2. LITERATURE REVIEW

Data clustering is a thrust area of research for statisticians as well as data mining researchers which resulted in the development of a vast variety of successful clustering algorithms. These algorithms can be categorized based on different issues like partitional, hierarchical, density-based, model-based, and grid-based. In addition to this categorization, a clustering algorithm can either be incremental or non-incremental based on its ability to handle dynamically arriving data. Non-incremental clustering methods generally process all the data records at a time. These algorithms require the entire datasets to be loaded into memory as they repeat the process of clustering from scratch whenever they have to reflect the recent enhancements to the database. Therefore they have a high requirement of memory space and redundancy in processing. For incremental clustering algorithms it is not necessary to store the entire dataset in the memory. So, the space requirements of incremental algorithms are very small as they make use of clustering solutions of the original database and incrementally update them as per the new chunks of data. Incremental clustering considers input data records one at a time and assigns them to the existing clusters. Here, a new input data record is assigned to a cluster without affecting the existing clusters significantly. [Hsu and Huang 2008].

Incremental clustering has attracted the attention of the research community with Hartigan’s Leader clustering algorithm [Hartigan 1975] which uses a threshold to determine if an instance can be placed in an existing cluster or it should form a
new cluster by itself. The LEADER algorithm partitions a data set into groups by virtue of a radius distance (T). A leader object is associated with each group and all other objects in the group lie within the distance T from that object. The first data point is selected and assigned as the first leader object, A. Subsequently, the remaining samples are examined and those that are within the distance T are assigned to group one. The first data sample examined that falls outside the radius T is assigned as the next leader object, B. This procedure is iterated to identify cluster centre C as well as the remaining centers. It requires processing a data record only once. Though the LEADER algorithm was originally devised to handle static databases only, its methodology naturally accepts dynamically arriving data records. Hence was considered as a forerunner of incremental clustering algorithms.

Similarly there are some more algorithms which are designed originally for static databases that could handle incrementally growing databases. For example, COBWEB and CLASSIT algorithms are designed for categorical and numerical datasets respectively.

COBWEB [Fisher 1987] is an unsupervised conceptual clustering algorithm that produces a hierarchy of classes. It organizes observations into a classification tree. Each node in a classification tree represents a class or concept and is labeled by a probabilistic concept that summarizes the attribute-value distributions of objects classified under the node. This classification tree can be used to predict missing attributes or the class of a new object. Its incremental nature allows clustering of new data to be made without having to repeat the clustering already made. It has been successfully used in engineering applications [Fisher et al. 1993]. COBWEB uses a
heuristic measure called *category utility* to guide search. This metric was originally developed as a means of predicting the basic level in human classification hierarchies [Gluck and Corter 1985].

Category utility can be viewed as a function that rewards traditional virtues held in clustering generally similarity of objects within the same class and dissimilarity of objects in different classes. In particular, category utility is a tradeoff between intra-class similarity and inter-class dissimilarity of objects, where objects are described in terms of (nominal) attribute-value pairs. It is defined as the *increase* in the expected number of attribute values that can be correctly guessed \( P(C_K) \sum_i \sum_j P(A_i = V_{ij} | C_K)^2 \) given a partition \( \{C_1 ...... C_n\} \) over the expected number of correct guesses with no such knowledge \( \sum_i \sum_j P(A_i = V_{ij})^2 \).

From Eq. 2.1, it can be seen that more formally, \( CU ( \{C_1, C_2, ..., C_n\} ) \) equals

\[
\frac{\sum_{k=1}^{n} P(C_k) \left[ \sum_i \sum_j P(A_i = V_{ij} | C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2 \right]}{n}
\]  

(2.1)

The denominator, \( n \), is the number of categories in a partition. Averaging over categories allows comparison of different size partitions. If an attribute value, \( A_i = V_{ij} \), is independent of class membership, then \( P(A_i = V_{ij} | C_K) = P(A_i = V_{ij}) \) and \( P(A_i = V_{ij} | C_K)^2 - P(A_i = V_{ij})^2 = 0 \). If this is true for all the attribute's values then the attribute is effectively *irrelevant* for any expression of category makeup.
CLASSIT [Gennary et al. 1989] is an alternative version of COBWEB. It handles continuous or real valued data and organizes them into a hierarchy of concepts. The nodes in the hierarchy are partially ordered according to their generality, with concepts lower in the hierarchy being more specific than their ancestors. Thus, the root node summarizes all instances that have been ordered, terminal nodes correspond to single instances and intermediate nodes summarize clusters of observations. It assumes that the attribute values of the data records belonging to a cluster are normally distributed. As a result, its application is limited.

Another such algorithm was developed by Fazil Can to cluster documents [Can 1993]. Can introduced an incremental clustering algorithm C²ICM (Cover-Coefficient-based Incremental Clustering Methodology). The CC (cover-coefficient) concept is used to determine the number of clusters and cluster seeds, then to assign non-seed documents to the clusters initiated by the seed documents to generate a partitioned clustering structure. Since periodic updating of clusters is required due to dynamic nature of document databases these changes should be reflected in the partition generated without significantly affecting the current clusters.

Though efforts were made to test the ability of a clustering algorithm to handle dynamically growing databases prior to 1997 Charikar et al. defined the incremental clustering problem and proposed a incremental clustering model which preserves all the desirable properties of HAC (hierarchical agglomerative clustering) while providing a extension to the dynamic case. [Charikar et al.1997]. This paper analyzes several natural greedy algorithms and proved that they perform rather poorly in the dynamic setting. New deterministic and randomized incremental
clustering algorithms like doubling algorithm and clique algorithm were introduced to efficiently maintain clusters of small diameter as new points are inserted with good performance.

2.1 Incremental clustering algorithms for mining dynamic and large datasets

A major advantage of the incremental clustering algorithms is their limited space requirement since the entire dataset is not necessary to store in the memory. Therefore, these algorithms are well suited for a dynamic environment and for very large datasets [Hsu and Huang 2008].

BIRCH [Zhang T et al. 1996] is especially suitable for large number of data items. BIRCH stands for Balanced Iterative Reducing and Clustering using Hierarchies. It uses cluster representatives instead of actual data points. BIRCH uses a hierarchical data structure called Cluster Feature tree (CF tree) for partitioning the incoming data points in an incrementally and dynamically. It minimizes memory usage and scans data only once from the disk. BIRCH is the first clustering algorithm in the database area to handle noise (data points that are not part of the underlying pattern) effectively.

BIRCH employs the following four phases during the clustering process.

Phase 1: Linearly scans all data points and inserts them into the CF tree such that it reflects the clustering information of the dataset. This phase creates an in-memory summary of data.
Phase 2: This phase is optional. It scans the leaf entries in the initial CF tree to rebuild a smaller CF tree, while removing more outliers and merging sub clusters.

Phase 3: A global or semi-global algorithm is employed to cluster all leaf entries and the information in the CF vectors is sufficient, for calculating distance and quality metrics.

Phase 4: This phase uses the centroids of the clusters produced by the above phase as seeds, and redistributes the data points to its closest seed to obtain a set of new clusters. This allows points belonging to a cluster to migrate and ensures that all copies of a given data point go to the same cluster.

Incremental DBSCAN was presented by Ester et al., which is suitable for mining in a data warehousing environment where the databases have frequent updates. After insertions and deletions are made to the database, the clustering discovered by DBSCAN has to be updated. Due to the density-based nature of DBSCAN, the insertion or deletion of an object affects only the small neighborhood of that object. It identifies which part of the existing clustering space is affected by the arrival of new chunk of data and accordingly makes incremental updates to the existing clustering solution. It is an efficient, incremental clustering algorithm for metric databases (that is, databases with a distance function for pairs of objects [Ester et al.1998].

2.2 Incremental clustering of numerical data

Recently, the incremental clustering of numerical data has received a great deal of attention among data mining researchers. BIRCH and Incremental DBSCAN
algorithms work for numerical data. As they are already discussed above, a brief review of some other recent researches for incremental clustering is presented here.

The GRIN algorithm, [Chen et al. 2002] is an incremental hierarchical clustering algorithm for numerical data sets based on gravity theory in physics. It delivers good clustering quality with $O(n)$ time complexity as it is immune from the order of input data and the optimal parameter settings are not sensitive to the distribution of the data set. The incremental nature of the GRIN algorithm implies that it is particularly suitable for handling the already huge and still growing databases in modern environments. Its hierarchical nature provides a highly desirable feature for many applications in biological, social, and behavior studies.

Two main challenges in the design of incremental clustering algorithms are addressed by Chen et al. The first challenge is how to reduce information loss due to the data abstraction (or summarization) operations. The second challenge is that the clustering result should not be sensitive to the order of input data. In this paper, centroid, radius, and the number of data instances it contains are used as the features of the abstraction model to represent a sub cluster due to the fact that any cluster, regardless of its shape and whether it has a uniform density, can be decomposed into a number of spherical sub clusters, each with virtually uniform density. The summarization process is applied to a dendrogram generated by a hierarchical agglomerative clustering algorithm. A cluster is said to be spherical, if the cluster satisfies either one of the following two conditions:
1) A cluster containing less than Min data instances is considered as a spherical cluster by default, because such a cluster does not contain sufficient number samples for any meaningful statistical test to be conducted.

2) For a cluster containing Min or more data instances, a χ² goodness of fit test [Hogg and Tanis 2001] is conducted to check whether the data instances in a cluster are uniformly distributed or not. The test is applied to each node in the dendrogram in order of their formation. A node is tested only after all of its descendents have been tested and the node that has passed the test will be marked as a spherical cluster.

Since the design of GRIN uses the statistical test-based summarization approach for reducing information loss & distortion and to control sub clusters formation it is able to achieve near linear scalability and is not sensitive to input ordering [Chen et al. 2005].

Serban and Campan have presented an incremental algorithm known as Core Based Incremental Clustering (CBIC), based on the k-means clustering method which is capable of re-partitioning the object set when the attribute set changes [Serban and Campan 2005]. The input to the CBIC method is an already partitioned set, partitioned prior to the change in the attribute set either by k-means or by CBIC. For feature-extended object sets, CBIC method arrived at the result more efficiently than the k-means, because the k-means has to run from the scratch each time the attribute set was changed. Experiments proving the efficiency of the method were also described. The new demand points that arrive one at a time have been assigned
either to an existing cluster or a newly created one by the algorithm in the incremental versions of Facility Location and k-median to maintain a good solution [Fotakis 2006].

2.3 Incremental clustering of categorical data

Al-Razgan et. al. focusses on ensembles for categorical data to the partitions provided by the COOLCAT algorithm [Al-Razgan et. al., 2007]. It incrementally builds clusters based on the entropy criterion. Formally, the entropy measures the uncertainty associated to a random variable. Let $X$ be a random variable with values in $S(X)$, and let $p(x)$ be the corresponding probability function of $X$. The entropy of $X$ is defined in Eq. 2.2 as follows:

$$H(X) = - \sum_{x \in S(X)} p(x) \log(p(x))$$

COOLCAT consists of two main phases. During the initialization phase, it bootstraps the algorithm and selects two points with maximum pair wise entropy and places them in two different clusters. It then proceeds incrementally, selecting the point that maximizes the minimum pair wise entropy with the previously chosen points. At the end, the $k$ selected points are the initial seeds of the clusters. During the incremental phase, it constructs $k$ clusters. For each data point, it computes the entropy resulting from placing the point in each cluster, and then assigns the point to the cluster that gives the minimum entropy. By including reprocessing at the end of each incremental step, COOLCAT alleviates the risk imposed by the order of the input of points.
The paper ‘An Incremental Clustering with Attribute Unbalance Considered for Categorical Data’ [Chen et al. 2009] analyses limitations of the categorical clustering algorithms. Based on the observations, a new similarity measure is proposed for categorical data which considers the unbalance of attributes. An incremental clustering algorithm with linear computing complexity is presented and results indicate that it outperforms other proposed categorical clustering algorithms.

A framework for performing clustering on the categorical time-evolving data is proposed by Chen H.L et al. Instead of designing a specific clustering algorithm, a generalized clustering framework is used which utilizes existing clustering algorithms and detects if there is a drifting concept or not in the incoming data. In order to detect the drifting concepts, the sliding window technique is adopted based on which the latest data points in the current window are tested to see if the characteristics of clusters are similar to the last clustering result or not [Chen H.L et al. 2009].

A mechanism named MAximal Resemblance Data Labeling (MARDL) is used to allocate each unlabeled data point into the corresponding appropriate cluster based on a practical categorical clustering representative, named “Node Importance Representative” (NIR) [Chen H.L et. al., 2005] which represents clusters by measuring the importance of each attribute value in the clusters. Based on NIR, the “Drifting Concept Detection” (DCD) algorithm is proposed. In DCD, the incoming categorical data points at the present sliding window are first allocated into the corresponding proper cluster at the last clustering result, and the number of outliers that are not able to be assigned into any cluster is counted. After that, the distribution
of clusters and outliers between the last clustering result and the current temporal clustering result are compared with each other. If the distribution is changed (exceeding some criteria), the concepts are said to drift. In the concept-drifting window, the data points will do reclustering, and the last clustering representative will be dumped out. On the contrary, if the concept is steady, the clustering representative (NIR) will be updated.

This framework not only detects the drifting concepts in the categorical data but also explains the drifting concepts by analyzing the relationship between clustering results at different times. The analyzing algorithm is named “Cluster Relationship Analysis” (CRA). When the drifting concept is detected by DCD, the last clustering representative is dumped out. Therefore, each clustering representative that had been recorded represents a successive constant clustering result, and different dumped-out representatives describe different concepts in the data set. By analyzing the relationship between clustering results, we may capture the time-evolving trend that explains why the clustering results have changes in the data set.

A fast incremental clustering algorithm is proposed by Xiaoke et al., where the radius threshold value is changed dynamically. It restricts the number of the final clusters and reads the original dataset only once. At the same time an inter-cluster dissimilarity measure taking into account the frequency information of the attribute values is introduced for handling the categorical data as in Eq. 2.3.

\[
dif(C^{(t)}_i, C^{(t)}_j) = \sum_{o \in (C_i \cup C_j \cup D)} \frac{|RelFreq_{C_i \cup D_i} (o) - RelFreq_{C_j \cup D_j} (o)|}{(RelFreq_{C_i \cup D_i} (o) + RelFreq_{C_j \cup D_j} (o)) \cdot o \cdot (C_i \cup C_j \cup D_i)}
\]  

(2.3)
This algorithm overcomes the problem of inadequate memory when clustering the large scale data sets, and also accurately reflects the characteristics of the data set [Xiaoke et al. 2009].

An incremental clustering for categorical data using clustering ensemble was proposed by Li Taoying et. al., and connects incremental clustering and clustering ensemble and take the advantages of both of them. Redundant attributes are pruned and then true values of different attributes are used to form clustering memberships according to the values of categorical attributes in the incremental mode. Next clustering ensemble is used to merge or divide clusters to gain optimal clustering. This algorithm is applied on UCI datasets and results show that it is effective. [Li Taoying et. al., 2010].

An incremental algorithm to cluster categorical data with frequency based similarity measure is proposed which is similar to M-Squeezer, with new similarity measures proposed based on frequency of attribute values with respect to the number of attributes, domain and the cluster size, cardinality of the domain of matching attribute values.

i) Frequency of attribute values with respect to the number of attributes is given in Eqs 2.4 and 2.5.

\[
sim(X_i, Q_j) = \sum_{l=1}^{m} \hat{c}(x_{i,l}, q_{j,l})
\]

(2.4)
ii) Frequency of attribute values with respect to domain and the cluster is given as in Eqs 2.6 and 2.7.

\[
\text{sim}(X_l, Q_j) = \sum_{i=1}^{m} \vartheta(x_{ii}, q_{jj})
\]

(2.6)

\[
\vartheta(x_{ii}, q_{jj}) = \begin{cases} 
\frac{f(v_{i}/C_j)}{f(C_j)f_{m}} & \text{if } x_{ii} = q_{jj} \\
0 & \text{if } x_{ii} \neq q_{jj} 
\end{cases}
\]

(2.7)

iii) Frequency of attribute values with respect to cardinality of the domain of matching attribute values is given below in Eqs 2.8 and 2.9.

\[
\text{sim}(X_l, Q_j) = \frac{\sum_{l=1}^{m} \vartheta(x_{ii}, q_{jj})}{m + 1/\sum_{l=1}^{m} w_l}
\]

(2.8)
By varying the threshold value, the best number of clusters can be selected from the result obtained [Aranganayagi and Thangavel 2010].

2.4 Swarm intelligence based incremental clustering algorithms

Chen and Meng have implemented the clustering process by using clustering agents. These clustering agents move in a three-dimensional space and have the abilities of memory, communication, analysis, judgment, coordination and so on.

This paper presents two clustering criteria to evaluate the clustering results:

(1) Rate of contractility ($C_r$) $: C_r = \left( \frac{m_{best}}{m_{result}} \right) \times 100\%$ (2.10)

(2) Rate of veracity ($V_r$) $: V_r = \left( \frac{m_{right}}{m_{all}} \right) \times 100\%$ (2.11)

where $m_{best} =$ number of perfect clustering results

$m_{result} =$ number of experimental results

$m_{right} =$ number of correct clustering data

$m_{all} =$ number of all data
This algorithm is applicable in periodically incremental environment and has many merits like insensitivity to the order of the data, capability of dealing with the exceptional, high-dimensional and complicated data [Chen and Meng 2004].

Bo Liu et al., have aimed at constructing the knowledge model incrementally for a dynamically changing database by making use of a swarm of special agents, ie an ant colony, and imitate their natural behaviors to form clusters of arbitrary shape gradually. There is no need to pre-specify the number of clusters. The clustering model includes initializing clusters, modifying the previously discovered knowledge using the new data without retraining the old data, and maintaining clusters with a changing grid. Information entropy is applied to model behaviors of agents, such as picking up and dropping objects, and to guide agent movement by pheromone in incremental stages. Fewer parameters are needed to set; clustering speed is fast and advantageous to data mining systems which modify databases either periodically or in batches [Bo Liu et al. 2006]. Information entropy and pheromone concentration are specified in the following Eqs. 2.12 to 2.14

\[ E(s^2) = - \sum_{i=1}^{n} \sum_{x \in A_i} p(x) \log p(x) \]  
\[ P(x) = \frac{x_{num}}{obj_{num}} \]  
\[ \tau(x, y) = \frac{obj_{num}}{1 + s \times s} \]
Where $E(S^2)$ denotes the entropy of the $s \times s$ area in which the agent lies

$$T(x,y)$$
denotes the pheromone concentration in $s \times s$

$x_{num}$ is the number of objects whose attribute $x_i$ has value $x$

$\text{obj}_{num}$ is the total number of objects in $s \times s$.

### 2.5 Incremental clustering of streaming data

DUCstream is an incremental, one-pass density-based algorithm, which finds high-quality clusters with considerably little time and memory in the data stream environment [Jing et al. 2005]. The density of a unit, $\text{den} (u)$ is defined as the number of points that belong to it $\text{den} (u) = | v_i | v_i \in u$. The relative density of $u$, $\text{rel}_{\text{den}} = \text{den} (u) / | D |$. If $u$’s relative density is greater than the density threshold $\gamma$, then $u$ is referred to as a dense unit. Evolving clusters are identified on the basis of the dense units, which contain relatively large number of points. It discards noisy and obsolete units through dense units detection. The clustering result is updated using the changed dense units. A bitwise clustering representation is used to update and store away the clustering results efficiently.

The need to extract and retain meaningful information from data streams produced by applications such as large scale surveillance, network packet inspection and financial transaction monitoring has motivated the design of an incremental graph-based clustering algorithm [Sebastian and Mihai 2009]. Cluster representation involves the use of dynamically updated sparse graphs in conjunction with a
repository of representative vertices. Thus a cluster’s history can be rebuilt and it can rapidly adapt to significant patterns previously observed. The aim of RepStream algorithm is to capture such patterns in order to recall them at some future time should the change reoccur. RepStream is a single phase incremental algorithm that updates two sparse graphs of k-nearest neighbour connected vertices in order to identify clusters among data points. The first graph is used to capture the connectivity relationships amongst the most recently seen data points and to select a set of representative vertices where as the second graph is used to track the connectivity between the chosen representative vertices. The connectivity of the representative vertices on both graphs then forms the basis for the algorithm’s clustering decision making.

RINO-STREAMS [Hawwash and Nasraoui 2010], is based on incrementally updating the clustering model using the newly arrived data point, and thus maintaining the model summaries (clusters) over time. Evolving clusters are extracted from a massive data stream in one pass. A robust estimation of centroids is used to represent the location of the cluster center with respect to other clusters at any time. It is also based on a robust distribution independent statistical test (Chebyshev) for the detection of outliers and for the merging of compatible clusters, thus assuring robustness and compactness of the extracted cluster model. The distinguished contributions of this paper are its statistically based robustness to noise, dynamic estimation of scales, and fast analytical optimization compared to existing stream clustering methods. Moreover, it can adapt to the evolution of the clusters in the input data stream.
2.6  Incremental clustering algorithms for other applications

A new evolutionary clustering technique has been proposed by Gorunescu and Dumitrescu, to represent an evolutionary variant of the incremental clustering technique. The usually used Euclidean distance is replaced by a function called ‘Category utility’ which maximizes both the probability that instances in the same class have common attribute values and the probability that instances from different classes do not. In addition to the crossover and mutation operators, a new variation operator ‘increment’ is introduced to retain the incremental nature. It is applied for every chromosome, taking randomly a gene of the current one, whose value is necessarily zero, and assigning it either the number for the next cluster to be formed or the number of an existing cluster. It actually puts an undistributed instance in a new or an existing cluster [Gorunescu and Dumitrescu 2003].

Fotakis proposed an incremental algorithm which accomplishes a constant performance ratio for Facility Location by considering the case of uniform facility costs, where the opening cost of a facility is the same for each and every point. The definition of Incremental Facility Location is as follows: Demand points arrive one at a time and must be assigned to either an existing or a new facility upon arrival. At any point in time, the algorithm can also merge a facility with another one by closing the first facility and re-assigning all the demands currently assigned to it to the second facility. The assignment cost of a demand is its distance from the facility the demand is currently assigned to. Utilizing this algorithm as a basic unit, he developed the first incremental algorithm that utilizes O(k) medians to accomplish a constant performance ratio for k-Median [Fotakis 2006].
A fast incremental clustering algorithm called ICGD is given by Chen et al., which makes use of grid and density to achieve real time clustering of the dynamic data. Capturing the shape of data space using condensation points, utilizing grid-based and density-based clustering methods founded on the theories of climbing hill algorithm and connectedness, implementing incremental cluster utilizing the difference data were the novelties of the proposed algorithm [Chen et al. 2007]. The algorithm has the good characteristics of grid-based and density-based clustering methods, and at the same time it does not have the drawback of quality degradation associated with the conventional grid-based clustering method that arises because of little or no contemplation to data distribution when the grids are partitioned. Its accuracy and effectiveness in realizing incremental clustering process has been confirmed by experimental results.

The Belief K-modes Method (BKM) just like traditional clustering methods starts with a known dataset of objects identified by a given set of attributes. But, the attribute set changes in many applications. Hence, to cluster the unsure data present in such changing environment, an Incremental Belief K-modes Method (IBKM) has been presented by Hariz and Elouedi. The main aim of this method was to manage clusters efficiently without performing clustering from the scratch each time new attributes were introduced [Hariz and Elouedi 2008].

In the paper ‘Extensions of vector quantization for incremental clustering’ the conventional vector quantization has been extended by Edwin. The proposed method has demonstrated a one-pass incremental and evolving variant of vector quantization, which builds up and updates clusters sample per sample with new incoming data. It
also omits the pre-definition of the number of clusters, which has to be sent as parameter into conventional vector quantization. A different distance strategy is incorporated by taking the distance of new incoming points to the range of influence of clusters and not to the cluster centers themselves. Also, a satellite deletion strategy is proposed to remove not significant clusters (satellites) after the complete learning process. A split-and-merge strategy is described, which guides the incremental clustering process to cluster partitions with a high quality can be generically applied after each incremental learning step for all incremental clustering variants, to update cluster centers and ranges of influence. In addition, two main extensions were made: choosing the ‘winning cluster’ determined by the distances between a data point and the surface of all clusters and using an online split-and-merge approach for deleting the cluster satellites. A bad priori setting of the most essential parameter(s) was restrained by both the extensions; hence creation of a wrong cluster partition was avoided [Edwin 2008].

An anomaly detection algorithm [Fei R. et al. 2008] has been built for dynamically updating normal profile of system usage pattern. Program behavior was the characteristic that was utilized to model the usage pattern of system behavior. Each weighting values of system calls are clustered by DBSCAN which generate a series of statistical profiles for system monitoring. It has been proved that the insert or delete operation affects clusters generated by DBSCAN only in the neighborhood of inserted or deleted object. Therefore, a new density-based incremental clustering algorithm is given which implements updating just by executing local changes on clustering results generated by DBCSCAN. It was much more efficient than
conventional updating methods that employed re-clustering. It has been proved based on the experiments with 1998 DARPA BSM audit data, that normal profiles created by the proposed algorithm was less sensitive to noise data objects than profiles created by the analogous incremental algorithm ADWICE. The algorithm has incremental detection quality and reduced false alarm rate.

Serhat and Gunsel proposed a Nonnegative matrix factorization (NMF) based incremental clustering algorithm in which each sample is processed incrementally and cluster centroids are updated accordingly. Nonnegative matrix factorization deals with the problem of factorizing a nonnegative data matrix into two matrices under the constraint that factor matrices are restricted to be nonnegative. Incremental nonnegative matrix factorization (INMF), is considered as a factorization tool for clustering and performs clustering by optimizing a cost function equivalent to K-means. An extension to INMF is made for addressing the problem of determining the number of clusters which also corresponds to optimal rank selection in factorization. Unlike K-means clustering which allows decreasing the number of clusters, this algorithm permits suitable increase of the number of clusters enabling a compact and efficient clustering scheme for rank selection. It is shown that INMF as a clustering tool provides additional benefits such as reducing computational load and being suitable to online clustering applications [Serhat and Gunsel 2009].

2.7 Handling mixed data

Huang proposed an algorithm based on the k-means paradigm for clustering large data sets with mixed numeric and categorical values. [Huang 1997]. The
algorithm handles objects with numeric and categorical attributes and clusters them in a way similar to k-means. Objects are clustered with respect to k prototypes instead of k means. This method dynamically updates the k prototypes in order to maximize the intra cluster similarity of objects. The object similarity measure is derived from both numeric and categorical attributes. The similarity measure on numeric attributes is the square Euclidean distance whereas the similarity measure on categorical attributes is the number of mismatches between objects and cluster prototypes. Weight $\gamma_1$ is introduced to avoid favoring either type of attribute. The choice of $\gamma_1$ is dependent on distributions of numeric attributes. Generally $\gamma_1$ is related to the overall average standard deviation $\sigma$ of numeric attributes.

A GA-based clustering algorithm for large data sets with mixed and categorical values has presented common cost function, trace of the within cluster dispersion matrix adjustment based clustering algorithm for mixed data sets [Li Jie et al. 2003]. For a given number of objects $n$, the number of possible partitions of the object-set is definite but extremely large. It is impractical to investigate every partition in order to find a better solution. A common solution is to choose a clustering criterion called a cost function to guide the search for a partition. The trace of the within cluster dispersion matrix is used as the cost function. Valid clustering result has been obtained by optimizing the cost function using genetic algorithm (GA). The feasibility the GA-based clustering algorithm for huge data sets with mixed numeric and categorical values has been evident from the experimental results.
A novel divide-and-conquer technique [Zengyou et al. 2005] is used to solve the problem of datasets with mixed type of attributes. First the original mixed dataset is divided into sub datasets: the pure categorical dataset and the pure numeric dataset. Next, existing well established clustering algorithms designed for different types of datasets are employed to produce corresponding clusters. Finally, the clustering results on the categorical and numeric dataset are combined as a categorical dataset, on which the categorical data clustering algorithm is exploited to get the final clusters. This algorithm gets clustering output by splitting both categorical and numeric dataset; therefore it is named as CEBMDC (Cluster Ensemble Based Mixed Data Clustering). A graph-partitioning based algorithm is used for clustering numeric dataset and the squeezer algorithm is used as the clustering algorithm for both categorical data clustering and cluster ensemble.

Amir and Lipika have proposed a co-occurrence of values based effective cost function and distance measure. The importance of an attribute towards the clustering process has been taken into account by the measures. The proposed cost function $\delta (p,q)$ denotes the distance between a pair of distinct values $p$ and $q$ of an attribute and is computed as a function of their co-occurrence with other attribute values. Significance of an attribute means the contribution of an attribute towards clustering which is denoted by the weight in the distance function is computed in terms of cost function. Thus, the weighing values are extracted from the attribute value distributions within the data and need not be user defined. The distance between any two categorical attribute values is computed as a function of the overall distribution of values in a single class, as also the overall distribution of values in the
data set. This distance measure also gives the measure of significance for each attribute and is capable of providing good insight into the heterogeneity or homogeneity of data objects. A modified description of cluster center is provided where the cluster centre for numeric attributes is represented by its mean but for categorical attributes the cluster center is represented by the proportional distribution of each categorical value in the cluster. Using real world data sets, the performance of the algorithm has been analyzed and its effectiveness has been demonstrated by comparing it with other clustering algorithms [Amir and Lipika 2007].

A fuzzy c-means-type algorithm for Gauss-Multinomial-distributed data called KL-FCM-GM has been presented which employs a probabilistic dissimilarity functional to permit efficient handling of data with mixed numeric and categorical attributes. In this paper, the Gath–Geva (GG) algorithm which is one of the most popular methodologies for fuzzy c-means (FCM)-type clustering of data comprising numeric attributes is extended and a novel FCM-type algorithm has been proposed [Sotirios 2011]. The dissimilarity functional is selected as the negative log-likelihood of the $i^{th}$ postulated cluster with respect to the data point $x_j$, i.e., the negative log of the probability density function $p(x_j \mid \Theta_i)$.

$$d_{ij} = -\log p(x_j \mid \Theta_i) \tag{2.15}$$

Fuzzification is attained by means of a regularization technique where a Kullback–Leibler (KL) divergence term is introduced into the fuzzy objective function as shown in Eq. 2.15.
where \( \pi_i \) is the weight of the \( i \)th cluster and the parameter \( \lambda \) is the model’s degree of fuzziness of the fuzzy membership values. The effectiveness of the proposed approach has been evaluated utilizing benchmark data, and compared with alternative fuzzy and non-fuzzy clustering algorithms.

### 2.8 Clustering of mixed data points incrementally

An incremental clustering algorithm based on distance hierarchy is proposed by Hsu and Huang in which the similarity information embedded between categorical attribute is considered during clustering [Hsu and Huang 2008]. Each attribute of the data is associated with a distance hierarchy with link weights representing the distance between concepts. The distance between two mixed data patterns is then calculated according to distance hierarchies. The distance hierarchy tree is a concept hierarchy structure which facilitates the representation and computation of the distance between categorical values. Experiments have been performed using artificial simulation materials and a sample of gathered actual data related to family income.

This literature study highlights the important research advancements in the field of incremental clustering. Various incremental clustering algorithms for mining dynamic and large datasets containing either numerical or categorical attributes have been studied. Swarm-based incremental clustering algorithms and incremental clustering algorithms for data streams and other applications were also discussed. It
can be seen that there are a considerable number of clustering algorithms for handling mixed data but significant research has not been done on incremental clustering algorithms of the same. This signifies the need and development of simple and efficient incremental clustering algorithms for mixed data.

In this thesis, the author proposes two incremental clustering algorithms namely, CFICA (Cluster Feature-Based Incremental Clustering Approach for numerical data) and M-CFICA (Cluster Feature-Based Incremental Clustering Approach to Mixed Data) which are presented in chapter 3 and chapter 4 respectively.