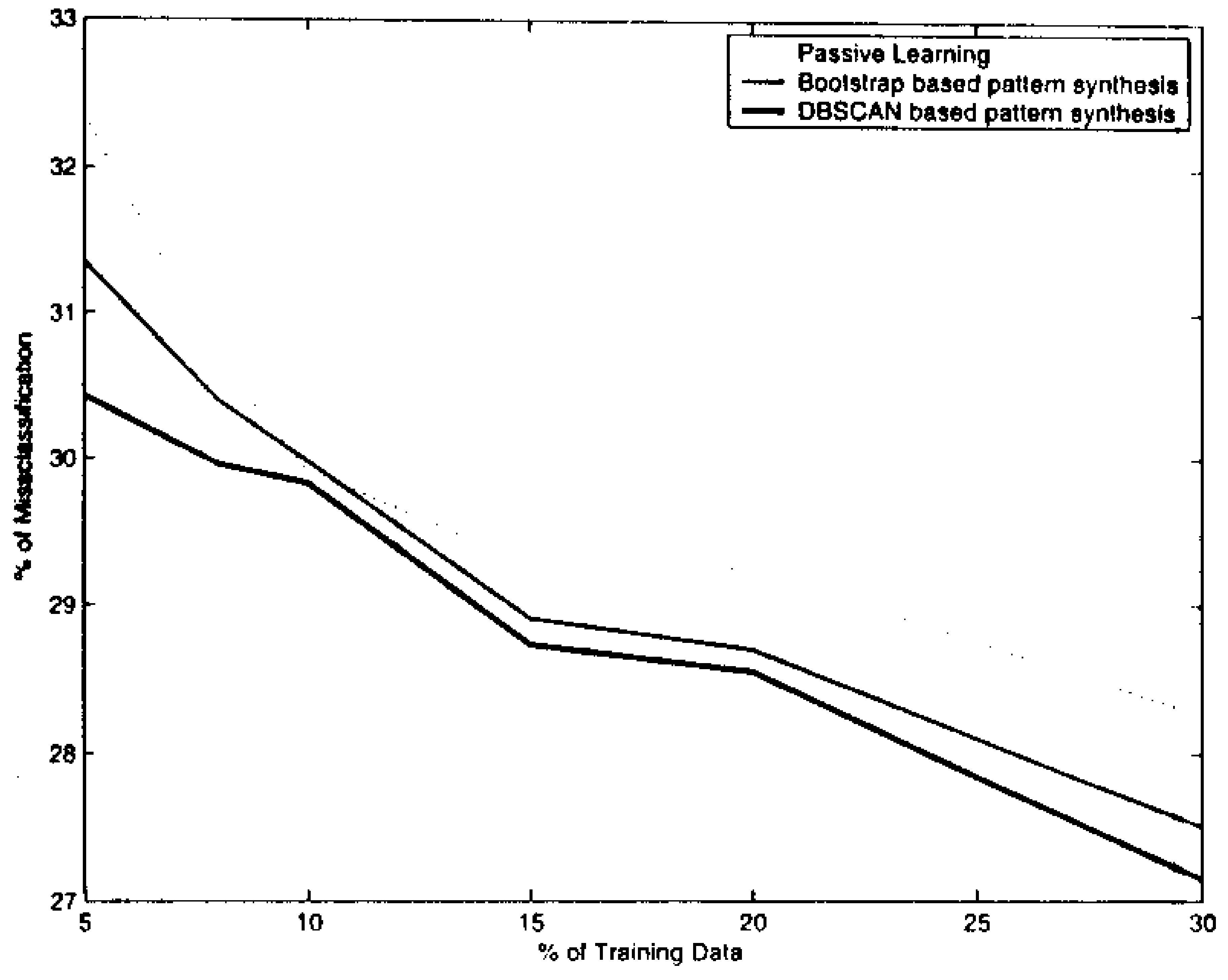

Figure 5.2: percentage training data vs percentage of misclassification for Diabetes Dataset

5.2 Analysis of Results

From the graphs one can see that the classification accuracy is increased in both the active learning cases when compared to that of the passive learning case when a very less percentage of training data is used. Moreover, in the active learning case which uses patterns synthesised by DBSCAN based technique, the classification accuracy is better than the other one which uses patterns synthesised using Bootstrap 1 technique.

Also from the figure 5.1 one can observe that as more percentage of training data is used, the classification accuracy of active learning using patterns synthesised by Bootstrap 1 technique has decreased. This is as expected because the Bootstrap techniques, during pattern synthesis does not consider the information of overlap with other classes. So as the percentage of training data is increased the amount of information present in the training data regarding the overlap between classes also increases, which ofcourse goes unused resulting in generating more patterns with wrong labels. Because of this reason the classification accuracy of an active learner using patterns synthesised by Bootstrap methods decreases. But when the patterns synthesised by DBSCAN based technique is considered this is not the case because we do not create patterns in regions where we have less confidence. This is a good reason why we need to have greater value of *conf* usually around 0.9, see chapter 3 for details.

The next chapter makes conclusions and presents directions for further study.

Chapter 6

Conclusion & Discussion

6.1 Conclusion

In this dissertation, we have proposed a new DBSCAN based pattern synthesis method to be used for active learning and compared it with an existing Bootstrap based pattern synthesis technique. Also both these methods are compared with conventional passive learning algorithm.

From the results obtained, we concluded that this method of pattern synthesis, in conjunction with active learning gives better classification accuracy when compared to the passive learning method. But when there is more data the Bootstrap 1 technique, creates patterns that deteriorate the learning performance. This is not the case with DBSCAN based pattern synthesis method, it performs better than the conventional passive learning method even when the training data set is large and classes are overlapping with each other.

6.2 Future Work

Though we have proposed a pattern synthesis technique, we have not given any algorithm to choose the parameters for obtaining better results using this technique. Currently, it is the responsibility of the user to utilize the problem domain knowledge in choosing the right parameters. This is one aspect where we require further exploration, the algorithm should automatically choose appropriate parameters r and k for a given $conf$ value.

The effect of active learning using classifier combination over large data sets is to be studied in isolation as well as in conjunction with the pattern synthesis techniques.

Bibliography

[V.N. Vapnik 1999] Vladimir N. Vapnik, "An Overview of Statistical Learning Theory," IEEE transactions on neural networks, vol. 10, no. 5, sep 1999.

[Cohn et al., 1996] Davin A Cohn, Zoubin Ghahramani, Micheal I. Jordan, "Active Learning with statistical Methods", Journal of Artificial Intelligence Research 129-145,(4) 1996.

[Baram et al., 2004] Yoram Baram, Ran El-Yaniv,Kobi Luz, "Online choice os Active Learning Algorithms," Journal of Machine Learning 255-291,(5) 2004.

[Duda et al., 2000] Richard O, Duda, Peter E. Hart, David G. Stork, "Pattern Classification", second edition, 2000, John Wiley publishers.

[Kuncheva,2000] Ludmila I. Kuncheva, "Fuzzy Classifier Design," 2000, Physica-Verlag.

[Haykin, 1999] Simon Haykin, "Neural Networks: A Comprehensive Foundation," 1999, Pearson.

[Josef Kittler, 1998] Josef Kittler, "On Combining Classifiers," IEEE transactions on Pattern Analysis and Machine Intelligence, vol 20, no 3, mar 1998

[Yoshihiko Hamamoto et al., 1997] Yoshihiko Hamamoto, Shunji Uchimura, Shingo Tomita, "A Bootstrap Technique for Nearest Neighbour Classifier Design," IEEE transactions on Pattern Analysis and Machine Intelligence, vol 19, no 1, jan 1997

[B. Efron, 1979] B Efron, "Bootstrap Methods: Another Look at the Jackknife," Annual Statistics, vol, 7, 1-26, 1979.

[H. S. Seung et al., 1992] H. S. Seung, M. Opper, H. Sompolinsky,"Query By Committee," ACM, 1992.

[S. K. Pal, D. Dutta Majumder, 1977] S. K. Pal, D. Dutta Majumder, "Fuzzy sets and decision making approaches in vowel and speaker recognition," IEEE transactions on Systems, Man and Cybernetics, vol. 7, pp. 625-629, 1977.