2006

Gaussian Mixture Approach to Detect Drift

Mamidi Sree Kalyan Chakravorty

GAUSSIAN MIXTURE APPROACH TO DETECT DRIFT

BY

MAMIDI SREE KALYAN CHAKRAVORTY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

July 2006

MASTER OF SCIENCE THESIS
OF
MAMIDI SREE KALYAN CHAKRAVORTY

Approved:

Thesis Committee

Ankur Teredesai
Chair: Dr. Ankur Teredesai

Roger Gaborski
Reader: Dr. Roger Gaborski

Hans-Peter Bischof
Observer: Dr. Hans-Peter Bischof

ROCHESTER INSTITUTE OF TECHNOLOGY

July 2006

Library Rights Statement

In presenting the thesis *Gaussian Mixture Approach to Detect Drift* in partial fulfillment of the requirements for an advanced degree at the Rochester Institute of Technology, I agree that the Library shall make it freely available for inspection. I further agree that permission for copying as provided for by the Copyright Law of the U.S. (Title 17, U.S. Code) of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

I hereby grant permission to the RIT Library to copy my thesis for scholarly purposes.

M. Sree Kalyan Chakravorty

Mamidi Sree Kalyan Chakravorty

Date

Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: Gaussian Mixture Approach to detect drift.

Name of author: Mamidi Sree Kalyan Chakravorty
Degree: Master of Science
Program:
College: Department of Computer Science.

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, _____, hereby **grant permission** to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: _____ Date: _____

Print Reproduction Permission Denied:

I, Mamidi Sree Kalyan Chakravorty, hereby **deny permission** to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: M. Sree Kalyan Chakravorty Date: _____

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive

I, Mamidi Sree Kalyan Chakravorty, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: M. Sree Kalyan Chakravorty Date: _____

Abstract

Historically it has been difficult to measure the deviation in the notion of a concept. Several schemes have been proposed to attack this challenging problem [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. The central notion of all these efforts is to detect the change point where the data mining model deviates significantly with respect to the data characteristics that it was trained or built on. The process of detecting such change points is often termed as concept drift. Current state of algorithms assume attribute independence, view the problem as a supervised learning problem and also need tagged data. The proposed algorithm does not make any assumption among attribute independence and uses the covariance summary to detect concept drift in an unsupervised setting. The algorithm proposed in this thesis monitors the underlying characteristics of the input data, maintains data summaries of the various snapshots in time and utilizes effective distance metrics to determine when concept drifts. The technique was evaluated against synthetic and real data sets.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Classification and clustering models often need to be updated to accommodate for changes in the underlying data characteristics. In recent years several research efforts have been devoted for detecting when and how the characteristics of the dataset change over time or over other factors influencing the generation of the data. The central notion of all these efforts is to detect the change point where the data mining model deviates significantly with respect to the data characteristics that it was trained or built upon. The process of detecting such a change point is often termed as concept drift. In this thesis an algorithmic framework is proposed that monitors the underlying characteristics of the input data, maintains data summaries of the various snapshots in time and utilizes the Kullback-Leibler divergence distance metric to determine the concept drift. Unlike algorithms [13] which assume independence among the attributes, the proposed algorithm keeps covariance information among the attributes while detecting drift. The covariance information is kept in the data summary and the algorithm operates on these summaries to detect the drift. Even though we assume that drift is time dependent, the framework is general enough to detect deviations with respect to any other parameters of the dataset.

Classifiers take raw data as input and produce class labels as output. This is illustrated in Figure 1. Correspondingly, in clustering, input data is organized into groups or clusters. Cluster labels are illustrated in Figure 2. An object belonging to a concept exhibits more similarity (less distant) with objects belonging to the same concept. On the other hand objects belonging to different concepts exhibit lesser similarity (more distant).

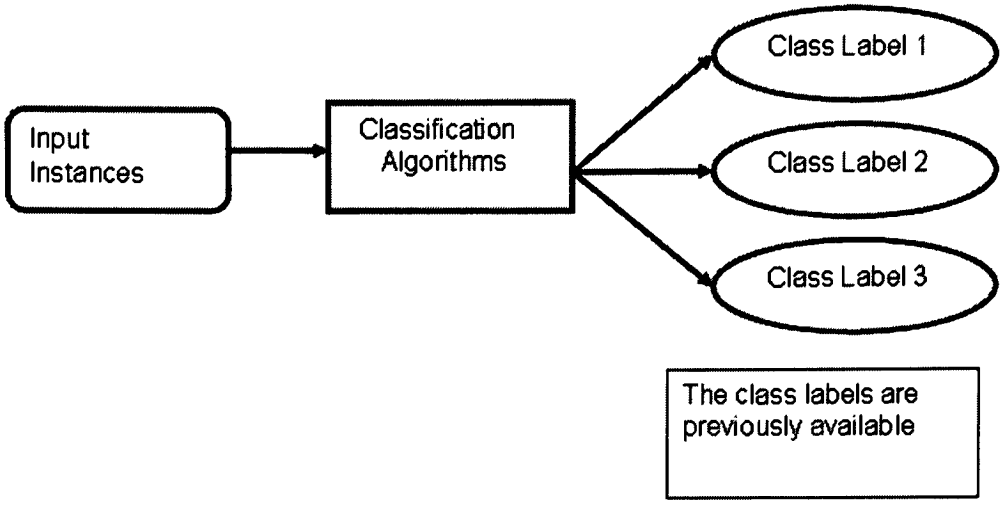

Figure 1: A class label represents a concept. Input instances are classified into different concepts by the classification algorithm when it assigns them a class label.

Thus, a concept is usually represented by a class or a cluster label. Formally a concept is defined by:

Definition 1.0.1 ***Concept:*** *A concept is a descriptor of the output of a data mining algorithm. In case of classification, the class label is a concept. Whereas in clustering, the cluster labels are the concepts.*

The concepts describing buying habits of customers for example are never constant. Such concepts are influenced by external and hidden factors. For example the concepts of fruit and dairy products are highly correlated in summer, especially around the month of July, since people tend to buy strawberries and cream. But this relationship is not exhibited throughout the entire year. Similar convergence and divergence is observed in various physical phenomena, in web access patterns, news feeds, financial time series etc. This notion of change of

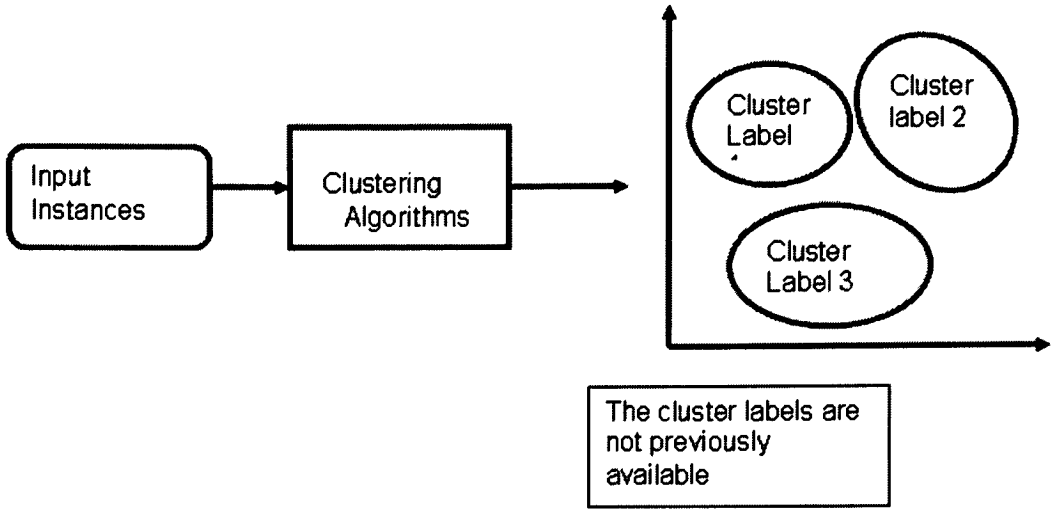

Figure 2: A cluster label represents a concept. Input instances are clustered into different clusters by the clustering algorithm when it assigns them a cluster label.

the concept or rather the constituent elements that come together to describe a concept is termed as concept drift. Cunningham et al., [14] formally define concept drift as follows:

Definition 1.0.2 *Concept Drift: Let* i_1.. i_n *denote the instances. Let* dx *denote the time of the arrival of instances. Let there be two concepts A and B for a sequence of instances. For instances* i_1 *to* i_k *the concept A was stable, after* i_k + dx *the concept B is stable. Between* i_k *and* i_k + dx *the concept is drifting between A and B. When* $dx = 1$ *the drift is called sudden drift in concept.*

Tsymbal et al., [15] categorizes concept drift as (a) Sudden Drift (b) Gradual Drift. Sudden drifts deal with instantaneous and abrupt changes. On the other hand gradual concept drifts could be classified as moderate and slow drifts.

Further concept drifts could be characterized by:

- Virtual concept drift where the underlying distribution of the data changes
- Real concept drift where the data actually gets shifted

Various learning algorithms [1, 4] have been used as base models to detect concept drift. The characteristics of the learning algorithm for detecting concept drift are [15]:

- Robustness to noise
- Tracking recurring contexts
- Quick adaptation to drifts

Klinkenberg et al., [16] proposes various indicators which can detect concept drift. These indicators are:

- Change in classifier accuracy
- Change in the properties of the classification model
- Change in the properties of the data distribution

In this thesis a new algorithm is proposed to detect unsupervised concept drift. This algorithm addresses the issues of scalability by summarizing the data points and then detecting clusters using the modified Expectation-Maximization(EM) [17] on the summaries. Further Kullback-Leibler divergence is used to detect the concept drift. From my understanding this is a novel way to detect concept drift.

1.1 Road map

Chapter 2 discusses the related work in the field of detecting concept drift.

Chapter 3 proposes the approach and discusses in detail the different phases of the algorithm.

Chapter 4 details the experiments being conducted on various datasets.

Chapter 5 concludes the thesis and proposes future work for concept drift.

Chapter 2

Related Work

The approaches for detecting concept drift could be broadly classified as [15]:

- Window-based
- Instance weighing scheme
- Ensemble methods

In window-based methods, instances relevant to the current concept are selected. A window of fixed or variable size is maintained over the current concept and concept drift is tracked by change in classifier accuracy. Instance weighing schemes deal with assigning weights to each instance. The weight assignment is based on how old the instance is with respect to the current concept. Ensemble methods use a combination of concept descriptions which are combined using a criteria like voting to select the most relevant concept.

In this chapter we will discuss various systems developed to detect concept drift in data.

2.0.1 Stagger

Stagger [1] proposed one of the preliminary methods of detecting concept drift. Concept is represented as a boolean function of the attribute values. Sufficient and necessity weights [18] are associated with a concept. The sufficiency and necessity conditions are expressed by (a) Logical Sufficiency or positive likelihood ratio, and (b) Logical Necessity or negative likelihood ratio [19]. Concept drift is tracked by modifying the weights in an incremental fashion using Bayesian statistics. This approach is classified as an ensemble- based

approach of detecting concept drift.

2.0.2 Forgetting

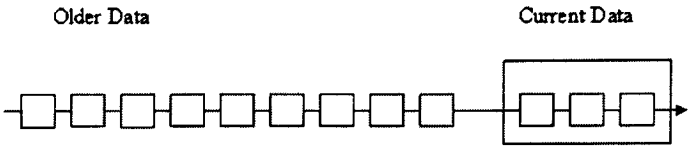

Figure 3: Window-based methods to detect concept drift. A window of fixed or variable size is maintained over the current data.

Salganicoff et al., [20] propose a *time-based forgetting* mechanism to detect concept drift. Here the concept descriptions are stored and a window of fixed size is maintained as new instances arrive. Arrival of a new instance results in deletion of the oldest concept description. Widmer and Kubat et al., [4] extend the approach proposed by Salganicoff [20]. In this scenario rather than deleting the oldest concept description, a new description is added if the learner detects that the classification is improved by the addition of a new description.

FLORA(Floating Rough Approximation) maintains a window over the current examples and learns new concepts and forgets old concepts as shown in Figure 3.

Concept Description in FLORA framework is described by three sets [4]:

- ADES (containing positive examples)

- NDES (containing negative examples)
- PDES (general items containing positive as well as some negative examples)

Maintaining these three sets assists in how positive and negative instances are handled. When positive instances arrive, they are either included in the ADES set, existing instances move from NDES to PDES or they are 'confirmed'. Negative samples are added to the NDES or might result in a transfer of items from NDES to PDES. The motivation of PDES is to keep the relevant examples in the set so that they can be recalled when required. The basic FLORA algorithm learns and forgets the instances from these three sets. This approach of detecting concept drift is classified as the window-based methods of detecting drift.

The problem with the FLORA framework was that it used windows of fixed size. FLORA2 overcame this problem by dynamically adjusting the window size. FLORA2 uses *heuristic* of *predictive performance* to keep the correct size of window [4]. Sharp drops in predictive performance and if the number of items in the ADES set increase are indicators of concept drift. FLORA was extended further in FLORA3 and FLORA4 for handling recurring contexts and noise respectively.

2.0.3 AQ-PM

AQ-PM proposed by Maloof et al., [7] is an online learning system which maintains partial memory of the past training instances. Partial memory systems iterates though all the instances and finds the instances which it misclassifies and includes the instances for future learning. The missed instances are included with instances in partial memory for the training sets. Concepts are

learned from these trained sets of data and the partial memory is updated. AQ-PM selects *extreme examples* (instances that lie on the boundary of concept descriptions) and places them at the boundaries of concept description. By selecting the training examples effectively partial memory systems detect the concept drift.

2.0.4 Multiple Windows

Lazarescu et al., [6] proposes multiple windows to track concept drift. The input to this method could either be labeled or unlabeled data. This algorithm defines concept drift in terms of consistency and persistence. Consistency refers to the change that occurs between consecutive instances of the target concept. Persistence is defined in terms of the size of the window, if the size is greater than half of the window size of the observed instances then the concept is persistent. It tracks concept drift based on three competing windows. .

Three windows are assigned to each concept. Windows of size small, medium and large are introduced. The size of the small window is fixed to S, the size of the medium window ranges between 2S to a maximum M and finally window of larger size ranges from 2M to L. The small window deals with detection of fast changing concepts, a window of medium size handles slower changing concepts and large window deals with very slow changing concepts. In addition to the three windows a persistence component is added to the concept. Window with the largest size is returned as the best concept for the window. The incoming data fills in the three windows. This algorithm can detect unsupervised concept drift, it uses K-means to label the data and then applies the multiple windows algorithm to track concept drift. As compared to FLORA this approach can

detect continuous change in the target concept. FLORA cannot handle continuous change in concept since it can handle only step-like changes, the single window of dynamic size determines whether a change occurs for the instances kept in the dynamic window. The estimate of rate of change in data is not kept in FLORA which results in FLORA being unable to detect continuous change in concept drift.

2.0.5 Support Vector Machines

Klinkenberg et al., [10] proposed a window-based method of detecting concept drift using support vector machines (SVM). Compared to the FLORA2 system, which dynamically adjusts the window size based on heuristics, this approach adjusts the window size dynamically through SVM. The algorithm works by training the input instances through SVM and computing $\xi\alpha$-estimates [10]. The $\xi\alpha$-estimators are based on the idea of leave-one-out estimation [10]. The first instance is removed from the input instances and the resulting instances are used for training to find a classification rule. This rule is tested on the instance which was left out, if the instance is not classified correctly, the instance is responsible for producing a leave-one-out error. This process is repeated for all the training examples. A window size with with the minimum $\xi\alpha$-estimate of the error rate is chosen.

The heuristic-based approach of detecting concept drift suffers from the following drawbacks [10]:

- Requires tuning of parameters
- Non transferable to other domains
- Lack of proper theoretical foundation

The SVM-based approach of detecting drift involves least parameterization and has a strong theoretical foundation.

2.0.6 SEA

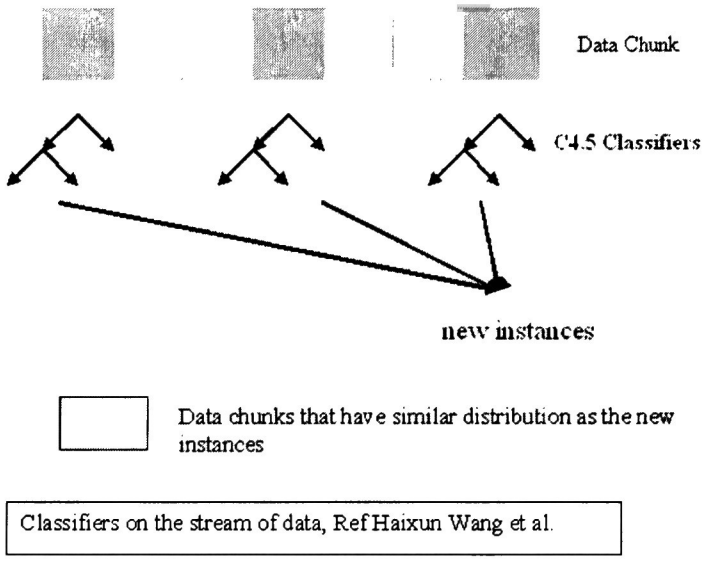

Figure 4: Stream Ensemble Algorithm (SEA) maintains a set of classifiers over the stream of data

The scalability of the window-based algorithms is not addressed [21]. Ensemble-based algorithms have been proposed for detecting concept drift in data streams. The "Streaming Ensemble Algorithm " -(SEA) [8] detects concept drift using the collection of C4.5 classifiers as specified in Figure 4. SEA reads data and creates a classifier which is added to the collection if it improves accuracy. One of the drawbacks of this system is if the maximum number of classifiers are reached a new classifier cannot be added. The maximum number

of classifiers is an input to the algorithm. Wang et al., [9] proposed detection of concept drift using the weighted ensemble method. In this approach classifiers are weighted by their computation of accuracy in classifying training data. According to Kolter et al., [21] dynamic weighted majority also falls in the category of ensemble algorithms wherein *experts* are added dynamically with respect to changes in performance while detecting concept drift.

Chapter 3

Proposed Approach

3.1 Approach

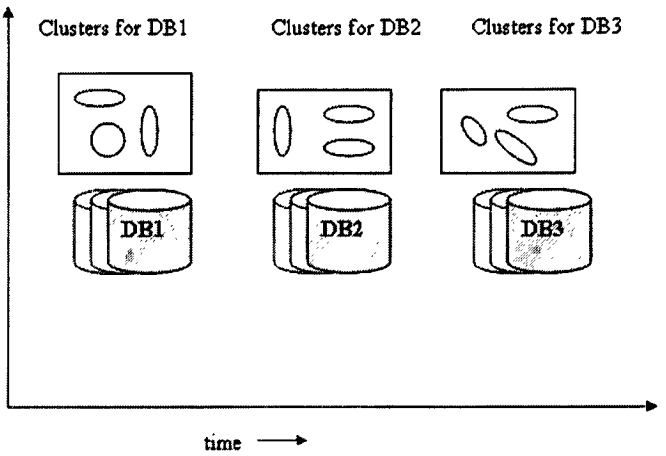

Figure 5: Proposed Approach Block Diagram

In Figure 5, the database clusters are shown at particular instances of time. The cluster shapes and sizes change with time. The motivation of our algorithm is based on measuring the distance between clusters (Gaussian mixture models) at different instances of time using statistical techniques like Kullback-Leibler divergence. Measuring the distance between the clusters detects concept drift. We have also addressed the scalability of the algorithm by condensing data points into summaries and applying clustering algorithm on the summaries.

The proposed algorithm for detecting the concept drift is as follows:

1. Read in the dataset

2. Calculate the summaries of the data

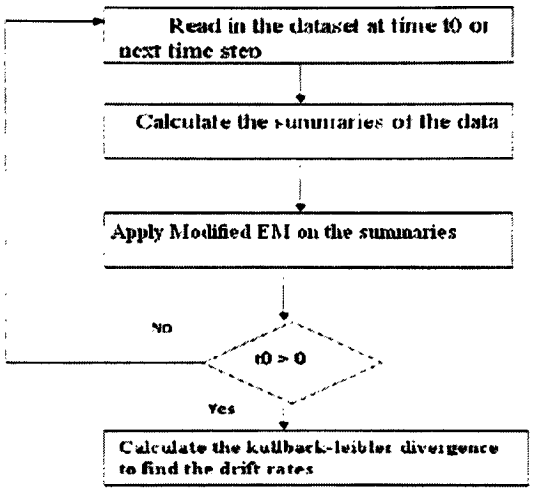

Figure 6: Algorithm for detecting Drift

3. Apply the modified Expectation-Maximization algorithm on these summaries to find parameters for the resultant Gaussian mixture models.

4. Save the Gaussian mixture model parameters.

5. Repeat step 1 to step 4 for the new incoming data.

6. Calculate the distance between the Gaussian mixtures using the Kullback-Leibler divergence to find the drift.

The algorithm operates in three phases. The primary task of Phase1 is data summarization, which consists of reading in data and calculating summaries

of the data. Phase2 is responsible for obtaining the clusters from the modified Expectation-Maximization algorithm applied to these summaries. Gaussian mixture models are obtained for data summaries as part of Phase2. The Kullback-Leibler distance is calculated between the Gaussian Mixture models as part of Phase3. The distance value obtained is directly proportional to drift.

3.2 Phase1: Data Summarization

With huge amounts of data being collected, the scalability of the algorithms needs to be addressed. The motivation of summarization is to not operate on all data points, but condense the data points into summaries. The algorithm operates on summaries of data points rather than operating on all data points, although there is a trade off between accuracy and performance improvement. The accuracy of the algorithm decreases but a considerable improvement in performance is achieved. Condensing a group of data points into summary can be achieved through techniques such as STING [22], BIRCH [23] etc. Statistical Information Grid Method (STING) [22] is a procedure which summarizes the data points by storing the mean, standard deviation, minimum and maximum values of attributes. Balanced Iterative Reducing and Clustering Hierarchies (BIRCH) is another data summarization method proposed by Zhang et al. [23]. In this algorithm the data summary is characterized by a *cluster feature* (CF). A cluster feature stores the number of data points as well as linear and square sum of the data points. It relies on the concept that not every data instance is important and treats closely related data as one instance. This method consists of building the CF tree, condensing the CF tree, global clustering and finally refining the clusters. We discuss these summarization procedures in detail.

3.2.1 STING

Most of the data summarization techniques require a single scan of data points. To make the algorithm scale to huge datasets, STING [22] relies on the concept that data points are summarized into fewer points. These summarized points although resulting in a loss of accuracy, can achieve big performance improvements for the algorithm operating on the data.

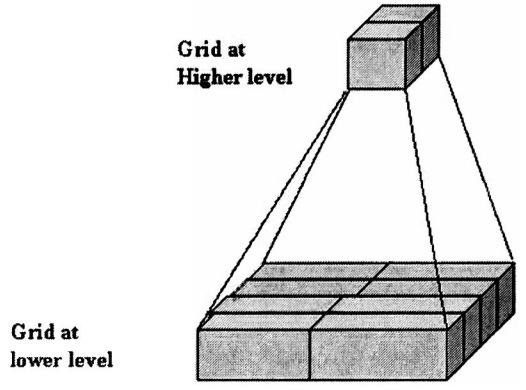

Figure 7: Sting-based Data Summarization. Cells in the lower level of the grid are summarized by cells at the higher level. In this case four cells at the lower level are summarized by one cell in the level above.

STING, designed for spatial data mining deals with extracting patterns in a spatial setting. Figure 7 indicates how STING divides the area into rectangular cells and forms a hierarchy of the cells. Each summarized data cell in STING has dependent and independent attribute parameters. The summary information for the cell is enumerated in Table 1. The hierarchy of cells is generated once the data is loaded into the database. The dataset is scanned once to calculate the cell values of the bottommost level. The cell values at the bottommost level is less than the number of data instances. In the hierarchy cells at higher levels summarize the cells in the level directly below. Parameters in the higher level

cells are generated by combining the statistical information of all the cells below. In Figure 7 the higher-level cell has summary information of all four cells below. Query answering in case of the STING algorithm depends on determining the correct layer from the grid and calculating the *confidence interval* (probability of a cell relevant to a query), then accordingly going down the hierarchy if the confidence interval is not met.

The STING+ [24] algorithm is based on STING but can handle incremental updates to the data, STING is a passive algorithm that cannot handle updates to the data, whereas STING+ handles incremental updates.

3.2.2 BIRCH

BIRCH is a data summarization procedure proposed by Zhang et al.,[23]. The BIRCH algorithm is an incremental algorithm that does not iterate over all data instances. It relies on the concept that not every data instance is important and treats closely related data as one instance.

The crux of the BIRCH algorithm is defining the Cluster Feature. The Cluster Feature is defined by:

- N number of data points
- $\vec{LS}$ linear sum of all data points = $\sum_{i=1}^{N} \vec{X_i}$
- SS - square sum of the N data points = $\sum_{i=1}^{N} \vec{X_i^2}$

This Cluster Feature summarizes a cluster of data points. The Cluster Feature is compact and has significant information about all points in the clusters.

A CF-tree is a height balanced tree similar to B-tree. Two parameters decide the branching factor (a) B = branching factor related to internal node (b) L = number of entries in a leaf node

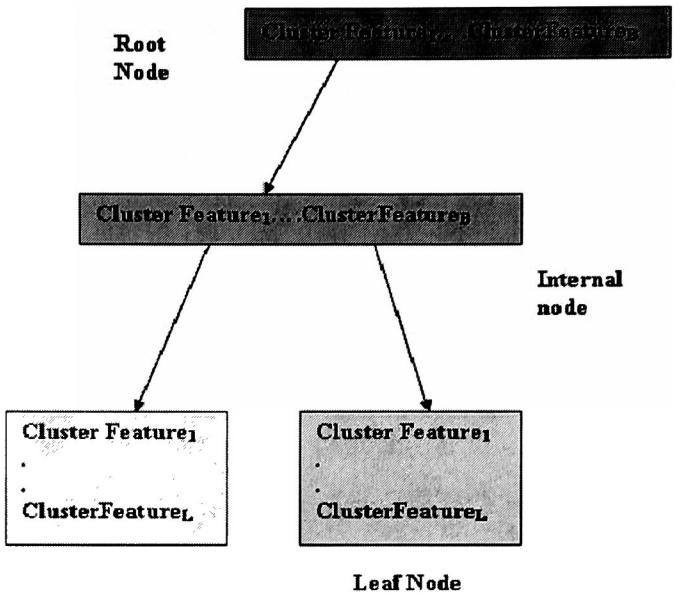

Figure 8: Birch-based Cluster Feature Construction. In the leaf nodes there are L entries and in non-leaf nodes there are B entries of cluster features respectively.

The BIRCH algorithm operates in four phases. Phase1 deals with the the CF tree construction, Phase2 of the algorithm deals with condensing the CF tree, global clustering and refining the clusters comprise Phase3 and Phase4 respectively. The mandatory phases of the algorithm are Phase1 and Phase2, whereas Phase3 and Phase4 are optional. The CF tree construction is equivalent to the B^+ tree [25] construction. When the CF node of the tree gets filled, the nodes splits and the tree size increase by one. After the first phase, a scenario could arise wherein objects in one cluster have split and kept in different nodes. The third phase, which deals with global clustering takes care of this scenario and represents the actual dataset.

3.2.3 Grid-Based Summarization

Machine-learning algorithms while treating concept drift assume independence among the attributes. This need not be true in a real-life scenario, especially in concept drift setting. Therefore the need for a data summarization procedure that keeps covariance information among the attributes is necessary. Adaptive grid-based summarization introduced by Huidong Jin et al., [26] is a simple grid-based summarization procedure that partitions the data and summarizes the data point within a subcluster.

Summaries are expressed by $\{n_m, v_m, \Gamma_m\}$ where:

- n_m is the number of points in the subcluster
- v_m is the mean of items in the subcluster
- Γ_m is the outer product of items in the subcluster.

The memory requirement for these summaries is high since calculating the covariance information involves storing the entire matrix of $\left(\Gamma_m - v_m v_m^T\right)$.

This summary information of Γ_m must be included in the E and M-steps of the EM algorithm. For the covariance matrix approximation in the M-step of the algorithm, we rewrite the equation to be [26]

$$Np_k^{j+1} \cdot \sum_k^{j+1} = \sum_{i=1}^{k} \xi_{ik}^j (x_i - \mu_k^{j+1})(x_i - \mu_k^{j+1})^T$$

$$= \sum_{m=1}^{M} \sum_{x_i \in them^{th} subcluster} \xi_{ik}^j (x_i - \mu_k^{j+1})(x_i - \mu_k^{j+1})^T$$

Data items in the m^{th} subcluster are assumed to have the same membership probability say $\alpha_{mk}^{(j)}$, the equation above can be written as [26]

$$Np_k^{j+1}.\sum_k^{j+1} = \sum_{i=1}^{M} \alpha_{mk}^{(j)} \sum_{x_i \in DS_m} (x_i - \mu_k^{j+1})(x_i - \mu_k^{j+1})^T$$

$$= \sum_{m=1}^{M} \alpha_{mk}^{(j)} n_m \left[\sum_{x_i \in DS_m} \frac{x_i x_i^T}{n_m} - (\sum_{x_i \in DS_m} \frac{x_i}{n_m})(\sum_{x_i \in DS_m} \frac{x_i}{n_m})^T \right] +$$
$$\sum_{m=1}^{M} \alpha_{mk}^{(j)} n_m \left[(\sum_{x_i \in DS_m} \frac{x_i}{n_m} - \mu_k^{j+1})(\sum_{x_i \in DS_m} \frac{x_i}{n_m} - \mu_k^{j+1})^T \right]$$
$$= \sum_{m=1}^{M} \alpha_{mk}^{(j)}.n_m \left[\left(\Gamma_m - v_m v_m^T \right) + \left(v_m - \mu_k^{j+1} \right) \left(v_m - \mu_k^{j+1} \right)^T \right]$$

If $\left(\Gamma_m - v_m v_m^T \right)$ can be approximated by $\delta_m \delta_m^T$ then δ_m can be treated in a similar way as $\left(v_m - \mu_k^{j+1} \right)$. The term in the gaussian density function which we desribe in the next section $\left(v_m - \mu_k^{j+1} \right) \sum_k^{-1} \left(v_m - \mu_k^{j+1} \right)^T$ can have an additional term of $-\frac{1}{2} \delta_m \sum_m^{-1} \delta_m^T$ inserted in equation 3.

We discuss in the EMADS section how δ_m is approximated. Compared to BIRCH which is based on the tree-based indexing of summarization, adaptive grid-based summarization is based on hash indexing. For grid-based summarization an additional member is added to the summary, $\{c_m, n_m, v_m, \Gamma_m\}$, the c_m is calculated from a hash function based on the attribute values. Tuples consisting of attribute values from all dimensions is provided as an input to the hash function which returns the cell number. This is maintained as a list of data summaries on which the modified EM algorithm operates.

Table 1 gives a brief description of the summary information stored by these summarization algorithms:

3.3 Phase2: Unsupervised Methods to Detect Drift

Unsupervised algorithms could be classified as (a)Generative and (b)Discriminative [27]. Generative models assume a parametric form of data. In generative models the parameters of the models need to be generated which

Table 1: Summary Information for Different Algorithms

Data Summarization	Summary Information
STING	n - number of data points in the cell m - mean of all values in the cell s - standard deviation of all the values in the cell min Minimum attribute value max - Maximum attribute value distr = Type of distribution
BIRCH	N number of data pts. LS - linear sum of all data pts. (sum of linear sum) SS - square sum of the N data pts (sum of squares)
Adaptive Grid-based	nm = Number of points mean = Mean of points Average of the outer products of the nth items

maximize the probability of generation of data given the model. Based on the similarity matrix defined over the input dataset, discriminative models cluster data items. Model-based clustering which has good theoretical foundations, falls into the category of generative class of clustering algorithms.

Advantages of model-based clustering algorithms are:

- Cluster interpretation in terms of probability
- Construction of online algorithms using learning techniques [28]
- Efficient coverage of data

Discriminative clustering deals with introducing a distance metric over the dataset. Graph theoretic clustering falls into the category of discriminative models. Unlike generative models of clustering, discriminative models do not assume any parametric form of data generation.

3.3.1 Expectation-Maximization

In model-based clustering, data is generated by a finite mixture of underlying probability distributions like normal distributions. The Gaussian mixture models for clustering have been studied extensively [29]. The parameters of the Gaussian distributions can be estimated using likelihood and maximized using the Expectation-Maximization (EM) [17] algorithm.

A mixture model in case of EM is characterized by [27]

$$p\,(x|\theta) = \sum_{i=1}^{K} p_k \psi(x_i|\theta_k) \tag{1}$$

where K is the number of mixtures and ψ is the component density function, p_k is the mixing proportion for the k^{th} cluster. The values of p_k range from 0 to 1 with the property of $\sum_{i=1}^{K} p_k = 1$. The $\psi(x_i/\theta_k)$ could be single variate as well as multivariate.

Single variate Gaussian mixture models are characterized by:

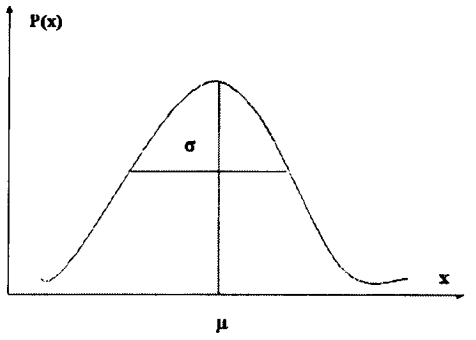

Figure 9: Single Variate Gaussian

$$p(x|\theta) = \frac{1}{\sigma\sqrt{2\pi}} * exp(-\frac{1}{2}(\frac{(x-\mu)}{\sigma})^2) \tag{2}$$

Multivariate Gaussian mixture model is expressed as [30]:

$$p(x|\theta) = (2\pi)^{-1/2} \left(\sum\right)^{-1/2} exp\left[\frac{-1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\right] \quad (3)$$

Where,

- μ = mean of the multivariate Gaussian
- $\sum$ = covariance of the multivariate Gaussian
- θ = represents parameters mean μ and covariance $\sum$

It is important to fit the model to the data set, i.e., estimating the mixture parameters. Likelihood [31] determines the accuracy of the mixture model with respect to the data and it is often represented as $P(\theta|x)$. Representing the likelihood in terms of a log functions is more beneficial as log function are monotonically increasing.

$$L(\theta) = log\left(P(\theta|x)\right) \quad (4)$$

$$L\left(\theta\right) = log\left[\prod_{i=1}^{N} p(x_i|\theta\right] = \sum_{i=1}^{N} logp(x_i|\theta) \quad (5)$$

We want to estimate the maximum likelihood i.e

$$L(\theta) = arg\ max L(\theta)$$

The EM algorithm in Data Mining is expressed by [12]:

1. Initialize:

   means μ_k and covariance $\sum_k$

2. E step: Compute membership probability:

$$\xi^{(j)} \leftarrow \frac{p_k \theta(x_i|\mu, \sum)}{\sum_{m=1}^{k} p_k \theta(x_i|\mu, \sum)}$$

3. M step: Update the mixture model parameters

$$p_k^{j+1} \leftarrow \frac{\sum_{i=1}^{K} \xi^{(j)}}{N}$$

$$\mu_k^{j+1} \leftarrow \frac{\sum_{i=1}^{N} \xi^{(j)}.x_i}{N.p_k}$$

$$\sum_{k}^{j+1} \leftarrow \frac{\sum_{i=1}^{N} \xi^{(j)} * (x_i - \mu_k^{j+1})(x_i - \mu_k^{j+1})^T}{N.p_k^{j+1}}$$

4. Repeat step 2 and 3 until

$$L(\theta^{i+1}) - L(\theta^i) \leq Threshold$$

3.3.2 Types of EM

The EM algorithm is not scalable since it operates on all data points. The scalability of EM can be achieved by techniques like random sampling, weighted sampling [32] and summary statistics [23]. We discuss briefly scalable Expectation-Maximization techniques.

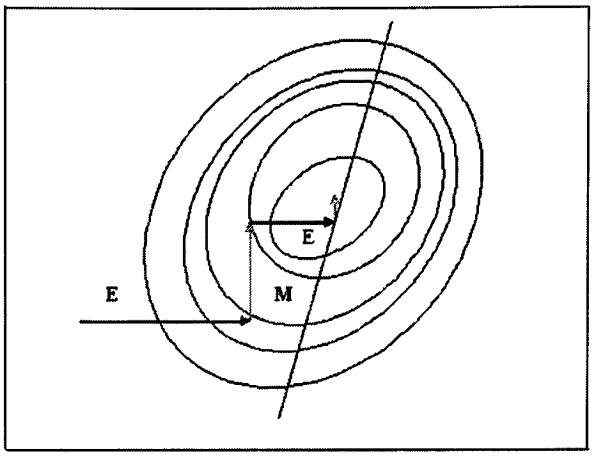

Figure 10: EM Algorithm from Trevor Hastie et al., [12].

Incremental EM

The incremental EM approach achieves scalability by performing partial E steps. It operates through static blocks of data, computing the E-step partially on blocks of data in a cyclic fashion.

The partition of data into blocks introduces three issues in the case of incremental Expectation-Maximization step:

- Number of cases to be placed in each block (selecting the correct block size)
- Handling convergence in case of blocks of data.
- Initial pass through the data

The incremental version of the EM algorithm operates by computing the probability of the i^{th} block of data. The data log-likelihood is set to the previous value or otherwise is computed separately. Finally the log-likelihood is maximized at the M-step. The incremental version of EM is dependent on extra

cost of storage of previous values of data log-likelihood. Efficient version of data log-likelihood could also be calculated by adding the difference between adding new and old components.

Lazy EM

Lazy EM [33] is based on the principle that all data points are not of equal importance. Lazy EM identifies the important data points and operates upon them. The significant cases are identified based on inferred completion specified by $p_i^n = p(X_i|y_i, \theta^n)$ where $X = \{X_1, X_2, ..., X_N\}$ are the N data cases characterized by parameters θ, X_i is a set of variables describing a case for parameter θ_i and $y \in Y \subseteq X$ is the incomplete data. If change in inferred completion is large between two successive iterations the instance is marked to be significant. As the first step of E all the log-likelihood is calculated. The significant cases are identified and E-step is run on these data points.

The Lazy EM divides the E-step into two parts where points are identified that need to be ignored in the lazy iterations. The E-step is responsible for computation of complete data log-likelihood. The M-step is responsible for maximizing the data log likelihood. Efficient updates of the data log-likelihood for the lazy step can also be calculated. Lazy EM operates on partial E steps but at the cost of identifying the significant cases and maintaining inferred completions.

ScalableEM

Scaling EM to large databases [13] is designed to operate on limited memory and in one scan of the database. It operates by initializing the model parameters. A memory resident buffer is filled by a sample of records from the database. The EM algorithm is applied on these records to update the resultant mixture model

parameters. Once the updated mixture model parameters are obtained, close data regions in the clusters are identified which can be summarized. These close data regions are compressed to sufficient statistics and all the records belonging to the close data regions are purged.

In scalableEM the sufficient statistics for a subcluster S is defined as $\{\theta, \Gamma, n\}$, where :

- $\theta = \sum_{x \in S} x$ (The vector sum of vectors in the subcluster)
- $\Gamma = \sum_{x \in S} xx^T$ (The matrix sum of outer product of the vectors)
- n = number of vectors in subcluster S with d attributes

The ScalableEM makes an assumption that there is no covariance information between attributes, hence it stores diagonal covariance matrices. Advantage of diagonal covariance matrices are easier implementation and computation for the scalableEM algorithm . The mean and covariance from the subcluster summary statistics is calculated by:

- $\mu^s = \frac{\theta}{n}$ (Mean over the subcluster S)
- $\sum^s = \frac{\Gamma - \frac{\theta\theta^T}{n}}{n}$ (Covariance Matrix over S)

The limited memory buffer records are representative of the entire dataset. Data summarization in case of scalable EM comprise of two phases:

- Primary data compression
- Secondary data compression

Primary data compression deals with those records which do not affect cluster memberships. Secondary data compression compresses the dense regions of data which are not near the Gaussian means i.e, the outliers.

In the data summarization phase three set of data records are maintained

- DS = Data records compressed in primary data compression
- CS = Data records compressed in secondary data compression
- RS = Data records retained in the memory buffer

Primary data compression purges the data records in the set DS. For data points which are not near the means of the Gaussians are compressed as part of secondary data compression and placed in the set of CS.

A sample of records is read and the memory buffer is filled, EM is applied on these records in the memory to update the mixture model parameters. As part of EM the data is identified which needs to be compressed and the data is compressed by sufficient statistics. The EM algorithm is applied again on the next sample until the stopping criteria.

EMADS

The data summaries in adaptive grid-based data summarization has the covariance information. Therefore the algorithm operating on these summaries needs to take into account the covariance information. Expectation-Maximization algorithm for data summaries(EMADS) proposed by Huidong Jin et al., [26] modifies the E and M-steps of the EM algorithm to have the covariance information.

The data summary consisting of $\left(\Gamma_m - v_m v_m^T\right)$ needs to be further simplified . Huidong Jin et al., [26] proposes that the positive semi-definite matrix of $\left(\Gamma_m - v_m v_m^T\right)$ can be approximated by the first covariance vector. The covariance vector is defined as the square root of the maximum eigen value say λ_d multiplied by its corresponding eigen value c_d.

Eigen values decomposition of a matrix A is given by $A.v = .v$ where A is a matrix and v is a vector. If this equation holds true, the λ is the eigen value and v is the eigen vector corresponding to the eigen value. To approximate the covariance matrix of $\left(\Gamma_m - v_m v_m^T\right)$ we need the maximum eigen value λ_1 and the corresponding eigen vector c_1. The product of square root of λ_1 and c_1 i.e $\sqrt{\lambda_1}c1$ approximates the positive semi-definitve matrix to δ_m. The simplified data summary is given by $\{n_m, v_m, \delta_m\}$

The density function is modified to have the covariance information.

$$\psi(subcluster|\theta) = (2\pi)^{-D/2}\left(\sum\right)^{-1/2} exp\left[\frac{-1}{2}(\delta_m^T \sum_k^{-1} \delta_m)(mean-\mu)^T \sum^{-1}(mean-\mu)\right] \tag{6}$$

The EMADS algorithm for the modified density function is given by [26]

1. **Initialization**:

   Set the current iteration $j = 0$. Initialize mixture model parameters $p_k^j\,(>0)$ means μ_k and covariance $\sum_k$ such that $\sum_{k=1}^K p_k^j = 1$ and $\sum_k^j$ is a positive definitive matrix. K is the number of input clusters given in advance.

2. **E step**: Compute membership probability:

$$r_{mk}^{(j)} = \frac{p_k^j \psi\left(x_i|\mu, \sum\right)}{\sum_{m=1}^k p_k \psi(x_i|\mu, \sum)}$$

3. **M step**: Update the mixture model parameters given $r_{mk}^{(j)}$

$$p_k^{j+1} = 1/N * \sum_{i=1}^{K} n_m . r_{mk}^j$$

$$\mu_k^{j+1} = \frac{\sum_{i=1}^{M} n_m r_{mk}^j . v_m^j}{N . p_k^{j+1}}$$

$$\sum_k^{j+1} = \frac{\sum_{i=1}^{N} n_m r_{mk}^j * (\delta_m * \delta_m^T + (x_i - \mu)(x_i - \mu)^T)}{N . p_k^{j+1}}$$

4. Repeat step 2 and 3 until

$$\left| L(\psi^{j+1}) - L(\psi^i) \right| \geq \left| L.\left(\psi^j\right) \right|$$

Increment j to $j+1$

Phase2 of the project deals with Expectation-Maximization algorithm applied to the summaries obtained from Phase1. EMADS [26] modifies the E and M-steps of the algorithm to include the covariance information. As part of Phase2 the EMADS algorithm is applied to the summaries obtained from Phase1. δ_m obtained from Phase1 keeps the covariance information into account and in EMADS, the E and M-steps are modified to have these covariance information. The output of this phase is Gaussian mixtures. The scalability is achieved since the E and M steps of an algorithm are not applied to all instances but only on the summaries of points obtained through Phase1.

The cluster dissimilarity obtained from EM at different instances of time can be calculated by different distance metrics. The Kullback-Leibler(KL) divergence [34] is a means to detect the difference between probability distribution functions.

Given two single variate Gaussian distribution the KL divergence is given

by the formula:

$$kl(p||q) = \sum p(x) log \frac{p(x)}{q(x)} \tag{7}$$

Properties of KL distance are:

- $KL(p||q) \neq KL(q||p)$
- $KL(p||q) = 0$ only if $p = q$

KL distance is non-negative.

In case of multivariate Gaussian distribution of N_0 and N_1 the $KL\left(N_0||N_1\right)$ is given by the formula [30]

$\mathrm{KL}(N_0||N_1) =$

$1/2 * ln\ det\left(\sum_1 / \sum_2\right)$

$+ 1/2 * tr\left(\sum_1^1 \sum_0\right) +$

$1/2 * \left(\mu_1 - \mu_0\right)^T \sum_1^{-1} \left(\mu_1 - \mu_0\right) - N/2$

- μ = mean of the multivariate Gaussian
- $\sum$ = covariance of the multivariate Gaussian
- tr = trace function

The multivariate KL distance is used to calculate the distance between the clusters obtained at various instances of time.

Phase3 of the project consists of finding the difference between the Gaussian mixtures obtained at various instances of time. The KL distances between the clusters at various instances of time are calculated. If the distance between the clusters are high, a drift has occurred. The user can also specify the threshold for the KL distance that indicates drift.

3.3.3 Algorithm Complexity

The time and space complexity of the algorithm is discussed in this section. First we analyze the time complexity of this algorithm. Before we discuss the time complexity of individual phases, a brief introduction to the notations:

- S = Number of summaries
- C = Number of clusters
- A = Number of dimensions/Attributes
- N = Number of data points
- I = Number of iterations

The algorithm operates in three phases, Phase1 comprising data summarization, Phase2 applying the EM algorithm to get the Gaussian mixtures and Phase3 calculating the distance between the clusters. Since all points are iterated in Phase1 to get the summaries the complexity of this phase is O(NA). Phase2 deals with the EM algorithm applied to these summaries, the EM algorithm consists of the E and M steps. In the E step we compute the membership probabilities. Since there are S summaries and C the number of clusters, $S * K$ membership probabilities needs to be calculated. The E-step involves $O(A^2)$ calculation. So for $S * K$ membership probabilities, the E step involves $O(S * C * A^2)$. Similarly for the M step there are $O(S * C * A^2)$. The E and M steps in EM continue for the number of iterations I. So combined complexity for that E and M steps are $O(S*C*A^2*I)$. Finally Phase3 iterates over all the clusters to get the KL distance measures so complexity here is $O(C)$ So in all the total time complexity of the algorithm is $O(NA)+O(S*C*A^2*I)+O(C)$.

In Phase1 of summarization there is a additional space complexity of $O(S * A^2)$. Phase2 of clustering over the summaries introduces an additional space complexity of storing $S(2A + 1)$ floating point numbers. $S * C$ floating point numbers for membership and $C(A^2 + A + 1)$ for mixture model [26]. So the combined space complexity of Phase2 is $2SA + SC + SA^2 + CA + C + S$. Phase3 involves storing the Gaussian mixture model storage for calculating the KL distance which introduces additional space complexity of $O(C*A)$. The time complexity of BIRCH is $O(NA)$. BIRCH can only find clusters of spherical size since it used the concept of radius. Whereas in out scenario the time complexity of our proposed algorithm is directly proportional to the number of summaries not to the number of points.

The BIRCH algorithm operates by loading the data into memory and constructing the cluster feature (CF) tree. If M is the available memory and P is the page size, maximal size of the tree is M/P.The cost of constructing the CF tree is given by $O(D * N * B(1 + log_B M/P)$ [23], where D is the number of dimensions, B is the branching factor of the tree. Hence BIRCH scales linearly to data. BIRCH algorithm can only performs well when cluster shapes are spherical [35]. There is no such performance criterion for our proposed approach of detecting drift.

Chapter 4

Experiments

There aren't any publicly available datasets that have significant drifts in them [15, 36]. The standard benchmark for detecting drift are synthetic datasets like *stagger concepts* and *sea concepts* . We evaluate concept drift on the forest cover dataset from UCI KDD [37].

Stagger concept as specified in [21] is a standard benchmark of testing concept drift. In this case we did not use the label value of the dataset. The stagger data consists of shape, size and color attributes. The values for these three attributes are as follows:

- $shape \in \{triangle, circle, rectangle\}$
- $size \in \{small, medium, large\}$
- $color \in \{green, blue, red\}$

For the first batch, the target concept is $color = red \wedge size = small$. For the next batch, the target concept is $color = green \vee shape =$ circle. Finally, for the last batch, the target concept is $size = medium \vee size = large$. The generated data is divided into two sets of files. At time t_0, the first batch of data consisting of 80 samples is taken as input. Correspondingly at time t_1, the second batch is taken as input for the algorithm.

All experiments were conducted on a Intel Pentium 2.80GHz machine. The algorithm was developed in Java and for visualization of resultant Gaussian mixtures parameters, MATLAB was used. The Gaussian mixtures are obtained after 10 runs on the algorithm for each batch.

Figure 11 and 12 represent the stagger concepts at time t_0 and t_1 respectively. The orientation and shape of the clusters have changed between t_0 and time t_1. The KL distance values between the three clusters is enumerated in table 6.

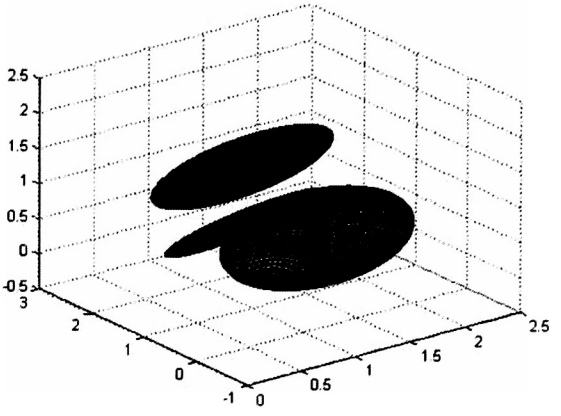

Figure 11: Stagger at time=t0

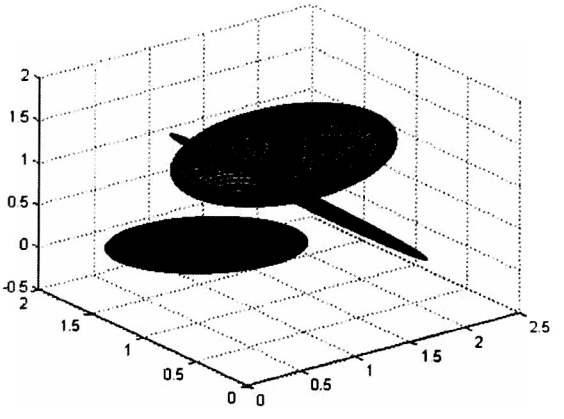

Figure 12: Stagger at time=t1

4.0.4 Sea Concepts

To address the scalability of the algorithm, the proposed algorithm was tested on the Sea Concepts dataset. As discussed in Kolter et al.,[21] Sea

Concepts consists of three attributes, $x_i \in R$ such that $0 < xi < 10$. The target concept is $x1 + x2 <= b$, where $b \in \{7; 8; 9; 9.5\}$. For the first batch the target concept is $b = 8$, for the next batch the target concept is $b = 9$ and for the final batch $b = 7$. The presentation of training examples lasts for 50,000 time steps.

Figure 13, 14, 15, 16 represent the Sea Concepts at time t_0 and t_1, t_2 and t_3 respectively.

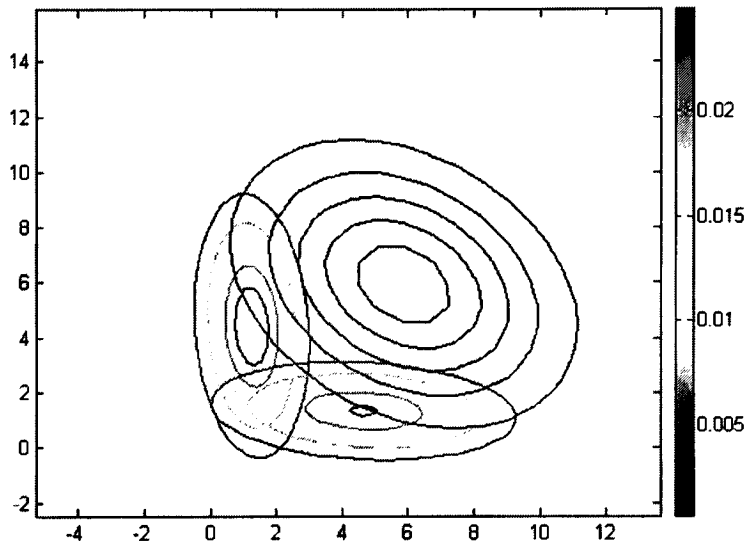

Figure 13: Sea Concepts at time=t0

As depicted in 14, 15, 16 the shape and the orientation of the Gaussian mixtures have changed. The KL distance measures are calculated and enumerated in table 6. We formulated the results and observed that the distance values of less than 10 result in less drift. KL distance values greater than 10 indicate higher drift. The specific KL distance value of 10 was experimentally observed.

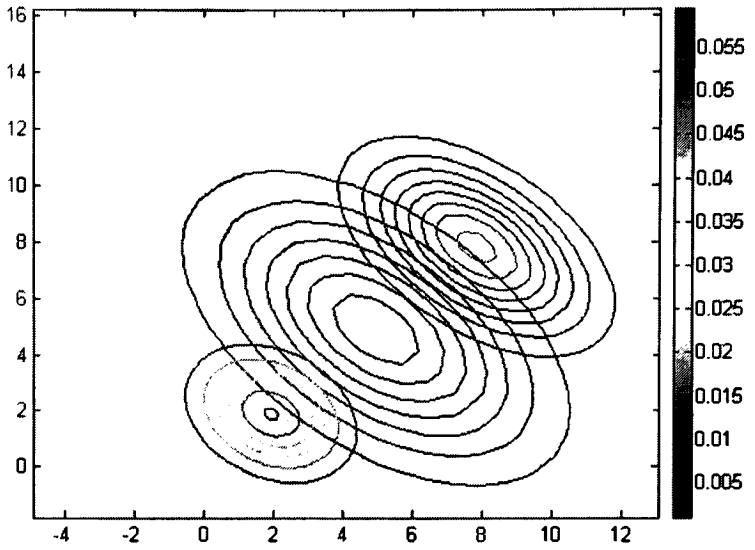

Figure 14: Sea Concepts at time=t1

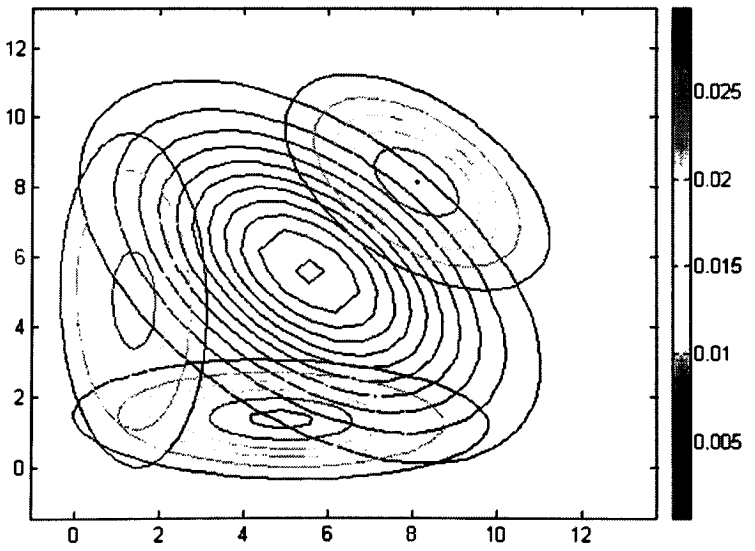

Figure 15: Sea Concept at time=t2

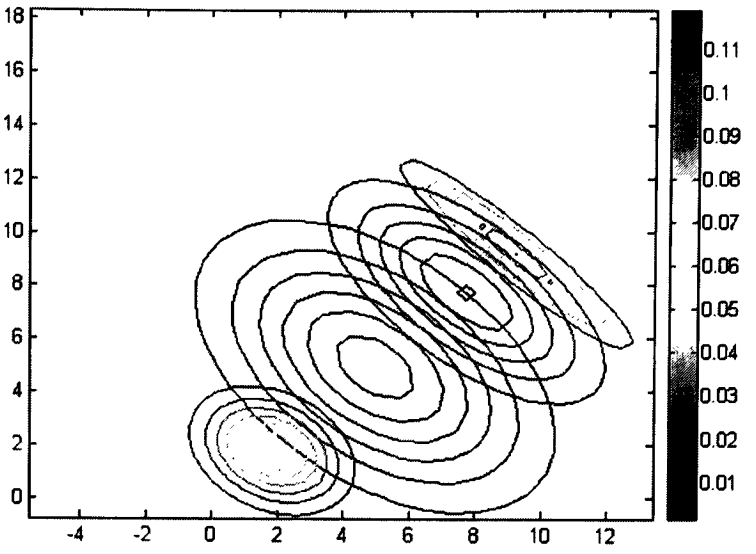

Figure 16: Sea Concept at time=t3

Scalability

We tested our algorithm against the EM algorithm from WEKA [38]. The EM algorithm was applied to the SEA dataset, clusters were obtained and the KL distance was calculated for the clusters. We applied the proposed algorithm on these datasets and measured the time taken, Table 2 enumerates the results obtained.

Table 2: Time taken by the EM and Proposed Algorithm on the Sea Dataset

Dataset at Instance	EM (msecs)	Proposed Algorithm (msecs)
0	92641	60775
1	92375	63922
2	109762	75567

The proposed algorithm scales as fast as the EM algorithm this can be observed from Figure 17.

We calculated the percentage accuracy of the proposed algorithm against the EM algorithm [39]. We formulate our results below

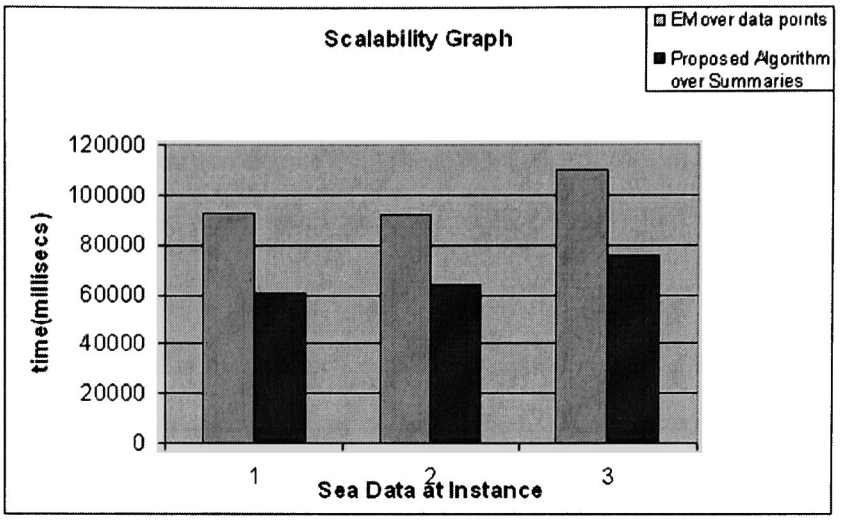

Figure 17: Scalability Graph

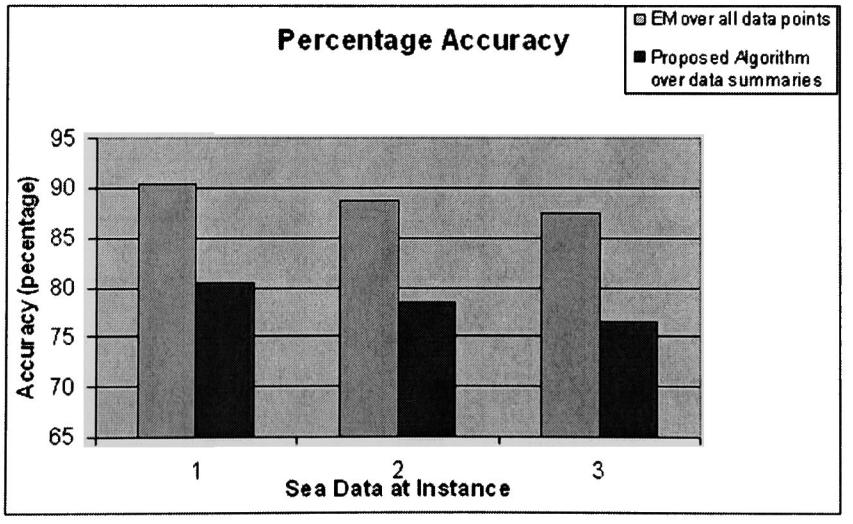

Figure 18: Percentage Accuracy Graph

Table 3: Accuracy of the EM and the proposed algorithm on the sea dataset

Dataset at Instance	EM (Percent)	Proposed Algorithm (Percent)
0	90.452	80.3451
1	88.6175	78.432
2	87.4327	76.557

From figure 18 we can depict that there is a loss of accuracy of approximately 10% of the proposed algorithm compared to EM algorithm [39].

Input parameter

We tested the EM algorithm for different input values of clusters. The EM was initialized by running the K-means algorithm. The timing for different value of clusters were computed and enumerated in the table. Further we varied the input number of clusters for our proposed algorithm. As the number of clusters increase the time taken by both EM and our proposed algorithm increase too. Table 4 and Figure 19 shows the results obtained from varying the number of clusters parameter on the EM algorithm whereas Table 5 and Figure 20 shows the results obtained on the proposed algorithm.

Table 4: EM algorithm performance on the number of clusters parameter

Number of cluster	Instance0	Instance1	Instance2
1	2000	3813	5312
2	11062	27203	10687
3	81250	95609	122187
4	103547	132375	352125
5	138828	349359	423938

Table 5: Proposed algorithm performance on the number of clusters parameter

Number of cluster	Instance0	Instance1	Instance2
1	20125	19969	19844
2	39063	38484	39750
3	57219	58031	57390
4	74672	76453	76172
5	94469	95329	93937

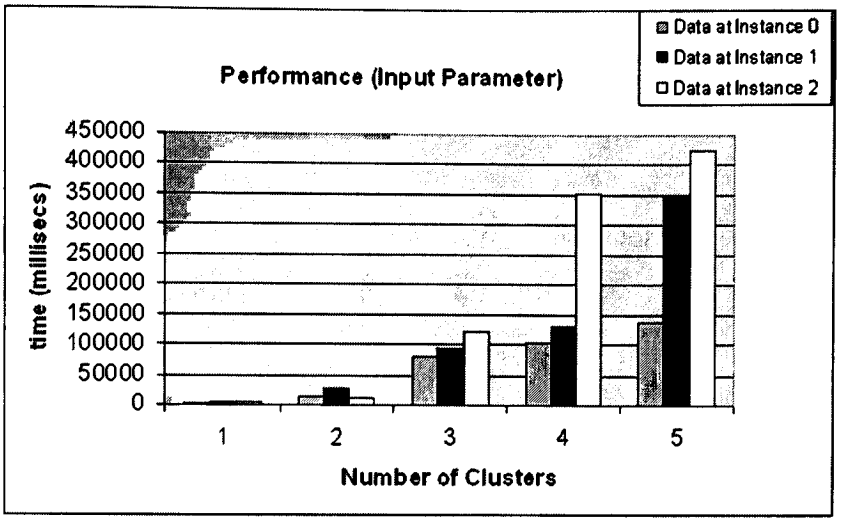

Figure 19: Performance Based on number of clusters

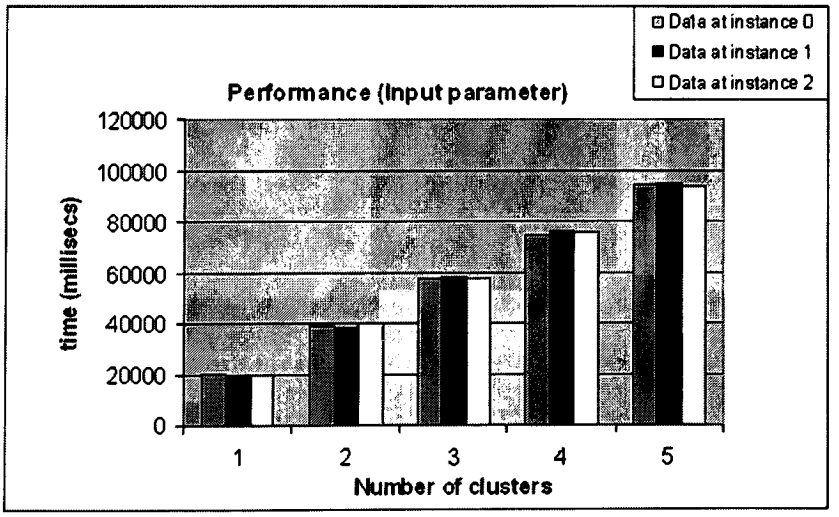

Figure 20: Proposed Algorithm Performance based on number of clusters

Discussion

From the results the proposed concept drift algorithm runs almost as fast the plain EM algorithm from WEKA with loss of accuracy of approximately 10%. The accuracy loss was the result of the concept drift algorithm operating on the summaries rather than the actual points. The advantage of working over summaries resulted in improvement of performance. The approximation of summaries reflecting the actual points and further approximating the covariance matrix to the first covariance vector could be the result of this. Further the grid-based summarization procedure using the hash function to map similar attributes values to the same cell might not be very accuracte. The need for effective hash indexing summarization procedure needs to be developed. Since the EM algorithm from WEKA does not make such assumption this resulted in higher accuracy as compared to out algorithm. The initialization of the input parameter to the algorithm had similar effects for both the proposed algorithm and the EM algorithm. As the number of clusters increased the time taken by the algorithm increased.

4.0.5 ForestCover

Obtained from the UCI KDD archive [37], the forest cover dataset consists of five attributes. The values are scaled between $[0, 1]$. The dataset is divided into two set of files. At instance t_0 the first instance is presented to the system and the clusters are obtained. At instance t_1 the next batch of data is provided to the system.

Table 6: KL distance for ForestCover Data

Data Set	KL distance
Stagger Concepts	
Cluster1	25.2132
Cluster2	20.397
Cluster3	19.8125
Sea Concepts	
Cluster1	4.6673
Cluster2	6.4132
Cluster3	11.9602
Cluster4	16.4321
ForestCover	
Cluster1	4.2132
Cluster2	4.397
Cluster3	9.8743
Cluster4	5.4321

Chapter 5

Conclusions and Future Work

A new algorithm was proposed in this thesis for detecting unsupervised concept drift. The proposed algorithm unlike other algorithms does not assume statistical independence among the attributes and operates in an supervised setting. Covariance information of the attributes are stored from summaries of the data points. The covariance matrix is further simplified by approximating it by the maximum eigen value and its corresponding eigen vector termed as the first covariance vector. By storing the summaries of the data points the scalability of the algorithm is addressed. The results confirm that by storing the summaries, the proposed algorithm is more efficient as compared to plain EM algorithm. It also confirms that with a loss of accuracy of around 10% the performance enhancement is achieved. Furthermore this technique uses the standard techniques of Expectation-Maximization to detect clusters (Gaussian mixtures) and Kullback-Leibler divergence which have strong theoretical foundations. Experiments were conducted for synthetic as well as real data sets.

Market basket analysis deals with finding the buying habits of customers. The proposed algorithm can find application in market basket analysis. It can be used to find out when the *concept* of buying habits of customers change. The algorithmic framework can be extended to find the drift with respect to other parameters of the dataset as well, currently the framework uses only time as the parameter. The proposed algorithm can also be used to determine how the concepts of web browsing changes for users over time.

Merging concepts [18] is a complex form of drift occurring in sound processing. In merging concept two concepts merge into one single concept. The

algorithm can be extended to support merging concepts.

List of References

[1] J. Schlimmer and R. G. Jr., "Beyond incremental processing: Tracking concept drift," In Proceedings of the Fiffth National Conference on Artificial Intelligence, pages 502-507. Morgan Kaufman, Philadelphia, PA, 1986.

[2] D. H. Widyantoro, "Relevant data expansion for learning concept drift from sparsely labeled data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 401–412, 2005. Fellow-John Yen.

[3] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 128–137, ACM Press, 2004.

[4] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[5] M. A. Maloof and R. S. Michalski, "Selecting examples for partial memory learning," *Machine Learning*, vol. 41, no. 1, pp. 27–52, 2000.

[6] M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift.," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, 2004.

[7] M. Maloof and R. Michalski, "AQ-PM: A system for partial memory learning," in *Proceedings of the Eighth Workshop on Intelligent Information Systems*, (Warsaw, Poland), pp. 70–79, Polish Academy of Sciences, 1999.

[8] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 377–382, ACM Press, 2001.

[9] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers.," in *KDD*, pp. 226–235, 2003.

[10] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proceedings of ICML-00, 17th International Conference on Machine Learning* (P. Langley, ed.), (Stanford, US), pp. 487–494, Morgan Kaufmann Publishers, San Francisco, US, 2000.

[11] M. Kubat, J. Gama, and P. E. Utgoff, "Incremental learning and concept drift: Editor's introduction.," *Intell. Data Anal.*, vol. 8, no. 3, pp. 211–212, 2004.

[12] R. T. Trevor Hastie and J. Friedman, eds., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY, USA: Springer-Verlag, 2001.

[13] P. Bradley, U. Fayyad, and C. Reina, "Scaling em (expectationmaximization) clustering to large databases," Technical Report MSR-TR-98-35, Micrososft Research, Washington, August 1998.

[14] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering.," *Knowl.-Based Syst.*, vol. 18, no. 4-5, pp. 187–195, 2005.

[15] A. Tsymbal, "The problem of concept drift: Definitions and related work," Technical Report TCD-CS-2004-15, Department of Computer Science, The University of Dublin, Dublin, Ireland, April 2004.

[16] R. Klinkenberg and I. Renz, "Adaptive information filtering: Learning in the presence of concept drifts," In Proceedings of In Learning for Text Categorization. Menlo Park, CA, pp. 33-40., 1998.

[17] A. P. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm (with discussion)," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38,, 1977.

[18] M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift.," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, 2004.

[19] M. Lazarescu, *Incremental learning for querying multimodal symbolic data.* PhD thesis, 2000.

[20] M. Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching.," *Artif. Intell. Rev.*, vol. 11, no. 1-5, pp. 133–155, 1997.

[21] J. Kolter and M. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," Technical Report CSTR-20030610-3, Department of Computer Science, Georgetown University, Washington, DC, June 2003.

[22] W. Wang, J. Yang, and R. R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Twenty-Third International Conference on Very Large Data Bases* (M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, eds.), (Athens, Greece), pp. 186–195, Morgan Kaufmann, 1997.

[23] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: A new data clustering algorithm and its applications," *Data Min. Knowl. Discov.*, vol. 1, no. 2, pp. 141–182, 1997.

[24] W. Wang, J. Yang, and R. Muntz, "STING+: An approach to active spatial data mining," in *Fifteenth International Conference on Data Engineering*, (Sydney, Australia), pp. 116–125, IEEE Computer Society, 1999.

[25] D. Comer, "Ubiquitous b-tree," *ACM Comput. Surv.*, vol. 11, no. 2, pp. 121–137, 1979.

[26] "Scalable model-based clustering for large databases based on data summarization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1710–1719, 2005. Member-Huidong Jin and Member-Man-Leung Wong and Senior Member-K. -S. Leung.

[27] H. M. D. Hand and P. Smyth, eds., *Principles of Data Mining.* MIT Press, 2001.

[28] J. Sinkkonen and S. Kaski, "Clustering based on conditional distributions in an auxiliary space.," *Neural Computation*, vol. 14, no. 1, pp. 217–239, 2002.

[29] C. Fraley, "Algorithms for model-based Gaussian hierarchical clustering," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 270–281, 1999.

[30] Wikipedia, "Multivariate normal distribution — wikipedia, the free encyclopedia," 2006.

[31] T. K. G McLachlan, ed., *The EM Algorithm and Extensions.* John Wiley Sons, Inc, 1997.

[32] C. R. Palmer and C. Faloutsos, "Density biased sampling: an improved method for data mining and clustering," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 82–92, ACM Press, 2000.

[33] B. Thiesson, C. Meek, and D. Heckerman, "Accelerating em for large databases," *Mach. Learn.*, vol. 45, no. 3, pp. 279–299, 2001.

[34] S. Kullback and R. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, 1951.

[35] B. L. Milenova and M. M. Campos, "O-cluster: Scalable clustering of large high dimensional data sets.," in *ICDM*, pp. 290–297, 2002.

[36] T. Fawcett, ""in vivo" spam filtering: a challenge problem for kdd," *SIGKDD Explor. Newsl.*, vol. 5, no. 2, pp. 140–148, 2003.

[37] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth, "The UCI KDD archive of large data sets for data mining research and experimentation," *SIGKDD Explorations*, vol. 2, no. 2, pp. 81–85, 2000.

[38] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.* 2005.

[39] I. H. Witten and E. Frank, *Data mining: practical machine learning tools and techniques with Java implementations.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.