UNIVERSITY OF AGDER

Short-term Forecasting of Electricity Consumption using Gaussian Processes

Girma Kejela

Supervisor
Ole-Christoffer Granmo

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2012
Faculty of Engineering and Science
Department of Information and Communication Technology

Abstract

Although state-of-the-art models like linear regression and neural networks have been widely used for electricity consumption forecasting, the demand for improved prediction accuracy is still very high. A small improvement in prediction accuracy has a significant economic value for the energy industry. Gaussian processes (GPs) are becoming a more and more popular tool in machine learning, and in this thesis we will investigate how the GPs can be used for electricity consumption forecasting.

Its non-parametric nature make GPs a natural approach to addressing complex stochastic problems that are difficult to solve using the earlier parametric models. The drawback of the GP prediction model is that it requires computation which grows as $O(n^3)$, where n is the number of training points. To deal with such problems we used a novel approach known as the k-nearest neighbor (kNN) similarity search, which explores the training set and selects k elements of the dataset that are closest to a given query point. By using the output of the kNN search as a training set for GPs, the computational cost will be reduced from $O(n^3)$ to$O(k^3)$, where $k \ll n$.

The experimental sets in this thesis are based on a real life dataset obtained from Eidsiva Energy, and our goal is to forecast electricity consumption a day ahead (for the next 24 hours). In addition to the GPs, the neural networks (NNs) and local linear regression (LLR) models are also implemented and tested using test inputs over the same period of time as the GPs. Empirical analyses of the results show that the GPs outperform both the NNs and the LLR models.

Keywords: Gaussian Processes (GPs), k-Nearest Neighbors (kNN) similarity search, Neural Networks (NNs), Local Linear Regression (LLR), Short-Term Forecasting of Electricity Consumption (STFEC)

Preface

This master thesis is submitted in partial fulfillment of the requirements for the Master of Science degree in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science. The project was carried out under the supervision of Professor Ole-Christoffer Granmo at the University of Agder, Norway.

I would like to thank my supervisor, Professor Ole-Christoffer Granmo for giving me the opportunity to work on this challenging and interesting problem. He initiated and supported this work with his long standing experiences and research background. Without his inspiring, productive and supportive supervision, this work could never have been done. I am also very grateful to Eidsiva Energy, who provided real life datasets used in the experimental sets of the thesis.

Table of Contents

List of Tables

List of Figures

List of Algorithms

General Regulation for Notations

The notation of variables in this thesis are based on the following regulations:

- Matrices are capitalized (e.g. **X**), and vectors are not capitalized but bold faced (e.g. **x**)
- A superscript asterisk, such as X*, indicates reference to a test set quantity
- In most cases X denotes the inputs or independent variables (also called features, e.g. feature selection refers to each input dimension)
- y denotes the output, which is also called dependent variable or response.
- Experiments in this thesis involves a data set in form: $\boldsymbol{D} \equiv \{\boldsymbol{x_{ij}}, \boldsymbol{y_i} | i = 1, \dots, n \; and \; j = 1, \dots, d\}$ of n input–output pairs. Where the input is in an $n \times d$ matrix form.
- All references (including reference to other section of the thesis, scientific papers, books, figures, tables, equations etc.) are marked in green.
- Some literatures use the term 'predictor' as an input variable, but in this thesis predictor refers to the model that predicts electricity consumption (EC), i.e., GPs, NNs or LLR.

1 Introduction

On a broad view, the problem of forecasting electricity consumption can be categorized under machine learning, which is the study of computer algorithms that improve automatically through experience. In order to predict how a trend will continue, the prediction model should be able to generalize the knowledge in historical data to unseen future. That is, to predict electricity consumption (EC), the algorithm needs to learn the underlying function that maps the input variables such as temperature, hour of day, etc. to an output variable. To achieve this, we implement three regression techniques, namely: Gaussian processes (GPs), neural networks (NNs), and local linear regression (LLR) models.

1.1 Background

At any point in time, electricity demand must be met by generating exactly the same amount of power. Generally load is viewed as uncontrolled variable, and the generation must constantly adjust to reliably meet the demand. [41][42]

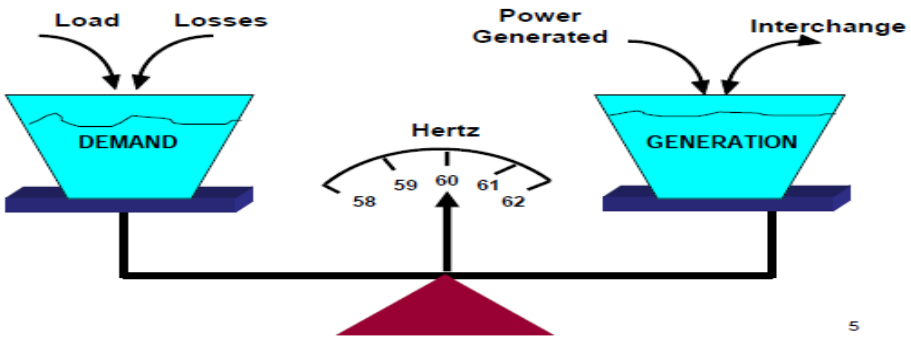

Fig. 1.1 [40]: Electricity supply/demand balance and system frequency

Fig. 1.1 shows that unbalanced supply-demand causes fluctuation in system frequency, and this may result in unwanted load shedding and power system instability (if the generating units fail to respond fast). System frequency is controlled through adjustment of the guide vane position (see Fig. 1.2) using speed governor feedback. For instance, if the demand is greater than generation the system frequency will fall below the frequency set. Based on the difference between the two frequencies, the governor will adjust the guide vane position in such a way that more water will flow into the turbine and thus the power generation will increase to serve the new demand.

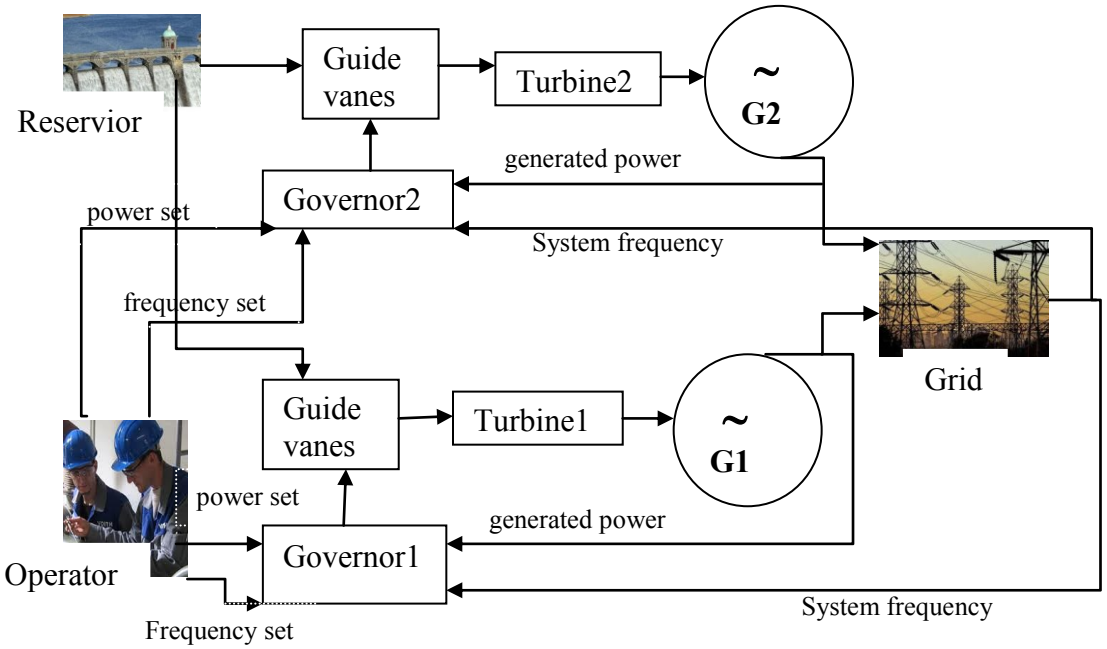

Fig. 1.2: Block diagram of governor control for hydropower plant

Since power generators couldn't operate beyond their design rate value, adjusting generated power to meet the demand is possible only if we have kept enough reserve. On the other hand, maintaining reserve involves running the generator bellow its full

capacity or running additional generators. The task of the system operator is to operate power system in a safe, secure, and economic manner. To perform those demanding tasks the system operator should be able to schedule in advance and dispatch in real time. Unit commitment schedule ensures that there are sufficient generation resources committed to meet forecasted system demand at minimum production cost. [41]

If the forecasted electricity consumption is inaccurate, the system operator (SO) may overestimate or underestimate the demand. If the SO overestimate the demand, he may schedule larger reserve than the system requires which is uneconomical. On the other hand, underestimation of the demand means less reserve schedule and thus insecure operation. Therefore, a secure[1] and economic operation of electric power system requires a precise estimate of future electricity consumption. Besides, accurate forecasts of electricity consumption are required to reduce spinning reserve, schedule maintenance and optimize energy trading mechanisms [5][7]. Therefore, an accurate forecasting of future electricity consumption helps energy companies to provide reliable and uninterrupted electricity at a competitive price.

Thesis statement: The purpose of this thesis is to study how Gaussian processes (GPs) can be used for Short-term Forecasting of Electricity Consumption (STFEC). The seasonal, daily and hourly variations in electricity consumption (EC) and respective weather information are stored in a large dataset, and the GPs use the dataset to learn trends of EC. The main problem with GPs is that the computational requirement scales as $O(n^3)$, where n is the number of training data points. To cope with this problem, we propose a novel approach known as the k nearest neighbor (kNN) similarity search. A

[1] Security refers to power system's ability to withstand disturbances.
[2] The function f is distributed as a Gaussian Process with mean function $m(\mathbf{x})$ and covariance

kNN similarity search is used to select subsets of the training candidates that are closest to the test points. Features that form the input dimension will be selected by analyzing the influence of each input variable on EC and testing the contribution of each feature to the overall prediction accuracy. Finally, the GP models will be compared with the state-of-the-art regression models, namely: the neural networks and local linear regression models.

Learning Methods: Training is a vital process in machine learning. There are two forms of training namely: supervised and unsupervised trainings. In supervised learning the training inputs and the anticipated/ideal outputs are provided. On the contrary, in unsupervised training only the training inputs are available and the ideal outputs are absent. A supervised training data consists of a set of training examples, where each example is a pair comprising a training input **X** and an anticipated output **y**. A supervised learning algorithm analyzes the training data and produces an inferred function, which is known as a regression function (for quantitative output), or a classifier (for qualitative output). [3][23] Our work focuses on the regression functions that generalize knowledge from the training data to unseen situation at the test points.

Parametric and nonparametric regression: Based on the assumption made regarding the underlying function, regression models can be categorized into two: parametric and nonparametric. In parametric regression, we assume an explicit model regarding the relationship between the inputs and the response. Linear regression is a classic example of parametric models, which uses the least squares approach to estimate the parameters. In a nonparametric regression model, the predictor does not take a

predetermined form but is constructed according to information derived from the data. [15][16] A Gaussian process is an example of nonparametric regression

1.2 PROBLEM STATEMENT

As mentioned at the beginning of this chapter, the problem in this thesis is categorized under machine learning in which we want to implement algorithms that learn from experience. Knowing that the experience is represented in form of training input-output pairs $D = \{\mathbf{X}, \mathbf{y}\}$, the problem can specifically be stated as follows: *Given a dataset comprising hourly observations of electricity consumption (EC)* $\boldsymbol{y}$ *at training input* $\boldsymbol{X}$*, what will the EC* $\boldsymbol{y^*}$ *be at a new query* $\boldsymbol{X^*}$ *? The aim is to develop a regression model that learns the relationship between the input variables and EC during the training process, and generalize the knowledge to unseen situations when predicting.*

However, modeling electricity consumption is a challenging task as it depends on many variables: *temperature, hours of day, days of week, holidays, seasons of year, population, economy, and many others.* Moreover, distribution of electric grid over a large geographical area leaves the grid exposed to many unpredictable incidents. For example, thousands of people may interact with the grid, in unpredictable ways and from different locations. Fortunately, most of the random incidents are very small and some of those incidents, which represents majority of electricity consumption, follow some trend that depends on time, weather conditions, and occasions like public holidays. Representing those trends over a specific function is still a problem since no definite function can efficiently model the stochastically varying incidents in electric grid. That is to say that the path followed by EC trends is non deterministic, and thus the problem is

more convenient for nonparametric regression approach such as the Gaussian processes (GPs).

1.3 LITERATURE REVIEW

The use of Gaussian Processes (GP) in electric load forecasting is quite a new concept. However, Gaussian Process prediction has recently become more and more popular in different areas. Different studies, including these presented in this section, show that GP based prediction outperforms traditional methods like linear regression and neural networks. In order to summarize different approaches to the problem, selected related works are briefly presented below.

An interesting work by Atallah et al. 2008 [19], used GP for predicting and correcting missing data and outliers in body sensor networks, which was used for home-monitoring of chronically ill patients. The GP prediction framework was trained on recorded datasets to study the relationship between activities of daily living (such as walking, sleeping etc.) and physiological parameters. Based on the training set and other sensor inputs, GP was used to infer channel output. The difference between inferred output and received data was used to predict the missing data. A data is considered noisy when the deviation from the predicted value is greater than the standard deviation of the signal and the errors can be corrected by replacing the noisy part of the signal by the predicted value.

The work by Brahim-Belhouri et al. 2001 [18], focused on comparing GPs with Radial Basis Function (RBF) neural networks. They experimented on dataset generated by Mackey-Glass equation and found out that GP based Bayesian learning is more accurate than the RBF neural networks.

In another paper, by Leith et al. 2004 [21], GP was used on Irish load data, and comparison with other popular forecasting methods like Basic Structural Models(BMSs) and Seasonal Auto-Regressive Integrated(SARI) show that GP is more accurate than the other two.

Mahalanobis kernel GP was used by Mori et al. 2009 [22] for forecasting temperature in Tokyo, from June to September 2002. The analysis based on 366 training data point and 122 test data points shows an average prediction error of 2.56%. It was indicated in this paper that the proposed GP method is better than other methods that were used earlier, including MLP (Back propagation), RBFN with SVC and SVR with Mahalanobis kernel.

A paper by Chakhchoukh et al 2009 [23] wasn't used GP for prediction, but the general approach in re-modeling the dataset to deal with problems caused by outliers and weekend load is useful. They analyzed the electric consumption in France using data provided by RTE. The focus of this work was on increasing the robustness of ARIMA model which was already used by RTE. Identification of outliers and public holidays in the data was the significant methods used to increase robustness of prediction.

One more paper I would like to present before concluding this section is a non-Gaussian approach by Osman et al., 2009 [25]. They used neural network for short-term load forecasting on a dataset obtained from Egyptian Unified System. The dataset was analyzed in order to find out the most correlated weather data. From the analysis, they learned that temperature, humidity, and historical load are the most informative data for prediction in Egyptian power grid. In this work the outliers and weekend data were discarded from the training set to optimize the accuracy of prediction.

1.4 RESEARCH QUESTIONS

- **How can we apply Gaussian Processes for short-term forecasting of electricity Consumption (STFEC)?**

  The core research element is to study methods of handling the challenges stated in the problem statement and apply the GPs for STFEC. However, this question is too general and many sub-questions will arise as we try to answer it. Thus, the main research question is divided into subordinated research questions that are empirically analyzable and more precise in specifying what we want to measure, what we want to optimize, and which methods and scenarios we want to compare.

Subordinated Research Questions

- **Is it possible to overcome the limited data handling capabilities of GPs, and let the GPs learn the most relevant information stored in a large dataset?**

  As stated earlier, datasets represent experiences in the learning process; therefore we need to process large time series data in order to acquire the knowledge needed for prediction. However, processing a large dataset is computationally expensive, and particularly in GP regression, it scales $O(n^3)$, where n represents the size of data. In fact, time series data are naturally sparse and only 'small' subsets of the data holds relevant information we need for prediction. If we need only a portion of data for prediction, processing the whole data only adds computational burden. On the way to solution, we need to answer the following questions:

  - Can we measure relevance or similarity of points in our data in relation to a given test point?

- If so, can we select the most similar/relevant points based on this measurement?
- If yes, can we tradeoff between computational burden and prediction accuracy to decide the number of closest data points we should include in our selection?
- Then, can we use the selected points as training set for the GP, and thus reduce the computational requirement?

The first three questions will be answered in Section 2.2.2, where we discuss about kNN similarity search and the role the kNN search in solving computational problem of GP is explained in Section 5.4.

- **How is the performance of GPs compared to the state-of-the-art models like neural networks (NNs) and local linear regression (LLR) models?**

  First we want to implement the models, and then measure the performance of each. The performance is measured using the following methods:

  - By computing the Mean Absolute Percentage Error (MAPE) and the correlation coefficient between the predicted and the actual values
  - By plotting the predicted and actual values
  - Error frequency distribution plot to evaluate the validity of each model.

  Evaluation and comparison of the regression models is presented in Section 6.2.

- **How can we select the best feature vectors that optimize the performance of the predictors (GPs, NNs, and LLRs)?**

  The independent variables that form the input dimension play a vital role in determining the prediction accuracy. Thus an implementation of any regression

model should involve design of feature vectors (FV) in such a way that the input variables that are strongly related to the output are included in the FVs. Here, we need to answer one question. Can we measure how strong an input variable is related to an output? The answer is yes, at least we can visually analyze. Section 2.1, exploratory data analysis (EDA) provides an insight to the approach. Furthermore, we will test the influence of each feature by considering one feature at a time and observing its contribution to the performance of a regression model.

- **How the predictors (GPs, NNs and LLRs) perform under scenarios like different seasons of a year, different days of a week and different hours of a day?**

  This question arises because of the fact that there is high variation in EC with changes in seasons of a year, days of a week, and hours of a day. For example, EC trend in summer is very different from EC trend in winter; similarly EC during work days is different from EC in weekends or holidays. This is a form of validation test in which we want to analyze the following:

  i. whether all the information about those variations are included in the feature vectors, and
  ii. whether the implemented regression models can effectively generalize the information to unseen situations at test points

1.5 SOLUTION OVERVIEW

We approached the problem first by studying the relationships between the independent variables and the dependent variable (electricity consumption in this case). Correlation coefficients between different variables in the data were analyzed to see if

there is any linear relationship between the variables. In addition to the correlation coefficients, scatter plots where used to visualize if there is some relationship between the candidate features and EC or no relationship at all. This analysis was used as a step stone in selecting the feature inputs and it will be referred as the first round feature selection.

Once the input variables are identified from the first round feature selection, three regression models (namely: GPs, NNs and LLR) have been implemented in order to predict electricity consumption. The input variables that are selected in the first round cannot be optimal since there might exist some correlation between the independent variables themselves. Inclusion of correlated inputs in the feature vector may reduce prediction accuracy and adds unnecessary computational complexity. Thus, we need a second round feature selection. This was done through further experiments by considering one feature at a time and then observing their effect by analyzing the performance of the predictor.

1.6 CONTRIBUTIONS

The main contribution of this work is answering the research questions. For example, solving the issue that arises with large dataset is not specific to STFEC; it can be applied to similar problems that involve GP regression. Moreover, it is a novel approach that is implemented outside the GP model (i.e., no change to the full GP regression model). Therefore, it can be used irrespective of the type of covariance function or likelihood function we chose for the GP. The kNN search can also be used with other regression models, for example implementation with the LR model shows better performance compared with respective LR model without kNN search.

From the industrial point of view this work has significant contributions as it investigates various prediction models and experimentally selects feature vectors and models that give better prediction accuracy. As discussed in previous sections a small improvement in prediction accuracy has a significant economic value in the energy industry. Moreover, this work can be extended to similar problems in other areas such as in the finance sector, weather forecasting etc.

1.7. THESIS OUTLINE

The thesis report is organized in 8 chapters. Chapter 2 introduces data pre-processing. Section 2.1 covers exploratory data analysis (EDA), correlation coefficients (CC), and feature selection. Section 2.2 is concerned with data structure in which we will discuss the KD tree and the k nearest neighbor (kNN) similarity search.

Chapter 3 briefly presents the linear regression model. Section 3.1 introduces the linear regression model. Section 3.2 presents how the regression parameters are computed by using the list square method. Finally, in Section 3.3 we will discuss how the local linear regression model was implemented for STFEC.

Chapter 4 introduces the neural networks (NNs). Section 4.1 presents the mathematical model of NNs. Section 4.2 introduces the commonly used activation functions. Section 4.3 presents the architecture of NNs with some guiding 'rules' to follow when deciding the number of hidden layers and number of neurons in each layer. In Section 4.4 we will discuss training of the NNs by presenting two popular training algorithms namely: the backpropagation and resilient propagation (RPROP). Section 4.5 briefly discusses the implementation of NNs for STFEC.

Chapter 5 introduces the Gaussian processes. Section 5.1 discusses GP regression. Section 5.2 presents the role and types of covariance functions. Section 5.3 discusses how the GP learns the hyperparameters. Section 5.4 discusses the challenge of working with big dataset in GP. Finally, Section 5.5 presents the implementation of GPs for STFEC.

Chapter 6 presents experimental results with a brief description. Chapter 7 summarize and discusses the results. Chapter 8 finalizes the report with conclusion and further work.

2. Data Pre-Processing

The first step in designing a regression model is studying the dependence between different variables in the data. The aim is to explore the data in order to reveal patterns and features that will help us understand, analyze and model the data. Exploratory data analysis (EDA) visualizes the underlying patterns in the data. It will tell us whether there is linear relationship, nonlinear relationship, or no relationship at all between variables in the data. While EDA can be seen as a visualization model, there exists a numerical equivalent called correlation coefficients. But correlation coefficients (CC) tell us only whether there exists a linear relationship; it cannot always distinguish between nonlinear relationship and no relationship. The EDA and CC are used for feature selection.

The kNN similarity search explores the data and chooses k elements of the data that are closest to a given query point. The naive neighbor search implementation involves the brute-force computation of distances between all pairs of points in the dataset. A KD tree is a tree based data structure that is intended to improve the efficiency of the brute-force approach. [39]

2.1 Data Analysis

2.1.1 Exploratory Data Analysis

Exploratory data analysis (EDA) is a collection of techniques for revealing information about the data and methods for visualizing them to see what they can tell us about the underlying process that generated it [15]. EDA explores data without assumptions about relationships between variables, error distribution, etc. in order to discover what they can explain about the phenomena we are studying. In this section we will investigate the relationship between electricity consumption (EC) and other variables

such as temperature, cloud cover, wind speed, hours of day, days of week and seasons of year. Moreover, the relationship between current electricity consumption (EC) and historical EC such as previous hour power, previous day power, previous week power etc. will be investigated.

A simple scatter plot can reveal the relationship between two variables. For example a scatter plot of electricity consumption versus temperature shows how electricity consumption may vary with temperature. If points in the scatter plot are clustered around a straight-line it shows linear relationship between the variables; if points are clustered around a curve, it means that the variables are nonlinearly related; points scattered randomly on the x-y plane without clustering around any path shows the variables are not related. Thus, plotting a scatter diagram is an important step in selecting the feature vectors and to select an appropriate regression model that maps one variable to the other.

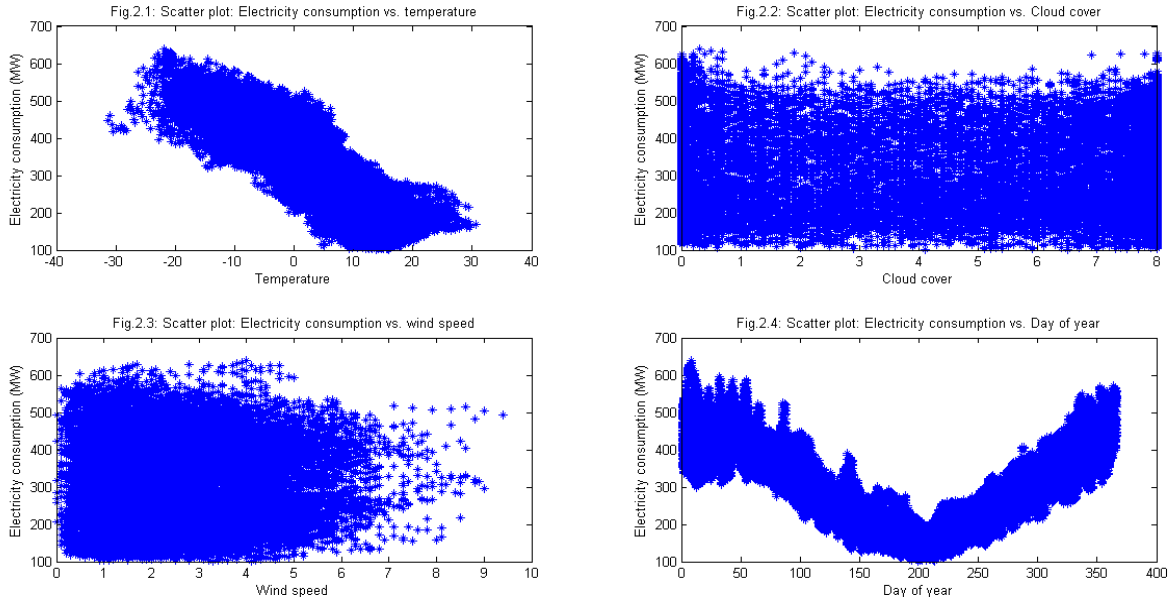

Fig. 2.1–2.4: Scatter plots for EC against independent variables

In the scatter diagram of Fig. 2.1, the points seem to cluster around a straight line, and this shows that the relationship between temperature and EC is approximately linear with negative slope. In Fig. 2.2 and Fig. 2.3, the absence of tight cluster about a line or a curve reveals that the relationship between EC and cloud cover or wind speed is not that strong. In Fig. 2.4 the points seem to cluster around a curve and this shows that there is nonlinear relationship between electricity consumption and days of year.

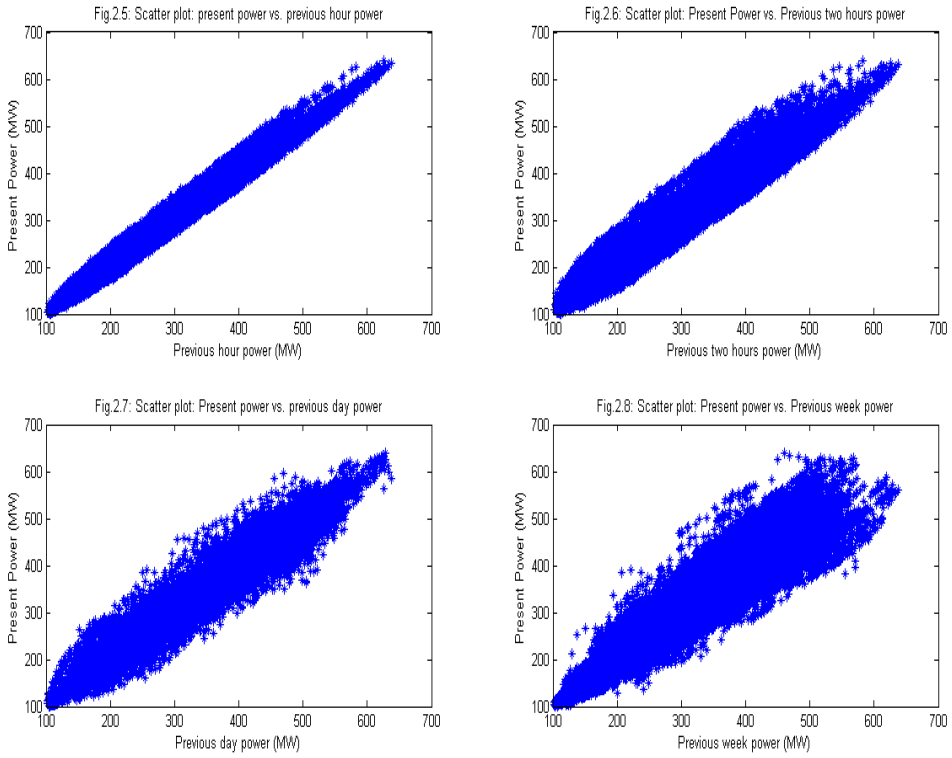

Fig. 2.5–2.8: Scatter plot of EC against previous ECs.

As can be seen from Fig. 2.5-2.28, the present consumption is linearly related to previous electricity consumptions and the relationship is stronger with recent

consumptions than with the older ones. However, the strong correlation doesn't necessarily mean we should include all the previous consumptions. This is because those previous hour, previous two hours, previous day, previous week, etc. consumptions are strongly correlated with each other and only some of them will provide additional information required for prediction.

2.1.2 Correlation Coefficients

In Section 2.1 we have seen that if a scatter plot of a variable y against variable x tightly clusters about a straight line, then x and y are linearly related. A numerical measure of linear relationship between variables is referred to as sample correlation coefficient. The sample correlation coefficient for n pairs of observations $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ is given by:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{[\sum_{i=1}^{n}(X_i - \bar{X})^2][\sum_{i=1}^{n}(Y_i - \bar{Y})^2]}} = \frac{S_{xy}}{\sqrt{S_x^2 S_y^2}} \tag{2.1}$$

Where: sample mean of X and Y are: $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ (2.2)

sums of squared deviations: $S_x^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2$, $S_y^2 = \sum_{i=1}^{n}(Y_i - \bar{Y})^2$ (2.3)

and the sum of cross product of deviations is: $S_{xy} = \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$ (2.4)

The correlation coefficient r can vary between -1 and 1. The value $|r| = 1$ shows a perfect linear relationship between X and Y, which is equivalent to a tight clustering of data points around a straight line. An absence of linear relationship is represented by $r = 0$. The proportion of variability in the y values that can be explained by a linear

relation is precisely represented by r^2, which is derived by analyzing the residual of linear regression model in Section 3.2. Thus, the sample correlation coefficient r can explain whether the linear regression model is an appropriate choice to model the relationship or not. But it may fail to distinguish nonlinear relationship from no relationship.

Autocorrelations: Autocorrelation is used to investigate the possibility of dependence among successive observations. The term autocorrelation (or serial correlation) refers to a correlation between observations that are adjacent in time order or any other order of data collection [9]. For example the correlation between consecutive members of time series data $X_1, X_2, \dots, X_n$ is known as *lag 1* correlation or first-order autocorrelation and is given by:

$$r_1 = \frac{\sum_{i=2}^{n}(X_{i-1}, \bar{X})(X_i - \bar{X})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{2.5}$$

Table 2.1 summarizes the correlation coefficients for 15 input variables. It can be seen from the table that, the temperature and historical EC have strong linear relation to present EC, which is in agreement with our observations in the scatter plots.

Table 2.1: Correlation coefficients of candidate features and EC

Actual Temperature	Forecasted Temperature	Cloud cover	Wind speed	Hour of day	Day of week	Day of year	Year	Weekend?	Previous hour power	Previous two hours power	Previous three hours power	Previous day power	Previous week power	Previous two weeks power
-0.85	-0.85	0.08	0.01	0.12	-0.09	-0.22	0.04	0.09	0.99	0.98	0.95	0.98	0.94	0.91

2.1.3 Feature Selection

Feature subset selection is the process of identifying and removing irrelevant and redundant features from a training data set. In general, relevance or redundancy of features is defined as follows [36]:

- **Relevant features** are those features which have an influence on the output and not correlated with any other features.
- **Irrelevant features** are those features which have little influence on the output.
- **Redundant features** are those which are correlated with at least one of the other features, and thus provide no additional information.

The influences of 15 features, shown in table 2.1, have been analyzed using EDA and CC. Based on this analysis we selected the first round feature vector $FV1$ consisting of:

$$FV1 = \{Temperature, previous\ hour\ EC, previous\ two\ hours\ EC,$$
$$previous\ three\ hours\ EC, previous\ day\ EC,$$
$$previous\ week\ EC, day\ of\ year, hours\ of\ day\}$$

There are some redundant features that are correlated with other features in $FV1$. Particularly, historical ECs are strongly correlated with one another and only some of them provide additional information that is needed for prediction. Moreover, features like days of week, cloud cover etc., have been excluded from $FV1$, because their influences on EC were not clear from EDA or CC analysis. Thus, we need second round feature selection $FV2$, in which each feature is tested using the respective predictors and features that optimize the performance of each predictor (see table 7.1) are selected.

Second round features $FV2$ for GPs and LLR comprises 7 elements:

$$FV2_{GP\&LLR} = \{hour\ of\ day, temperature, previous\ hour\ EC, previous\ week\ EC,$$

$$day\ of\ week, day\ of\ year, cloud\ cover\}$$

Similarly, 8 features have been selected for NNs:

$$FV2_{NN} = \{hour\ of\ day, temperature, previous\ day\ EC, previous\ week\ EC,$$
$$day\ of\ week, weekend\ info, day\ of\ year, total\ days\}$$

2.2 DATA STRUCTURE

2.2.1 KD tree

A KD tree (KD is a shorthand for k-dimensional) is a space-partitioning data structure for organizing points in a k-dimensional space. KD tree is a hierarchical decomposition of space along different dimensions and it can be used for answering k nearest neighbor queries in logarithmic time and linear space [30][32].

Fig.2.10: k-d tree decomposition on x-y plane

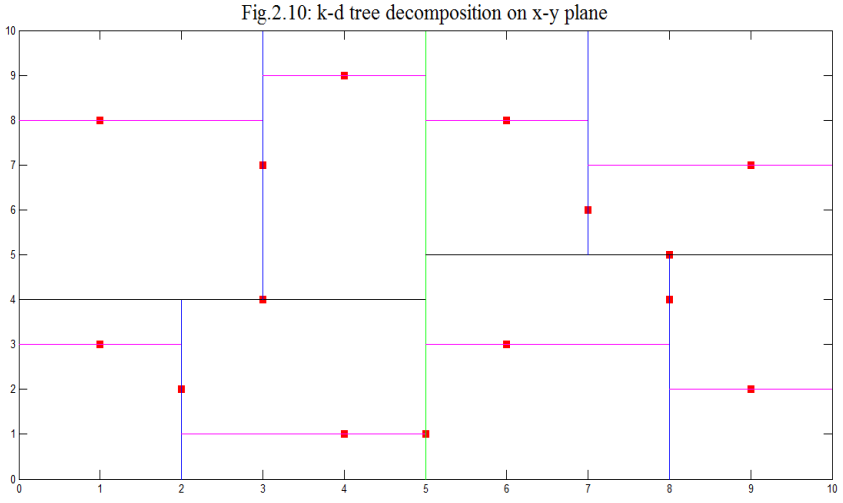

Fig.2.9 Decomposition of KD tree on x-y plane.

The construction of KD tree using Fig. 2.9 is as follows. Let the first partition be across the x-axis, shown by green line in the figure. The median of all data points will be

point (5, 1) and it is considered to be the root node of the tree, as shown in Fig. 2.10. The second partition is the horizontal line (black) to the left side of the first partition and this line passes through point (3, 4). This point will be considered as the left side branch from the root node on the KD tree diagram. Similarly the plane to the right side of the green line is divided by the black horizontal line which passes through point (8, 5). Thus, point (8, 5) will be considered as the right side branch from the root node. If we continue splitting the plane through the x- and y-axes alternatively and consider the median of the points as the next node, we can build the KD tree similar to Fig. 2.10.

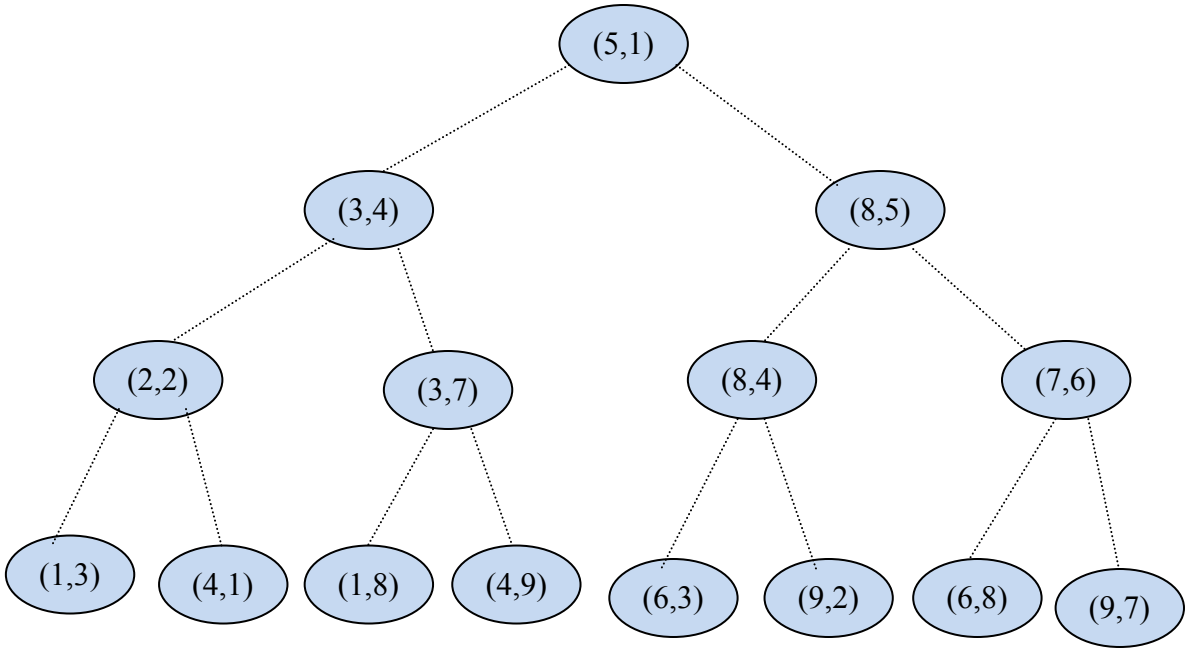

Fig.2.10: KD tree formation from 2d-plane decomposition

When splitting the plane and constructing the tree the choice of the hyperplane direction is explained in [30] as follows: "*if for a particular split the x-axis is chosen, all the points in the sub-tree with a smaller x value than the node will appear in the left sub-*

tree and all the points with larger x value will be in the right sub-tree." The same holds if y-axis is chosen, points with a smaller y value than the node will appear in the left sub-tree, while points with larger y value will be in the right sub-tree.

2.2.2 k-Nearest Neighbors (kNN) Similarity Search

The k nearest neighbors (kNN) similarity search is a method of finding the k elements of a data set *S* which are closest to a given query point *Q*. Similarity or closeness measures the strength of a relationship between two objects, while dissimilarity, on the other hand, measures the difference between two objects. In the kNN algorithm dissimilarity is measured by one of the following distance metrics: Euclidian, Manhattan, Chebyshev, or Mahlanobis.

Fig. 2.9: kNN search illustration using data with two features

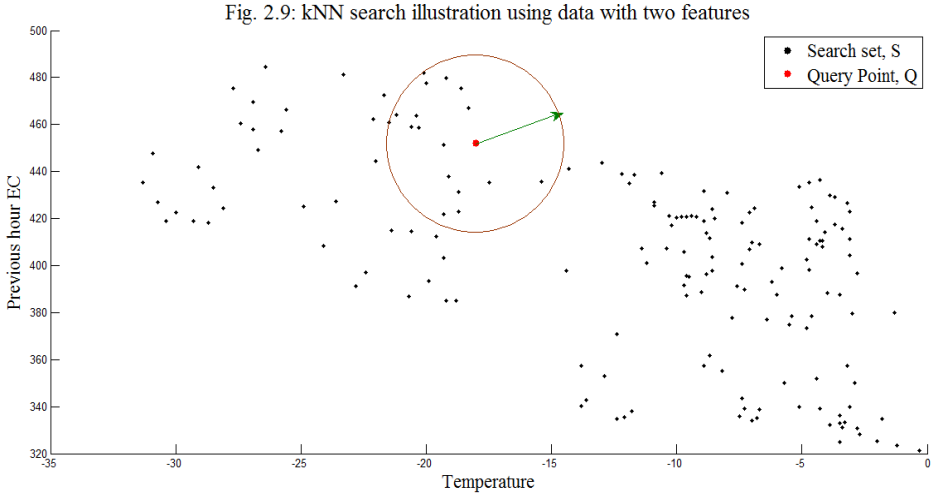

Fig.2.11: A kNN search in which 17 nearest points to the query point Q are selected and subscribed in a circle.

In this thesis we used Euclidian distance which is given by the following formula:

$$dist(q,s) = \sqrt{(q_1 - s_1)^2 + (q_2 - s_2)^2 + \cdots + (q_d - s_d)^2} \qquad (2.6)$$

Where, $q \in Q$ *is the query point, and* $s \in S$ *is the search point* and d represents the dimension of query vector. The inputs to the kNN algorithm are: a set of n data set S, a set of m query points Q and the number of nearest neighbors we are searching k. If Q is a vector of dimension d, then the output of the kNN search will be a $k \times d$ matrix.

Algorithm 2.1*: kNN similarity Search using brute-force search*

Input: *query Q, Data set (Search set) S, The number of objects we want to retrieve k.*

1. *Compute the distances between the query item Q and all the reference points in the training set S.*
2. *Sort them based on the distance to Q in non-decreasing order.*
3. *Return k points with smallest distance to Q.*

The naive implementation of kNN similarity search involves the brute-force computation of distances between all pairs of points in the dataset. For N samples in D dimensions, this approach scales as $O(DN^2)$ [39]. In this thesis we used a more efficient approach based on KD tree, in which the computational cost of kNN search can be reduced to $O(DN \log(N))$ or better.

3. Linear Regression

Linear regression (LR) models provide simple and interpretable description of how input variables are related to the output. This relationship is given as a linear function whose domain is the input variables and range is the output. LRs have been widely used for prediction in different areas, among others: temperature forecasting, financial market forecasting and electricity consumption forecasting etc.

In this thesis we used local linear regression (LLR) model, in which we applied the classical linear regression model on *chosen* closest data points to the test point. The LLR model can be seen as LR model plus kNN similarity search. Since the kNN similarity search is already discussed in the previous Chapter, this Chapter will focus on LR model. Finally, Section 3.3 discusses how the classical LR model was used with kNN similarity search to implement the LLR model.

3.1 Linear Regression Model

Regression analysis is a body of statistical methods dealing with the formulation of mathematical models that depict relationships among variables, and the use of those modeled relationships for the purpose of prediction and other statistical inferences [9]. A linear regression model assumes that the regression function $f(x)$ is linear in the input **x**.

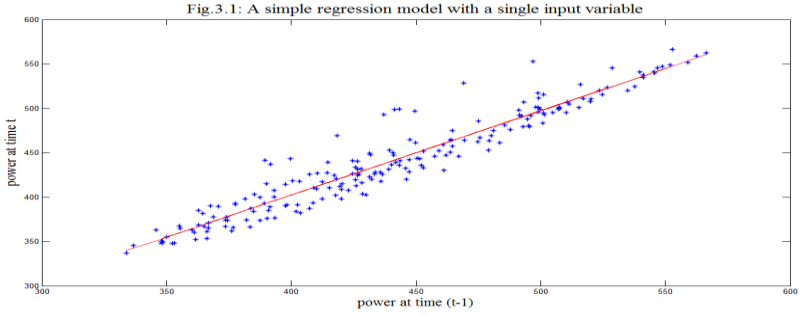

Fig.3.1: A simple linear regression model which maps EC at time t-1 to EC at time t.

The underlying function in Fig. 3.1 (red line) is given as $P_t = 22.56 + 0.95P_{t-1}$, and its parameters are computed from the data. Once we know the function we can compute (i.e., predict) P_t for any value of P_{t-1}. This only show how linear regression works in its simplest form, but most real life applications involve more than one input variables and computation of the parameters is a bit more complex.

Assume we are given a d-dimensional input X and we want to predict a real-valued output **y.** The linear regression model that relates the inputs to outputs is defined as:

$$f(x) = \beta_0 + \sum_{j}^{d} x_j \beta_j \qquad (3.1)$$

Where $\beta_j's$ are unknown parameters also called regression coefficients, and *d* is the input dimension. The independent variables x_j may represent [8]:

- Quantitative inputs;
- Transformations of quantitative inputs, such as log, $\frac{1}{x}$, etc.;
- Basis expansions such as x_i^2, x_i^3 etc. and;
- Interaction between variables, for example $x_i.x_j$.

In this thesis the independent variables comprises: quantitative inputs, such as temperature, cloud cover, wind speed, hour of day, etc. and autoregressive inputs like previous hour EC and previous week EC.

We represent the deviation of the response y_i from the underlying function by error e_i, and y_i is related to the independent variable x_i by the following equation:

$$y_i = \beta_0 + \sum_{j=1}^{d} \beta_j x_{ij} + e_i, \qquad for\ set\ of\ observations\ i = 1,2,\dots,n \qquad (3.2)$$

Substituting in (3.1) we get: $y_i = f(x) + e_i$ (3.3)

Where the data set is given as: $\{y_i, x_{i1}, \dots, x_{id}\}_{i=1}^{n}$, and the $e_1, \dots, e_n$ represent independent and normally distributed additive error components with mean of zero and unknown variance σ^2. The parameters β_0 and β_j are also unknown and should be computed from the training data using the least square method. For convenience, we write Equation (3.2) in a vector form as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.4}$$

Where,

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \cdot \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_{11} & \cdot & \cdot & \cdot & x_{1d} \\ x_{21} & \cdot & \cdot & \cdot & x_{2d} \\ \vdots & \cdot & \cdot & & \vdots \\ \cdot & & & \cdot & \cdot \\ x_{n1} & \cdot & \cdot & \cdot & x_{nd} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \cdot \\ \vdots \\ \beta_d \end{pmatrix}, \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \cdot \\ \epsilon_n \end{pmatrix}$$

3.2 The Least Square Method (LSM)

Before discussing the use of LSM in linear regression let's start by stating a more broad definition of LSM.

Definition: *The method of least square is a standard approach to the approximate solutions of over-determined systems, i.e., sets of equations in which there are more equations than unknowns. 'List squares' mean that the overall solution minimizes the sum of the squares of the errors made in the result of every single equation.* [34]

Training the linear regression model involves estimation of the parameters β_0 and β_j in (3.2). In a simple linear regression model the estimation of those parameters can be viewed as fitting the best straight line on the scatter plot like in Fig. 3.1. The least square

method is a method of estimating regression parameters that minimizes the residual sum of squares (also called overall discrepancy), which is given by [9]:

$$RSS = \sum (Observed\ response - Predicted\ response)^2 \qquad (3.5)$$

Where, the predicted response involves the unknown parameters $\beta_j's$, and the parameter values thus determined are called the least square estimates.

Given a set of the training input-output pairs, $\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$, we can estimate the parameters β using a least square method. We want to compute a value of β that minimizes the residual sum of squares, which is given as:

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{d} x_j \beta_j)^2 \qquad (3.6)$$

If the data of input dimension *d*, and length *N* is arranged as follows:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nd} \end{bmatrix} \quad y = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{Bmatrix} \quad \hat{\beta} = \begin{Bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_d \end{Bmatrix}, \qquad (3.7)$$

Then the residual sum of the square can be written as:

$$RSS(\beta) = (y - X\beta)^T (y - X\beta) \qquad (3.8)$$

The value of β that minimizes the residual square of sum is computed by setting the derivative of equation (3.8) to zero and the resulting coefficient vector $\hat{\beta}$ is given as:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T y} \qquad (3.9)$$

Prediction: The goal of regression analysis is to estimate the response corresponding to a specified value of the input variable.

$$Observation\ \mathbf{y} = (training\ input\ matrix\ \mathbf{X}) \times (parameter\ \boldsymbol{\beta}) \qquad (3.10)$$

After computing the least square estimate of the parameter using (3.9), the prediction at an unseen input matrix $\mathbf{X}^*$ can be computed as:

$$Prediction\ \mathbf{y}^* = (test\ input\ matrix\ \mathbf{X}^*) \times \left(estimated\ parameter\ \widehat{\boldsymbol{\beta}}\right) \qquad (3.11)$$

For example, suppose we only have a single input variable x, and we want to predict the electricity consumption y for a specified input variable $\mathbf{x}^*$. The expected response at a value $\mathbf{x}^*$ of the independent variable $\mathbf{x}$ is given as follows:

$$E(y|\, x^*) = \alpha + \beta x^* \qquad (3.12)$$

An unbiased estimator of (3.12) will be: $\hat{\alpha} + \hat{\beta}x^*$, where:

- $\hat{\alpha}$ is the least square estimate of α and is given by: $\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$,
- $\hat{\beta}$ is the least square estimate of β and is given by: $\hat{\beta} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{S_{xy}}{S_x^2}$

A validation test for LR model: The validity of the linear regression model evaluated by examining how much of the variation in the values of the response variable can be explained by the linear model. After computing the least square estimates $\hat{\alpha}$ and $\widehat{\beta}$, we can view any observed y_i as consisting of the following two components [9]:

$$\underset{(Observed\ value)}{y_i} = \underset{Explained\ by\ LR}{(\hat{\alpha} + \hat{\beta}x_i)} + \underset{Deviation\ from\ LR}{(y_i - \hat{\alpha} - \hat{\beta}x_i)} \qquad (3.13)$$

If the residuals (the last component in (3.13)) are all zero which is an ideal situation of perfect fit, we can say that all the observations are explained by the linear model. By representing the overall deviation from linearity as the squared sum of the residuals:

$$SSE = \sum (y_i - \hat{\alpha} - \hat{\beta} x_i)^2 = S_y^2 - \hat{\beta}^2 S_x^2 \qquad (3.14)$$

Here, $S_y^2 = \sum (y_i - \bar{y})^2$ is the total variability of y and *SSE* stands for sum of squares due to error. By computing for S_y^2 from (3.14), we obtain the following [9]:

$$\underset{\substack{Total \\ variation \\ in\ y}}{S_y^2} = \underset{\substack{Variation \\ explained\ by \\ linear\ relation}}{\hat{\beta}^2 S_x^2} + \underset{\substack{Residual \\ unexplained \\ by\ LR}}{SSE} \qquad (3.15)$$

If the linear regression model provides a god fit for the data, the last term in the above equation (the *SSE*) will be close to zero. An index that measure how well the linear model fits the data is given as a ratio of the *SS* explained by linear relation and the total *SS* in y as follows:

$$r^2 = \frac{\hat{\beta}^2 S_x^2}{S_y^2} = \frac{S_{xy}^2}{S_x^2 S_y^2} \qquad (3.16)$$

Here we can see that *r* is equivalent to the sample correlation coefficient discussed in section 2.2 and r^2 close to 1 means the linear regression is a valid model for the given problem.

3.3 IMPLEMENTATION

The feature vector (FV) for LLR comprises 7 variables and it is similar to the FV used for GPs.

$$FV = \{hour\ of\ day, temperature, previous\ hour\ EC,$$
$$previous\ week\ EC, day\ of\ week, day\ of\ year, cloud\ cover\}$$

The kNN similarity search was applied to the 7 dimensional candidate training sets to select *k* closest points to a query point (i.e., the test inputs from LLR's perspective). After experimenting for different *k* values, we picked $k = 75$ as an optimal

k value. Thus the training input to the LLR model is a *75×7* matrix form. Algorithm 3.1 illustrates how the proposed model works.

***Algorithm 3.1**: Local Linear regression (LR) model for 24 hour prediction*

1. ***Input** data for selected features and EC, the value of k*
2. ***do***
    a. *specify the test point and the candidate training set*
    b. *select k closest points to the test point from candidate training sets using kNN search*
    c. *use the output from 4 as training set for LR*
    d. *compute the regression parameters using LSM*
    e. *predict EC at t + counter (i.e., compute* $predictedEC_{t+counter}$*)*
    f. $previousHourEC_{t+counter+1} = predictedEC_{t+counter}$ *(we use the predicted value as previous hour value on the next iteration)*
    g. *increase counter by 1*
3. ***while** (counter<=24)*
4. ***Output:** return predicted electricity consumption for the next 24 hours*

Because of its strong influence on 'current' EC, we considered the previous hour EC as one of the features. However the requirement from the company (Eidsiva Energy) demands prediction over 24 hours ahead and we don't know what the previous hour EC was, for the rest of the prediction period except for the first hour of the prediction period. Thus, we designed an iterative model in which we predict the next hour EC, and append it to the previous hour EC column for the next prediction. In addition to picking the previous hour EC, this recursive implementation allows us to apply local regression to data points closest to a single prediction point.

4 Neural Networks

A Neural network (NN) is an information processing structure that is inspired by the way human brain processes information. An NN consists of simple processing units called neurons, which communicate by sending signals to each other over a large number of weighted connections. Neural networks develop information processing capabilities by learning from examples. The learning process comprises adapting the network in a way that it will produce the correct output for the set of examples. The resulting network then generalizes the knowledge to unseen situation when presented with cases not found in the set of examples. [29][13]

4.1 Mathematical Model of Neural Network

A simple neuron model: The fundamental building block for neural networks is a neuron – a programming construct that mimic the properties of biological neurons. A single neuron model, shown in Fig. 4.1, consists of three distinct functional operations:

- *The weight function* – input multiplied by weight
- *The net input function* – sum of weight function and the bias, and
- *The activation function* – computes output from net input

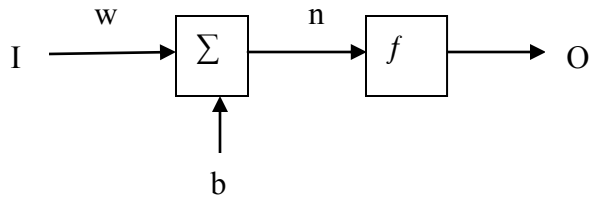

Fig. 4.1: A simple neuron model

$$O = f(wI + b) \tag{4.1}$$

Where, *I* is input, *w* represents weight, *b* represent bias, *n* is net input, *O* is output and *f* represents an activation function. The weights are numbers that change as neural network learns. A mathematical model of multiple inputs NN is given as follows:

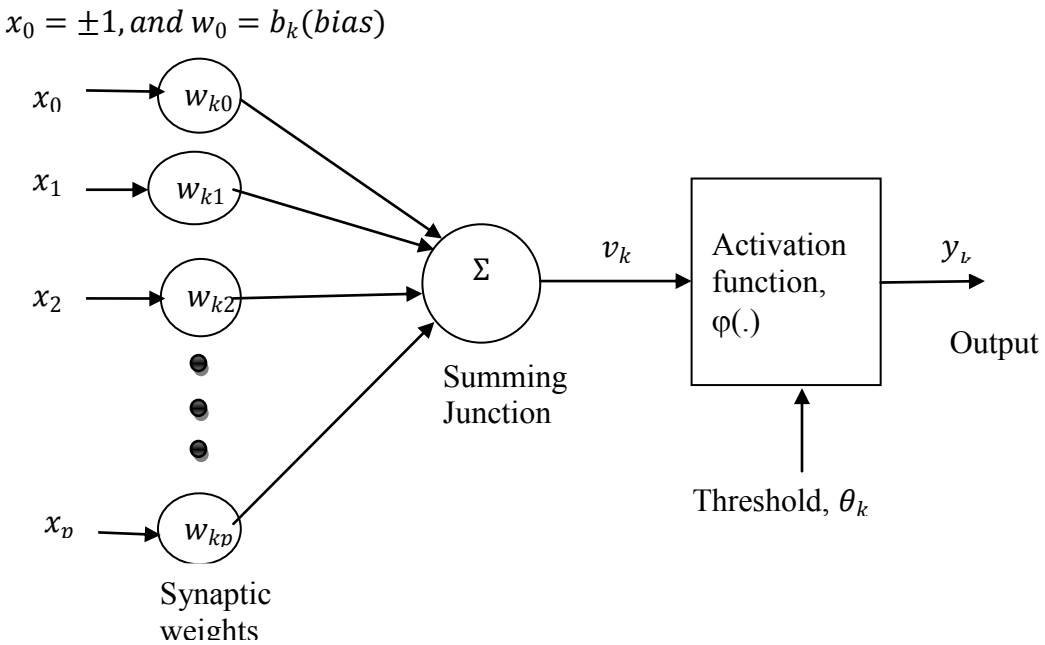

Fig. 4.2 [29]: Mathematical model of a neural network

V_k, in the above Figure, represents the weighted sum of inputs given by:

$$V_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (4.2)$$

The output of the neuron y_k is computed from the V_k and threshold θ_k through the activation function block. Thus, the net input to activation function will be:

$$net_k = v_k + \theta_k \qquad (4.3)$$

Assuming the TANH activation function is used, the activation value (output) of unit *k* is given by:

$$y_k = f(net_k) = \frac{e^{2net_k} - 1}{e^{2net_k} + 1} \tag{4.4}$$

4.2 ACTIVATION FUNCTIONS

An activation function controls the amplitude of the output. They are used to scale data output from a layer. We will discuss three commonly used activation functions namely: hyperbolic tangent function, sigmoid function and linear function.

The Sigmoid Activation Function: A sigmoid activation function takes input of any value and scales the output into the range *0 to 1*. Sigmoid function returns only positive values, and it is represented by the following formula:

$$\frac{1}{1+e^{-x}} \tag{4.5}$$

Hyperbolic Tangent Activation Function: As mentioned previously, the sigmoid activation function does not return values less than zero. The hyperbolic tangent activation function takes input of any value and scales the output into the range *-1 to 1*. This activation function is useful particularly when keeping the sign of input data is important. The TANH function is given as:

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4.6}$$

Linear Activation Function: A less commonly used activation function is linear function, and is defined as follows:

$$f(x) = x \tag{4.7}$$

However, the practical application of linear activation function is limited, since it simply passes the input without modifying.

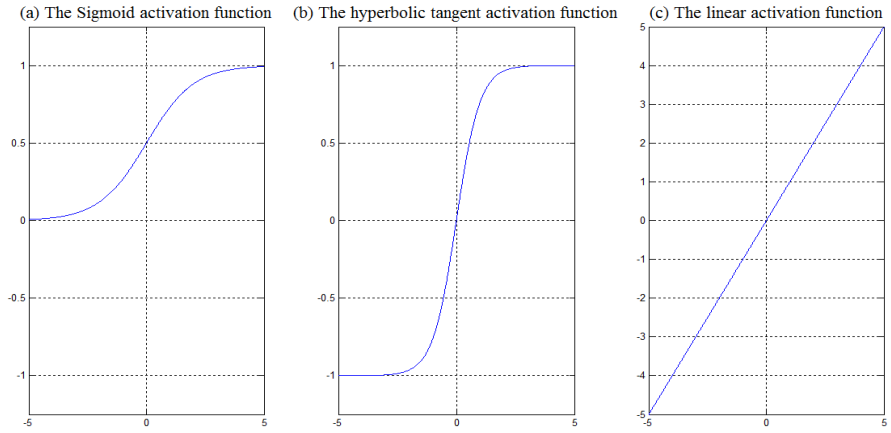

Fig.4.3: Activation functions (a) Sigmoid, (b) Hyperbolic tangent (c) linear

4.3 NEURAL NETWORK ARCHITECTURE

While studying a single neuron could be useful to understand the basic concept, most of the real life applications require a number of interconnected neurons. A neural network consists of three types of units:

- *The input units* – receive data from outside of the NN.
- *The hidden units* – both the input and output data from those units remain within the NN.
- *The output units* – sends data from the NN to the outside world.

Feedforward neural network: In feedforward neural networks, neurons are only connected forward as shown in Fig 4.4. Each layer of the neural network contains connections to the next layer but there are no connections back. Thus the data flow from

input to output unit is strictly in the forward direction. An alternative topology to feedforward NN is the recurrent NN, which contains feedback connections. Contrary to the feedforward networks, the dynamic properties of the network are important in recurrent NNs. [3][29][13]

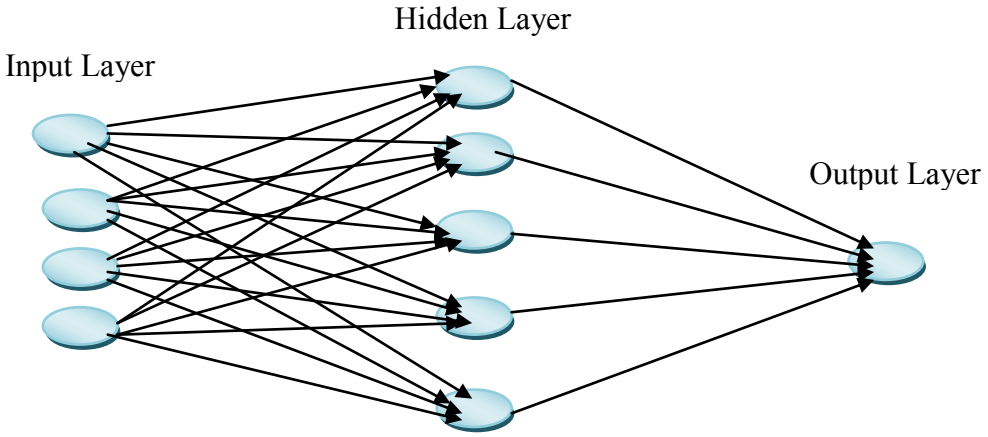

Fig.4.4: Feedforward Neural Network with one hidden layer

Layers, synapses, and weight matrix: Neural networks are made up of layers; a layer in turn is a collection of similar neurons. Layers are connected with one another through synapses and hold an array of threshold values −one threshold value for each of the neurons in the layer. The synapses contain the weight matrix, which allows the connections between each of the source layer neurons to have its connections weighted to the target neuron layer. The NN learns by adjusting the weights and the threshold. [37]
The weight matrix W between two layers is given by:

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ w_{s1} & w_{s2} & \cdots & w_{sk} \end{bmatrix} \tag{4.8}$$

Where, k is number of elements in source layer and s is the number of neurons in the target layer. The row indices on the elements of matrix W indicate the destination neuron of the weight, and the column indices indicate the input source for the weight. [6]

Designing NN Architecture for a particular application: The performance of a neural network is highly influenced by the number of hidden layers and number of neurons in each hidden layer. Though the hidden layers do not directly interact with external environment, they have a tremendous influence on the final output. Therefore, both the number of hidden layers and the number of neurons in each of these hidden layers should be carefully considered. Unfortunately, there exists no strict rule to follow and therefore we need to experiment by considering different options and observing the performance of each option. The following are some of tips that we can use as a starting point when deciding the number of hidden layers and number of hidden neurons [3]:

- NN with one or two layers work for most practical problems
- The number of hidden neurons should be between the size of the input layer and the size of the output layer
- The number of hidden neurons should be $^2/_3$ the size of the input layer plus the size of the output layer
- The number of hidden neurons should be less than twice the size of the input layer.

4.4 Training Neural Networks

After the network has been configured, as in Section 4.3, the network parameters (i.e., weights and threshold) need to be adjusted, in such a way that the network performance will be optimized. This process is referred to as training the network. There

are two forms of training that can be employed in NNs, supervised and unsupervised trainings. In unsupervised training the NN is provided with training sets, but not with anticipated outputs. On the other hand, supervised training involves providing the NN with both training sets and the anticipated outputs. In this thesis both the training inputs X and the anticipated outputs y are provided and thus we will focus on supervised training.

Supervised training: In supervised training, the neural network adjusts the values in the weight matrix based on the differences between the expected output and the actual output. The exact process of learning in NNs is determined by the learning algorithm used. Some of the commonly used supervised learning algorithms are: backpropagation, simulated annealing and genetic algorithm. Error calculation is also an important part of training NNs and all training algorithms are aimed at reducing the rate of error. In determining the error rate, two error values need to be computed:

- the error for each element of the training set as it is processed, and
- the average of the errors for all of the elements of the training set across each sample

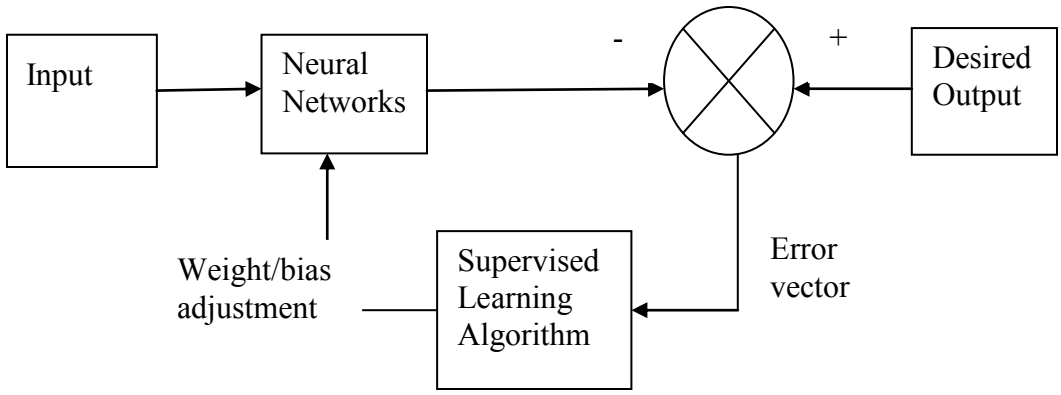

Fig.4.5: Supervised training of neural network

Computation of the values of the weight matrix involves many steps in the training process. It begins by creating a random weight matrix. An output error is then calculated for each element of the training set, and is used as a steppingstone in the calculation of the total error for the entire training set. The error function represents the performance of the network and is used by supervised learning algorithms for iterative optimization. [3]

The Backpropagation Algorithm: One of the most popular supervised learning algorithms for feedforward NNs is the backpropagation. The backpropagation algorithm uses a gradient-descent technique to minimize the error function. The values used to adjust the weight matrix are obtained by calculating the partial derivative of the error function with respect to each weight w_{ij}. This gives a gradient vector representing the steepest increasing direction in the weight space. [13][38]

Error function is used to calculate the error between the ideal and actual output of a neural network. Assuming that we have N training input points and a single output unit, the error for each training point p is given by:

$$E_p = (ideal_p - actual_p)^2 \tag{4.9}$$

Where, 'actual' represents the actual output and 'ideal' is for ideal/anticipated output. Thus, the total error is given as:

$$E = \frac{1}{2}\sum_{p=1}^{N}(ideal_p - actual_p)^2 \tag{4.10}$$

The actual weight update values at iteration *t* are computed using the following equation:

$$\Delta w_{ij}(t) = -\epsilon \frac{\partial E}{\partial w_{ij}}(t) + \alpha \Delta w_{ij}(t-1) \qquad (4.11)$$

Where, w_{ij} is the weight between source neuron *i* and destination neuron *j*; and ϵ is a value between *0 and 1*, which represents learning rate (or the step size). The α represents the momentum term that determines how much influence the previous iterations' learning will have on the current iteration's update. Momentum is used to prevent a training algorithm from getting trapped in a local minimum.

Algorithm 4.1: *The backpropagation training algorithm*

1. ***Input:*** *training data, the learning rate ϵ, and momentum parameter α*
2. *Create a random weight matrix*
3. *Input the training data*
4. *set the step size and momentum*
5. ***Do***
    a. *Compute the output for each layer and units in a forward direction*
    b. *Calculate errors between the ideal and actual output at the output neuron*
    c. *Compute the weight update values for each of the preceding layers by back propagating the error*
    d. *Adjust the weights based on c*
    e. *Increase epoch counter by 1*
6. ***While (****epoch < max_Epoch) && (Error > min_Error)*

The Resilient Propagation (RPROP) algorithm: This is the learning algorithm we used in this thesis. The RPROP is a local adaptive learning scheme, which performs batch learning in feed-forward neural networks. The basic principle of RPROP is to eliminate the harmful influence of the size of the partial derivative on the weight step. The RPROP algorithm considers only the sign of the derivative to indicate the direction of the weight update and the size of the weight update is solely determined by an individual update value $\Delta_{kj}(t)$ for each weight w_{kj} . The evolution of the update value $\Delta_{kj}(t)$ is determined by a second learning rule, which is based on the observed behavior of the partial derivative during two successive weight-steps given by: [13]

$$\Delta_{kj}(t) = \begin{cases} \eta^{+}.\Delta_{kj}(t-1), & if \ \frac{\partial E}{\partial w_{kj}}(t).\frac{\partial E}{\partial w_{kj}}(t-1) > 0 \\ \eta^{-}.\Delta_{kj}(t-1), & if \ \frac{\partial E}{\partial w_{kj}}(t).\frac{\partial E}{\partial w_{kj}}(t-1) < 0 \\ \Delta_{kj}(t-1) & Otherwise \end{cases} \tag{4.12}$$

Where $0 < \eta^{-} < 1 < \eta^{+}$

As can be seen from (4.12), the change in sign of the partial derivative of weight w_{kj} shows that the last update was too big and the algorithm has jumped over a local minimum, thus the update value Δ_{kj} is decreased by the factor η^{-}. On the other hand, if the derivative retains its sign, then it shows that the update value is slightly increased to accelerate convergence in shallow regions. [13] The rule for the weight update itself is given as follows:

$$\Delta w_{kj}(t) = \begin{cases} -\Delta_{kj}(t), & if \ \frac{\partial E}{\partial w_{kj}}(t) > 0 \\ \Delta_{kj}(t), & if \ \frac{\partial E}{\partial w_{kj}}(t) < 0 \\ 0 & Otherwise \end{cases} \tag{4.13}$$

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t) \qquad (4.14)$$

Equations (4.13 & 4.14) model rule for updating the weight, which can be interpreted as:

- If the derivative is positive, this shows increase in error value, the weight decrease by its update value.
- If the derivative is negative, the weight increases by the update value.
- Otherwise the weight will be left unchanged.

4.5 IMPLEMENTATION

To implement neural networks for STFEC, we used a C# library known as Encog. The usage of Encog framework is explained in [37] & [38]. Multi layered feed forward NN was implemented with the following parameters:

- 8 inputs neurons,
- 4 hidden layers with 7 neurons in each, and
- 1 output neuron

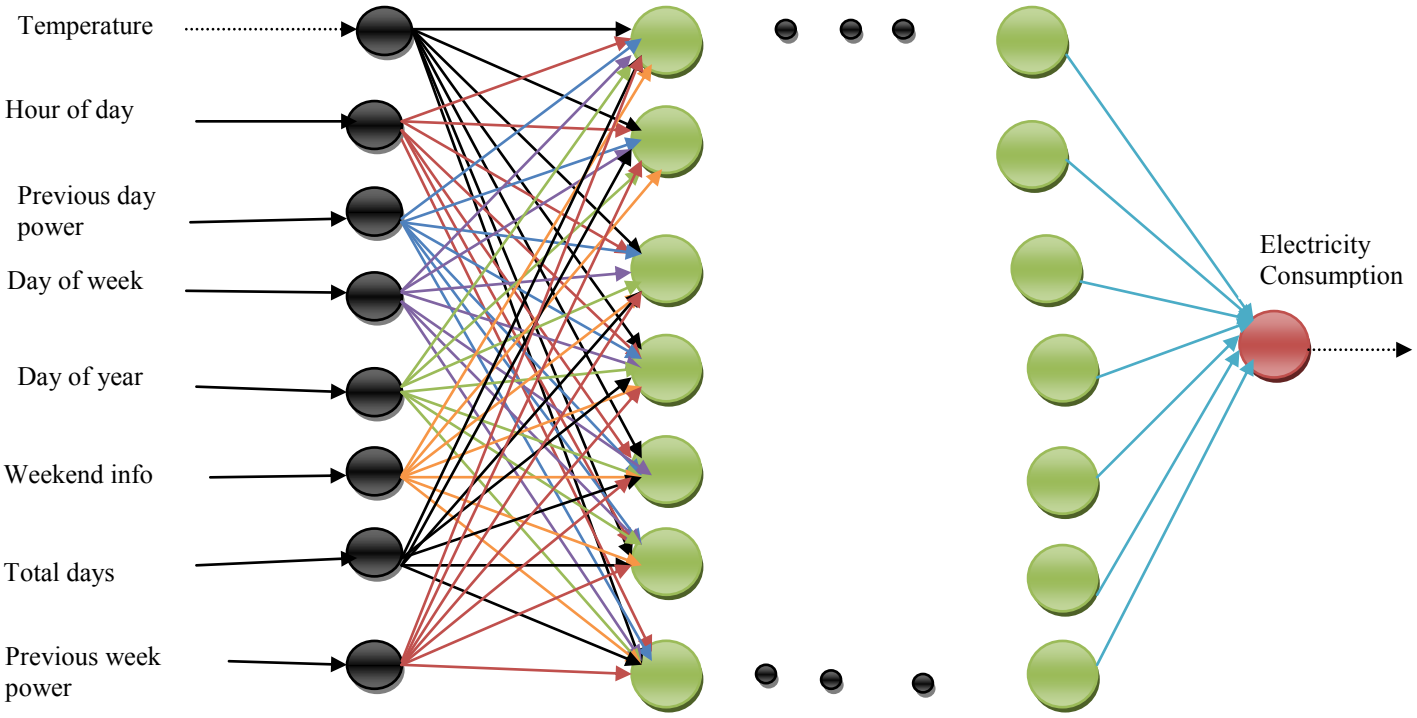

Fig.4.6: A feedforward neural network for electricity consumption prediction

In this implementation, the error function is computed using root mean square errors, and the resilient propagation (RPROP) algorithm is used to train the network. As shown in Fig. 4.8, the feature vectors are quite different from the one used in LLR and GPs. This is because training NNs is quit complex compared to LLR and GPs. To use the predicted EC as previous hour EC in training, the training process should be updated for every predicted EC value. The initial weight matrices are random (see Algorithm 4.1) and the number of iterations that minimize the average prediction error is unknown (it changes every time we train the network). A more practical way of using Feedforward RPROP NN is by performing a series of training and validation tests and save the network with best performance. NNs are strong at learning patterns; therefore a well tested and optimized NN can be used for long period of time.

5 Gaussian Processes

As mentioned in previous two sections, regression is a process of estimating the underlying function $f(x)$ that maps a set of inputs x to target y. In linear regression model we assumed this function to be linear and the least square method is used to compute the parameters from training data. The underlying function may also be assumed to be quadratic, cubic or even non-polynomial. The problem of such models is explained in Rasmussen [1] stating that "*if a model based on a certain class of functions is used and the target function is not well modeled by this class, then the prediction will be poor*."

Gaussian processes, on the other hand, are a family of stochastic processes that deal with more than one possible reality of how a process might evolve. Rather than assuming a particular deterministic function (path), GPs consider probability of other paths that a process may follow. GPs learn the most probable path from the data − a desirable flexibility that reduces errors caused by wrong assumption of the underlying function.

5.1 Gaussian Process Regression

The definition of GPs is given in Rasmussen [1][4] as: *A Gaussian process is a collection of random variables, any finite number of which have consistent joint Gaussian distributions.* A GP is fully specified by its mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(x) \backsim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big)^{2} \tag{5.1}$$

[2] The function *f* is distributed as a Gaussian Process with mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$.

Where the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ are given respectively as:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{5.2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\big[\big(f(\mathbf{x}) - m(\mathbf{x})\big)\big(f(\mathbf{x}') - m(\mathbf{x}')\big)\big] \tag{5.3}$$

If we let the mean function to be zero, Bayesian linear regression model will be:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} \text{ with prior } \mathrm{w} \sim \mathcal{N}\big(0, \Sigma_{\mathrm{p}}\big) \tag{5.4}$$

By substituting equation (5.4) into (5.3) we obtain:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}\mathbf{w}^T] \phi(\mathbf{x}') = \phi(\mathbf{x}) \Sigma_p \phi(\mathbf{x}') \tag{5.5}$$

$$m(\mathbf{x}) = \phi(\mathbf{x}) \mathbb{E}[\mathbf{w}] = 0 \tag{5.6}$$

This shows that $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian with zero mean and covariance $\phi(\mathbf{x}) \Sigma_p \phi(\mathbf{x}')$.

Assuming noise free observations, i.e., known $\{(\mathrm{x}_i, f_i) | i = 1, \ldots, n\}$, the joint distribution of the training outputs, $\boldsymbol{f}$, and the test outputs f^* according to the prior is given as:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}^* \end{bmatrix} \backsim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \tag{5.7}$$

To prediction we need to compute the posterior distribution over functions after observing the data. In this case we only consider those joint prior distributions that agree with the observed data points. That means, the posterior distribution will be the probability of function values $\boldsymbol{f}^*$ at test input X^* given the training input X, the test input X^* and observation $\boldsymbol{f}$ (here the observation $\boldsymbol{y}$ is represented as $\boldsymbol{f}$ because of the noise free assumption). Thus, the posterior distribution can be written as:

$$p(\boldsymbol{f}^*|X^*, X, \boldsymbol{f}) \backsim \mathcal{N}\{K(X^*, X)K(X, X)^{-1}\boldsymbol{f}, K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)\} \quad (5.8)$$

However, in most realistic modeling the function $\boldsymbol{f}$ itself is not known in advance. Instead, we only know the noisy version y which is given as:

$$y = f(x) + \epsilon \quad (5.9)$$

Under noisy observation, the mathematical modeling for prior and posterior distribution is similar to the one given under noise free observation, except that we substitute the y (noisy version) for **f** and, $K(X, X) + {\sigma_n}^2 I$ for $K(X, X)$. Where ${\sigma_n}^2$variance of additive independent identically distributed Gaussian noise ϵ. Thus, the predictive distribution is given as:

$$p(\boldsymbol{y}^*|X^*, y, X, \theta, \sigma^2) \sim \mathcal{N}\big(m(X^*), v(X^*)\big) \quad (5.10)$$

Where the mean and covariance functions are given by:

$$m(X^*) = K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1}\boldsymbol{y} \quad (5.11)$$

$$v(X^*) = K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X^*) \quad (5.12)$$

From the above equation, we can notice the following concepts [1]:

- The predictive covariance function $v(X^*)$is independent of the observation ***y*,** while the predictive mean $m(X^*)$ is a linear combination of observations ***y*.**
- The predictive covariance $v(X^*)$ is given as the difference between two terms:
  i. the prior covariance of test input $K(X^*, X^*)$ minus
  ii. a positive term $K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X^*)$

  This shows that when the test input *X** is far away from the training input *X*, the positive term becomes almost zero and the predictive covariance becomes almost

the same as the prior covariance. This means, when the training input is far away from the test input its contribution to prediction becomes almost negligible. This concept is vital to understand how the proposed kNN similarity search selects the most relevant training inputs for a given GP test cases.

To make predictions we find a posterior density over the latent variables and then integrate over that posterior density. The marginal likelihood is equal to the integral over the product of the likelihood function and the prior density. For Gaussian likelihood, the product of two Gaussians will be Gaussian and the integral can be solved analytically. Thus, the Gaussian likelihood is given by [12]:

$$p(y|X,\theta,\sigma^2) = \int p(y|X,\theta,\sigma^2)p(f|X,\theta)df \quad (5.13)$$

$$= \int \mathcal{N}(f,\sigma^2 I)\mathcal{N}(0,K)df \quad (5.14)$$

$$= \frac{1}{(2\pi)^{\frac{n}{2}}|K+\sigma^2 I|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}y^T(K+\sigma^2 I)^{-1}y\right) \quad (5.15)$$

Where K is an $n \times n$ covariance matrix, θ is hyperparameters and σ represents the noise variance. By computing the logarithm of (5.15), we get the log marginal likelihood:

$$\log p(y|X,\theta,\sigma^2) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|K+\sigma^2 I| - \frac{1}{2}y^T(K+\sigma^2 I)^{-1}y \quad (5.16)$$

5.2 Covariance Functions

A covariance function comprises our assumptions regarding the underlying function that we wish to learn, and it is an important ingredient in GP predictor. As was stated in Chapter 1, the goal of supervised learning is to infer the input−output mapping from an empirical data. Under such circumstances, the notion of similarity between data

points is crucial and the covariance functions in GP define the closeness/similarity between data points [1].

Several alternative covariance functions have been used in GPs. The most widely used (also implemented in this thesis) is the squared exponential (SE) covariance function which is given as:

$$K(x, x') = \delta^2 \exp\left(-\frac{(x - x')}{2\gamma^2}\right) \tag{5.17}$$

Where δ & γ are hyper parameters. The δ controls the smoothness of the curve while γ represents the characteristic length scale. The SE covariance function is stationary (invariant to transition, i.e., $K(x, x') = K(x - x')$) and infinitely differentiable.

A non-stationary covariance function that can be useful to deal with problems known to be periodic is the periodic covariance function. Mathematically given as:

$$K(x, x') = \exp\left(-\frac{2sin^2\left(\frac{x - x'}{2}\right)}{\lambda^2}\right) \tag{5.18}$$

Some of other covariance functions are listed as follows:

- Seasonal trend (quasi-periodic smooth): expressed in Rasmussen [1] as it was used for CO_2 prediction to model seasonal trends when the period is quite long (e.g. period = one year).
- Matern class covariance function is recommended as an alternative to the SE kernel because such strong smoothness as in SE is unrealistic for modeling of many physical processes. Mathematically defined as:

$$K_{matrn}(x - x') = \frac{2^{1-v}}{\Gamma(v)}\left(\sqrt{2v}(x - x')\right)^{v} K_v\left(\frac{\sqrt{2v}(x - x')}{l}\right) \qquad (5.19)$$

Where, v and l are positive parameters and K_v is a modified Bessel function.

- Short- and medium-term anomaly (rational quadratic)

Some problems are best modeled by combination of covariance functions instead of one particular covariance function. As stated in [27], constructing new covariance from old ones could improve prediction accuracy. There are several ways to do so, for example *sum, product, and convolution of covariance functions* can be used.

5.3 Learning Hyperparameters

As discussed in Section 5.2, a GP prediction model relies on the covariance function. However, we do not know a priori the most appropriate hyperparameters and noise variance [12]. One way of learning the appropriate hyperparameters is to think the marginal likelihood in (5.13) as the likelihood of the hyperparameters θ and noise variance σ^2, and then to find the values of θ and σ^2 that maximize the likelihood. Once we found the maximum likelihood hyperparameters θ_{ML} and maximum likelihood noise variance σ^2_{ML}, we can substitute $\theta_{ML}$and σ^2_{ML} in place of θ and σ^2 respectively in Equation (5.10).

In this thesis we used an approach known as maximum a posteriori (MAP), which is close to the maximum likelihood, but in MAP we incorporate our prior belief about the hyperparameters and noise variance in the learning process. This can be done by finding the posterior density over the hyperparameters and noise variance as follows [12]:

$$p(\theta, \sigma^2|\mathrm{y}, \mathrm{X}) \sim p(y|X, \theta, \sigma^2)\mathrm{p}(\theta, \sigma^2) \qquad (5.20)$$

As can be seen in (5.20), the posterior density is proportional to the likelihood function times the prior density. Instead of maximizing the likelihood function (as in maximum likelihood approach), we find the hyperparameters and noise variance that maximize the posterior density. The values found in this way are referred to as maximum a posterior (MAP), and prediction is given by substituting θ_{MAP} for θ and σ^2_{MAP} for σ^2 in equation (5.10).

Another alternative approach is the Markov Chain Monte Carlo (MCMC), which is used to simulate the posterior distribution by numerically generating a set of samples. This method is not implemented in this thesis and we will not discuss it here, but interested readers can refer Boyle [12] & Vanhatalo et al [2] for more detailed information about the MCMC.

5.4 Working with Large Datasets

The challenges with using naïve implementation of GP models are the fast increasing computational and memory requirements. The evaluation of the inverse and determinant of the covariance matrix in marginal likelihood, Equation (5.15), and its gradient scales as $O(n^3)$ in time, where n is the number of training samples. This limits the applicability of GPs to moderate size data sets [2][1].

In recent years, dozens of research papers have come out suggesting different ways of handling larger data in GP models. Given the importance of including sufficient data during the training process and the powerfulness of GPs for regression, this is not a surprise. Làzaro-Gredilla 2010 [5] is one of the recent works which proposed an algorithm called Sparse Spectrum Gaussian Process (SSGP). The main idea of SSGP is to

'sparsify' the spectral representation of the GP, resulting in reduced computational cost: from $O(n^3)$ to $O(nm^2)$, where m is the number of spectral points.

In this thesis, we used a novel approach, in which kNN similarity search algorithm (explained in Section 2.3) was implemented to select 'k' training sets, among n training candidates, that are closest to a given query point. This reduces the computational requirement from $O(n^3)$ to $O(k^3)$, where k is a constant in the kNN similarity search algorithm representing the number of closest points we want to retrieve.

5.5 IMPLEMENTATION

For the implementation of GPs we used the kernel functions provided by GPStuff toolbox (detail about GPStuff can be found in [2]). Particularly we used the squared exponential covariance function and Gaussian likelihood. Optimum hyperparameters are computed using maximum a posteriori (MAP).

The features that determine the input dimension are selected by using methods discussed in Section 2.1.3. We selected 7 variables that give optimum performance:

$$FV = \{hour\ of\ day, temperature, previous\ hour\ EC,$$
$$previous\ week\ EC, day\ of\ week, day\ of\ year, cloud\ cover\}$$

To predict 24 hours ahead as required by Eidsiva energy, but at the same time to include the previous hour EC as one of the features, we used recursive prediction that is similar to the one discussed in Section 3.3. The kNN similarity search was applied to the 7 dimensional candidate training sets, to select k closest points to a given query point (test inputs from GP's perspective). After experimenting for different k values, we picked $k = 700$ as an optimal k value. Thus the training input to the GPs is a 700×7 matrix form. Algorithm 5.1 illustrates how the proposed model works.

***Algorithm 5.1**: Gaussian processes (GPs) with kNN similarity search: 24 hour ahead prediction*

1. ***Input** data for selected features and EC, the k value, prior hyperparameters, type of the kernel function*
2. ***do***
    a. *specify the test inputs and the candidate training set*
    b. *Apply the kNN search algorithm to select k closest points to the test points*
    c. *use the output from 4 as training set for GP*
    d. *compute prior distribution on hyperparameters*
    e. *compute the posterior density over hyperparameters and noise variance*
    f. *find the hyperparameters and noise variance to maximize the posterior density*
    g. *use the maximum a posteriori (MAP) values from f in predictive distribution*
    h. *predict EC at t + counter (i.e., compute* $predictedEC_{t+counter}$*)*
    i. $previousHourEC_{t+counter+1} = predictedEC_{t+counter}$ *(we use the predicted value as previous hour value on the next iteration)*
    j. *increase counter by 1*
3. ***while** (counter<=24)*
4. ***Output:** return predicted electricity consumption for the next 24 hours*

6 Experimental Results

In this chapter we will present results of our experiments with brief explanations of figures and parameters in the figure. The discussion and conclusion are kept for the next Chapters.

6.1 Experimental setup

Dataset: We used an hourly empirical data from 01.01.2008 to 13.02.2011, which is provided by Eidsiva Energy. The experiments are conducted by dividing the dataset into training set and test set. All the experimental results represent observations of tests from 01.01.2010 to 13.02.2011, except for the experiment in Section 7.1 which is result of tests on dataset from 24.01.2011 to 13.02.2011.

Table 6.1: Parameter settings

	Chosen value	Studied range
k in the kNN similarity search	700 for GP, 75 for LR	$1-2000$
Number of hidden layers in NN	4	$1-6$
Number of neurons in each layer	7	$4-20$

Performance evaluation: The performances of the predictors are evaluated by using the following empirical measures:

$$Absolute\ percentage\ error = \left|\frac{Actual - Predicted}{Actual}\right| \times 100\% \qquad (6.1)$$

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{Actual_i - Predicted_i}{Actual_i}\right| \qquad (6.2)$$

The correlation coefficient between the actual EC and the predicted EC is given by:

$$CC = \sum_{i=1}^{N} \frac{(Actual_i - \overline{Actual})(Predicted_i - \overline{Predicted})}{\sqrt{\sum_{i=1}^{N}(Actual_i - \overline{Actual})^2}\sqrt{\sum_{i=1}^{N}(Predicted_i - \overline{Predicted})^2}} \quad (6.3)$$

6.2 Evaluation of The Predictors

In this Section we will present plots of the experimental results of the three predictors: NNs, GPs, and LLR. The plots for each predictor are one to one, and we can compare the methods visually by observing the deviation of predicted values from actual values in each corresponding plots. By comparing the plots under Section 6.2.2 with the corresponding plots under Sections 6.2.1 & 6.2.3, we can seek answer to one of the research questions that states: **How the performance of GPs is compared with the state-of-the-art models like NNs and LLR models?**

6.2.1 Neural Networks

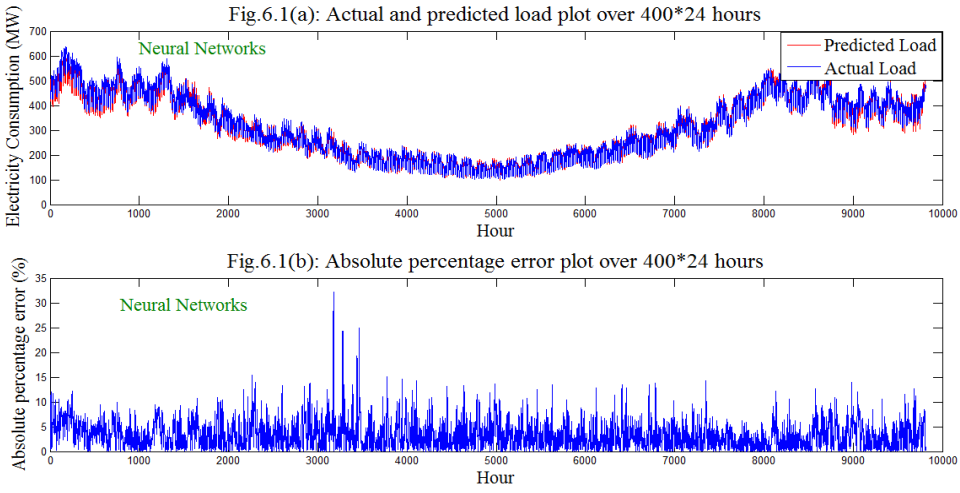

Fig.6.1 Neural network plot over 9814 test points.

Fig. 6.1 (a) shows plot of actual and predicted EC. Fig. 6.1 (b) shows the discrepancy between the actual values and predicted values. The performance of the NN can be evaluated by observing how close the predicted values (red curve) are to the actual values (blue curve). The discrepancy is measured by absolute percentage error, from which we can see the distribution of the errors over the whole test period. We can also see how the electricity consumption (EC) is distributed throughout the year. The prediction starts from January and we can see that the lowest consumption occurs at about the midpoint of the curve which represents summer time electricity consumption. In addition to the seasonal variation, we can also observe some local variations which may represent daily EC trend but not very clear in this figure. The rest of the plots will let us closely observe the trends over a shorter period like 24 hours of day, 7 days of week, or days of year, etc.

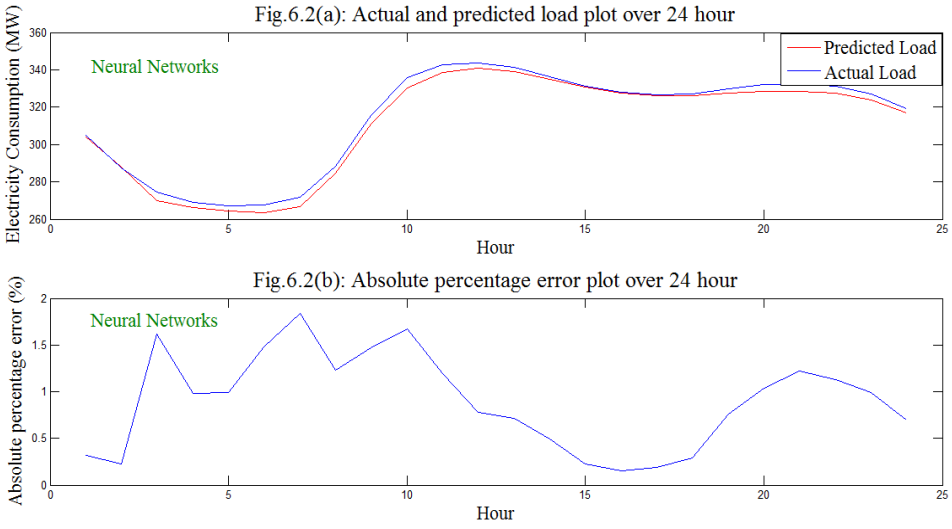

Fig.6.2 Neural network plot over 24 hours of day

The mean predicted and actual values at hour k are given by:

$$\overline{EC_k} = \frac{1}{N}\sum_{i=1}^{N} EC_k\,, \quad for\ k = 1,2,\dots,24\ and\ N\ number\ of\ test\ days.$$

Fig. 6.2 (a) shows plot of predicted and actual EC vs. 24 hours of day. Fig. 6.2 (b) shows the discrepancy of predicted EC from actual EC. Here, we can see the EC variations throughout 24 hours of day. For instance, minimum consumption at around 5AM and peak at about 11AM. By observing the variations of the absolute percentage error and the closeness of the reed curve to the blue curve, we can seek answer to one of the research questions: **how the NN respond to changes in hours of a day**.

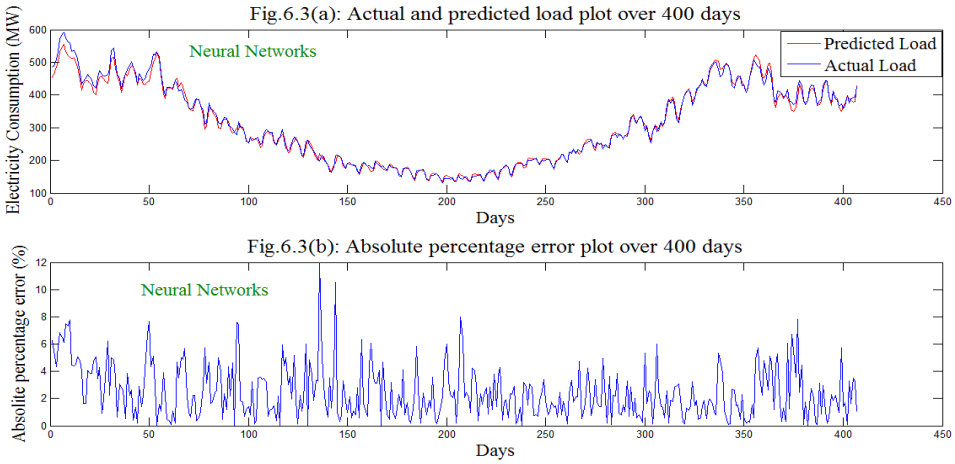

Fig.6.3 Neural network plot over 400 days

The average predicted and actual values at day *d* are given by:

$$\overline{EC_d} = \frac{1}{24}\sum_{i=1}^{24} EC_d\,, \quad for\ d = 1,2,\ldots,400$$

Fig. 6.3 (a) shows plot of predicted and actual EC. Fig. 6.3 (b) shows the deviation of predicted EC from actual EC. By observing the error plot and the closeness of the red curve to the blue curve throughout days of year, we can answer one of the research questions: **how the NNs respond to variations in days of a year**?

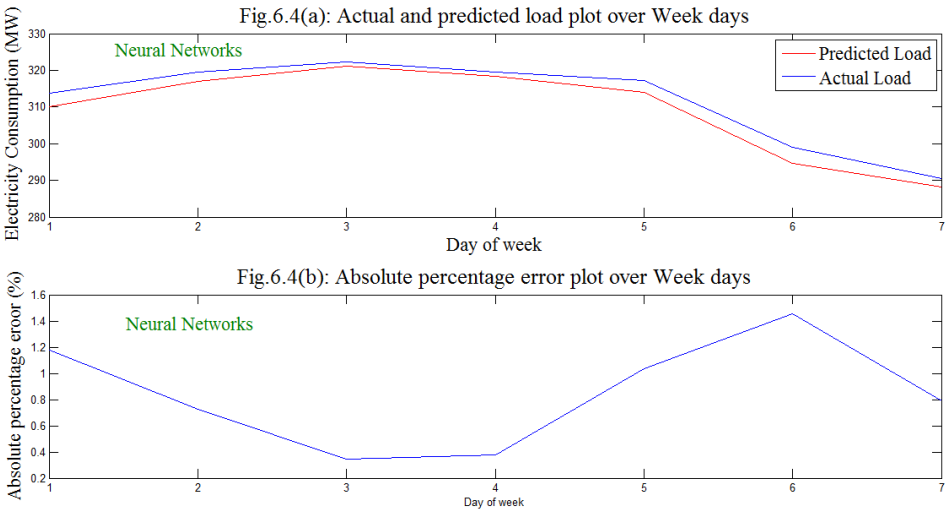

Fig.6.4: Neural network plot over 7 days of a week

The average predicted and actual values at day of week k are given by:

$$\overline{EC_k} = \frac{1}{N}\sum_{i=1}^{N} EC_k\,, \quad for\ k = 1,2,\ldots,7$$

$$here\ 1\ means\ Monday, 2\ is\ for\ Tuesday, etc$$

$$and\ N = \{24 \times number\ of\ weeks\ considered\ under\ the\ test\}.$$

Fig. 6.4 (a) shows plot of predicted and actual EC. Fig. 6.4 (b) shows the discrepancy of predicted EC from actual EC. Here, we can see the variation of EC in relation to days of week. For instance, the maximum EC is observed on Wednesday (day 3 in the figure) and minimum on Sunday (day 7). By observing the error plot variation from Monday to Sunday, we can answer one of the research questions: **how the NN perform under different days of week?**

Fig. 6.5(a) shows plot of predicted and actual EC. Fig. 6.5(b) shows the discrepancy of predicted EC from actual EC. Here, we can see the distribution of errors from the first week in January 2010 to the second week in February 2011.

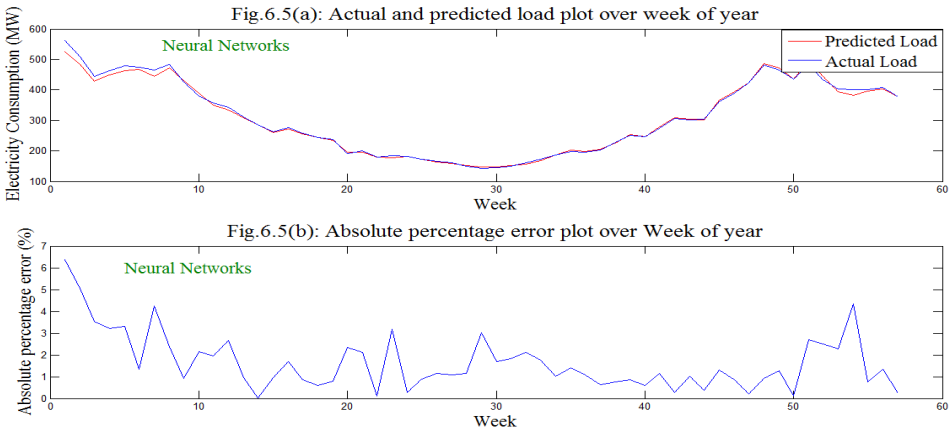

Fig.6.5: Neural network plot over 57 weeks

The average predicted and actual values at week *k* are given by:

$$\overline{EC_k} = \frac{1}{N}\sum_{i=1}^{N} EC_k\,, \quad for\ k = 1,2,\dots,57\ and\ N = 7 \times 24.$$

6.2.2 Gaussian Processes

The plots in this section are one-to-one with the plots in Section 6.2.1 & 6.2.3, and we can compare the models by observing the corresponding plots in other sections.

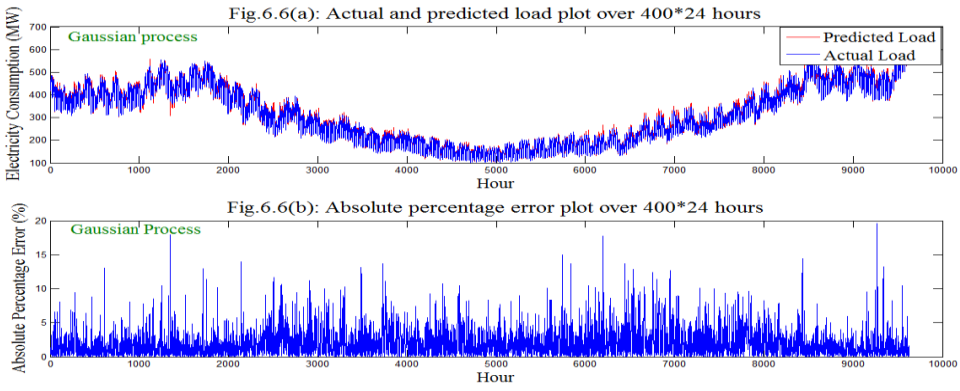

Fig.6.6: Gaussian process plots over 9624 test points.

Fig. 6.6 (a) shows plot of predicted and actual EC. Fig. 6.6 (b) shows the deviation of predicted EC from actual EC. Here also we can see seasonal variation of electricity consumption, and the distribution of error throughout the test period. These plots can be compared with plots in Fig. 6.1 & 6.11, to evaluate the performance of GPs against NNs and LLR models.

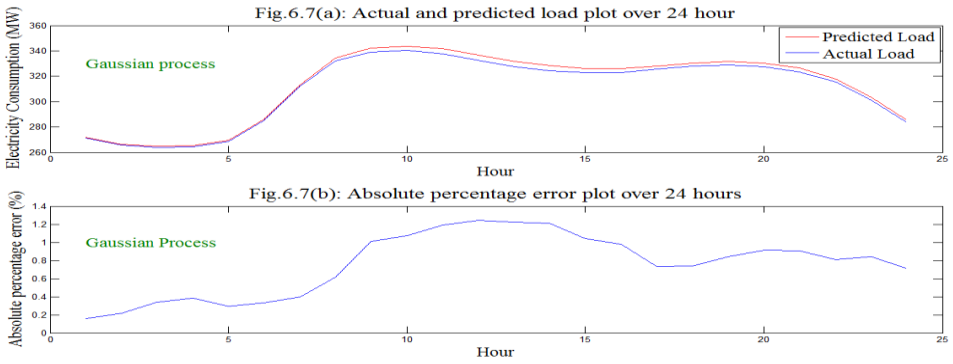

Fig.6.7: Gaussian process plots over 24 hours of day.

Fig. 6.7 (a) shows plot of predicted and actual EC. Fig. 6.7 (b) shows the magnitude of deviation of predicted EC from actual EC. We can also observe the distribution of this deviation with respect to 24 hours of day. These plots can be compared with plots in Fig. 6.2 & Fig. 6.12, to evaluate the performance of GPs against NNs and LLR models based on the magnitude and distributions of errors over 24 hours of day.

Fig. 6.8 (a) shows plot of predicted and actual EC. Fig. 6.8 (b) shows the magnitude of deviation of predicted EC from actual EC. Here, we can see the variation in electricity consumption throughout days of year. These plots can be compared with plots in Fig. 6.3 & Fig.6.13, to evaluate the performance of GPs against NNs and LLR models based on the magnitude and distribution of errors over seasons of year.

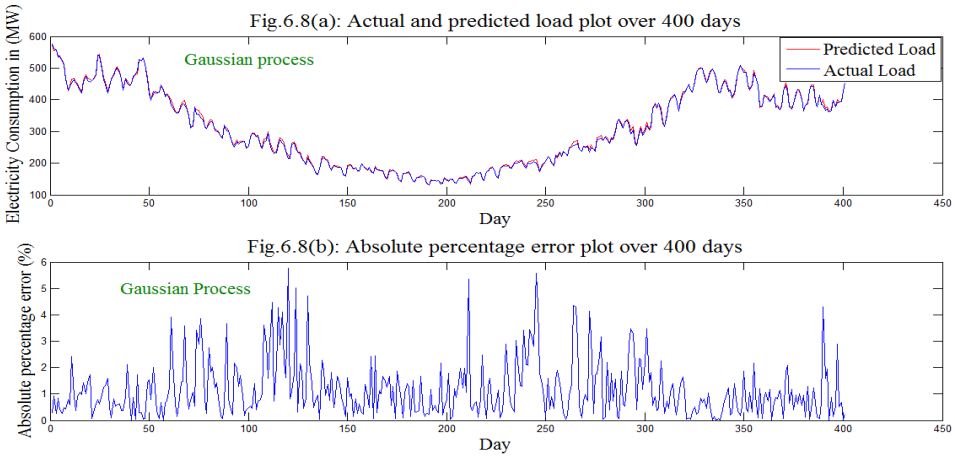

Fig.6.8: Gaussian process plots over 400days.

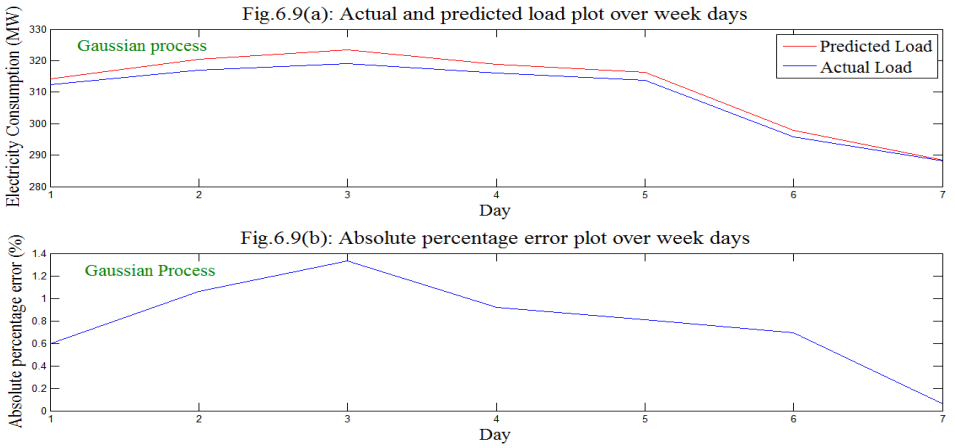

Fig.6.9: Gaussian process plots over 7 days of week.

Fig. 6.9 (a) shows plot of predicted and actual electricity consumption. Fig. 6.9 (b) shows the deviation of predicted EC from actual EC. Here we can see the variation in electricity consumption and distribution of errors over days of week (i.e., from Monday to Sunday). These plots can be compared with plots in Fig. 6.4 & Fig.6.14, to evaluate the performance of GPs against NNs and LLR models based on the magnitude and distribution of errors over days of week.

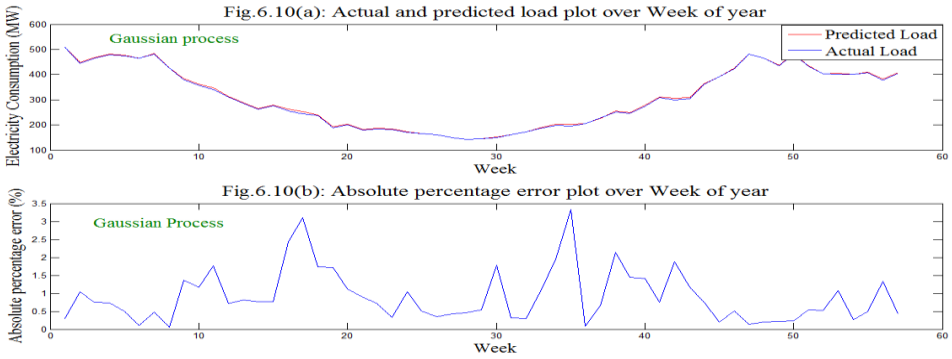

Fig.6.10: Gaussian process plots over 57 weeks

Fig. 6.10 (a) shows plot of predicted and actual EC. Fig. 6.10 (b) shows the Absolute percentage error. These plots can be compared with plots in Fig. 6.5 & Fig.6.15, to evaluate the performance of GPs against NNs and LLR models based on the magnitude and distribution of errors over weeks of year.

6.2.3 Local Linear Regression Model

This Section presents plots that help us to evaluate the performance of LLR model. We can also compare with the corresponding plots in Sections 6.2.1 & 6.2.2.

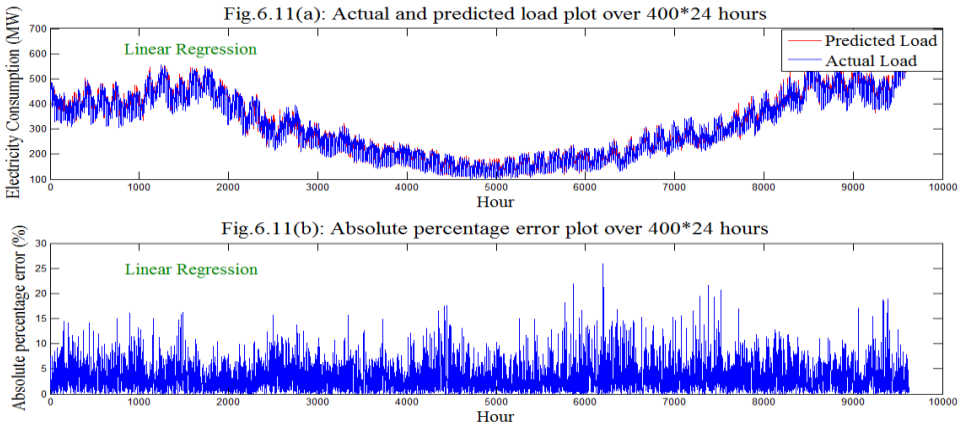

Fig.6.11: Local linear regression plots over 9624 test points.

Fig. 6.11 (a) shows plot of predicted and actual EC. Fig. 6.11 (b) shows the deviation of predicted EC from actual EC. These plots visualize the distribution of electricity consumption and prediction errors throughout the test period for the LLR model.

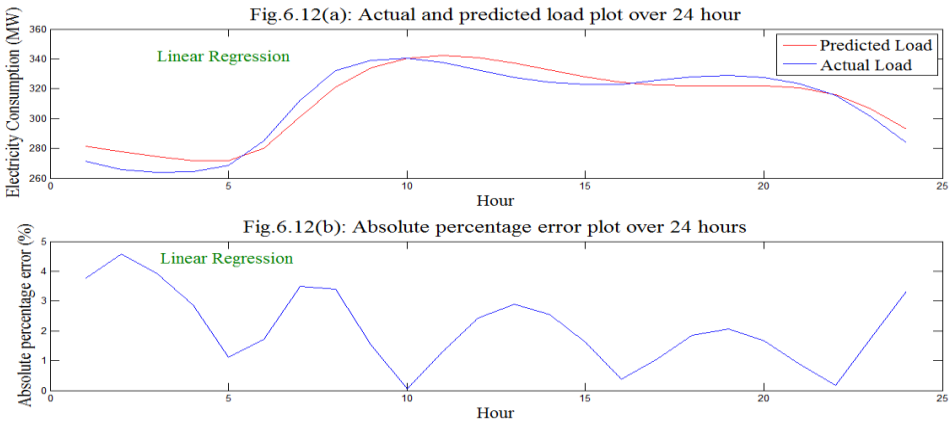

Fig.6.12: Local linear regression plots over 24 hours of day.

Fig. 6.12 (a) shows plot of predicted and actual electricity consumption. Fig. 6.12 (b) shows the deviation of predicted EC from actual EC. These plots visualize the distribution of electricity consumption and prediction errors over 24 hours of day for the local linear regression model.

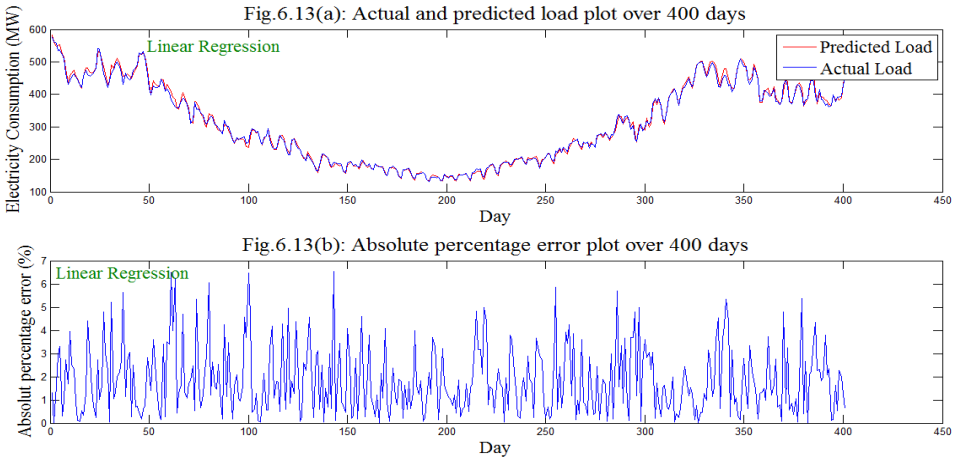

Fig.6.13: Local linear regression plots over 400 days.

Fig. 6.13 (a) shows plot of predicted and actual EC. Fig. 6.13 (b) shows the deviation of predicted EC from actual EC. These plots visualize the distribution of electricity consumption and prediction errors days of year for the local linear regression model.

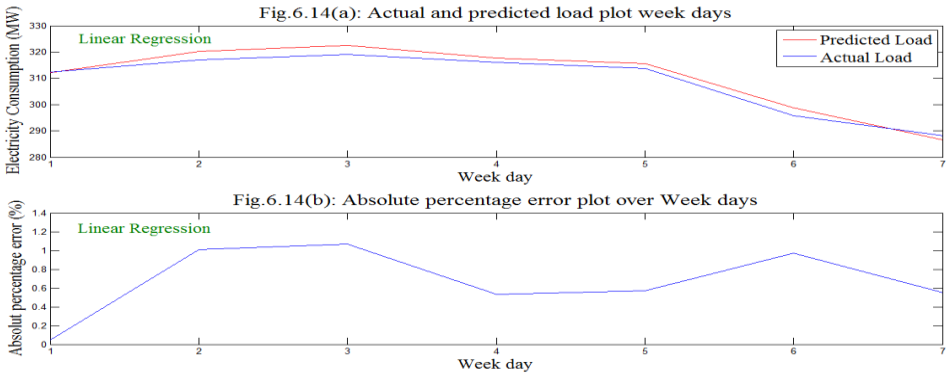

Fig.6.14: Local linear regression plots over 7 days of week.

Fig. 6.14 (a) shows plot of predicted and actual EC. Fig. 6.14 (b) shows the deviation of predicted EC from actual EC. These plots visualize the distribution of electricity consumption and prediction errors over days of week for the LLR model.

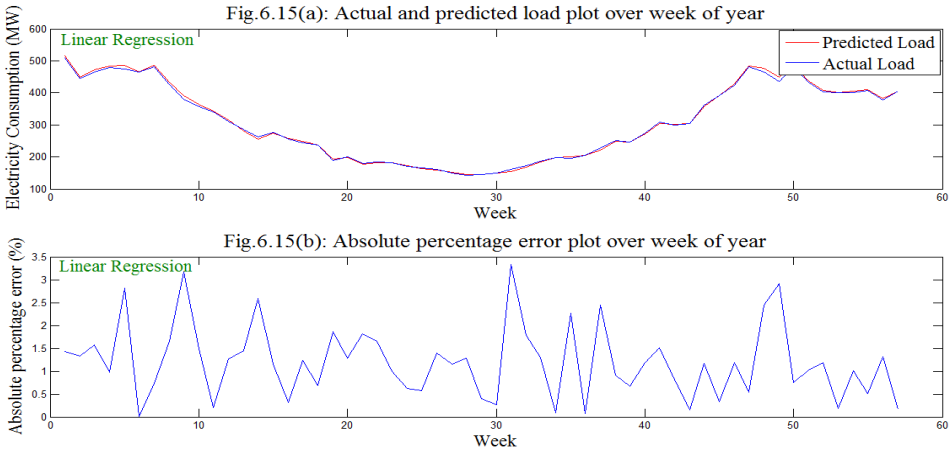

Fig.6.15: Local linear regression plots over 57 weeks.

Fig. 6.15 (a) shows plot of predicted and actual EC. Fig. 6.15 (b) shows the deviation of predicted EC from actual EC. These plots visualize the distribution of electricity consumption and prediction errors over 57 consecutive weeks for the local linear regression model.

6.3 Effect of Forecasted Temperature

Electricity consumption is highly influenced by temperature. When we consider test input to predict EC 24 hours ahead, we don't know the actual temperature in that period and we should take the forecasted temperature as one of the input variables. Thus, it is important to analyze additional discrepancy caused by the error of temperature predictor. We can analyze this effect for GPs by comparing the plots in this section with the corresponding plots in Section 6.2.2. The effect of discrepancies in temperature predictor for NNs and LLR are also summarized in Table 7.2.

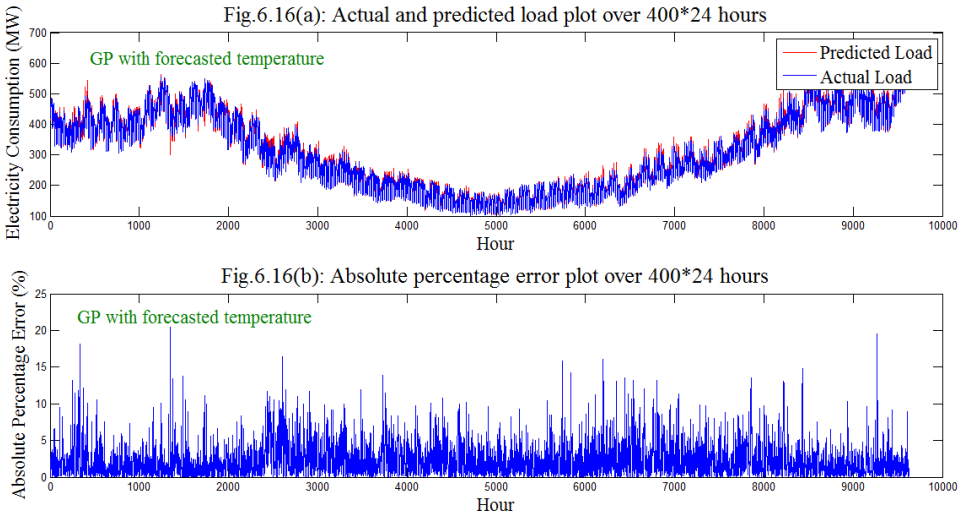

Fig.6.16: Gaussian process plots over 9624 test points, with forecasted temperature.

Fig. 6.16 (a) shows plot of predicted and actual EC. Fig. 6.16 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.6 to evaluate the effect of errors in forecasted temperature on the performance of GPs over the whole test period.

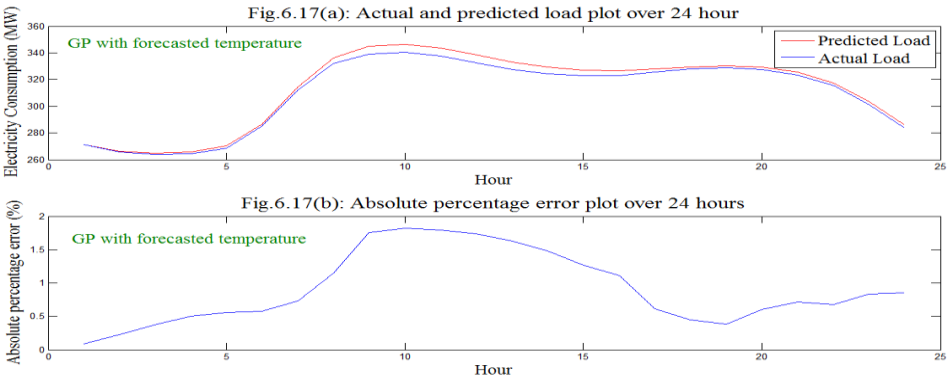

Fig.6.17: Gaussian process plots over 24 hours, with forecasted temperature.

Fig. 6.17 (a) shows plot of predicted and actual EC. Fig. 6.17 (b) shows the deviation of predicted EC from actual EC. By comparing these plots with the corresponding plots in Fig. 6.7, we can evaluate the effect of errors in forecasted temperature on the performance of GPs over 24 hours of day.

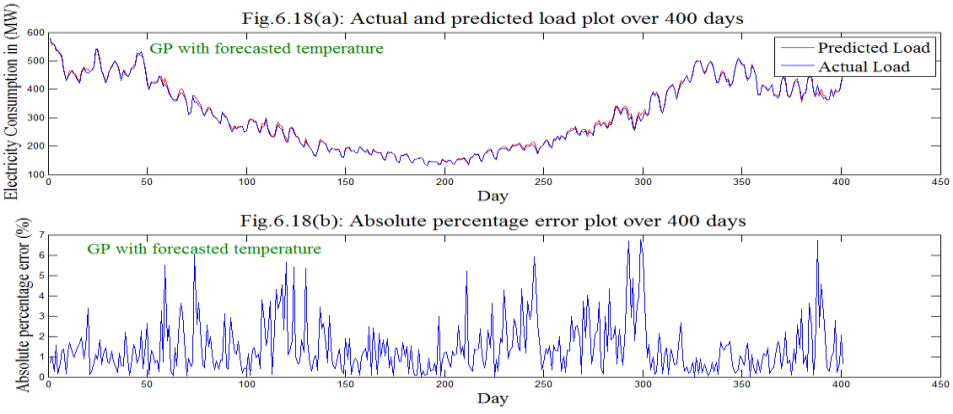

Fig.6.18: Gaussian process plots over 400 days, with forecasted temperature.

Fig. 6.18 (a) shows plot of predicted and actual EC. Fig. 6.18 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.8 to evaluate the effect of errors in forecasted temperature on the performance of GPs over days of year.

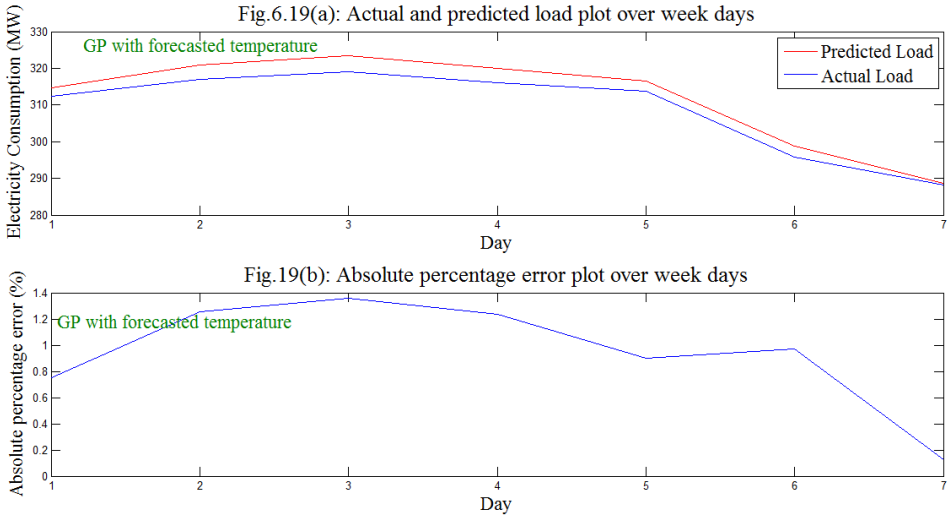

Fig.6.19: Gaussian process plots over 7 days of week, with forecasted temperature.

Fig. 6.19 (a) shows plot of predicted and actual EC. Fig. 6.19 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.9 to evaluate the effect of errors in forecasted temperature on the performance of GPs over 7 days of week.

Fig. 6.20 (a) shows plot of predicted and actual EC. Fig. 6.20 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.10 to evaluate the effect of errors in forecasted temperature on the performance of GPs over weeks of year.

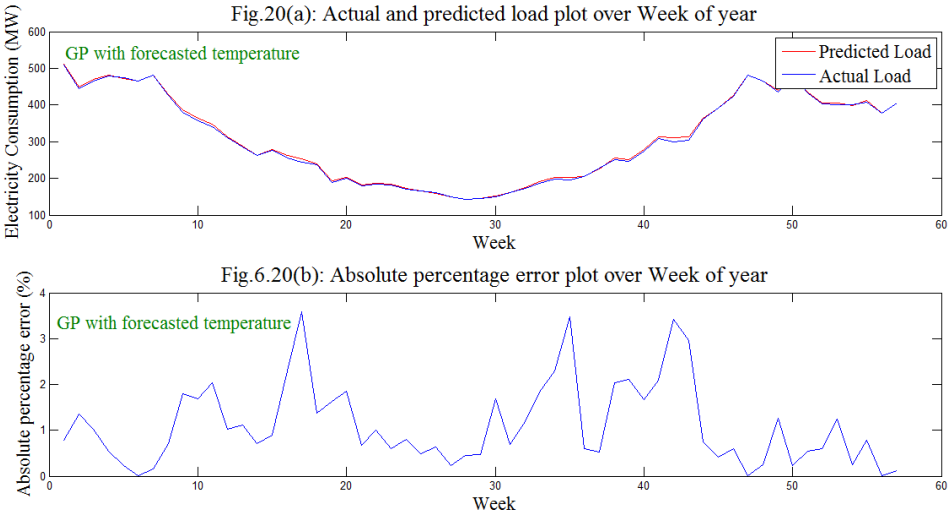

Fig.6.20: Gaussian process plots over 57 weeks, with forecasted temperature.

6.4 FEATURE VECTORS FOR NNS

In order to include the previous hour EC in the feature vector in NNs, as we did in GPs and LLR models, we used a two-step prediction:

- In the first step, we predict using other features (i.e., excluding the previous hour EC).
- We train the network by including the previous hour EC, and used the predicted values from step 1, with one hour lag (t-1), as one of the test inputs during prediction.

The performance of this approach can be evaluated against the normal (one-step) approach by comparing plots in this section with the corresponding plots in Section 6.2.1. Moreover, the mean absolute percentage errors (MAPE) for both approaches are given in Table 7.2.

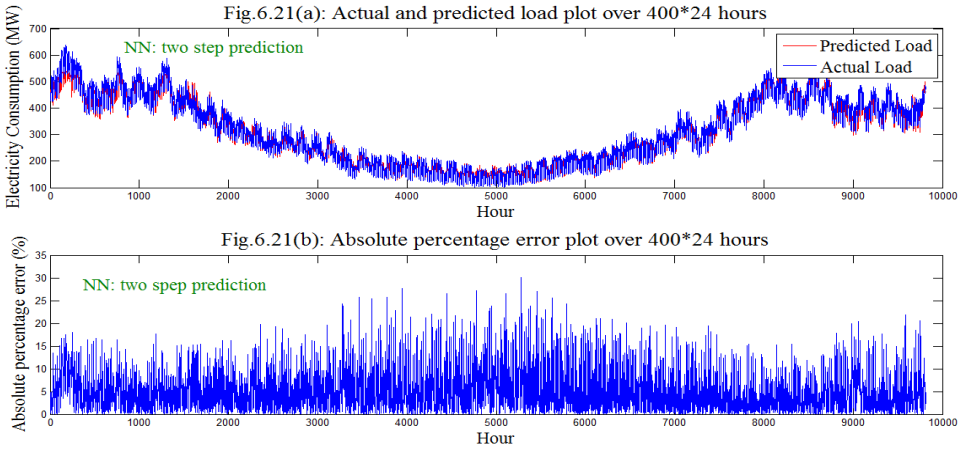

Fig.6.21: Neural networks: two-step prediction plot over 9814 test points.

Fig. 6.21 (a) shows plot of predicted and actual EC. Fig. 6.21 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.1 to evaluate the performance of 'two-step' NN with the normal (one-step) NN over the whole prediction period.

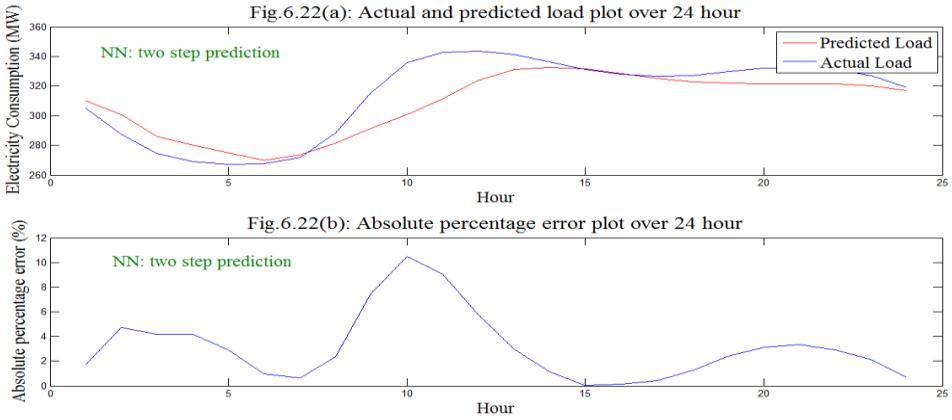

Fig.6.22: Neural networks: two-step prediction plot 24 hours of day.

Fig. 6.22 (a) shows plot of predicted and actual EC. Fig. 6.22 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots

in Fig. 6.2 to evaluate the performance of 'two-step' NN with the normal (one-step) NN over 24 hours of day.

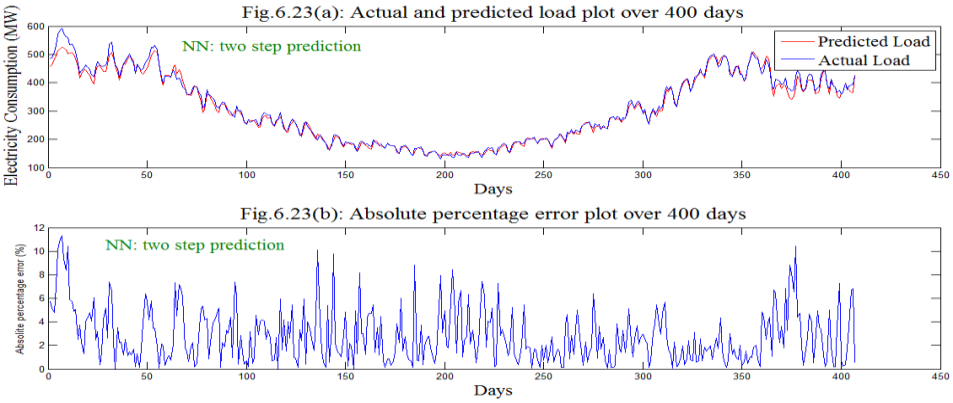

Fig.6.23: Neural networks: two-step prediction plot over 400 days.

Fig. 6.23 (a) shows plot of predicted and actual EC. Fig. 6.23 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.3 to evaluate the performance of 'two-step' NN with the normal (one-step) NN over days of year.

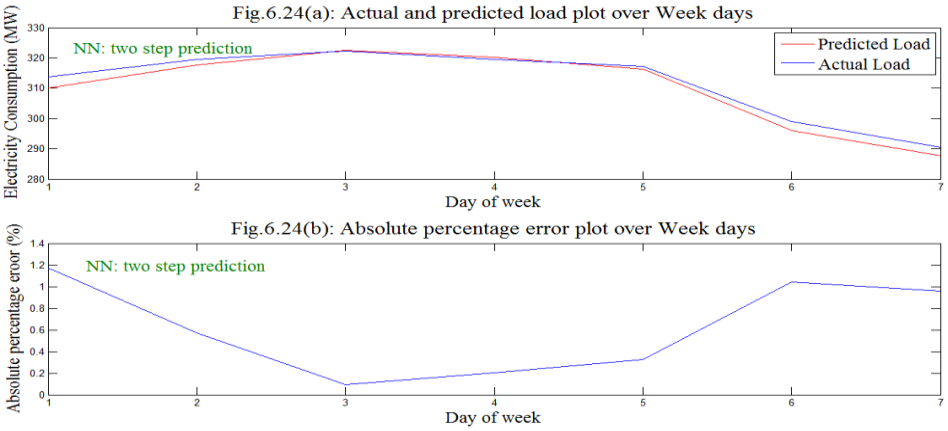

Fig.6.24: Neural networks: two-step prediction plot over 7 days of week.

Fig. 6.24 (a) shows plot of predicted and actual EC. Fig. 6.24 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.4 to evaluate the performance of 'two-step' NN with the normal (one-step) NN over the 7 days of week.

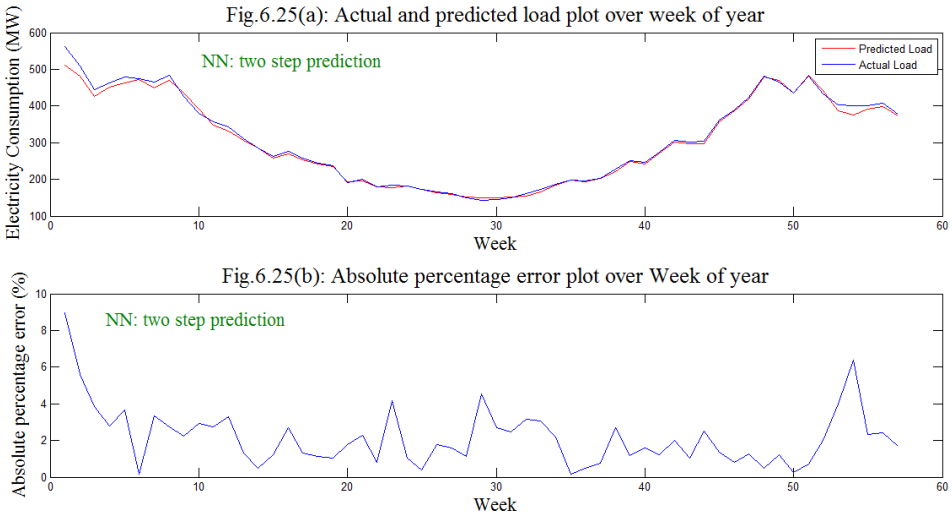

Fig.6.25: Neural networks: two-step prediction plot over 57 weeks.

Fig. 6.25 (a) shows plot of predicted and actual EC. Fig. 6.25 (b) shows the deviation of predicted EC from actual EC. We can compare these plots with the corresponding plots in Fig. 6.5 to evaluate the performance of 'two-step' NN with the normal (one-step) NN over 57 weeks.

6.5 PREDICTION ERROR FREQUENCY DISTRIBUTION

Fig.6.26: Frequency distribution of errors

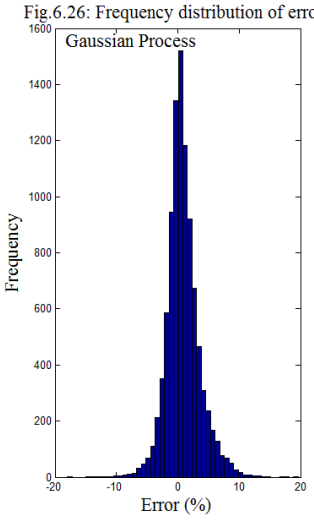

Fig.6.27: Frequency distribution of errors

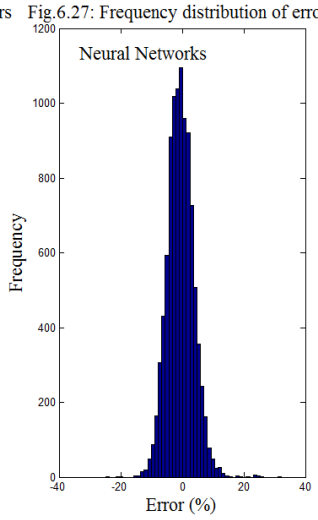

Fig.6.28: Frequency distribution of errors

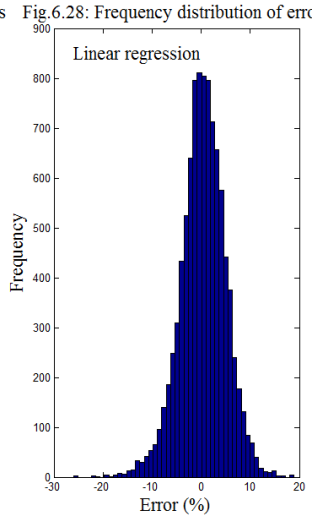

Fig.6.26 –6.28: Prediction error frequency distribution for GPs, NNs and LR.

Fig. 6.26–6.28 shows that the mean of the residuals in each predictor is approximately zero. It also shows that the most frequent error for the predictors is approximately zero (since the histogram bar at 0 is the highest). This shows how good the prediction curve for each model fits the test data.

7 Summary of Results and Discussion

In this chapter we will present tabular summary of results and discuss them in relation to the research questions.

- **How can we select the best feature vectors that optimize the performance of the predictors (GPs, NNs and LLR)**

  Our approach to feature selection is discussed in Section 2.1.3; in this section we will discuss how the selected features affect the performance of the predictors. Table 7.1 summarizes the influence of each element of the feature vectors on the performance of the GP[3] model. This experiment was done by starting with one of the most influential features and adding more features one after the other to see their influence on the overall performance. This step-by-step addition of features refers only to the input of GP, in the kNN search we used all of the features since similarity measure based on few features is inaccurate and it will be difficult to distinguish the influence of each feature. Let us define the subsets of the feature vectors as follows:

$$A_1 = \{previous\ hour\ EC\}\ ;\ A_2 = \{A_1, temperature\};$$

$$A_3 = \{A_2, hour\ of\ day\}$$

$$A_4 = \{A_3, day\ of\ week\};\ \ A_5 = \{A_4, day\ of\ year\};\ \ A_6 = \{A_5, cloud\ cover\};$$

$$A_7 = \{A_6, previous\ week\ power\}$$

  As can be seen from Table 7.1, each feature contributes to improving the prediction accuracy and to predict values that are more correlated with the actual

[3] The same procedure has been followed to select features for NNs and LLR models. Since most of the features are common to the three predictors, we didn't include the procedures for the other two in order to avoid repetition.

values. For example by using only the information about previous hour electricity consumption, the prediction error MAPE is 6.23%, when we add the temperature information the MAPE go down to 5.39%, and finally when we use all of the 7 features the MAPE becomes 1.55%. Similarly, the correlation between the predicted value and the actual values increase from 0.69 to 0.98 as we add more features. This shows that each selected feature has significant contribution to the performance of the predictor.

Table 7.1[4]: The influence of each feature on the overall performance of GPs

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
MAPE	6.23	5.39	3.60	2.97	2.52	2.14	1.55
CC[5]	0.69	0.80	0.92	0.96	0.96	0.97	0.98

- **How is the performance of GPs compared with the state-of-the-art models (NNs and LLR models)**

  Table 7.2 summarizes the prediction error MAPE for GPs, NNs and LLR models. The minimum MAPE is 2.03% which represents prediction error of GPs when actual (measured) temperature is used as one of its features. The second minimum MAPE (2.33%) is also for the GPs when the forecasted temperature is used instead of the measured temperature. This shows that the Gaussian processes outperform the NNs and the LLR models. The same table shows that the third and fourth minimum MAPE are 3.35% and 3.46% both for NNs with measured and

[4] The table represents test result of prediction 24 hours ahead over 20 days.
[5] Represents the correlation between actual and predicted electricity consumption (EC)

forecasted temperature respectively. Thus, neural networks outperform the local linear regression model on predicting electricity consumption. A quite strange observation is 5.1% MAPE for two-step prediction using NNs. This is clearly less accurate than both the LLR and 'one-step' NNs, then why 'two-step' prediction? That is why we didn't consider this model and thus left the previous hour EC out of the feature vectors for NNs.

Table 7.2: Mean absolute percentage error (MAPE) for GPs, NNs and LR

	Gaussian processes		Neural Networks			Linear Regression	
Test scenarios			With actual temperature		With predicted temperature		
	With actual temperature	With predicted temperature	One step prediction	Two step prediction		With actual temperature	With predicted temperature
MAPE (%)	2.03	2.33	3.35	5.10	3.46	3.56	3.80

The two-step approach: The 'two-step' prediction for the NN is given with an intension of including the previous hour EC as one of the features. We want to predict EC 24 hours ahead, which means that we don't know the previous hour EC except for the first prediction hour (i.e., at t+1). To solve this problem, in GP and LLR, we used an iterative approach in which we predict EC at t+1, then we use this predicted value as previous hour test input for predicting EC at t+2, and continuing the iteration until we predict EC at t+24. However, training in NNs is

quit complex and this iterative approach will be inconvenient to the users and it is even more impractical to test at about 9600 test points (for all seasons of a year).

In two-step prediction, we use the predicted EC in one-step-prediction as test input in place of the unknown previous hour EC. This approach enables us to use the same feature vectors for all predictors, but it is inaccurate. Our observation is that, the feedback error during the training phase (in the second-step) falls very fast since the previous hour EC provides strong information about the predicted EC (see table 2.1: correlation coefficients). However when we use the predicted values (which is deviated from actual value by MAPE=3.35%, shown in Table 7.2), the error becomes even higher (MAPE=5.1%, shown in the same table). From this observation, we can infer that the network was over trained (over fitted), which means that a strongly informative input provided during the training phase are replaced by relatively less informative test inputs during prediction. Neural networks are strong in recognizing patterns; however this feature comes at the cost of over fitting particularly in large neural nets. Thus, instead of training the network every time we predict it is more convenient to train the NN with large data, validate and save the trained network and use it for prediction when needed. For example, in this thesis we trained the NN using data from the first two years and predict EC in the third year. As can be seen from Table 7.3, the errors are fairly distributed over months of the year.

The effect of error in temperature forecast: It is the measured (actual) temperature that is directly related to the electricity consumption, however, we are

only provided with the forecasted temperature during the prediction period (24 hours ahead). Therefore, practical use of the predictors is dependent on their performance when used with the predicted temperature. Except for a slight increment in errors (MAPE), which is expected because of the errors in temperature predictor, the predictions of GP, NNs and LLR, with forecasted temperature are close to their actual temperature counterparts.

- **How the predictors perform under scenarios like different seasons of a year, different days of a week, and different hours of a day?**

  To answer this research question, we will discuss each scenario one after the other.

  Performance throughout a year: The overall MAPE is a strong measure of prediction accuracy. However, we need more detailed observation on error distributions against changing scenarios like different seasons of year. Table 7.3 together with Figures 6.3, 6.8 & 6.13, shows that the errors are fairly distributed throughout the year. This indicates that the predictors learn the relationship between seasons of year and electricity consumption, and thus responds well to the seasonal variations.

Table 7.3: Mean absolute error (MAPE) of each predictor with respect to months of year.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
MAPE_GP	1.43	1.38	2.30	2.30	2.90	2.10	1.84	2.16	2.52	2.54	1.86	1.49
MAPE_NN	3.07	3.28	3.20	3.71	4.39	3.34	3.56	3.09	2.86	2.89	2.32	2.87
MAPE_LR	3.87	3.26	4.30	3.90	4.23	3.46	3.07	3.40	3.12	3.56	3.20	3.31

Performance throughout the 7 days of week: Days of week also have high influence on electricity consumption; particularly EC variation between the workdays and the weekends is significant. Thus, it is important to test the distribution of errors (MAPE) throughout days of week in order to confirm whether we had included enough features that make it possible to learn those variations. Table 7.4 together with Figures 6.4, 6.9 & 6.14; show that the errors are fairly distributed throughout days of week for all predictors. This means that the predictors respond well to EC variations throughout days of week.

Table 7.4: Mean absolute error (MAPE) of each predictor with respect to days of week.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
MAPE_GP	1.92	1.94	2.17	1.97	2.39	2.06	1.79
MAPE_NN	3.98	3.40	2.90	3.05	3.13	3.64	3.17
MAPE_LR	4.27	3.41	3.45	3.35	3.51	3.48	3.50

Performance throughout hours of day: Electricity variations throughout a day are also another feature that the predictors need to learn. Table 7.5 together with Figures 6.2, 6.7 & 6.12 show that the error (MAPE) is distributed fairly throughout 24 hours of day for each predictor.

Table 7.5: Mean absolute error (MAPE) of each predictor with respect to 24 hours of day.

Hour	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00
MAPE_GP	0.92	1.14	1.18	1.34	1.54	2.47	3.25	3.09
MAPE_NN	2.81	3.22	3.47	3.48	3.51	3.51	3.56	3.83
MAPE_LR	3.34	4.22	4.11	3.55	3.12	4.24	5.41	4.80
Hour	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00
MAPE_GP	2.68	2.44	2.38	2.26	2.20	2.08	2.07	2.08
MAPE_NN	4.14	3.90	3.28	3.00	2.99	3.06	3.14	3.14
MAPE_LR	3.71	3.43	3.54	3.73	3.82	3.66	3.22	2.92

Hour	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00
MAPE_GP	2.05	2.00	1.92	1.88	1.91	1.88	1.87	2.16
MAPE_NN	3.29	3.32	3.41	3.57	3.50	3.31	3.02	2.85
MAPE_LR	3.05	3.19	3.08	2.86	2.74	2.65	3.02	4.11

- **Is it possible to overcome the limited data handling capabilities of GPs, and let the GPs learn the most relevant information stored in a large dataset?**

  We used the kNN similarity search to overcome the computational requirement of the GPs. The resulting GP model can predict electricity consumption with better accuracy when compared with the NNs and LLR models. Thus, the answer to this research question is affirmative and the Gaussian process approach improves the prediction accuracy over the state-of-the-art models.

8 Conclusions

In this thesis we have investigated how Gaussian processes can be used for short-term forecasting of electricity consumption. We started by analyzing the data in order to identify features that influence electricity consumption. Data analysis techniques like EDA and correlation coefficients have been used as steppingstone for selecting relevant features and excluding the redundant features.

To cope with the computational requirement of the GPs which grows fast with the size of the dataset, we have implemented the kNN similarity search. The kNN similarity search explores the dataset and selects k closest points to a given query point. By using the output of the kNN similarity search as a training input for GPs the computational requirements of GPs have been reduced from $O(n^3)$ to $O(k^3)$, where $k \ll n$. The kNN similarity search is an intuitive approach that could provide more dimensions to similar researches on addressing the problem of handling large datasets, particularly in GPs. Moreover, its implementation doesn't involve any change to the full GP regression model and thus it can be used irrespective of the type of kernel function used with the GP. It is also used to improve the classical linear regression model by applying regression on local data points closest to the test point i.e., local linear regression.

We have studied different state-of-the-art prediction models and compared them with GPs in order to reinforce our argument for the proposed GP model. The empirical analyses of the results from this study show that the GPs outperform both the NNs and LLR models. This is an important result since even a small improvement in prediction accuracy has significant economic values in the energy industry.

The practical applicability of the predictors has been tested by considering different scenarios that influence electricity consumption. Each model has been tested against variations in seasons of year, days of week and hours of day using real-life dataset from Eidsiva Energy. The result shows that the predictors respond well to those variations. This indicates that information regarding those scenarios is included in the feature vector and the predictors can generalize the information to unseen future.

Further work: One possible continuation of this work can be investigating other similarity measures like dynamic time warping (DTW) for the kNN similarity search. The Euclidean distance is a widely used metric in the area; however more intuitive and accurate measure of similarity could potentially improve the prediction accuracy of the GPs and LLR models. The other possibility is to apply this approach to similar problems in other areas such as, weather forecast or financial market forecasting.

In this work the EC forecast window was limited to a day ahead due to unavailability of weather forecast for longer period. If enough weather data can be found, it may be feasible to do both weather forecasting and EC forecasting with possibility of extending the prediction period (to a week ahead, for example).

Appendix A: List of Symbols and Notations

GP(s)	Gaussian Process(es)
NN/ANN	Neural Networks/Artificial Neural Networks
KNN	K-Nearest Neighbours
LR	Linear regression
LLR	Local linear regression
EC	Electricity consumption
STFEC	Short-term forecasting of electricity consumption
Ɗ	data set: $Ɗ = \{(x_i, y_i) \mid i = 1, \ldots, n\}$
E[ ]	expectation
X (x)	training input-matrix (vector)
X* (x*)	test input-matrix (vector)
y	training target (observation)
y*	test target (predicted value)
f*	GP posterior (prediction)
$\bar{\mathbf{f}}_*$	GP posterior (predictiv) mean
MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
Predictor	Regression model such as GPs, NNs, LR
RPROP	Resilient PROPagation
RBF	Radial Basis Function
SARI	Seasonal Auto-Regressive Integral

ARIMA	autoregressive integreted moving average
RTE	Electricity company in France
MLP	multi-layered perceptron
RBFN	eadial basis function network
SVR/SVC	Support vector regression/support vector clustering
FV	feature vector
EDA	exploratory data analysis
CC	Correlation coefficient
KD tree	k-dimensional tree
LSM	least square method
SO	system operator
DTW	dynamic time warping

References

[1] C.E. Rasmussen & C.K.I. Williams 2006, Gaussian Processes for Machine Learning, the MIT press

[2] Jarno Vanhatalo, Jaakko Riihmäki, Jouni Hartikainen and Aki Vehtari 2011, Bayesian Modeling with Gaussian Processes using the MATLAB Toolbox GP-Stuff

[3] Jeff Heaton 2008, Introduction to Neural Networks for C#, Second Edition, Heaton Research Inc

[4] Joaquin Quiñonero_Candela, C.E. Rasmussen, C.K.I. Williams 2007, Approximation Methods for Gaussian Process Regression, Microsoft Research, Technical Report MSR-TR-2007-124

[5] M. Làzaro-Gredilla, J. Quiñonero-Candela, C.E. Rasmussen, A. R. Figueiras-Vidal, Sparse Spectrum Gaussian Process Regression, Journal of Machine learning Research 11 (2010).

[6] Mark Hudson Beale, Martin T. Hagan & Howard B. Demuth, Neural Network User's Guide, 1992-2011 The Math Works, Inc.

[7] Douglas J. Leith et al, Gaussian Process Prior Models for Electrical Load Forecasting, 8^{th} International Conference on Probabilistic Methods Applied to Power System, Iowa State University, September 12-16, 2004

[8] Trevor Hastie, Robert Tibshirani and Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Second Edition, Springer Science+ Business Media LLC 2009.

[9] Gouri K.Bhattacharyya & Richard A.Johnson, Statistical Concepts and Methods, John Wiley & Sons, 1977

[10] Shawn Brown & Jack Snoeyink, GPU Nearest Neighbor Searches Using a Minimal kd-tree, University of North Carolina, can be found at web address: http://cs.unc.edu/~shawndb/

[11] Simon Heinzle, Gael Guennebaud, Mario Botsch, Markus Gross, A Hardware Processing Unit for Point Sets, The Eurographics Associations 2008, ETH Zurich.

[12] Philip Boyle, Gaussian Processes for Regression and Optimization, Victoria University of Wellington 2007

[13] Fawzi M. Al-Naima and Ali H. A1-Timemy, Resilent Back Propagation Algorithm for Bearst Biopsy Classification Based on Artificial Neural Networks, From web address: http://cdn.intechopen.com/pdfs/8853/

[14] Alicia Troncoso Lora et al., Time Series Prediction: Application to the short-term energy demand, ©Springer-Verlag Berlin Heidelberg 2004.

[15] Wendy L. Martinez and Angel R. Martinez, Computational Statistics Handbook with MATLAB, Chapman & Hall/CRC 2002.

[16] http://en.wikipedia.org/wiki/Nonparametric_regression

[17] M. Ebden, Gaussian Process for Regression: A Quick Introduction, August 2008

[18] Sofiane Brahim_Belhouari and Jean-Marc Vesin, Bayessian Learnung using Gaussian Process for time series prediction, 2001 IEEE

[19] Louis Atallah et al, Gaussian Process Prediction for Cross Channel Consensus in Body Sensor Networks, 2008 IEEE

[20] Rasmussen C.E. and Williams C.K.I., Gaussian Process for Machine Learning, Cambridge: The MIT press, 2006

[21] Douglas J. Leith et al, Gaussian Process Prior Models for Electrical Load Forecasting, 8^{th} International Conference on Probabilistic Methods Applied to Power System, Iowa State University, September 12-16, 2004

[22] Hiroyuki Mori and Daiseke Kanaoka, GP-Based Temperature Forecasting for Electric Load Forecasting, 2009 IEEE

[23] Yacine Chakhckoukh, Patrick Panciatici and Lamine Mili, New robust method applied to short-term load forecasting, Paper for presentation at 2009 IEEE Bucharest Power Tech Conference, June 28th - July 2nd, Bucharest, Romania

[24] Keem Siah Yap et al, Short Term Load Forecasting Using a Hybrid Neural Network, First International Power and Energy Conference PECon 2006 November 28-29, 2006, Putrajaya, Malasia

[25] Zainab H. Osman et al, Neural Network Based Approach for Short-Term Load Forecasting, 2009 IEEE

[26] Edward B.Magrab et al, An engineer's guide to MATLAB, second edition 2005

[27] Ed Snelson, Gaussian process model for machine learning (tutorial), Gatsby computational Neuroscience unit, UCL, 26 October 2006.

[28] Leo Breiman, Statistical Modeling: The two cultures, statistical science, vol.16, No.3, 199-231

[29] NeuroAI, Intelligent system and Neural Networks, From website: http://www.learnartificialneuralnetworks.com/

[30] http://en.wikipedia.org/wiki/K-d_tree

[31] Mitchel T., Machine Learning, McGraw Hill 1997

[32] Rina Panigraphy, Nearest Neighbor Search using Kd-trees, December 2006. Can be found at: http://www.chrisbirke.com/whitepapers/2006 Nearest Neighbor Search using Kd-trees.pdf

[33] Andrew W. Moore, An introductory tutorial on kd-trees, Carnegie Mellom University. Which can be found at: http://www.autonlab.org/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf?branch=main&language=en.

[34] http://en.wikipedia.org/wiki/Least_squares

[35] http://en.wikipedia.org/wiki/Linear_regression

[36] M. Karagiannopoulos, D. Anyfantis, S.B. Kotsiantis, P. E. Pintelas, Feature Selection for Regression Problems, University of Patras Greece. Can be found at: http://www.math.upatras.gr/~dany/Downloads/hercma07.pdf

[37] Jeff Heaton, Introduction to Encog for C#, ©2010 Heaton Research, Inc

[38] Encog machine learning framework, ©2011 Heaton Research, Inc. http://www.heatonresearch.com/wiki/Training

[39] http://scikit-learn.org/dev/modules/neighbors.html

[40] Jerry Webb 2007, Ancillary Services Markets, found at web address: http://www.narucpartnerships.org/Documents/Ancillary_Services_and_Demand_response_eng_Jerry_Webb.pdf

[41] The scheduling and dispatch processes: A basic guide. Found at web address: http://www.eirgrid.com/media/Dispatch%20and%20Scheduling%20Processes%20-%20December%2009.pdf

[42] M.S.R. Murty, Basic grid operation. Web address: http://www.sari-energy.org/PageFiles/What_We_Do/activities/CEB_Power_Systems_Simulation_Training,_Colombo,_Sri_Lanka/Course_ppts/Lecture_02_PS_Opn.pdf