

Jaypee University of Information Technology
Waknaghat, Distt. Solan (H.P.)

Learning Resource Center

CLASS NUM:

BOOK NUM.:

ACCESSION NO.: SP08008 / SP0812008

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date

DATA MINING SUITE USING EVOLUTIONARY COMPUTATION

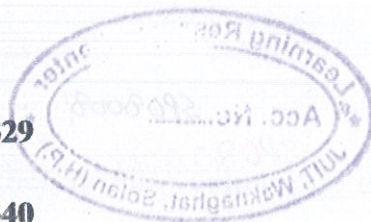
Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

By

Kushal Bansal 081329

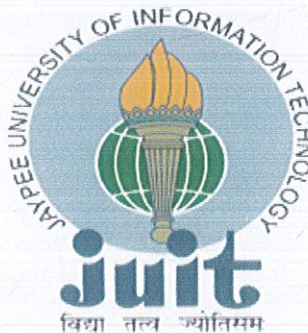
Vibhor Gera 081340



Under the guidance of

Mr. Pardeep Kumar

Senior Lecturer, CSE&IT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY WAKNAGHAT

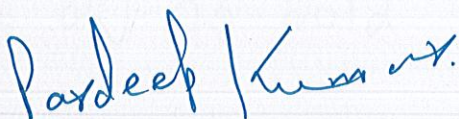
SOLAN, HIMACHAL PRADESH, INDIA

CERTIFICATE

This is to certify that the work titled "**DATA MINING SUITE USING EVOLUTIONARY COMPUTATION**" submitted by :

1. Kushal Bansal (081329)
2. Vibhor Gera (081340)

in partial fulfillment for the award of degree of Bachelors of Technology in Computer Science Engineering from Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.



(Signature of Supervisor)

MR.PARDEEP KUMAR

Date: 29-05-2012

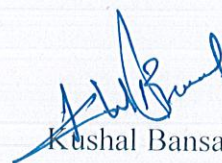
ACKNOWLEDGEMENT

Exchange of ideas generates the new object to work in a better way whenever a person is helped and cooperated by others his heart is bound to pay gratitude and obligation to them. To develop a project is not possible without rich guidance of someone experienced. It is essentially a collective work, where every step is taken with all precautions and care.

Therefore our first and foremost duty is to thank all persons who provided us with basic help in forming the outline and strategy for my humble effort.

We thank Mr. Pardeep Kumar, who gave us inspiration to do work in this field and gave us his precious time whenever needed. Thanks may be matter of merely formality but with us it is expression of heartfelt gratitude to our project supervision. We are highly indebted for his gestures, invaluable suggestions and boosting confidence to make this successful. The success of this work is mostly due to his suitable guidance.

Date: 29-05-2012



Kushal Bansal (081329)



Vibhor Gera (081340)

Table of Contents

1. Introduction.....	4
2. Algorithms	
a. Decision Tree.....	9
b. Neural Network.....	18
c. Evolutionary Neural Network.....	26
3. Results.....	34
4. Conclusion.....	44

INTRODUCTION

Data mining (the analysis step of the **knowledge discovery in databases** process, or KDD), a relatively young and interdisciplinary field of computer science is the process of discovering new patterns from large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics and database systems. The goal of data mining is to extract knowledge from a data set in a human-understandable structure and involves database and data management, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of found structure, visualization and online updating.

The term is a buzzword, and is frequently misused to mean any form of large scale data or information processing (collection, extraction, warehousing, analysis and statistics) but also generalized to any kind of computer decision support system including artificial intelligence, machine learning and business intelligence. In the proper use of the word, the key term is *discovery*, commonly defined as "detecting something new". Even the popular book "Data mining: Practical machine learning tools and techniques with Java" (which covers mostly machine learning material) was originally to be named just "Practical machine learning", and the term "data mining" was only added for marketing reasons. Often the more general terms "(large scale) data analysis" or "analytics" or when referring to actual methods, artificial intelligence and machine learning are more appropriate.

The actual data-mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). This usually involves using database techniques such as spatial indexes. These patterns can then be seen as a kind of summary of the input data, and used in further analysis or for example in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

Process

The **knowledge discovery in databases (KDD) process** is commonly defined with the stages (1) Selection (2) Preprocessing (3) Transformation (4) *Data Mining* (5) Interpretation/Evaluation. It exists however in many variations of this theme such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases:

(1)Business Understanding, (2)Data Understanding, (3)Data Preparation, (4)Modeling, (5) Evaluation, and (6)Deployment or a simplified process such as (1)Pre-processing, (2)Data mining, and (3)Results validation.

Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns actually present in the data, the target dataset must be large enough to contain these patterns while remaining concise enough to be mined in an acceptable timeframe. A common source for data is a data mart or data warehouse. Pre-process is essential to analyze the multivariate datasets before data mining.

The target set is then cleaned. Data cleaning removes the observations with noise and missing data.

Data mining

Data mining involves six common classes of tasks:

- Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors and require further investigation.
- Association rule learning (Dependency modeling) – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.
- Clustering– is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.
- Classification– is the task of generalizing known structure to apply to new data. For example, an email program might attempt to classify an email as legitimate or spam.
- Regression– Attempts to find a function which models the data with the least error.

- Summarization – providing a more compact representation of the data set, including visualization and report generation.

Results validation

The final step of knowledge discovery from data is to verify the patterns produced by the data mining algorithms occur in the wider data set. Not all patterns found by the data mining algorithms are necessarily valid. It is common for the data mining algorithms to find patterns in the training set which are not present in the general data set. This is called overfitting. To overcome this, the evaluation uses a test set of data on which the data mining algorithm was not trained. The learned patterns are applied to this test set and the resulting output is compared to the desired output. For example, a data mining algorithm trying to distinguish spam from legitimate emails would be trained on a training set of sample emails. Once trained, the learned patterns would be applied to the test set of emails on which it had not been trained. The accuracy of these patterns can then be measured from how many emails they correctly classify. A number of statistical methods may be used to evaluate the algorithm such as ROC curves.

If the learned patterns do not meet the desired standards, then it is necessary to reevaluate and change the pre-processing and data mining. If the learned patterns do meet the desired standards then the final step is to interpret the learned patterns and turn them into knowledge.

Challenges

Geospatial data repositories tend to be very large. Moreover, existing GIS datasets are often splintered into feature and attribute components, that are conventionally archived in hybrid data management systems. Algorithmic requirements differ substantially for relational (attribute) data management and for topological (feature) data management. Related to this is the range and diversity of geographic data formats, that also presents unique challenges. The digital geographic data revolution is creating new types of data formats beyond the traditional "vector" and "raster" formats. Geographic data repositories increasingly include ill-structured data such as imagery and geo-referenced multi-media.

There are several critical research challenges in geographic knowledge discovery and data mining. Miller and Han offer the following list of emerging research topics in the field:

- **Developing and supporting geographic data warehouses** – Spatial properties are often reduced to simple aspatial attributes in mainstream data warehouses. Creating an integrated

GDW requires solving issues in spatial and temporal data interoperability, including differences in semantics, referencing systems, geometry, accuracy and position.

- **Better spatio-temporal representations in geographic knowledge discovery** – Current geographic knowledge discovery (GKD) methods generally use very simple representations of geographic objects and spatial relationships. Geographic data mining methods should recognize more complex geographic objects (lines and polygons) and relationships (non-Euclidean distances, direction, connectivity and interaction through attributed geographic space such as terrain). Time needs to be more fully integrated into these geographic representations and relationships.
- **Geographic knowledge discovery using diverse data types** – GKD methods should be developed that can handle diverse data types beyond the traditional raster and vector models, including imagery and geo-referenced multimedia, as well as dynamic data types (video streams, animation).

In four annual surveys of data miners, data mining practitioners consistently identified that they faced three key challenges more than any others:

- Dirty Data
- Explaining Data Mining to Others
- Unavailability of Data / Difficult Access to Data

Privacy concerns and ethics

Some people believe that data mining itself is ethically neutral. It is important to note that the term data mining has no ethical implications. The term is often associated with the mining of information in relation to peoples' behavior. However, data mining is a statistical method that is applied to a set of information, or a data set. Associating these data sets with people is an extreme narrowing of the types of data that are available in today's technological society. Examples could range from a set of crash test data for passenger vehicles, to the performance of a group of stocks. These types of data sets make up a great proportion of the information available to be acted on by data mining methods, and rarely have ethical concerns associated with them. However, the ways in which data mining can be used can raise questions regarding privacy, legality, and ethics. In particular, data mining government or commercial data sets for national security or law enforcement purposes, such as in the Total Information Awareness Program or in ADVISE, has raised privacy concerns.

Data mining requires data preparation which can uncover information or patterns which may compromise confidentiality and privacy obligations. A common way for this to occur is through data aggregation. Data aggregation is when the data are accrued, possibly from various sources, and put together so that they can be analyzed. This is not data mining per se, but a result of the preparation of data before and for the purposes of the analysis. The threat to an individual's privacy comes into play when the data, once compiled, cause the data miner, or anyone who has access to the newly compiled data set, to be able to identify specific individuals, especially when originally the data were anonymous.

It is recommended that an individual is made aware of the following before data are collected:

- the purpose of the data collection and any data mining projects,
- how the data will be used,
- who will be able to mine the data and use them,
- the security surrounding access to the data, and in addition,
- how collected data can be updated.

In the United States, privacy concerns have been somewhat addressed by their congress via the passage of regulatory controls such as the Health Insurance Portability and Accountability Act (HIPAA). The HIPAA requires individuals to be given "informed consent" regarding any information that they provide and its intended future uses by the facility receiving that information. According to an article in Biotech Business Week, "In practice, HIPAA may not offer any greater protection than the longstanding regulations in the research arena, says the AAHC. More importantly, the rule's goal of protection through informed consent is undermined by the complexity of consent forms that are required of patients and participants, which approach a level of incomprehensibility to average individuals." This underscores the necessity for data anonymity in data aggregation practices.

One may additionally modify the data so that they are anonymous, so that individuals may not be readily identified. However, even de-identified data sets can contain enough information to identify individuals, as occurred when journalists were able to find several individuals based on a set of search histories that were inadvertently released by AOL.

ALGORITHMS

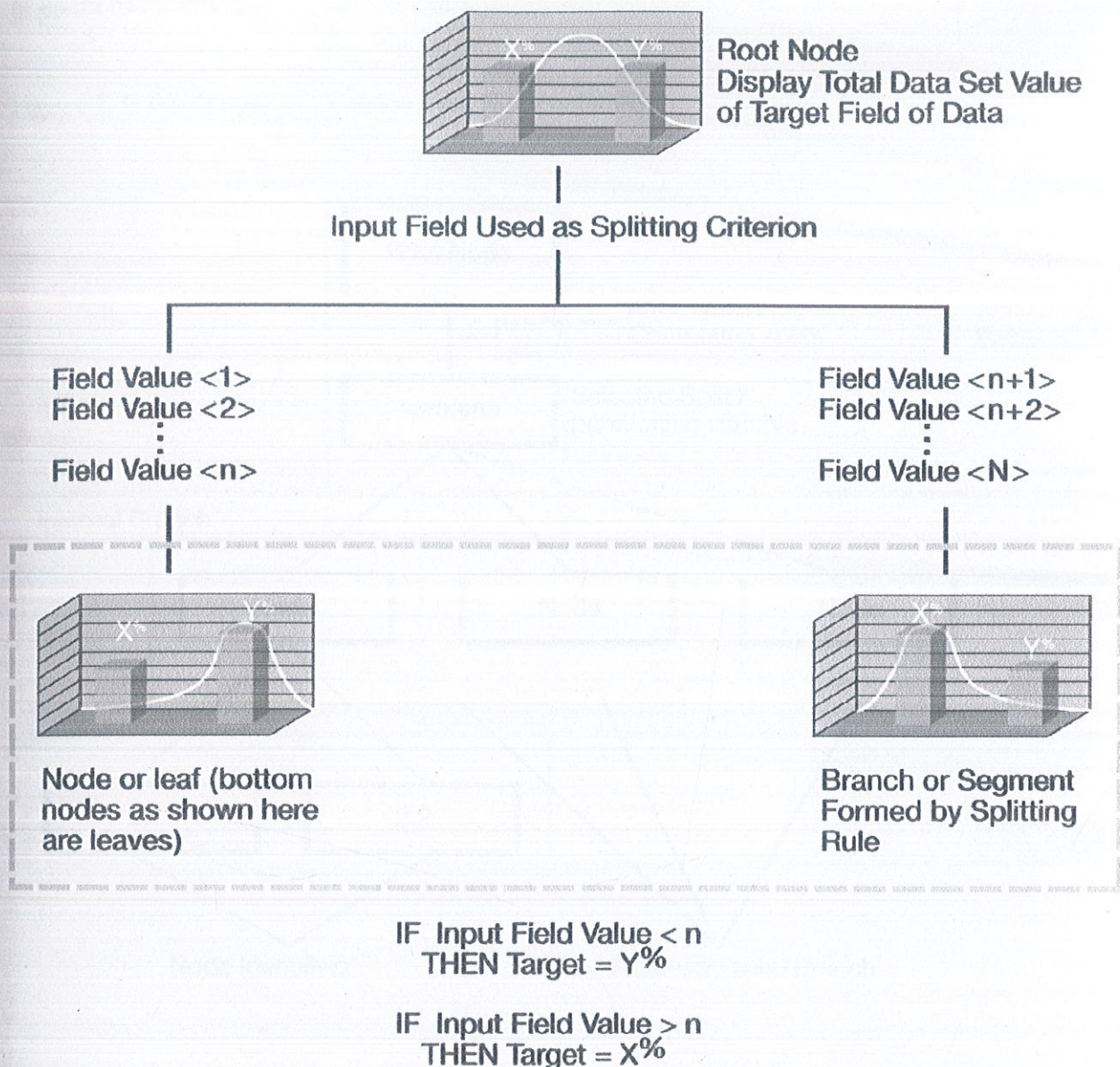
a) Decision Tree

Decision trees are a simple, but powerful form of multiple variable analysis. They provide unique capabilities to supplement, complement, and substitute for

- traditional statistical forms of analysis (such as multiple linear regression)
- a variety of data mining tools and techniques (such as neural networks)
- recently developed multidimensional forms of reporting and analysis found in the field of business intelligence

Decision trees are produced by algorithms that identify various ways of splitting a data set into branch-like segments. These segments form an inverted decision tree that originates with a root node at the top of the tree. The object of analysis is reflected in this root node as a simple, one-dimensional display in the decision tree interface. The name of the field of data that is the object of analysis is usually displayed, along with the spread or distribution of the values that are contained in that field. A sample decision tree is illustrated in Figure 1.1, which shows that the decision tree can reflect both a continuous and categorical object of analysis. The display of this node reflects all the data set records, fields, and field values that are found in the object of analysis. The discovery of the decision rule to form the branches or segments underneath the root node is based on a method that extracts the relationship between the object of analysis (that serves as the target field in the data) and one or more fields that serve as input fields to create the branches or segments. The values in the input field are used to estimate the likely value in the target field. The target field is also called an outcome, response, or dependent field or variable. The general form of this modeling approach is illustrated in Figure 1.1. Once the relationship is extracted, then one or more decision rules can be derived that describe the relationships between inputs and targets. Rules can be selected and used to display the decision tree, which provides a means to visually examine and describe the tree-like network of relationships that characterize the input and target values. Decision rules can predict the values of new or unseen observations that contain values for the inputs, but might not contain values for the targets.

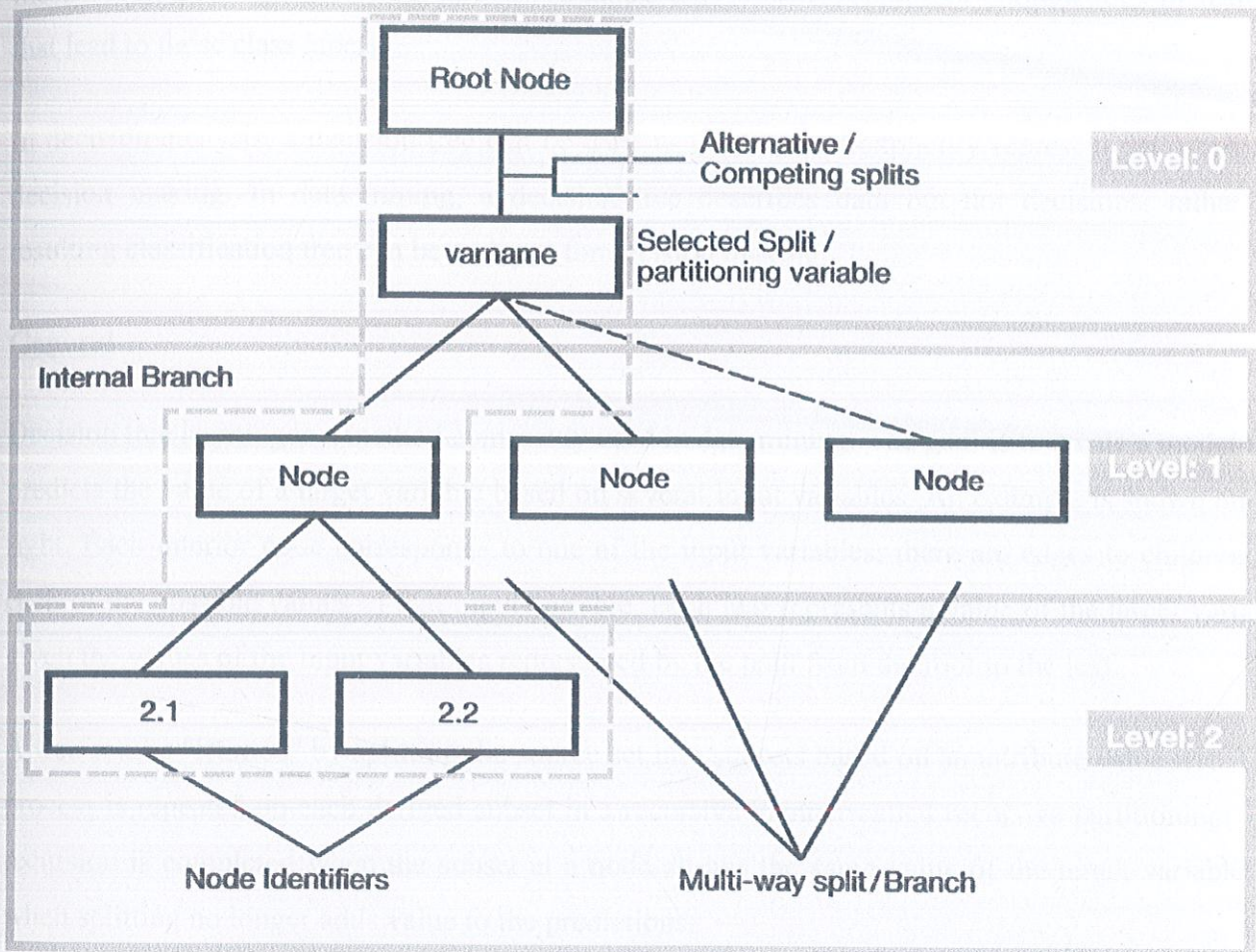
Figure 1.1: Illustration of the Decision Tree



Each rule assigns a record or observation from the data set to a node in a branch or segment based on the value of one of the fields or columns in the data set.¹ Fields or columns that are used to create the rule are called *inputs*. Splitting rules are applied one after another, resulting in a hierarchy of branches within branches that produces the characteristic inverted decision tree form. The nested hierarchy of branches is called a *decision tree*, and each segment or branch is called a *node*. A node with all its descendent segments forms an additional segment or a branch of that node. The bottom nodes of the decision tree are called *leaves* (or *terminal nodes*). For each leaf, the decision rule provides a unique path for data to enter the class that is defined as the leaf. All nodes, including the bottom leaf nodes, have mutually exclusive assignment rules; as a result, records or observations from the parent data set can be found in one node only. Once the decision rules have been

determined, it is possible to use the rules to predict new node values based on new or unseen data. In predictive modeling, the decision rule yields the predicted value.

Figure 1.2: Illustration of Decision Tree Nomenclature



Although decision trees have been in development and use for over 50 years (one of the earliest uses of decision trees was in the study of television broadcasting by Belson in 1956), many new forms of decision trees are evolving that promise to provide exciting new capabilities in the areas of data mining and machine learning in the years to come.

For example, one new form of the decision tree involves the creation of *random forests*. Random forests are multi-tree committees that use randomly drawn samples of data and inputs and reweighting techniques to develop multiple trees that, when combined, provide for stronger prediction and better diagnostics on the structure of the decision tree. Besides modeling, decision trees can be used to explore and clarify data for dimensional cubes that can be found in business analytics and business intelligence.

Decision tree learning

Decision tree learning, used in statistics, data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. More descriptive names for such tree models are **classification trees** or **regression trees**. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making.

General

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown on the right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

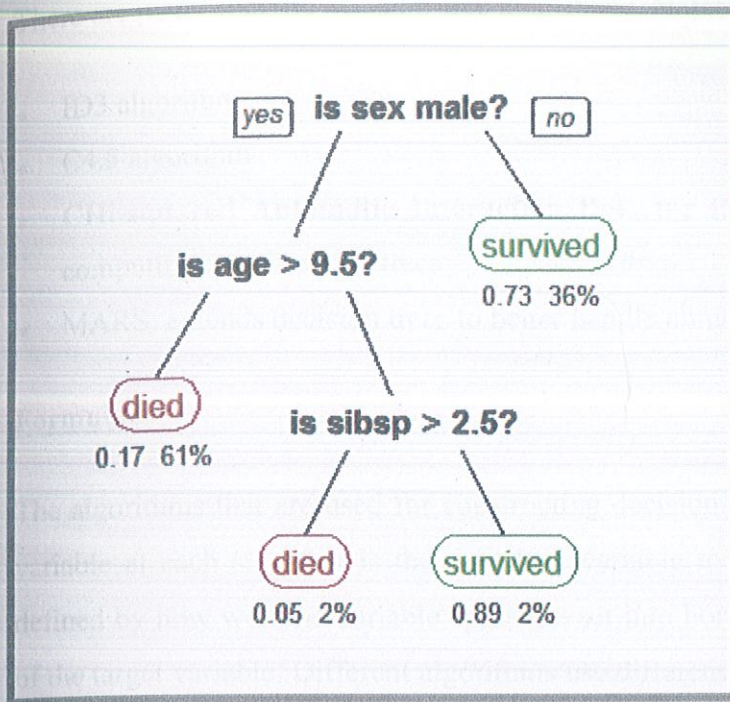
A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

In data mining, trees can be described also as the combination of mathematical and computational techniques to aid the description, categorisation and generalisation of a given set of data.

Data comes in records of the form:

$$(\mathbf{x}, Y) = (x_1, x_2, x_3, \dots, x_k, Y)$$

The dependent variable, Y , is the target variable that we are trying to understand, classify or generalise. The vector \mathbf{x} is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task.



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

Types

Decision trees used in data mining are of two main types:

- **Classification tree** analysis is when the predicted outcome is the class to which the data belongs.
- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to both of the above procedures, first introduced by Breiman et al. Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.

Some techniques use more than one decision tree for their analysis:

- A **Random Forest** classifier uses a number of decision trees, in order to improve the classification rate.
- **BoostedTrees** can be used for regression-type and classification-type problems.

There are many specific decision-tree algorithms. Notable ones include:

- ID3 algorithm
- C4.5 algorithm
- **CHi-squared Automatic Interaction Detector (CHAID)**. Performs multi-level splits when computing classification trees.
- MARS: extends decision trees to better handle numerical data

Formulae

The algorithms that are used for constructing decision trees usually work top-down by choosing a variable at each step that is the next best variable to use in splitting the set of items. "Best" is defined by how well the variable splits the set into homogeneous subsets that have the same value of the target variable. Different algorithms use different formulae for measuring "best". This section presents a few of the most common formulae. These formulae are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

Decision tree advantages

Amongst other data mining methods, decision trees have various advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation.
- **Requires little data preparation.** Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- **Able to handle both numerical and categorical data.** Other techniques are usually specialised in analysing datasets that have only one type of variable. Ex: relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.
- **Uses a white box model.** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is difficult to understand.
- **Possible to validate a model using statistical tests.** That makes it possible to account for the reliability of the model.
- **Robust.** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

- **Performs well with large data in a short time.** Large amounts of data can be analysed using standard computing resources.

Limitations

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.
- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting.^[8] Mechanisms such as pruning are necessary to avoid this problem.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large. Approaches to solve the problem involve either changing the representation of the problem domain (known as propositionalisation)^[9] or using learning algorithms based on more expressive representations (such as statistical relational learning or inductive logic programming).
- For data including categorical variables with different number of levels, information gain in decision trees are biased in favor of those attributes with more levels.

Machine learning

Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. Machine learning is concerned with the development of algorithms allowing the machine to learn via inductive inference based on observing data that represents incomplete information about statistical phenomenon and generalize it to rules and make *predictions* on missing attributes or future data. An important task of machine learning is classification, which is also referred to as pattern recognition, in which machines "learn" to automatically recognize complex patterns, to distinguish between exemplars based on their different patterns, and to make intelligent predictions on their class.

Machine learning, knowledge discovery in databases (KDD) and data mining

These three terms are commonly confused, as they often employ the same methods and overlap strongly. They can be roughly separated as follows:

- Machine learning focuses on the prediction, based on *known* properties learned from the training data
- Data mining (which is the analysis step of Knowledge Discovery in Databases) focuses on the discovery of (previously) *unknown* properties on the data

However, these two areas overlap in many ways: data mining uses many machine learning methods, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, the performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in KDD the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Human interaction

Some machine learning systems attempt to eliminate the need for human intuition in data analysis, while others adopt a collaborative approach between human and machine. Human intuition cannot, however, be entirely eliminated, since the system's designer must specify how the data is to be represented and what mechanisms will be used to search for a characterization of the data.

Algorithm types

Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm.

- **Supervised learning** generates a function that maps inputs to desired outputs (also called **labels**, because they are often provided by human experts labeling the training examples). For example, in a classification problem, the learner approximates a function mapping a vector into classes by looking at input-output examples of the function.

- **Unsupervised learning** models a set of inputs, like clustering. See also data mining and knowledge discovery.
- **Semi-supervised learning** combines both labeled and unlabeled examples to generate an appropriate function or classifier.
- **Reinforcement learning** learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.
- **Transduction** tries to predict new outputs based on training inputs, training outputs, and test inputs.
- **Learning to learn** learns its own inductive bias based on previous experience.

b) Neural networks in data mining

Companies have been collecting data for decades, building massive data warehouses in which to store it. Even though this data is available, very few companies have been able to realize the actual value stored in it. The question these companies are asking is how to extract this value. The answer is Data mining. There are many technologies available to data mining practitioners, including Artificial Neural Networks, Regression, and Decision Trees. Many practitioners are wary of Neural Networks due to their black box nature, even though they have proven themselves in many situations. This paper is an overview of artificial neural networks and questions their position as a preferred tool by data mining practitioners.

INTRODUCTION

Data mining is the term used to describe the process of extracting value from a database. A data-warehouse is a location where information is stored. The type of data stored depends largely on the type of industry and the company. Many companies store every piece of data they have collected, while others are more ruthless in what they deem to be "important". Consider the following example of a financial institution failing to utilize their data-warehouse. Income is a very important socio-economic indicator. If a bank knows a person's income, they can offer a higher credit card limit or determine if they are likely to want information on a home loan or managed investments. Even though this financial institution had the ability to determine a customer's income in two ways, from their credit card application, or through regular direct deposits into their bank account, they did not extract and utilize this information.

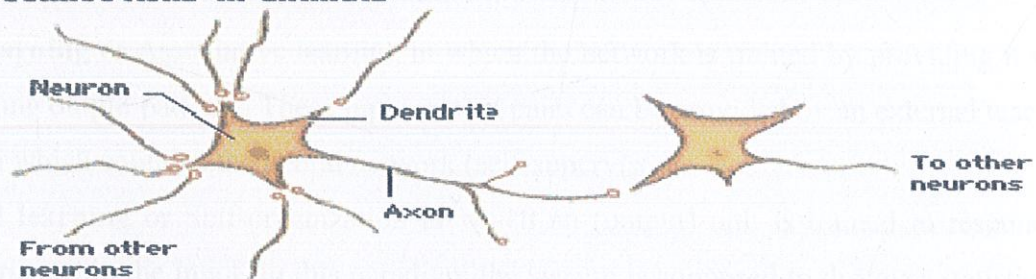
Another example of where this institution has failed to utilize its data-warehouse is in cross-selling insurance products (e.g. home, life and motor vehicle insurance). By using transaction information they may have the ability to determine if a customer is making payments to another insurance broker. This would enable the institution to select prospects for their insurance products.

These are simple examples of what could be achieved using data mining. Four things are required to data-mine effectively: high-quality data, the "right" data, an adequate sample size and the right tool. There are many tools available to a data mining practitioner. These include decision trees, various types of regression and neural networks.

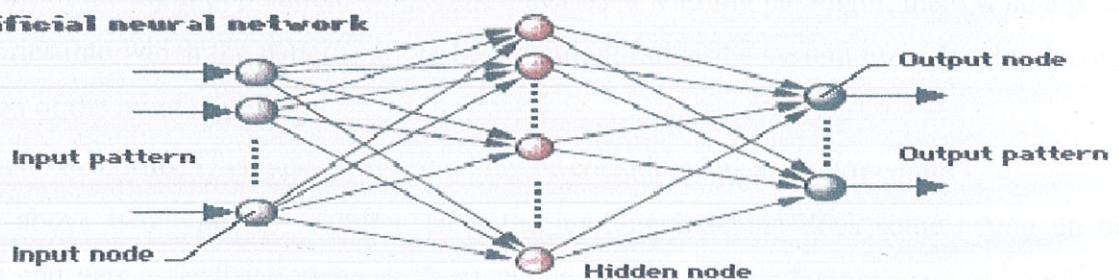
ARTIFICIAL NEURAL NETWORKS

An **artificial neural network** (ANN), often just called a "neural network" (NN), is a mathematical model or computational model based on biological neural networks, in other words, is an emulation of biological neural system. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

Neural connections in animals



Artificial neural network



Neural Network Topologies

Feedforward neural network: The feedforward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through

the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. The data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to

inputs of units in the same layer or previous layers. **Recurrent network:** Recurrent neural networks that do contain feedback connections. Contrary to feedforward networks, recurrent neural networks (RNs) are models with bi-directional data flow.

While a feedforward network propagates data linearly from input to output, RNs also propagate data from later processing stages to earlier stages.

Training Of Artificial Neural Networks

A **neural network** has to be configured such that the application of a set of inputs produces (either 'direct' or via a relaxation process) the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights

explicitly, using a priori knowledge. Another way is to '**train**' the **neural network** by feeding it teaching patterns and letting it change its weights according to some learning rule. We can categorize the learning situations as follows:

- **Supervised learning** or Associative learning in which the network is trained by providing it with input and matching output patterns. These input-output pairs can be provided by an external teacher, or by the system which contains the neural network (self-supervised).
- **Unsupervised learning** or Self-organization in which an (output) unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover statistically salient features of the input population. Unlike the supervised learning paradigm, there is no a priori set of categories into which the patterns are to be classified; rather the system must develop its own representation of the input stimuli.

Reinforcement Learning This type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment. The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and accordingly adjusts its parameters.

In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Using neural networks as a tool, data warehousing firms are harvesting information from datasets in the process known as data mining. The difference between these data warehouses and ordinary databases is that there is actual manipulation

and cross-fertilization of the data helping users makes more informed decisions.

the architecture or model; the learning algorithm; and the activation functions. Neural networks are programmed or "trained" to ". . . store, recognize, and associatively retrieve patterns or database entries; to solve combinatorial optimization problems; to filter noise from measurement data; to control ill-defined problems; in summary, to

estimate sampled functions when we do not know the form of the functions." It is precisely these two abilities (pattern recognition and function estimation) which make artificial neural networks (ANN) so prevalent a utility in data mining. As data sets grow to massive sizes, the need for automated processing becomes clear. With their "model-free" estimators and their dual nature, neural networks serve data mining in a myriad of ways.

Data mining is the business of answering questions that you've not asked yet. Data mining reaches deep into databases. Data mining tasks can be classified into two categories: Descriptive and predictive data mining. Descriptive data mining

provides information to understand what is happening inside the data without a predetermined idea. Predictive data mining allows the user to submit records with unknown field values, and the system will guess the unknown values based on previous patterns discovered from the database.

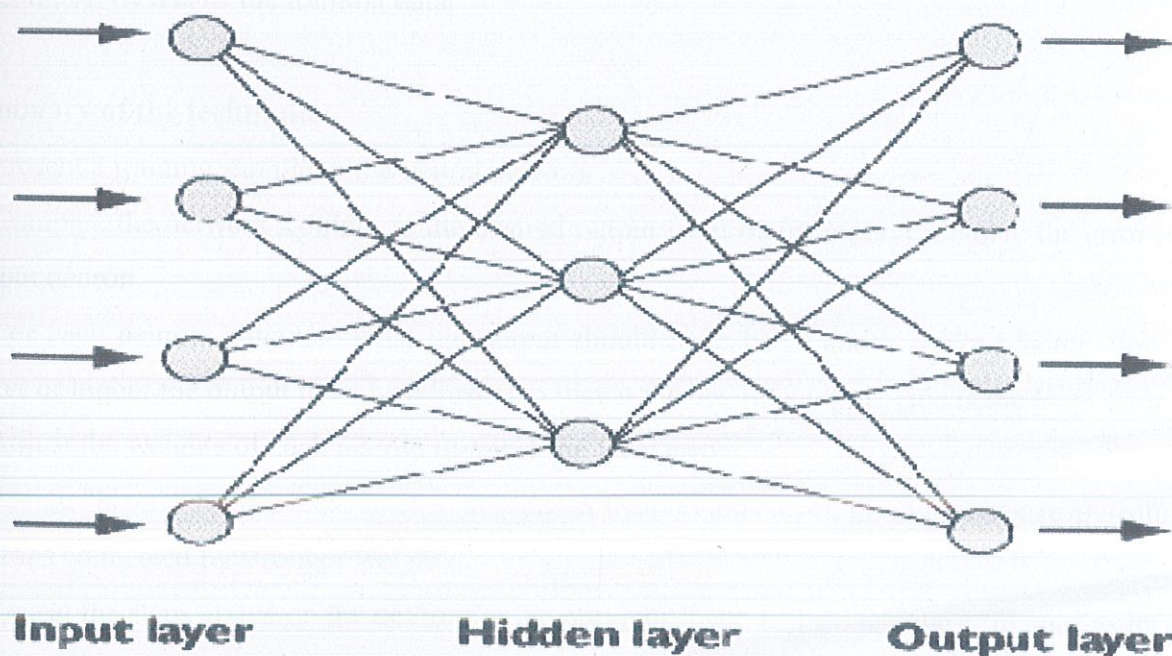
Data mining models can be categorized according to the tasks they perform: Classification and Prediction, Clustering, Association Rules. Classification and prediction is a predictive model, but clustering and association rules are descriptive models. The most common action in data mining is classification. It recognizes patterns that describe the group to which an item belongs. It does this by examining existing items that already have been classified and inferring a set of rules. Similar to classification is clustering. The major difference being that no groups have been predefined.

Prediction is the construction and use of a model to assess the class of an unlabeled object or to assess the value or value ranges of a given object is likely to have. The next application is forecasting. This is different from predictions because it estimates the future value of continuous variables based on patterns within the data. Neural networks, depending on the architecture, provide associations, classifications, clusters, prediction and forecasting to the data mining industry.

Financial forecasting is of considerable practical interest. Due to neural networks can mine valuable information from a mass of history information and be efficiently used in financial areas, so the applications of neural networks to financial forecasting have been very popular over the last few years. Some researches show that neural networks performed better than conventional statistical approaches in financial forecasting and are an excellent data mining tool. In data warehouses, neural networks are just one of the tools used in data mining. ANNs are used to find patterns in the data and to infer rules from them. Neural networks are useful in providing information on associations, classifications, clusters, and forecasting. The back propagation algorithm performs learning on a feed-forward neural network.

Feedforward Neural Network

One of the simplest feed forward neural networks (FFNN), such as in Figure, consists of three layers: an input layer, hidden layer and output layer. In each layer there are one or more processing elements (PEs). PEs is meant to simulate the neurons in the brain and this is why they are often referred to as neurons or nodes. A PE receives inputs from either the outside world or the previous layer. There are connections between the PEs in each layer that have a weight (parameter) associated with them. This weight is adjusted during training. Information only travels in the forward direction through the network - there are no feedback loops.



The simplified process for training a FFNN is as follows:

1. Input data is presented to the network and propagated through the network until it reaches the output-layer. This forward process produces a predicted output.
2. The predicted output is subtracted from the actual output and an error value for the networks is calculated.
3. The neural network then uses supervised learning, which in most cases is back propagation, to train the network. Back propagation is a learning algorithm for adjusting the weights. It starts with the weights between the output layer PE's and the last hidden layer PE's and works backwards through the network.
4. Once back propagation has finished, the forward process starts again, and this cycle is continued until the error between predicted and actual outputs is minimized.

The Back Propagation Algorithm

Backpropagation, or **propagation of error**, is a common method of teaching artificial neural networks how to perform a given task. The back propagation algorithm is used in layered feedforward ANNs. This means that the artificial neurons are organized in layers, and send their signals "forward", and then the errors are propagated backwards. The back propagation algorithm uses supervised learning, which means that we provide the algorithm with examples of the inputs and outputs we want the network to compute, and then the error (difference between actual and expected results) is calculated. The idea of the back propagation algorithm is to reduce this error, until the ANN *learns* the training data.

Summary of the technique:

1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat the steps above on the neurons at the previous level, using each one's "blame" as its error.

Applications for Neural Networks

Neural Networks are successfully being used in many areas often in connection with the use of other AI techniques.

A classic application for NN is image recognition. A network that can classify different standard images can be used in several areas:

- ☐ Quality assurance, by classifying a metal welding as whether it holds the quality standard.
- ☐ Medical diagnostics, by classifying x-ray pictures for tumor diagnosis.
- ☐ Detective tools, by classifying fingerprints to a database of suspects.

A well known application using image recognition is the Optical Character Recognition (OCR) tools that we find available with the standard scanning software for the home computer. Scansoft has had great success in combining NN with a rule based system for correctly recognising both characters and words, to get a high level of accuracy¹.

All the network topologies and algorithms have their advantages and disadvantages. When it comes to understanding the spoken language the best found solutions use a combination of NN for phoneme recognition and an Expert system for Natural language processing, where neither AI technique can be adapted to solve the problem in whole. Kohonen himself succeeded in creating a 'phonetic typewriter' by using his self-organising networks for the phoneme recognition and a rule base for applying the correct grammar. Another popular application for NN is Customer Relationship Management (CRM). Many companies have at the same rate as electronic data storage has become commonplace built up large customer databases. By using Neural Networks for data mining in these databases, patterns however complex can be identified for the different types of customers, thus giving valuable customer information to the company. One example is the airline reservation system AMT2 which could predict sales of tickets in relation to destination, time of year and ticket price. The NN strategy was well suited for the purpose because the system could be updated continuously with the actual sales.

In relation to the recent trends in Management strategies CRM has reached a high priority, because of the prospects of a successful CRM system adding value to the business in terms of not only better prediction of customer needs but also predicting which customers will be the most valuable for the company.

Some rules of thumb exist for evaluating whether a problem is suitable for a Neural Network implementation:

- ☐ There must be a large example dataset of the problem in order to be able to train the network.
- ☐ The data relationships in the problem are complex and difficult or impossible to program using conventional techniques.

- The output does not need to be exact or numeric.
- The desired output from the system changes over time, so a high flexibility is needed.

Many commercial NN programs exist both for stand-alone or built-in applications.

Problems using Neural Networks

Local Minimum

All the NN in this paper are described in their basic algorithm. Several suggestions for improvements and modifications have been made. One of the well-known problems in the MLP is the *local minimum*: The net does not settle in one of the learned minima but instead in a local minimum in the Energy landscape (Illustration 10 The Energy Landscape).

Approaches to avoid local minimum:

- The *gain term* in the weight adaption function can be lowered progressively as the network iterates. This would at first let the differences in weights and energy be large, and then hopefully when the network is approaching the right solution, the steps would be smaller. The tradeoff is when the gain term has decreased the network will take a longer time to converge to right solution. (Formula 6 MLP Adapt weights)
 - A local minimum can be caused by a bad internal representation of the patterns. This can be aided by the adding more internal nodes to the network.
 - An extra term can be added to the weight adaption: the *Momentum term*. The Momentum term should let the weight change be large if the current change in energy is large. (Formula 9 MLP Momentum term)
 - The network gradient descent can be disrupted by adding random noise to ensure sure the sytem will take unequal steps toward the solution. This solution has the advantage, that it requires no extra computation time.
- A similar problem is known in the Hopfield Net as *metastable states*. That is when the network settles in a state that is not represented in the stored patterns. One way to minimise this is by adjusting the number of nodes in the network(N) to the number of patterns to store, so that the number of patterns does not exceed $0.15N$. Another solution is to add a probabilistic update rule to the Hopfield network. This is known as the Boltzman machine.

Practical problems

There are some practical problems applying Neural networks to applications. It is not possible to know in advance the ideal network for an application. So every time a NN is to be built in an application, it requires tests and experiments with different network settings or topologies to find a solution that performs well on the given application. This is a problem because most NN requires a long training period – many iterations of the same pattern set. And even after many iterations there is no way other than testing to see whether the network is efficiently mapping the training sets. A solution for this might be to adapt newer NN technologies such as the bump tree which need only one run through the training set to adjust all weights in the network. The most commonly used network still seems to be the MLP and the RBF3 even though alternatives exist that can drastically shorten processing time.

In general most NN include complex computation, which is time consuming. Some of these computations could gain efficiency if they were to be implemented on a parallel processing system, but the hardware implementation raises new problems of physical limits and the NN need for changeability.

c) Evolutionary Neural Networks

Artificial Neural Networks and Evolutionary Algorithms are both abstractions of natural processes. They are formulated into a computational model so that the learning power of neural networks and adaptive capabilities of evolutionary processes can be harnessed in an artificial life environment. "*Adaptive learning*", as it is called, produces results that demonstrate how complex and purposeful behaviour can be induced in a system by randomly varying the topology and the rules governing the system. Evolutionary algorithms can help determine optimized neural network architectures, as well as, provide faster mechanisms to train the network to identify its purpose. Adaptation being the keyword here, modifications or reformulations in the network's objectives can also be easily handled without much effort.

The brain is a mesh of billions and billions of neurons. The process of neural communication, that gives the brain most of its functional complexity, has been a key area of research for many years now. There has been a great deal of effort to understand the processes of information storage and retrieval going on inside the brain. A rather deterministic procedure has been contrived to explain it, which in turn inspired the foundations of an artificial model - artificial neural networks.

Every neuron is charged with its *excitation potential*, which is much like an electrical impulse. This electric pulse propagates down a connecting linkage, called the axon, starting at the nucleus of the cell and ending in another neuron. At the end of the axon, chemicals, called *neural transmitters*, are released because of this excitation potential. These chemical carriers bind themselves to receptors of the neighbouring neurons, and then either activate the neuron or inhibit it from reaching its excitation potential.

The receptors on the receiving cell control the threshold to which a neuron has to be pushed in order to achieve its excitation potential. The neural transmitters are capable of causing the potential of a cell go up or down, and hence the number of receptors actually determine whether a neuron has to be charged or not. The cell can control this number of receptors depending on how frequently it has been excited. More number of excitations in a relatively smaller interval of time will result in an increase in this number making it more prone the next time around; whereas an inactive neuron will have less of these receptors making it less likely to fire. The chemical makeup decides whether a neuron reacts to impulses or not.

It can be seen that the information stored in a neuron is an evolutionary process, rather than being a fixed step biological one. Over time, an active neuron may loose its influence and an inactive one gain more engagement. The configuration of the whole network changes frequently, and adapts very well to changing or new levels of information. Moreover, this network is massively parallel, robust, fault tolerant and amenable to learning. Although it is not an easy task to design an artificial neural network with all of these attributes, the basic functionalities can always be modeled.

Artificial Neural Networks (ANN)

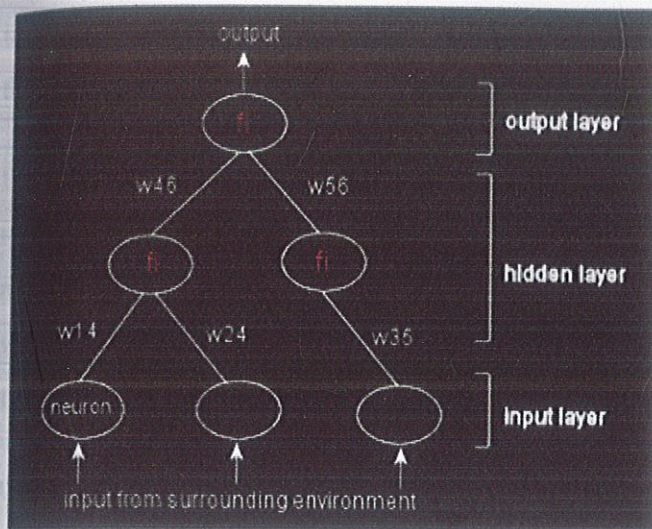
An ANN is an interconnected network of very simple calculating units called *neurons*. Every connection in the network is assigned a weight which specifies the extent of possible influence. The structure of the network determines how the neurons influence each other. The whole network can be represented using a directed graph where an incoming edge to a node acts like an input to a neuron and outgoing edges are outputs from the neuron.

Each node performs a transfer of stimuli based on its activation function. A node produces an output only if the weighted sum of its input is greater than its set threshold. More specifically, a nonlinear transfer function of this parameter is used in actual implementations. We may write,

$$y_i = f_i (\sum w_{ij} x_j - T_i)$$

where y_i is output at node i , x_j is the input from the j th node, and w_{ij} is the connection weight between nodes i and j . T_i is the threshold of the node. f_i is nonlinear such as Heaviside, Sigmoid or Gaussian function. We shall not worry about the mathematical complexities of these functions and assume that every node has the same transfer function.

Figure 1: A typical neural network.



An ANN consists of an input layer, an output layer and one or more hidden layers. No transfer function is applied at the input layer and direct inputs are transferred as outputs from this layer. The input layer acts like the biological sensory system, providing information about the surrounding environment. Activations are calculated from the next layer onwards (the hidden layers) and fed into higher layers till it reaches the final output layer. This kind of an ANN structure is called a *feedforward network* - output from one layer goes to neurons in the next layer. We can also have feedback networks, commonly called a *recurrent network*.

Learning in ANNs is achieved by varying the connection weights iteratively so that the network is trained to perform certain tasks. It generally involves the minimization of some error function - say the total mean square error between the actual and expected output - under the supervision of a trainer. This is often called *supervised training*. However, in some cases, the exact desired output is not known. *Reinforcement learning* is used in such cases and training is based only on whether the actual output is correct or not. *Unsupervised learning* tries to find correlations among input data when no information on the correctness of the output is available. The rule followed to update the connection weights - *the learning rule* - determines how well the network converges towards its desired optimality.

In this article, we shall see an evolutionary methodology to figure out an optimized ANN, both in terms of architecture and connection weights.

Evolutionary Algorithms (EA)

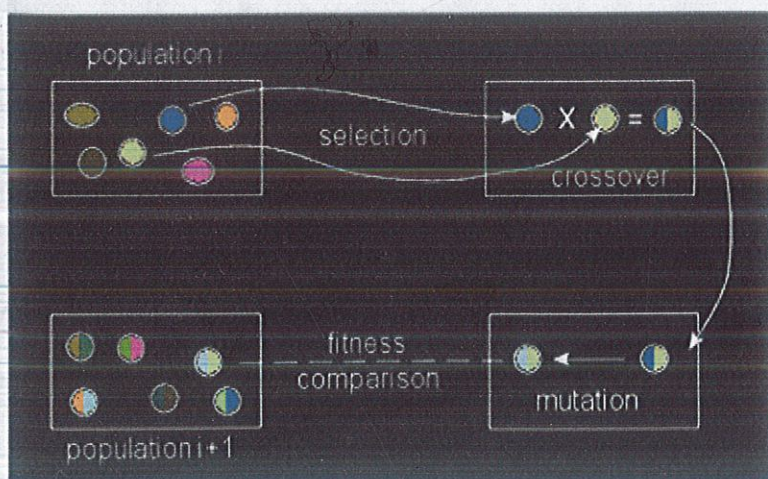
Evolutionary algorithms refer to a class of algorithms based on probabilistic adaptation inspired by the principles of natural evolution. They follow a stochastic search strategy on a population of individuals, each representing a possible solution to the problem. They are broadly classified into three main forms - *evolution strategies*, *genetic algorithms*, and *evolutionary programming*.

EAs start with a randomly generated set of trial solutions. Each of these solutions is assigned a *fitness value* depending on how well it suits the solution. For example, given a function to maximize, the value of the function itself can serve as a fitness factor. In the case of an ANN, where we need to minimize the mean square error, the inverse of the same value ($1/\text{MSE}$) can be used for fitness evaluation.

Once all the members of the population are assigned fitness values, a *selection* process is carried out where better individuals (high fitness value) stand a greater chance to be selected for the next operation.

Selected individuals undergo *recombination* and *mutation* to result in new individuals. Low fitness individuals are discarded from the population and better ones are included. The whole process is repeated with this new population until some termination criteria is satisfied. The average fitness of the population is expected to increase over generations, and finally converge at the point where the optimum is found.

Figure 2: Genetic operations in an EA.



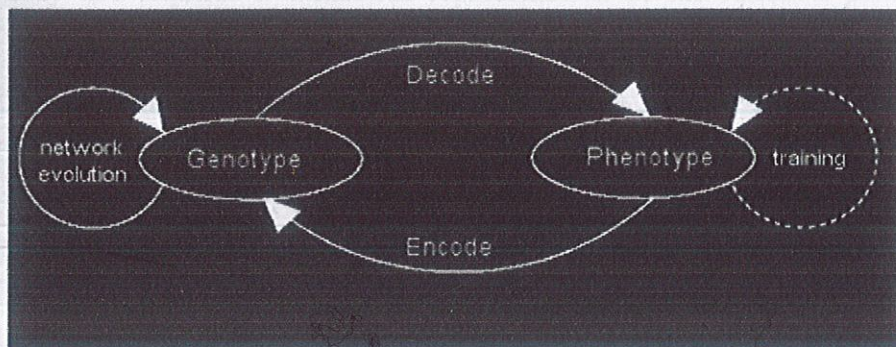
A suitable genetic representation of trial solutions is important in order to work with EAs. A function to map this representation back to real problem parameters is also required. We shall see how such a representation is done in case of an ANN. Apart from this, recombination and mutation operators need to be specified in a way that the offspring generated is a valid trial solution.

EAs for ANNs

The applications of EAs in ANNs are mostly concentrated in finding suitable network topologies and then training the network. EAs can quickly locate areas of high quality solutions when the domain is very large or complex. This is important in ANN design and training where the search space is infinite, highly dimensional and multimodal.

The evolution of connection weights introduces an adaptive and global approach to training. Unlike *gradient-based* training methods, viz. back propagation, EAs rely on probabilistic search techniques and so, even though their search space is bigger, they can ensure that better solutions are being generated over generations. Optimal network architectures can also be evolved to fit a given task at hand. The representation and search operators used in EAs are two most important issues in the evolution of architectures. It is shown that EAs relying on the crossover operator do not perform very well in searching for optimal network topologies.

Figure 3: Evolutionary design of an ANN.



It is also possible to simultaneously evolve both the structure and weights of an ANN. However, what is mostly done is to evolve the network first and then subject this network to a fixed number of training steps. Although evolution in this manner is straight forward, it may result in a *noisy fitness* evaluation. Different random initial weights may produce different training results; different training algorithms may produce different training results from the same set of initial weights. The other way is to partially train the network before any genetic operator is applied to it. Mutation itself contains some level of training and then a final training phase once the structure is decided.

In the next section we shall see how genetic representations, performance evaluation, and search operators can be defined for a typical EA to be used for ANN design and training. We shall be looking specifically at a *Gaussian mutation* based real coded evolutionary method for training, and a *direct coded* method for evolution of architectures.

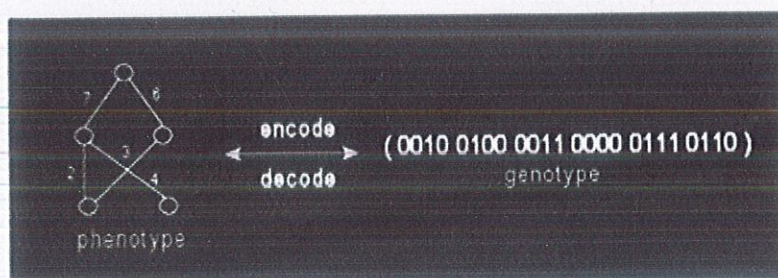
Evolution of Connection Weights

Training a given network topology to recognize its purpose generally means to determine an optimal set of connection weights. This is formulated as the minimization of some network error function, over the training data set, by iteratively adjusting the weights. The mean square error between the target and actual output averaged over all output nodes serve as a good estimate of the fitness of the network configuration corresponding to the current input.

One of the major advantages of using EAs for optimizing functions is its ability to scan through the global search space simultaneously instead of restricting itself to localized regions of gradient shifts. Gradient-based algorithms are prone to getting trapped in a local minimum of the function, especially when the function is multimodal or non-differentiable. EAs work with a population of trial solutions, and hence the only information they need is some kind of an evaluation scheme to check the performance of the solutions. They continuously scan out the bad ones and allow the better ones grow.

Any EA requires a consistent representation of the function parameters. The genetic operators (crossover and mutation) have to be then decided in conjunction with the representation scheme. The training performance may vary from one scheme to another, and it is very likely that the operators may disrupt the genes in a particular representation, meaning not allowing the information to pass over to the next generation. This may happen when information pertaining to some functional unit (say a node in an ANN) is scattered in the gene. Randomly breaking the gene into two parts and interchanging one of them with another gene will disperse this information.

Figure 4: Binary representation of connection weights.



Encoding - a *genotype representation* - an ANN's weights can be done in two ways - binary and real valued. In either of the cases, it is just a concatenation of the network's weights in a string. In the binary form, each connection weight is represented by a bit string of certain length. These strings are concatenated to form what is called a *chromosome*. Each chromosome acts as an individual of the population which is then subjected to the genetic operators. This kind of a representation is simple and straightforward, but can result in very long chromosomes resulting in inefficient evolution. If too few bits are used, certain real valued weights may not get well approximated by discrete bits. Another choice is to somehow use a real valued chromosome and design genetic operations that can be applied to such chromosomes. In the next section we shall see one such scheme.

A real valued evolutionary process

A real valued representation of the weights can be done using an n -dimensional vector. The connection weight vector can be written as $U = (u_1, u_2, u_3, \dots, u_n)$.

where $u_1, u_2, u_3, \dots, u_n$ are the connection weights of the network. We will also be requiring what is called a *variance vector*. Let $S = (s_1, s_2, s_3, \dots, s_n)$ represent this variance vector. The importance of this vector will be made clear at a later stage. Each individual of our population will be a pair of these two vectors - (U_i, S_i) . Each individual defines an ANN.

A fitness value now needs to be assigned to each of these individuals. We shall be using the mean square error function to evaluate the fitness. Calculating the mean square error involves summing over the square of the differences between the actual and expected output at each node, and then taking the average over all output nodes.

$$\text{Error} = \frac{\sum (\text{Output}_{\text{actual}} - \text{Output}_{\text{expected}})^2}{\text{Number of output nodes}}$$

Our objective throughout the process would be to minimize this error. Hence, the fitness we assign shall be the inverse of this error - more the error, less the fitness. At this stage some form of recombination could occur, but we shall omit this step and take mutation as our primary genetic operator. For each member of the population, an offspring is created by mutation as follows.

$$s'_i = s_i \exp(\tau' N(0,1) + \tau N_i(0,1))$$

$$u_i' = u_i + s_i' N(0,1) \quad \text{for } i = 1, 2, 3, \dots, n$$

This scheme is known as the Gaussian mutation scheme. $N(0,1)$ is a normally distributed random value with mean 0 and variance 1. A new number is generated for every component (connection weight) of every population member. $N_i(0,1)$ is a similar random number but is generated only once for each population member. The values of τ' and τ are commonly set to:

$$\tau' = 1 / \sqrt{2 \sqrt{n}}$$

$$\tau = 1 / \sqrt{2n}$$

As we can see, the variance vector S_i has been used to decide the amount of shift the component weights should be given from their current value. As such, the variance values tell us how much the corresponding weight values are susceptible to deviate from their current value. Weights that are contributing well towards the convergence of the real and expected output are not changed much in their value.

The next stage is to evaluate the fitness of the offspring and generate the next population. The next population is generated using a method called *tournament selection*. For each individual in the set of parents and offspring, a fixed number of individuals are chosen uniformly from all parents and offspring. A comparison of the fitness value is carried out and the individual is assigned a “win” if its fitness is not smaller than the opponent’s. The individuals with the most wins are taken to form the next generation. The process is carried on until some halting criteria are satisfied.

Although EAs offer an attractive way to optimize ANN weights, they are relatively slow in local fine tuning in comparison to gradient methods. An obvious approach in this case would be to integrate a local gradient search with the EA. EAs global search ability can be used to locate a good region in the space and then a local search procedure can locate the near optimal point in that region. The EA will find out a good set of initial weights to start with for the local search algorithm. Since EAs are not sensitive to initial conditions as opposed to gradient methods, such kind of a *hybrid training* algorithm will be quite competitive in terms of faster convergence.

Evolution of Architecture

A neural network’s performance is highly dependant on its structure. The interaction allowed between the various nodes of the network is specified using the structure only. An ANN structure is

not unique for a given problem, and there may exist different ways to define a structure corresponding to the problem. Depending on the problem, it may be appropriate to have more than one hidden layer, feedforward or loopback connections, or in some cases, direct connections between the input and output layer. Hence, deciding on the size of the network is also an important issue. Too small a network will prohibit it from learning the desired input to output mapping; too big a one will fail to match inputs properly to previously seen ones and lose on the generalization ability.

Determining an optimal network topology has always been a problem. It is even impossible to prove that a given structure is optimal. Most often the structure is determined by trial and error methods. Different combinations of nodes and connections are tried out so that it gives maximum level of response within the given constraints. Such methods rely on overall performance of the network, so parts of the network that contributed well are difficult to identify. EAs can be helpful in this case, with their natural makeup of exchanging information. The search space here is also too big, similar architectures may have quite difference performance; different architectures may result in similar performance. This makes EAs a better candidate as opposed to algorithms which start with a maximal (minimal) network and then deletes (adds) layers, nodes or connections when necessary during training.

The genotype representation of a neural architecture is critical to the working of an evolutionary ANN design system. Considerations have to be taken so that the optimal structures are representable in it, meaningless structures are excluded, genetic operators yield valid offspring, and the representation do not grow in proportion to the network. Ideally, the representation should be able to span all potentially useful structures and omit unviable network genotypes. The encoding scheme also constrains the decoding process. For example, a neural network requiring a recurrent structure should have a representation expressive enough to describe recurrent networks. Also the decoding mechanism should be able to read this representation and transform it into an appropriate recurrent network.

Low level or *direct encoding* techniques mostly specify the connections only. *Indirect encodings* are more like grammatical rules; these rules suggest a context free graph grammar according to which the network can be generated. Direct encoded genotypes increase too fast in length with a growing network. Thus, the maximum topological space has to be limited by the user. This may exclude the fittest structure in the lot, or may result in networks with special connectivity patterns.

The performance of a network structure cannot be judged until it has been trained. Usually, the network is trained for a fixed number of steps and the performance is measured on an "evaluation set" data. By using different sets of data for training and evaluation, networks with better generalization abilities are given a preference in evolution.

One of the major challenges with evolving ANNs is to find a meaningful way to crossover disparate topologies. Usual genetic operators will fail to preserve the structural innovations occurring as part of the evolutionary process. Some kind of a speciation is required so that individuals compete primarily within their own niches, and not with the population at large. We shall now see a method, NeuroEvolution of Augmenting Topologies (NEAT), which uses methods for historical markings, speciation, and incremental growth from minimal structure for efficient evolution of network structure.

NeuroEvolution of Augmenting Topologies (NEAT)

A genotype in NEAT is made up of several connection genes. A connection gene contains information like the two nodes at the ends of the connection, the weight, and a special attribute called the *innovation number*. Apart from the connection genes, layer information (input, output, or hidden) regarding the nodes is also made a part of the genome.

An effort has been made to keep track of the historical origin of each gene, so that genes which do not follow from the same ancestor do not compete against each other. This has been achieved by assigning a global innovation number to each gene; this number is incremented each time a new gene appears. It should also be taken care that the offspring inherits the same innovation number on each gene. This allows that historic information is passed on over generations throughout the process of evolution.

The innovation number helps NEAT counter the problem of competitions between fundamentally different topologies. During the process of recombination, genes with same innovation number are lined up against each other; the ones from the more fit parents are inherited. Genes that do not find an innovation number match (disjoint or excess genes) in the other parent are passed on to the offspring.

Mutation in a neural architecture can happen in two ways - by adding a connection gene, or by adding a node gene. Two unconnected nodes can be connected by adding a new connection gene to the genome. An existing connection can also be spilt into two and a new node can be introduced in

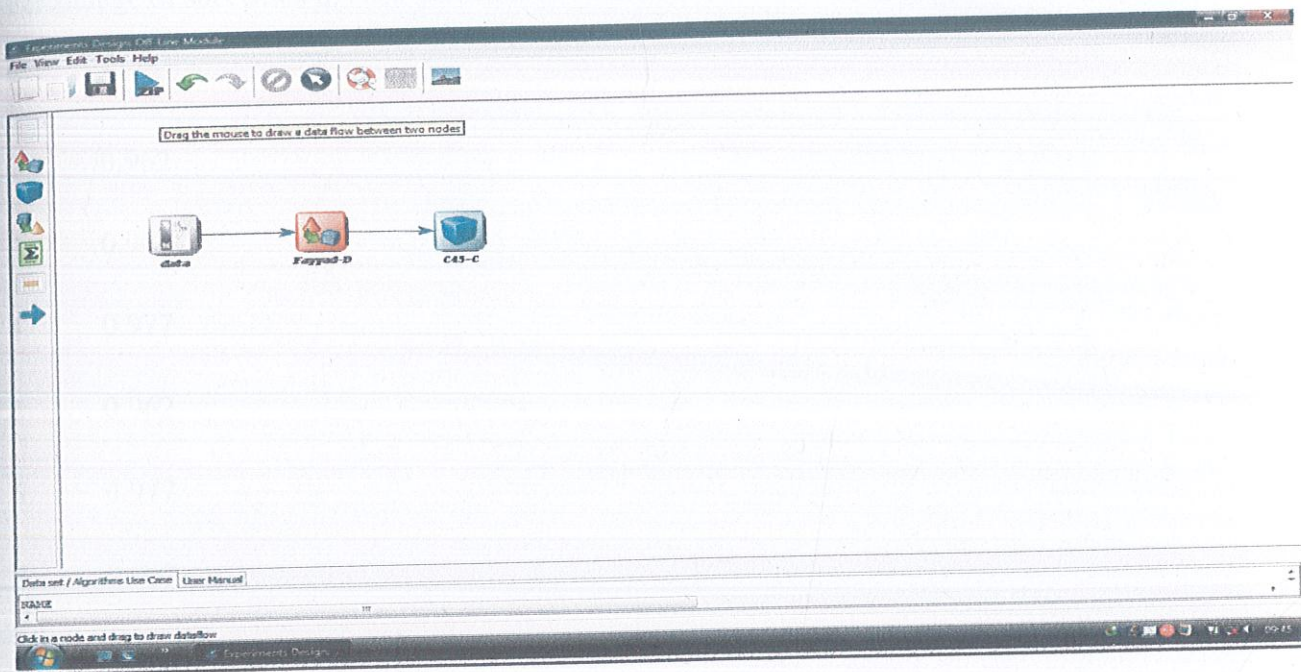
between. The old connection is disabled and two new connection genes are added. New nodes and connections can be very easily included into the structure using this methodology.

The selection of parents for mating determines what kind of topological features gets transferred to the offspring. If random selections are made then it is possible that structural innovations get lost during the breeding process. NEAT handles this problem by dividing the whole population into species depending on their topological similarities, and then mating the best performing individuals from each species to generate the offspring. The number of offspring generated depends on the overall fitness of the whole species in comparison to that of the others.

NEAT divides the population into different species on the basis of a compatibility distance measure. This measure is generally derived from the number of disjoint and excess genes between two individuals. If an individual's distance measure from a randomly selected one is less than a threshold value, then both individuals are placed into the same species. Once the classification is done, the original fitnesses are adjusted by dividing by the number of individuals in the species. A species grows if this average adjusted fitness is more than the population average, otherwise it shrinks in size. By doing so, NEAT does not allow any particular structure dominate over the whole population, but at the same time allows for the growth of the better performing ones.

RESULTS

Test Result 1:



- **Data Set Used:** Iris
- **Pre-processing algorithm:** Discretization—Fayyad-D
- **Data Mining Algorithm:** Decision Tree—C45-C

Relation:Iris

Set: training

Total percentage of successes:

0.963

Percentage of successes in each partition:

1 0.962

2 0.962

3 0.962

4 0.977

5 0.962

6 0.977

7 0.962

8 0.955

9 0.955

10 0.955

Confusion matrix (rows=real class;columns=obtained class):

50 0 0

0 25 0

0 0 75

Set: test

Total percentage of successes:

0.933

Percentage of successes in each partition:

1 0.933

2 0.866

3 0.933

4 0.933

5 0.933

6 0.933

7 0.933

8 0.933

9 1.0

10 0.933

Confusion matrix (rows=real class; columns=obtained class):

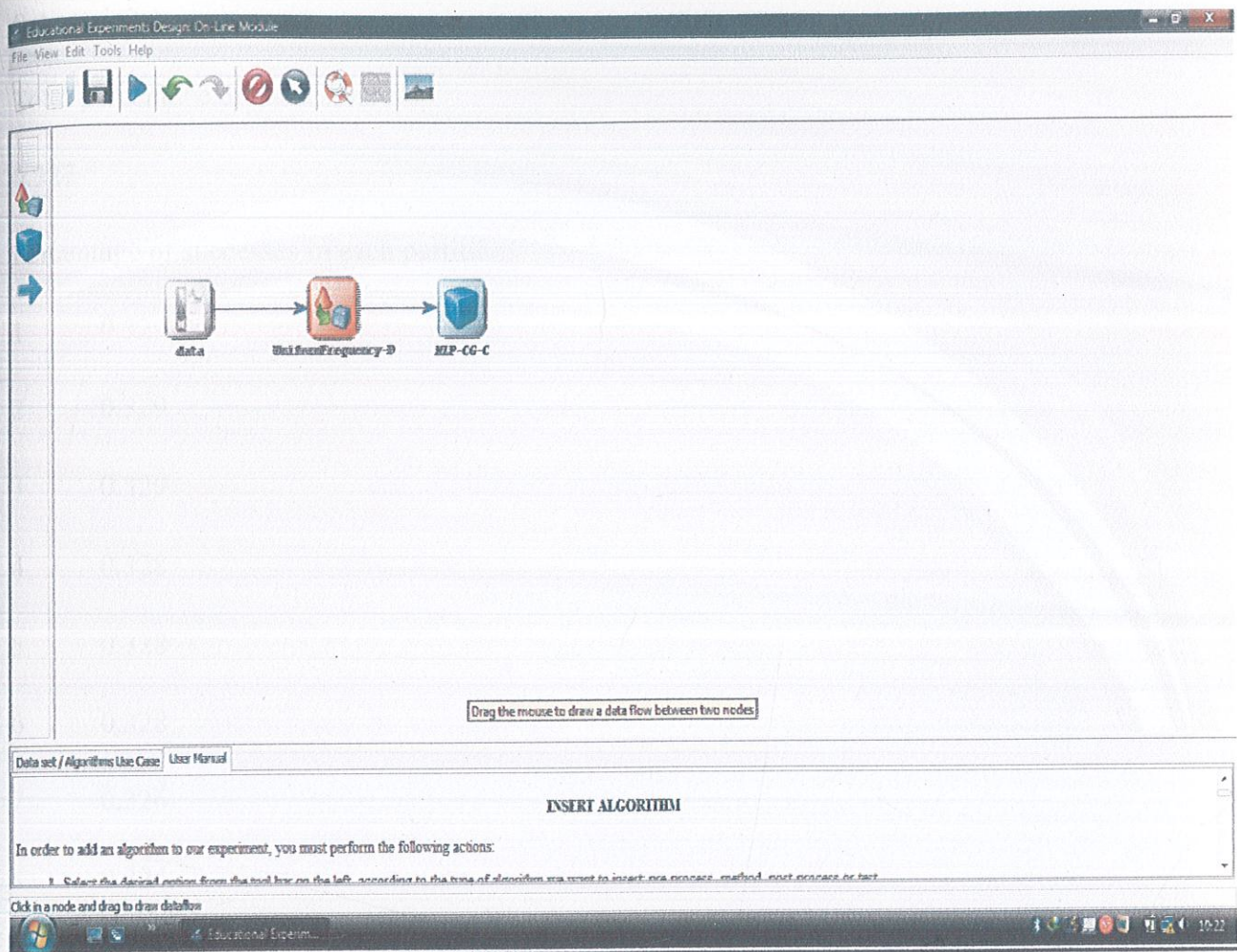
30 0 0

0 45 0

0 0 75

Run Time : 6.75

Test Result 2:



- **Data Set Used:** Glass
- **Pre-processing algorithm:** Discretization---UniformFrequency-D
- **Data Mining Algorithm:** Neural Networks---MLP-CG-C

Relation : glass

Set : training

Total percentage of successes:

0.327

Percentage of successes in each partition:

1 0.329

2 0.329

3 0.329

4 0.328

5 0.328

6 0.328

7 0.326

8 0.324

9 0.324

10 0.321

Confusion matrix (rows=real class;columns=obtained class):

630 0 0 0 0 0 0

684 0 0 0 0 0 0

153 0 0 0 0 0 0

0 0 0 0 0 0 0

117 0 0 0 0 0 0

81 0 0 0 0 0 0

261 0 0 0 0 0 0

Set : test

Total percentage of successes:

0.327

Percentage of successes in each partition:

1 0.304

2 0.304

3 0.304

4 0.318

5 0.318

6 0.318

7 0.333

8 0.35

9 0.35

10 0.388

Confusion matrix (rows=real class;columns=obtained class):

70 0 0 0 0 0 0

76 0 0 0 0 0 0

17 0 0 0 0 0 0

0 0 0 0 0 0 0

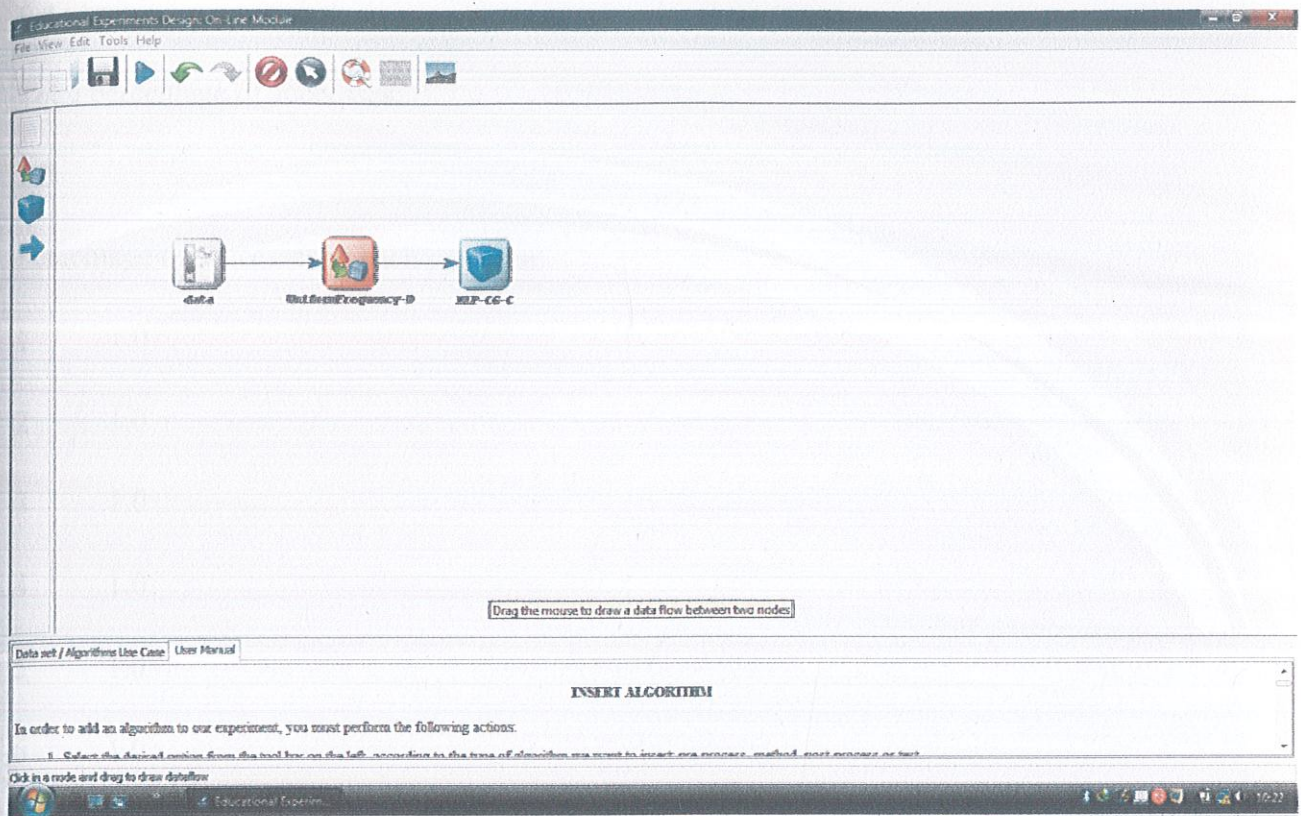
13 0 0 0 0 0 0

9 0 0 0 0 0 0

29 0 0 0 0 0 0

Run Time : 8.828s

Test Result 3:



- **Data Set Used:** Wine
- **Pre-processing algorithm:** Evolutionary Training Set Selection CHC- TSS
- **Data Mining Algorithm:** Evolutionary Neural Networks—NNEP-C

Relation : wine

Set : training

Total percentage of successes:

1.0

Percentage of successes in each partition:

1 1.0

2 1.0

3 1.0

4 1.0

5 1.0

6 1.0

7 1.0

8 1.0

9 1.0

10 1.0

Confusion matrix (rows=real class;columns=obtained class):

531 0 0

0 639 0

0 0 432

Set : test

Total percentage of successes:

0.949

Percentage of successes in each partition:

1 0.833

2 1.0

3 1.0

4 0.944

5 1.0

6 0.941

7 0.888

8 1.0

9 0.888

10 1.0

Confusion matrix (rows=real class;columns=obtained class):

57 2 0

2 66 3

1 1 46

Run Time: 1394.205 s

CONCLUSION

It has to be noted that the same input-output mapping can be implemented by different neural network architectures. Given a problem in hand, the topology for an ANN is not unique. Genotype representation of two structurally different ANNs will be different even though the functional mapping they define may be same. EAs are not able to detect these symmetries and hence a crossover in such a case would very often result in an unviable offspring. The search space is drastically increased and the efficiency of the operators is also severely affected. Moreover, in networks where more than one task needs to be learnt, there are chances of incompatible roles getting combined leading to similar problems. A simple solution to these problems would be to restrict the selection operator to smaller populations, and to introduce intuitive biased measures in crossover and mutation.

ANNs are capable of learning very high-order statistical correlations that are present in a training environment. Learning algorithms provide a powerful mechanism for generalizing behavior to new environments. However, the purpose of the networks then merely reduces to the simulation of a set of goals. Networks in which endogenous goals plays an important role in determining behavior are difficult to train using usual training algorithms. In such cases evolutionary methodologies are the appropriate mechanism for developing goals and purposeful behavior, rather than goal-directed explicit training.

The notion of an "optimal solution" in EAs is a bit deceptive. Fitness evaluations are all done with respect to the current population only; hence a good solution is better only in comparison to the members of the current population. There exists no way to check how good a solution is in the overall search space. The end conditions are also not explicit, and so the quality of the solution has to be a compromising act.

Global search methods like EAs are usually computationally expensive. They are more advisable to be used when little or no prior information about the network is available, and the performance value required out of the ANN is high. More effectively, they are good algorithms to start with the design and once some knowledge is gained, other purposive algorithms can come up with the solution faster. Parallel implementations of these algorithms will also become more and more purposeful as the need for designing real world applications arises. As the idea of natural adaptation become more and more promising, EAs and ANNs can serve as quite effective tools in narrowing

the gap between simulated and adaptive behavior. The direction of learning is going to be the one from which it can obtain the best advantage.

REFERENCES:

[http://en.wikipedia.org/wiki/Keel_\(software\)](http://en.wikipedia.org/wiki/Keel_(software))

<http://www.keel.es/>

http://en.wikipedia.org/wiki/Data_mining

http://en.wikipedia.org/wiki/Decision_tree

<http://www.red3d.com/cwr/evolve.html>

<http://www.amazon.com/Evolutionary-Computation-Basic-Algorithms-Operators/dp/0750306645>

Agard, B. and Kusiak, A. (2004), Data mining based methodology for the design of product families,

International Journal of Production Research, 42(15), 2955-2969.

Bäck, T., Fogel DB, Michalewicz Z., and Beck T. (2000a), Evolutionary Computation 1: Basic Algorithms and Operators, Institute of Physics Publishing, Bristol, UK.

Bäck, T., Fogel DB, Michalewicz Z., and Beck T. (2000b), Evolutionary Computation 2: Advanced Algorithms and Operators, Institute of Physics Publishing, Bristol, UK.

Coello, CA, Van Veldhuizen DA, and Lamont, GB (2002), Evolutionary Algorithms for Solving Multi- Objective Problems, Plenum Press, US.