A Study of Clustering and Classification Algorithms Used in Datamining

Sandhia Valsala **Jissy Ann George** **Priyanka Parvathy**

College of Computer Studies AMA International University Salmabad,Kingdom of Bahrain

Abstract
Clustering and classification of data is a difficult problem that is related to various fields and applications. Challenge is greater, as input space dimensions become larger and feature scales are different from each other. The term "classification" is frequently used as an algorithm for *all* data mining tasks[1]. Instead, it is best to use the term to refer to the category of supervised learning algorithms used to search interesting data patterns. While classification algorithms have become very popular and ubiquitous in DM research, it is just but one of the many types of algorithms available to solve a specific type of DM task[12]. In this paper various clustering and classification algorithms are going to be addressed in detail. A detailed survey on existing algorithms will be made and the scalability of some of the existing classification algorithms will be examined.

Keywords
DM, clustering, classification, supervised learning, scalability

I. Introduction

There are so many methods for data classification. Generally the selection of a particular method may depend on the application. The selection of a particular methodology for data classification may depend on the volume of data and the number of classes present in that data. Further, the classification algorithms are designed in a custom manner for a specific purpose to solve a particular classification scenario.

The essence of clustering data is to identify homogeneous groups of objects based on the values of their attributes. It is a problem that is related to various scientific and applied fields and has been used in science and in the field of data mining for a long time, with applications of techniques ranging from artificial intelligence and pattern recognition to databases and statistics [13]. There are different types of clustering algorithms for different types of applications and a common distinction is between hierarchical and partitioning clustering algorithms. But although numerous related texts exist in the literature, clustering of data is still considered an open issue, basically because it is difficult to handle in the cases that the data is characterized by numerous measurable features. This is often referred to as the curse of dimensionality. Although hierarchical clustering methods are more flexible than their partitioning counterparts, in that they do not need the number of clusters as an input, they are less robust in some other ways. More specifically, errors from the initial steps of the algorithm tend to propagate throughout the whole procedure to the final output. This could be a major problem, with respect to the corresponding data sets, resulting to misleading and inappropriate conclusions. Moreover, the considerably higher computational complexity that hierarchical algorithms typically have makes them inapplicable in most real life situations, due to the large size of the data sets. Works in the field of classification focus in the usage of characterized data, also known as training data, for the automatic generation of systems that are able to classify (characterize) future data. This classification relies on the similarity of incoming data to the training data. The main aim is to automatically generate systems that are able to correctly classify incoming data [12]. Although the tasks of classification and clustering are closely related, an important difference exists among them. While in the task of classification the most important part is the distinction between classes, i.e. the detection of class boundaries, in the task of clustering the most important part is the identification of cluster characteristics. The latter is usually tackled via the selection of cluster representatives or cluster centroids). Typically, in order to achieve automatic classification systems generation, one first needs to detect the patterns that underlie in the data, in contrast to simply partitioning data samples based on available labels [17], and then study the way these patterns relate to meaningful classes. Efficient solutions have been proposed in the literature for both tasks, for the case in which a unique similarity or dissimilarity measure is defined among input data elements [6]. When, on the other hand, multiple independent features characterize data, and thus more than one meaningful similarity or dissimilarity measures can be defined, both tasks become more difficult to handle.

II. The Role of Classification in Data Mining

Based on the data collected, data mining algorithms are used to either produce a description of the data stored, or predict an outcome[10]. Different kinds of algorithms are used to achieve either one of these tasks. However, in the

Manuscript received October 5, 2011
Manuscript revised October 20, 2011

overall KDD process, any mixture of these tasks may be called upon to achieve the desired results. The Steps involved in KDD are:

a) Description tasks: These tasks describe the data being mined and they are:

i) Summarization: To extract compact patterns that describe subsets of data. The method used to achieve this task are Association Rule algorithms.

ii) Segmentation or Clustering: To separate data items into subsets that are similar to each other. Partition-based clustering algorithms are used to achieve this task.

iii) Change and Deviation Detection: To detect changes in sequential data (such as protein sequencing, behavioral sequences, etc.).

iv) Dependency Modeling: To construct models of causality within the data.

b) Prediction tasks: To predict some field(s) in a database based on information in other fields.

i) Classification: To predict the most likely state of a categorical variable (its class).

ii) Regression: To predict results that are numeric continuous variables.

Classification

Model Construction

Model construction is building the model from the training set

- Each tuple/sample is assumed to belong a prefined class
- The class of a tuple/sample is determined by the class label attribute
- The training set of tuples/samples is used for model construction
- The model is represented as classification rules, decision trees or mathematical formulae

Model Usage

- Classify future or unknown objects
- Estimate accuracy of the model
- the known class of a test tuple /sample is compared with the result given by the mode
- accuracy rate = percentage of the tests tuples /samples correctly classified by the model

III. Existing Methods in Data Classification

With an enormous amount of data stored in databases and data warehouses, it is increasingly important to develop powerful tools for analysis of such data and mining interesting knowledge from it. Data mining is a process of inferring knowledge from such huge data. Data Mining has three major components Clustering or Classification, Association Rules and Sequence Analysis. Data Classification is an important step in data mining applications.

By simple definition, in classification/clustering we analyze a set of data and generate a set of grouping rules which can be used to classify future data. For example, one may classify diseases and provide the symptoms which describe each class or subclass. This has much in common with traditional work in statistics and machine learning. However, there are important new issues which arise because of the sheer size of the data. One of the important problem in data mining is the Classification-rule learning which involves finding rules that partition given data into predefined classes. In the data mining domain where millions of records and a large number of attributes are involved, the execution time of existing algorithms can become prohibitive, particularly in interactive applications. In Data classification one develops a description or model for each class in a database, based on the features present in a set of class-labeled training data[17]. There have been many data classification methods such as decision-tree methods, such as C4.5, statistical methods, neural networks, rough sets, database-oriented methods etc.

Data Classification Methods

The following list shows the available data classification methods.

- **Statistical Algorithms** Statistical analysis systems such as SAS and SPSS have been used by analysts to detect unusual patterns and explain patterns using statistical models such as linear models. Such systems have their place and will continue to be used.
- **Neural Networks** Artificial neural networks mimic the pattern-finding capacity of the human brain and hence some researchers have suggested applying Neural Network algorithms to pattern-mapping. Neural networks have been applied successfully in a few applications that involve classification.
- **Genetic algorithms** Optimization techniques that use processes such as genetic combination,

mutation, and natural selection in a design based on the concepts of natural evolution.

- **Nearest neighbor method** A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset. Sometimes called the k-nearest neighbor technique.
- **Rule induction** The extraction of useful *if-then* rules from data based on statistical significance.
- **Data visualization** The visual interpretation of complex relationships in multidimensional data.

Many existing algorithms suggest abstracting the test data before classifying it into various classes. There are several alternatives for doing abstraction before classification: A data set can be generalized to either a minimally generalized abstraction level, an intermediate abstraction level, or a rather high abstraction level. Too low an abstraction level may result in scattered classes, bushy classification trees, and difficulty at concise semantic interpretation; whereas too high a level may result in the loss of classification accuracy.

1) Classification-rule learning

Classification-rule learning involves finding rules or decision trees that partition given data into predefined classes[1]. For any realistic problem domain of the classification-rule learning, the set of possible decision trees is too large to be searched exhaustively. In fact, the computational complexity of finding an optimal classification decision tree is NP hard.

Most of the existing induction-based algorithms use **Hunt**'s method as the basic algorithm. Here is a recursive description of Hunt's method for constructing a decision tree from a set T of training cases with classes denoted $\{C_1, C_2, \ldots, C_k\}$.

Case 1 T contains one or more cases, all belonging to a single class C_j : The decision tree for T is a leaf identifying class C_j.

Case 2 T contains no cases: The decision tree for T is a leaf, but the class to be associated with the leaf must be determined from information other than T.

Case 3 T contains cases that belong to a mixture of classes: A test is chosen, based on a single attribute, that has one or more mutually exclusive outcomes $\{O_1, O_2, .., O_n\}$. T is partitioned into subsets $T_1, T_2, \ldots, T_n$, where T_i contains all the cases in T that have outcome O_i of the chosen test. The decision tree for T consists of a decision node identifying the test, and one branch for each possible outcome. The same tree building machinery is applied recursively to each subset of training cases.

2) Decision Trees

A decision tree is a classification scheme which generates a tree and a set of rules, representing the model of different classes from a given data set[12]. The set of records available for developing classification methods is generally divided into two disjoint subsets as follows:

(i) a training set - used for deriving the classifier
(ii) a test set - used to measure the accuracy of the classifier

The accuracy of the classifier is determined by the percentage of the test examples that are correctly classified. The attributes of the records are divided into two types as follows:

Numerical attributes - attributes whose domain is numerical
Categorical attributes - attributes whose domain is not numerical

There is one distinguished attribute called the *class label*. The goal of the classification is to build a concise model that can be used to predict the class of the records whose class label is not known.

A decision tree is a tree where the internal node - is a test on an attribute, the tree branch - is an outcome of the test, and the leaf node - is a class label or class distribution.

There are two phases of decision tree generation:

Tree construction

- at start, all the training examples are at the root
- partition examples based on selected attributes
- test attributes are selected based on a heuristic or a statistical measure

Tree pruning

- identify and remove branches that reflect noise or outliers
- One rule is generated for each path in the tree from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are generally simpler to understand than trees

Tree Construction Principle

There are various methods of building decision trees from a given training data set. Some basic concepts involved in the building of decision trees are discussed below.

Splitting Attribute

With every node of the decision tree, there is an associated attribute whose values determine the partitioning of the data set when the node is expanded.

Splitting Criterion

The qualifying condition on the splitting attribute for data set splitting at a node is called the splitting criterion at that node. For a numeric attribute, the criterion can be an equation or an inequality. For a categorical attribute, it is a membership condition on a subset of values.

3) Decision Tree Construction Algorithms

A number of algorithms for inducing decision trees have been proposed over the years. They differ among themselves in the methods employed for selecting splitting attributes and splitting conditions. These algorithms can be classified into two types. The first type of algorithms is the classical algorithms which handle only memory resident data. The second category can handle the efficiency and scalability issues. These algorithms remove the memory restrictions and are fast and scalable.

4) CART (Classification And Regression Trees)

It is one of the popular methods of building decision trees. CART builds a binary decision tree by splitting the records at each node, according to a function of a single attribute[15]. CART uses the gini index for determining the best split. The initial split produces two nodes, each of which is split in the same manner as the root node. If no split is found which reduces the diversity of a given node, it is labeled as the leaf node.

When the full tree is grown, only the leaf nodes remain. At the end of the tree growing process, every record of the training set has been assigned to some leaf of the full decision tree. Each leaf can now be assigned a class and an error rate. The error rate of a leaf node is the percentage of incorrect classification at that node. The error rate of an entire decision tree is a weighted sum of the error rates of all the leaves. Each leaf's contribution to the total is the error rate at that leaf multiplied by the probability that the record will end up there.

5) ID3 algorithm

The ID3 algorithm (**Quinlan**86) is a decision tree building algorithm which determines the classification of objects by testing the values of the their properties[12]. It builds the tree in a top down fashion, starting from a set of objects and a specification of properties. At each node of the tree, a property is tested and the results used to partition the object set. This process is recursively done till the set in a given subtree is homogeneous with respect to the classification criteria - in other words it contains objects belonging to the same category. This then becomes a leaf node. At each node, the property to test is chosen based on information theoretic criteria that seek to maximize information gain and minimize entropy. In simpler terms, that property is tested which divides the candidate set in the most homogeneous subsets.

6) C4.5 algorithm

This algorithm was proposed by **Quinlan** (1993). The C4.5 algorithm generates a classification-decision tree for the given data-set by recursive partitioning of data[14]. The decision is grown using **Depth-first** strategy. The algorithm considers all the possible tests that can split the data set and selects a test that gives the best information gain. For each discrete attribute, one test with outcomes as many as the number of distinct values of the attribute is considered. For each continuous attribute, binary tests involving every distinct values of the attribute are considered. In order to gather the entropy gain of all these binary tests efficiently, the training data set belonging to the node in consideration is sorted for the values of the continuous attribute and the entropy gains of the binary cut based on each distinct values are calculated in one scan of the sorted data. This process is repeated for each continuous attributes.

7) SLIQ and SPRINT algorithms

SLIQ (Supervised Learning In Quest) developed by IBM's Quest project team, is a decision tree classifier designed to classify large training data [1]. It uses a pre-sorting technique in the tree-growth phase. This helps avoid costly sorting at each node.

SLIQ keeps a separate sorted list for each continuous attribute and a separate list called class list. An entry in the class list corresponds to a data item, and has a class label and name of the node it belongs in the decision tree. An entry in the sorted attribute list has an attribute value and the index of data item in the class list. SLIQ grows the decision tree in **breadth-first** manner.

For each attribute, it scans the corresponding sorted list and calculate entropy values of each distinct values of all the nodes in the frontier of the decision tree simultaneously. After the entropy values have been

calculated for each attribute, one attribute is chosen for a split for each nodes in the current frontier, and they are expanded to have a new frontier. Then one more scan of the sorted attribute list is performed to update the class list for the new nodes.
While SLIQ handles disk-resident data that are too large to fit in memory, it still requires some information to stay memory-resident which grows in direct proportion to the number of input records, putting a hard-limit on the size of training data. The Quest team has recently designed a new decision-tree-based classification algorithm, called SPRINT (**S**calable **P**a**R**allelizable **IN**duction of decision **T**rees) that for the removes all of the memory restrictions.

8) CHAID (Chi-Square Automatic Interaction Detector)

CHAID, proposed by Kass in 1980, is a derivative of AID (Automatic Interaction Detection), proposed by Hartigan in 1975. CHAID attempts to stop growing the tree before overfitting occurs. The decision tree is constructed by partitioning the data set into two or more subsets, based on the values of one of the non-class attributes. After the data set is partitioned according to the chosen attributes, each subset is considered for further partitioning using the same algorithm. This process is repeated for each subset until some stopping criterion is met. In CHAID, the number of subsets in a partition can range from two up to the number of distinct values of the splitting attribute. In this regard, CHAID differs from the CART, which always forms binary splits, and from ID3 and C4.5, which form a branch for every distinct value.

9) Naïve k-means algorithm

One of the most popular heuristics for solving the k-means problem is based on a simple iterative scheme for finding a locally optimal solution. This algorithm is often called the k-means algorithm. There are a number of variants to this algorithm, so to clarify which version we are using, we will refer to it as the naïve k-means algorithm as it is much simpler compared to the other algorithms described here.
The naive k-means algorithm partitions the dataset into 'k' subsets such that all records, from now on referred to as points, in a given subset "belong" to the same center. Also the points in a given subset are closer to that center than to any other center. The partitioning of the space can be compared to that of Voronoi partitioning except that in Voronoi partitioning one partitions the *space* based on distance and here we partition the *points* based on distance.
The algorithm keeps track of the centroids of the subsets, and proceeds in simple iterations. The initial partitioning is randomly generated, that is, we randomly initialize the centroids to some points in the region of the space. In each iteration step, a new set of centroids is generated using the existing set of centroids following two very simple steps. Let us denote the set of centroids after the i^{th} iteration by $C^{(i)}$. The following operations are performed in the steps:

(i) Partition the points based on the centroids C(i), that is, find the centroids to which each of the points in the dataset belongs. The points are partitioned based on the Euclidean distance from the centroids.

(ii) Set a new centroid $c(i+1) \in C(i+1)$ to be the mean of all the points that are closest to $c(i) \in C(i)$ The new location of the centroid in a particular partition is referred to as the new location of the old centroid.

The algorithm is said to have converged when recomputing the partitions does not result in a change in the partitioning. In the terminology that we are using, the algorithm has converged completely when $C^{(i)}$ and $C^{(i-1)}$ are identical. For configurations where no point is equidistant to more than one center, the above convergence condition can always be reached. This convergence property along with its simplicity adds to the attractiveness of the k-means algorithm.
The naïve k-means needs to perform a large number of "nearest-neighbor" queries for the points in the dataset. If the data is 'd' dimensional and there are 'N' points in the dataset, the cost of a single iteration is O(kdN). As one would have to run several iterations, it is generally not feasible to run the naïve k-means algorithm for large number of points.
Sometimes the convergence of the centroids (i.e. $C^{(i)}$ and $C^{(i+1)}$ being identical) takes several iterations. Also in the last several iterations, the centroids move very little. As running the expensive iterations so many more times might not be efficient, we need a measure of convergence of the centroids so that we stop the iterations when the convergence criteria is met. Distortion is the most widely accepted measure.
Clustering error measures the same criterion and is sometimes used instead of distortion. In fact k-means algorithm is designed to optimize distortion. Placing the cluster center at the mean of all the points minimizes the distortion for the points in the cluster. Also when another cluster center is closer to a point than its current cluster center, moving the cluster from its current cluster to the other can reduce the distortion further. The above two steps are precisely the steps done by the k-means cluster. Thus k-means reduces distortion in every step locally. The k-Means algorithm terminates at a solution that is locally optimal for the distortion function. Hence, a natural choice

as a convergence criterion is distortion. Among other measures of convergence used by other researchers, we can measure the sum of Euclidean distance of the new centroids from the old centroids. In this thesis we always use clustering error/distortion as the convergence criterion for all variants of k-means algorithm.

Definition 1: *Clustering error* is the sum of the squared Euclidean distances from points to the centers of the partitions to which they belong.

Mathematically, given a clustering ϕ, we denote by $\phi(x)$ the centroid this clustering associates with an arbitrary point x (so for k-means, $\phi(x)$ is simply the center closest to x). We then define a measure of quality for ϕ:

$$distortion_{\ \phi} = \frac{1}{N}\sum_{x}\left|x-\phi(x)\right|^2$$

Where |a| is used to denote the norm of a vector 'a'. The lesser the difference in distortion over successive iterations, the more the centroids have converged. Distortion is therefore used as a measure of goodness of the partitioning. In spite of its simplicity, k-means often converges to local optima. The quality of the solution obtained depends heavily on the initial set of centroids, which is the only non-deterministic step in the algorithm. Note that although the starting centers can be selected arbitrarily, k-means is fully deterministic, given the starting centers. A bad choice of initial centers can have a great impact on both performance and distortion. Also a good choice of initial centroids would reduce the number of iterations that are required for the solution to converge. Many algorithms have tried to improve the quality of the k-means solution by suggesting different ways of sampling the initial centers, but none has been able to avoid the problem of the solution converging to a local optimum.

10) Kd-trees

A very important data structure that is used in our algorithm is a kd-tree. A kd-tree is a data structure for storing a set of finite points from a d-dimensional space.

Kd-trees are simple data structures with the following properties:

(i) They are binary trees;
(ii) The root node contains all the points;
(iii) A node is split along a split-plane such that points to the left are part of the left sub-tree, points to the right are part of the right sub-tree;
(iv) The left and right sub-trees are recursively split until there is only one point in the leaf or a certain condition is satisfied.

This is the basic kd-tree structure. There exist several variants of the kd-tree based on the way in which they choose the splitting plane, the termination criteria, etc.

Originally designed to decrease the time in nearest neighbor queries, kd-trees have found other applications as well. Omohumdro has recommended it in a survey of possible techniques to increase speed of neural network

Though kd-trees give substantial advantage for lower dimensions, the performance of kd-trees decreases/drops in higher dimensions. Other data structures like AD trees have been suggested for higher dimensions but these have never been used for k-means. After this brief introduction to the kd-trees (which is the primary data structure used in our algorithm), we discuss the two main approaches that try to counter the shortcomings of the k-means algorithm.

11) The Greedy K-means Algorithm

The local convergence properties of k-means have been improved in this algorithm. Also it does not require the initial set of centroids to be decided. The idea is that the global minima can be reached through a series of local searches based on the global clustering with one cluster less.

Assumption: The assumption used in the algorithm is that the global optima can be reached by running k-means with the (k-1) clusters being placed at the optimal positions for the (k-1) clustering problem and the k^{th} cluster being placed at an appropriate position that is yet to be discovered.

Let us assume that the problem is to find K clusters and K' ≤ K. We Use the above assumption, the global optima for k = K' clusters is computed as a series of local searches. Assuming that we have solved the k-means clustering problem for K' – 1 clusters, we have to place a new cluster at an appropriate location. To discover the appropriate insertion location, which is not known, we run k-means algorithm until convergence with each of the points in the entire set of the points in the dataset being added as the candidate new cluster, one at a time, to the K' – 1 clusters. The converged K clusters that have the minimum distortion after the convergence of k-means in the above local searches are the clusters of the global k-means.

We know that for k = 1, the optimal clustering solution is the mean of all the points in the dataset. Using the above method we can compute the optimal positions for the k = 2, 3, 4, ... K, clusters. Thus the process involves computing the optimal k-means centers for each of the K = 1, 2, 3… K clusters. The algorithm is entirely deterministic.

Though the attractiveness of the global k-means lies in it finding the global solution, the method involves a heavy cost. K-means is run N times, where N is the number of points in the dataset, for every cluster to be inserted. The complexity can be reduced considerably by not running the K-means with the new cluster being inserted at each of the dataset points but by finding another set of points that could act as an appropriate set for insertion location of the new cluster.

The variant of the kd-tree splits the points in a node using the plane that passes through the mean of the points in the node and is perpendicular to the principal component of the points in the node. A node is not split if it has less than a pre-specified number of points or an upper bound to the number of leaf nodes is reached. The idea is that even if the kd-tree were not used for nearest neighbor queries, merely the construction of the kd-tree based on this strategy would give a very good preliminary clustering of the data. We can thus use the kd-tree nodes centers as the candidate/initial insertion positions for the new clusters.

The time complexity of the algorithm can also be improved by taking a *greedy approach*. In this approach, running k-means for each possible insertion position is avoided. Instead reduction in the distortion when the new cluster is added is taken into account without actually running k-means. The point that gives the maximum decrease in the distortion when added as a cluster center is taken to be the new insertion position.

K-means is run until convergence on the new list of clusters with this added point as the new cluster. The assumption is that the point that gives the maximum decrease in distortion is also the point for which the converged clusters would have the least distortion. This results in a substantial improvement in the running time of the algorithm, as it is unnecessary to run k-means for all the possible insertion positions. However, the solution may not be globally optimal but an approximate global solution.

12) Self-Organising Map

The SOM can be characterised as "an unsupervised network that seeks to learn a continuous topological mapping of a set of inputs onto a set of outputs in such a way that the outputs acquire the same topological order as the inputs, by means of self-organisation based on data examples" (Openshaw and Wymer 1994). Neurons are typically organized in a 2D grid, and the SOM tries to find clusters such that any two clusters that are close to each other in the grid space have codebook vectors that are close to each other in the input space.

In the self-organization process the input vectors are presented to the network, and the cluster unit whose weight vector is closest (usually in terms of Euclidean distance) is chosen as the winner. The next step is to update the value of the winning unit and neighbouring units, this will approximate the values of the units to the one of the input vector. This can be viewed as a motion of the units in the direction of the input vector, the magnitude of this movement depends on the learning rate, which decreases along the process in order to obtain convergence. Bearing in mind that Vector Quantization (VQ) is essentially the same as the k-means algorithm, and that the VQ is a special case of the SOM, in which the neighbourhood size is zero, one can say that there is a close relation between SOM and k-means. Openshaw and Wymer (1994) go further and say that the basic SOM algorithm "…is essentially the same as a K means classifier; with a few differences due to neighbouring training which might well be regarded as a form of simulated annealing and it may provide better results and avoid some local optima."

13) Genetic Algorithm

Evolution has proven to be a very powerful mechanism in finding good solutions to difficult problems. One can look at the natural selection as an optimisation method, which tries to produce adequate solutions to particular environments.

In spite of the large number of applications of GA in different types of optimisation problems, there is very little research on using this kind of approach to the clustering problem. In fact, and bearing in mind the quality of the solutions that this technology has showed in different types of fields and problems (Beasley, Bull and Martin, 1993a, Mitchell, 1996) it makes perfect sense to try to use it in clustering problems.

The flexibility associated with GA is one important aspect to bear in mind. With the same genome representation and just by changing the fitness function one can have a different algorithm. In the case of spatial analysis this is particularly important since one can try different fitness functions in an exploratory phase.

In the genome each gene represents a data point and defines cluster membership. All necessary evolution operators can be implemented with this scheme. As pointed by Demiriz et all (1999) the major problem associated with this representation scheme is that it is not scalable, on the other hand it seems to be computationally efficient when the number of data points is not too large.

IV.REFERENCES

[1] Holsheimer, M., Kersten, M., Mannila, H., Toivonen, H. "A Perspective on Databases and Data Mining", Proceedings KDD '95.

[2] Carpenter, G.A. & Grossberg, S. (2003), Adaptive Resonance Theory, In M.A. Arbib (Ed.), The Handbook of

Brain Theory and Neural Networks, Second Edition. Cambridge, MA: MIT Press
[3] Grossberg, S. (1987), Competitive learning: From interactive activation to adaptive resonance, Cognitive Science (Publication).
[4] Carpenter, G.A. & Grossberg, S. (1987), ART 2: Self-organization of stable category recognition codes for analog input patterns, Applied Optics.
[5] Carpenter, G.A., Grossberg, S., & Rosen, D.B. (1991), ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition, Neural Networks (Publication).
[6] Carpenter, G.A. & Grossberg, S. (1990), ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures, Neural Networks (Publication)
[7] Carpenter, G.A., Grossberg, S., & Rosen, D.B. (1991b), Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks (Publication)
[8] Carpenter, G.A., Grossberg, S., & Reynolds, J.H. (1991), ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, Neural Networks (Publication)
[9] Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., & Rosen, D.B. (1992), Fuzzy ARTMAP: A neural network architecture.
[10] ftp://ftp.fas.sfu.ca/pub/cs/han/kdd
[11] http://www.data-miner.com/
[12] http://www.cs.purdue.edu/homes/ayg/CS590D/resources.html
[13] http://www.kd1.com/
[14] Pieter Adriaans, Dolf Zantinge - Data Mining – Addison Wesley Longman – 1999
[15] Arun K Pujari - Data mining techniques - Universities Press – 2001
[16] Michael J. A. Berry, Gordon S. Linoff – Mastering Data Mining – Wiley Computer Publishing – 2001
[17] Rhonda Delmater, Monte Hancock – Data Mining Explained – Butterworth-Heinemann – 2001
[18] http://www.revue-texto.net/Reperes/Biblios/Forest_Biblio.html
[19] http://cns-web.bu.edu/~steve/

Sandhia Valsala, is presently associated with AMA International University, Bahrain as Asst Professor in the Computer Science Department. She holds a Master's degree in Computer Applications from Bharatiyar University, Coimbatore and is currently pursuing her Phd from Karpagam University Coimbatore.

Jissy Ann George is presently associated with AMA International University, Bahrain as Asst Professor in the Computer Science Department. She holds a Master's degree in Information Technology from Alagappa University, Coimbatore .

Priyanka parvathy is presently associated with AMA International University, Bahrain as Asst Professor in the Computer Science Department. She holds a Master's degree in VLSI/CAD, Manipal Academy of Higher Education.