CHAPTER 2

BACKGROUND STUDY

2.1 CARDIOTOCOGRAPHY

Cardiotocography (CTG) is a technical means of recording (-graphy) the fetal heartbeat (cardio-) and the uterine contractions (-toco-) during pregnancy, typically in the third trimester. The machine used to perform the monitoring is called a cardiotocograph, more commonly known as an electronic fetal monitor (EFM).

Figure 2.1 Cardiotocography
Simultaneous recordings are performed by two separate transducers, one for the measurement of the fetal heart rate and a second one for the uterine contractions. Each of the transducers may be either external or internal. External measurement means taping or strapping the two sensors to the abdominal wall. Internal measurement requires a certain degree of cervical dilatation.

Figure 2.2 Schematic explanation of cardiotocography

Heart rate (A) is calculated from fetal heart motion determined by ultrasound, and uterine contractions are measured by a tocodynamometer (B). These numbers are represented on a time scale with the help of a running piece of paper, producing a graphical representation.

2.1.1 Internal Method

Internal measurement of FHR involves attaching a scalp electrode (or spiral electrode) to the fetal head to adequately measure the electric
activity of the fetal heart. This measures the R-R interval on a fetal electrocardiogram. An automatic analysis of the ST waveform of the fetal electrocardiogram (STAN). "STAN", which includes analysis of the ST segment, is available on some equipment. This involves inserting a pressure catheter into the uterine cavity. Internal measurement is more precise, and might be preferable when a complicated childbirth is expected. Internal measurement is required to calculate Montevideo units.

2.1.2 External Method

The heart (cardio) sensor is an ultrasonic sensor, similar to a Doppler fetal monitor that continuously emits ultrasound and detects motion of the fetal heart by the characteristic of the reflected sound. The pressure-sensitive contraction transducer, called a tocodynamometer (TOCO) has a flat area that is fixated to the skin by a band around the belly. The pressure required to flatten a section of the wall correlates with the internal pressure, thereby providing an estimate of it.

2.1.3 CTG Observations

Cardiotocography (CTG) is a record of the Fetal Heart Rate (FHR) either measured from a transducer on the abdomen or a probe on the fetal scalp. In addition to the fetal heart rate another transducer measures the uterine contractions over the funds (Xiaojun et al 2012).

The CTG trace generally shows two lines. The upper line is a record of the fetal heart rate (Beats per Minute). The lower line is a recording of uterine contractions from the TOCO.

**Baseline Rate:** This should be between 110 and 150 Beats Per Minute (BPM) and is indicated by the FHR when stable (with accelerations
and decelerations absent). It should be taken over a period of 5 - 10 minutes. The rate may change over a period of time but normally remains fairly constant (Diogo et al 2005, Xiaojun et al 2012).

This is a section of CTG showing a typical normal baseline rate (Aggarwal et al 2012).

**Figure 2.3 Normal Baseline rate of CTG**

Baseline variability: The amount in Beats per Minute (BPM) by which the baseline varies

- **Type 1**  
  < 5 BPM  
  Abnormal

- **Type 2**  
  5 – 10 BPM  
  Normal Sleep

- **Type 3**  
  10 – 25 BPM  
  Normal Active

- **Type 4**  
  > 25 BPM  
  Abnormal
**Bradycardia:** If between 110 and 100 BPM it is suspicious whereas below 100 BPM it is pathological. A steep sustained decrease in rate is indicative of fetal distress and if the cause cannot be reversed the fetus should be delivered (Shahad Nidhal et al 2010)

**Figure 2.4 Bradycardia of CTG**

**Tachycardia:** A suspicious tachycardia is defined as being between 150 and 170 BPM whereas a pathological pattern is above 170 BPM. Tachycardias can be indicative of fever or fetal infection and occasionally fetal distress (with other abnormalities). An epidural may also induce a tachycardia in the fetus (Shahad Nidhal et al 2010).

This is a section of CTG showing a tachycardia (Shahad Nidhal et al 2010).
Baseline variations: The short term variations in the baseline should be between 10 and 15 BPM (except during intervals of fetal sleep which should be no longer than 60 minutes). Prolonged reduced variability along with other abnormalities may be indicative of fetal distress (Stirrat et al 2003). This is a section of CTG showing decreased baseline variability (Shahad Nidhal et al 2010).
Accelerations: This is defined as a transient increase in heart rate of greater than 15BPM for at least 15 seconds. Two accelerations in 20 minutes are considered a reactive trace. Accelerations are a good sign as they show fetal responsiveness and the integrity of the mechanisms controlling the heart (Stirrat et al 2003).

This section of CTG shows a typical acceleration in response to stimulus (Shahad Nidhal et al 2010).

![Acceleration showing a transient increase of greater than 15 BPM](image)

**Figure 2.7 Acceleration in response to stimulus of CTG**

Decelerations: These may either be normal or pathological. They are normally perfectly benign. Late decelerations persist after the contraction has finished and suggest fetal distress. Variable decelerations vary with each other in timings and shape and may be indicative of hypoxia or cord compression (Stirrat et al 2003).

Declaration = transient fall in baseline rate > 15 BPM lasting more than 15 seconds
**Type 1 (Early):** Synchronous with uterine contraction, the nadir of the heart rate trace corresponds to the peak of the uterine contraction. Uniform, repetitive, periodic slowing of FHR with onset early in the contraction and return to baseline at the end of the contraction are usually due to fetal head compression and therefore occur in first and second stage labour with descent of the head. It may be due to head compression, cord compression or early hypoxia.

**Type 2 (Late):** synchronous with uterine contraction, the nadir of the heart rate trace occurs after the peak of the uterine contraction. Uniform, repetitive, slowing of FHR with onset mid to end of the contraction and nadir more than 20 seconds after the peak of the contraction and ending after the contraction. The greater the lag time the more serious the significance. The worst picture is of shallow late decelerations, loss of baseline irregularity and tachycardia. A fetal pH measurement is mandatory.

**Type 3 (Variable):** Deceleration unrelated to uterine contractions. Variable, repetitive, periodic slowing of FHR with rapid onset and recovery. Time relationships with contraction cycles are variable and they may occur in isolation. Sometimes they resemble other types of deceleration patterns in timing and shape. If they appear consistently, fetal hypoxia is likely. Check fetal pH if the pattern persists after turning the patient on her side (or if other adverse features are present).

A normal CTG is a good sign but a poor CTG does not always suggest fetal distress. A more definitive diagnosis may be made from fetal blood sampling but if this is not possible or there is an acute situation (such as a prolonged bradycardia) intervention may be indicated. Prolonged deceleration of <100 BPM for 3 minutes or 80 BPM for 2 minutes
The following CTGs show examples of early, late and variable decelerations (Stirrat et al 2003).

**Figure 2.8 Early Decelerations of CTG**

![Early Decelerations of CTG](image)

**Figure 2.9 Late Decelerations of CTG**

![Late Decelerations of CTG](image)
2.2 DATA MINING OR KNOWLEDGE DISCOVERY IN DATABASES

Data Mining or Knowledge Discovery in Databases (KDD) is the nontrivial extraction of implicit, previously unknown, and useful information from data. Data mining can be defined as “A decision support process in which information patterns in data are searched”. Data mining uses sophisticated statistical analysis and modeling techniques to find patterns and relationships hidden in organizational databases. Once found, the information needs to be presented in a suitable form, with graphs, reports etc. Data Mining includes a number of different technical approaches for extraction of information such as clustering, data summarization, learning classification rules, finding dependency networks, analyzing changes, and detecting anomalies. Basically data mining is concerned with the analysis of data and the use of software techniques for finding patterns and regularities in data sets.
2.2.1 Data Mining Process

The data mining process can be divided into four major steps.

**Data Selection:** The target subset of data and the attributes of interest are identified by examining the entire raw dataset. This includes selecting or segmenting the data according to some criteria e.g. all those people who own a car.

**Data Cleaning:** In this step, noise and outliers are removed, field values are transformed to common units and some new fields are created by combining existing fields to facilitate analysis. The data is typically put into a relational format, and several tables might be combined in a denormalization step. Also the data is reconfigured to ensure a consistent format as there are possibilities of inconsistent formats because the data is drawn from several sources e.g. sex may record as f or m and also as 1 or 0.

**Data Mining:** This stage is concerned with the extraction of patterns from the data. Data mining algorithms can be applied to extract the interesting patterns of data.

**Interpretation & Evaluation:** The patterns identified by the system are interpreted into knowledge which can then be used to support human decision-making e.g. prediction and classification tasks, summarizing the contents of a database or explaining observed phenomena. The patterns are presented to end users.

2.2.2 Data Mining Functions

Data mining methods may be classified by the function they perform or according to the class of application they can be used in. The data mining functions are as follows.
**Classification:** The clustering techniques analyze a set of data and generate a set of grouping rules that can be used to classify future data. The mining tool automatically identifies the clusters, by studying the pattern in the training data. Once the clusters are generated, classification can be used to identify, to which particular cluster the input belongs. For example, one may classify diseases and provide the symptoms, which describe each class or subclass.

**Associations:** Given a collection of items and a set of records, each of which contain some number of items from the given collection, an association function is an operation against this set of records which return patterns that exist among the collection of items. These patterns can be expressed by rules such as "72% of all the records that contain items A, B and C also contain items D and E." The specific percentage of occurrences (in this case 72) is called the confidence factor of the rule. Also, A, B and C are said to be on an opposite side of the rule to D and E. Associations can involve any number of items on either side of the rule.

A typical application that can be built using an association function is Market Basket Analysis. Thus, by invoking an association function, the market basket analysis application can determine affinities such as "20% of the time that a specific brand toaster is sold, customers also buy a set of kitchen gloves and matching cover sets."

**Sequential / Temporal patterns:** Sequential/temporal pattern functions analyze a collection of records over a period of time for example to identify trends. The identity of a customer who made a purchase is known. An analysis can be made on the collection of related records of the same structure. Sequential pattern mining functions can be used to detect the set of customers associated with some frequent buying patterns. For example a set of insurance claims can lead to the identification of frequently occurring
sequences of medical procedures applied to patients which can help identify good medical practices as well as detect some medical insurance fraud.

**Clustering/Segmentation:** Clustering and segmentation are the processes of creating a partition so that all members of each partition set are similar according to some measure. A cluster is a set of objects grouped together because of their similarity or proximity. When learning is unsupervised then the system has to discover its own classes i.e. the system clusters the data in the database. The cluster can be formed by using the rules or functions.

### 2.3 DATA MINING TECHNIQUES

The data mining techniques are as follows:

**Cluster Analysis:** In an unsupervised learning environment the system has to discover its own classes. The data in the database can cluster as shown in the Figure 2.7. The first step is to discover subsets of related objects and then find descriptions e.g. D1, D2, D3 etc. which describe each of these subsets.

![Figure 2.11 Discovering clusters and descriptions in a database](image)

*Figure 2.11 Discovering clusters and descriptions in a database*
Clustering and segmentation basically partition the database so that each partition or group is similar according to some criteria. Clustering/segmentation in databases are the processes of separating a data set into components that reflect a consistent pattern of behavior. Once the patterns have been established they can then be used to "deconstruct" data into more understandable subsets and also they provide sub-groups of a population for further analysis or action, which are important when dealing with very large databases.

**Induction:** Induction is the inference technique, which can be used to infer the generalized information from the database. Induction has been used in the following ways within data mining.

**Decision trees:** Decision trees are simple knowledge representations and they classify examples to a finite number of classes. The nodes are labeled with attribute names, the edges are labeled with possible values for this attribute and the leaves labeled with different classes. Objects are classified by following a path down the tree, by taking the edges, corresponding to the values of the attributes in an object.

The following is an example of objects that describe the weather at a given time. The objects contain information on the outlook, humidity etc. Some objects are positive examples denoted by P and others are negative i.e. N. Classification in this case is the construction of a tree structure, illustrated in the Figure 2.8 which can be used to classify all the objects correctly.
Rule induction: A data mine system has to infer a model from the database that is it may define classes such that the database contains one or more attributes that denote the class of a tuple which are the predicted attributes while the remaining attributes are the predicting attributes. Class can then be defined by condition on the attributes. When the classes are defined, the system should be able to infer the rules that govern classification. Production rules have been widely used to represent knowledge in expert systems and they have the advantage of being easily interpreted by human experts because of their modularity i.e. a single rule can be understood in isolation and doesn't need reference to other rules. The structure of such rules is in the form of “if-then rules”.

Neural networks: Neural networks are an approach to computing that involves developing mathematical structures with the ability to learn. Neural networks can derive meaning from complicated or imprecise data and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Neural networks identify patterns or trends in data. They are good for prediction or forecasting. Neural networks use a set of
processing elements (or nodes) analogous to neurons in the human brain. These processing elements are interconnected in a network that can then identify patterns in data once it is exposed to the data, i.e., the network learns from experience just as people do. This distinguishes neural networks from traditional computing programs that simply follow instructions in a fixed sequential order. The structure of a neural network is shown in the Figure 2.13.

![Figure 2.13 Structure of a neural network](image.png)

The first layer represents the Input layer, in this case 3 input labels. The middle layer is called the hidden layer, with a variable number of nodes. The output layer in this case has two nodes representing output values are trying to determine from the inputs. Neural networks suffered from long learning times which become worse as the volume of data grows.

**Data Visualization:** Data visualization makes it possible for the analyst to gain a deeper, more intuitive understanding of the data and can work well for data mining. Data mining allows the analyst to focus on certain patterns and trends and explore in-depth using visualization. The data visualization can be overwhelmed by the volume of data in a database but in conjunction with data mining, it can help with exploration.
### 2.4 PROBLEMS FACED IN DATA MINING

The problems with data mining are as follows:

1. **Limited Information:** If some attributes essential to knowledge of the application domain are not present in the data it is impossible to discover significant knowledge about a given domain.

2. **Noise and missing values:** Errors in either the values of attributes or class information are known as noise. By using Bayesian techniques the corresponding records of missing data or average over the missing values have to omit.

3. **Uncertainty:** Uncertainty refers to the severity of the error and the degree of noise in the data.

4. **Size, updates, and irrelevant fields:** Databases are large and dynamic and their contents are changing as information is added, modified or removed. So, it is difficult to ensure that the rules are up-to-date and consistent with the most current information.

### 2.5 APPLICATIONS OF DATA MINING

Data mining has many and varied fields of application, some of which are listed below.

**Marketing:** Identify buying patterns from customers & Market basket analysis.
Banking: Detect patterns of fraudulent credit card use & Identify 'loyal' customers.

Insurance and Health Care: Claims analysis, Predict which customers will buy new policies & Identify fraudulent behavior.

Transportation: Determine the distribution schedules & Analyze loading patterns.

Application Areas

The application areas of data mining are very widely prevalent. As Bill Palace (1996) described “It primarily used today by companies with a strong consumer focus - retail, financial, communication, and marketing organizations. It enables these companies to determine relationships among "internal" factors such as price, product positioning, or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And, it enables them to determine the impact on sales, customer satisfaction, and corporate profits. Finally, it enables them to "drill down" into summary information to view detail transactional data.”

Those application areas of data mining included

Marketing

Data mining tools are able to provide continually to track customer satisfaction and feedback and be proactive in retaining the best customers and increasing their lifetime value, develop and deliver market-driven products by conducting ongoing surveys of the target customers, optimize marketing efforts, targeting the right customer segments with appropriate offers delivered through the right channel, and support all Customer Relationship Management (CRM) initiatives
Finance

A great challenge for commercial and investment institutes financial management is the risk management. Data mining tools can provide the quantitative analysts to guide them to better trades and sales of higher-margin products to their corporate customers.

Banking & Insurance

For banking system, data mining techniques can be used to detect patterns of fraudulent credit card use, identify 'loyal' customers, predict customers likely to change their credit card affiliation, determine credit card spending by customer groups, find hidden correlations between different financial indicators, and identify stock trading rules from historical market data. For insurance system, data mining techniques can do the claims analysis, for example, which medical procedures are claimed together; do prediction task on customers’ new buying policies; identify behaviour patterns of risky customers; and identify fraudulent behaviour.

Bioinformatics

Data mining techniques can be used for extracting explicit information from bioinformatics data, express this information as logical rules and decision trees, apply these new procedures to a range of scientific problems related to bioinformatics and chemistry-informatics.

Internet

With the explosive growth of information sources available on the World Wide Web, it has become increasingly necessary for users to utilize automated tools in order to find, extract, filter, and evaluate the desired information and resources. Thus, a technology term named web mining which
is under the field data mining is used to solve those problems. Web mining technology gives rise to the necessity of creating server-side and client-side intelligent systems that can effectively mine for knowledge, both across the Internet and in particular Web localities.

2.6 FUTURE TRENDS OF DM

Data Mining and KDD technology had brought many new developments, methods, and technologies in the recent decade. Also the improvement of integration of techniques and the application of data mining techniques had contributed in handling of new kinds of data types and applications. However, the field of data mining is still young enough that the possibilities are still limitless. Han et al (2001) indicated: “The field of data mining and knowledge discovery in databases (KDD) has been growing in leaps and bounds, and has shown great potential for the future”.

One important reason for the rapid growth in this field is the free internet data mining resource. Those resources included articles, journal, toolkit and raw data. There are two famous website for data mining researchers to share their research achievements (KD nuggets and Kluweronline).

The tomorrow of Data Mining and KDD will be bright and promising. Its expanding applications can be used to integrate several technologies and methods, such as investigating the space resource, broadening its applicability to business applications, and making programs and interfaces easier for end-users to use. There is no doubt that data mining will become one of the key technology areas in this century.
2.6.1 The Preprocessing Tasks

The preprocessing tasks used to clean the noisy data or any other unwanted things within the data.

2.6.1.1 Preparing the Mining Base

The initial mining base will often have to be created by integrating data from different data sources. This task can range from a simple joining of tables within a relational database system to sophisticated data fusion.

In order to be able to, even, simply combine data from different data tables, the coding and structuring of analogous data fields must be matchable. Often it is not trivial how to match data from different sources – the definition of what comprises a case might differ, the domain of values for a feature might differ between different sources etc.

The actual steps that have to be taken to integrate data from different sources will vary widely from application to application and will not be discussed here. The final output of this preprocessing task should be a mining base that contains all the data (but no more) that is considered necessary for the data mining to succeed. The data should be in a format that is supported as a central format by the data mining system.

The central data format of a data mining system should be capable of coding all the information that is considered relevant and that is supported by at least one of the utilized data mining algorithms. It will be then the job of preprocessing tasks to provide each algorithm with just the information it can handle.
Some types of information that might well be coded in the mining base but are not supported by all algorithms are:

**Numeric/symbolic/binary features:** some algorithms will not be able to handle all of these formats directly, but of course the full original information for those algorithms that can want to preserve.

**Non-value indicators:** the original feature might contain several different non-value indicators, the most important of which are missing value indicators. E.g. several different missing value indicators might code the cause or other background information of why a value is missing. Non-value indicators might also indicate that a value is not applicable for that case – this might occur when a table contains a field that only makes sense for some of the cases.

**Censored values (values only known to be greater/less than some value):** while quite common with lifetime analyses, currently not supported by most machine learning algorithms.

### 2.6.1.2 Metadata

An important part of the mining base is metadata and background knowledge that describes the data stored in the mining base. As with the actual data, the meta data base should at least contain all the meta data that can be utilized with any of the available algorithms. But since metadata will also be utilized by the person carrying out or observing the data mining task, it might also be advisable not to restrict the metadata to just this set but include as much information as possible.
Some of the most important meta data are:

- A textual description of what symbolic values actually mean. When symbolic values are numerically coded, the symbolic values themselves are typically never used.

- Possible ranges of numeric features (as opposed to actual ranges observed in the data base).

- Scale information: nominal, ordinal, interval, cardinal.

- Ordering information for ordinal values (especially when symbolically coded).

- Documentation of what value is “active” for symbolically coded binary data.

- Documentation of the meaning of all non-value indicators.

- Abstraction hierarchies for symbolic values.

In addition, background a-priori information about distributions will be useful, e.g. knowledge of the true distribution of values if the mining base represents a sample that causes the observed distributions to be biased.

**2.6.1.3 Data Recoding**

Data recoding does not change the amount of information in the data base. While one could think of several ways to recode data, only consider the following tasks to be useful:

- Recoding of nominal features with n possible values into n (or n-1) binary indicator values: this is often essential for neural nets, many statistical methods (regression, k-NN), but can also
be useful for recoding the target variable for learning algorithms that can only learn binary classifications.

- Recoding of ordinal features with n possible (symbolic and ordered) values into rank numbers 1 to n or 0 to n-1.

- Inverting the above transformations (this is usually a cosmetical step to make the values more readable by replacing them with their labels or descriptions).

More complex recoding steps (e.g. splitting a field containing symbolic values that actually describe several characteristics of a case at once – this situation is often encountered as a result of bad database design) can be seen as special cases of more complex attribute transformation functions as discussed in section 2.7.

2.6.1.4 Outlier detection

Ripley defines outliers to be ‘examples that did not come (or are thought not to have) from the assumed population of examples’. Usually this means that one or more features contain erroneous values, thus rendering the example ‘unlikely’.

Outliers should preferably be handled within the learning algorithm: an example that is not likely to belong to any of the possible classes could be designated as outlier. Another reason for preferring outlier detection within the learning algorithm is that erroneous values will only make an example an outlier if they occur in a feature that is relevant in the generated model. In reality, outliers cannot usually be explicitly handled by current learning algorithms and outlier detection and handling has to be done in a preprocessing step.
In some situations at least some outliers may be detectable by utilizing background knowledge about the data base (incorrect date formats, frequent typing errors, convenient (to the operator) default values for required entries, spelling errors, systematic measurement errors, information gained from previous learning steps). Sometimes, spotting of incorrect data can be aided by data visualization techniques and other more complex analysis techniques. In some cases, methods like the detection of association rules or partial determinations can pinpoint such errors when they either report “illegal” associations, or fail to find logically entailed relationships. This kind of knowledge about incorrect data can ultimately be formulated as conditions on the values of an example that must be true.

2.6.1.5 Handling of corrupt values

Two options remain to handle cases that are detected to be corrupt values: removing them or trying to fix them. This is very similar to how missing values can be treated in an individual preprocessing step. If corrupt values are replaced by missing value indicators, these two tasks can be handled by similar or even equal methods.

Removing: drop examples that contain corrupt values from the mining base.

Fixing: try to replace the corrupt values with correct ones

The second alternative bears the risk of introducing new incorrect values and letting the subsequent learning algorithm handle the examples as noise might usually be preferable.
2.6.1.6 Missing / non-value indicators

The most common non-value indicators are missing value indicators. Depending on the application and the target algorithm, these can be interpreted differently:

**Unsupported value:** the algorithm is not able to process missing value indicators. In that case, several possibilities of how to handle the situation

**Supported value:** the algorithm is able to handle missing value indicators and want to treat those indicators semantically as missing data

**‘Informative missing value’:** the missing value indicator actually tells us more than just the fact that data is missing, e.g. that somebody denied answering a question. In that case, it might be worthwhile to recode the additional information into a separate field to make it usable for further analysis.

As with corrupt values, can either try to replace missing values with a corrected value or remove all examples that contain missing values. In addition, can treat a missing value indicator as a special, additional value as a feature can assume.

2.6.1.7 Filtering out examples with missing values

This strategy will not introduce new wrong data but has the potential drawback of drastically reducing the amount of available data (especially if there are many examples where at least one feature contains a missing value) and changing the distribution of cases in the database.
2.6.1.8 Replacing missing data

In order to replace missing data, need to know what the correct value of some attribute should have been. Usually this information is not available. The only alternative is to utilize a model that tries to maximize the probability of substituting the correct value. No matter what model used, there must be aware that such replacement is error-prone and might actually worsen subsequent data analysis. Some simple strategies to replace incorrect or missing data values are:

- Replace symbolic values with the class-conditional or unconditional mode
- Replace numeric values with the mean, trimmed mean or median (again class-conditional or unconditional)

If the information about the process that was responsible for the introduction of missing values, it might be able to utilize a tailor-made model for replacing them.

More sophisticated methods try to take into consideration the density of the joint distribution, i.e. the conditional probabilities of the attribute’s values given all the known attributes of the case:

- The EM algorithm.
- Finding a predictive model for the attribute using one of the available learning algorithms.

Note that it might be reasonable to consider the confidence of the model when assigning a replacement for a missing value: when the
confidence is extremely low, the complete record should be removed from the data (this will typically be the case when many attributes in the record have missing values). Similarly, when the confidence for assigning a value to an attribute will be low for many records, it is advisable to drop the entire column (this can only happen if many values are missing for that attribute). Simplified strategies for these cases are to remove a record when the number of attributes with missing values exceeds some threshold and to remove an attribute when the number of missing values in the attribute exceeds some threshold. This simplified strategy can also be used as a preprocessing step for algorithms that can handle missing values.

As already stated above, treatment of incorrect or especially missing data might be easier and more effective within the analysis or learning algorithm, for example, the CART system introduced the interesting notion of surrogate attributes, whose values serve as a decision substitute when a value is missing. A surrogate attribute is the attribute most closely correlated to the attribute in question.

2.6.1.9 Treating Missing Value Indicators as Additional Values

This strategy might be reasonable when the missing value indicates some additional information, e.g. a denial to answer. It also will be necessary, when the number of missing values is extremely high. This can be the case when data table encodes sparse data, where measurements are only available for a small subset of all possible features for most cases.
2.7 KNOWLEDGE DISCOVERY PROCESS

Data mining is part of a larger iterative process called knowledge discovery. The following summarizes the steps of the knowledge discovery process.

Define the Problem. This initial step involves understanding the problem and figuring out what the goals and expectations are of the project.

Collect, clean, and prepare the data. This requires figuring out what data are needed, which data are most important and integrating the information. This step requires considerable effort, as much as 70% of the total data mining effort (Vlachos et al 2003).

2.7.1 Data Mining

This model-building step involves selecting data mining tools, transforming the data if the tool requires it, generating samples for training and testing the model, and finally using the tools to build and select a model.

2.7.2 Validate the Models

Test the model to ensure that it is producing accurate and adequate results.

2.7.3 Monitor the Model

Monitoring a model is necessary as with passing time, it will be necessary to revalidate the model to ensure that it is still meeting the requirements. A model that works today may not work tomorrow and it is therefore necessary to monitor the behavior of the model to ensure whether it is meeting performance standards.
2.8 THE DATA MINING PROCESS

The goal of identifying and utilizing information hidden in data has three requirements (Xiaojun et al 2012):

1. The captured data must be integrated into organization-wide views instead of specific views.
2. The information contained in the integrated data must be extracted.
3. The obtained information must be organized in ways that enable decision-making.

The data mining process can be classified as going through a series of four steps. This consists of transforming the already summarized data found in a data warehouse into information that can produce useful results. These four steps can be summarized into (Xiaojun et al 2012):

1. Data selection
2. Data transformation
3. Mining the Data
4. Interpretation of Results

Data selection consists of gathering the data for analysis. Data transformation will then convert appropriate data to a particular format. Data mining will then extract the desired type of information yielding in results to be interpreted. In Figure 2.10, the data-mining tool will extract the relevant information from the data warehouse environment. In order for the data-mining tool to work, the sub-processes of data selection and transformation must take place prior to data mining. The results are then passed to a
decision-oriented databases or data mart, where the user can make a recommendation based on the results and put the recommendations into action. Of course this assumes that all of the four steps will be successfully completed, which is not always the case.

Data selection can be the most important step in the process. This is due to the complexity in finding and constructing pre-selection criteria before the extraction of data actually transpires. The variables selected and the range of each of them should be determined in this step. For example, a marketing executive wishing to improve sales figures will pre-select those customers that have been most active in making purchases and observe their behavior. An executive can mine all the data, but this can turn out to be a very costly operation because the data-mining tool will have to search through all this data and moreover if results are generated, they have more risk in predicting an optimal recommendation. Carefully choosing the data is therefore a very important step.

Figure 2.14 Data Mining Process
Once the data to be mined has been chosen, the next step in the data mining process usually consists of transforming the data into the particular formats needed by the data-mining tool. Data are further synthesized by computing certain rations and applying algorithms to convert the data to a particular type suitable for future applied tools (Xiaojun et al 2012).

Once the data have been selected and required transformations done, a data-mining tool can now be applied. Specific predictions about futuristic events based on previous collected data can yield in significant hidden findings through the use of well designed algorithms, a topic of discussion in later sections. Using a data warehouse alongside with a mining tool is usually recommended as this allows for a more efficient organization of the collected data in ways that can facilitate and optimize analysis. Furthermore, the mining tool can also interface with a DSS for further interpretation of the data.

However, a data mining system need not always interact with a data warehouse, and in fact, data mining can still extract pertinent information if given raw data from a database. The main advantage of using a data warehouse is that most of the data are already in already integrated in a suitable format of choice making it easier for a data-mining tool to extract the higher quality information.

The final step in the data mining process consists of interpreting the results. Once the extracted information is analyzed and interpreted, the most relevant information can be passed onto the decision-maker through a DSS. Result interpretation can consist not only of interpreting the output but also of further filtering the data and passing that information to the decision support system. In the case that the interpreted results are not satisfactory, it may be necessary to repeat any of the previous steps until the information generated
contains the maximum added value to the data miner. As such, data mining is a very complex process. Many steps need to be performed correctly before feeding of data to the data mining tool. Furthermore it is not guaranteed that the data-mining tool will yield significant results in any steps of the mining process. Certainly, performing many trials are recommended as this can reveal error corrections in any of the four steps. Any of the previously mentioned steps can be modified to continue investigating the data and searching for hidden patterns. This is the challenge of the data mining organization and though it can be a painstaking process, the more data that is mined, the more likely the data miner will learn from the process. The use of tools such as DSS and a warehouse environment complement the data mining tools used to find useful facts buried in layers of data. To maximize the efficiency of data mining, both of these and other tools must provide high quality delivery information to the data-mining tool. The use of good complementary tools to sift through data along with a powerful data-mining tool should be part of a well designed environment (Xiaojun et al 2012).

2.9 THE ROLL OF CLUSTERING AND 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 (Klimesova et al 2010). Different kinds of algorithms are used to achieve either one of these tasks. However, in the overall KDD process, any mixture of these tasks may be called upon to achieve the desired results. The Steps involved in KDD are:

**Description tasks**: These tasks describe the data being mined and they are:
**Summarization:** To extract compact patterns that describes subsets of data. The methods used to achieve this task are Association Rule algorithms.

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

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

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

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

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

**Regression:** To predict results that is numeric continuous variables.

**2.9.1 Classification Process**

Classification is a two step process.

Step 1: Model construction, i.e., build the model from the training set

Step 2: Model usages, i.e., check the accuracy of the model and use it for classifying new data
2.9.2 Model Construction

- Model construction is building the model from the training set.
- Each tuple/sample is assumed to belong a predefined 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.

2.9.3 Model Usage

- The model usage may be used to classify future or unknown objects.
- It is used to estimate the accuracy of the model.
- The known class of a test tuple/sample is compared with the result given by the model accuracy.
- Accuracy rate = percentage of the tests tuples /samples correctly classified by the model.

2.9.4 Stages in a Clustering Task

Figure below represents the typical sequencing of clustering activity (Xiaojun et al 2012). A pattern (or feature vector) is a single data item that is used by clustering algorithm. It typically consists of a vector of d measurements (where d the dimensionality of the data):
The individual scalar components $x_i$ of a pattern $X$ are called features (or attributes). Pattern representation refers to the number of classes, the number of available patterns, the number, type and scale of the features available to clustering algorithms. Feature selection is the process of identifying the most effective subset of original features to use in clustering. Feature extraction is the use of one or more transformations of the input features to produce new salient features. Either or both of these techniques can be used to obtain what is called a feature set (or feature vector).

Pattern proximity is usually measured by a distance function defined on pairs of patterns. A variety of distance functions are in use in various communities. A simple distance measure can often be used to reflect dissimilarity between two patterns, where other similarity measures can be used to characterize the conceptual similarity between two patterns. The Euclidean distance metric can be defined as follows:

$$d_2(X_i, X_j) = \left( \sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2 \right)^{1/2}$$

$$= \| X_i - X_j \|$$

(2.2)

where $X_1$ and $X_2$ are two patterns. Euclidian distance metric works well when the data set has “compact” or “isolated” clusters.
The Grouping step represents the organization of patterns into clusters based on pattern similarity. There are many clustering methods available, and each of them may give a different grouping of a dataset. The choice of a particular method will depend on the type of output desired, the known performance of method with particular types of data, the hardware and software facilities available and the size of the dataset.

The taxonomy of clustering algorithms can be seen in the figure.

![Figure 2.16 Taxonomy of clustering approaches](image)

In general, clustering methods may be divided into two categories based on the cluster structure, which they produce. The non-hierarchical methods divide a dataset of N objects into M clusters, with or without overlap. These methods are sometimes divided into partitioning methods, in which the classes are mutually exclusive, and the less common clumping method, in which overlap is allowed. Each object is a member of the cluster with which it is most similar, however the threshold of similarity has to be defined. The hierarchical methods produce a set of nested clusters in which each pair of objects or clusters is progressively nested in a larger cluster until
only one cluster remains. The hierarchical methods can be further divided into agglomerative or divisive methods. In agglomerative methods, the hierarchy is built up in a series of N-1 agglomerations, or Fusion, of pairs of objects, beginning with the un-clustered dataset. The less common divisive methods begin with all objects in a single cluster and at each of N-1 steps divide some clusters into two smaller clusters, until each object resides in its own cluster.

**Agglomerative vs. Divisive**

This aspect relates to algorithmic structure and operation. An agglomerative approach begins with each pattern in a distinct (singleton) cluster, and successively merges clusters together until a stopping criterion is satisfied. A divisive method begins with all patterns in a single cluster and performs splitting until a stopping criterion is met.

**Monothetic vs. polythetic**

This aspect relates to the sequential or simultaneous use of features in the clustering process. Most algorithms are polythetic; that is, all features enter into the computation of distances between patterns, and decisions are based on those distances. A simple monothetic algorithm considers features sequentially to divide the given collection of patterns.

The major problem with this algorithm is that it generates $2^d$ clusters where $d$ is the dimensionality of the patterns. For large values of $d$ ($d>100$ is typical in information retrieval applications), the number of clusters generated by this algorithm is so large that the data set is divided into uninterestingly small and fragmented clusters.
**Hard vs. fuzzy**

A hard clustering algorithm allocates each pattern to a single cluster during its operation and in its output. A fuzzy clustering method assigns degrees of membership in several clusters to each input pattern. A fuzzy clustering can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership.

**Deterministic vs. Stochastic**

This issue is most relevant to partitional approaches designed to optimize a squared error function. This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labelings.

**Incremental vs. Non-incremental**

This issue arises when the pattern set to be clustered is large, and constraints on execution time or memory space affect the architecture of the algorithm (Yeh et al 1995). The early history of clustering methodology does not contain many examples of clustering algorithms designed to work with large data sets, but the advent of data mining has fostered the development of clustering algorithms that minimize the number of scans through the pattern set, reduce the number of patterns examined during execution, or reduce the size of data structures used in the algorithm’s operations. A cogent observation in is that the specification of an algorithm for clustering usually leaves considerable flexibility in implementation.
2.10 CLASSIFICATION ALGORITHMS

2.10.1 Classification-rule Learning

Classification-rule learning involves finding rules or decision trees that partition given data into predefined classes (Diogo et al 2005). 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.
2.10.2 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 (Adam et al 2004). 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.

2.10.3 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.

2.10.4 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 (Frawley et al 1992). 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.

2.10.5 ID3 algorithm

The ID3 algorithm (Quinlan86) is a decision tree building algorithm which determines the classification of objects by testing the values of the properties (Adam et al 2004). 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 sub tree 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.

2.10.6 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 (Thomson et al 2001). 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.

### 2.10.7 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 (Xiaojun et al 2012). 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 calculates entropy values of each distinct value 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 (Scalable PaRallelizable INduction of decision Trees) that for the removes all of the memory restrictions.
2.10.8 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 over fitting 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.

2.10.9 Naive 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 is used, it will refer as the naïve k-means algorithm as it is much simpler compared to the other algorithms described here.

The naïve 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 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, 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:

Step 1: 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.

Step 2: 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 are used, 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, then need a measure of convergence of the centroids so that, stop the iterations when the convergence criteria are 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, can measure the sum of Euclidean distance of the new centroids from the old centroids. In this thesis, 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 \( \phi \), 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 \( x \)). Then define a measure of quality for \( \phi \):

\[
\text{distortion } \phi = \frac{1}{N} \sum_x |x - \phi(x)|^2
\]  

(2.3)
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.

2.10.10 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:

- They are binary trees;
- The root node contains all the points;
- 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;
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. Omohundro 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), The two main approaches that try to counter the shortcomings of the k-means algorithm are discussed.

2.10.11 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 kth 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 $K' \leq K$. As use the above assumption, the global optima for $k = K'$ clusters are computed as a series of local searches. Assuming that the k-means clustering problem for $K' - 1$ clusters have solved, a new cluster at an appropriate location have to place. To discover the appropriate insertion location, which is not known, 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 $K' - 1$ cluster. 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, can compute the optimal positions for the $k = 2, 3, 4, \ldots K$, clusters. Thus the process involves computing the optimal k-means centers for each of the $K = 1, 2, 3\ldots K$ clusters. The algorithm is entirely deterministic.

Though the attractiveness of the global k-means lies in 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. 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.

2.10.12 Nearest-neighbour


Bremner et al (2005) presented in Output-sensitive algorithms for computing nearest-neighbor decision boundaries which classify the objects.

2.10.13 Naive-Bayes

It is a simple induction algorithm that assumes a conditional independence model of attributes given the label (Domingos et al 1996, Langley et al 1992, Duda et al 1973, Good 1965).
2.10.14 OODG

This refers to Oblivous read-Once Decision Graph induction algorithm described in Kohavi (1995c).

2.10.15 Lazy decision trees

It is an algorithm for building the “best” decision tree for every test instance described in Friedman et al (1996).

2.10.16 Decision Table

This is a simple lookup table. A simple algorithm that is useful with feature subset selection.

2.10.17 K-means

It is largely known and used, and represents a kind of standard. Churchill (1995) notes "currently, the most popular partitioning method is the k-means approach…” obviously its popularity comes from the fact that it has been established as the best partitioning method. Openshaw et al (1995) also argue that "conventional best practice is to employ a K-means nearest neighbour classifier to spatial census data that has been ortho-normalised". Schaffer and Green (1998) also point out "During the late 1980s, however, non-hierarchical methods, such as k-means, became dominant for segmenting the large data sets typically encountered in marketing."

The central idea in k-means is to start with an initial partition and then from there, by changing cluster membership, improve the partition. As Anderberg (1973) notes, "The broad concept for this methods (non-hierarchical clustering methods) is very similar to that underlying the
steepest descendent algorithms used for unconstrained optimisation in nonlinear programming”.

Typically the k-means algorithm starts with an initialization process in which seeds positions are defined. This initial step can have a significant impact on the performance of the method (Bradley and Fayyad 1998) and can be done in a number of ways (Anderber 1973, Bradley et al 1998). After the initial seed had been defined, each data element is assigned to the nearest seed. The next step consists in repositioning the seeds. This can be done after all elements are assigned to the nearest seeds or as each one of the elements is assigned. After this, a new assignment step is necessary and process will go on until no further improvement can be made, in other words a local optimum has been found. Considering that the assignments will be done on the basis of the distance to the nearest seed, implicitly this process will produce a minimization of the "sum of the L2 distance squared between each data point and its nearest cluster center" (Bradley et al 1998).

2.10.18 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."

2.10.19 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 al (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.

2.11 EVOLUTIONARY APPROACHES

Evolutionary approaches, motivated by natural evolution, make use of evolutionary operators and a population of solutions to obtain the globally optimal partition of the data (Diogo et al 2005, Shekhar et al 2003). Candidate solutions to the clustering problem are en-coded as chromosomes. The most commonly used evolutionary operators are: selection, recombination, and mutation. Each transforms one or more input chromosomes into one or more output chromosomes. A fitness function evaluated on a chromosome determines a chromosome’s likelihood of surviving into the next generation. A high-level description of an evolutionary algorithm applied to clustering is given below.

An Evolutionary Algorithm for Clustering

**Step 1:** Choose a random population of solutions. Each solution here corresponds to a valid k-partition of the data. Associate a fitness value with each solution. Typically, fitness is inversely proportional to the squared error value. A solution with a small squared error will have a larger fitness value.
**Step 2:** Use the evolutionary operators selection, recombination and mutation to generate the next population of solutions. Evaluate the fitness values of these solutions.

Repeat step 2 until some termination condition is satisfied.

The best-known evolutionary techniques are Genetic Algorithms (GAs), Evolution Strategies (ESs), and Evolutionary Programming (EP). Out of these three approaches, GAs has been most frequently used in clustering. Typically, solutions are binary strings in GAs. In GAs, a selection operator propagates solutions from the current generation to the next generation based on their fitness. Selection employs a probabilistic scheme so that solutions with higher fitness have a higher probability of getting reproduced.

There are a variety of recombination operators in use; crossover is the most popular. Crossover takes as input a pair of chromosomes (called parents) and outputs a new pair of chromosomes (called children or offspring).

GAs represent points in the search space as binary strings, and rely on the crossover operator to explore the search space. Mutation is used in GAs for the sake of completeness, that is, to make sure that no part of the search space is left unexplored. ESs and EP differ from the GAs in solution representation and type of the mutation operator used; EP does not use a recombination operator, but only selection and mutation. Each of these three approaches has been used to solve the clustering problem by viewing it as a minimization of the squared error criterion. Some of the theoretical issues such as the convergence of these approaches were studied (Cerioli et al 1999).

GAs performs a globalized search for solutions whereas most other clustering procedures perform a localized search. In a localized search, the
solution obtained at the ‘next iteration’ of the procedure is in the vicinity of the current solution. In this sense, the k-means algorithm, fuzzy clustering algorithms, ANNs used for clustering, various annealing schemes (see below), and tabu search are all localized search techniques. In the case of GAs, the crossover and mutation operators can produce new solutions that are completely different from the current ones.

Perhaps the earliest paper on the use of GAs for clustering is where a GA was used to minimize the squared error of a clustering. Here, each point or chromosome represents a partition of $N$ objects into $K$ clusters and is represented by a Kary string of length $N$. An improved representation scheme is proposed where an additional separator symbol is used along with the pattern labels to represent a partition. Using this representation it permits them to map the clustering problem into a permutation problem such as the traveling salesman problem, which can be solved by using the permutation crossover operators. This solution also suffers from permutation redundancy.

More recently, it investigated the use of edge-based crossover to solve the clustering problem. Here, all patterns in a cluster are assumed to form a complete graph by connecting them with edges. Offspring are generated from the parents so that they inherit the edges from their parents. In a hybrid approach proposed, the GA is used only to find good initial cluster centers and the k-means algorithm is applied to find the final partition. This hybrid approach performed better than the GA.

A major problem with GAs is their sensitivity to the selection of various parameters such as population size, crossover and mutation probabilities, etc. Grefenstette has studied this problem and suggested guidelines for selecting these control parameters. However, these guidelines may not yield good results on specific problems like pattern clustering. It was reported that hybrid genetic algorithms incorporating problem-specific
heuristics are good for clustering. A similar claim is made about the applicability of GAs to other practical problems. Another issue with GAs is the selection of an appropriate representation which is low in order and short in defining length.

It is possible to view the clustering problem as an optimization problem that locates the optimal centroids of the clusters directly rather than finding an optimal partition using a GA. This view permits the use of ESs and EP, because centroids can be coded easily in both these approaches, as they support the direct representation of a solution as a real-valued vector. ESs was used on both hard and fuzzy clustering problems and EP has been used to evolve fuzzy min-max clusters. It has been observed that they perform better than their classical counterparts, the k-means algorithm and the fuzzy c-means algorithm. However, all of these approaches suffer (as do GAs and ANNs) from sensitivity to control parameter selection. For each specific problem, one has to tune the parameter values to suit the application.

2.12 COMPARISON OF TECHNIQUES

In this section, various deterministic and stochastic search techniques to approach the clustering problem as an optimization problem have examined. A majority of these methods use the squared error criterion function. Hence, the partitions generated by these approaches are not as versatile as those generated by hierarchical algorithms. The clusters generated are typically hyperspherical in shape. Evolutionary approaches are globalized search techniques, whereas the rest of the approaches are localized search technique. ANNs and GAs are inherently parallel, so they can be implemented using parallel hardware to improve their speed.

Evolutionary approaches are population-based; that is, they search using more than one solution at a time, and the rest are based on using a
single solution at a time. ANNs, GAs, SA, and Tabu search (TS) are all sensitive to the selection of various learning / control parameters. In theory, all four of these methods are weak methods in that they do not use explicit domain knowledge. An important feature of the evolutionary approaches is that they can find the optimal solution even when the criterion function is discontinuous.

An empirical study of the performance of the following heuristics for clustering was presented in Mishra et al (1994) SA, GA, TS, Randomized Branch-and-Bound (RBB), and Hybrid Search (HS) strategies were evaluated. The conclusion was that GA performs well in the case of one-dimensional data, while its performance on high dimensional data sets is not impressive. The performance of SA is not attractive because it is very slow. RBA and TS performed best. HS is good for high dimensional data. However, none of the methods was found to be superior to others by a significant margin.

An empirical study of k-means, SA, TS, and GA was presented in Al-Sultan and Khan. TS, GA and SA were judged comparable in terms of solution quality, and all were better than k-means. However, the k-means method is the most efficient in terms of execution time; other schemes took more time (by a factor of 500 to 2500) to partition a data set of size 60 into 5 clusters. Further, GA encountered the best solution faster than TS and SA; SA took more time than TS to encounter the best solution. However, GA took the maximum time for convergence that is, to obtain a population of only the best solutions, followed by TS and SA. An important observation is that in both Mishra et al (1994) and Al-Sultan et al (1996) the sizes of the data sets considered are small; that is, fewer than 200 patterns.

A two-layer network was employed, with the first layer including a number of principal component analysis subnets, and the second layer using a
competitive net (Frawley et al 1992). This network performs partitional clustering using the regularized Mahalanobis distance. This net was trained using a set of 1000 randomly selected pixels from a large image and then used to classify every pixel in the image. Phanendra Babu et al (2000) proposed a stochastic connectionist approach (SCA) and compared its performance on standard data sets with both the SA and k-means algorithms. It was observed that SCA is superior to both SA and k-means in terms of solution quality. Evolutionary approaches are good only when the data size is less than 1000 and for low dimensional data.

In summary, only the k-means algorithm and its ANN equivalent, the Kohonen net have been applied on large data sets; other approaches have been tested, typically, on small data sets. This is because obtaining suitable learning/control parameters for ANNs, GAs, TS, and SA are difficult and their execution times are very high for large data sets. However, it has been shown that the k-means method converges to a locally optimal solution. This behavior is linked with the initial seed election in the k-means algorithm.

So, a good initial partition can be obtained quickly using any of the other techniques, and then k-means would work well even on problems with large data sets. Even though various methods discussed in this section are comparatively weak, it was revealed through experimental studies that combining domain knowledge would improve their performance. For example, ANNs work better in classifying images represented using extracted features than with raw images, and hybrid classifiers work better than ANNs. Similarly, using domain knowledge to hybridize a GA improves its performance. So it may be useful in general to use domain knowledge along with approaches like GA, SA, ANN, and TS. However, these approaches (specifically, the criteria functions used in them) have a tendency to generate
a partition of hyper-spherical clusters, and this could be a limitation. For example, in cluster-based document retrieval, it was observed that the hierarchical algorithms performed better than the partitional algorithms.

2.13 OUTLIER DETECTION ALGORITHMS

Outlier detection has been extensively studied in the field of statistics, and a number of discordancy tests have been developed. Most of these studies treat outliers as “noise” and they try to eliminate the effects of outliers by removing outliers or develop some outlier-resistant methods.

However, in data mining, can consider outliers “meaningful input signals” rather than “noise”. In some cases, outliers represent unique characteristics of the objects, which are important to an organization. Law enforcement is one area where outlier detection is critically important. In law enforcement, want to associate criminal incidents caused by the same person/group and detect outliers from this behavior.

Outlier detection in spatial data has its roots in traditional data mining schemes, where the goal is to discover useful and notable patterns in the data. The primary difference between the two is that in traditional outlier detection, the information being mined are typically transactions, or similarly structured data and in spatial outlier detection, objects and regions are the emphasis.

Tests for outliers have primarily been superseded by the use of robust methods. Outlier tests are poor in that outliers tend to damage results long before they are detected. Robust methods attempt to compensate rather than reject outliers.
The structure of outlier detection techniques can be broken down as shown in the following diagram.

![Diagram of Outlier Detection Techniques]

**Figure 2.17 Structure of Outlier Detection Techniques**

In fact, traditional (global) outlier detection and spatial outlier detection have a common process of operation (Stirrat et al 2003). By first determining the classification scheme based on locality and how these interact, then can discover inconsistencies based on these findings. The main difference is that spatial outlier detection makes use of locality in defining the neighborhood of interest whereas global outlier detection treats this as just another variable.

The significance of incorporating spatial elements into the process is that while in traditional methods, finding patterns was the relatively simple task of comparing variables and common collections of these, spatial data has the general tendency of not being easily quantified. To work around this trait, can define both a region as belonging within an imposed boundary and spatial proximity as whether a common edge is shared between two boundaries. These independent regions could be assigned a particular code given the predominance of objects within the region, but this does not adequately explain variances on objects within said region.
For such a case, outlier detection algorithms are invaluable for discovering anomalous and unexpected behavior among spatial objects. In most instances, readings from remote sensors serve as the inputs to these algorithms and over time, changes in the behavior of either the sensors themselves or the data they monitor can be discovered. However, not all outlier detection algorithms are made equal, and methods that seemed feasible with lower-dimensional spatial data do not scale well to higher-dimensional data. This section will focus on explaining how to handle outlier detection in low-dimensional data and the changes that need to be made to when the data scales up to higher dimensionality.

2.13.1 Input Data of Outlier deduction

A key component of any outlier detection technique is the input data in which it has to detect the outliers. Input is generally treated as a collection of data objects or data instances Tan et al (2005). Each data instance can be described using a set of attributes (also referred to as variable, characteristic, feature, field, or dimension). The data instances can be of different types such as binary, categorical or continuous. Each data instance might consist of only one attribute (univariate) or multiple attributes (multivariate). In the case of multivariate data instances, all attributes might be of same type or might be a mixture of different data types. One important observation here is that the features used by any outlier detection technique do not necessarily refer to the observable features in the given data set.

Several techniques use preprocessing schemes like feature extraction (Addison et al 1999), or construct more complex features from the observed features (Ertoz et al 2004), and work with a set of features which are most likely to discriminate between the normal and outlying behaviors in the data. A key challenge for any outlier detection technique is to identify a best
set of features which can allow the algorithm to give the best results in terms of accuracy as well as computational efficiency.

Input data can also be categorized based on the structure present among the data instances. Most of the existing outlier detection algorithms deal with data in which no structure is assumed among the data instances. Then such data are referred as point data. Typical algorithms dealing with such data sets are found in network intrusion detection domain (Ertoz et al.2004) or in medical records outlier detection domain (Laurikkala et al. 2000). Data can also have a spatial, sequential or both types of structures. For sequential data, the data instances have an ordering defined such that every data instance occurs sequentially in the entire data set. Time-series data is the most popular example for this case and has been extensively analyzed with respect to outlier detection in statistics (Abraham et al 1989, Abraham et al 1997). Recently, biological data domains such as genome sequences and protein sequences (Eisen et al 1998, Teng 2003) have been explored for outlier detection. For spatial data, the data instances have a well defined spatial structure such that the location of a data instance with respect to others is significant and is typically well defined. Spatial data is popular in traffic analysis domain (Shekhar et al 2001) and ecological and census studies (Kou et al 2006). Often, the data instances might also have a temporal (sequential) component giving rise to another category of spatio-temporal data, which is widely prevalent in climate data analysis (Blender et al 1997). Later in this section, will discuss the situations where the structure in data becomes relevant for outlier detection.

2.13.2 Type of Supervision

Besides the input data (or observations), an outlier detection algorithm might also have some additional information at its disposal. A labeled training data set is one such information which has been used
extensively (primarily by outlier detection techniques based on concepts from machine learning (Mitchell 1997) and statistical learning theory (Vapnik 1995)). A training data set is required by techniques which involve building an explicit predictive model. The labels associated with a data instance denote if that instance is normal or outlier. Based on the extent to which these labels are utilized, outlier detection techniques can be divided into three categories.

### 2.13.3 Supervised Outlier Detection Techniques

Such techniques assume the availability of a training data set which has labeled instances for normal as well as outlier class. Typical approach in such case is to build predictive models for both normal and outlier classes. Any unseen data instance is compared against the two models to determine which class it belongs to. Supervised outlier detection techniques have an explicit notion of the normal and outlier behavior and hence accurate models can be built. One drawback here is that accurately labeled training data might be prohibitively expensive to obtain. Labeling is often done manually by a human expert and hence requires a lot of effort to obtain the labeled training data set. Certain techniques inject artificial outliers in a normal data set to obtain a fully labeled training data set and then apply supervised outlier detection techniques to detect outliers in test data (Abe et al 2006).

### 2.13.4 Semi-Supervised Outlier Detection Techniques

Semi-Supervised outlier detection techniques assume the availability of labeled instances for only one class. It is often difficult to collect labels for other class. For example, in space craft fault detection, an outlier scenario would signify an accident, which is not easy to model. The typical approach of such techniques is to model only the available class and declare any test instance which does not fit this model to belong to the other class. Techniques that assume availability of only the outlier instances for
training are not very popular. The primary reason for their limited popularity is that it is difficult to obtain a training data set which covers every possible outlying behavior that can occur in the data. The behaviors which do not exist in the training data will be harder to detect as outliers. Dasgupta et al (2000, 2002) have used only outlier instances for training. Similar semi-supervised techniques have also been applied for system call intrusion detection (Forrest et al 1996). On the other hand, techniques which model only the normal instances during training are more popular. Normal instances are relatively easy to obtain. Moreover, normal behavior is typically well-defined and hence it is easier to construct representative models for normal behavior from the training data. This setting is very similar to as novelty detection (Markou and Singh 2003a, 2003b) and is extensively used in damage and fault detection.

### 2.13.5 Unsupervised Outlier Detection Technique

The third category of techniques does not make any assumption about the availability of labeled training data. Thus these techniques are most widely applicable. The techniques in this category make other assumptions about the data. For example, parametric statistical techniques, assume a parametric distribution of one or both classes of instances.

Similarly, several techniques make the basic assumption that normal instances are far more frequent than outliers. Thus a frequently occurring pattern is typically considered normal while a rare occurrence is an outlier (Varun et al 2010). The unsupervised techniques typically suffer from higher false alarm rate, because often times the underlying assumptions do not hold true. Availability of labels governs the above choice of operating modes for any technique. Typically, semi-supervised detection and unsupervised modes have been adopted more. Generally speaking, techniques which assume availability of outlier instances in training are not very popular. One
of the reasons is that getting a labeled set of outlying data instances which cover all possible type of outlying behavior is difficult. Moreover, the outlying behavior is often dynamic in nature (for e.g - new types of outliers might arise, for which there is no labeled training data). In certain cases, such as air traffic safety, outlying instances would translate to airline accidents and hence will be very rare. Hence in such domains unsupervised or semi-supervised techniques with normal labels for training are preferred.

2.13.6 Output of Outlier Detection

The nature of outliers discussed above imposes a requirement on the structure of the outlying patterns detected by the technique. Another requirement for any outlier detection technique is the manner in which the outliers are reported. Typically, outlier detection techniques fall into one of the following two categories (Varun et al 2010).

Labeling Techniques

The techniques in this category assign a label (normal or outlier) to each test instance. Thus they behave like a classification algorithm in this respect. If the test input is a set of instances, the technique provides a set of outliers and a set of normal instances. The benefit of such techniques is that they provide an exact set of outliers for the analysts. The drawback of these techniques is that they do not differentiate between different outliers; no ranking among the outliers is provided. Often times, there is a confidence associated with a pattern being an outlier, and in such cases a zero-one decision is not feasible. This motivates the need for the scoring type of techniques discussed below (Varun et al 2010).
Scoring Techniques

These techniques assign an outlier score to each pattern depending on the degree to which that pattern is considered an outlier. Thus the output of such techniques is a ranked list of outliers. An analyst may choose to either analyze top few outliers or use a cut-off threshold to select the outliers. The drawback of a ranked list of outliers is the choice of the threshold to select a set of outliers. Often times choosing this threshold is not straightforward and has to be arbitrarily fixed. Besides defining the nature of data and outliers, the application domain can also impose certain constraints such as desired degree of accuracy and computational efficiency (Varun et al 2010). In domains such as safety-critical systems, the accuracy of the algorithm is a foremost requirement. On the other hand, online systems such network intrusion detection systems require the algorithms to be efficient and scalable. Often times, the techniques have to maintain a balance between the accuracy and efficiency aspects of the solution. Recently privacy preservation of data has also become an important constraint in several domains and outlier detection algorithms have to address this problem (Vaidya et al 2004).

2.14 MACHINE LEARNING TECHNIQUES

2.14.1 Learning Paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. Usually any given type of network architecture can be employed in any of those tasks.

2.14.2 Supervised Learning

In supervised learning, a set of example pairs and the aim is to find a function f in the allowed class of functions that matches the examples are
given. In other words, there is a wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error which tries to minimise the average error between the network’s output, \( f(x) \), and the target value \( y \) over all the example pairs. When one tries to minimise this cost using gradient descent for the class of neural networks called Multi-Layer Perceptrons, one obtains the well-known backpropagation algorithm for training neural networks. Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher," in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

### 2.14.3 Unsupervised Learning

In unsupervised learning, some data \( x \), and the cost function to be minimised can be any function of the data \( x \) and the network's output, \( f \) are given. The cost function is dependent on the task (what are trying to model) and our a priori assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model \( f(x) = a \), where \( a \) is a constant and the cost \( C = (E[x] - f(x))^2 \). Minimizing this cost will give us a value of \( a \), that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: For example in compression it could be related to the mutual information between \( x \) and \( y \). In statistical modelling, it could be related to the posterior probability of the
model given the data. (Note that in both of those examples those quantities would be maximized rather than minimised) Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

2.14.4 Reinforcement Learning

In reinforcement learning, data \( x \) is usually not given, but generated by an agent's interactions with the environment. At each point in time \( t \), the agent performs an action \( y_t \) and the environment generates an observation \( x_t \) and an instantaneous cost \( c_t \), according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimises some measure of a long-term cost, i.e. the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated. More formally, the environment is modeled as a Markov decision process (MDP) with states and actions with the following probability distributions: the instantaneous cost distribution \( P(c_t \mid s_t) \), the observation distribution \( P(x_t \mid s_t) \) and the transition \( P(s_{t+1} \mid s_t) \), while a policy is defined as conditional distribution over actions given the observations. Taken together, the two define a Markov chain (MC). The aim is to discover the policy that minimises the cost, i.e. the MC for which the cost is minimal. ANNs are frequently used in reinforcement learning as part of the overall algorithm. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

2.14.5 Learning Algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework,
determining a distribution over the set of allowed models) that minimises the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods simulated annealing, and Expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks.

2.15 INTRODUCTION TO MATLAB SOFTWARE

MATLAB is a technical computing environment for high-performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment where problems and solutions are expressed just as they are written mathematically - without traditional programming.

MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. This allows you to solve many numerical problems in a fraction of the time it would take to write a program in a language such as Fortran, Basic, or C.

MATLAB also features a family of application-specific solutions called toolboxes. Toolboxes are collections of MATLAB functions (M-files) that extend the MATLAB environment in order to solve particular classes of problems.
**File Management Commands**

- **dir** : show the files in the present directory
- **cd** : show the present directory
- **cd dir** : changes the directory, it is important that all the M-file you need during each working session are in the present directory in which you are working.
- **type filename** : list all the contents of filename
- **which filename** : displays the path of file name. This can be very useful to know if a file is part of standard Matlab package
- **what** : list the .m files that are in directory
- **who** : list all variables in the workspace
- **whose** : list all variables in the workspace along with size in bytes, array dimensions, and object type
- **clear** : removes all variables from the workspace, it is very helpful to type it at the beginning of each M-file in order to clear the space every time you rerun the same program
- **clear x** : removes just the variable x.

**Load and save data and results**

To load data, you need the data to be an ASCII file that does not contain any label or date, just columns of number. It is very easy if you have your data in an excel file, then cancel the column of the date and the first rows that usually just contains the name of the variables, and save them in txt format. Then to load the matrix of data, type:
load specify path where data file is\filename.txt;

Example: load c:\project\return.txt;

To save data or matrix of results, type:

save path that specify where you want to save the file\ new filename matrix_name -ascii

Example: save d:\Das\beta betaf -ascii;

**Arithmetic operations**

* : multiplication;
/ : division;
+ : addition;
- : subtraction;
^ : exponentiation
' : transposition

**Matrices**

When creating a matrix by inputting numbers or variables: use [] around the full data; spaces separate columns, semicolons separate rows.

>> A(i,j) == ith row and jth column of matrix A.
>> A(:,j) == all terms in the jth column of matrix A.
>> A(i, 4:6) == the 4th through 6th columns of row i in matrix A.

**Functions**

Matlab Programs are saved in M-files. These are text files that contain Matlab commands, and they are saved with .m extension. Any text
editor can be used to create them, but the one which comes with Matlab is recommended. When the .m files are executed, the commands are implemented as if you typed them directly. An .m file can contain and call other .m files. For example in one .m file you have programmed the objective function that you want to minimize or maximize, and then in another .m file where you have loaded the data and write down the command to implement the optimization you can call the .m file that contains the objective function. To execute the .m simply type the name of the file at the command line. It is important to note that .m that contains function have a special syntax for the first line.

function [out1,…outN] = func_name (inp1,….,inpN)

Example:

function f = test(param)

in this case there is one output f and only one input param. The function corresponding to the above syntax should be saved in a file like test.m. Hence in the above example, the file containing that first line has to be saved as test.m.

2.16 SUMMARY

In this section, different techniques for clustering Classification and outlier detection were studied. Detecting outliers in multidimensional data is a challenging problem. How to overcome Outlier deduction in multidimensional data is studied. The background study of Matelab was done.