3. DIFFERENT TECHNIQUES FOR LOAD FORECASTING

3.1 Introduction

Load forecasting has been in existence for many decades, and a wide variety of techniques have been developed. Owing to its growing importance in smart grid utilities, research in this area has been resulting in the development of various forecasting techniques. In general, load forecasting techniques can be classified as either traditional or modern techniques.

Traditional Techniques: Time series and multiple regression models are the most widely considered examples of traditional techniques for the short term load forecasting. Historical load data is considered in the former, while the relations between load and the corresponding variation factors are analysed in the later. These statistical techniques are mature and mathematically proven, but most of them are linear ones.

Modern Techniques: With power system growth and the increase in their complexity, many factors have become influential to the electric power generation and consumption. Also, with the recent developments of smart grid implementations, there are new factors coming into picture that are influencing the electric demand. The relationship between the load and its exogenous factors is complex and includes non-linear characteristics, making it quite complex to model through traditional techniques. Besides not giving the required precision, most of the traditional techniques are not robust enough. Thus, the modern/intelligent approaches including expert system, knowledge system, especially artificial neural networks have been practiced recently in the area of short term load forecasting. In the recent past, support vector regression technique based on statistical theory and wavelets have gained importance in short term load forecasting applications. Now-a-days, these modern techniques are gaining practical importance and have been widely used in electric utilities to do the short term load forecasting. Some of the most popular traditional and modern techniques are discussed in this chapter.

3.2 Traditional/Statistical Techniques

3.2.1 Regression

Regression is a very widely used statistical technique in load forecasting and it is often easy to implement. The modelling is performed by usually considering the
relationship of electric load consumption and other factors such as weather conditions, day types, and random disturbances. This method considers that the load can be represented as a combination of a standard load trend and a trend linearly dependent on some factors that affect the load [18].

The mathematical model representing the regression method can be written as:

\[ \hat{L}(t) = L_n(t) + \sum_{i=1}^{N} a_i x_i(t) + \varepsilon(t) \]  

(3.1)

where

- \( \hat{L}(t) \) is the forecasted load at time \( t \),
- \( L_n(t) \) is the normal load at time \( t \),
- \( a_i \) is the estimated coefficient,
- \( x_i(t) \) are the independent variables such as weather effect,
- \( \varepsilon(t) \) is a white noise component and
- \( N \) is the number of observations.

The steps for the regression approach is described below

- select appropriate variables and data, that influence the load
- assume the functional relationship between inputs and outputs
- find proper coefficients for this functional relationship

3.2.1.1 LINEAR REGRESSION

Linear Regression is the most simple and common method for forecasting the load, affected by various factors ranging from weather, population growth, electricity prices, economic growth etc. The linear regression model can be represented as:

\[ \hat{L}(t) = \beta_0(t) + \beta_1 x_1(t) + \beta_2 x_2(t) + \ldots + \beta_m x_m(t) + \varepsilon \]  

(3.2)

where,

- \( \hat{L}(t) \) is the forecasted load at time \( t \),
- \( \beta_i \) are the regression coefficients corresponding to \( x_i \) and
- \( \varepsilon \) is an error term.
This model assumes that the error term $\varepsilon$ has a zero mean value and constant variance. The regression coefficients ($\beta_i$) of the model are calculated from past load data and the affecting factors.

Let the regression coefficients $\beta_i$ ($i=0, 1, 2, \ldots m$) be estimated in terms of $b_i$ ($i=0, 1, 2, \ldots m$). Then, equation (3.2) can be represented as

$$
\hat{L}(t)=b_0(t)+b_1x_1(t)+b_2x_2(t)+\ldots+b_mx_m(t) \tag{3.3}
$$

The parameters $b_i$ can be obtained using the least square estimates method, which minimizes the residual sum of squared errors (SSE) [148], from

$$
B = [b_0 \ b_1 \ b_2 \ \ldots \ b_m]^T = (X^T X)^{-1} X^T L \tag{3.4}
$$

where $L = \begin{bmatrix} \hat{L}_1 \\ \hat{L}_2 \\ \vdots \\ \hat{L}_n \end{bmatrix}$ and $X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1m} \\ 1 & x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nm} \end{bmatrix}$ \tag{3.5}

### 3.2.2 Time Series

Time series data usually contain different components like trend, cyclical pattern, seasonal component and random component. Initially, a pattern is established with the available data and then the model is created to forecast the value with respect to time. The time series model characteristics can be represented as [149]

$$
\hat{L}(t)=L_n(t)+S(t)+R(t) \quad t=-1, 0, 1, 2, \ldots \tag{3.6}
$$

where

- $\hat{L}(t)$ is the forecasted load at time $t$,
- $L_n(t)$ denotes the standard load value,
- $S(t)$ is the seasonal component and
- $R(t)$ is the random component.

In these methods, the load pattern is considered as a time series signal with known seasonal, weekly and daily periodicities. These periodicities predict the load roughly at the given season, day of the week and time of the day. The difference between the forecasted
load and the actual load can be treated as a stochastic random process, which can be analyzed by using techniques like Kalman filter method and Box-Jenkins method.

3.2.2.1 KALMAN FILTER METHOD

Towards the end of 1980s, as computers became more powerful, it became possible to record longer time series and apply more complex algorithms to them. Kalman filtering techniques characterizes dynamical systems by a state space representation. The theory of Kalman filtering provides an efficient computational (recursive) means to estimate the state of a process in a way that minimizes the mean of the squared error. The filter supports estimation of the past, present and even future states and it can do so even when the precise nature of the modeled system is unknown [150]. A significant challenge in the use of Kalman filtering based methods is the estimation of the state space model parameters. This requires the estimation of the covariance matrix. Because of the high non-stationarity of the load pattern, it is difficult to estimate the covariance matrix accurately.

3.2.2.2 BOX-JENKINS METHOD

Autoregressive Moving Average models, as described by the Box-Jenkins methodology, are a very rich class of possible models. The assumptions for this class of models are,

- the series is stationary or can be transformed to one using a simple transformation such as differencing
- the series follows a linear model

The original Box-Jenkins modeling procedure involves an iterative three stage procedure of model identification, model estimation and model validation. Later work presented in [151], includes a preliminary stage for data preparation and a final stage for forecasting.

- Data preparation can involve several sub steps. If the variance of the series changes with the level then logarithms are applied on the data to transform to constant variance series. Similarly, the data needs to be checked if it is stationary and if there is any significant seasonality which needs to be modeled. Differencing approach enables to handle stationarity and remove seasonality.
In model identification step, the order of the autoregressive and moving average terms need to be identified to obtain a good fit to the data. Several graph based approaches exist, which include the autocorrelation function and partial autocorrelation function approaches and new model selection tools such as Akaike’s Information Criterion have been developed.

In model estimation step, the value of the model coefficients are determined in order to obtain a good fit on the data. The main approaches are non-linear least squares and maximum likelihood estimation.

Model validation involves testing the residuals. As the Box-Jenkins models assume that the error term should follow a stationary univariate process, the residuals should have nearly the properties of independent and identically distributed normal random variables. If the assumptions are not satisfied, then a more appropriate model needs to be formulated. The residual analysis should hopefully provide some clues on how to develop a more appropriate model.

### 3.2.2.2.1 Autoregressive Model

An Autoregressive model of order $p \geq 1$ is defined as

$$X_t = b_1 X_{t-1} + \ldots + b_p X_{t-p} + \varepsilon_t$$  \hspace{1cm} (3.7)

where

$b_i$ are the parameters and

$\{\varepsilon_t\} \sim N(0, \sigma^2)$, known as white noise.

This model can be written as an AR (p) process. The equation explicitly specifies the linear relationship between the current value and its past values.

### 3.2.2.2.2 Moving Average Model

A Moving Average model of order $q \geq 1$ is defined as

$$X_t = a_1 \varepsilon_{t-1} + \ldots + a_q \varepsilon_{t-q} + \varepsilon_t$$  \hspace{1cm} (3.8)

where

$a_i$ are the parameters and
\( \{ \varepsilon_t \} \sim N(0, \sigma^2) \), known as white noise.

This model can be written as an MA (q) process. For \( h < q \), there is a correlation between \( X_t \) and \( X_{t-h} \) due to the fact that they depend on the same error terms \( \varepsilon_{t-j} \).

3.2.2.2.3 Auto Regressive Moving Average Model

Combining the AR and MA forms together gives the popular ARMA model, which can be defined as

\[
X_t = b_1X_{t-1} + \ldots + b_pX_{t-p} + \varepsilon_t + a_1\varepsilon_{t-1} + \ldots + a_q\varepsilon_{t-q}
\]

where

\( \{ \varepsilon_t \} \sim N(0, \sigma^2) \), known as white noise and

\((p, q)\) are the order of the models.

ARMA models are a popular choice for approximating various stationary processes.

3.2.2.2.4 Auto Regressive Integrated Moving Average Model

An ARIMA model is a generalization of an ARMA model. A time series which needs to be differenced to be made stationary is said to be an “integrated” version of a stationary series. So, an ARIMA \((p, q, d)\) process is one, where the series need to be differenced, ‘d’ times to obtain an ARMA \((p, q)\) process.

3.2.3 Holt-Winters Exponential Smoothing Method

Exponential smoothing is a procedure where the forecast is continuously revised in light of more recent experience. This method assigns exponentially decreasing weights as the observation gets older. This method consists of the following steps

- Deciding on the model to use and setting initial values of model parameters
- Updating the estimates of model parameters and finally
- Forecasting the values

Single exponential smoothing, used for short range smoothing, assumes that the data fluctuates around a reasonably stable mean (no trend or seasonality). Double exponential smoothing method is used when the data shows a trend. Finally, triple
exponential smoothing, also called Holt-Winters smoothing, can handle both trend and seasonality.

There are two main Holt-Winters smoothing models depending on the type of seasonality – multiplicative seasonal model and additive seasonal model. The difference between the two is that in the multiplicative case, the size of the seasonal fluctuations varies depending on the overall level of the series whereas in the additive case, the series shows steady seasonal fluctuations. So, an additive seasonal model is appropriate for a time series when the amplitude of the seasonal pattern is independent of the average level of the series. For further details on Holt-Winters Exponential Smoothing, any standard book can be referred.

3.2.4 Time-of-day Model

Time-of-day model stores different load values based on the observations of past load, for each discrete time interval of the forecasting period. The model assumes that the current week’s load profile can be forecasted based on the previous week’s actual load. For different typical weeks and different weather conditions in a year, load profiles are stored which are then heuristically combined with the recent load pattern to create the forecast of the current week load. The usual representation of the model is given by [18]

$$\hat{L}(t) = \sum_{i=1}^{n} a_i f_i(t) + \varepsilon(t)$$  \hspace{1cm} (3.10)

where

- $\hat{L}(t)$ is the forecasted load at time t,
- $f_i(t)$ is usually sinusoids with a period of 24 hours or 168 hours corresponding to forecast lead time,
- $a_i$ are slowly time varying coefficients,
- $n$ is the number of factors and
- $\varepsilon(t)$ represents the modelling error.

The main advantage of these models is that they are very simple and most of the parameter values can be obtained from linear regression or exponential smoothing.
3.3 Modern/Intelligent Techniques

3.3.1 Artificial Neural Networks

It is modern trend to use computers to solve complex problems. The main drawback of computer is its inability to think and make decisions on its own. For including these characteristics, features from physiology of brain is borrowed and this has given rise to artificial neural systems or artificial neural networks. The concept was put forward as early as 1946 but the advances in this field were made in 1980’s. For the past ten years, it got popularized and now ANN is proved to be one of the important tools for solving most of the complex problems.

In an artificial neural network, the processing elements are known as neurons. These neurons are interconnected by information channels called synaptic connections. A single neuron can be represented as shown in the Figure 3.1. Here, a set of inputs labelled as $X_1, X_2, X_3,...X_n$ are applied to the artificial neuron. These inputs are collectively referred to as vector $[X]$. Each signal is multiplied by an associated weight $W_1, W_2, W_3...W_n$ before it is applied to the summation block. The summation block adds all of the weighted inputs algebraically producing an output that is termed as ‘NET’. This can be compactly stated in the vector notation as

$$NET = [X][W]$$  (3.11)

The NET signal is further processed by an activation function ‘F’ to produce the neurons output signal OUT.

Although a single neuron can perform certain simple pattern detection functions, the power of neural computation can be obtained from connecting neurons into networks. Layered Neural Network: Here the neurons are arranged in layers. A neuron receives information from the neuron of the previous layer and gives information to the neuron of
the next layer. Neurons of same layers are not connected. The first layer is called the ‘Input’ layer and the last one is called the ‘Output’ layer. If more than two layers are present, the in-between layers are called ‘Hidden’ layers. Input layer receives external inputs while output layer provides the output of the system. In Figure 3.2, a layered network with a single hidden layer is shown. Each input neuron or node can have multiple inputs while there can be only one output from a node which can be duplicated and given as inputs to many neurons. In the figure $W_{ij}$s are the weights of connection from input layer ‘$i$’ to the hidden layer ‘$j$’ and $W_{jk}$s are the weights of connection from hidden layer ‘$j$’ to output layer ‘$k$’. The functions of each layer in the network are defined as

- the input layer neurons receives the inputs
- the input of each hidden layer neuron is obtained by summation of the input vector set and the connection weights between the input layer and hidden layer
- the input of the output neuron is obtained by the weighted sum of outputs of the hidden layer neurons
- the output of a neuron is based on the type of the transfer function considered in the output layer
3.3.1.1 NEURAL NETWORK TOPOLOGIES

Based on the arrangement of neurons and their connections between different layers, different neural network topologies are developed. The most common neural network topologies are

- Radial basis function network (RBFN)
- Multilayer perceptron and
- Recurrent neural network

Among these, the most popular neural network architecture and the most widely reported neural network for STLF research problem is multilayer perceptron network. The overview of MLP is provided in section 3.3.1.2.
3.3.1.2 MULTILAYER PERCEPTRON FEEDFORWARD NETWORK

The Multilayer perceptron network consists of an input layer with a set of neurons, one or more hidden layers of computation nodes, and an output layer of computation nodes. MLP’s have been applied successfully in various domains where the inputs and the outputs are known beforehand, by training them in a supervised manner with error back-propagation algorithm.

The back-propagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a feedforward network and the desired output [152]. In this technique, the activation flows forward through the network while, error flows backward to the input, changing the interconnection weights as it goes. It employs the function that is differentiable everywhere and the sigmoid function satisfies this requirement. Before starting the training process, all the weights must be initialized to small random numbers say, between -0.1 to 0.1. This ensures that the network is not saturated by large values of weights.

3.3.1.2.1 Adjusting the Weights of the Output Layer

The weights in the output layer has a suffix ‘k’, in general and when a particular weight between neuron ‘p’ in hidden layer and ‘q’ in the output layer is considered, it is denoted by $W_{pq,k}$. To find the incremental weight, the first step is to find the difference between the output obtained from neuron ‘q’ and the desired output. This is then multiplied by derivative of sigmoid function i.e., $[\text{out}_{qk} \cdot (1 - \text{out}_{qk})]$. Calculated error for neuron ‘q’ in layer ‘k’, thereby producing the value $\delta_{qk}$ given by

$$\delta_{qk} = [\text{out}_{qk} \cdot (1 - \text{out}_{qk})] \cdot \left( \text{desired out}_{qk} - \text{out}_{qk} \right)$$  \hspace{1cm} (3.12)

Now, $\delta_{qk}$ is multiplied by the output ‘$\text{out}_{pj}$’ of neuron ‘p’ from the hidden layer ‘j’. This product is then multiplied by a learning coefficient ‘$\eta$’ and the result is added to the weight $W_{pq,k}$ giving rise to new weight, or in other words new weight is given by

$$W_{pq,k}(n+1) = W_{pq,k}(n) + \Delta W_{pq,k}$$  \hspace{1cm} (3.13)

where

$$\Delta W_{pq,k} = \eta \cdot \delta_{qk} \cdot \text{out}_{pj}$$  \hspace{1cm} (3.14)
An identical process is performed for each such weight proceeding from a neuron in the hidden layer to a neuron in the output layer.

In the above equations,
\[ \Delta W_{pq,k} (n) = \text{value of a weight from neuron 'p' in the hidden layer 'j' to the neuron 'q' in the output layer 'k' at step 'n'}, \]
\[ W_{pq,k} (n+1) = \text{value of the weight at step '(n+1)'}, \]
\[ \delta_{qk} = \text{value of the } \delta \text{ for the neuron 'q' in the output layer 'k' and} \]
\[ \text{out}_{pq} = \text{value of 'out' for neuron 'p' in the hidden layer 'j'} \]

3.3.1.2.2 Adjusting the Weights in the Hidden Layers

Hidden layers have no desired output. So ‘\( \delta \)’ for hidden layer must be calculated without the help of desired output.

First ‘\( \delta \)’ is calculated for each neuron in the output layer using equation (3.12). Weights are now adjusted according to equations (3.13) and (3.14). Now \( \delta_{pj} \) i.e., ‘\( \delta \)’ for a neuron ‘\( p \)’ in hidden layer ‘\( j \)’ is given by

\[ \delta_{pj} = \text{out}_{pj} [1 - \text{out}_{pj}] \sum_{q=1}^{r} \delta_{qk} W_{pj,k} \]  

(3.15)

Where \( \delta_{pj} = \text{value of ‘} \delta \text{‘ for the neuron ‘} p \text{‘ in hidden layer ‘} j \text{‘}, \) 
\[ \text{out}_{pj} = \text{value of ‘out’ for neuron ‘p’ in hidden layer ‘j’ and} \]
\[ r = \text{number of neurons in the output layer.} \]
This is repeated for all hidden layers.

3.3.1.2.3 Learning Rate Parameter, \( \eta \)

The learning rate parameter ‘\( \eta \)’ plays an important role in the training of the network. A small value of \( \eta \) means, the network learns slowly and on the other hand, a large value of \( \eta \) means the network learns quickly. Smaller value of \( \eta \) might take more number of iterations and larger value of \( \eta \) might lead to divergence. Hence, it should be selected carefully and usually, \( \eta \) is selected in the range of 0.05 to 0.25 to ensure that the network will converge to a solution. The value of \( \eta \) can be varied during the training process.
3.3.1.2.4 Momentum Factor, $\theta$

Another way to increase the speed of convergence is to use a technique called ‘Momentum’. While calculating the change in weight, we add only a fraction of change in weight of the previous iteration. This additional term tends up to help the weight change going in the same direction. The momentum factor ‘$\theta$’ is usually set to positive value less than one. Including the momentum factor $\theta$ the weight change equations become,

$$\Delta W_{pq,k} (n) = \eta \delta_{qk} \text{out}_{pj} + \theta \Delta W_{pq,k} (n-1)$$ (3.16)

and

$$W_{pq,k}(n+1) = W_{pq,k}(n) + \Delta W_{pq,k} (n)$$ (3.17)

3.3.1.3 ANN TRAINING AND GENERALIZATION

The development of supervised ANN model involves mainly, training process and validation process. During the training process, various input and output patterns are presented continuously to the neural network and weights between different layers are adjusted to obtain the exact mapping between input and output. After the training is completed, the network is supposed to learn the relationship between input and output behaviour and eventually able to provide a desired output for a given unknown input pattern. However, optimal training time should be achieved during the training process to avoid overtraining. Some of the popular stopping mechanisms to prevent overtraining of the network are early stopping (the most common), regularization, pruning, information criterion pruning (ICP) and cross validation pruning (CVP). More details on these methods can be found in [153]. A detailed reference on over fitting is discussed in [154].

Training the network by employing very low error goal value normally results in a reduced error function but, this does not guarantee better accuracy and robustness of the network because the network may not be able to generalize for an unknown set of inputs. The generalization ability of the network is explored during the validation process.

3.3.1.4 ANN LEARNING PARADIGMS

The most popular and common ANN learning paradigms for training the neural networks are supervised and unsupervised learnings. Recently, the other learning techniques like reinforced and competitive learnings are also gaining momentum. During
the training process, the network synaptic weights are adjusted and over a period of time, when the error reaches error goal, the network is supposed to learn the input and output patterns behaviour and act like a human. However, the process of adjusting synoptic network weights and biases are based on mathematical algorithms.

3.3.1.4.1 Supervised Learning

In supervised learning, the input(s) and the desired output(s) are presented to the neural network. Suppose, the input vector is represented by \([x_1, x_2, ..., x_m]\) and the corresponding output vector is denoted by \([y_1, y_2, ..., y_n]\), the objective of supervised learning is to obtain an optimal rule such that,

\[
[y_1, y_2, ..., y_n] = f[x_1, x_2, ..., x_m] + \epsilon
\]

where

\(\epsilon\) represents the approximation error.

The idea is that the network calculates the output by considering the initial weights and the input vector and the calculated output is then compared to the desired output or target value. The difference in error is fed back to the network and weights are adjusted to minimize the error to the least possible value. Figure 3.3 illustrates a basic network structure for a supervised ANN model.

![Figure 3.3 Supervised learning process](image)

If the desired error goal is not reached, the network weight vectors are again updated and this process continues till the stopping criterion is reached. Some of the key parameters to consider for a supervised learning session are
time required for each iteration
number of iterations taken to reach the error goal and
local minima/global minima

3.3.1.4.2 Unsupervised Learning

In contrast to supervised learning, here, no external target value is presented as output to the network for training. It draws inferences from various input patterns presented to the network during the training session. During the training process, the weight values of neural network are adjusted such that the data provided to the network is partitioned into clusters. The data in each cluster ideally share some common characteristic. It is also often referred to as “self-organized learning”. Figure 3.4 shows the network structure for the unsupervised learning process.

Figure 3.4 Unsupervised learning process

3.3.1.5 SUMMARY

The ANNs are good at modelling non-linear behavior and generalization. It does not require any mathematical equations to describe the functional relationship between input and output variables. They can learn from experience from the data provided to them and able to generalize from previous examples to new ones. They are more robust as compared to conventional techniques.

3.3.2 Rule based Expert Systems [155]

An expert system is a computer program with domain knowledge base in the form of rules. It has the ability to infer, reason, explain and act as a knowledge expert. It usually consists of four modules – user interface, database, inference engine and explanation module. The user interacts with the rule based expert system through user interface. Graphical user interfaces are the most common interfaces for user friendliness. The
database is used to store the domain knowledge which is represented as facts in the form of IF-THEN rules. These facts are used by inference engine to arrive at the solution. The knowledge base is developed by having close interactions with domain experts in that field. The knowledge base can be evolved with new rules during the course of time. Inference engine, as the name suggests, makes inferences. It identifies all the rules satisfied by the facts, prioritizes them, and finally executes the rule with the highest priority. It gets operated in cycles, executing a group of tasks until the stopping criteria is reached. The tasks include conflict resolution, act, match and check for stopping criterion. Explanation module provides the user about the reasoning process of the system for the given inputs. By keeping track of all the rules that got fired, this presents a chain of reasoning that led to the final conclusion. Some expert systems also contain knowledge acquisition facility which helps the user to enter the knowledge directly into the system. This avoids the need of knowledge engineer to explicitly code the knowledge into the database.

3.3.3 Fuzzy Systems

The term fuzzy set was introduced by Dr. L. A. Zadeh in the mid-1960s to represent and manipulate data that possesses uncertainty. Fuzzy set theory is a generalization of conventional set theory to represent vagueness in the data. Hence, in this theory, there is no sharp distinction between members and non-members of a fuzzy set. Each member of the class is assigned a degree of membership function that represents its grade of belongingness to that fuzzy set. So, everything is a matter of degree.

A fuzzy system is a logical system that mimics the human way of thinking in handling uncertainty and vagueness. In general, a fuzzy rule system has four modules [156],

- Fuzzification – the process that transforms the crisp input into fuzzy input
- Fuzzy rules – rules defined in the form of IF-THEN logic statements between input output variables
- Fuzzy inference – the process that identifies different rules and combine their outputs and
- Defuzzification – the process that transforms the fuzzy output into crisp output
Since its inception, it has gained lot of significance and number of products ranging from washing machines, air conditioners, cameras etc., have been developed using fuzzy logic. The main advantage of fuzzy logic is that there is no need of mathematical equations of inputs and outputs. Properly designed fuzzy logic systems can be very robust and can be used for load forecasting. To forecast the load, the domain knowledge of utility operator must be incorporated in the form of fuzzy rules.

3.3.4 Evolutionary Approach

The algorithms developed under the common term of evolutionary computation are inspired from the study of evolutionary behavior of biological processes. They are mainly based on selection of a population as a possible initial solution of a given problem. By considering various evolutionary operators, such as crossover, recombination, selection and mutation, the fitness of the initial population can be steadily improved through stepwise processing.

When applying a genetic algorithm to a load forecasting problem, initially an appropriate model (either linear or non-linear) need to be considered and an initial population of candidate solutions is created. Then, a candidate solution is obtained by randomly choosing a set of parameter values for the selected forecasting model. Each solution is then ranked based on its prediction error over a set of training data. The next generation of solutions is created by selecting more fitting solutions and applying a crossover or mutation operation. This process of generating new populations is repeated until the fittest solution has a sufficiently small prediction error or repeated generations produce no reduction of error.

3.3.5 Support Vector Machines

Support vector machines, proposed by V. Vapnik [77], is a supervised learning technique based on statistical analysis and robust regression theory. It is based on structural risk minimization (SRM) principle and basically non-linearly map the input space into a very high dimensional feature space via a kernel. The most commonly used kernels are linear, polynomial, radial basis function and sigmoid. SVMs can be applied for classification, regression and clustering problems and the main advantages of SVM are
• the solution to an SVM problem is global and unique
• the computational complexity of SVMs does not depend on the dimensionality of the input space
• it is based on simple geometric interpretation and hence give a sparse solution
• avoidance of over fitting of data
• high training speed

The formulation of SVMs which is based on structural risk minimization principle is superior to empirical risk minimization (ERM) principle used by conventional neural networks. SRM minimizes an upper bound on the expected risk, whereas ERM minimizes the error on the training data. Due to this difference, SVMs have greater ability to generalize, which is the goal of statistical learning.

SVMs have been successfully applied to real-world data analysis problems, often providing improved results compared with other techniques. A clear advantage of the support vector approach is that sparse solutions to classification and regression problems are usually obtained and only a few samples are involved in the determination of the classification or regression functions. This fact facilitates the application of SVMs to problems that involve a large amount of data. The theory of support vector regression is briefly introduced in section 3.3.5.1.

3.3.5.1 SUPPORT VECTOR REGRESSION

Given training data \((x_1, y_1), (x_2, y_2) \ldots \ldots (x_n, y_n)\) where \(x_i\) are input vectors and \(y_i\) are the associated output value of \(x_i\), the Support Vector Regression is an optimization problem.

\[
\begin{align*}
\min_{w, b, \xi, \xi^*} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\
\text{subject to} & \quad y_i - (w^T \phi(x_i) + b) \leq \varepsilon + \xi_i, \\
& \quad (w^T \phi(x_i) + b) - y_i \leq \varepsilon + \xi_i^* \quad \text{and} \\
& \quad \xi_i, \xi_i^* \geq 0, \quad i = 1, 2, \ldots \ldots n,
\end{align*}
\] (3.19)
where

\[ x_i \text{ is mapped to a higher dimensional space,} \]

\[ \zeta_i \text{ is the upper training error (} \zeta_i^* \text{ is the lower) subject to the } \varepsilon \text{-insensitive tube} \]

\[ |y - (w^T \phi(x) + b)| \leq \varepsilon. \]

The parameters which control the regression quality are the cost of error \( C \), the width of tube \( \varepsilon \) and the mapping function, \( \phi \).

The constraints of (3.19) imply that we would like to put most data \( x_i \) in the tube \( |y - (w^T \phi(x) + b)| \leq \varepsilon \). If \( x_i \) is not in the tube, there is an error \( \zeta_i \) or \( \zeta_i^* \) which we would like to minimize in the objective function. For traditional least-square regression, \( \varepsilon \) is always zero and data are not mapped into higher dimensional spaces.

### 3.4 Comparison of Statistical Techniques and Artificial Intelligence Techniques

One of the major concerns for statistics is modelling of probabilistic problems, specifically inference, and these models are popular to formally structure knowledge. These techniques require a functional relationship between the input and output variables through which the trend can be obtained. It needs complex mathematical calculations which consume a lot of time to arrive at the output. AI is a broad term for anything that has to do with machines doing reasoning, planning a task and/or cooperating with other entities etc. Given a set of patterns, AI techniques can deduce the relationship between the input and output variables. AI techniques have been found very good for modelling the non-linear relationship between inputs and outputs. Though AI models may take some time for training, which depends on the amount and quality of data, the testing is almost instantaneous. These models are more robust to random disturbances when compared to the statistical models. Also, these techniques handle noisy data in a better way.

### 3.5 Comparison of Artificial Neural Networks and Support Vector Machines

The development of ANNs followed a heuristic path, with applications and then extensive experimentation preceding theory. On the other hand, the development of SVMs is based on sound theory first, then implementation and applications. The main advantage of SVMs over ANNs is that while ANNs can suffer from multiple local minima, the solution to an SVM is always global and unique. Another advantage is that SVMs give a
sparse solution based on simple geometric interpretation. Also, unlike ANNs, the computational complexity of SVMs does not depend on the dimensionality of the input space. ANNs are based on empirical risk minimization principle while SVMs use structural risk minimization principle. Practically, SVMs often outperform ANNs due to the fact that SVMs are less prone to over fitting.

The number of mandatory tuneable parameters in SVMs are less when compared to the number of parameters required for NNs and the choice of parameter values may not be that crucial for getting good forecasting results. Given the input training data, SVMs are designed to systematically optimize its structure. The training process of SVMs involve solving a quadratic optimization, and does not require any random initialization of weights as training NN does. So, any SVM with the same parameter settings trained on identical data should be able to provide identical results. This increases the repeatability of SVM forecasts by reducing the number of training runs required to obtain the optimum SVM parameter settings when compared to NN training.