

ADAPTIVE BAYESIAN APPROACH FOR CLASSIFICATION OF DATA STREAMS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

INTEGRATED DUAL DEGREE

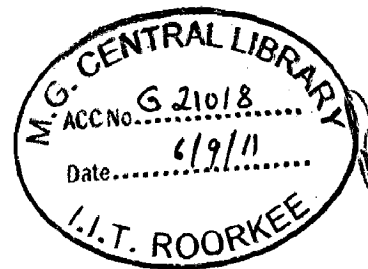
in

COMPUTER SCIENCE AND ENGINEERING

(With Specialization in Information Technology)

By

PARTHSARTHI MISHRA



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2011**


CANDIDATE'S DECLARATION

I hereby declare that the work is being presented in the dissertation work entitled "**Adaptive Bayesian Approach for Classification of Data Streams**" towards the partial fulfilment of the requirement for the award of the degree of **Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology)** submitted to the **Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from May, 2010 to May, 2011 under the guidance and provision of **Dr. Durga Toshniwal, Assistant Professor, Department of Electronics and Computer Engineering, IIT Roorkee.**

I have not submitted the matter embodied in this dissertation work for the award of any other degree and diploma.

Date: 27th June, 2011

Place: Roorkee



(PARTHSARTHI MISHRA)

CERTIFICATE

This to certify that the work contained in the dissertation entitled "**Adaptive Bayesian Approach for Classification of Data Streams**" by Parthsarthi Mishra of Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology), has not been submitted elsewhere for a degree or diploma to the best of my knowledge.

Date: 27th June, 2011

Place: Roorkee


27/6/11.
Dr. Durga Toshniwal
Assistant Professor,
E&CE Department
IIT Roorkee, India

ACKNOWLEDGEMENTS

I would like to take this opportunity to extend my heartfelt gratitude to my guide and mentor **Dr. Durga Toshniwal**, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for her trust in my work, her able guidance, regular source of encouragement and assistance throughout this dissertation work. I would state that the dissertation work would not have been in the present shape without her inspirational support and I consider myself fortunate to have done my dissertation under her.

I also extend my sincere thanks to **Dr. S.N. Sinha**, Professor and Head of the Department of Electronics and Computer Engineering for providing facilities for the work.

I would like to thank all my friends who supported and encouraged me to finish this work.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing constant support, encouragement, and love.

PARTHSARTHI MISHRA

LIST OF FIGURES

Figure 2.1	A typical classification task	6
Figure 3.1	Framework for the proposed ANB scheme.	16
Figure 3.2	Pseudocode for merging a data point in class	18
Figure 3.3	Training process for Adaptive Bayesian classification scheme	19
Figure 3.4	Pseudocode for algorithm that computes the data given class probability, $P(X C)$	21
Figure 3.5	Pseudocode for algorithm computes the most probable class C for the given data point X	21
Figure 3.6	Class Prediction and Model Updating during the testing phase of Adaptive Naïve Bayes Classifier	23
Figure 3.7	Inability of Bayesian methods to identify new class data points	24
Figure 3.8	Occurrence of a cohesive set of outliers in the chunk with identification of new class	26
Figure 3.9	Class Prediction and Model Updating including New Class Detection in Adaptive Naïve Bayes Classifier	27
Figure 4.1	The WEKA explorer after opening the iris dataset.	32
Figure 5.1	Graph showing the comparison of accuracy obtained between proposed scheme, simple Naïve Bayes and CART	35
Figure 5.2	Graph showing the accuracy obtained by Naïve Bayes and proposed scheme on Spambase using 5% training set	36

LIST OF TABLES

Table 2.1	Example of a data stream describing electric power metering	5
Table 2.2	Steps for merging a data point into a supervised microcluster	13
Table 4.1	Data members used to implement the supervised microclusters.	29
Table 4.2	Some important member functions used in the implementation of the supervised microclusters.	30
Table 5.1	Summary of the datasets used in our results.	34
Table 5.2	Comparison of accuracy obtained between proposed scheme and simple Naïve Bayes.	35
Table 5.3	Results obtained for each data chunk while testing new class detection scheme.	37

ABSTRACT

Data Streams are temporally ordered, fast moving, massive, and potentially infinite in nature. They may be generated at high rates as a result of measurements generated continuously by sensor networks, web logs, computer network traffic etc. The storage, querying and mining of data streams are highly computationally challenging tasks. Classification is a problem of supervised grouping of data in order to extract meaningful patterns. Data streams are too large to fit in main memory and are typically stored in secondary storage devices. Besides the considerations of running time and memory usage another important issue that is important in dealing with data streams is that of *Concept Drift* i.e. change in the underlying data distributions.

Linear scans are the only cost-effective access method for data streams as random access is prohibitively expensive. Also, there is a need for an efficient summarization technique to maintain the past data leaving enough memory for processing of future data. The classification algorithm needs to be incremental in nature in order to account for the underlying changes in the data distributions (concept drift). Thus there should be a mechanism to update the summary in order to keep the classifier sensitive to such changes. These issues make classification of data stream a very challenging task.

In our work, an adaptive classification model has been proposed that dynamically evolves with the data stream thus providing improved results. Our method is based on the Naïve Bayesian classifier. Naive Bayesian classification is a probabilistic technique used to classify data based on Bayes' theorem. Supervised microclusters provide an efficient approach to store the summary of past data. This summary is then used to determine probabilities for Naïve Bayesian classifier. A novel class detection approach has also been proposed that determines new class points by delayed classification of data points in the chunk. This accounts for the *concept drift* in the data stream. The empirical results show that higher classification accuracy is achieved as compared to the static methods. In the present form, the proposed work is applicable to numerical data streams only.

Table of Contents

CANDIDATE’S DECLARATION	i
ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	iii
LIST OF TABLES	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
CHAPTER 1: INTRODUCTION	1
1.1 Introduction	1
1.2 Motivation	1
1.3 Problem Statement	2
1.2 Organization of Thesis	3
CHAPTER 2: BACKGROUND AND LITERATURE REVIEW	4
2.1 Data Streams	4
2.2 Classification	5
2.2.1 Classification by Decision Tree Induction	7
2.2.2 Bayesian Classification	8
2.2.3 Rule Based Classification	9
2.2.4 k-Nearest Neighbour Classifiers	10
2.2.5 Support Vector Machines (SVM)	11
2.3 Classification of Data Streams	11
2.3.1 Supervised Microclusters	12
2.3.2 Novel Class Detection in Data Streams	14
2.4 Research Gaps	14
CHAPTER 3: PROPOSED ADAPTIVE BAYESIAN APPROACH FOR CLASSIFICATION OF DATA STREAMS	16
3.1 Block 1 – Preprocessing	17

3.2 Block 2 - Training the Classifier	17
3.2.1 Maintenance of Class Data Using Supervised Microclusters	17
3.2.2 Training the Adaptive Naïve Bayes Classifier.....	18
3.3 Block 3 - Class Prediction Using the Adaptive Naïve Bayes Classifier	20
3.3.1 Calculation of Class Probabilities.....	20
3.3.2 Prediction of Class Label for a given Data Point	21
3.3.3 Class Prediction without Novel Class Detection	22
3.4 Block 4-Class Prediction with Novel Class Detection	24
3.4.1 Identification of Outliers.....	24
3.4.2 Detecting a New Class	25
CHAPTER 4: IMPLEMENTATION DETAILS	29
4.1 Implementation	29
4.2 Design and Implementation of Classes Based on Supervised Microclusters	29
4.3 Implementation of Training and Testing phases of Classifier	30
4.3.1 Using WEKA for Comparative Analysis	31
CHAPTER 5: RESULTS AND PERFORMANCE ANALYSIS	33
5.1 Datasets	33
5.1.1 Spambase Dataset	33
5.1.2 Iris Dataset	33
5.1.3 KDD Cup'99 IDS Dataset	33
5.2 Performance of Adaptive Bayesian Classification Scheme on Static Datasets	34
5.3 Performance of Adaptive Bayesian Classification Scheme on Streaming Data.....	36
5.4 Performance of Novel Class Detection Scheme in Adaptive Naïve Bayesian Classification.....	37
CHAPTER 6: CONCLUSION AND FUTURE WORK	38
6.1 Conclusion	38
6.2 Suggestions for Future Work	39
REFERENCES	40
LIST OF PUBLICATIONS	45

CHAPTER 1

INTRODUCTION

1.1 Introduction

Data mining refers to extracting or “mining” knowledge from large amounts of data. It is sometimes also referred as “*knowledge discovery from databases (KDD)*”. Data mining can be defined as “*the nontrivial extraction of implicit, previously unknown and potentially useful information from data*” [1]. Data mining involves an integration of techniques from multiple disciplines such as database and data warehouse technology, statistics, machine learning, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial or temporal data analysis. Typical data used for applying mining techniques include scientific data, financial data, marketing data, medical data, web clickstreams, network intrusion data, demographical data etc. Some important data mining techniques are- classification, clustering, and association rule mining. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions.

Large amounts of data are being generated nowadays by sensor networks, share markets, web clickstreams, Business information systems, telecommunication providers etc. These continuously growing large datasets are also known as data streams [2]. Data streams find large number of relevant applications [3], [4]. Mining of such massive volumes of data is often difficult on account of large space and processing requirements. The streams considered here are streams of items (or elements) in the form of structured data and should not be confused with streams of audio and video data.

1.2 Motivation

Data streams are temporally ordered, fast changing, massive, and potentially infinite. The problems generally face while mining data streams are as follows:

1. It is not feasible to store an entire data stream or to scan through it multiple times due to its tremendous volume. Therefore an efficient storage method needs to be developed that

stores the summary of past data in such a way so as to minimize the loss of information and at the same time use minimum possible memory space. This allows us to use the remaining memory space for processing the incoming data.

2. Moreover, stream data tend to be of rather low level of abstraction, whereas most analysts are interested in relatively high-level dynamic changes, such as trends and deviations [1].
3. Also, the streams may evolve considerably over time [5]. Thus the patterns discovered by the algorithm in the data from the past, may not be valid for the new data due to some unexpected changes in the generating process (e.g. a political event which affects the stock prices) [6]. This phenomenon is known as *concept drift* i.e. a concept that changes over time. The rate of drift is defined as the probability that a target function disagrees over two successive examples [7]. Thus we require a method that adapts with time to account for the *concept drift* and improve the efficiency of our algorithm.

So to discover knowledge or patterns from data streams, it is necessary to develop single-scan, on-line, evolving stream processing and analysis methods. There is thus a great scope to evolve the data mining techniques that address the issues associated with data streams.

1.3 Problem Statement

The problem statement for the proposed research work is as follows:

“To develop an adaptive method for improved classification of evolving data streams using Naïve Bayesian approach.”

The above problem statement can be divided into the following sub problems:

- *To develop a supervised microcluster model for training the classifier as well as efficiently maintaining the summary of past data.*
- *To design algorithms for computing probabilities to determine the class label for a data instance.*
- *To develop algorithms for updating the model so that it may be sensitive to underlying changes.*
- *To design technique for identification of a new class in the data stream.*

- *The proposed method is applicable to numeric data only.*

1.4 Organization of Thesis

This report comprises of six chapters including this chapter that introduces the topic and states the problem. The rest of the dissertation report is organized as follows.

A brief description of literature review on data streams and classification is provided in Chapter 2. The topics include major classification techniques including a background theory on Naïve Bayes classifier. Some existing data stream classification techniques have also been mentioned.

Chapter 3 provides a detailed description of supervised microclustering technique, proposed algorithms for probability calculations and new class determination approach. Flow diagrams for explanation of entire proposed scheme have also been shown. Algorithms for identification of outliers and condition for new class determination have been discussed.

The implementation details of the proposed model discussed in Chapter 3 alongwith brief descriptions is provided in Chapter 4. The data structure for maintenance of class data of the classifier has also been discussed.

Chapter 5 discusses the results for the adaptive Naïve Bayesian classification scheme including a comparative analysis showing the improvement achieved by the adaptive nature of the classifier. Results of the testing of the new class detection strategy are also shown.

Chapter 6 concludes the work, points out some shortcoming and gives some directions for future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Data Streams

Nowadays many applications are generating data at very high rates in the form of transient data streams. Examples of such applications include financial applications, network monitoring, security, telecommunication data management, web applications, manufacturing, sensor networks, and others. In the data stream model, individual data items may be relational tuples, e.g., network measurements, call records, web page visits, sensor readings, and so on. They are time-varying, rapid and possibly infinite and thus open new fundamental research problems. This rapid generation of continuous streams of information has challenged our storage, computation and communication capabilities in computing systems. The vast amounts of data arriving in high speeds need employment of online, adaptive and fast mining techniques, to perform real-time extraction of hidden knowledge and information.

Typical data mining tasks include clustering, classification and association mining. These techniques help find interesting patterns, regularities, and anomalies in the data. These traditional data mining techniques cannot directly be applied to data streams. This is because most of them require multiple scans of data to extract the information, which is unrealistic for stream data. The amount of previously happened events is usually overwhelming, so they can be either dropped after processing or archived separately in secondary storage. Sometimes we need to maintain an efficient summary of the past data for the processing tasks. Furthermore, we also need to consider the problem of resource allocation in mining data streams. Due to the large volume and the high speed of streaming data, mining algorithms must cope with the effects of system overload. Table 1 shows an example of a data stream representing electric power consumption measured by many communicating meters in households (Refer to [8]).

Some applications require prediction of class labels or new values of data instances in the stream given some knowledge of class membership or data values of previous instances in the stream. We train our models to learn this prediction in an automated fashion by providing labeled examples. A major issue with stream mining is that the underlying patterns may change over

time. This problem is referred to as *concept drift*. Thus the distribution underlying the instances or the rules underlying their labeling may change over time, i.e. the goal of the prediction, the class to be predicted or the target value to be predicted, may change over time.

Table 2.1 Example of a data stream describing electric power metering

Timestamp	Meter	Active P (kW)	Reactive P (kVAR)	U (V)	I (A)
16/12/2006-17:26:12	86456422	5,374	0,498	233,29	23
16/12/2006-17:26:32	64575861	5,388	0,502	233,74	23
16/12/2006-17:26:11	89764531	3,666	0,528	235,68	15,8
16/12/2006-17:29:28	25647898	3,52	0,522	235,02	15

The foundations on which stream data mining solutions rely come from the field of statistics, complexity and computational theory [9]. Data stream processing systems face high memory and computing requirements due to the huge volumes of data they encounter and its potentially high arrival rate. Many summarization techniques have been adopted from the field of statistics in order to deal with these resource constraints. They provide means to examine only a subset of the whole dataset or to transform the data to an approximate smaller size data representation so that known data mining techniques can be used. Summarization techniques refer to the process of transforming data to a suitable form for stream data analysis so as to minimize information loss and achieve good accuracies in mining tasks at the same time. Summarization techniques are often used for producing approximate answers from large databases. They synthesize techniques for data reduction and synopsis construction. This can be done by summarizing the whole dataset or choosing a subset of the incoming stream to be analyzed. When summarizing the whole dataset techniques such as sampling, sketching and load shedding are used. For choosing a subset of the incoming stream synopsis data structures and aggregation techniques are used.

2.2 Classification

Classification [10-13] is an important data mining problem. It is the process of categorizing the data into some known class-labels. Classification can be defined as the task of learning a target function f that maps each attribute set x to one of the predefined class labels. Data classification

is a two-step process. In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or “learning from” a training set made up of database tuples and their associated class labels. A tuple, X , is represented by an n -dimensional attribute vector, $X = (x_1, x_2, \dots, x_n)$, depicting n measurements made on the tuple from n database attributes, respectively, A_1, A_2, \dots, A_n . Each tuple, X , is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. The class label attribute is discrete-valued and unordered. It is categorical in that each value serves as a category or class. The individual tuples making up the training set are referred to as training tuples and are selected from the database under analysis. This step is also known as *supervised learning* as the class label of each training tuple is provided [1]. Once built, the next step is to evaluate the classifier using the test data given. After this step, the classifier can be used for predicting the class label of new and unseen data.

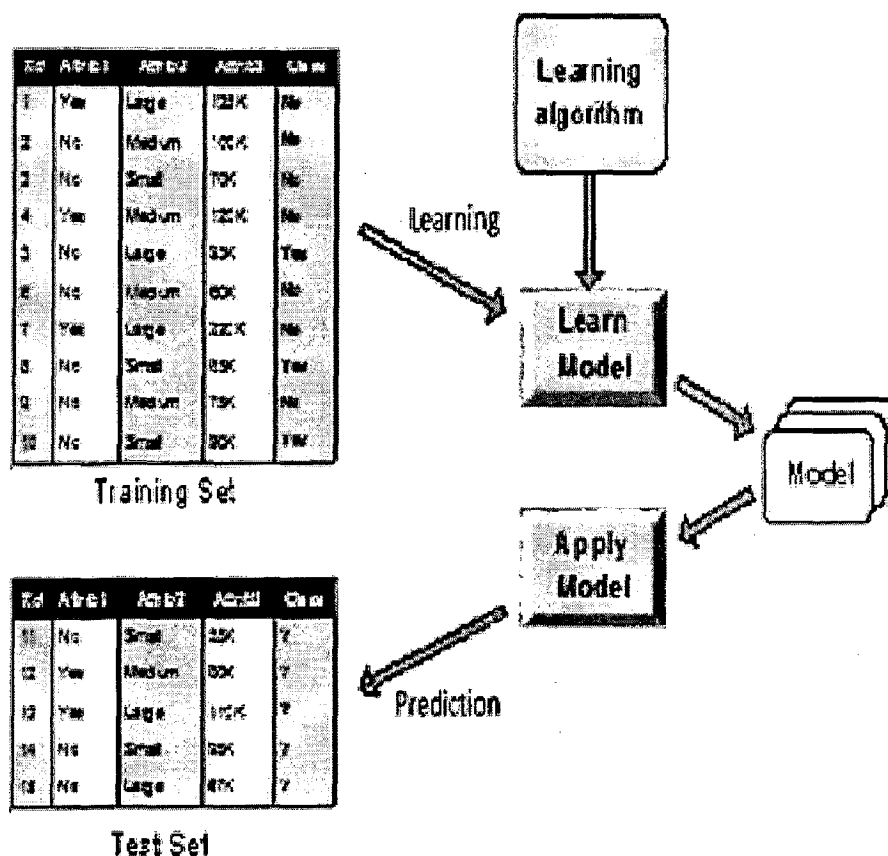


Figure 2.1: A typical classification task

The following measures can be used to assess the performance of the classification model:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total Number of predictions}} \quad (2.1)$$

$$\text{And, Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} \quad (2.2)$$

Following is a list of some major classification approaches which are discussed in brief in the sections that follow.

- Classification by Decision Tree Induction
- Bayesian Classification
- Rule Based Classification
- Nearest Neighbor Classifier
- Support Vector Machines (SVM)

2.2.1 Classification by Decision Tree Induction

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a hierarchical tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. Given a tuple, X , for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple.

The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. In general, decision tree classifiers have good accuracy [1]. However, over-fitting to the training data due to noise or lack of representative samples may lead to poor generalization errors. Decision trees are the basis of several commercial rule induction systems. ID3 [14], C4.5 [15] and CART [16] are some of the commonly used decision tree algorithms.

2.2.2 Bayesian Classification

Bayesian classifiers use Bayes' theorem to predict the class membership probabilities i.e. the probability that a given tuple belongs to a particular class. These classifiers show high accuracy and speed when applied to large databases. Simple Naïve Bayesian classifier is found to be comparable in performance with decision tree and selected neural network classifiers.

Bayes' Theorem

Let X be a data tuple. In Bayesian terms, X is considered "evidence." As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis, such as that the data tuple X belongs to a specified class C . For classification problems, we want to determine $P(H|X)$, the probability that tuple X belongs to class C , given that we know the attribute description of X . The *prior* probabilities $P(H)$ and $P(X)$, and the *posterior* probability $P(X|H)$, may be estimated from the given data. Bayes' theorem provides a way of calculating the posterior probability, $P(H|X)$, from $P(H)$, $P(X|H)$, and $P(X)$. Bayes' theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \quad (2.3)$$

Naïve Bayesian Classifier

Given a d -dimensional data point X and classes $C_1 \dots C_m$ a Naïve Bayes classifier predicts the class of X based on the highest posterior probability conditioned on X . That is the classifier predicts that X belongs to the class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Thus we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called maximum posteriori hypothesis. By Bayes theorem,

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (2.4)$$

As $P(X)$ is constant for all the classes only $P(X|C_i)P(C_i)$ needs to be maximized. If the class prior probabilities are not known, then the classes are assumed equally likely, that is, $P(C_i) =$

$P(C_2) \dots = P(C_m)$. Prior probabilities may also be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D .

To compute $P(X|C_i)$, the classifier makes naïve assumption of *class conditional independence*. This presumes that the values of attributes are conditionally independent of one another, given the class label of the tuple. Thus,

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^n P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i) \end{aligned} \quad (2.5)$$

A continuous valued attribute is assumed to have a Gaussian distribution with a mean μ and standard deviation σ defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.6)$$

so that,
$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \quad (2.7)$$

Where μ_{C_i} and σ_{C_i} are mean and standard deviation, respectively, of the values of attribute A_k for all the data tuples belonging to class C_i [1].

2.2.3 Rule Based Classification

Rules are a good way of representing information or bits of knowledge. A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

IF *condition* THEN *conclusion*.

An example is rule R1,

R1: IF *rank* < 8000 AND *age* < 18 THEN *college* = IIT.

The “IF”-part (or left-hand side) of a rule is known as the *rule antecedent* or *precondition*. The “THEN”-part (or right-hand side) is the *rule consequent*. In the rule antecedent, the condition consists of one or more attribute tests (such as *rank* < 8000, *age* < 18) that are logically ANDed. The rule’s consequent contains a class prediction (in this case, we are predicting the college that

a student will get). If the condition (that is, all of the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule is satisfied and that the rule covers the tuple.

A rule R can be assessed by its coverage and accuracy. Given a tuple, X , from a class labeled data set, D , let n be the number of tuples covered by R ; $n_{correct}$ be the number of tuples correctly classified by R ; and N be the number of tuples in D . We can define the *coverage* and *accuracy* of R as

$$Coverage(R) = \frac{n}{N} \quad (2.8)$$

$$Accuracy(R) = \frac{n_{correct}}{n} \quad (2.9)$$

That is, a rule's coverage is the percentage of tuples that are covered by the rule (i.e., whose attribute values hold true for the rule's antecedent). For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify [1]. Some of the popularly used rule based classifiers are CN2 [17] and RIPPER [18].

2.2.4 k -Nearest Neighbor Classifiers

Nearest-neighbor classifiers are based on learning by analogy, i.e. by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n -dimensional space. In this way, all of the training tuples are stored in an n -dimensional pattern space. When given an unknown tuple, a k -nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. The unknown tuple is assigned the most common class among its k nearest neighbors. Nearest-neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple. Missing value imputation for a particular attribute can also be done in a similar way. "Closeness" is defined in terms of a distance metric, such as Euclidean distance. This choice may be critical for the accuracy of the classifier [1].

The k -NN classifiers offer several advantages. They do not require model building and can produce arbitrary shaped decision boundaries. They have some limitations too. k -NN classifiers

are prone to noise and are very slow when classifying data tuples. Thus they can't be used as such for classification of large datasets. Also, they are not a very good choice for classification of categorical attributes due to the use of distance as a metric.

2.2.5 Support Vector Machines (SVM)

A SVM is based on maximal margin hyper-plane in order to ensure that their worst-case generalization errors are minimized. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (that is, a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. *Support vectors* (training tuples) and *margins* (defined by support vectors) are used to find this hyperplane. In case when data is linearly separable, there are infinite lines (hyperplanes) separating the classes but we want to find the best one (the one that minimizes classification error on unseen data) SVM searches for the hyperplane with the largest margin, i.e., maximum marginal hyperplane (MMH). The training time for SVMs can be quite large but the accuracy is very high owing to their ability to model complex nonlinear decision boundaries (margin maximization). They are much less prone to overfitting than other methods [1].

2.3 Classification of Data Streams

Traditional classification methods like ID3 [14], C4.5 [15], CART [16], IFN [19], face new problems while dealing with data streams. These methods generally store and process the entire training set of examples. As the training set grows in size the memory requirements increase and computation time becomes prohibitively long. Another distinguishing characteristic of data streams is that they are time-varying, as opposed to traditional database systems, where only the current state is stored. This change in the nature of the data takes the form of changes in the target classification model over time and is referred to as concept drift. Concept drift is an important consideration when dealing with stream data. In order to account for the concept drift, the classification methods need to be adaptive and should evolve with time.

Several authors have studied the idea of implementing a decision tree technique for classification of stream data. Domingos et al. [20], [21] have studied the problem of maintaining decision trees over data streams. In [19] they have developed the VFDT system. It is a decision tree learning system based on Hoeffding trees. Ganti et al. [22] have developed analytically two algorithms GEMM and FOCUS for model maintenance and change detection between two data sets in terms of the data mining results they induce. The algorithms have been applied to decision tree models and the frequent itemset model. Techniques such as decision trees are useful for one-pass mining of data streams but these cannot be easily used in the context of an on-demand classifier in an evolving environment.

The problem of concept drift has been addressed by many authors. Wang et al. [23] have proposed a general framework for mining concept drifting data streams. The proposed technique uses weighted classifier ensembles to mine data streams. CVFDT, a later version of VFDT, was developed by Hulten, Spencer, and Domingos [24] to handle concept drift in time-changing data streams [34]. Last [25] has proposed an online classification system which dynamically adjusts the size of the training window and the number of new examples between model re-constructions to the current rate of concept drift. Aggarwal et al. in [26] have presented a different view on the data stream classification problem from the perspective of a dynamic approach, in which simultaneous training and testing streams are used for dynamic classification of datasets. They developed a k-nearest-neighbor-based method for classify evolving data streams.

2.3.1 Supervised Microclusters

The high volume of the data stream makes it essential to store summary statistics efficiently. Supervised microclustering is a way to achieve summarization such that adding and removing a data point is easy. Let us assume that data stream consist of a set of multidimensional records X_1, \dots, X_k . Each X_i is a multidimensional record containing d dimensions which are denoted by $X_i = (x_1, \dots, x_d)$. During initialization, each training data point is associated with a class label C_j .

We will first begin by defining the concept of supervised microclusters. This concept has been adopted for Bayesian classifier from [27]. While the microclustering concept of [27] is useful for unsupervised clustering, we need to make modifications in order to use this approach for the classification process. Aggarwal et al. [28] extended supervised microcluster technique to be

used for classification. The supervised microclusters are created from the training data stream and then are continuously updated during the testing phase. Each such microcluster corresponds to a set of points from the data, all of which belong to the same class.

Definition 2.1 A supervised microcluster for a set of d -dimensional points X_1, \dots, X_n with belonging to the class C is defined as the $(2d+2)$ tuple (SS, LS, n, C) , wherein SS and LS each correspond to a vector of d entries. The definitions of each of these entries are as follows:

- For each dimension, the sums of the squares of the data values are maintained in SS . Thus, SS contains d values. The p^{th} entry of SS is equal to $\sum_{i=1}^n (x_i^p)^2$.
- For each dimension, the sums of the data values are maintained in LS . Thus, LS contains d values. The p^{th} entry of LS is equal to $\sum_{i=1}^n x_i^p$.
- The number of data points is maintained in n .
- C represents the class label of that microcluster.

This model can be incrementally implemented in data stream scenario as merging a new data point into the microcluster is easy. Merging a data point X into a microcluster can be performed steps mentioned in Table 2.1.

Table 2.2 Steps for merging a data point into a supervised microcluster.

1. $\sum_{i=1}^d SS[i] = SS[i] + X_i^2$ (Updating the squared sum)
2. $\sum_{i=1}^d LS[i] = LS[i] + X_i$ (Updating the linear sum)
3. $n = n+1$ (Updating the count)

This summary information is an extension of the cluster feature vector concept discussed in [29]. Since each component in the definition of the microcluster is an additive sum over different data points, this data structure can be updated easily over different data streams as discussed above. The above microclustering model is primarily designed for handling the continuous attributes.

2.3.2 Novel Class Detection in Data Streams

The arrival of a new class in a data stream can be referred to as *concept evolution*. This problem also needs to be addressed in stream classification technique. For example, if we are using our classifier for intrusion detection and a new kind of attack arrives then our classifier should be able to deduce that it is a new class of attack.

Traditional “one class” novelty detection techniques [30-32] could only distinguish between the normal and anomalous data. A single data point at a time was processed and identified as being a “normal” instance or an anomaly/novel class instance. They were unable to distinguish between different types of anomalies. Masud et al. [33] addressed this problem using a classifier ensemble and developed a “multi-class” framework that could distinguish between multiple classes as well as discover the emergence of new class. The idea was to identify a group of outliers that had strong cohesion among themselves such that they might form a new class.

2.4 Research Gaps

- Naïve Bayes’ algorithm is a fast and efficient method for classification of large datasets and the results produced are comparable to selected decision tree and neural network based algorithms. The application of Naïve Bayesian technique in streaming data scenario has not been explored much. Thus there is a lot of scope for their use in streaming data scenario.
- Most of the existing stream mining techniques use a classifier ensemble to handle the concept drift. There is a scope to develop a framework for classifiers, so that they can adapt with change and can easily incorporate new data instances using an effective data summarization technique.
- There is scope to adapt the supervised microcluster approach developed in [28] and to handle concept drift in Bayesian classification scenario where we deal in probabilities and there is no distance metric involved.
- Masud et al. [33] have addressed the problem of new class detection using a classifier ensemble. This technique works in “multi class” scenario as well by checking outliers for strong cohesion among themselves such that a new class can exist. There is a scope to

extend this approach in Naïve Bayesian classification such that distance metric is not involved.

CHAPTER 3

Proposed Adaptive Bayesian Approach for Classification of Data Streams

In this chapter, we discuss the proposed adaptive Naïve Bayesian classification scheme that works on the principle of simultaneous testing and training of the model. To use this scheme we need an effective data summarization technique so that the aggregate data may be used for classification effectively. Our model uses supervised microclusters for maintaining the past history which helps in predicting class of a data point.

We begin by proposing a microcluster based scheme to maintain class statistics. Then the classifier is trained by merging data points in their corresponding classes (maintained using supervised microcluster). After that in the testing part the algorithms to predict the class label of the data point are discussed. Here the class label prediction and updating of classes is done in conjunction. Then the novel class detection scheme is proposed. Figure 3.1 shows the overall framework of the proposed Adaptive Naïve Bayesian (ANB) classifier scheme.

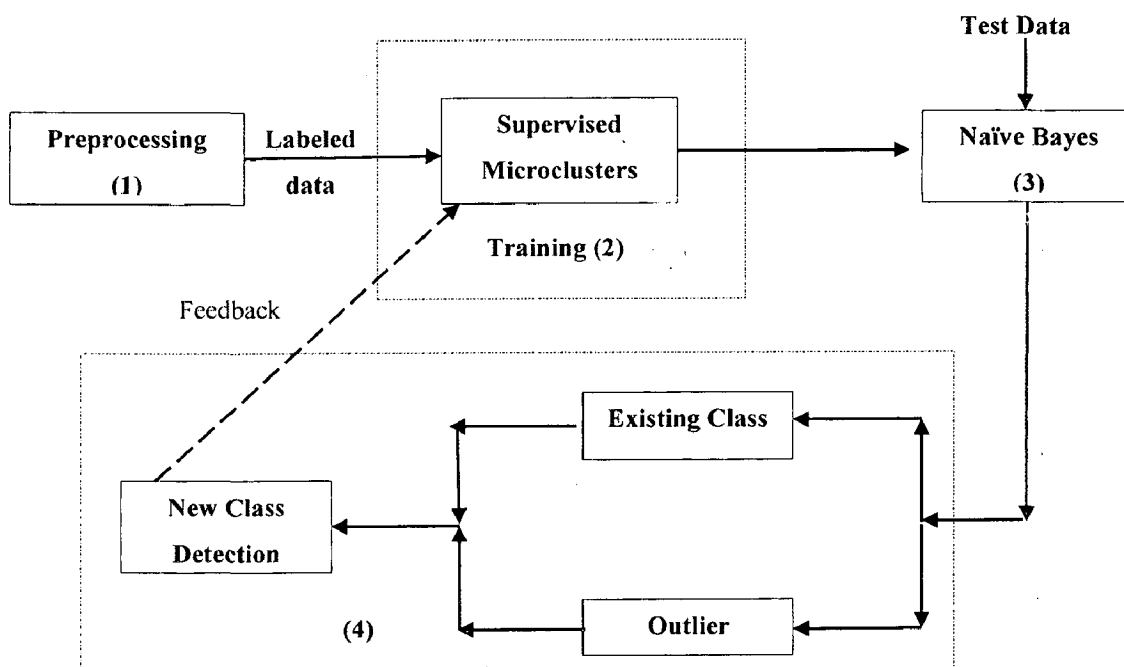


Figure 3.1 Framework for the proposed ANB scheme

3.1 Block 1 – Preprocessing

The data is converted into the *arff* format and the class labels are converted into numerals. This helps in training the classifier quickly as the in the training phase, the class can be accessed quickly by the given index.

3.2 Block 2 - Training the Classifier

After pre-processing the data, we build our classifier. The class statistics in the classifier are maintained using the supervised microcluster structure. The pre-processed data instances are then used to train the classifier. This phase builds the class model which can be used for testing using Naïve Bayes method.

3.2.1 Maintenance of Class Data Using Supervised Microclusters

We adopt the supervised microcluster (SM) model (as discussed in Section 2.3.1) to maintain our class past data so that it may be used for calculation of probabilities for the Bayesian classification later. Each class C will be represented by a corresponding SM consisting of four fields

- *Squared Sum (SS)* to maintain the sum of squared data values along each of the d dimensions of the data points belonging to the class C .
- *Linear Sum (LS)* to maintain the sum of data values along each of the d dimensions of the data points belonging to the class C .
- n to maintain the count of data points belonging to class C .
- *class_id* to maintain the unique identity of the class C .

The above representation provides an efficient way to maintain the summary of past data in the class as well as facilitates fast calculation of class probabilities that can be used to predict the class label of a data point.

3.2.2 Training the Adaptive Naïve Bayes Classifier

Training or model construction is the process by which we build the classifier. For this purpose a dataset D_{train} is taken that consists of d dimensional data points $X = \{x_1, \dots, x_d\}$. A set S of classes is maintained which is initialized to be empty. Each of the data point X has a class label $class_id_X$. These class labels are used to train classifier in the following way. Consider a data point X having class $class_id_X$

1. If the class having $class_id$ as $class_id_X$ already exists in set S then merge X in that class using algorithm mentioned in Figure 3.1.
2. If no such class exists, then create an empty new class C_{new} using the SM model as discussed in Section 3.1. Merge X in C_{new} using algorithm mentioned in Figure 3.1.

Merging Data Point in Class

Merging a data point in class is a fairly easy task now as the class uses is implemented as a supervised microcluster (SM). We increment the count by 1 and add all the squared data values of X along d dimensions in SS and add all the data values (as such) of X along d dimensions in LS . A pseudocode for the merging algorithm is shown in Figure 3.2.

Algorithm 1. Merge-point (C, X)

X: the data point
C: the class where data point X has to be merged
d: the number of dimensions

begin:

1. $C.n = C.n + 1$
2. **for** i from 1 to d
3. $C.SS[i] = C.SS[i] + X[i]*X[i]$
4. $C.LS[i] = C.LS[i] + X[i]$
5. increment i by 1
6. **end for**

end:

Figure 3.2 Pseudocode for merging a data point in class.

The entire training process as discussed above has been put together in Figure 3.3.

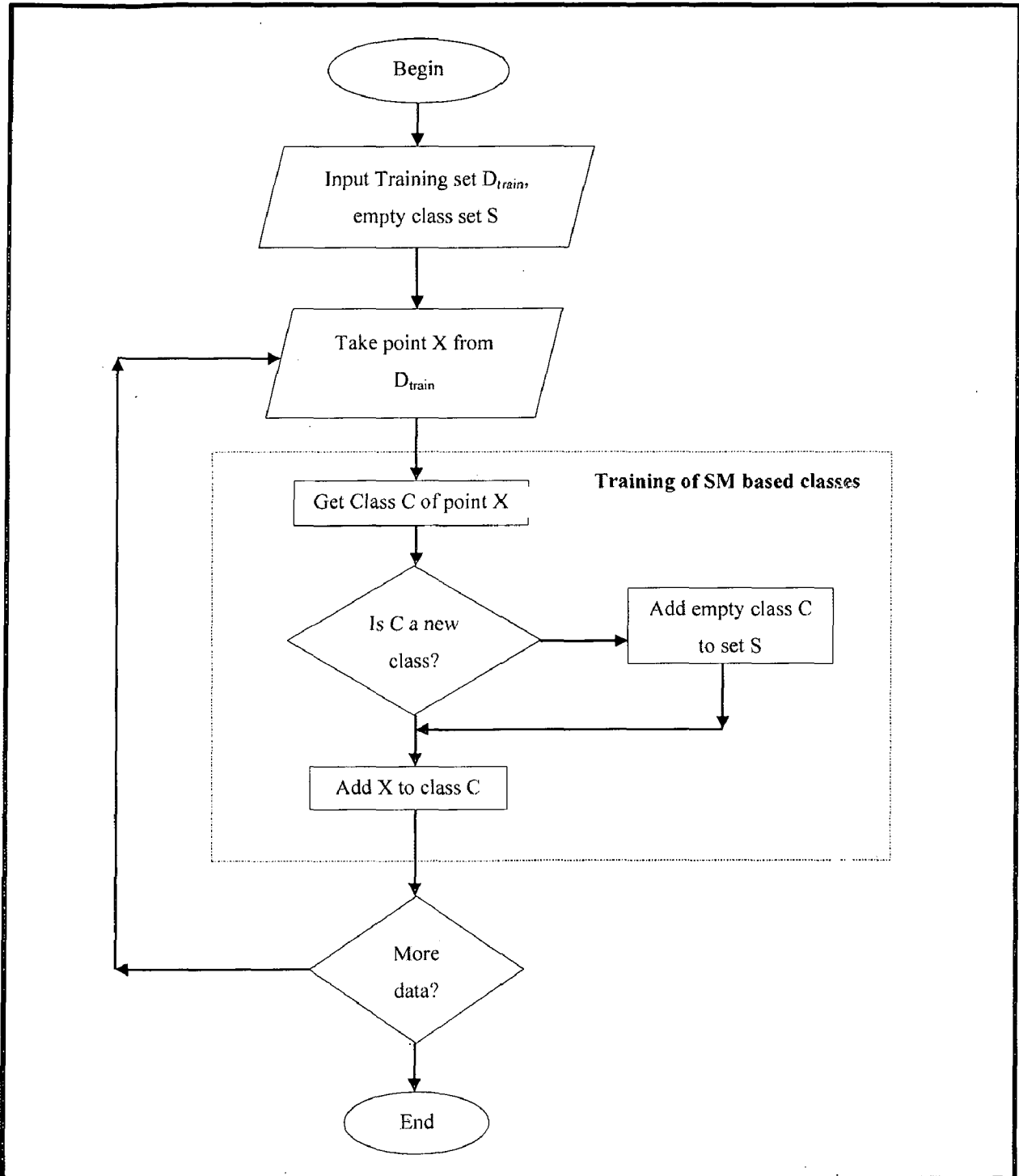


Figure 3.3 Training process for Adaptive Bayesian classification scheme

3.3 Block 3 - Class Prediction Using the Adaptive Naïve Bayes Classifier

We begin by describing methods for calculation of class probabilities and prediction of class label for a given data point. Class prediction without new class detection can be done in this phase. The novel class detection scenario is discussed in the next section.

3.3.1 Calculation of Class Probabilities

For a given class C and attribute k , the mean $\mu_{C,k}$ and standard deviation $\sigma_{C,k}$ of the values of all the data points can be calculated from the following equations:-

$$\mu_{C,k} = C.LS[k]/C.n \quad (3.1)$$

$$\sigma_{C,k} = \sqrt{\frac{C.SS[k]}{C.n} - \frac{C.LS[k]^2}{C.n^2}} \quad (3.2)$$

Now using $\mu_{C,k}$ and $\sigma_{C,k}$ as calculated from above equations (Eq. 3.1 and 3.2) and Eq. 2.6 and 2.7 we can calculate $P(x_k|C)$, the probability that a data point X having value x_k along the k^{th} dimension belongs to class C . Since naïve Bayes classifier assumes class conditional independence we just need to calculate $P(x_k|C)$ for $1 \leq k \leq d$ and using Eq. 2.5 we can find $P(X|C)$, the prior probability that indicates the likelihood that given a class C , an element with values same as that of X occurs in the class.

Using the above discussed methodology, we can develop algorithm to find the prior probabilities and consequently posterior probabilities which will lead us to class prediction for a given data point. The pseudocode for computation $P(X|C)$ is presented in Figure 3.4. We use a local variable *prob* initialized to zero for computing the required value. We iterate a loop through all dimensions and calculate dimensional probability $P(x_i|C)$. We take logarithm of this value and add it to *prob*. The logarithm approach is used avoid the problem of diminishing fractions as our system may fail to represent a value very close to zero and treat it as zero. Finally *prob* has the required value which is then returned from the method. The complexity of the algorithm is $O(d)$.

```

Algorithm 2. getDGCProbability( X, C )
X: data point
C: class

begin:
1. prob = 0
2. for i from 1 to d
3.      $\sigma_i = C.LS[i]/C.n$ 
4.      $\mu_i = \sqrt{\frac{C.SS[i]}{C.n} - \frac{C.LS[i]^2}{C.n^2}}$ 
5.     prob = prob + log( g(xi,  $\mu_i$ ,  $\sigma_i$  ) )
6. end for
7. return prob
end:

```

Figure 3.4 Pseudocode for algorithm that computes the data given class probability, $P(X|C)$

3.3.2 Prediction of Class Label for a given Data Point

After the computation of prior probabilities we need to compute the posterior probability $P(C|X)$ that indicates that given a data point X , the likelihood that X belongs to class C . Finally the class showing the highest posterior probability will be the predicted class for the given data point X . Figure 3.5 shows the pseudocode for the computation of class given data point X .

```

Algorithm 3. getClassGivenData ( X, S )
X: data point
S: set of classes
M: number of classes

begin:
1. mi = 1
2. mp = getDGCProbability( X,S[1] )+log(P( S[1] ))
3.
4. for i from 2 to M
5.     p = getDGCProbability(X,S[i] )+log( P( S[i] ) )
6.     if( p > mp )
7.         mp = p
8.         mi = i
9.     end if
10.    increment i by 1
11. end for
12. return mi
end:

```

Figure 3.5 Pseudocode for algorithm computes the most probable class C for the given data point X

The algorithm takes the data point X , the set of classes S as inputs. Another variable M is used to denote the size of set S . Our algorithm computes the index of the class in S that has the maximum $P(C_i|X)$. We use index variable mi to find this. For each class we use function Algorithm 2 (Fig. 3.3) to find the prior probability $P(X|C_i)$ and then its product with $P(C_i)$ is calculated. In actual implementation shown here we have computed this using logarithm because of the problem discussed in Section 3.3.1. So, in each iteration i , $P(C_i|X)$ is computed and is compared with previous maximum denoted by mp . In case, it is found higher than mp , we update mi and mp to i and $P(C_i|X)$ respectively. Finally, mi is returned from the method. The complexity of step 5 is $O(d)$ and this step is repeated $M-1$ times. As M is the size of S , we can also denote it as $|S|$. Hence the complexity of Algorithm 3 becomes $O(|S|d)$.

Using the above described methods (Section 3.3.1 and 3.3.2), we can formulate the scheme for class prediction using our adaptive naïve Bayesian classifier. We have described two different schemes here, one without novel class detection and one with novel class detection. The methodologies discussed in above sections (Section 3.3.1 and 3.3.2) apply to both as we will see.

3.3.3 Class Prediction without Novel Class Detection

In the Data stream scenario, prediction and updating needs to be performed in conjunction so that the model may be sensitive to the underlying changes thus accounting for the *concept drift*. In this scheme we assume that our classifier has been trained with instances of all the classes in the training or model construction phase. So the concept drift might occur in the properties of data instances belonging to the class but the possibility of the emergence of a new class has been ruled out. The entire process has been depicted in flow diagram shown in Fig. 3.6. We have a set S of classes created after the training phase where we maintain statistics about the data instances belonging to that class. A data stream D (can be a static dataset too) having N data instances is taken as input. Each data instance $X = \{x_1, \dots, x_d\}$ is a d dimensional data point. For each data point X , steps of the process can be outlined as follows:-

1. Calculate the C (or $index_C$) using the Algorithm 2 (Fig. 3.4) and Algorithm 3 (Fig 3.5).
2. Call a **MergePoint**($S[index_C], X$), in order to merge the point X in set S . The idea involved here is that when we merge X in the predicted class, we account for the *concept drift* occurring in the data stream and keep our model updated.

3. Check for another data point. If any more data, then goto step 1.
4. Exit.

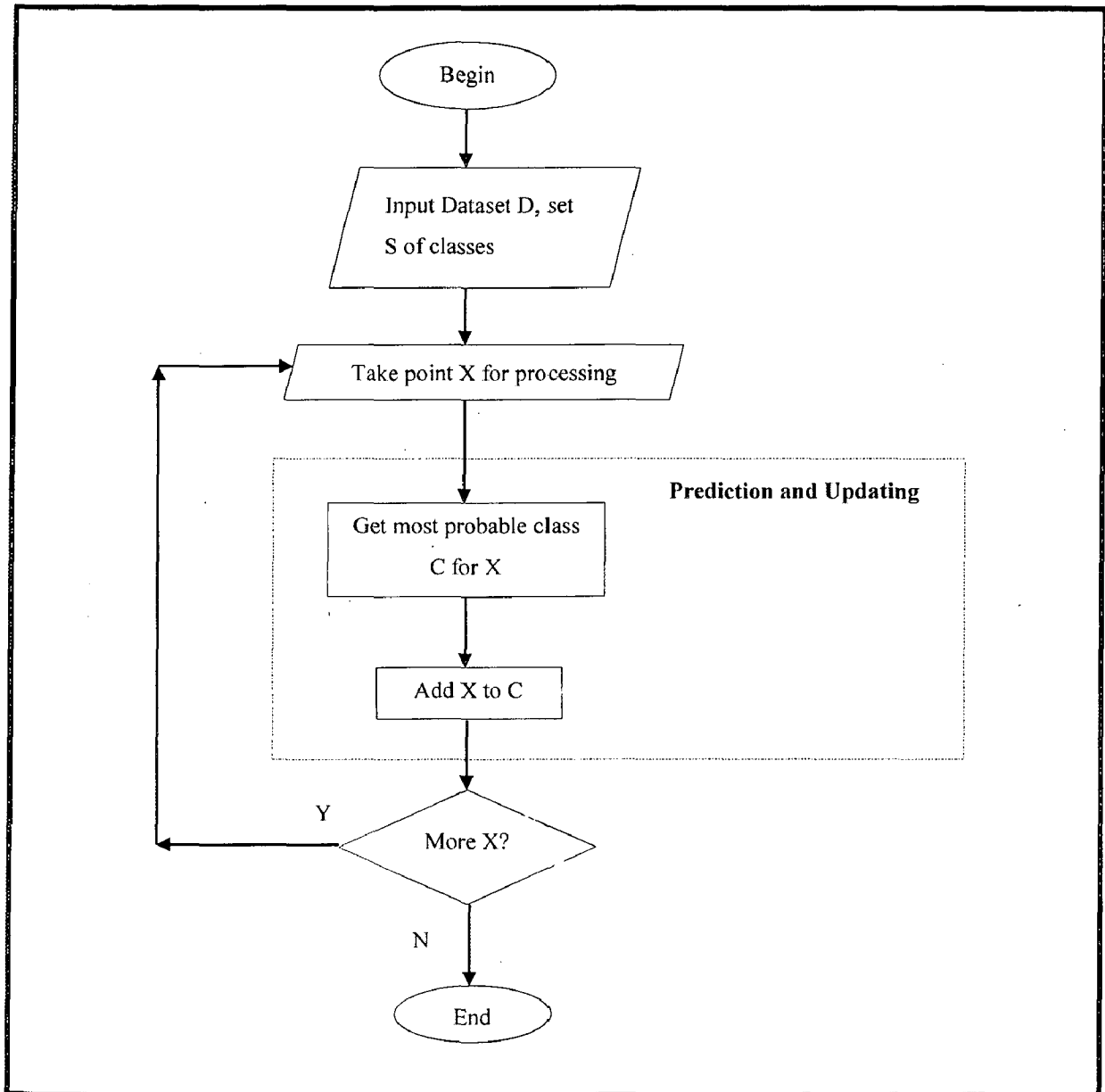


Figure 3.6 Class Prediction and Model Updating during the testing phase of Adaptive Naïve Bayes Classifier

3.4 Block 4-Class Prediction with Novel Class Detection

This is the generalized classification scenario for data stream where the assumption that no new class will emerge during the classification of data stream is dropped. Here we have to take care of the fact that our classifier may not be trained with all the classes that might emerge during the classification of data stream. This problem is referred to as “concept evolution” [33]. To handle this problem, the stream is divided into chunks of data and updating of classes is delayed till the end of the chunk.

Consider the scenario depicted in Figure 3.7. Here the classifier is trained with two classes. Suppose data points of a third class arrive during the classification process. Since naïve Bayesian classifier uses the principle of *maximum likelihood*, it cannot be used as such in this scenario as it will classify the other class data points as members of either one of those classes. Thus first of all our classifier needs to be trained to detect an outlier.

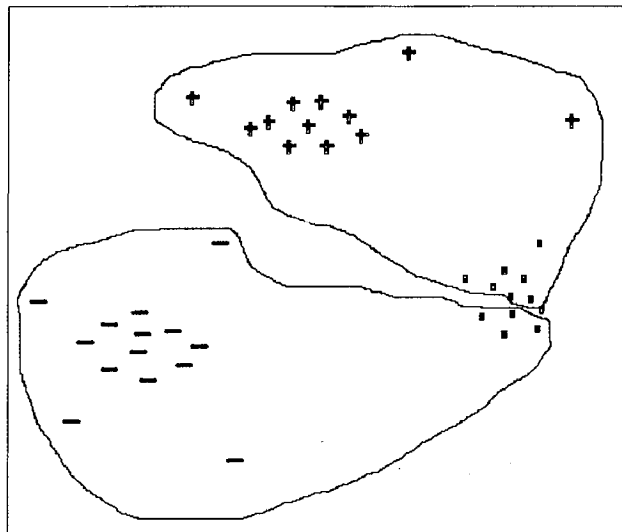


Figure 3.7 Inability of Bayesian methods to identify new class data points

3.4.1 Identification of Outliers

Since we are using naïve Bayesian classifier and maintaining class data statistics using supervised microclusters, we have to identify outliers for a class C using some threshold on probability $P(X|C)$ where X is the data point. Now, we define a term *mean probability* (μ_C) of a class.

Definition 3.1 The term *mean probability* (μ_C) of a class is defined as follows.

$$\mu_C = \frac{\sum_{i=1}^N P(X_i|C)}{N} \quad (3.3)$$

Here N is the number of data points belonging to class C . *Mean probability* (μ_C) of a class C indicates the average of belongingness of all the points to class C . We use μ_C as a parameter in our following discussion.

Using μ_C (Eq. 3.3), we can define outlier as follows:-

Definition 3.2 A data point X is considered an outlier if it satisfies the following condition:-

$$P(X|C) < \beta\mu_C \quad (3.4)$$

Where C is the predicted class for X and β is a constant value. If the above condition is not satisfied than the data point X is labeled as a member of class C .

After detecting the outliers we need to find whether they belong to a new class or not. The conditions to establish arrival of a new class are discussed in the following section.

3.4.2 Detecting a New Class

After processing the entire data chunk we have a set of outliers, X_{out} . The test for the occurrence of a new class C_{new} will be conducted if the following condition is satisfied:-

$$|X_{out}| > q \quad (3.5)$$

where q is a pre-defined constant which defines a minimum threshold for occurrence of a new class. Next the concept of "nearest neighborhood" is introduced.

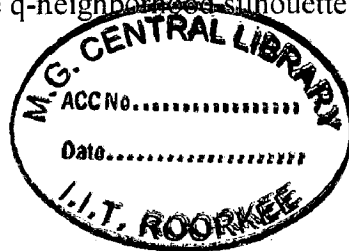
Definition 3.3 ($\lambda_{c,q}$ -neighborhood). $\lambda_{c,q}(x)$ is the set of q nearest neighbors of x within class c [33].

Let $\bar{D}_{c,q}$ be the mean distance from x to $\lambda_{c,q}(x)$, i.e.,

$$\bar{D}_{c,q} = \frac{1}{q} \sum_{x_i \in \lambda_{c,q}(x)} D(x, x_i) \quad (3.6)$$

where $D(x_i, x_j)$ is the euclidian distance between the data points x_i and x_j .

To detect the presence of a new class, we use q -neighborhood silhouette coefficient (q -NSC) as a cohesion metric which is defined as follows.



Definition 3.3 (*q*-NSC). *q*-NSC (x), or *q*-neighborhood silhouette coefficient of x can be defined as follows [33]:-

$$q\text{-NSC}(x) = \frac{\bar{D}_{c_{pred},q} - \bar{D}_{c_{out},q}}{\max(\bar{D}_{c_{pred},q}, \bar{D}_{c_{out},q})} \quad (3.7)$$

where $\bar{D}_{c_{pred},q}$ is the mean euclidian distance of x from $\lambda_{c_{pred},q}$ and $\bar{D}_{c_{out},q}$ is the mean euclidian distance of x from $\lambda_{c_{out},q}$. Here C_{pred} is the predicted class label for the x . The value of *q*-NSC lies between $[-1, 1]$. The positive value indicates that x is closer to outlier instances (more cohesion) and farther away from existing class instances (more separation).

Condition for New class: We say that a new class exists if there are atleast $q' (> q)$ x_i 's such that $x_i \in X_{out}$ with positive *q*-NSC. A new class is made using these outliers the set S is updated accordingly.

The scenario of occurrence of a new class and its identification because of cohesion among them can be visualized using Fig. 3.8.

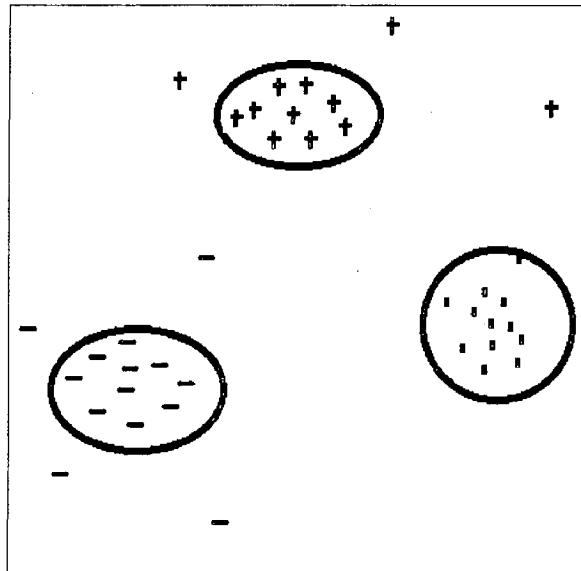


Figure 3.8 Occurrence of a cohesive set of outliers in the chunk with identification of new class

Now using the above discussion and putting together all the pieces, we can extend the class prediction and model updating (Fig. 3.6) discussed in Section 3.3.3 as shown in Figure 3.9.

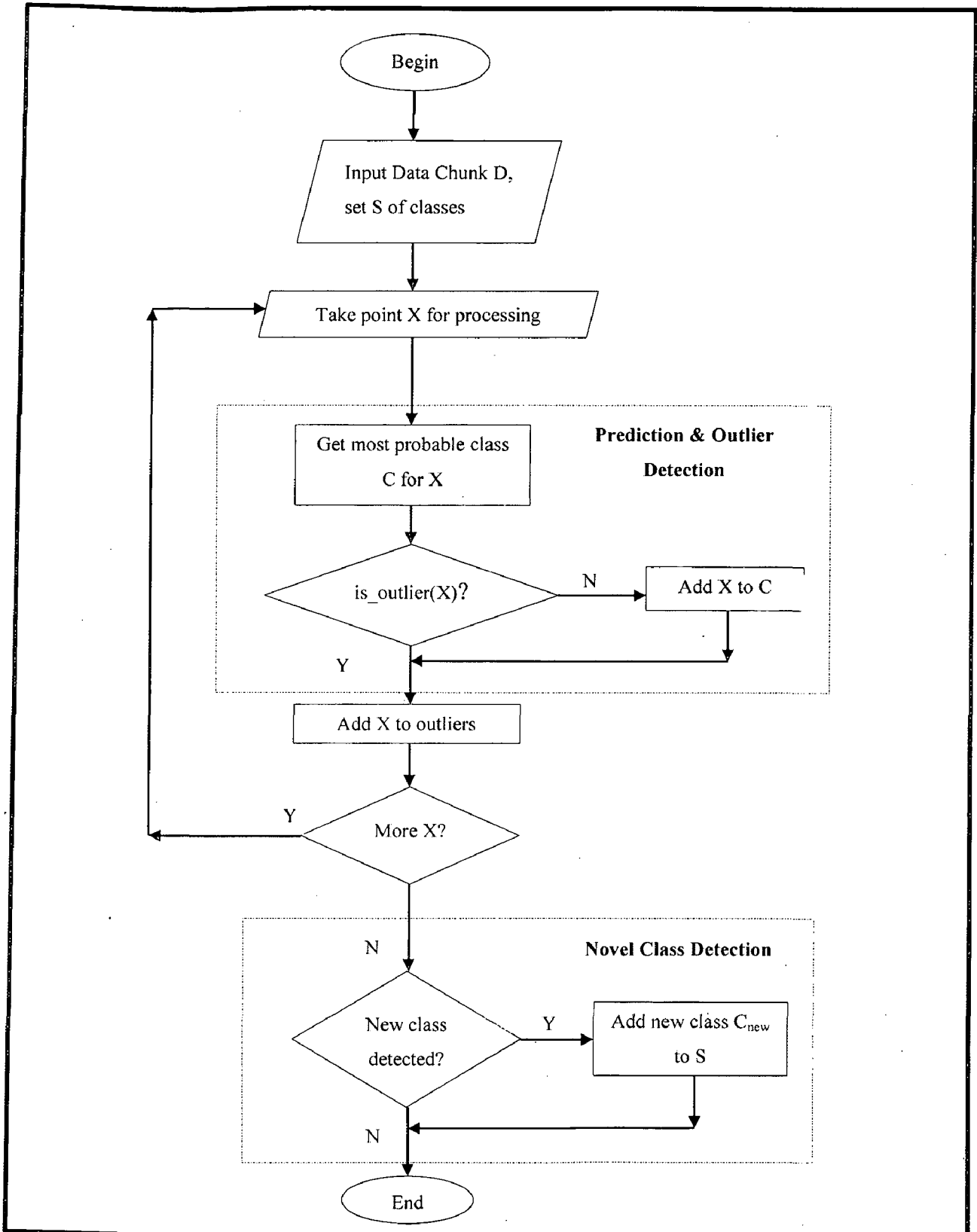


Figure 3.9 Class Prediction and Model Updating including New Class Detection in Adaptive Naïve Bayes Classifier

After prediction of classes and determination of outliers, the set of outliers is tested for condition of cohesion to determine the existence of a new class. If the new class is found, a new class C_{new} is added to S . All the corresponding outlier points are then added to this new class. Thus our model is now trained with C_{new} .

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 Implementation

The implementation of the proposed algorithms and all its pre-requisites are coded using C++ programming language. The rich set of third party open source libraries (e.g. Boost Libraries), containers (like Vectors) with built in methods and rich file handling features are the main reasons for choosing C++ as the language. Its modularity and object oriented nature gives the freedom to individually build and test the modules.

4.2 Design and Implementation of Classes based on Supervised Microclusters

As discussed in Section 3.2, the classes (of our Adaptive Naïve Bayes Classifier) are maintained using supervised microclusters (Refer to Section 2.3.1). For actual implementation, these microclusters (classes of classifier) are represented using *Class* (C++) named *NBClassContinuous*. The following data members were used to maintain the past data statistics:

Table 4.1 Data members used to implement the supervised microclusters.

Data Members	Description
vector< double > SquaredSums	To store <i>SS</i> as described in Section 3.1.
vector< double > LinearSums	To store <i>LS</i> as described in Section 3.1.
unsigned int numberOfPoints	To store <i>n</i> as described in Section 3.1.
int class_id	To store <i>class_id</i> as described in Section 3.1.
double probSum	Used (in novel class detection only) for computation of <i>mean probability</i> μ_C (Refer Definition 3.1) of the class.

In C++, vectors provide an elegant alternative for arrays. They have rich set of constructors and utility methods. They can be easily operated upon by using C++ STL algorithms. Also, vectors show good compatibility with external libraries. Thus vectors make an excellent choice

considering the need to perform basic linear algebra methods on the data members for various calculations as described in Sections 3.3 and 3.4. Table 4.2 gives a description of some important member functions.

Table 4.2 Some important member functions used in the implementation of the supervised microclusters.

Member Functions	Description
NBClassContinuous (unsigned int <i>_numberOfAttributes</i>)	Class constructor that takes the number of attributes/dimensions as input to initialize the data members
void mergePoint (Point& p)	Function that takes a data point as reference parameter and implements Algorithm 1 of Fig. 3.1
double getProbability (Point& p)	Function that takes a data point as reference parameter and implements Algorithm 2 of Fig. 3.3
double getMeanProb ()	(Used in novel class detection only) To get mean probability μ_c as defined in Definition 3.1.

For all the above mentioned functions *Point* was typedef'ed as `vector<double>`.

4.3 Implementation of Training and Testing phases of Classifier

The set of classes, *NBClasses* is maintained as `vector< NBClassContinuous >`.

In the training phase, the data was read from a text file using file handling operations. Then the read text was processed using [35] and the data point *X* and class label *C* were extracted. This data point was then merged into the entry corresponding to class *C* in *NBClasses* using *mergePoint* function. This way our model is trained with the training set.

In the testing phase, after reading data and getting the point using [35], we compute the prior probabilities using *getProbability*. We assume equal probabilities $P(C)$ for each class in our

implementation and thus ignore this factor. After that using Algorithm 3 (Fig 3.5), we calculate the class label C_p for our data point. This data point is then merged into the class C_p using *mergePoint* to update our classifier.

In novel class detection, the above testing process is modified as follows. Instead of reading, a single point at a time, we take in the entire data chunk (of a pre-specified size) from the file. The entire chunk is then processed to get the data points. We compute the outliers using the Definition 3.1 and 3.2. This is implemented using *getMeanProb* function of classes. The outliers identified and the class data points are maintained using a vector of *indices* to the data points stored in chunks. We then compute the q-NSC for each outlier. If the q-NSC is found to be positive then the point is declared to be a *p_outlier*. After processing the entire outlier set with similar procedure, if the number of *p_outliers* is found to be higher than q , then a new class (C_{new}) is created and added to set S . All the *p_outliers* are then merged into C_{new} , using the *mergePoint* method.

4.3.1 Using WEKA for Comparative Analysis

WEKA [36] stands for Waikato Environment for Knowledge Analysis. This tool was written in Java, developed at the University of Waikato, New Zealand. It provides many implementation s of many machine learning algorithms and is available under GNU General Public License. We used WEKA for comparing the results of existing popular classification methods with our proposed work. The WEKA explorer after opening the iris dataset [36], has been shown in Fig. 4.1.

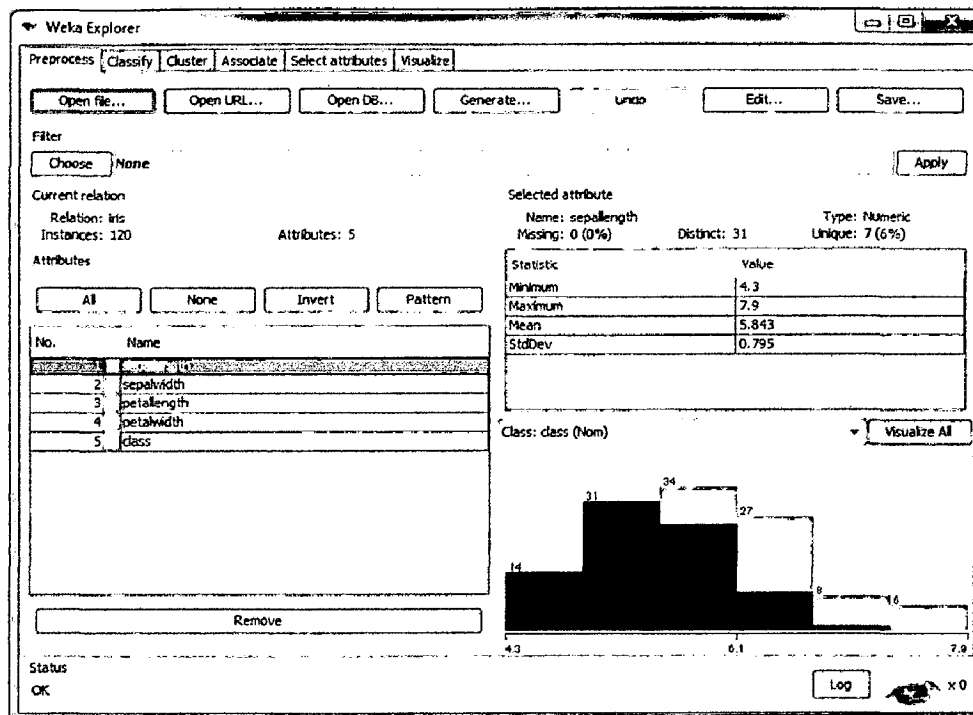


Figure 4.1 The WEKA explorer after opening the iris dataset.

WEKA generally processes data that is available in a single flat file. WEKA has a good collection of static algorithms but currently, it lacks implementation of data stream mining algorithms.

CHAPTER 5

RESULTS AND PERFORMANCE ANALYSIS

In this chapter, we verify the proposed approach by testing it on some existing datasets. The proposed scheme is applied on some well-known datasets, both static and streaming. The results obtained are stated in the following sections. We begin by giving a description of datasets followed by comparative analysis and results obtained.

5.1 Datasets

For the discussion of obtained results, we divide our datasets into two categories- static datasets and data streams. Two static datasets- *Spambase* dataset and *Iris* dataset, were used for analysis. The comparative analysis of Naïve Bayes classifier with the proposed method (ANB) has been done using static datasets indicating that ANB can be applied in static data environment as well. *KDD Cup'99 IDS Dataset* is used as streaming data because of its massive size. All these datasets were taken from UCI Machine Learning repository [37].

5.1.1 *Spambase* Dataset

Dataset consists of 4601 data instances describing e-mails. Each data point has 57 continuous attributes and 1 nominal class attribute indicating whether the e-mail is spam or not. The task is to predict whether a given data point represents a spam or not (binary classification).

5.1.2 *Iris* Dataset

This is a popular database in machine learning literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Each instance has 4 continuous attributes. The aim is to predict the class of iris plant using the 4 attributes.

5.1.3 *KDD Cup'99 IDS* Dataset

We used the 10% Corrected version of the dataset that consists of approximately 494021 data points. Each data instance has total 42 attributes, including 34 continuous attributes, 7 categorical

attributes and 1 class attribute. In our work, we have used only the 34 continuous attributes for classification. Each data point represents either attack or a normal connection. Attacks can be classified into four major categories [38]:-

- DOS (denial-of-service)
- R2L (unauthorized access from a remote machine)
- U2R (unauthorized access to local superuser privileges)
- PROBING (surveillance and other probing).

Thus there are total 5 classes including the normal scenario. All the above discussed datasets have been summarized in Table 5.1.

Table 5.1 Summary of the datasets used in our results.

S. No.	Dataset	No of Data Points	Attribute Description	No. of Classes
1	Spambase	4601	57 continuous; 1 class attribute	2
2	Iris	150	4 continuous; 1 class attribute	3
3	KDD Cup'99 IDS (10%)	494021	34 continuous; 7 categorical; 1 class attribute	5

5.2 Performance of Adaptive Bayesian Classification Scheme on Static Datasets

The proposed Adaptive Naïve Bayes scheme was applied to classify the above mentioned datasets in static environment. No new class detection was involved here. The datasets were split in training and testing sets with different ratios. The same task was performed using simple Naïve Bayes and CART and the percentage accuracy obtained (Table 5.2) clearly show the effectiveness of our proposed adaptive scheme (ANB).

Table 5.2 Comparison of accuracy obtained between proposed scheme and simple Naïve Bayes.

S. No	Dataset	Percentage split	Percentage Accuracy		
			Naïve Bayes	CART	ANB
1	Spambase	10-90	76.41	85.56	89.65
2	Spambase	20-80	80.95	86.77	90.63
3	Iris	25-75	88.33	92.5	94.17

The above results (Table 5.2) are graphically depicted in Figure 5.1.

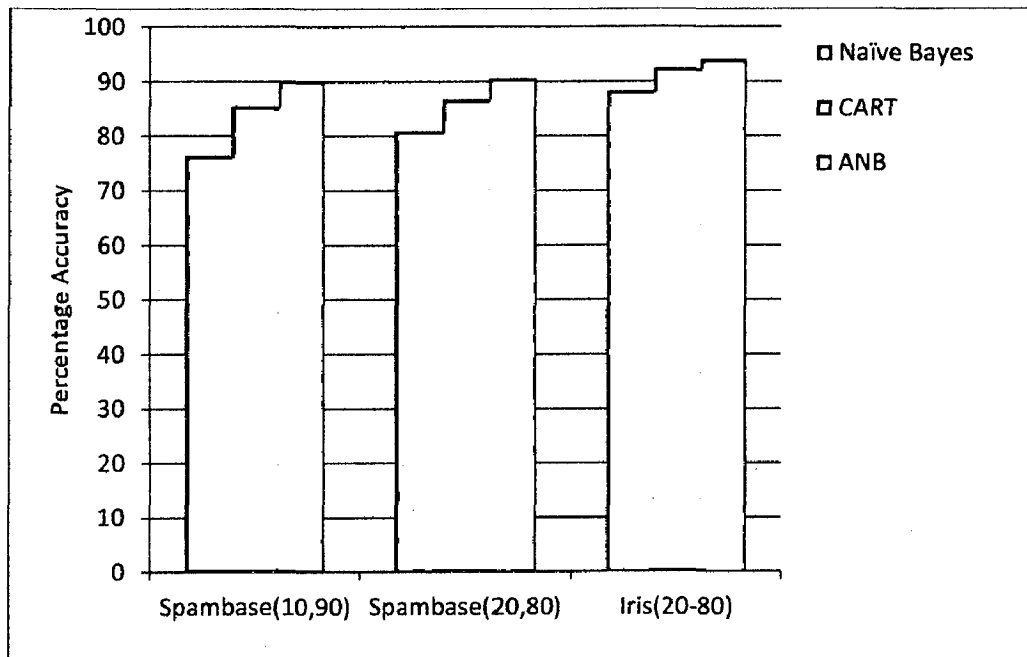


Figure 5.1 Graph showing the comparison of accuracy obtained between proposed scheme, simple Naïve Bayes and CART.

The comparison clearly shows the improvement in results when the size of training set was increased (Refer row 1 and 2 of Table 5.2) while in case of our proposed method, similar results were obtained in both cases. This shows the adaptive nature of our classifier as it adapts with the underlying changes in the test set and thus produces improved results.

5.3 Performance of Adaptive Bayesian Classification Scheme on Streaming Data

The proposed scheme was tested on KDD Cup'99 Network Intrusion dataset. The following results were obtained when the dataset was divided as 5% training set and 95% test set. These results are depicted in Fig 5.2. The figure shows the accuracy obtained for the five different classes as well as overall final accuracy as described in Section 5.1.3. Figure 5.2 clearly shows the effectiveness of our Adaptive Naïve Bayesian scheme. The higher accuracy is the result of the summary statistics maintained using supervised microclusters which enables our classifier to evolve with stream.

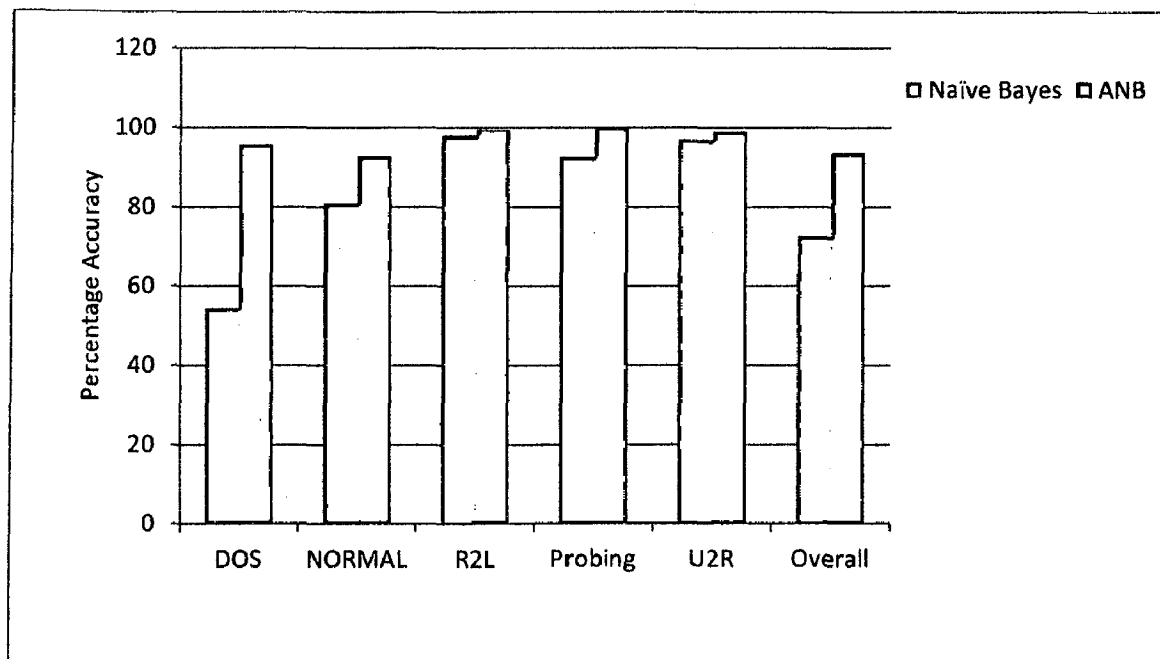


Figure 5.2 Graph showing the accuracy obtained by Naïve Bayes and proposed scheme on *Spambase* using 5% training set.

5.4 Performance of Novel Class Detection Scheme in Adaptive Naïve Bayesian Classification

For the purpose of result evaluation for new class detection scheme, we define a few metrics which are as follows:-

- (i) F_n : Percentage of new class instances misclassified as existing class.
- (ii) F_p : Percentage of existing class instances misclassified as new class.
- (iii) F_e : Percentage of existing class instances misclassified other than F_p .

The results were obtained on the KDD Cup'99 IDS dataset. The parameters used were $\beta = 1$ and $q = 50$. We divided the data stream into data chunks of size 2000 each. Each data chunk consisted of at most three and atleast two classes. The testing was done in parts as following. We first train our classifier with just 2 classes using the first two data chunks for training. Then we introduce data points of a new class in 3rd chunk and checked for results. After this we use the first three chunks to train our classifier and introduce another new class in 4th chunk and check for results. Similar procedure is repeated for the 5th chunk. The results obtained are shown in Table 5.3.

Table 5.3 Results obtained for each data chunk while testing new class detection scheme.

S. No.	No. of Classes Used For Training	No. of New class Instances	F_n (%)	F_p (%)	F_e (%)
1	2	1000	4.00	0.08	0
2	3	1500	2.67	1.47	25.22
3	4	1875	2.13	1.47	43.35

From the results in Table 5.3, it can be seen that our proposed ANB method is able to detect new class instances with low error rates.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In our work, we have adapted Naïve Bayesian technique for classification of data streams. Other than the constraints of memory and computation complexity in dealing with data streams, we might also encounter a change in the underlying data distribution. This change is referred to as *concept drift*. Our proposed scheme (ANB classifier) presents an efficient model that aims improving classification results by accounting for these factors. Naïve Bayes is based on probabilities and uses the principle of *maximum likelihood* to determine the class of a data point. Supervised microclusters are used to train the classifiers and efficiently summarize the past history of data points belonging to a class in the classifier. The methods for merging a data point into the supervised microcluster and calculating prior probabilities are developed. During the testing (prediction) phase, we perform the prediction and updating of our classifier in conjunction so that our classifier may be sensitive to the underlying changes. Thus an adaptive scheme has been developed which predicts the class of the data point and then adapts by merging the data point in the class.

The proposed scheme was modified for the accommodation of new class data instances. We have introduced concepts like *mean probability* and *outliers* using class membership probabilities. Then we have stated a condition for coherence for confirmation of a new class. Thus our classifier not only accounts for the underlying changes occurring in a class distribution it also accounts for the emergence of a new class. In this way, our model truly accounts for the *concept drift*.

The implementation of the proposed scheme was done in C++ and various methods were successfully implemented. Finally, results were obtained and they clearly show the impact of the developed adaptive strategy. The proposed new class detection scheme was also tested by training the classifier with different number of classes which has shown promising results.

6.2 Suggestions for Future Work

The proposed adaptive Naïve Bayesian scheme only deals with continuous attributes. There is scope to extend these ideas to deal with categorical attributes also. A way to do this may be to store the frequency counts of each class value for categorical attributes. Then use these counts to predict to calculate dimensional probabilities. After prediction of class for the given data point, these frequencies can be updated.

At some rare situations, ANB classifier may get too influenced by past data points. This might result in wrong class predictions of data points. Thus, there is a scope to develop a strategy to specify a *time horizon* such that the data points belonging to time before that may be removed from the classes and thus higher accuracy may be achieved. The microcluster framework facilitates the subtraction of data points easily. The issue that needs to be dealt with here is the strategy to maintain the past data on secondary storage.

REFERENCES

- [1] J. Han and M. Kamber, *Data Mining Concepts And Techniques*, Morgan Kaufmann, 2006.
- [2] Aggarwal, C.C.; Yu, P.S.; , "LOCUST: An Online Analytical Processing Framework for High Dimensional Classification of Data Streams." *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on* , vol., no., pp.426-435, 7-12 April 2008.
- [3] C. C. Aggarwal, "Data Streams: Models and Algorithms." Springer, 2007.
- [4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems." in *ACM PODS Conference Proceedings*, 2002.
- [5] C.C. Aggarwal, "A Framework for Diagnosing Changes in Evolving Data Streams." in *Proc. ACM SIGMOD Conf.*, pp. 575-586, June 2003.
- [6] M. Last, "Online Classification of Nonstationary Data Streams." *Intelligent Data Analysis*, Vol. 6, No. 2, pp. 129-147, 2002.
- [7] D.P. Helmbold and P.M. Long, "Tracking Drifting Concepts by Minimizing Disagreements." *Machine Learning*, No. 14, pp. 27-45, 1994.
- [8] T. Abdessalem, R. Chiky, G. Hébrail, J.L. Vitti, "Using Data Stream Management Systems to analyze Electric Power Consumption Data." *International Workshop on Data Stream Analysis (WDSA)*. Caserta (Italie), March 2007.
- [9] S. Muthukrishnan, "Data Streams: Algorithms and Applications." In *Proceedings of the 14th Annual Association for Computing Machinery - Society for Industrial and Applied Mathematics (ACM – SIAM) Symposium on Discrete Algorithms*, Baltimore, USA, pp. 58-75, 2003.

- [10] L. Breiman, "Random Forests." *Machine Learning Journal*, vol. 45, no. 1, pp. 5-32, 2001.
- [11] R. Duda, and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
- [12] J. H. Friedman, "A recursive partitioning decision rule for non-parametric classifiers." *IEEE Trans. on Computers*, vol. C-26, pp. 404-408, 1977.
- [13] S. K. Murthy, "Automatic construction of decision trees from data: a multidisciplinary survey." *DMKD Journal*, number 2, pp. 345-389, 1998.
- [14] J.R. Quinlan. "Induction of Decision Trees." *Machine Learning*, Vol. 1, No. 1, pp. 81-106, 1986.
- [15] J. R. Quinlan. "C4.5: Programs for Machine Learning." *Morgan Kaufmann*, 1993.
- [16] L. Breiman, J.H. Friedman, R.A. Olshen, & P.J. Stone. *Classification and Regression Trees*, Wadsworth, 1984.
- [17] P. Clark and T. Niblett. "The CN2 induction algorithm." *Machine Learning*, 3:261–283, 1989.
- [18] W. Cohen. Fast effective rule induction. In *Proc. 1995 Int. Conf. Machine Learning (ICML'95)*, pages 115–123, Tahoe City, CA, July 1995.
- [19] O. Maimon and M. Last. "Knowledge Discovery and Data Mining," *The Info-Fuzzy Network (IFN) Methodology*, Kluwer Academic Publishers, 2000.

- [20] P. Domingos and G. Hulten. "Mining High-Speed Data Streams." *In Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining*, 2000.
- [21] P. Domingos, G. Hulten, and L. Spencer. Mining time-changing data streams. *In Proc. of the 2001 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp 97–106, 2001.
- [22] V. Ganti, J. Gehrke, and R. Ramakrishnan. "Mining Data Streams under Block Evolution." *SIGKDD Explorations*, vol. 3, 2002.
- [23] H. Wang, W. Fan, P. Yu and J. Han. "Mining Concept-Drifting Data Streams using Ensemble Classifiers." *in the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington DC, USA, Aug. 2003.
- [24] G. Hulten, L. Spencer, and P. Domingos. "Mining time-changing data streams." *In Proc. 2001 ACM SIGKDD Intl. Conf. Knowledge Discovery in Databases (KDD'01)*, San Francisco, CA, Aug. 2001.
- [25] M. Last. "Online Classification of Nonstationary Data Streams" *Intelligent Data Analysis*, Vol. 6, No. 2, pp. 129-147, 2002.
- [26] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. "On Demand Classification of Data Streams." *In Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining*, Seattle, WA, Aug. 2004.
- [27] C.C. Aggarwal, J. Han, J. Wang, and P. Yu. "CluStream: A Framework for Clustering Evolving Data Streams," *in Proc. Int'l Conf. Very Large Data Bases*, pp. 81-92, Sept. 2003.

- [28] C.C. Aggarwal, J. Han, J. Wang, and P. Yu. "A framework for on-demand classification of evolving data streams" *Knowledge and Data Engineering, IEEE Transactions on*, vol.18, no.5, pp. 577- 589, May 2006.
- [29] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases" in *Proc. 1996 ACM SIGMOD Int'l Conf. Management of Data*, pp. 103-114, June 1996.
- [30] M. Markou and S. Singh. "Novelty Detection: A Review. Part 1: Statistical Approaches, Part 2: Neural Network Based Approaches," *Signal Processing*, vol. 83, pp. 2481-2497, 2499-2521, 2003.
- [31] S.J. Roberts. "Extreme Value Statistics for Novelty Detection in Biomedical Signal Processing," in *Proc. Int'l Conf. Advances in Medical Signal and Information Processing*, pp. 166-172, 2000.
- [32] D. Yan Yeung and C. Chow. "Parzen-Window Network Intrusion Detectors," in *Proc. Int'l Conf. Pattern Recognition*, pp. 385-388, 2002.
- [33] Masud, M.M., Jing Gao, Khan, L., Jiawei Han, Thuraisingham, B.M., "Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints," *Knowledge and Data Engineering, IEEE Transactions on*, vol.23, no.6, pp.859-874, June 2010.
- [34] Safavian, S.R., Landgrebe, D., "A survey of decision tree classifier methodology," *Systems, Man and Cybernetics, IEEE Transactions on*, vol.21, no.3, pp.660-674, May/Jun 1991.

- [35] Jeremy Siek John R. Bandela. "Boost Tokenizer Class- Boost 1.46.1." Internet: http://www.boost.org/doc/libs/1_46_1/libs/tokenizer/tokenizer.htm, Feb. 16, 2008 [April 10, 2011].
- [36] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten; The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.
- [37] A. Frank and A. Asuncion,"University of California Machine Learning Repository [<http://archive.ics.uci.edu/ml/>]," School of ICS, California, 2010.
- [38] M. Tavallae, E. Bagheri, W. Lu, A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA)*, 2009.

LIST OF PUBLICATIONS

- [1] Parthsarathi Mishra and Durga Toshniwal. "Classification of Data Streams Using Adaptive Naïve Bayes Algorithm." In *Proc. Of International Conference on Information Technology and Management (ICITM)*, 2011 (Accepted for publication in July, 2011).