CHAPTER - 4

METHODOLOGY AND TOOLS FOR ADSSAP

Solving problem in Case Based Reasoning involves gathering a problem description then calculating the similarity of the current problem with the previous problem stored in case base with the correct solutions. Similar cases are retrieved and adapted for differences in current problem and retrieved cases problem description. The solution suggested by the system is then evaluated. Based on the evaluation if required modification of the suggested solution is done, then the problem description and its solution can then be retained as a new case in case base and thus the system has learned to solve a new problem.

4.1 Case Based Reasoning Methodology

Case Based Reasoning (Schank et al. 1977; Waston et al. 1994) is based upon the concept that similar problems are best solved with similar solutions (Kolodner, 1993). It is able to utilize the specific knowledge of previously experienced, concrete problem situations or cases to solve new cases. At the highest level of generality, a general CBR cycle may be described by the following four processes (Aamodt et al., 1994).

![Figure 4.1: CBR Cycle Adapted from Aamodt & Plaza (Aamodt et al., 1994)]
• **Retrieve** the most similar case or cases. The Retrieve task starts with a (partial) problem description, and ends when a best matching previous case has been found. The goal of the matching task is to return a set of cases that are sufficiently similar to the new case given a similarity threshold of some kind, and the selection task works on the set of cases in case base and chooses the best match. Well known methods for case retrieval are Nearest neighbor, Induction, Knowledge guided Induction and Template retrieval.

• **Reuse** the information and knowledge in that case to solve the problem. The reuse of the retrieved case solution focuses on the differences among the past and the current case and identifying part of a retrieved case can be transferred to the new case. Two types of reuse concept works that are copy and adapt.

• **Revise** the proposed solution. When a case solution generated by the reuse phase is not correct an opportunity for learning from failure arises. This phase is called case revision and consists of two tasks: first evaluating the case solution generated by reuse, if successful, learning from the success. Otherwise repair the case solution using domain-specific knowledge.

• **Retain** the parts of this experience likely to be useful for future problem solving. This is the process of incorporating what is useful to retain from the new problem solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves selecting which information from the case to retain, in what form to retain it, how to index the case for later retrieval from similar problems, and how to integrate the new case in the memory structure.

4.1.1 **Case Retrieval**

The Retrieval process begins with a problem description, and finishes when a best matching past case has been found. The subtasks of retrieval are to Identify Features, Initially Match, Search, and Select, the identification task basically comes up with a set of appropriate problem descriptors, the purpose of the matching function is to return a set of cases that are nearly like the new case. Based on the
similarity threshold decided the selection procedure works on this set of cases and selects the best matching case or at least a first case to work on.

Syntactic similarity evaluation sometimes known as knowledge poor approach is used in domains where general domain knowledge is very difficult or tough to acquire. On the other hand, semantic oriented techniques generally known as knowledge intensive are able to use the contextual meaning of a problem definition in its matching, for domains where general domain knowledge is available. Well known methods for case retrieval are provided below. These methods can be directly used for case retrieval or combined in various combinations to form hybrid retrieval strategies.

- Nearest Neighbour
- Induction
- Knowledge Guided Induction
- Template Retrieval and etc.

4.1.2 Case Reuse

The retrieved cases from case base are reused by concentrating on two concepts first is identifying the differences between the past cases and the current case and then identifying which part of a retrieved case can be transferred to the new case. Two methods are used for it.

- Copy
- Adapt

4.1.2.1 Copy

In simple reuse tasks the differences are abstracted away considering non appropriate and similarities are considered appropriate. The output class of the retrieved case is considered to be the output class of new case as its solution. This is a simplest type of reuse.

4.1.2.2 Adapt

In Adapt process the systems have to take into consideration the differences between the past cases and current case hence the reused part cannot be directly
converted into the new case but requires an adaptation process that takes into account those differences. There are two main methods of reusing past cases through adapt as described below

- Transformational Reuse
- Derivalional Reuse

**Transformational Reuse**

The solution of the previous case is not directly a solution for the new case but there is some knowledge in the form such that when some transformational operators $T$ is applied on that knowledge then it can be converted into the old solution for the new case. A way to arrange these transformational operator $T$ is to index them around the differences identified between retrieved and current cases.

**Derivalional reuse**

Derivalional reuse finds out that how the problem was solved in the retrieved case. The retrieved case stores the information related to the method used for solving the retrieved problem, justification for using any operators, sub objectives considered, alternate solution produced, failed search paths, etc.

### 4.1.3 Case Revision

When the solution generated by reuse for case is not correct then learning from this incorrect solution takes place. This procedure is called case revision and performs two tasks. First step is evaluating the solution generated by the case. If the solution is correct then learning from the case solution i.e. retainment takes place otherwise repairing of case solution take place using domain specific knowledge.

#### 4.1.3.1 Evaluate solution

The evaluation of the solution generated for the case is done by applying the solution in the real conditions or by asking a teacher. This step is usually outside the CBR system, since it require the application of suggested solution to the actual problem. The results generated from applying the solution may take some time to appear, based on the type of application.
4.1.3.2 Repair fault

Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them using domain specific knowledge.

4.1.4 Case Retainment

In the process of Retainment the useful information from the new problem solution is retain into the case base of the existing knowledge. The learning from success or failure of the solution is based on the evaluation and possible repair of the results generated by case reuse. It requires choosing which information from the case to retain, in what form to retain and how to retain and create index for later use from similar problems. and how to store the new case in the memory.

4.1.4.1 Extract

In Case Based Reasoning the case base is updated based on the way the problem was solved. If problem was solved by use of a previous case then a new case may be created or the generalisation of old case may be included in the present case. If the problem is solved by asking user or by other methods then an entire new case is constructed. In any case, resource of learning is required to be identified and explanation or justification for the solution to the problem may also be included in the new case.

4.1.4.2 Index

The indexing problem is a main and much targeted issue in case-based reasoning. It is required to decide that what type of indexes should be created and used for future retrieval and how to the search space of indexes should be structured. This is basically problem of knowledge acquisition and should be analysed as domain knowledge analysis and modelling step. A simplest solution to the problem is to create indices for all input features. This is a technique of syntax based methods within memory based reasoning.

4.1.4.3 Integrate

In this final step the new case knowledge is use to update the knowledge base. If no new case and index is created then it becomes the main step of Retain. By
modification of the indices of existing cases, Case Based Reasoning systems learn to assess the similarity in better way. The fine tuning of existing indexes is an important part learning in Case Based Reasoning. Based on the failure or success of a typical case solution importance of a particular solution or Index strengths are adjusted. The association for features that are identified as useful for retrieving a successful case is strengthened, while the association for features that lead to unsuccessful cases being retrieved is weaken. In this way, the index structure has a role of tuning and adapting the case memory to its use.

4.2 Case Based Reasoning Approaches

There are three Case Based Reasoning approaches

4.2.1 Textual CBR Approach

Case acquisition in easier in textual CBR approach. This approach is useful in domain where large collections of know-how documents already exist and the user is able make use of knowledge contained in the documents. The approach is well suited in the conditions where there are few couple of hundred cases at a time and when each case has a short description.

4.2.2 Conversational CBR Approach

The conversational CBR approach is useful for high volume of simple problems that are solved repeatedly. The customer is guided by the system with predefined dialogs. However, the case base is manually organized by the case author, it is complex and costly work when the cases have many attributes to describe. The conversational approach is used for applications in which only a few questions are needed for making decisions.

4.2.3 Structural CBR Approach

The structural CBR approach is well suited for cases that are described with predefined attributes and values. In various structural Case Based Reasoning systems, attributes are organized as tables or as sets of tables with relations, or structure in object oriented technique. The structural CBR approach is useful for the areas where additional knowledge along with cases, must be used in order to produce good results. The domain model makes sure that new cases have low maintenance effort
and are of high quality. This approach gives better results than other methods described but it requires an investment to produce the domain model.

4.3 Challenges with Knowledge Representation in CBR

There is no Case Based Reasoning technique appropriate for each type of application. The biggest problem in Case Based Reasoning is to identify techniques that is fit for problem solving and learning in specific domains and for specific application.

The representation problem in Case Based Reasoning is mainly the issue of deciding what should be stored as a case in case base and identifying a proper structure for describing the case and determining how the case memory should be indexed and organized so that effective retrieval and reuse of cases can be done. Two models for storing case memory are

- Dynamic memory model of Schank and Kolodner
- Category-exemplar model of Porter and Bareiss

4.3.1 The Dynamic Memory Model

The concept is to arrange specific cases that share identical qualities under a general structure called a Generalized Episode(GE) that contain three different types of objects:

- Norms: Common features that present in all cases indexed under a GE.
- Indices: Are discriminating features for each GE cases. An index may provide link to a more specific generalized episode or directly to a case. Index have two values index name and index value.
- Cases: Are the information stored.

When a new case detail is given and the best match is searched then input case structure is pushed down the network structure starting from the root node. The search process is identical for retrieval and storing of cases. The GE case with most features in common with the input case is found. During saving of a new case, if a feature of the new case is exactly same as a feature of an existing case, a generalized episode is created. The two cases are then listed separately under different indices
below this generalized episode. If two cases or two GEs end up under the same index, a new generalized episode is created automatically thus making the memory structure is dynamic in nature.

### 4.3.2 The Category & Exemplar Model

The PROTOS system, built by Bareiss, (1998) proposes to organise the case in case memory by an alternative way. Here cases are called as exemplars. Various features are assigned different degree of importance in for a case's membership to a category. Generalisation attempt of any set of cases is done very cautiously. This concept representation of cases forms the basis for this memory model.

The case memory is created using a network structure of categories, cases, and index pointers and a category is associated with each case. An index may point to a category or case. There are of three kinds of indices
- Feature links pointing from problem features to cases or categories.
- Case links called exemplar links pointing from categories to its associated cases.
- Difference links points from cases to the neighbour cases with a small difference number of features between them.

A feature is expressed by a name and a value. A category's exemplars are sorted based on the degree of similarity in the category. Categories are inter linked within this memory organization by semantic network which also contains the intermediate states and features referred.

Finding a case in the case base based on input description is done by joining the input features of problem case into a pointer to the case or category that have most of the features as common. The links to best matching cases are traversed, and values are returned from these cases. A new case is stored by creating appropriate feature indices for case. If a new case have only minor differences to the input case then the new case is not retained or the two cases may be merged by creating links in the semantic networks.
4.4 Machine Learning in AI Techniques

Machine learning is the subfield of computer science that gives computers the ability to learn without being designing the programs. Machine learning concentrates on the development of programs that can learn by themselves to grow and change when exposed to new data. It has been created for research in field of pattern recognition and computational learning in artificial intelligence. Machine learning concentrates on study of algorithms that can learn from the data and make predictions of new data. Machine learning is employed in tasks where designing and programming of algorithms is not feasible.

4.4.1 Supervised Learning

Supervised Learning is the machine learning task of finding the solution by inferring from labelled training data. In supervised learning training data consist of a set of examples. Each example a pair consist of an input values and output value. A supervised learning algorithm uses training data to analyse and produces solution in the form an inferred function, which is used to get solution for new cases. An ideal situation will allow the algorithm to correctly find the class labels for new unseen cases. It require that learning algorithm perform generalization task on training data in reasonable way so that unseen cases can be solved.

Few approaches used supervised machine learning tasks are

- Logistic Regression
- Decision Tree
- Naive Bayes classifier
- Nearest Neighbor

4.4.2 Unsupervised Learning

In Unsupervised Learning no labels are provided to the learning algorithm. Finding structure in the input by itself is the task of algorithms or methods. Unsupervised learning can be used to discover hidden patterns in data. Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
Few approaches for unsupervised learning include:

- Clustering
- K Means
- Neural Networks
- Approaches for learning latent variable models

4.4.3 Semi Supervised Learning

Problems where you have a large amount of input data $X$ and only some of the data is labelled $Y$ are called semi-supervised learning problems. These methods sit in between both supervised and unsupervised learning. Many real world machine learning problems fall into this area. This is because it can be expensive or time-consuming to label data as it may require access to domain experts. Whereas unlabelled data is cheap and easy to collect and store. Hence class of unsupervised learning tasks and techniques are used for large amount of unlabelled data for training and supervised learning is used on a small amount of labelled data. Many machine learning researchers have found that improvement in learning accuracy can be increased by using large amount unlabelled data together with a small amount of labelled data. Semi supervised learning methods are of practical value. Few semi supervised learning models are as described.

- Self Training
- Generative Models
- Graph-Based Algorithms
- Multiview Algorithms

4.4.4 Reinforcement Learning

A computer program communicates with the changing environment and perform certain objective without a teacher to identify whether it has come close to its goal. Example of reinforcement learning is to play a game against an opponent and learn from it.

Reinforcement learning is an area of machine learning motivated by behaviourist psychology. It concentrates on identifying the methods so that software agents takes actions in an problem domain to maximize the collective rewards. Due
to its generality behaviour this concept is studied in many fields such as game theory, operation research, control theory, multi agent systems, simulation based optimization, statistics, swarm intelligence and genetic algorithms.

Reinforcement learning varies from conventional supervised learning in that correct input/output pairs are not given, neither sub task for optimization are explicitly corrected. Further attention is paid on online performance, which includes finding a balance between discovery of unknown fields and exploitation of current knowledge. Few techniques of reinforcement learning are as below

- Q Learning
- Sarsa
- Monte-Carlo Methods

4.5 Classification and prediction Algorithms

Various algorithm are present for classification and prediction tasks. An overview of various algorithm present for classification and prediction tasks is presented below.

4.5.1 ZeroR

ZeroR is the classification technique that relies on the target solution and ignores all predictors. ZeroR classification just predicts the majority class in the data. Although ZeroR have no predictability power then also it is useful for determining a baseline performance mark for other classification methods. A frequency table is constructed for the sample cases and the most frequent value is selected.

4.5.2 Linear Regression

Straight-line regression analysis includes two variables first is a response variable y, and second is a pedictor variable x. It is the easiest form of regression, and defines y as a linear function of x.

That is,

\[ y = a + cx \]  \quad (4.1)
where the variance of $y$ is assumed to be constant, and $a$ and $c$ are regression coefficients $a$ specifies the $y$ intercept and $c$ specifies slope of the line.

The regression coefficients $c$ and $a$ can also be thought of as weights, and can be equivalently written as,

$$y = w_0 + w_1 x$$  

(4.2)

Least squares method can be used to solve these coefficients which estimates the best fitting straight line by minimizing the error between the actual data and the line estimate.

4.5.3 Logistic Regression

Logistic Regression (Le et al. 1992) is a probabilistic statistical classification algorithm. It is a form of is a form of parametric regression (Zikus et al. 2002). It allows predicting a class from a set of variables that can be continuous, discrete, nominal or a combination. The algorithm is presented by Agresti (1996). It is used to predict the class label based on the values of one or more predictor variables. Logistic regression gives us a linear classifier. The decision boundary separating the two predicted classes is the solution, which is a point if $x$ is one dimensional, a line if it is two dimensional, etc.

The formula below is stated as the probability of occurrence of an event $E = 1$

$$\log \left( \frac{p}{1-p} \right) = B_0 + B_1 X$$  

(4.3)

$B_0 + B_1 X$ is line regression equation where expected probability of occurrence of an event $E = 1$ for a given value of $X$ is

$$p = \frac{e^{(B_0 + B_1 X)}}{1 + e^{(B_0 + B_1 X)}}$$  

(4.4)

4.5.4 k-Nearest-Neighbor

Nearest neighbor classifiers are based on the concept of learning by similarity, that is, by evaluating the given test cases with training cases that are similar to it. The training cases are described by $n$ features. Each case represents a point in an $n$-dimensional space. Thus all training cases are stored in an $n$-dimensional
plane. When given an unknown case a k-nearest-neighbor classifier tries to find out the k training cases that are close to the unknown case.

These k training cases are called the k nearest neighbors of the unknown case. The nearest neighbor value is calculated in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or case is

Let $X_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ be first case

Let $X_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, be second case

Distance between $X_1$ and $X_2$ is

$$ \text{dist}(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \tag{4.5} $$

For each numeric feature, we take the difference between the values of that attribute in case $X_1$ and case $X_2$, then find the square of the difference, and accumulate it. The square root is calculated for the total accumulated distance value.

**4.5.5 Decision Tree**

General Decision Tree algorithm is provided below changes will be there as per the actual method implemented. It generate decision tree. Generate a decision tree from the training tuples of data partition D.

**Algorithm:** Decision_Tree_Generation_Algorithm

**Step 1:** Create N as a node;  
**Step 2:** If records in partition D are of similar class C then  
**Step 3:** Return N as a leaf node with class labeled C;  
**Step 4:** If no attributes are present then  
**Step 5:** Return N as leaf node with label of class having majority records in D; // majority voting  
**Step 6:** Apply Attribute_selection_method (D, attribute list) to find the best Splitting Attribute from list of Attributes;  
**Step 7:** Label node N with Splitting Attribute;
**Step 8:** If splitting attribute have discrete value and Multiway splits allowed then

**Step 9:** Attribute_list = Attribute_List- Splitting Attribute; // remove splitting attribute

**Step 10:** For each value j of splitting attributes // partition the records and generate subtrees for each partition

**Step 11:** Let Dj be the set of data records in D satisfying value j; // a single partition

**Step 12:** If Dj is empty then

**Step 13:** Attach a leaf node labeled with the majority class in D to node N;

**Step 14:** Else attach the node returned by Generate decision tree (Dj, attribute list) to node N;

**Step 15:** Endfor

**Step 16:** Return N;

**Figure 1.5:** Algorithm Decision Tree Induction adopted from Data Mining Concepts and Techniques by Jiawei Han (Han et. al, 2006)

### 4.5.6 Simple Cart

Simple CART stands for Classification and Regression Trees (Breiman et al. 1984). It is one of the decision tree algorithms. It does the feature selection by applying minimum cost complexity pruning. Cart can in two forms as classification tree or as a regressive tree based on the type of the response variable. If it is categorical the classification tree is made else if continuous then regression tree is made. Splitting algorithms are different for classification and regression trees. CART methodology consists of three parts:

- Construction of tree
- Choice of the right tree size
- Classification of new data using constructed tree

The algorithm constructs binary trees hence each internal node has exactly two outgoing edges. Classification tree is built by splitting rule that is each time data is divided into two parts with maximum homogeneity.
Classification Tree

Classification trees constructed using learning samples for which the class of that sample is known in advance. For example

Let $t_p$, $t_l$, $t_r$ - parent, left and right nodes,

$x_j$ - where x is a case and variable j

$x_j^R$ - best splitting value of variable $x_j$.

Maximum homogeneity of left and right child nodes will be equivalent to the maximization of change of impurity function $\Delta i(t)$:

$\Delta i(t) = i \left( t_p \right) - E \left[ i \left( t_c \right) \right]$ (4.6)

where $t_c$ - left and right child nodes of the parent node $t_p$. Assuming that the $P_l$, $P_r$ - probabilities of right and left nodes, we get:

$\Delta i(t) = i \left( t_p \right) - P_l i(t_l) - P_r i(t_r)$ (4.7)

CART solves the following maximization problem:

$x_j \in \mathcal{X}_j \Rightarrow i \left( t_p \right) - P_l i(t_l) - P_r i(t_r)$ (4.8)

In theory there are several impurity functions, but Gini splitting rule and Twoing splitting rule are widely used.

Regression Tree

Regression trees have no classes. It has response vector $Y$ which represents the values for each observation in variable matrix $X$. Splitting task in regression trees is made based on squared residuals minimization algorithm which implies that expected sum variances for two resulting nodes should be minimized.

$x_j \in \mathcal{X}_j \Rightarrow \arg \max \left[ \frac{\text{Var}(Y_l) - \text{Var}(Y_r)}{2} \right] (4.9)$

where $\text{Var}(Y_l), \text{Var}(Y_r)$- reponse vectors for corresponding left and right child nodes

$x_j \in \mathcal{X}_j \Rightarrow \text{optimal splitting question which satisfies the condition}$
4.5.7 ID3

The ID3 (Colin et al., 1996) classification algorithm is simple. ID3 creates a decision tree from a set of examples. The tree created is used to classify new sample cases. The example cases have several features and belong to a class may be yes or no. The leaf nodes have a class name whereas where as a nonleaf node is a decision node. The decision node have a feature test attached with each branch based on a possible value of one of the feature. ID3 uses information gain values to decide which feature will be evaluated at a decision node. The advantage of decision tree learning is that a program generates the knowledge rather than a knowledge engineer.

Gain(S, A) is information gain of example set S on attribute A is defined as

\[ \text{Gain}(S, A) = \text{Entropy}(S) - \sum \left( \frac{|S_v|}{|S|} \times \text{Entropy}(S_v) \right) \]  

Where: \( \sum \) is each value v of all possible values of attribute A

\( S_v = \) subset of S for which attribute A has value V

\( |S_v| = \) number of elements in \( S_v \)

\( |S| = \) number of elements in \( S \)

4.5.8 Naïve Bayes Classification

Naïve Bayes classifier uses the concept of probabilities to perform the classification task based on probabilities. It is assumes that variables probability is independent of each that is the any attribute value is not related to presence and absence of any other attribute. Naïve Bayes classifiers is supervised learning algorithm and can be trained very efficiently. The parameter estimation of this method uses the maximum likelihood concept in many applications. One can work with the naïve Bayes model without believing in Bayesian probability or using any Bayesian methods.

Let there be m Classes i.e. \( C_1, C_2, C_3 \ldots C_m \)

Naïve assumes that class is conditionally independent

\[ P(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i) \]  

\[ P(X|C_i) = P(X_1|C_i) \times P(X_2|C_i) \times \ldots \times P(X_n|C_i) \]
4.5.9 Decision Table

Decision Table algorithm (Kohavi, 1995) is found in the Weka classifiers under Rules. It comprises of two parts: schema and body. The schema describes the list of attributes while body describes labeled instances from schema features. The classifier of decision table is based on the concept of simple lookup table. There exist two types of decision table classifiers called Decision Table Majority (DTMaj) and Decision Table Local (DTLoc). If the new instance matching cell in the decision table is empty (i.e. without training instance), then DTMaj returns training set while DTLoc searches for entry in the decision table with larger cells (i.e having few matching attributes) and an answer is returned from native region.

4.5.10 Decision Stump

A one-level decision tree called Decision Stump (Iba et al. 1992) has been developed such that single root node is connected to leaf node. Decision Stump is also called 1-rules due to its use of single input feature value for making prediction.

Different variants of trees can be created based on the input feature. The decision tree for nominal features either comprises of single leaf for each possible value of feature or two leaves such that one leaf represents some chosen category and second leaf for the remaining categories. Whereas the decision tree for continuous features select some threshold value for feature and comprises of two leaves, one leaf for below threshold values and other leaf for above threshold values. However, rarely, multiple thresholds may be chosen and the stump therefore contains three or more leaves.

![Decision Stump Algorithm](image)

**Figure 4.2**: Example of Decision Stump Algorithm

4.5.11 DTNB Algorithm

DTNB algorithm (Hall et al., 2008) is based on simple Bayesian network which represents conditional probability in a Decision table. The attributes are split by Learning algorithm into disjoint subset of decision table and naïve Bayes. The
authors used Forward selection method such that all the attributes are initially modeled by Decision table. Then Naïve Bayes is used to model selected attributes at each step and Decision table to model remainder attributes.

An attribute is dropped from the model entirely at each step in the algorithm. The overall estimates for class probability can be generated by combining estimates for class probability of the NB and DT.

Assuming $X^\top$ is the set of attributes in the DT and $X^\bot$ the one in NB,

Overall class probability is computed as

$$Q(y/X) = \alpha \times Q_{DT}(y/X^\top) \times Q_{NB}(y/X^\bot)/Q(y)$$

(4.13)

Where $\alpha$ is a normalization constant, $Q_{DT}(y/X^\top)$ represents the estimates of class probabilities obtained from the DT and $Q_{NB}(y/X^\bot)$ represents the estimates of the class probability obtained from NB, and $Q(y)$ is prior probability of class.

### 4.5.12 Neural Networks

The neural network consists of connected input/output units such that weights are associated with each connection. The correct class label of input tables can be predicted by weight adjustments in the network during learning phase. Due to the presence of connections between units neural network is also called connectionist learning.

Neural Networks require long training times thus has its applicability where it is feasible.

![Diagram of perceptron of Neural Network](image)
Where \( X_i \) is \( i \)th input

\( W_{ij} \) is weight attached to \( X_i \) input

To compute the net input to the unit \( j \) is

\[
I_j = \sum_{i=0}^{n} W_{ij} X_i + \theta_j
\]  

(4.14)

### 4.5.13 Support Vector Machines

A promising new method has been developed for classifying both linear as well as nonlinear data called Support Vector Machines (SVM). The algorithm is as follows.

The original training data is transformed into higher dimension using nonlinear mapping. It searches within new dimension a linear optimal that separates hyperplane with nonlinear mapping to data of high dimension from two class hyperplane. The SVM finds this hyperplane using support vectors (training tuples) and margins (defined by the support vectors).

### 4.5.14 Genetic Algorithms

The genetic algorithm is developed to evolve population of candidate solution to optimization. A set of properties exists for each candidate solution which can be altered and mutated. The solutions traditionally can be represented as binary strings of 0s and 1s, but other encodings can be used.

The evolution is an iterative process which starts from population of randomly generated individuals. In each generation i.e. population in each iteration, fitness is evaluated for every individual in the population. The fitness describes the objective function value in the optimization problem solved. The fittest persons are stochastically selected from current population and modification i.e. recombination and possibly random mutation of each individual genome forms a new generation. The new generation thus created of candidate solutions is then used for the next iteration of algorithm. The algorithm is terminated when either the fitness level has been reached satisfactorily for the population or maximum number of generations produced.
A typical genetic algorithm requires

- Solution domain’s genetic representation,
- Evaluation of solution domain using fitness function.

Each candidate solution has a standard representation through array of bits. Similarly arrays of other types and structures can be used for representation of candidates solution. The main characteristics to make convenient genetic representations are easy alignment of their parts due its fixed size which easily supports its crossover functions

4.6 WEKA as a Data Analysis and Prediction Tool

Weka is a workbench (Hall et al., 2009) which contains algorithms and visualization tools for predictive modeling and analysis of data along with graphical user interfaces for accessing the functions easily. This technology supports wide range of applications and it have capabilities to become the main component of intelligent information system. It has enabled compact generalizations including recorded information of large databases.

The original version of WEKA (Waikato Environment for Knowledge Analysis) was originally design as a tool for data analysis from agricultural domains. The recent version has started its development in 1997 which is fully Java based. It has its wide applications now including research and educational purposes.

The development of workbench is also supported to provide integrated environment that helps in accessing different machine learning techniques and in incorporating essential tools for pre-processing and post processing that we found useful for real world data sets.

The GNU General Public License has made free availability of workbench. WEKA has supported various standard data mining tasks including data pre-processing, classification, clustering, visualization, regression and feature selection. The prediction of all WEKA techniques is based on the assumption that a single flat file or relation provides the data as fixed number of attributes (numeric, nominal or other attribute types).
4.7 Summary of Chapter

Chapter throws light case based reasoning method in detail. It also talks about CBR cycle. After that various classification and prediction algorithms are discussed in detail. This algorithm helps machine learn from the given known set of data and then use the learning for classifying and predicting unknown data. In depth details of case based reasoning and classification methods provide us the knowledge and insight to identify the most suitable methods to solve the problems. In the last an overview of weka tool is provided.