MULTI-VARIABLE MODEL OF A NEURAL NETWORK BASED WEATHER FORECASTER USING 2-STAGE FEATURE SELECTION

by

KUNAL VORA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2012

ACKNOWLEDGEMENTS

I hereby express thankfulness towards my advisor Prof. Micheal Manry. He did a great job of filling up an empty mind with knowledge by diligently working with me towards this thesis and research. I thank him for allowing me to research under him and for regularly updating my knowledge on the subject and in signal processing in general. I also believe that he probably offered the most useful classes that are now a part of my curriculum. I hope he will be satisfied with my work.

I also thank Prof. Subbarao and Prof. Wei Jen Lee for dedicating enough of their precious time for reviewing my thesis. I hope it meets their expectations.

I also express my hearty thanks and gratefulness to all my friends and colleagues, more importantly my family for supporting me throughout my academic and professional carrier.

April 12, 2012

ABSTRACT

MULTI-VARIABLE MODEL OF A NEURAL NETWORK BASED WEATHER FORECASTER

USING 2-STAGE FEATURE SELECTION

KUNAL VORA, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Prof. Micheal T Manry

This thesis proposes a novel approach for designing a neural network based forecaster that predicts more than one variable at a time. A second order two stage neural network training algorithm is used that employs orthogonal least square for training the output weights.

In order to reduce the size of the network and train the forecaster optimally it uses time-domain feature selection and KLT transform based feature selection. The forecaster works well and the feature selection reduces the number of required inputs on the order of 70 %.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Scope of neural networks and research

Neural networks have now become an important tool for non-linear system analysis, approximation, detection and control. Applications extend from non-linear control [55], [56], to target recognition [57], [58], to text processing and reading [59], [60], to remote sensing [61], [62], all briefly mentioned in [1]. Here we shall give a brief introduction on the important functions that neural networks are able to emulate within certain capabilities.

Pattern Association and Recognition - This problem is posed in the statistical sense by a (Ideal) Bayes classifier that can discriminate between the patterns belonging to different classes. The neural networks are capable to do the same either by using the polynomial basis formulation like a functional link network, or a multilayer perceptron which tunes the non-linear basis function during the training process. It is this capability of the neural networks that makes it the most valuable tool for applications like face recognition [65], [66], fingerprint recognition [63], [64], and speech processing [67], [68].

Function Approximation - Function approximation application of the neural network is the most widely used of its all application areas. This includes the estimation, prediction, inverse modeling or system Identification. Above three are basically input-output mapping where vector '**x**' is mapped to '**d**' by an invertible mapping '**Y**' as defined below. And '**y**' is the approximated function for 'Y' using a neural network, ϵ is the approximation error-

$$d=Y(\mathbf{x}),\ \mathbf{x}=Y^{-1}(d)$$
$$\|Y(\mathbf{x})-y(\mathbf{x})\|<\epsilon \text{ ,for all } \mathbf{x} \qquad (1)$$

Controller Design - The controller design of a non-linear plant is basically an inverse design problem. This architecture of the neural network basically uses the Reference signal and the feedback of the plant thereby generating the error signal. This error signal is the used to tune the parameter of the controller during the training process which most of the times is online type thereby generating a control input which is feed to the plant. This kind of learning is also subdivided into direct and indirect learning. Such networks are mostly recurrent networks.

This thesis falls in the category of the function approximation. The problem can be posed as an estimation or as a prediction. The major difference is in the time steps used for training. In estimation mostly we estimate current value while in prediction we try to estimate a value ahead in time.

1.2 Recent Research

The training algorithm for neural network are mostly oriented towards improving the training by tuning the hidden weights more, trying to keep the size of network as small as possible, in order to improve the computation efficiency. Large neural networks can definitely be used where needed, but in order to keep it possible to research on standard compilers and standard machines we try to keep the size of networks optimal. The most common algorithms used during the trials are 1st order algorithms like conjugate gradient to solve for output weights and back propagation to tune the hidden weights. We have also worked on the application based useful version of these algorithms which can be found on the [15].

The main problems that are found with 1st order algorithms are number of iterations being very large, there is no certainty of the networks being trained within a given time, while second order algorithms face the problem of inverting the Hessian matrix. The version of hidden weight optimization using multiple optimal learning factor [3], [4] algorithm that we use is the one in which the Hessian being collapsed, which has a very positive effect on the training. The 2nd order algorithms like Newtons algorithm and others like Levenberg Marquadt [4] have also been worked on in [2]. Other versions of training algorithms with complex hidden weights training is worked upon in [3], [4]. Clustering based algorithms like Radial basis functions and Piecewise Linear Network training algorithms were worked in details in [5] and [6]. The present version of algorithm that is used in the following thesis is a modified version of the Ortho-normal Least Squares with optimal ordering (for output weights), which has been customized to incorporate pruning. For hidden layer training, in order to accelerate training, we shall use an optimized version of Back-Propagation with multiple optimal learning factors as developed in [4] and [3], but modified again to reduce the size of networks with each iteration, till we find an optimal size of network. The concept is very close to pruning. But in the application of forecasting, since we wish to have more accuracy then optimal size of network we may sometimes have to compromise on the size of network being big, because we may not wish to throw away certain hidden units after training. We shall deal with this in details in later sections. Also a different approach of one forecaster for one hour of the day has been employed in [54], [23], [25] which can face problems of memorization due to availability of less number of training patterns.

1.3 About This Research

The problem that is addressed in this thesis is the one of forecasting/prediction of weather variables including air temperature, dew point temperature, relative humidity, solar radiation, wind direction and wind speed.

This problem of prediction has been attempted and successfully addressed previously. A group has addressed the problem rather statistically for the purpose of wind power generation. In their paper [7] they have given an account of number of methods. [8] has addressed the problem of forecasting/predicting any non-linear multivariable process using the neural networks approach. Thereby they have demonstrated the black-box modeling capability of neural networks for prediction using parametric estimation. Today the use of machine learning using the neural networks is also being made widely for finding the future effects of pollution on the ozone. Prediction of atmospheric parameters, and weather variables is also becoming an important part of geological research. In [9], a group has used the feed-forward neural networks approach coupled with pruning to do air-quality prediction. The group in paper [10] has given a detailed account of prediction using the neural networks. [11] also has addressed the issue of temperature forecasting. Any variable prediction can be independently addressed using the time-series forecast. This has been accounted for in [12] for exchange rate forecasting.

1.4 Organization of chapters in this thesis

Chapter-2 elaborates on the behavior of individual input variable's and their correlations plots and the pre-processing of each variable. In chapter 3 we discuss neural network training and the memorization problems and ways of avoiding them. In chapter 5 we discuss both the feature selections, time domain feature selection (PLOFS) and transform based feature selection. Chapter 5 will also show some results and error values and plots for PLN feature selection.

Chapter 6 has all the important results. Results will be represented separately for individual and then prediction obtained from the multivariable network will also be shown in this chapter. Multivariable network and single variable networks give almost identical performance, thus demonstrating the eligibility of concept of multivariable network. This section will also show the plots of error against the number of hidden units obtained due to Optimal ordering in OLS, thereby helping us to find the optimal number of hidden units needed in the network.

There is an appendix in the end which has the theory of Gram Schmidt Ortho-Normalization.

CHAPTER 2

DESCRIPTION ON WEATHER RELATED VARIABLES

In this chapter, section 2.1 we show some correlation sequences and relation between all types of variables using plots of autocorrelation, correlation matrices, just to demonstrate the behavior of each variable. We will show some pre-processing techniques for removing the high frequency noise from the data. Such noise makes forecasting the time-series extremely difficult. Then we show some important preprocessing method for each variable before training.

2.1 Description on the behavior of the variables

The problem at hand is to predict the five variables that we have decided to use for this research. The five variables are, air temperature (v_1), relative humidity (v_2), magnitude of wind speed (v_3), wind direction (v_4), and solar radiation (v_5). Before we go to neural network training for the predictor, we need to understand the behavior of these variables. This discussion in this section does not involve anything but intuitive understanding of the relation between the given any two variables and between the variable and itself (regression). With help of the comparative plots and the plots of the auto-correlation and cross-correlation of the variables, we can say something about what the predicted values of the variable should look like.

The cross-correlation sequence between two time series is as defined:

$$C_{v1v2}(m)=\left(\frac{1}{N}\right)\cdot\sum_{n=1}^{N-m-1} v_1(n+m)\cdot v_2(n) \qquad (2)$$

The auto-correlation is a special case of cross-correlation defined as:

$$R_{v1v1}(m)=\left(\frac{1}{N}\right)\cdot\sum_{n=1}^{N-m-1} v_1(n+m)\cdot v_1(n) \qquad (3)$$

This quantity in case of weather related variable is a very useful quantity for prediction.

Heuristically the value of variables like temperature, solar radiation and humidity will be pretty closely related or highly auto-correlated with the values of same variables 24 hr before. To say that the two values will be just scaled version of each other will be true to a very large extent. This same principle can be extended at annual level i.e. the average temperature is ought to be higher during summer of each year and will be less as winter approaches. Wind speed and direction will not fall in this category because they are really very random as will be seen from their plots coming later.

The temperature plot for 250 hrs of Madras location for year 2006 is shown below. For finer comparisons we shall take blocks of small lengths of these variables and try to observe some behavior closely.

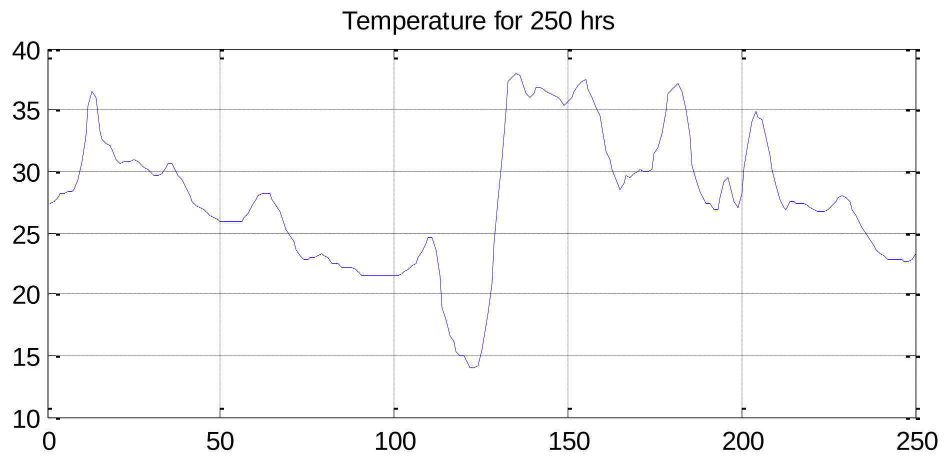


Figure 2.1 Temperature Measurement for 250 hrs

As such temperature alone may not show any periodicity. But if we look at the same temperature measure in the light of the effect of solar radiation then it will be visible that the peaks of both of them matches very closely as shown in figure 2.3.

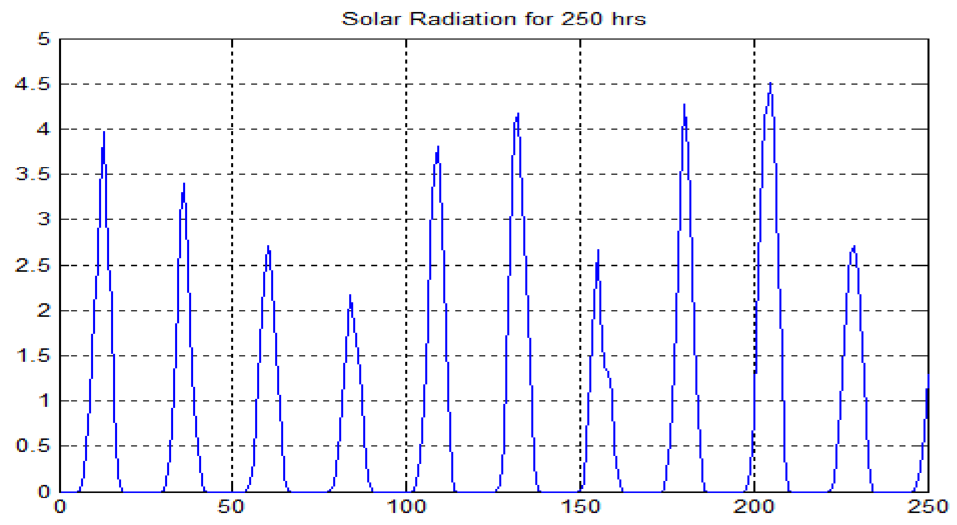


Figure 2.1 Solar radiation variations for 250 hours

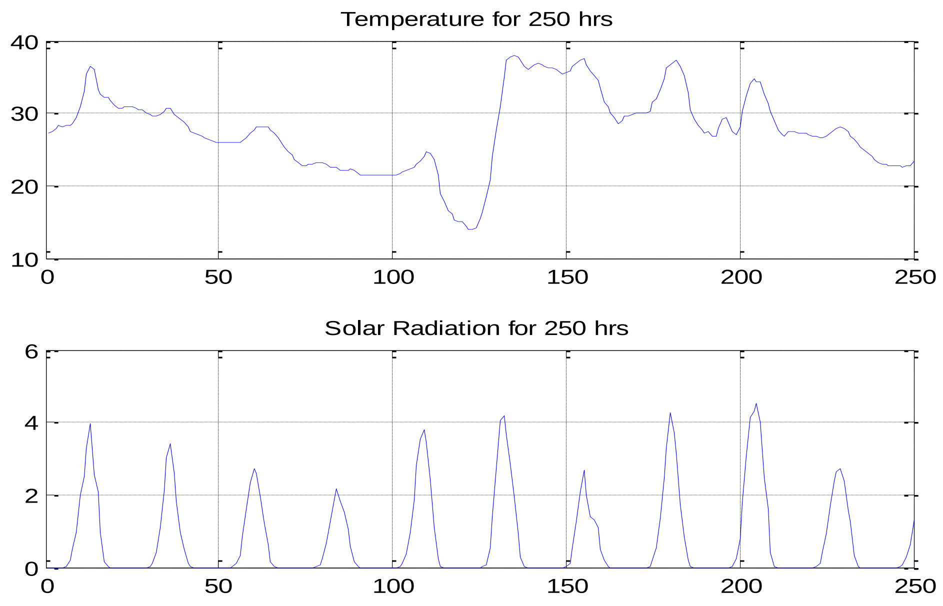


Figure 2.2 Temperature against solar radiations for 250 hours

In the same manner the relative humidity will be related in the opposite way. As the temperature rises the relative humidity would in general reduce as shown in the figure 2.4.

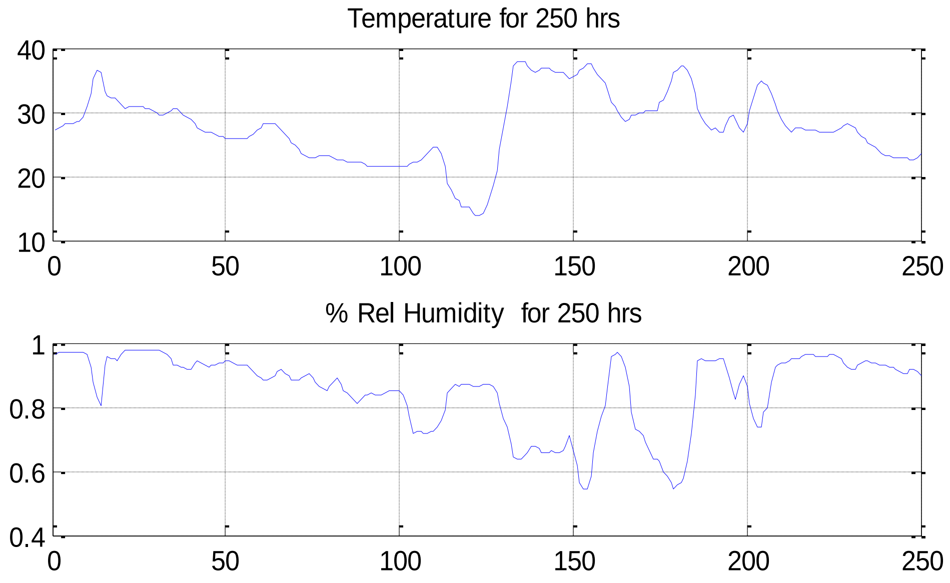


Figure 2.3 Temperature against humidity variations over 250 hours

As seen from the plots that rise in temperature is related to solar radiation while the rise in temperature leads to reduction in relative humidity. Also as shown from the figure 2.2 the solar radiations show a periodicity of 24 hours. So somehow this would lead to periodicity of 24 hrs into the temperature and relative humidity. This makes prediction of this variable a little easy. But if we look at figure 2.5 which shows the direction and speed of wind, then we see that no such periodicity of relation is observed at all. The measurements of the direction as mostly available are measured by scattero-meter. This gives measurements in degrees. This variable can be better understood in modulated form. Degree scale is not directly usable. The reason is that a slight change in wind direction causes a large discontinuity in the measurement as in figure 2.5. Besides, this variable is a circular variable. As seen from the figure 2.5 the direction has been converted to radian form between -π to π. Even then, at certain points where the wind completes a rotation of 360 degrees, there is a discontinuity of 2π radians as seen from the plot. For this reason we need to use an un-warped version of direction measurement, giving us continues version of the signal.

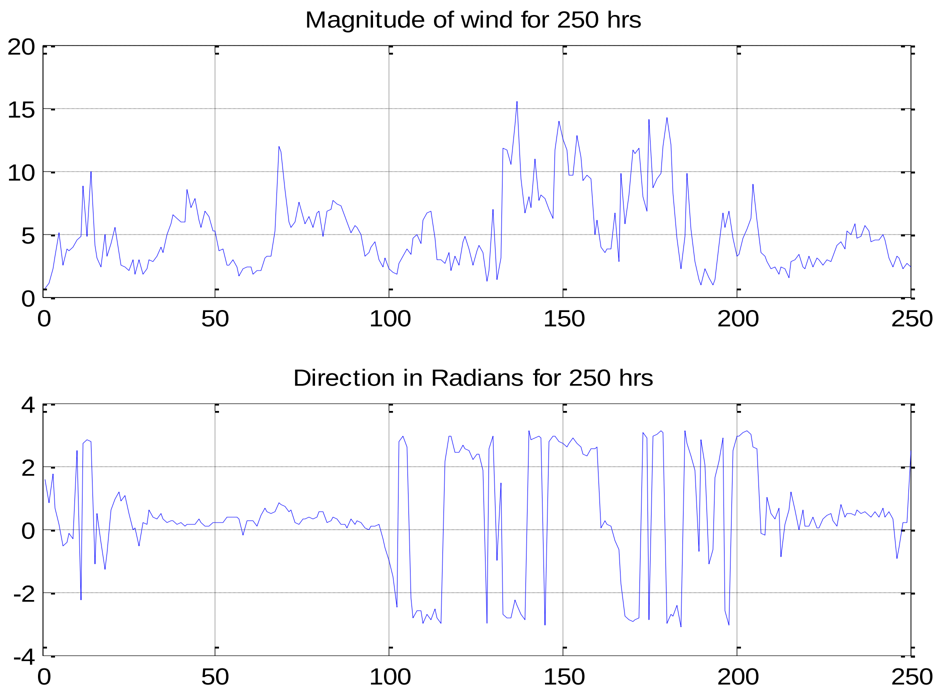


Figure 2.4 Wind speed and direction variation over 250 hours.

This treatment is a sort of post processing that we will need to do before the error calculation after the prediction is done. We shall show the details in section 2.3. For now, to understand this variable it will suffice to say that we modulate the sine and cosine of the wind direction with the magnitude of wind speed, thereby representing it in polar form.

Coming back to other 3 variables, as mentioned before the auto-correlation sequence of these variables is worth looking into at least once and thereby understanding the relation. Figure 2.6, 2.7, 2.8 shows the auto correlation of the temperature, relative humidity and solar radiation over 24 and 100 hrs respectively. It will be easy to spot out that the value of measurement 24 Hrs apart are highly correlated, and that the same pattern is repeated over everyday, but with decreasing weight as separation in time increases. Later on PLN based time domain feature selection will justify this fact. It need be said here that the correlation sequences we are plotting are all normalized correlation sequences, i.e. maximum is ‘1’ and minimum will be ‘0’ scaled accordingly.

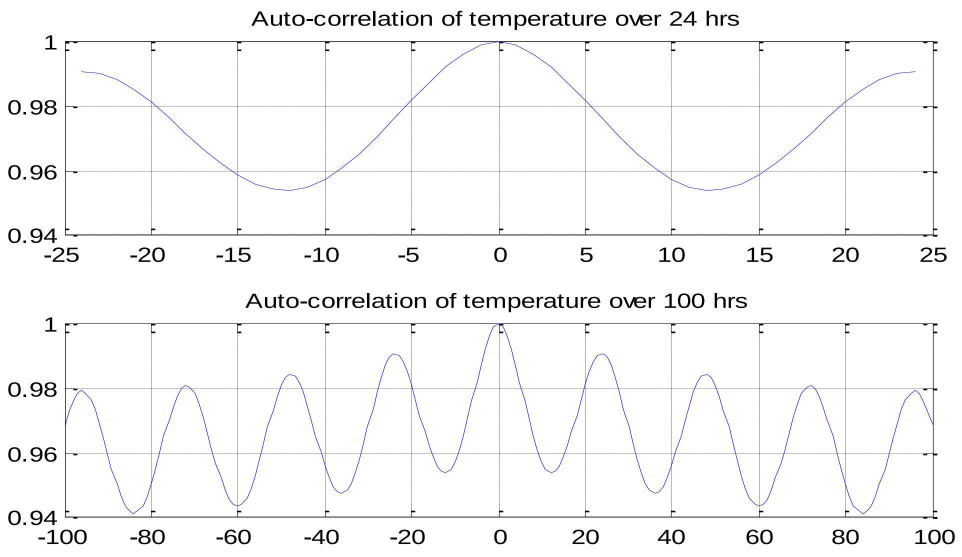


Figure 2.6 Normalized Auto-correlation plot of temperature for 25 and 100 hours

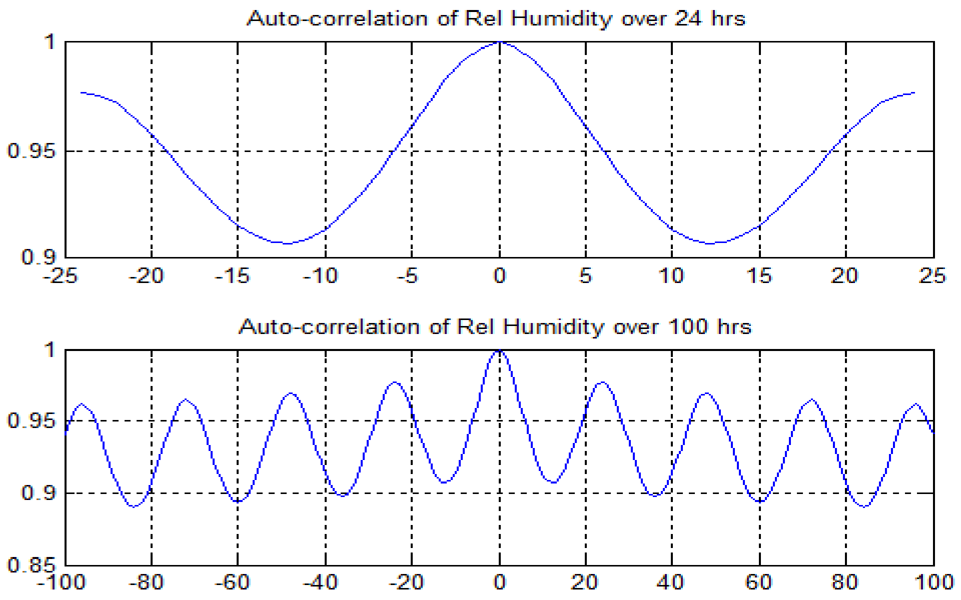


Figure 2.5 Normalized Auto-correlation plot of humidity for 25 and 100 hours

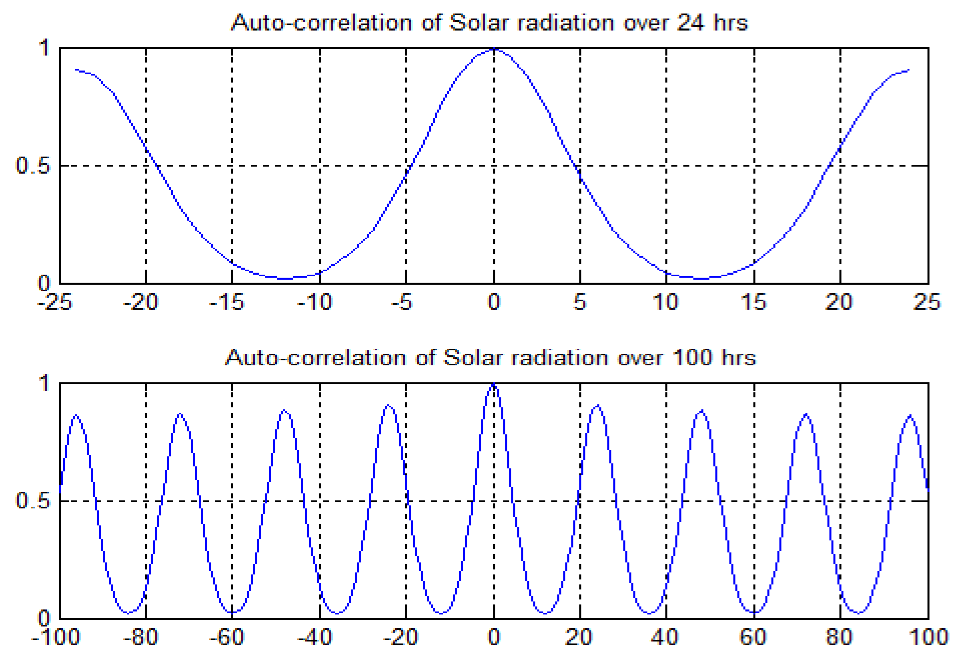


Figure 2.6 Normalized Auto-correlation plot of solar radiation for 25 hours and 100 hours

Apart from this, the same pattern of correlation also exists at annual level. This means that the average values of temperature, relative humidity and solar radiation for the 2nd month of this year is highly correlated with the 2nd month of previous year. We had availability of data over 6 years and the auto correlation of this these variables over six years is shown in figure 2.9, 2.10. It is seen from the figures that there is high correlation between the data of same month every year. As the years go further this correlation keeps reducing.

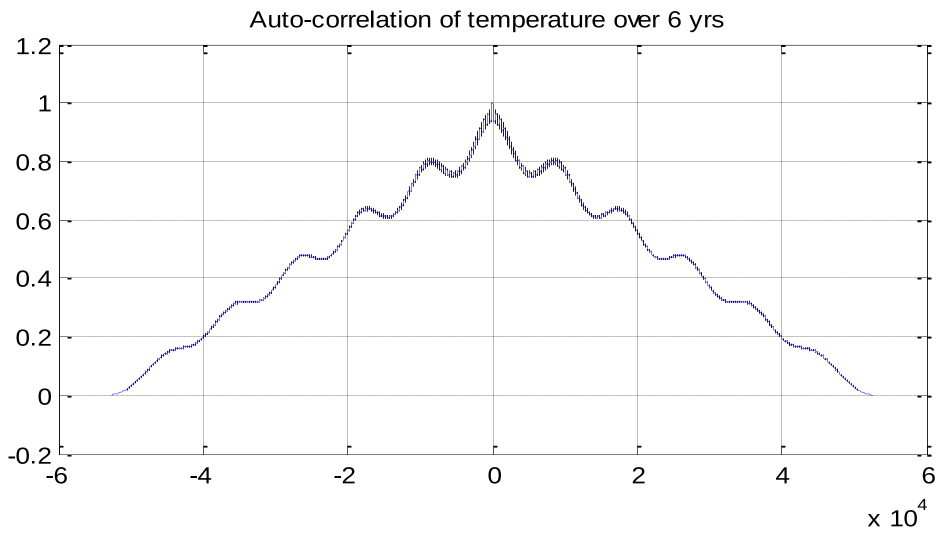


Figure 2.7 Normalized Auto-correlation plot of temperature for 6 years

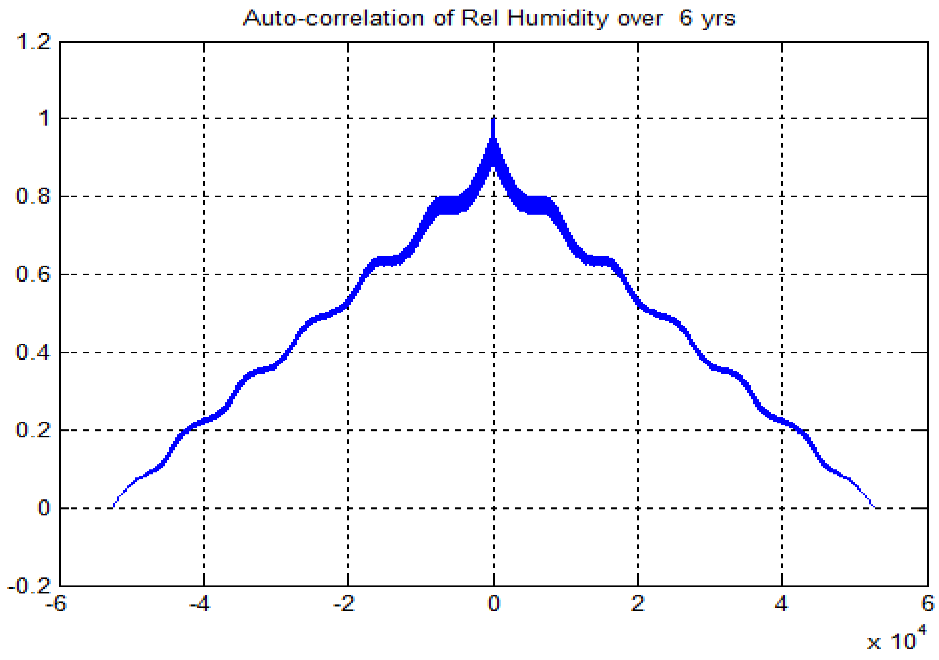


Figure 2.8 Normalized Auto-correlation plot of relative humidity for 6 years

The relationship that we tried to depict in figure 2.3 and 2.4 can also be solidly stated by showing the cross-correlation plots between solar radiation and temperature, and temperature and relative humidity. Figure 2.11 and 2.12 shows these plots over a 100 hr period.

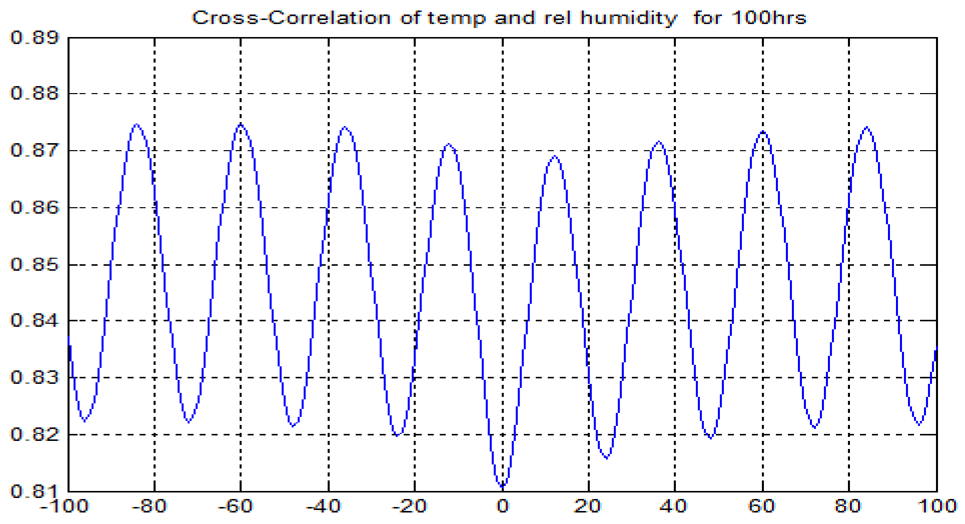


Figure 2.9 Normalized Cross-correlation between temperature and humidity for 100 hours

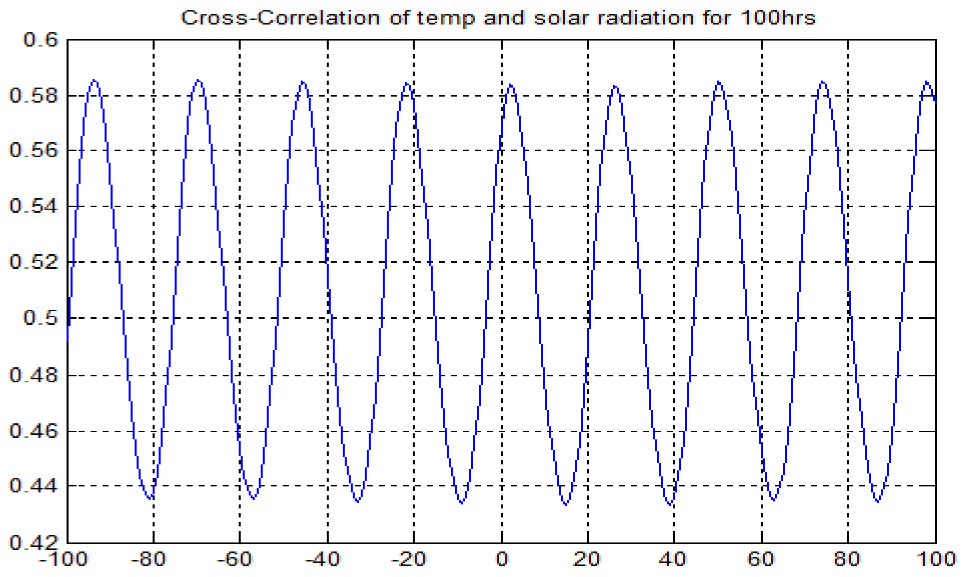


Figure 2.10 Normalized Cross-correlation between temperature and radiation for 100 hours

Observation can be made from figure 2.11 that since the cross-correlation starts with the minimum value at zero and goes to maximum, the relative humidity has an inverted relationship with temperature. Same thing can also be observed from cross-correlation between solar radiation and humidity (not shown here). At the same time a slight shift from zero in figure 2.12 shows that measurements of temperature are highly correlated with the measurements of solar radiation which were a few hours in past. Intuitively, this means that rise in solar radiation causes the rise in temperature only after an hour or so! That seems pretty practical. This kind of auto-correlation and cross correlation does not seem to apply to the wind magnitude and wind direction because they are seen to be highly random with abrupt changes figure 2.5. For this reason in section 2.3 we will show how to treat them, using modulation and using time variable as input in the neural network.

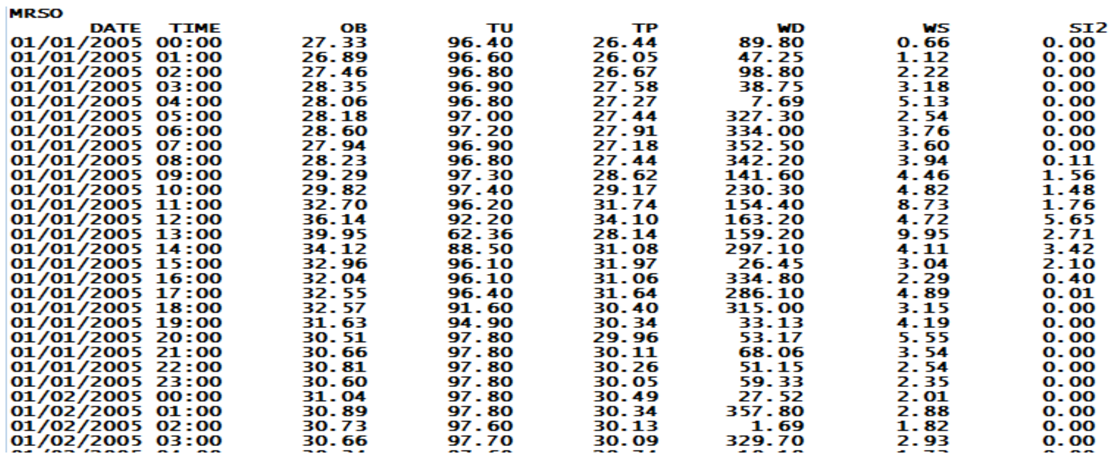
2.2 Initial treatment

One problem faced by researchers in this area is to get authentic data, which has not been manipulated by the websites on which the data is found. Also most of the data found has lot of 'MISSING' ,'BAD' values or the measurements are not available at all, may be because the station is down or sensor fails etc. Missing or bad values have to be either removed from the data or a new section of data is used altogether. Care has to be taken that during training these bad values are not encountered during the training. The data for current thesis is downloaded from [16].

The data available on web page [16] has certain format which is shown below. As mentioned in previous section we have 5 variables- column 1 is v_1 which is temperature, column 2 is relative humidity which is v_2, column 3 is dew point temperature which we shall not use because its behavior is same as v_1. Column 4 and column 5 are wind direction (v_3) and wind speed (v_4) respectively which we will encode in cartesian form as $v_4 \cdot \sin(v_3)$ and $v_4 \cdot \cos(v_3)$ and column 6 is v_5 which is solar radiation. The number of data point that we have downloaded for the research are for 6 complete years which means that hourly measurement makes it 2190 days of data i.e. 52561 measurements. In this section we shall discuss the problems with the data downloaded from the web and in the section 2.3 and 2.4 we shall discuss the solutions to such problems jointly which are implemented in the form of pre-processing.

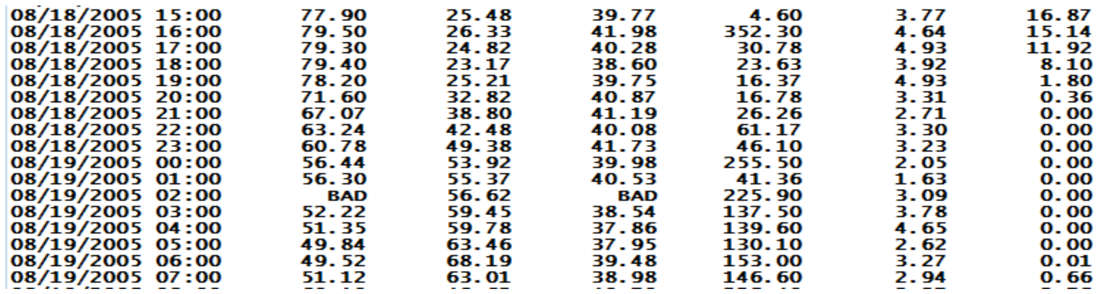
Figure 2.14 shows some location which has 'BAD' measurements. For the sake of simplicity, heuristically we replace such values with the values which are 24 hrs prior to that measurement. Also in this section we show that this data need pre-processing so we cannot leave these bad values as they are. Since the number of such values is not very large, this replacement do not make any significant impact on the training process. This is the first problem and its solution is not so difficult. Other problems are now discussed.

Looking at the nature of these variables and we find that a few of them need some processing. This processing depends totally on the data that we get on the web [16].



```
MRSO
      DATE  TIME          OB         TU         TP         WD        WS        SI2
01/01/2005 00:00       27.33      96.40      26.44      89.80      0.66       0.00
01/01/2005 01:00       26.89      96.60      26.05      47.25      1.12       0.00
01/01/2005 02:00       27.46      96.80      26.67      98.80      2.22       0.00
01/01/2005 03:00       28.35      96.90      27.58      38.75      3.18       0.00
01/01/2005 04:00       28.06      96.80      27.27       7.69      5.13       0.00
01/01/2005 05:00       28.18      97.00      27.44     327.30      2.54       0.00
01/01/2005 06:00       28.60      97.20      27.91     334.00      3.76       0.00
01/01/2005 07:00       27.94      96.90      27.18     352.50      3.60       0.00
01/01/2005 08:00       28.23      96.80      27.44     342.20      3.94       0.11
01/01/2005 09:00       29.29      97.30      28.62     141.60      4.46       1.56
01/01/2005 10:00       29.82      97.40      29.17     230.30      4.82       1.48
01/01/2005 11:00       32.70      96.20      31.74     154.40      8.73       1.76
01/01/2005 12:00       36.14      92.20      34.10     163.20      4.72       5.65
01/01/2005 13:00       39.95      62.36      28.14     159.20      9.95       2.71
01/01/2005 14:00       34.12      88.50      31.08     297.10      4.11       3.42
01/01/2005 15:00       32.96      96.10      31.97      26.45      3.04       2.10
01/01/2005 16:00       32.04      96.10      31.06     334.80      2.29       0.40
01/01/2005 17:00       32.55      96.40      31.64     286.10      4.89       0.01
01/01/2005 18:00       32.57      91.60      30.40     315.00      3.15       0.00
01/01/2005 19:00       31.63      94.90      30.34      33.13      4.19       0.00
01/01/2005 20:00       30.51      97.80      29.96      53.17      5.55       0.00
01/01/2005 21:00       30.66      97.80      30.11      68.06      3.54       0.00
01/01/2005 22:00       30.81      97.80      30.26      51.15      2.54       0.00
01/01/2005 23:00       30.60      97.80      30.05      59.33      2.35       0.00
01/02/2005 00:00       31.04      97.80      30.49      27.52      2.01       0.00
01/02/2005 01:00       30.89      97.80      30.34     357.80      2.88       0.00
01/02/2005 02:00       30.73      97.60      30.13       1.69      1.82       0.00
01/02/2005 03:00       30.66      97.70      30.09     329.70      2.93       0.00
```

Figure 2.11 Rawdata downloaded from [16]



```
08/18/2005 15:00       77.90      25.48      39.77       4.60      3.77      16.87
08/18/2005 16:00       79.50      26.33      41.98     352.30      4.64      15.14
08/18/2005 17:00       79.30      24.82      40.28      30.78      4.93      11.92
08/18/2005 18:00       79.40      23.17      38.60      23.63      3.92       8.10
08/18/2005 19:00       78.20      25.21      39.75      16.37      4.93       1.80
08/18/2005 20:00       71.60      32.82      40.87      16.78      3.31       0.36
08/18/2005 21:00       67.07      38.80      41.19      26.26      2.71       0.00
08/18/2005 22:00       63.24      42.48      40.08      61.17      3.30       0.00
08/18/2005 23:00       60.78      49.38      41.73      46.10      3.23       0.00
08/19/2005 00:00       56.44      53.92      39.98     255.50      2.05       0.00
08/19/2005 01:00       56.30      55.37      40.53      41.36      1.63       0.00
08/19/2005 02:00         BAD      56.62        BAD     225.90      3.09       0.00
08/19/2005 03:00       52.22      59.45      38.54     137.50      3.78       0.00
08/19/2005 04:00       51.35      59.78      37.86     139.60      4.65       0.00
08/19/2005 05:00       49.84      63.46      37.95     130.10      2.62       0.00
08/19/2005 06:00       49.52      68.19      39.48     153.00      3.27       0.01
08/19/2005 07:00       51.12      63.01      38.98     146.60      2.94       0.66
```

Figure 2.12 Rawdata with bad values

If the data we find on web is well conditioned or smooth than we need not do this processing. We found that the data on the above given web page had some discontinuities and roughness. So we decided to use a 1st order moving average processing [53] before we form the training file out of this data. This moving average process is used as given below-

$$v_1(n)=0.5\cdot v_1(n)+0.5\cdot v_1(n-1) \qquad (4)$$

This data-processing is very common for such kind of application [53], and as shown in the figure below; only temperature time-series was processed in this manner. Rest of 4 variables did not need such processing. The same 250 hours of temperature values are shown in figure 2.15 and readers can make it out that all that this processing does is smoothing the time series keeping the behavior of time-series same.

Another problem after the MA processing that this variable v_1 faced was the bias. Even the v_2 and v_5 also faces the same problem. It is obvious that the average values of the temperature, radiation and humidity are not the same from one month to next or from one day to next. Also average temperature for summer is not the same as in winter or monsoon or spring. Due to this these variables have a bias in them. For this reason after the pattern formation is done we shall need to use the mean removal and separating means approach which is the standard approach to deal with this problem.

Apart from this other variables have their own problems. As we mentioned just before we will encode the wind speed and direction as cartesian form. The sole reason to do this is that as you will be able to see that in column 4 the direction is in degrees. This causes a large discontinuity of 360 degrees from 360 to 0 or 0 to 360 when the direction is treated and an angle or the phase of the signal. Such discontinuity can be handled of course by neural networks but it will unnecessarily need a few extra hidden units. This will increase the size of neural network which are going to need to be as small as possible for the combined forecaster that we are proposing in this thesis. As such dealing with all 5 variables or the multi-variable model of this forecaster is already going to be large and so we don’t want to add more units to it. Besides we will see in chapter 3 how the large number of hidden units calls for larger number of iterations for training. For this reason modulation or encoded form of this variable is necessary. Now since we pre-process using the modulation we need special post processing after prediction is done. In order to bring it back to phase form or 0 to 360 form the post-processing will be described in next section which shows how to deal with the warping problem when we use arctangent function. We shall now elaborate the solutions to these main problems.

2.3 Useful Encoding Of Discontinuous Variables To Form Training Data

This section deals with few of the most important points needed to be mentioned before the measurements are used for forming a training file for neural network. In this section we basically reveal the most complex of all input variables. This variable is the wind speed and wind direction measurement. As seen in figure 2.13 the wind speed data is really very rough (columns 8 and 9). The changes from one hour to the next are very large.

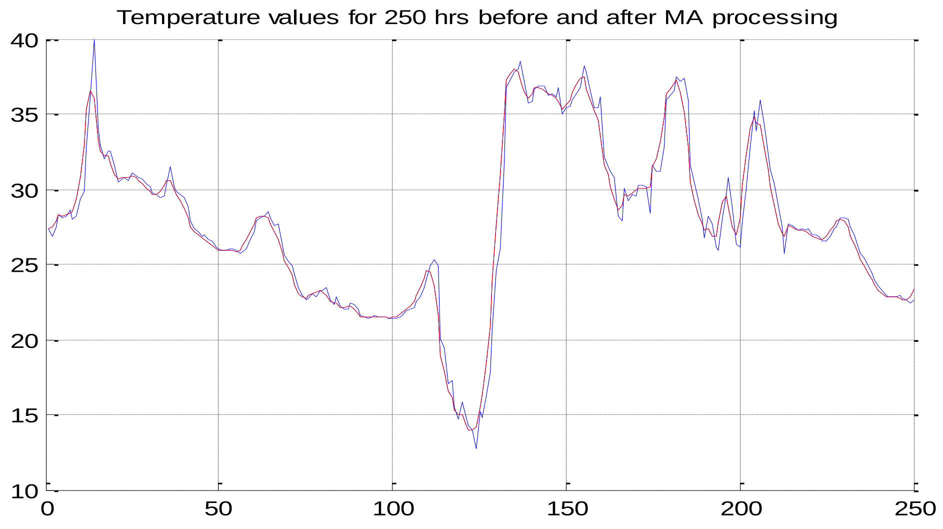


Figure 2.13 Temperature variations before and after moving average processing

At the same time the wind direction measurements are in degrees and after conversion to radians as in figure 2.5 still there are discontinuities from –π to π or the other way. We must understand the behavior of the wind direction variable before we start using it for prediction purpose. First thing that we would notice about it is that it is circular variable. It is not periodic at all. The circular behavior of this variable is shown in figure 2.16 which is a rose or a compass chart plotted in MATLAB. The direction of the wind can therefore be considered as a phase of the signal [23], [24], [25], wind speed being the magnitude of the complex number. In order to predict the correct phase of the system we need to use the unwrapped version of the phase.

The unwrapped version of the phase over a large time period of 250 hrs looks as follows in figure 2.17. The processing of the wind speed and direction is therefore done in a modulated manner. As mentioned before v_3 is the wind speed and v_4 wind direction in radians between $-\pi$ and π. The neural network input form that we will use to form the patterns for training will be then as follows-

$$V_3 = v_3 \cdot \cos(v_4),\ V_4 = v_3 \cdot \sin(v_4) \qquad (5)$$

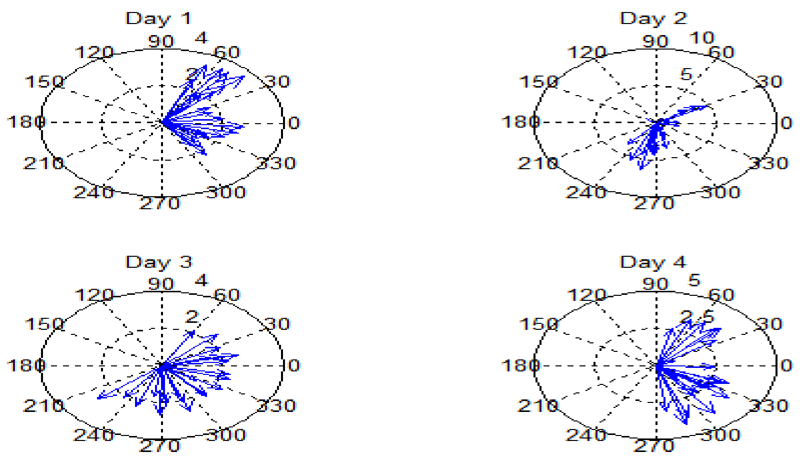


Figure 2.14 Rose chart/polar plot for 4 separate days of wind variations

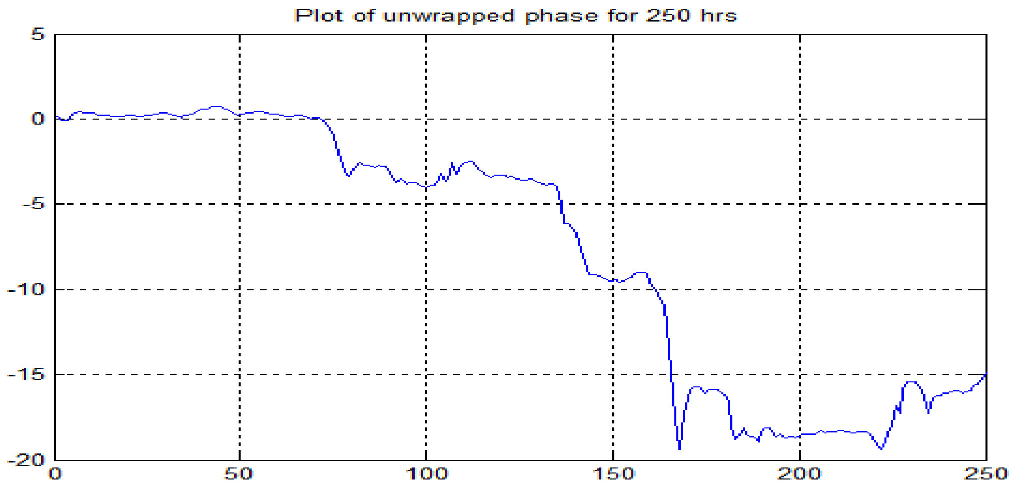


Figure 2.15 Plot of unwrapped phase/direction of wind.

This form of inputs can be feed to neural network as two separate inputs [54]. The number of inputs in the neural network would remain the same because we converted 2 variables of magnitude and phase into the 2 new variables real and imaginary part. The actual difference that this modulation would make will be visible from the difference between the auto-correlation plots of wind speed and wind direction in radians and auto-correlation plots of the real and imaginary parts of wind that we formed by the previous equation. Figure 2.18, 2.19, 2.20, 2.21 show these correlations over 6 year period.

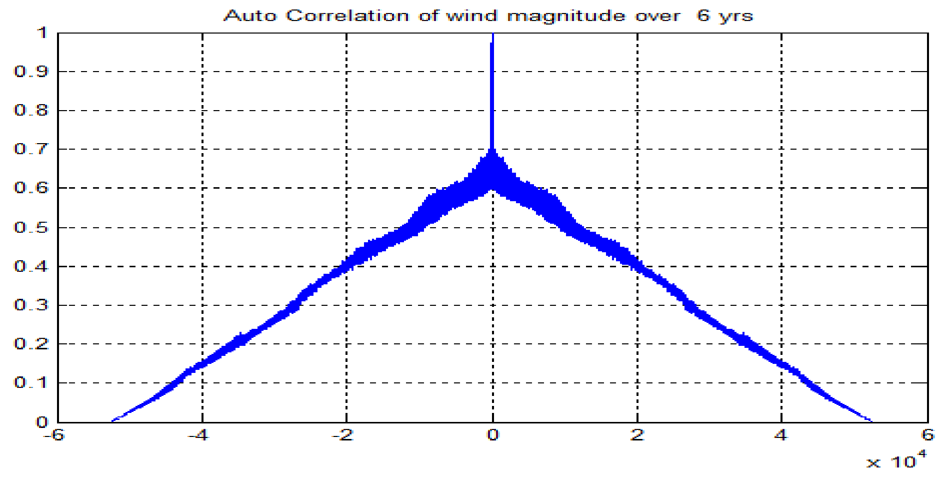


Figure 2.16 Normalized Auto-correlation of wind speed for 6 years before modulation/encoding

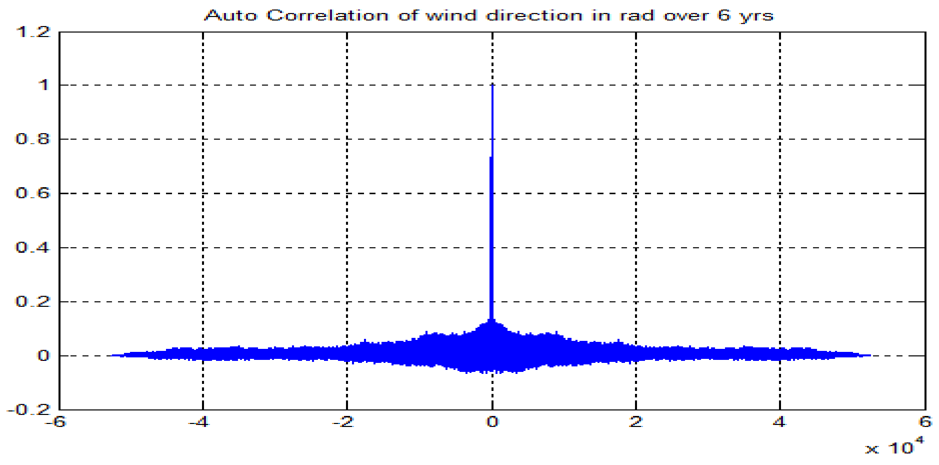


Figure 2.17 Normalized Auto-correlation of wind direction for 6 years before modulation/encoding

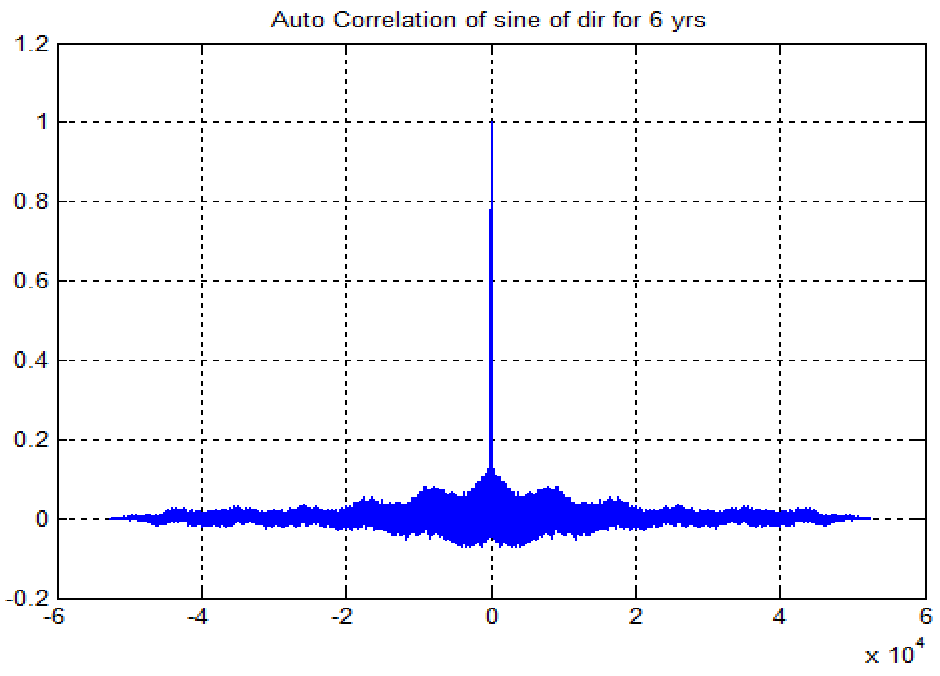


Figure 2.18 Normalized Auto-correlation of imaginary part of modulated/encoded wind for 6 years

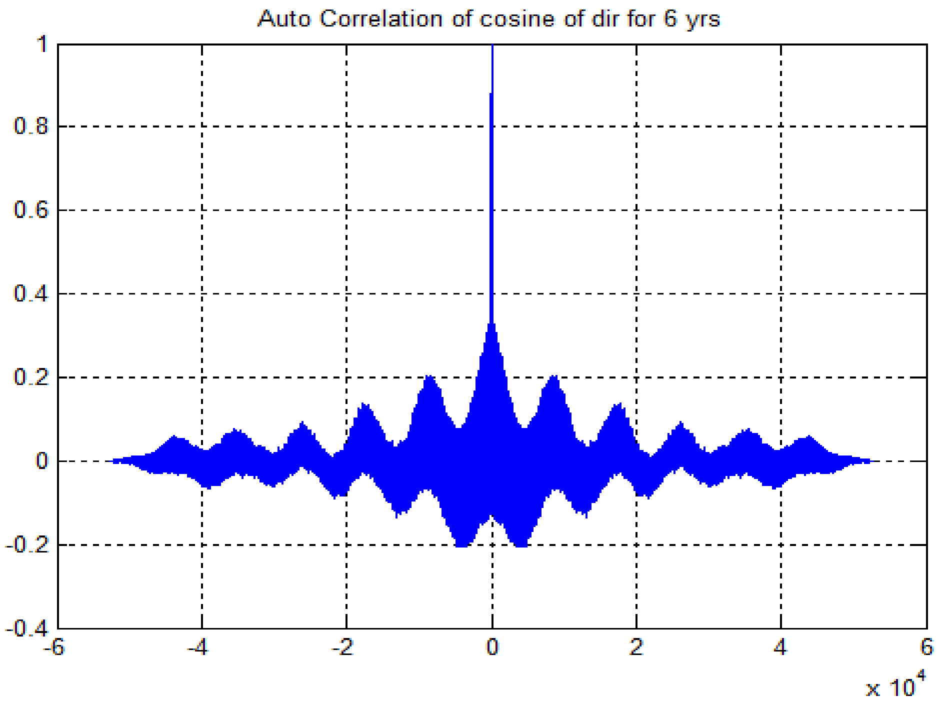


Figure 2.19 Normalized Auto-correlation of real part of modulated/encoded wind for 6 years

As it is visible that the large difference that we are making into the auto-correlation sequence of the variable by modulation. Figure 2.20 and 2.21 look pretty much in same form with 6 lobes alike figures 2.9 and 2.10. The lobes show that the measurements over the months of the year are correlated with each other from one year to the next, thereby making it useful for the prediction. No such lobes are seen in the figure 2.18 and 2.19.

One another most unique point about the method of prediction or neural network training that we are using is the use of time inputs. As shown in figure 2.13 the day, date, and the hour of the day are also encoded using the sine and cosines for making them useful for prediction. We first convert the month number of the year and the date of the month to the day of the year scale. This converts the date in figure 2.13 to 0 to 1 scale. Identically we convert the hour of the day to 0 to 1 scale. We use sine and cosine of these values [54] as follows-

$$V_6 = \sin\left(\text{doy}\cdot 2\cdot \frac{\pi}{365}\right)\ ;\ V_7 = \text{cosine}\left(\text{doy}\cdot 2\cdot \frac{\pi}{365}\right);\quad \text{doy=day of the year between 0 and 365} \tag{6}$$

$$V_8 = \sin\left(\text{hod}\cdot 2\cdot \frac{\pi}{24}\right);\ V_9 = \text{cosine}\left(\text{hod}\cdot 2\cdot \frac{\pi}{24}\right);\quad \text{hod=hour of the day between 0 and 24} \tag{7}$$

This gives us a way of keeping a track on what kind of changes in the variable took place at what time of year and what hour of the day. The figure 2.22 shows the modulated wind signals which look much smooth than what we saw in figure 2.5. Another advantage that this modulation will have is related to neural network principle component analysis. As seen in figure 2.5 the magnitude of wind and direction both will have positive numbers forever. These variables need to be preprocessed so that mean value of it over all training patterns is close to zero or at least be small as compared to its standard deviation. If all the values are positive, than weights of the ANN in the hidden layer can only increase or decrease together. This will slow down the back-propagation as the change in weight vectors can only be possible by going to and fro across the error surface .But only after this modulation we get positive and negative numbers.

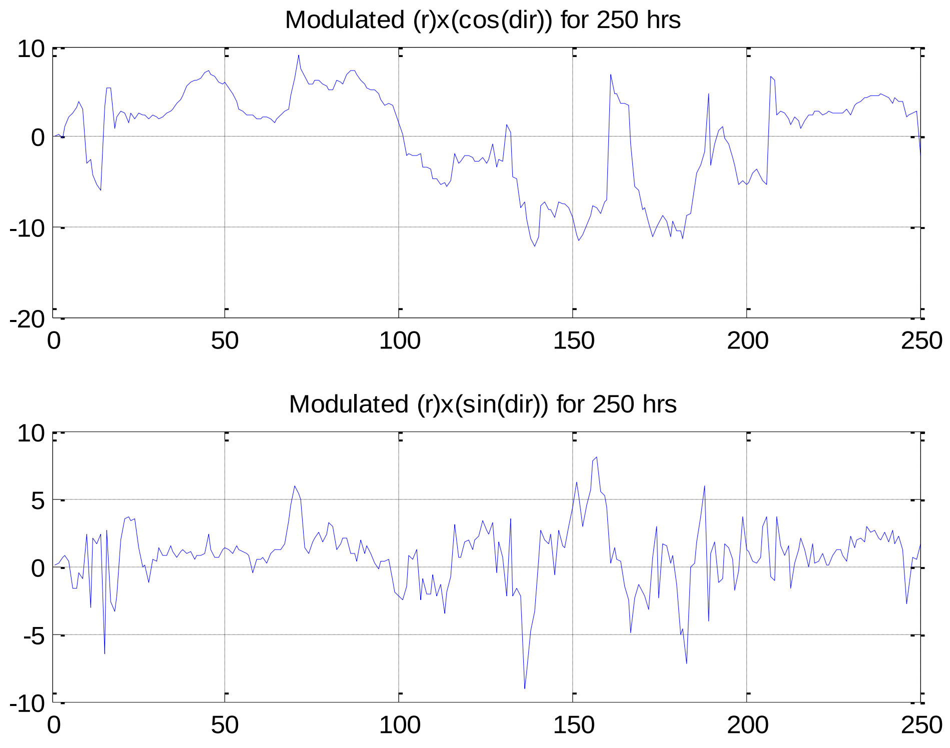


Figure 2.20 Variations in real and imaginary part of modulated/encoded wind for 250 hours

Now since we use the modulated version of the wind variables we also need to do some post-processing after the prediction obtained from the ANN undergoes reconstruction (elaborated in chapter 4). This post processing is simple as follows.

$$v_3=\sqrt{V_3^{\,2}+V_4^{\,2}}\,;v_4=\text{atan2}\left(\frac{V_4}{V_3}\right) \tag{8}$$

This ‘atan2’ includes the correction factor which is defined as follows.

$$\text{atan2}\left(\frac{V_4}{V_3}\right)=\arctan\left(\frac{V_4}{V_3}\right);V_3>0 \tag{9}$$

$$=\pi+\arctan\left(\frac{V_4}{V_3}\right);V_4\geq 0,V_3<0 \tag{10}$$

$$=-\pi+\arctan\left(\frac{V_4}{V_3}\right);V_4<0,V_3<0 \qquad (11)$$

$$=\frac{\pi}{2};V_4>0,V_3=0 \qquad (12)$$

$$=-\frac{\pi}{2};V_4<0,V_3=0 \qquad (13)$$

$$=\text{undefined};V_4=0,V_3=0,\text{totally unlikely} \qquad (14)$$

Even after this is performed we will need to unwrap the phase of the calculated v_4. Unwrapping this signal of phase in order to get a signal with no or rarely and 2π discontinuities is important post processing part in treating the v_4 variable, because these large discontinuities will contribute to error which will then be large. So after the phase signal is retained by atan2 as above we use the following algorithm for unwrapping the signal and then time domain prediction error calculation-

Assuming that the first sample phase is less than π/2. We start from second sample. We calculate the difference of phase between two consecutive (previous) samples (current sample and sample immediately to left of it).

- If this difference is larger than +π then we subtract 2π from the sample itself and from all the samples to the right.i.e. keep an account of number of 2π added and add that many to all the samples coming in future. This will increase the magnitude, but as such 2π rotation does not change where, in space, the direction is pointed towards.
- If this difference is less then –π then we add 2π to the sample itself and to all the samples in its right i.e. coming in future.

Even after this process is done there may still be some discrepancy. The problem is that when we calculate the error between the predicted phase and the actual phase, both are to be unwrapped using the above method. In that process it may happen that the unwrapping may not give same phase signal for both the signals, but will be 2π apart. As such it is not a problem because this difference is always then a multiple of 2π. This is an inherent problem in phase estimation. So we use modulo operation in calculating the error i.e.

$$\text{Error}(i)=\left(v_4(i)-\widehat{v_4}(i)\right)\%2\pi \qquad (15)$$

Here 'i' stands for sample number. What this does is if there are any extra 2π rotations they will be removed and the error will be brought down and will actually represent the error in pointing the direction. As such an error of 2π means an error of zero. We shall see the positive effects of this post processing of phase signal in chapter 5 where we shall put the results and plots of predicted and expected signals against each other.

Another small thing about the relative humidity is that, since it is measured in percentage we convert it to 0 to 1 scale by division by 100.

2.4 Mean removal and separating mean

In order to attend to the problem of bias we use the technique of separating mean after the pattern formation in time domain is done. Assuming the dimension of input vector is N (we will see later that this is the input space dimension initially in time-domain). N_v is the number of patterns for training. If $\mathbf{x}_p$ is the input of dimension N and $\mathbf{t}_p$ is the desired output of dimension M. Then two processes as in [69] are as follows-

Mean Removal-

$$m_i=\left(\frac{1}{N_v}\right)\sum_{p}^{Nv} x_p(i) \qquad (16)$$

$$\overline{x_p}(i)=x_p(i)-m_i \qquad (17)$$

Separating mean- In this process, as mentioned below, the means of the input of each pattern is subtracted from each of the output pattern. This calls for a separate post-processing step where in we add this mean of the each (training and testing and also during the normal processing) input pattern after the reconstruction of outputs is done by inverse KLT.

$$m_p = \left(\frac{1}{N}\right) \sum_{i}^{N} x_p(i) \tag{18}$$

$$\overline{t_p}(j) = t_p(j) - m_p;\ j \text{ from 1 to } M \tag{19}$$

These processes will remove any bias in any pattern as well as makes the input pattern zero-mean. This also makes the compression easy because if all the inputs are zero mean then the correlation matrix and covariance matrix are essentially the same. More importantly if we use this mean-separating method than we need to add the mean values of each respective pattern to the predicted output values. If we let $\overline{y_p}$ to be the approximated output from a trained neural network then the actual output y_p is defined as follows-

$$y_p(j) = \overline{y_p}(j) + m_p;\ j \text{ from 1 to } M \tag{20}$$

CHAPTER 3

NEURAL NETWORK TRAINING

In this chapter, section 3.1 we shall go through the basic MLP notations and structure. We shall also go through in detail the Back Propagation algorithm as in [17]. Then we shall also revise the OWO algorithm of OLS that we are using. The actual theory of Gram Schmidt Ortho-normalization can be found in appendices. In Section 3.2 we shall describe in detail the HWO-MOLF algorithm developed in the lab by previous students like [4], [3]. In 3.3 we shall discuss memorization and way of avoiding it.

3.1 MLP Notation And Basic Training Of OWO-BP

We shall start with an introduction on the general neural networks training paradigm and then specify in detail the notations that we shall use in the rest of this section. This section only deals with the training process that we follow for the forecaster. Of course a lot of preprocessing is done before the training is commenced especially in this case because we deal with variety of variables and also because we do not use the time domain data for training. In order to make the neural network smaller we use 2 stage feature selection and we use the KLT domain or these KLT features to train out network. Therefore the notation mentioned in this section for the neural network are that of the one which imply right before the training is to be started i.e. after the feature selection algorithms are already done.

We are using a MLP to approximate the non-linear function of forecasting. The figure for the same is given below if figure 3.1. The notation '$\mathbf{x_p}$' is used for the input pattern number 'p' to the neural network and $\mathbf{t_p}$ for the desired pattern number 'p' of the output. Index p of the patter runs from 1 to N_v. The dimension of $\mathbf{x_p}$ is assumed as 'N+1' and that of '$\mathbf{t_p}$' is taken as M. The notation '$\mathbf{y_p}$' is used to represent calculated or approximated output, and it has same dimension M as well. We shall use different indices to access the specific element in the vectors of these and many other matrices and vectors. These indices will be mentioned as and when required. We use the N_h as notation for number of the hidden units at the beginning of the training. The matrix $\mathbf{W_{oh}}$ is used as the weight matrix connection hidden units to output units and $\mathbf{W_{ih}}$ is used to represent the matrix of weights connecting the input to the hidden units. The dimension of $\mathbf{W_{oh}}$ is M X N_h and that of W_{ih} is N_h X (N+1). The weight matrix of bypass weights is $\mathbf{W_{oi}}$ which has dimension of M X (N+1). We will access the input pattern elements as $x_p(N+1)$ and hidden units as $N_h(k)$ while desired outputs as $t_p(i)$ and $y_p(i)$. The vector $n_p(k)$ is used to represent the net function defined later. The $O_p(k)$ notation will be used for the activation function for the hidden units 'k'.

In general input pattern is considered as a vector [$x_p(1)$ $x_p(2)$ … $x_p(N+1)$]. There are always N input elements as patterns and '1' is used as the thresholding input representing the bias. We have used $x_p(1)$ as '1'.

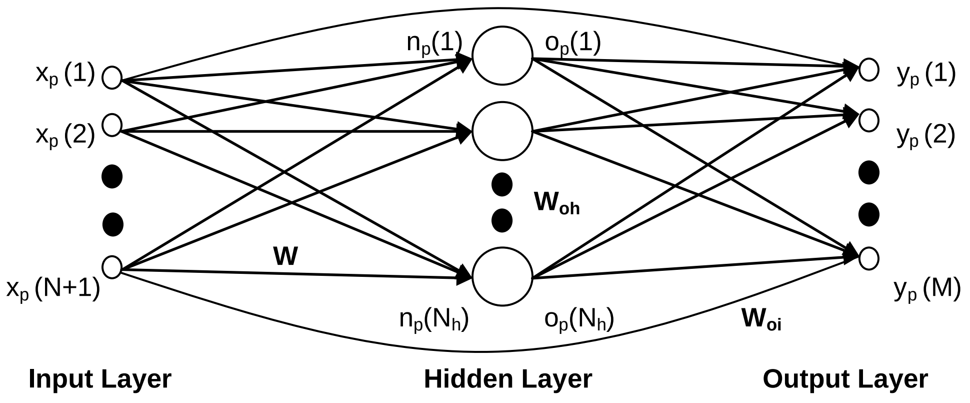


Figure 3.1 A fully connected MLP

The paradigm of training a MLP type neural network since [17] has that been of back propagation. Back propagation of algorithm basically uses the error between the desired function and the approximated function and derivatives of this error to train the layers of neural network. For a 3-layer network we take one backward derivative as mentioned in the following equations-

$$E = \frac{1}{N_v} \sum_{i=1}^{M} \sum_{p=1}^{Nv} [\, t_p(i) - y_p(i) \,]^2 \;=\; \sum_{i=1}^{M} E(i) \tag{21}$$

$$E(i) = \frac{1}{N_v} \sum_{p=1}^{Nv} [\, t_p(i) - y_p(i) \,]^2 \tag{22}$$

The net function for k^th hidden unit $n_p(k)$ is defined as –

$$n_p(k) = \sum_{n=1}^{N+1} w_{ih}(k,n).x_p(n) \tag{23}$$

$$\mathbf{n_p = W_{ih}.\ x_p} \tag{24}$$

There are plenty of activation functions that are being used today depending of application. Readers are referred to [1] for more about this. We are using the most common version of sigmoid function defined as below-

$$O_p(k) = f(n_p(k)) = \frac{1}{1 + \exp\left(-n_p(k)\right)} \tag{25}$$

The output $y_p(i)$ is calculated as follows-

$$y_p(i) = \sum_{n=1}^{N+1} w_{oi}(i,n) \cdot x_p(n) + \sum_{k=1}^{Nh} w_{oh}(i,k) \cdot O_p(k) \tag{26}$$

$$\mathbf{y_p = W_{oi} \cdot x_p + W_{oh} \cdot O_p} \tag{27}$$

$$\mathbf{Y_P = W_o \cdot X_p} \tag{28}$$

$$\mathbf{W_o} = [\mathbf{W_{oi} : W_{oh}}] \text{ and } \mathbf{X_p} = \mathbf{[x_p : O_p]} \tag{29}$$

Considering the above equation the output weights $\mathbf{W_o}$ can be solved using the gradient of error w.r.t output weights ($\frac{\partial E}{\partial \mathrm{W_o}}$). This derivative of error w.r.t weights (gradients) is as follows-(bold E is the expectation operator)

$$g(m) = -\frac{\partial E(i)}{\partial w(i,m)} = -2 \frac{1}{N_v} \sum_{p=1}^{Nv} \left[\left(t_p(i) - \sum_{k=1}^{N+Nh+1} W_o(i,k) \cdot X_p(k) \right) X_p(m) \right] \tag{30}$$

$$=-2[c(m,i)-\sum_{k=1}^{N+Nh+1} W_o(i,k)r(k,m)] \quad (31)$$

Equating the g(m) to zero yields-

$$\mathbf{C}= \mathbf{R}\cdot\mathbf{W}_o^T \quad (32)$$

$$\mathbf{C}= \frac{1}{N_v}\sum_{p=1}^{N_v}\mathbf{X}_p \cdot \mathbf{t}_p^T \quad (33)$$

$$\mathbf{R}= \frac{1}{N_v}\sum_{p=1}^{N_v}\mathbf{X}_p\cdot \mathbf{X}_p^T \quad (34)$$

This sets of linear equations can be solved for weights using any linear solver. Having defined these quantities we shall now go ahead with the basic Back propagation discussion given by [17]. We shall mention some theory and basic equations of gradient or back-propagation algorithms functioning.

The back-propagation algorithm is a version of gradient algorithm apart from others like 1st order conjugate-gradient and 2nd order like Gauss-Newton and Levenberg [3], [4], [5]. The derivative of error function with respect to the weight function is defined as below. This is also considered as gradient of error. It basically defines the slope of error. We assume here that error function can be minimized by the gradual changes in the weights so that we can find a position in the weight matrix which can approximate the given outputs tp as close as possible. As in [17] the derivatives (when there are more than one layers (MLP), are called 'delta' functions. Each layer has a delta function which depends on the derivative of the error functions. Delta function for each pattern for each output, at the output layer for the MLP is obtained by the first derivative of the error function using the chain rule-

$$\delta_{po}(i)=2[\, t_p(i) - y_p(i)\,] \quad (35)$$

This delta or error derivative propagated backwards at the hidden units is derived again using the chain rule is given below. They are also called delta functions but these are defined for each of the hidden layers units-

$$\delta_p(k)=O_p^{'}(k)\sum_{i=1}^{M}\delta_{po}(i)\cdot w_{oh}(i,k)\,,\ \boldsymbol{\delta_p}=[\,\delta_p(1),\,\delta_p(2),\ldots\,\delta_p(N_h)]^T \tag{36}$$

$$\frac{\partial E}{\partial w_{ih}(k,n)}=-g(k,n)=(\frac{1}{N_v})\sum_{p=1}^{Nv}\delta_p(k)\cdot x_p(n) \tag{37}$$

$$\mathbf{G}=(\frac{1}{N_v})\sum_{p=1}^{Nv}\boldsymbol{\delta_p}.\,\mathbf{x_p^T} \tag{38}$$

The back-propagation algorithm modifies the parameters of the MLP i.e. hidden weights using a learning factor which is constant scalar or a vector. The weights of hidden units are gradually tuned to reduce the error function at the output $y_p(i)$.

$$\mathbf{W_{ih}} \leftarrow \mathbf{W_{ih}} + z\cdot\mathbf{G} \tag{39}$$

$$\Delta\,\mathbf{W} = z\cdot\mathbf{G} \tag{40}$$

The learning factor z can be a scalar, a vector, or even a matrix depending on the adaptability desired as in [18], [19], [3], [4].This algorithm only uses the first order derivative of the error function w.r.t. to the weights. The back-propagation uses the first order derivative of the error with respect to weights, which is a Jacobian of size N_h X (N+1) computed as before. Second order methods with optimal learning factors will be elaborated in next section.

The training of the neural network goes on iteratively using the algorithm of OWO-BP. The OWO stands for any method that may be used for solving the output weights and bypass weights. This includes algorithms like conjugate gradient which is first order algorithm or may be even OLS. We shall use the later due to its inherent ortho-normal property and since we wish to prune a few basis functions (hidden units) later which do not contribute much towards approximation. The theoretical details of OLS can be found in appendix 1. The OWO algorithm of OLS we are using is modified version of OLS with an optimal ordering vector. This is a step for pruning in order to find and optimal size of network. With these output weights being solved for each iteration, the error value at each output also change in each iteration. These new calculated error values are back-propagated through the hidden units activation functions, using the gradients of which, we tune the hidden units. This is what BP stands for. For now to understand, the BP algorithm updates the weights **W** using the **G** which acts as the direction of learning and the z which is the learning factor tells the size of the step to go in each iteration of the update. In primitive versions of the training the learning factor z was used to be specified heuristically while in [18] and in other profound algorithms the learning factor is calculated optimally using the first order and second order derivatives of the error with respect to the learning factor. The BP in its pure form is what [17] talks about. After him researchers [18] have generated their own version of BP which adds lot of adaptability to the modeling. The learning factor calculation that we are about to show in next sections uses the Hessian matrix to derive optimal values of learning factors. The MOLF or multiple optimal learning factor is even more adaptable as it means a 'learning factor' for each of the hidden unit neuron. With this kind of iterative OWO-BP going on, the error value at the output is seen to reduce drastically, meaning the MLP weights are adapting to approximate the given patterns at the output.

One way of improving the convergence is to use second order derivatives or the hessians matrix. Newton method does that.

Newton's Method-

Newtons method uses the second order derivative of the error function w.r.t. network weights. The weight updates in this algorithm is-

$$\Delta \mathbf{W} = -\mathbf{H}^{-1}\cdot\mathbf{G} \tag{41}$$

H matrix in this algorithm is N_w X N_w , where N_w is the number of all the weights in the network $(N+1+M)N_h+ (N+1)M$.

The newtons method uses the quadratic approximation of the error function. This quadratic property is assumed to be with respect to the change in weights.

Assuming this change as-

$$\mathbf{e=w^{'}-w} \tag{42}$$

The Taylor's theorem for expansion of this error approximation is-

$$\mathbf{E' \approx E - e^T g + \tfrac{1}{2}\, e^T \cdot H \cdot e}$$

Taking-

$$\frac{\partial \mathbf{E'}}{\partial \mathrm{e}}=0 \tag{43}$$

$$\frac{\partial \mathbf{E'}}{\partial \mathrm{e}} = \mathbf{-g+H\cdot e}=0 \rightarrow \mathbf{H\cdot e=g} \tag{44}$$

$$\mathbf{w^{'}=w+e} \tag{45}$$

The inherent problem of calculating the inverse of this Hessian and the large size of Hessian makes the use of this method inappropriate for training process which is iterative. Thereby one way of applying this algorithm is only apply this inverse to the Hessian of the error with respect to only input weights. It is important to know the structure of this Hessian. The $\mathbf{H_o}$ is the hessian of the output weights whose dimension is $((N+N_h+1)\cdot M)$ X $((N+N_h+1)\cdot M)$. The off-diagonal component $\mathbf{H_{io}}$ may be rectangle of square matrices but in general of dimension $(N+1)N_h$ X $(N+N_h+1)M$.

$$\mathbf{H}=\begin{bmatrix} \mathbf{H_r} & \mathbf{H_{io}} \\ \mathbf{H_{io}}^T & \mathbf{H_o} \end{bmatrix} \tag{46}$$

This new hessian is $\mathbf{H_r}$ component of $\mathbf{H}$. The bypass weights and output weights are anyways solved for in the OWO part by OLS.

Each element of the this new Hessian is then calculated as follows-

$$\frac{\partial E}{\partial w_{ih}(k,n)}=\left(\frac{-2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}[\, t_p(i) - y_p(i)\,]\frac{\partial y_p(i)}{\partial w_{ih}(k,n)} \tag{47}$$

$$\frac{\partial E}{\partial w_{ih}(k,n)} = -g(k,n) = (\frac{1}{N_v})\sum_{p=1}^{Nv} \delta_p(k)x_p(n) \tag{48}$$

$$\delta_p(k) = O_p^{'}(k)\sum_{i=1}^{M} \delta_{po}(i)w_{oh}(i,k) \tag{49}$$

$$\therefore \frac{\partial E}{\partial w_{ih}(k,n)} = -g(k,n) = (\frac{1}{N_v})\sum_{p=1}^{Nv} O_p^{'}(k)x_p(n)\sum_{i=1}^{M} \delta_{po}(i)w_{oh}(i,k). \tag{50}$$

$$\delta_{po}(i) = 2[\, t_p(i) - y_p(i)\,] \tag{51}$$

$$\therefore \frac{\partial E}{\partial w_{ih}(k,n)} = -g(k,n) = (-\frac{2}{N_v})\sum_{p=1}^{Nv} O_p^{'}(k)x_p(n)\sum_{i=1}^{M} [\, t_p(i) - y_p(i)\,]w_{oh}(i,k). \tag{52}$$

The second derivative will then be simply the auto-correlation matrix **H** (Hessian still)-

$$\frac{\partial^2 E}{\partial w_{ih}(k,n)\partial w_{ih}(q,r)} = \left(\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M} \frac{\partial y_p(i)}{\partial w_{ih}(k,n)}\frac{\partial y_p(i)}{\partial w_{ih}(q,r)} \tag{53}$$

$$= \left(\frac{2}{N_v}\right)\sum_{i=1}^{M} w_{oh}(i,k)w_{oh}(i,q)\sum_{p=1}^{Nv} x_p(n)x_p(r)O_p^{'}(k)O_p^{'}(q) \tag{54}$$

In this algorithm the learning factors are not derived but only assumed or heuristically used. This shall be furthered in next section when we discuss HWO-MOLF algorithm.

3.2 Multiple Optimal Learning Factor

In this section we shall briefly discuss the Optimal learning factor z given in previous section for the second order algorithms. And then extend the approach for multiple optimal learning factors. This approach has been designed and implemented by others [3], [4] previously. So readers are referred to see them for detailed hessian analysis. Here we shall give the main equation of the optimal learning factor and the derivation for MOLF algorithm. What makes our algorithm different and special is that this algorithm is being used for back-propagation in the hidden units along with the modified OLS which actually orders the basis functions i.e. the hidden units and bypass weights.

Remember that gradient matrix derived in previous section was considered as negative of the direction in which the training should move. In that derivation we minimize the error function with respect to the weight matrix in order to get the new weight update through the hidden units. In order to derive an optimal learning factor we have to minimize the error function with respect to this learning factor [2], [3], [4].

Expressing the error function as a function of the learning factor, using equation (19), and $y_p(i)$ as a function of the hidden weights which are expressed as function of the learning factor z-

$$y_p(i)=\sum_{n=1}^{N+1} w_{oi}(i,n)x_p(n)+\sum_{k=1}^{Nh} w_{oh}(i,k)\cdot O_p\left(\left(\sum_{n=1}^{N+1}(w_{ih}(k,n)+z\cdot g(k,n))\right)x_p(n)\right) \tag{55}$$

Taking the first parital derivative w.r.t. the learning factor 'z'-

$$\frac{\partial E}{\partial z}=\left(-\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}[\, t_p(i)-y_p(i)\,]\sum_{k=1}^{Nh} w_{oh}(i,k)\cdot O_p{}'\left(n_p(k)\right)\sum_{n=1}^{N+1} g(k,n)x_p(n) \tag{56}$$

The Gauss-Newton approximation of the second partial derivative is expressed as-

$$\frac{\partial^2 E}{\partial z^2}=\left(\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}\left[\frac{\partial y_p(i)}{\partial z}\right]^2=\left(\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}\left[\sum_{k=1}^{Nh} w_{oh}(i,k)O_p{}'\left(n_p(k)\right)\sum_{n=1}^{N+1} g(k,n)x_p(n)\right]^2 \tag{57}$$

Using the derivative of the Taylor series expansion of the error as a function of learning factor 'z' and equation it to zero we arrive at the optimal learning factor as follows-

$$z=-\left(\frac{\frac{\partial E}{\partial z}}{\frac{\partial^2 E}{\partial z^2}}\right) \tag{58}$$

Of course many times this calculated OLF may not work as well as the heuristic learning factor used earlier. But what implies from this is that the elements of the reduced hessian weighed by the gradient elements can be used for calculation of the learning factor. If this learning factor is used to weigh the gradients, then it becomes a first order algorithm but with learning factor that is derived from the second order derivatives. But this derivation gives the mathematical support for the Multiple optimal learning factor calculation which work better than most of 2nd order algorithms.

The motivation behind the MOLF [3], [4], [5] is the intuitive reasoning behind the pruning. Road map of pruning is that we look at the error function as a function of number of hidden units. The hidden units or the basis functions are, in each iteration, evaluated in order to check its contribution in reducing the error function values. Depending on the non-linearity to be approximated different hidden units contributes to a different level. And in that process, a few hidden units get tuned very greatly based on the input it responds to. At the same time, some hidden unit may not contribute to the approximation to a large extent resulting into, it being tuned/trained to a very small extent. So it will make more sense if we could train each of the hidden units based on its importance (or train all of them, and then remove least important ones during actual processing or testing). This is where we use the multiple optimal learning factor, where in we use one OLF for each hidden unit, so that we can tune each of them independently (from each other).

We shall now go through its derivation which is a step ahead from OLF derivation done previously. Here we have a vector z_k instead of a scalar z. k runs from 1 to N_h. z_k will be used to update weights w(k,n) associated with only k^{th} hidden units, but for all inputs. Therefore 'n' runs from 1 to N+1.

Output $y_p(i)$ is expressed as a function of z_k as well –

$$y_p(i)=\sum_{n=1}^{N+1} w_{oi}(i,n)\cdot x_p(n)+\sum_{k=1}^{Nh} w_{oh}(i,k)\cdot O_p\left(\left(\sum_{n=1}^{N+1}(w_{ih}(k,n)+z_k\cdot g(k,n))\right)x_p(n)\right) \quad (59)$$

All other quantities remain same apart from z_k.

$$g_{molf}(e)=\frac{\partial E}{\partial z_e}=\left(-\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}[\, t_p(i)-y_p(i)\,]\frac{\partial y_p(i)}{\partial z_e} \quad (60)$$

$$=\left(-\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}[\, t_p(i)-y_p(i)\,]\, w_{oh}(i,e)O_p^{'}\left(n_p(e)\right)\sum_{n=1}^{N+1}x_p(n)g(e,n) \quad (61)$$

Letting –

$$\triangle n_p(e)=\sum_{n=1}^{N+1} x_p(n)g(e,n) \tag{62}$$

$$O_p(z_k)=f_p\left(\left(\sum_{n=1}^{N+1}(w_{ih}(k,n)+z_k\cdot g(k,n))\right)x_p(n)\right) \tag{63}$$

$$\frac{\partial E}{\partial z_e}=\left(-\frac{2}{N_v}\right)\sum_{p=1}^{Nv}\sum_{i=1}^{M}\left[t_p(i)-\sum_{n=1}^{N+1}w_{oi}(i,n)\cdot x_p(n)+\sum_{k=1}^{Nh}w_{oh}(i,k)\cdot O_p(z_k)\right] \cdot\left[w_{oh}(i,e)O_p^{'}\left(n_p(e)\right)\cdot\triangle n_p(e)\right] \tag{64}$$

Now the second derivative of error w.r.t. the individual learning factor gives us the element of hessian we call $\mathbf{H_{molf}}$. Using separately and expressing second derivative as product-

$$\frac{\partial^2 E}{\partial z_e \partial z_f}=\left(\frac{2}{N_v}\right)\sum_{i=1}^{M}[w_{oh}(i,e)w_{oh}(i,f)]\sum_{p=1}^{Nv}O_p^{'}\left(n_p(e)\right)O_p^{'}\left(n_p(f)\right)\cdot\triangle n_p(e)\cdot\triangle n_p(f) \tag{65}$$

$$\frac{\partial^2 E}{\partial z_e \partial z_f}=\sum_{n=1}^{N+1}\sum_{m=1}^{N+1}\left[\left[\frac{\partial^2 E}{\partial w_{ih}(e,n)\partial w_{ih}(f,m)}\right]\right]\ g(f,n)g(e,m),\ \text{where} \tag{66}$$

$$\left[\frac{\partial^2 E}{\partial w_{ih}(e,n)\partial w_{ih}(f,m)}\right]=\left(\frac{2}{N_v}\right)\sum_{p=1}^{Nv}x_p(m)x_p(n)O_p^{'}\left(n_p(e)\right)O_p^{'}\left(n_p(f)\right)\sum_{i=1}^{M}[w_{oh}(i,e)w_{oh}(i,f)] \tag{67}$$

$$h_{molf}(e,f)=\sum_{n=1}^{N+1}g_f(n)\sum_{m=1}^{N+1}h_R^{e,f}(n,m)\,g_e(m),\therefore \mathbf{H_{molf}}^{\mathbf{4}}\mathbf{=g_e^T H_R^{e,f} g_f} \tag{68}$$

Superscript means 4th dimension. Most significant part to understand here is that unlike the OLF case, here we have 2 separate indices(e and f) for each learning factor and 2 separate indices along the weight matrix(n and m). Therefore each element of Hessian $\mathbf{H_{molf}}$ has to be accessed using 4 indices for given fixed e and f. The two gradient vectors are $\mathbf{g_e}$ and $\mathbf{g_f}$ have element going from 1 to N+1 for n. Therefore taking all hidden units, **g** will be a matrix(may be non-square). The $h_R^{e,f}(n,m)$ is a reduced version of Hessian just like we had a reduced version of the Hessian in last section. $h_R^{e,f}(n,m)$ is a 2D matrix. Only difference is that, in last section we considered diagonal elements (such as 1 index from 2 by i=j), but here having 4 dimensions we have e=n and f=m to form the diagonal version of the 4D matrix. Each element on LHS is obtained be weighed gradients from the N+1 row and columns of reduced Hessian 2D matrix $\mathbf{H_R^{e,f}}$(since n and m indices go from 1 to N+1) .

Using the derivative of $\mathbf{z}$ using the Taylor expansion as in last section $\triangle \mathbf{z}$ can be calculated by inverting the Hessian $\mathbf{H_{molf}}$. We do this using the OLS and solve for the vector of learning factor.

$$\triangle \mathbf{z} = \mathbf{H_{molf}}^{-1}\mathbf{g_{molf}} \quad (69)$$

Once these values of $\mathbf{z}$ vector are calculated then the back-propagation using these learning factors for the hidden units can be done. Thereby overall OWO(OLS) - BP(HWO-MOLF) algorithm is as follows-

1. Initialize network
2. Calculate net functions and solve for O/P weights and bypass weights by OLS (coming next).
3. Calculate the $\mathbf{g_{molf}} = [\, -\partial E/\partial z_1, -\partial E/\partial z_2 \ldots, -\partial E/\partial z_{N_h}\,]$, Hessians and MOLF $\mathbf{z}$ (this also uses OLS).
4. Update hidden weights

$$w(k,n) \leftarrow w(k,n) + z_k \cdot g(k,n) \quad (70)$$

and go back to step 2.

For further reading on detailed advantages of this algorithm please refer to [20], [4].

Brief discussion OWO-OLS with optimal ordering -

Output weight optimization using OLS with optimal ordering [21]:

The concept of the output weight optimization is basically that of a linear equation solver. The neural network is actually a non-linear function approximator. But the non-linearity is mainly hidden inside the activation functions and its derivatives that we use for hidden unit outputs and back-propagation respectively. Once the hidden unit outputs have been derived in each iteration, after the learning factor calculation and input weight updates, the output weights of the neural network are solved for by using any kind of linear equation solver. But ortho-normality is always a favourable property that we wish to have in order to make sure linear independence of the outputs. For this reason OLS solution is used for solving for the output weights. Gram-Schmidt procedure is always a preferred in such cases. The theory behind it can be found in the appendix. But we need to use that process in a recursive manner, which is given in details in [21]. Again even an advanced version of this algorithm in which we actually order the hidden units and inputs separately is needed to be used.

The hidden units, as we mentioned are the basis functions once they are tuned to minimize the error function. People do use very large networks in very complex applications. In the case of prediction, as we will see in the next section, sometimes we need to use the data over last few days. Even if we use hourly measurements to do this, we end up having input patterns of dimension 100. At the same time the more values we want to predict ahead, larger will be the output patterns. So if we want to predict data for even one day ahead we will have a neural network with 100 inputs and 24 outputs in case of linear networks. Apart from this we have certain number of hidden units to process also, in each iteration. This will be the scenario for each weather variable. Now the predictor that we are proposing in this thesis is even more complicated and bigger. In section 2 we showed heuristically and by plots how the different variables are inter-related. The temperature and solar radiation and humidity all such variables are totally related to each other. So we propose a different forecaster in next chapter, which uses the input pattern of all these variables simultaneously. For 5 variables mentioned in chapter 2, this will lead to a very large network with large number of inputs and with large number of hidden units as well. For this reason we wish to exploit the ortho-normality of the Schmidt process to actually order the hidden units and the inputs of the neural network separately so that atleast a few can be removed. This process of pruning the hidden units is also given in detail in [21].

Pruning is a kind of feature selection process that is applied to the network after training, during the validation. We call it post-training feature selection wherein we chose basis function. While in this thesis, we use time-domain feature selection already before the data is fed into the network. This time-domain feature selection is justified for reasons of memorization problems described in next section. But the point is, that since we do that feature selection (pre-training feature selection) we do not need to do the pruning feature selection. In time-domain feature selection we actually try to find out, which hour of measurement is more important for predicting the measurements of the hours that we want. Once that is done, than using this OWO with optimal ordering will give us a further insight on whether we need pruning or not [22]. If during the OLS with optimal ordering we find error function value not being impacted upon by a few hidden units, than we need pruning, not otherwise. So using OLS with optimal order but no pruning is justified in this case of prediction application.

The theory related to this is given in [21] we shall not give the related mathematical treatment. The mathematical treatment for the OLS is given in appendix, which also gives the pseudo-code or the algorithm for coding this OLS which gives the optimal ordering of the hidden units and the inputs. Readers are recommended to refer to them in order to have a good understanding on that OWO part.

CHAPTER 4

MEMORIZATION AND WAYS TO AVOID IT

The problem of forecasting has been solved in different ways by a number of different researchers. The ideas of forecasting the weather variables have been also done. Most of the approaches referred to till now have their inherent problems and advantages. For example prediction has been a done in [23] for the application of load forecast. Similarly, [24] has done load forecast with a different perspective wherein they have used the specific samples to predict specific hour of measurement. They have also used weighing of samples for forecasting. While researchers in [25], [23] has used the approach of SVD-QR decomposition on the correlation matrix to select the most important inputs from the given input patterns. The different ways of approaching this problem has given out comfortably good results. But this thesis has a motive of finding a universal solution, a solution that can be used to predict all kind of weather variables, for all hours of the day, for all over the year. The problem approached by most of researchers has also been limited to predicting mostly temperature. We have been successful in predicting temperature, solar radiation, relative humidity, wind speed and wind direction. The best part is that we use the same network for all of them.

The biggest problem that most of the above researchers have knowingly or unknowingly faced in this problem of prediction is the problem of memorization without learning. In order to predict the variable to a better extent, most of the researchers end up using a network with a very large number of inputs. [26] gives an elaborate explanation of a relation of memorization with the number of inputs and the number of parameters or weights in the network. The size of the network plays the most important part in this problem. The best way to tackle this issue is to have a thin and a tall training file i.e. in other words try to have a very very large N_v and smallest possible N. The universal approximation theorem in [1], inherently maintains that there will be some error in the approximation i.e. the approximation cannot be 100 percent accurate. So no matter how much prior data we use to predict the future values we will not be able to predict with 100 percent accuracy. Of course the training process has an impact on the error, but there is an inherent value of error which is independent to the kind of training performed and the network dimensions. This error is called the bound on the approximation. The concept of bound is now also as old as neural networks and so we will present some results of bounds which have already been derived by famous mathematicians and statisticians. These bounds are basically defined right from the time before we start training.

The bounds depend on the number of inputs, number of hidden units, the number of training patterns used for training, and number of free parameters or the weights in the network. This is where the concepts of curse of dimensionality, empirical risk minimization, structural risk minimization, validation of network and pruning play a very important role. Bounds and all of the above are ways of finding an optimal number of inputs, optimal number of hidden units, number of training patterns, validation patterns and also the number of iteration of training needed to approximate the given function most optimally and efficiently. For the problem of prediction we will use these methods only to justify the fact that time-domain feature selection, KLT domain feature selection, and the validation after training using the OLS with optimal ordering are of significant importance, in order to deal with the large size of network that we may end up getting. This is because, as mentioned before if we want to deal with prediction of all 5 variables (humidity, solar radiation, temperature, wind speed and direction) simultaneously, then we do have a large input and hidden space, and so finding the right size of both along the minimum number of iterations will help a lot to run the final product speedily on most of the standard compilers with good accuracy.

4.1 Memorization or Pattern Storage

The predictor application of the neural networks is a function estimation model of MLP. Assuming that if **x** is the pattern taken randomly from an environment and also that this pattern represent the environment with negligible error (no misguidance/noise), the learning machine returns the output or the response **t**. Thereby {**x,t**} forms a training sample such that f(**t**|**x**). The MLP is a learning machine that is capable or made capable to implement such a function **y**(**x**,**w**) by training. Here **w** is an information vector/matrix which has a structure defined by N, M, N_h as well as type of activation function. Also **w** belongs to **W** which is a set of all such possible combinations of these structures. The neural network design is a task of choosing the specific **w** which will be able to implement this function **y**(**x**,**w**). The selection is based on the training set of N_v randomly independent observations, {**x**,**t**}. Thereby the risk minimization between the desired response **t** and response **y**(**x**,**w**) is called the training process or learning process. This risk function can be analytically defined for an observation as in [1] called empirical risk E_{emp}:

$$E_{emp}(\mathbf{w})=(1/N_v)\sum_{p=1}^{Nv}\left\|\mathbf{t}_p-\mathbf{y}(\mathbf{x}_p,\mathbf{w})\right\|^2 \qquad (71)$$

The main expectation out of the training process is that the machine should be able to **generalize** the behavior of the environment and not just **memorize** the patterns that are posed in front of it. Lagrangian interpolation wherein the N_v data points are exactly fit by the polynomial of degree N_v-1 is a classic example of memorization.

Here we shall introduce some important terms for explaining the bounds on pattern storage capability of a neural network. This bound on the storage is a measure of minimum number of patterns that are needed to train a network with no memorization occurring. The first term is storage capacity 'C'. It is the minimum number N_v of the random input vectors (patterns) that can be mapped exactly to corresponding desired output vector, thereby giving zero training error (memorizing). Second term is M_{MLP} is the number of multiplies needed to process a random pattern through the network after the training is complete (for MLP it is $N_h(N+M)+NM$).

This value is a true representative of the size of the network. Greater the M_{MLP}, bigger is the network. Third is the absolute free parameters $P_{ab}=N_w$ of a network are the actual weights and the thresholds that can be varied (tuned) during the training. Higher the ratio C/P_{ab} means good efficiency and so use of small networks is always preferred because they are not capable to memorize the training pattern. Indirectly therefore, the problem of memorization is faced when the number of patterns available to us for the training is very less. From the [28] and [29], it can be stated that lower bound on pattern storage/memorization capability of a fully connected neural network is $C_{MLP}=N+N_h+1$ while the upper bound for the same is $N_w/M=P_{ab}/M$. For fully connected MLP type neural network $P_{ab}=N_w=N_h(N+M+1)+M(N+1)$, is the number of weights or free parameters in the network. This implies that N_v must be atleast larger than this value. Also if we assume N_{in} as weights connecting to each output node, then a lower bound on its pattern storage is N_{in}. Apart from this, efficiency of a network for storing the patterns/memorizing is defined as Nw_{ef}/N_w ,where Nw_{ef} are effective free parameters. For further reading refer to [28] and [29]

A lot of measures to detect or measure this phenomenon of memorization have been statistically devised. There are certain number of N_v patterns that are needed to train the network with a specific network dimension (i.e. N_w, N_h, N and M) optimally. The concept of structure risk minimization (described next) is a methodology of training where in we try and vary this dimension of the network(by varying the N_h) by varying the N_w. As the training proceeds (i.e. with each iteration we change the N_w) and training error reduces there is a point during the early stages of training where the problem of approximation is over-determined due to larger network capacity. But after few iterations of changing N_w, a stage is reached in the process after which the problem is under-determined. This is where training error is still reducing but validation error starts increasing. Thus during the training we try to co-ordinate the P_{ab} (by changing N_w parameters), with the N_v patterns that are available to us for the problem. A result in [31] shows that for sigmoidal activation function network for a given N_v, $(N_w)^2$ is the number representing the optimal network, N_w is the number of free parameters or weights in the network.

Hughes [26] on the other hand showed an identical phenomenon for fixed N_v, wherein the error or empirical risk as compared to Bayes or optimal error increases with increase in N(input dimension). Also same phenomenon is displayed with increase in N_w (number of free parameters), keeping the N_v and N constant. Thereby it is concluded that as the number of inputs are increased, the memorization can increase and generalization deteriorates. This memorization would give an accurate output for the patterns that it has been memorized, but when a slightly different pattern is posed to it then error will be very large. That is, it does not know what the system response should be towards a new type of input.

4.2 Ways To Avoid Memorization

Having known what the theory behind the memorization is, we shall now elaborate on how to avoid the harmful effect of memorization. Then we show how are we in this thesis making sure that even with a large network it does not succumb to the effect of memorization.

There are a few ways of tackling this issue during the training.

- First method is wherein process of risk minimization is divided into training and validation. The validation is the process wherein new patterns are processed by a trained network and its response is observed and error is calculated. During this process of training and validation the most usual graphs/curves obtained for standard functions are described now. As the iterations of training are increased, then the training error keeps on reducing, but the validation error keeps on increasing after certain iterations. The concept of **early stopping** was therefore introduced, wherein, after each iteration, we calculate the training and the validation errors. As soon as we find the validation error increasing at some iteration number, we stop the training process. This is one way to substitute the structural risk minimization process in which we would change the number of free parameters of the system by deleting some synaptic weights. So instead of varying the number N_w we actually vary the N_{it} iteration numbers. This process is much simpler than changing the N_w. It is assumed that after some iterations the network is learning only noise. We shall see such phenomenon occurring in real for our application in chapter -5.

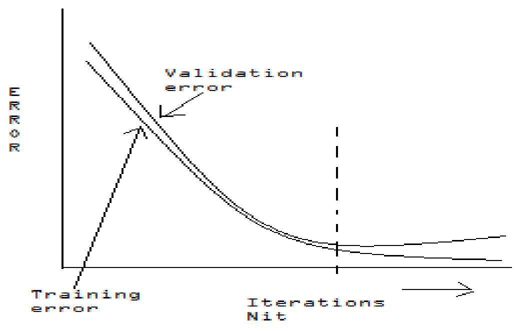


Figure 4.1 Standard Training and Validation Error curves versus Number of Iterations.

- Yet another method in order to avoid memorization is to generate larger number of patterns, so that even if a slightly larger network (large N or N_h) is used then also we would have enough data to train a larger networks without applying structural risk minimization (i.e. changing N_w, by changing N_h). The idea is to increase the sampling density which is proportional to $(N_v)^{1/N}$,[1].

As mention earlier, that the SRM, using the method of changing the N_w is a little cumbersome and difficult, because it makes training longer and complex. But it has an inherent advantage of giving us a small final network. So it would be really good if we can find another way of doing the SRM i.e. changing the number of free parameters during the training, to get a smaller and an optimal network. The answer to this question is pruning [22]. Pruning is the concept of evaluating the significance of each tuned hidden unit in risk minimization i.e. validation error. In order to realize pruning, in each training iteration we solve for output weights (OWO) using a special version of OLS which orders the hidden unit basis in the order of their significance towards reducing the validation error [21]. We calculate the risk function/validation error over each hidden units individually and try and order them in order of their significance. When we find that a last few neurons are not making much impact on the validation error reduction, we delete them. It is for this purpose that we need to order the hidden units or the basis function during the OWO process. We do this by OLS with optimal ordering [21] and try and find an optimal size of hidden layer thereby realizing the SRM in a novel way. The theory for OWO is given in appendix 1 and a small discussion is also made in section 3.2.

Finally after all this discussion, we can take home a few important heuristics that every researcher, designing neural network, should remember and try to implement.

- Memorization is an inherent problem with all neural networks and it increases with N. Therefore keep N as small as possible.
- The approximation error (risk function) can never go to zero, infact not below the bound values.
- In order to have a good accuracy, empirical fit ratio of N_h/N_v must be as small as possible.
- To get better accuracy on approximation, N_h must be large.
- From above 2 it is intuitive to know that N_v must be very large, as large as possible. Thereby N/N_v must be as small as possible i.e. small dimension of input training patterns.
- Validation error, and not the training error is the true measure of usefulness of training.
- We want C_{MLP}/N_w and C_{MLP}/M to be large, i.e. network must be as small as possible
- Given MLP and training algorithm we want to choose N_h such that $C_{MLP} <<<<< N_v$, and $N_v > P_{ab}/M$, to promote generalization and prevent memorization.

$$N_v >>>> P_{ab}/M \tag{72}$$

4.3 Precautions Taken In Our Network To Avoid Memorization

Large N_v:As referred before, in this section, this problem of prediction has been solved in various ways, all limiting to one variable or a single time series prediction. In this thesis we are a doing multiple time series prediction simultaneous. So the number of inputs to the networks as well as number of hidden units ought to be larger than other cases. It is therefore more important for us to follow all above heuristics as strongly as possible. Most importantly **get large number of patterns N_v** and **compress the number of inputs to as small a dimension possible** without affecting the approximation capability and then **throw away all possible useless hidden units by pruning**. Our first step will be to generate as many patterns from the given data. Other 2 will be addressed in next chapter.

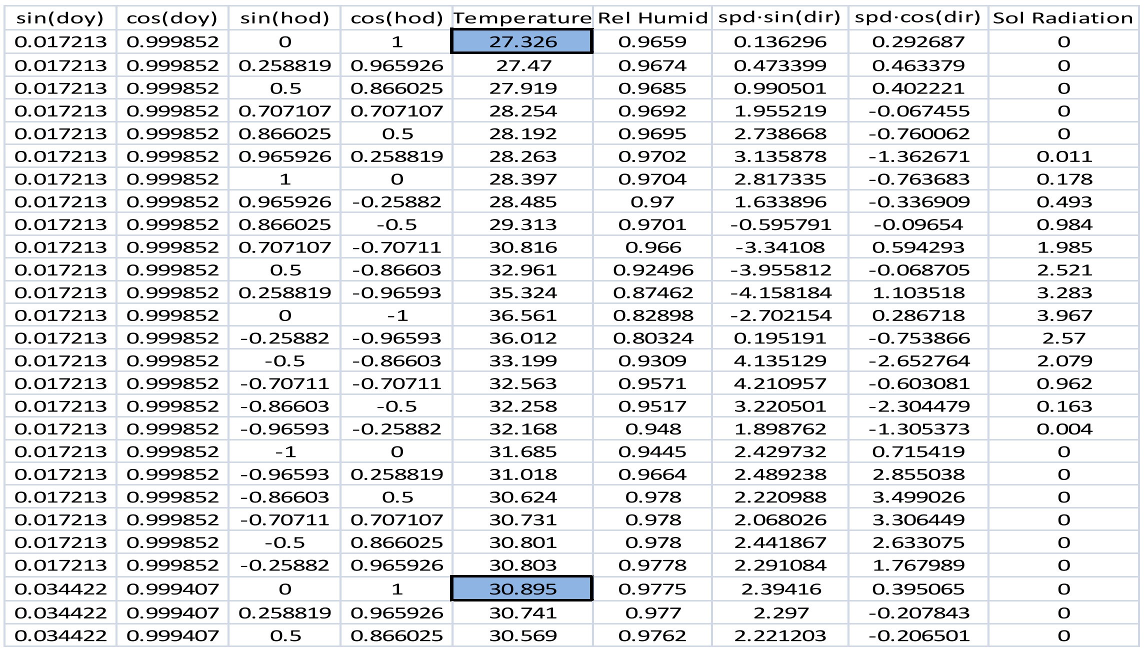
In order to get large number of patterns we have first of all gathered a large number of measurements which are over 6 years of data, as shown in chapter 1 and 2. On an hourly sampling bases, it means over 50000 measurement. We shall use data for almost 2 years for training i.e. almost 16000 measurements in raw data file for training.

Some researchers have used an approach of one neural network/forecaster for every hour of prediction. i.e. the concept has 24 neural network, one trained to predict value of variable for one fixed hour of the day.

This kind of approach has 2 very big drawbacks.

It is firstly assumed that a given variable will be possible to be predicted from the measurement 24 hrs prior to it. This is in general true (as we showed this fact in chapter 2 using the plots of auto-correlation). But this is not a universal truth. Some days weather may change drastically and it will not be possible to predict the correct value only from the 24 hrs previous measurement, but will be possible to predict a better value from 1 hr prior measurements. This will be visible in chapter 5 for all the variables, when we will use time domain feature selection algorithm to reduce the dimension of input space.

Also when we have to train a network for such a method we sample the given time series every 24 hours. For example looking at the raw data file in figure 3.3, wherein the 5th column is for temperature, then if we assume that 1st measurement is for 1200 hrs, then we get the next measurement after 24 measurements. So when we convert this raw data time series into patterns, and if we have data for 2 years, then also we end up getting 365*2 measurements. This means one measurement a day. Now, if we have only 730 measurements and we want to form a network with (say) N=12 inputs and M=1 output then we have only (730-13) =617 patterns, N_v=617 which is very less a number of patterns as compared to using the continuous windowing approach in which we will be able to get (2*365*24)-13 patterns, that we will use for training. So chances of memorization are very high in one forecaster/hr approach for fixed N. Besides in previous approach we have to train 24 networks all with insufficient data, while in our approach in this thesis, we train only one network using enough data. So in previous approach all of networks may show memorization, while in our approach we have very less chances of memorization.



sin(doy)	cos(doy)	sin(hod)	cos(hod)	Temperature	Rel Humid	spd·sin(dir)	spd·cos(dir)	Sol Radiation
0.017213	0.999852	0	1	27.326	0.9659	0.136296	0.292687	0
0.017213	0.999852	0.258819	0.965926	27.47	0.9674	0.473399	0.463379	0
0.017213	0.999852	0.5	0.866025	27.919	0.9685	0.990501	0.402221	0
0.017213	0.999852	0.707107	0.707107	28.254	0.9692	1.955219	-0.067455	0
0.017213	0.999852	0.866025	0.5	28.192	0.9695	2.738668	-0.760062	0
0.017213	0.999852	0.965926	0.258819	28.263	0.9702	3.135878	-1.362671	0.011
0.017213	0.999852	1	0	28.397	0.9704	2.817335	-0.763683	0.178
0.017213	0.999852	0.965926	-0.25882	28.485	0.97	1.633896	-0.336909	0.493
0.017213	0.999852	0.866025	-0.5	29.313	0.9701	-0.595791	-0.09654	0.984
0.017213	0.999852	0.707107	-0.70711	30.816	0.966	-3.34108	0.594293	1.985
0.017213	0.999852	0.5	-0.86603	32.961	0.92496	-3.955812	-0.068705	2.521
0.017213	0.999852	0.258819	-0.96593	35.324	0.87462	-4.158184	1.103518	3.283
0.017213	0.999852	0	-1	36.561	0.82898	-2.702154	0.286718	3.967
0.017213	0.999852	-0.25882	-0.96593	36.012	0.80324	0.195191	-0.753866	2.57
0.017213	0.999852	-0.5	-0.86603	33.199	0.9309	4.135129	-2.652764	2.079
0.017213	0.999852	-0.70711	-0.70711	32.563	0.9571	4.210957	-0.603081	0.962
0.017213	0.999852	-0.86603	-0.5	32.258	0.9517	3.220501	-2.304479	0.163
0.017213	0.999852	-0.96593	-0.25882	32.168	0.948	1.898762	-1.305373	0.004
0.017213	0.999852	-1	0	31.685	0.9445	2.429732	0.715419	0
0.017213	0.999852	-0.96593	0.258819	31.018	0.9664	2.489238	2.855038	0
0.017213	0.999852	-0.86603	0.5	30.624	0.978	2.220988	3.499026	0
0.017213	0.999852	-0.70711	0.707107	30.731	0.978	2.068026	3.306449	0
0.017213	0.999852	-0.5	0.866025	30.801	0.978	2.441867	2.633075	0
0.017213	0.999852	-0.25882	0.965926	30.803	0.9778	2.291084	1.767989	0
0.034422	0.999407	0	1	30.895	0.9775	2.39416	0.395065	0
0.034422	0.999407	0.258819	0.965926	30.741	0.977	2.297	-0.207843	0
0.034422	0.999407	0.5	0.866025	30.569	0.9762	2.221203	-0.206501	0

Figure 4.2 Figure for showing the 24 hrs period between 2 measurements

The second problem with one forecaster/hr approach is that, it predicts the value for each hour independently from the value of measurement one hour prior. On regular days this may be fine. But in cases of urgencies like rain and storms, it may be possible to predict such events more correctly from the measurements of just previous few hours rather then from values of the variable 24 hrs before. Intuitively, just because it did not rain yesterday does not mean it will not rain today or tomorrow. In terms of neural networks, we will observe this thing after we form time-domain training file. What we observe is, when we apply time-domain feature selection on the time domain training file, then one of the most important features is that of 24 hrs prior value but sometimes most important feature is the value of just previous hour. So we ought to be using both of these values in order to assure that our predictor can take an account of both of them. Indirectly speaking, the value of variable for next hour is more dependent (non-linearly) on the value of variable just a few hours before, and less dependent on value of measurements 24 hrs before.

Optimal N_h: In order to make the network as small as possible, so that it has a very small upper bound on pattern storage (so that it is just not capable to memorize any patterns) in each iteration we try to find optimal number of hidden units by using one pass validation processing. By passing the validation data through the network after each iteration of training and by ordering the hidden units in the OWO-OLS we find out the most significant hidden units of the network, thereby find the optimal value of N_h.

Smallest N: Thinner training patterns i.e. have smallest N possible as compared to N_v is the second precaution that we take in order to avoid memorization. We do this using 2-stage feature selection and use only the most important inputs and that too after compression, reducing the N by atleast 60%. In the next chapter we shall elaborate this process of reducing the size of input space N.

CHAPTER 5

TIME DOMAIN TRAINING FILES AND 2-STAGE FEATURE SELECTION

In this chapter, section 5.1 we shall explain some problems faced in training the network in time domain. Even after time domain feature selection we would prefer to transform the data in KLT domain. This is described in detail in this section 5.2. Also we will describe the format of input pattern with which we shall train a Multivariable neural network in 5.1. Each of the variables will have its own KLT matrix stored in the memory or calculated from the SVD of covariance matrix. This will be detailed in this section. Section 5.3 will contain some preview and theory of PLN based time domain feature selection. For details on PLN type training, readers are referred to [6]. This section will also refer to the lab Software Numap with relevant version [15]. More importantly each variable will have its own sequence of important time domain inputs. This is where we will show that 'time' itself as an input is an important feature for all the variables. Section 5.3 accounts for the reduction of size of input pattern achieved after both the feature selection are completed. We shall show in this section for each variable that there is hardly any impact on the prediction error due to the feature selection, though the size of network is reduced drastically, thus serving our purpose. After certain number of important inputs in time domain and KLT domain are taken in for training, using the last certain number of features will not help us reduce the prediction error, but unnecessarily increases the size of network. In section 5.4 we shall show that the prediction capability of our network in limited. As we try to predict values further in future the accuracy gets worsened. We shall show this by comparative plots and figures.

5.1 Time-Domain Training File Formats

In this section we shall first of show and elaborate on the raw data file and all the time series in it.Then we will elaborate on the formation of the time-domain training files obtained out of the raw data file by different methods. Basically we are running this forecaster in multiple ways.

- First method is wherein we predict individual time series independently.
- Second method is wherein we have to compulsorily process the neural network using 2 variables, or 3. This method is mainly meant for wind variable. Wind speed and direction are 2 time series in the raw data file. We form speed·sin(direction) and speed·cos(direction) out of it, which are again 2 time series. So we need to process them together always because they cannot be independently treated. Again in treating the wind alone we can keep 3 variables to be speed, sin(direction) and cos(direction) as in figure 5.3. We cannot use 3 variables as speed, speed·sin(direction), and speed·cos(direction) because that will make the inputs linearly dependent. Effects of such linear dependency are discussed in [3], [4] and [5].
- The third way we use this forecaster is the novel way that we propose for prediction of weather related variables. In this method we treat all 5 time series together. We form patterns out of each spatial variable in the raw data file and keep augmenting the training pattern. Thereby we end up getting very large patterns. But it is shown in chapter 2 that these variables are definitely dependent on each other. So the quality of forecast obtained by such a combined network is much better. Better in the way that the total training or validation error obtained by predicting the individual time series and summing them is more than the error obtained from this combined network summed over all the outputs, thereby proving our purpose.

After we have discussed everything about the formation of the time-domain training file, we shall turn to the PLN type feature selection [34]. We shall try to give a preview of our lab software ‘Numap’ that will give us a sequence of important features in order of their contribution toward the error function. We call this ‘time domain feature selection’ which drastically reduces the dimension of input space for training our forecaster especially important for the third way of running this forecaster i.e. the combined version. Then we shall elaborate on the KLT domain feature selection and site some references on the Singular value based feature selection that other researchers have used to select the number of rows to be used from the orthogonal matrix, again reducing down the size of training file after this second stage of feature selection. Finally we shall show how as we try to predict the values more further in future we get deteriorated performance due to increase in the number of outputs that the network has to solve for using the OLS. As a remedy to this we shall then show how can we keep predicting more values further in future by skipping directly to the hour that we want to predict, thereby keeping the M as low as possible.

For this section we shall give a detailed account on format of the initial time domain pattern files for all variables to be predicted and for all 3 above type of applications of prediction.

First of all as a reminder, we will be using the constant ‘1’ and 4 time related variables mentioned as in chapter 2 in the pattern file for all the variables. We shall show the initial time domain pattern file graphically here for ease of understanding. We shall consider in these images a pattern as a row vector and as we travel below it will be a number of patterns i.e. 1 to Nv. We shall now put the images of these files that we obtain from the raw data file by windowing. For simplicity of understanding we shall show this for smaller dimensions of the patterns and only 2 patterns will be formed.

Figure 5.1 shows the raw data file for individual time series prediction. As shown the 5.2 uses the windowing method to form the initial time domain pattern file for the first case of predictor. If we wish to train the network using this pattern form itself, ‘1’ has to be augment at the extreme left column.

For the second case wherein we have to deal with the 2 variables simultaneously then we need to use two time series from the raw data file simultaneously. We shall show a separate wind raw data file and then show 2 patterns formed (in figure 4.4 the first 19 cells are the inputs and last 9 are the outputs of the neural net.

sin(doy)	cos(doy)	sin(hod)	cos(hod)	Temperature	Humidity	speed·sin(dir)	speed·cos(dir)	Solar Radiation
0.017213	0.999852	0	1	$v_1(i-6)$	0.9659	0.136296	0.292687	0
0.017213	0.999852	0.258819	0.965926	$v_1(i-5)$	0.9674	0.473399	0.463379	0
0.017213	0.999852	0.5	0.866025	$v_1(i-4)$	0.9685	0.990501	0.402221	0
0.017213	0.999852	0.707107	0.707107	$v_1(i-3)$	0.9692	1.955219	-0.067455	0
0.017213	0.999852	0.866025	0.5	$v_1(i-2)$	0.9695	2.738668	-0.760062	0
0.017213	0.999852	0.965926	0.258819	$v_1(i-1)$	0.9702	3.135878	-1.362671	0.011
$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_1(i)$	0.9704	2.817335	-0.763683	0.178
0.017213	0.999852	0.965926	-0.25882	$v_1(i+1)$	0.97	1.633896	-0.336909	0.493
0.017213	0.999852	0.866025	-0.5	$v_1(i+2)$	0.9701	-0.595791	-0.09654	0.984
0.017213	0.999852	0.707107	-0.70711	$v_1(i+3)$	0.966	-3.34108	0.594293	1.985
0.017213	0.999852	0.5	-0.86603	32.961	0.92496	-3.955812	-0.068705	2.521
0.017213	0.999852	0.258819	-0.96593	35.324	0.87462	-4.158184	1.103518	3.283
0.017213	0.999852	0	-1	36.561	0.82898	-2.702154	0.286718	3.967

Figure 5.1 Windowing for single variable pattern formation

$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_{11}(i-6)$	$v_{11}(i-5)$	$v_{11}(i-4)$	$v_{11}(i-3)$	$v_{11}(i-2)$	$v_{11}(i-1)$	$v_{11}(i)$	$v_{11}(i+1)$	$v_{11}(i+2)$	$v_{11}(i+3)$
$tim_2(1)$	$tim_2(2)$	$tim_2(3)$	$tim_2(4)$	$v_{12}(i-6)=v_{11}(i-5)$	$v_{12}(i-5)=v_{11}(i-4)$	$v_{12}(i-4)=v_{11}(i-3)$	$v_{12}(i-3)=v_{11}(i-2)$	$v_{12}(i-2)=v_{11}(i-1)$	$v_{12}(i-1)=v_{11}(i)$	$v_{12}(i)=v_{11}(i+1)$	$v_{12}(i+1)=v_{11}(i+2)$	$v_{12}(i+2)=v_{11}(i+3)$	$v_{12}(i+3=v_{11}(i+4)$

Figure 5.2 Pattern construction for single variable forecasting

$$\begin{aligned} \mathbf{x}_1&=[tim_1(1)\ldots.tim_1(4):v_{11}(i-6)\ldots.v_{11}(i)],\ \mathbf{t}_1=[v_{11}(i+1)\ldots.v_{11}(i+3)] \\ \mathbf{x}_2&=[tim_2(1)\ldots.tim_2(4):v_{12}(i-6)\ldots.v_{12}(i)],\ \mathbf{t}_2=[v_{12}(i+1)\ldots.v_{12}(i+3)] \end{aligned} \tag{73}$$

sin(doy)	cos(doy)	sin(hod)	cos(hod)	Speed	sin(dir)	cos(dir)
0.017213	0.999852	0.707107	0.707107	1.362	0.214895	0.10007
0.017213	0.999852	0.866025	0.5	$v_{11}(i-4)$	$v_{21}(i-4)$	$v_{31}(i-4)$
0.017213	0.999852	0.965926	0.258819	$v_{11}(i-3)$	$v_{21}(i-3)$	$v_{31}(i-3)$
0.017213	0.999852	1	0	$v_{11}(i-2)$	$v_{21}(i-2)$	$v_{31}(i-2)$
0.017213	0.999852	0.965926	-0.25882	$v_{11}(i-1)$	$v_{21}(i-1)$	$v_{31}(i-1)$
$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_{11}(i)$	$v_{21}(i)$	$v_{31}(i)$
0.017213	0.999852	0.707107	-0.70711	$v_{11}(i+1)$	$v_{21}(i+1)$	$v_{31}(i+1)$
0.017213	0.999852	0.5	-0.86603	$v_{11}(i+2)$	$v_{21}(i+2)$	$v_{31}(i+2)$
0.017213	0.999852	0.258819	-0.96593	$v_{11}(i+3)$	$v_{21}(i+3)$	$v_{31}(i+3)$
0.017213	0.999852	0	-1	5.448	0.109085	-0.61327

Figure 5.3 Windowing for multi-variable pattern formation for wind prediction

$$\begin{aligned}\mathbf{x}_1&=[tim_1(1)\dots tim_1(4):v_{11}(i\text{-}4)\dots v_{11}(i):v_{21}(i\text{-}4)\dots v_{21}(i):v_{31}(i\text{-}4)\dots v_{31}(i)]\\ \mathbf{t}_1&=[v_{11}(i+1)\dots v_{11}(i+3):v_{21}(i+1)\dots v_{21}(i+3):v_{31}(i+1)\dots v_{31}(i+3)]\\ \mathbf{x}_2&=[tim_2(1)\dots tim_2(4):v_{12}(i\text{-}4)\dots v_{12}(i):v_{22}(i\text{-}4)\dots v_{22}(i):v_{32}(i\text{-}4)\dots v_{32}(i)]\\ \mathbf{t}_2&=[v_{12}(i+1)\dots v_{12}(i+3):v_{22}(i+1)\dots v_{22}(i+3):v_{32}(i+1)\dots v_{32}(i+3)]\end{aligned}\tag{74}$$

For the third form of the predictor we will show the raw data file but for the pattern we will give a vector notation only because we cannot put the whole pattern into the image in the document without reducing the clarity. Essentially the patterns for this case are formed in the same manner as in figure 5.3 and 5.4 except for the fact that now there will be total 9 columns in figure 5.3 and windows are formed out of all 5 columns on the right (spatial variables). For this reason for example, if we intend to use values of 5 previous hours and predict values for 3 hours in future for each variable then the pattern will look as below. We shall use notation 'tim' for the row vector (1 X 4) of time information and v_{1p}, v_{2p}, v_{3p}, v_{4p}, v_{5p} for spatial variables having past information and v_{1f}, v_{2f}, v_{3f}, v_{4f}, v_{5f} for vectors of desired output. Pattern vector of dimension 1 X 44 (say 'P') will look as follows-

$$\begin{aligned}P=[&tim(1\text{ X }4):v_{1p}(1\text{ X }5):v_{2p}(1\text{ X }5):v_{3p}(1\text{ X }5):v_{4p}(1\text{ X }5):v_{5p}(1\text{ X }5):\\ &v_{1f}(1\text{ X }3):v_{2f}(1\text{ X }3):v_{3f}(1\text{ X }3):v_{4f}(1\text{ X }3):v_{5f}(1\text{ X }3)].\end{aligned}\tag{75}$$

For the training purpose and in the theory of chapter 3 we have called- $[tim(1X4):v_{1p}(1X5):v_{2p}(1X5):v_{3p}(1X5):v_{4p}(1X5):v_{5p}(1X5)]$ this section of 1X29 as '$\mathbf{x_p}$' and the output part $[v_{1f}(1X3):v_{2f}(1X3):v_{3f}(1X3):v_{4f}(1X3):v_{5f}(1X3)]$ of dimension 1X15 is called '$\mathbf{t_p}$', the desired response. $[\mathbf{x_p}:\mathbf{t_p}]^T$ forms a training pattern.

These 3 formats of time domain pattern formats are the general formats. But as is seen already, the dimension of these patterns keep on increasing if we want to use data over more hours in past for predicting more hours in future. In practical cases to get good accuracy we encountered problems with input **x** of dimensions of 80s and output dimension of 30s. For this reason of course we are going to use feature selection in 2 stages which we will show in the next sections but there is a way to reduce number of outputs here itself. There is nothing that says that if we want to predict the value of variable 8 hours in future than we need to predict values for all the hours in future from 1 to 8. We can instead directly solve for the hour that we wish to predict skipping the values in between. Like as mentioned in chapter 3 section 3.1 and 3.2, by doing this we will reduce number of outputs in the neural network and more essentially OLS will have less number of values to solve for. This will enhance the performance of OLS and reduce the approximation error drastically as M itself is reduced to 1. The pattern formation for this case is shown below-

sin(doy)	cos(doy)	sin(hod)	cos(hod)	Temperature	Humidity	speed·sin(dir)	speed·cos(dir)	Solar Radiation
0.017213	0.999852	0	1	$v_{11}(i-6)$	0.9659	0.136296	0.292687	0
0.017213	0.999852	0.258819	0.965926	$v_{11}(i-5)$	0.9674	0.473399	0.463379	0
0.017213	0.999852	0.5	0.866025	$v_{11}(i-4)$	0.9685	0.990501	0.402221	0
0.017213	0.999852	0.707107	0.707107	$v_{11}(i-3)$	0.9692	1.955219	-0.067455	0
0.017213	0.999852	0.866025	0.5	$v_{11}(i-2)$	0.9695	2.738668	-0.760062	0
0.017213	0.999852	0.965926	0.258819	$v_{11}(i-1)$	0.9702	3.135878	-1.362671	0.011
$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_{11}(i)$	0.9704	2.817335	-0.763683	0.178
0.017213	0.999852	0.965926	-0.25882	28.485	0.97	1.633896	-0.336909	0.493
0.017213	0.999852	0.866025	-0.5	29.313	0.9701	-0.595791	-0.09654	0.984
0.017213	0.999852	0.707107	-0.70711	30.816	0.966	-3.34108	0.594293	1.985
0.017213	0.999852	0.5	-0.86603	32.961	0.92496	-3.955812	-0.068705	2.521
0.017213	0.999852	0.258819	-0.96593	$v_{11}(i+5)$	0.87462	-4.158184	1.103518	3.283
0.017213	0.999852	0	-1	36.561	0.82898	-2.702154	0.286718	3.967

Figure 5.4 Windowing for single output forecaster for single variable

$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_{11}(i-6)$	$v_{11}(i-5)$	$v_{11}(i-4)$	$v_{11}(i-3)$	$v_{11}(i-2)$	$v_{11}(i-1)$	$v_{11}(i)$	$v_{11}(i+5)$

Figure 5.5 Pattern construction for single variable forecasting with 1 output only

$$\mathbf{x}_1=[tim_1(1)\ldots.tim_1(4):v_{11}(i-6)\ldots.v_{11}(i)],\ \ \mathbf{t}_1=[v_{11}(i+5)]$$
$$\mathbf{x}_2=[tim_2(1)\ldots.tim_2(4):v_{12}(i-6)\ldots.v_{12}(i)],\ \ \mathbf{t}_2=[v_{12}(i+5)=v_{11}(i+6)] \qquad (76)$$

The only disadvantage in this case is that we have to train the forecaster separately for each hour. But we use the same neural network. Of course, over this time domain pattern we will still apply two stage feature selection that we will show in next sections.

Before we go to the next section we must say a few words about the combined forecaster that we are proposing in this thesis. The concept of combined forecaster is a multivariable approach of dealing with the forecaster. It is based on the facts as we presented in chapter 2. The fact that all weather variables are correlated not just with their own selves but also on each other makes it sensible to approach them in this way in order to get a better result. Also we are going to use the same approach to detect the bad measurements in the raw data in case of bad measurement. In essence we try to estimate the missing values of the variable based on its relationship with the other variables. This part of our work is under progress. Another important point to be attended is that as we try to predict more hours in future the prediction accuracy goes on deteriorating. After prediction for over 48 hours the prediction is not so worth looking at.

Thereby we shall now move towards time-domain feature selection theory and then the KLT transform based feature selection theory.

5.2 Tranform Based Feature Selection OR Compression

The feature selection approach for the neural network training is a necessity in many applications. The reason for this is that time-domain processing is not feasible in all applications. Karhunen Loeve Transform chosen because it gives optimal compression and reconstruction. One important thing for us in this application is the sequence of the steps. In brief we will have the sequence wherein we will apply the MA on time-series. Then we shall form the patterns using the windowing approach given in section 5.1. After this we would use the mean removal or separating mean on this time-domain pattern files given in section 2.2. Then we would use the time-domain feature selection using the PLN approach to get a thinner time-domain training file. After this we will use KLT transform based feature selection in order to get even a thinner file. The term ‘Thinner training file’ is being coined to indirectly represent the ratio of N/N_v. Thinner file means smaller ratio. Most researcher use DCT transform because of the existence of the fast algorithms already developed by researchers [35], [36]. But the KLT is a signal dependent transform and so it changes with the type of signals. But KLT has no fast algorithms developed yet. We shall not worry about the performance index of the KLT transforms in this thesis. For this, readers can go to [35], [37].

Now we shall elaborate the KLT domain feature selection approach and the KLT transform for this feature selection. There KLT transform basically is considered and optimal transform. The main reason for this is that it givens perfect reconstruction. Concept of KLT is based on energy of the signal. This KLT is applied on the Covariance or the correlation matrix depending on the variance of the signal. The correlation calculation equation was given in chapter 2. But here we shall elaborate the matrix approach of this so that we can explain the KLT compression.

The correlation matrix and subsequent covariance matrix calculation is displayed below using the expectation operator 'E[]'. '**R**' is being used to show the correlation matrix. The '$\boldsymbol{\sigma}$' is used to show the covariance matrix. The KLT transform which indirectly uses the SVD of the matrix actually diagonalizes the matrix. This means that it decouples the various inputs from each other. Thereby the unitary matrix obtained from the KLT or SVD are either row or column orthogonal. There orthogonality of the matrix is useful for the compression of the signal because it basically will give us coefficients which are independent of each other. To simplify, basically the KLT projects the input pattern of dimension N into an orthogonal space of N dimension which are de-correlated and decoupled.

The other factor that plays major role in the KLT compression is the dimension reduction. The KLT uses the energy approach. This means that the KLT transform basically concentrates the energy of the signal into a few top most coefficients. The coefficient at the trailing end go smaller and smaller. Thereby for compression we may use the top few orthogonal basis function (which may be rows or columns) in order to compress the signal into a smaller dimension. In this case, we train the neural network with these coefficients of the input and output space and then reconstruct the signal into original time domain by using the inverse KLT matrix of the output covariance/correlation.

The one very well known KLT transform is using the SVD of the correlation matrix. Assuming the correlation matrix of the pattern $\mathbf{x}_p$ of dimension N, $\mathbf{x}_p=[x_1,x_2,x_3,..x_{N-1},x_N]^T$ we get the N X N correlation matrix. We only consider the real values for all the variables.

$$R=E\left[\mathbf{x}_p\mathbf{x}_p^T\right]=E\left[\begin{bmatrix}x_1\\ \vdots\\ x_N\end{bmatrix}\cdot[x_1,\ldots,x_N]\right] \tag{77}$$

$$=E\begin{bmatrix}x_1x_1 & x_1x_2 & \cdots & x_1x_N\\ x_2x_1 & x_2x_2 & & x_2x_N\\ \vdots & & \ddots & \vdots\\ x_Nx_1 & x_Nx_2 & \cdots & x_Nx_N\end{bmatrix}=\begin{bmatrix}E[x_1x_1] & E[x_1x_2] & \cdots & E[x_1x_N]\\ E[x_2x_1] & E[x_2x_2] & & E[x_2x_N]\\ \vdots & & \ddots & \vdots\\ E[x_Nx_1] & E[x_Nx_2] & \cdots & E[x_Nx_N]\end{bmatrix} \tag{78}$$

If we consider the mean removed or mean separated values then the correlation matrix or the covariance matrix will be derived as follows with $\mathbf{m}_x=[m_1,m_2, \ldots.m_N]$-

$$\boldsymbol{\sigma}=E\left[(\mathbf{x}_p-\mathbf{m}_x)(\mathbf{x}_p-\mathbf{m}_x)^T\right]=E\left[\begin{bmatrix}x_1-m_1\\ \vdots\\ x_N-m_N\end{bmatrix}[x_1-m_1,\ldots,x_N-m_N]\right] \tag{79}$$

$$=E\begin{bmatrix}(x_1-m_1)(x_1-m_1) & (x_1-m_1)(x_2-m_2) & \cdots & (x_1-m_1)(x_N-m_N)\\ (x_2-m_2)(x_1-m_1) & (x_2-m_2)(x_2-m_1) & & (x_2-m_2)(x_N-m_N)\\ \vdots & & \ddots & \vdots\\ (x_N-m_N)(x_1-m_1) & (x_N-m_N)(x_2-m_2) & \cdots & (x_N-m_N)(x_N-m_N)\end{bmatrix} \tag{80}$$

$$=\begin{bmatrix}E[\sigma_1\sigma_1] & E[\sigma_1\sigma_2] & \cdots & E[\sigma_1\sigma_N]\\ E[\sigma_2\sigma_1] & E[\sigma_2\sigma_2] & & E[\sigma_2\sigma_N]\\ \vdots & & \ddots & \vdots\\ E[\sigma_N\sigma_1] & E[\sigma_N\sigma_2] & \cdots & E[\sigma_N\sigma_N]\end{bmatrix}=\begin{bmatrix}\sigma_{11} & \sigma_{12} & \cdots & \sigma_{1N}\\ \sigma_{21} & \sigma_{22} & & \sigma_{2N}\\ \vdots & & \ddots & \vdots\\ \sigma_{N1} & \sigma_{N2} & \cdots & \sigma_{NN}\end{bmatrix} \tag{81}$$

In order to obtain the KLT matrix we perform the singular value decomposition of this matrix which is given next. The aim is to get an orthogonal transformation matrix T such that $\mathbf{z}_{pi}=\mathbf{T}_i\mathbf{x}_p$, and $\mathbf{z}_{po}=\mathbf{T}_o\mathbf{t}_p$ so that the dimension of $\mathbf{z}_{pi}$ is less than N (N is the dimension of the input pattern after the time domain feature selection using the PLN) and $\mathbf{z}_{po}$ has dimension less than M. But $\widehat{\mathbf{x}_p}$ can be reconstructed from the $\mathbf{z}_{pi}$ with very good accuracy and $\widehat{\mathbf{t}_p}$ can be reconstructed from

$\mathbf{z_{po}}$. The crux here is that instead of $\mathbf{z_{po}}$,output of the neural network $\mathbf{y_p}=\widehat{\mathbf{z_{po}}}$ after the training is to be used to reconstruct i.e. $\mathbf{t_p}=[\mathbf{T_o}]^{-1}\,\widehat{\mathbf{z_{po}}}$of course, and $\widehat{\mathbf{t_p}}= [\mathbf{T_o}]^{-1}\,\mathbf{y_p}$. Thereby the training pattern $\{\mathbf{x_p},\mathbf{t_p}\}$ is transformed to $\{\mathbf{z_{pi}},\mathbf{z_{po}}\}$, using which the training of the neural network is done and after the training is completed then that output $\widehat{\mathbf{z_{po}}}$ is brought back in the time domain using the $[\mathbf{T_o}]^{-1}\cdot\widehat{\mathbf{z_{po}}}$

This implies that the input and the output of the neural network are transformed using separate KLT matrices. Now the SVD of the matrix-

$$\mathbf{R=U\cdot\Sigma\cdot V^T} \tag{82}$$

$$\mathbf{U^T\cdot R\cdot V=\Sigma} \tag{83}$$

The decomposition gives U and V matrices which satisfy the property as follows-

$$\sum_{i=1}^{N} U(i,j)U(i,k)=\delta_{jk}\ ;1\le j\le N,\ 1\le k\le N \tag{84}$$

$$\sum_{i=1}^{N} V(i,j)V(i,k)=\delta_{jk}\ ;1\le j\le N,\ 1\le k\le N \tag{85}$$

In matrix notation, using the orthogonality inverse is simply the transpose-

$$\mathbf{V^TV=U^TU=I=U\cdot U^T=V\cdot V^T} \tag{86}$$

Also since the covariance and the correlation matrices are symmetric- **U=V.**

The matrix $\mathbf{\Sigma}$ is the diagonal matrix – $[\Sigma_1,\Sigma_2,\ldots\Sigma_N]$ The application we are concerned with has only square matrices, so all the matrices concerned will have either N X N dimension of M X M dimension depending on the input pattern or the output pattern.

Now considering the eigenvector approach of this matrix expression.

$$\mathbf{U^T RV = \Sigma}$$

$$\mathbf{U^{-1} RV = \Sigma}$$

$$\mathbf{RV = U\Sigma} \tag{87}$$

U = [U$_1$, U$_2$, U$_3$…U$_N$] is column orthogonal matrix whose columns are orthogonal as mentioned before. Each column is of dimension N or M depending on whether the input or output is being compressed. The RHS of the last equation can be expressed as –

$$\begin{bmatrix} U_{11} & U_{21} & \cdots & U_{N1} \\ U_{12} & U_{22} & \cdots & U_{N2} \\ \vdots & & \ddots & \vdots \\ U_{1N} & U_{2N} & \cdots & U_{NN} \end{bmatrix} \begin{bmatrix} \Sigma_{11} & 0 & \cdots & 0 \\ 0 & \Sigma_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{NN} \end{bmatrix} = \left[\Sigma_{11} \begin{bmatrix} U_{11} \\ U_{12} \\ \vdots \\ U_{1N} \end{bmatrix} \quad \cdots\cdots \quad \Sigma_{NN} \begin{bmatrix} U_{N1} \\ U_{N2} \\ \vdots \\ U_{NN} \end{bmatrix} \right] \tag{88}$$

So here we call $\mathbf{U} = \mathbf{T}$ or the KLT matrix. $\mathbf{z_{pi}} = \mathbf{U^{-1}x_p} = \mathbf{U^{T}x_p}$, $\mathbf{z_{po}}=\mathbf{U^{-1}t}_p = \mathbf{U^{T}t_p}$ is called the forward transform for either the input pattern or the output pattern whichever is concerned,only fact is that the KLT for input and output will be different. This transformation of the input pattern or the output pattern results into a vector of length N or M respectively. But observing this vector we will be able to see that only first few elements of this transformed vector are large, while the ones at the trailing end are almost very close to zero. This is the concept of energy being concentrated into first few elements or the so called coefficients.

$$\mathbf{z_{pi}=U^T x_p=[U_1:U_2:U_3:\ldots:U_N]^T \cdot x_p=[}z_{\mathbf{pi1}}:z_{\mathbf{pi2}}:z_{\mathbf{pi3}}:z_{\mathbf{pi4}}\ldots:z_{\mathbf{piN}}\mathbf{]} \tag{89}$$

As seen z_{pi1},z_{pi2} and few later elements will have respectable magnitude. The trailing elements will be very small (close towards $z_{piN)}$. This is the concept of the KLT feature selection. We only use first few elements of vector $\mathbf{z_{pi}}$ thereby calling it the new input pattern. In the same way output patterns are also formed of the dimension smaller than the original M. The justification for doing this can be the assumption that the trailing coefficients represent the energy of the noise.

Now the question arises about how many columns of **U** matrix to be used i.e. how many elements of $\mathbf{z}_{pi}$ should be used for the forward and reverse transform case in order to get a perfect reconstruction. One approach to this question is to look at the singular values of the diagonal matrix **Σ.** A lot of researchers who has used QR decomposition have looked in this approach and used it for selecting the number of columns to be used [23], [24], [25] has used the ratio of two consecutive values for selecting this number. Another approach to this is to use energy approach. In this case we use the squared summation of the $\mathbf{z}_{pi}$ vector over the whole training file (p from 1 to N_v) and choose those indices which have maximum values.

$$\sum_{p}^{Nv} z(p,i)^2 = \mathbf{z}_i \tag{90}$$

Here $\mathbf{z}_i$ is the vector of the summation of square of the $\mathbf{z}_{pi}$ over whole training file. This way of feature selection also gives good results. Either ways, choosing first half of the $\mathbf{z}_{pi}$ coefficients will always suffice. This will be the worst case though. In either of the case the dimension of the input pattern is drastically reduced and the training gets speed up. For further reading on the KLT application and theory readers are referred to [35], [37].

One most important point need be mentioned. The KLT matrix of the individual variable will be unique. So in the case of the combined forecaster of case 3 of section 5.1 we will need to have 5 separate KLT matrices; one for each variable. Again 5 separate KLT matrices for the output also will be needed. Thereby we will have to store 10 KLT matrices in the memory. This is one complicated process, so we will elaborate this diagrammatically.

Using the same raw data format (figure 5.6) to pattern format (figure-5.7) we show the raw data format and using the windowing over that we get the pattern file as in figure 5.7. Assume for now that this is the pattern file obtained after the time-domain feature selection using the PLN that we will describe in next section. Then, after KLT feature selection on the input pattern wherein we use first 2, 2, 3, 2, 2 features for v_1, v_2, v_3, v_4, v_5 respectively will look as in figure 5.8. The same concept will apply for the output pattern also but for space constrain we have only shown one output for each variable, so the KLT feature selection will not reduce the dimension but the transformed coefficients are used still. In such cases wherein we want to predict only one value in future for each variable we need not use the KLT compression. As such the feature selection process are usually meant only for the input of the neural network.

sin(doy)	cos(doy)	sin(hod)	cos(hod)	Temperature	Humidity	speed·sin(dir)	speed·cos(dir)	Solar Radiation
0.017213	0.999852	0	1	27.326	0.9659	0.136296	0.292687	0
0.017213	0.999852	0.258819	0.965926	27.47	0.9674	0.473399	0.463379	0
0.017213	0.999852	0.5	0.866025	27.919	$v_{21}(i-4)$	0.990501	0.402221	0
0.017213	0.999852	0.707107	0.707107	$v_{11}(i-3)$	$v_{21}(i-3)$	$v_{31}(i-3)$	$v_{41}(i-3)$	$v_{51}(i-3)$
0.017213	0.999852	0.866025	0.5	$v_{11}(i-2)$	$v_{21}(i-2)$	$v_{31}(i-2)$	$v_{41}(i-2)$	$v_{51}(i-2)$
0.017213	0.999852	0.965926	0.258819	$v_{11}(i-1)$	$v_{21}(i-1)$	$v_{31}(i-1)$	$v_{41}(i-1)$	$v_{51}(i-1)$
$tim_1(1)$	$tim_1(2)$	$tim_1(3)$	$tim_1(4)$	$v_{11}(i)$	$v_{21}(i)$	$v_{31}(i)$	$v_{41}(i)$	$v_{51}(i)$
0.017213	0.999852	0.965926	-0.25882	28.485	0.97	1.633896	-0.336909	0.493
0.017213	0.999852	0.866025	-0.5	29.313	0.9701	-0.595791	-0.09654	0.984
0.017213	0.999852	0.707107	-0.70711	$v_{11}(i+3)$	$v_{21}(i+3)$	-3.34108	$v_{41}(i+3)$	$v_{51}(i+3)$
0.017213	0.999852	0.5	-0.86603	32.961	0.92496	$v_{31}(i+4)$	-0.068705	2.521
0.017213	0.999852	0.258819	-0.96593	35.324	0.87462	-4.158184	1.103518	3.283

Figure 5.6 Windowing for multi-variable forecaster with one output for each variable

$t_1(1)$	$t_1(2)$	$t_1(3)$	$t_1(4)$	$v_{11}(i-3)$	$v_{11}(i-2)$	$v_{11}(i-1)$	$v_{11}(i)$	$v_{21}(i-4)$	$v_{21}(i-3)$	$v_{21}(i-2)$	$v_{21}(i-1)$	$v_{21}(i)$	$v_{31}(i-3)$	$v_{31}(i-2)$	$v_{31}(i-1)$	$v_{31}(i)$	$v_{41}(i-3)$	$v_{41}(i-2)$	$v_{41}(i-1)$	$v_{41}(i)$	$v_{51}(i-3)$	$v_{51}(i-2)$	$v_{51}(i-1)$	$v_{51}(i)$	$v_{11}(i+3)$	$v_{21}(i+3)$	$v_{31}(i+4)$	$v_{41}(i+3)$	$v_{51}(i+3)$
$t_2(1)$	$t_2(2)$	$t_2(3)$	$t_2(4)$	$v_{12}(i-3)$	$v_{12}(i-2)$	$v_{12}(i-1)$	$v_{12}(i)$	$v_{22}(i-4)$	$v_{22}(i-3)$	$v_{22}(i-2)$	$v_{22}(i-1)$	$v_{22}(i)$	$v_{32}(i-3)$	$v_{32}(i-2)$	$v_{32}(i-1)$	$v_{32}(i)$	$v_{42}(i-3)$	$v_{42}(i-2)$	$v_{42}(i-1)$	$v_{42}(i)$	$v_{52}(i-3)$	$v_{52}(i-2)$	$v_{52}(i-1)$	$v_{52}(i)$	$v_{12}(i+3)$	$v_{22}(i+3)$	$v_{32}(i+4)$	$v_{41}(i+4)$	$v_{52}(i+3)$

Figure 5.7 Time-domain pattern for multivariable forecaster for one output for each variable

0.017213	0.999852	1	0	$z_{1i}^{v1}(1)$	$z_{1i}^{v1}(2)$	$z_{1i}^{v2}(1)$	$z_{1i}^{v2}(2)$	$z_{1i}^{v2}(3)$	$z_{1i}^{v3}(1)$	$z_{1i}^{v3}(2)$	$z_{1i}^{v4}(1)$	$z_{1i}^{v4}(2)$	$z_{1i}^{v5}(1)$	$z_{1i}^{v5}(2)$	$z_{1o}^{v1}(1)$	$z_{1o}^{v2}(1)$	$z_{1o}^{v3}(1)$	$z_{1o}^{v4}(1)$	$z_{1o}^{v5}(1)$
0.017213	0.999852	0.965926	-0.25882	$z_{2i}^{v1}(1)$	$z_{2i}^{v1}(2)$	$z_{2i}^{v2}(1)$	$z_{2i}^{v2}(2)$	$z_{2i}^{v2}(2)$	$z_{2i}^{v3}(1)$	$z_{2i}^{v3}(2)$	$z_{2i}^{v4}(1)$	$z_{2i}^{v4}(2)$	$z_{2i}^{v5}(1)$	$z_{2i}^{v5}(2)$	$z_{2o}^{v1}(1)$	$z_{2o}^{v2}(1)$	$z_{2o}^{v3}(1)$	$z_{2o}^{v4}(1)$	$z_{2o}^{v5}(1)$

Figure 5.8 Compressed pattern for multi-variable forecaster for one output for each variable

Each compressed pattern on figure 5.8 is obtained using separate KLT. Storing the KLT transform matrix for each variable is one of the drawbacks of this method. Before compression pattern looks as follows (for ‘**x**’ and ‘**t**’ the subscript assigns the pattern number while for ‘**v**’ first subscript represents variable number and second represents pattern number)-

$$\mathbf{x}_1 = [tim_1(1)\ldots tim_1(4): v_{11}(i-3)\ldots v_{11}(i):v_{21}(i-4)\ldots v_{21}(i): v_{31}(i-3)\ldots v_{31}(i):v_{41}(i-3)\ldots v_{41}(i):v_{51}(i-3)\ldots v_{51}(i)]$$
$$\mathbf{t}_1=[v_{11}(i+3):v_{21}(i+3):v_{31}(i+4):v_{41}(i+3):v_{51}(i+3)]$$
$$\mathbf{x}_2 = [tim_2(1)\ldots tim_2(4): v_{12}(i-3)\ldots v_{12}(i):v_{22}(i-4)\ldots v_{22}(i): v_{32}(i-3)\ldots v_{32}(i):v_{42}(i-3)\ldots v_{42}(i):v_{52}(i-3)\ldots v_{52}(i)]$$
$$\mathbf{t}_2=[v_{12}(i+3):v_{22}(i+3):v_{32}(i+4):v_{42}(i+3):v_{52}(i+3)] \quad (91)$$

Post-Compression 1st pattern for this case looks as follows (for ‘**z**’ the first subscript means the pattern number, super script displays the variable number while in subscript I and o means the input part and output part of pattern, the subscript on ‘**P**’ is represents in KLT domain, and index in bracket means the pattern number)–

$$P_z(1)=[tim_1(1 \text{ X } 4):z_{1i}^{(v1)}(1 \text{ X } 2):z_{1i}^{(v2)}(1 \text{ X } 3):z_{1i}^{(v3)}(1 \text{ X } 2):z_{1i}^{(v4)}(1 \text{ X } 2):z_{1i}^{(v5)}(1 \text{ X } 2):$$
$$z_{1o}^{(v1)}(1 \text{ X } 1):z_{1o}^{(v2)}(1 \text{ X } 1):z_{1o}^{(v3)}(1 \text{ X } 1):z_{1o}^{(v4)}(1 \text{ X } 1):z_{1o}^{(v5)}(1 \text{ X } 1)]. \quad (92)$$

This is the training pattern that we finally use for training the network.

5.3 Time Domain Feature Selection

This is the most novel part of our forecaster. It is as such also the very fruit of the previous research done by one of our previous researchers [13]. The concept that was proposed was to use clustering and piecewise linear solvers or OLS for doing feature selection (PLOFS). In this section we will give a preview on basic feature selection and then mention in some detail the algorithm for PLN based feature selection. For detailed analysis readers are referred to [6], [13].

Feature Selection- Feature selection is the statistical way of identifying the system. In feature selection we try to map the error function or the Empirical risk to the feature space (input space). Feature space can have different meaning in different cases. We shall try to explain this by an example-

The feature selection in the function link net means to select from the input of the neural network which are polynomial basis functions depending on the order of the polynomials. In cases like pruning or growing the feature selection may imply choosing from the hidden unit basis functions i.e. removing the hidden units one after other and keeping a track of which hidden unit causes the least impact on the error function. Similarly in case of Piecewise linear networks sometimes clusters can be features. In such cases we look at error function variations whence we remove certain clusters. So in general we map the error function to the input space or the feature space and choose the most important features which would make the error reduce to larger extent. Those features which would not make much reduction in the error function can be left or discarded. One very good reason to do the feature selection is to protect the network from the curse of dimensionality. Of course as we choose the most important features, dimension of input space is reduced.

Coming to our application of weather variable prediction using the measurements of the previous hour, for us the feature selection would mean choosing measurements of those hours which would contribute the most towards error reduction. Measurements of those hours which would not contribute in reduction of error significantly, those features can be discarded. Also this is the sole reason for doing the PLN based feature selection in time domain. The piecewise linear approximation is now already known well enough to approximate the non-linear function in a very good capacity. When we are in time domain the input pattern to the neural network is the pattern of variable that is formed out of measurements of last few hours or measurement over last few days in some case. Taking the measurements on the hourly basis this would mean the neural network with over 100 inputs. And hidden space even larger than that! Processing such a large network is a curse for the machine. Therefore the feature selection is a most important concept in treating the non-linear systems statistically.

Our knowledge on the systems, especially the one like ours, is very intuitive. For example as mentioned many times before, for predicting the value at any specific hour measurement of 24 hr prior to it is most important sometimes only. This intuitive knowledge is very crudely assumed to be true by a lot of researchers [23]-[25] When we use the concept of one forecaster/hr it means we are assuming the this intuitive knowledge is 100 percent accurate, and thus the assumption, that we need not use any other measurement apart from the one 24 hrs prior one in order to predict the value for a specific hour. But the result presented in chapter 6 using the PLN feature selection would prove this assumption wrong most of the times. Sometimes in case of sudden weather changes the value of measurement immediately prior to current value can be the most important feature to predict the value of next hour. This will be presented in chapter 5 and 6 when we put the results of the time domain feature selection. This is also a strong reason for not using the PLN network itself for prediction i.e. we don't assume that an independent network (one network for each hour) is an accurate approach implying that the assumption "that the network for one hour has nothing to do with the network of another hour, or the cluster of one hour has nothing to do with another hour (since PLN is based on clustering)" may not be always correct. The two networks (clusters) of consecutive hours have a strong correlation with each other. For these reasons we will limit the use of PLN for time-domain feature selection only and use a MLP for prediction purpose.

Dividing the feature space into specific number of partitions is a critical task as far as feature selection is concerned. To simplify this concept we again give an example of one forecaster/hr again. In this application of one forecaster/hr we divide the feature space in 24 partitions; one representing each hour. And then assume that within a partition we can use a linear solver (simple OLS) or a non-linear solver (hidden sigmoid activation and OLS for output weights) to get the solution. Let us say, it is a very rough way of exploiting the PLN feature selection where features are intuitively selected, and then only that one feature is used to extrapolate the non-linear function, assuming the one feature subspace has nothing to with other feature subspace. For simplicity we would go back to the training file format previously mention in other sections. For us the feature space (when we use PLN based feature selection is the input space) means each column in the training pattern file is a feature. So finally when we are done with the feature selection, we would get a sequence of features from this algorithm. Using only those columns or basis or features we shall form the new training file. Over this training file we will then use the KLT compression mentioned in previous section.

Previously a lot of algorithms have been proposed for feature selection. Enlisting a few-branch and bound algorithm has been found optimal search algorithm [38]. Principal component analysis and Independent component analysis have been also used for feature selection as in [39], [40], [41], [42], [13]. As far as neural network feature selection is concerned pruning, growing and hybrid approach have been used by [43]-[49]. Feature selection algorithm that we apply to hidden unit pruning is elaborated in [34]. In essence feature selection is a complicated 'search algorithm' and so its application is always limited by the computational complexity. As mentioned in [38] branch and bound also becomes impractical for feature more than 30 inputs. Exhaustive search is always ruled out when application of large neural network are to be made. To prevent nesting effect 'plus l minus r' method and floating search instead of exhaustive search is propose in [38], [50], [51]. Also in order to check the validity of the selected features validation error concepts like 'K-fold validation' and 'leave one out validation' are to be applied. As mentioned before we will definitely validate the network using totally new data, and also we will try to find the optimal number of hidden units from the validation error. But we will not be pruning the network. Instead in the second run of algorithm we start by using the N_h, that we find from the first run of the network validation.

In essence time-domain feature selection consists of 2 steps. One is feature sub-setting and then the subset evaluation. The criteria to be observed in subset evaluation is the validation error in most of the cases. In order to subset the features any of the algorithms mentioned in above paragraph is used. The way we do feature selection for linear case, is we calculate the auto correlation and cross-correlation of the training data. Then make subset of inputs using any of the sub-setting approaches. Then we solve the network for that subset using the OLS wherein we use the associated rows and columns of the auto and cross-correlation matrix. Then we evaluate that subset using the validation error. This way we get an optimal subset of the inputs (which are important features). Now we shall elaborate theoretically the PLN based feature selection algorithm (PLOFS). There are certain concepts like clustering, Forward OLS etc that are elaborated in [34]. But we shall try to explain these in as much shortness as possible and then references will be given.

PLOFS (Piecewise Linear-OLS based Feature Selection) [13]-

We shall give the algorithm and the necessary description in each step-

1) Initialize the N_s. N_s is the number of features of which size we want the feature subset. $N_s<N$. But in our case we shall use $N_s=N$. for the reason that we want to order all the inputs or the features in order of their contribution to the error function. Then we shall leave it to the user to choose the inputs that they want to use for training the forecaster. We shall provide the curve for size of feature subset verse the validation/prediction error in chapter 6. Users can choose from that.
2) Initialize the number of clusters N_c and cluster the training data with any random means or using any clustering algorithm like K-means or SOM or adaptive K-means. Once the clusters are formed we treat each cluster as a training file and solve for weights using the OLS using the auto and cross-correlation matrices calculated of each cluster independently. Each cluster will have its own auto and cross-correlation matrices. This way we get N_c linear networks i.e. one for each cluster. Now when we want to process a new pattern using this network, we first of all calculate the distance of that pattern with mean of all the clusters. The mean from which the distance is minimum we apply the weight matrix of that cluster to calculate the output of the network for that pattern. As readers would already have noted that we partition the input space into a number of subspaces; one represented by a cluster. And we assume that within each cluster/subspace the behavior of the non-linear system will be linear. This is how PLN

training proceeds in general [6]. Of course this will call for calculation of auto and cross-correlation matrics N_c times and OLS is also applied N_c times making this algorithm highly dependent on the number of cluster that we divide the input space into. So in order to avoid this we try to reduce number of clusters using cluster pruning.

3) In order to prune or reduce the number of clusters we basically use the validation data. Using the patterns in validation data we in essence order the clusters as follows. We use backward selection on the clusters to subset them. We make subsets of N_c-1 clusters till each cluster is removed once. And we calculate the error of the network with this one cluster removed and see the rise in the validation error from the validation error calculated when all the clusters are in network. Then we delete another cluster and use the validation data and process it in the network and calculate the rise in error (remember the existing number of clusters are N_c-1). This way remove each cluster once and see the rise in error for each cluster being removed. The one cluster, removing which, we get the least increase in the error is the one to be pruned or removed. Remember that deleting the cluster means reassigning the patterns of that cluster to the cluster closest to that pattern and updating the auto and cross correlation matrix of that cluster. This way we remove one cluster. Then we make subsets of N_c-2 and run the same process again to remove second cluster. This way we can keep removing the clusters till we get the number of clusters we want to keep in the PLN. This way we get a graph of validation error versus the number of clusters, from which we can choose optimal number of clusters to be kept in the PLN.

4) Once we have found an optimal size of subset of clusters then we use the forward selection on each input. This is the actual feature selection that we need in order to find the most important inputs. What we do in this step is that after we fix the number of cluster and solidifying the network we make subsets using the forward selection i.e. increasing the size of subset from 1 onwards. Using the linear network of each cluster

we ortho-normalize the linear network of each cluster for training patterns using only that input subset and using each of the clusters. Calculate the training and validation error of the network over that input subset only. The one subset that gives the least error over whole network (i.e. error out of each linear network summed up over all patterns) is kept. Then we increase the size of subset from 1 to 2 and ortho-normalize the network using those 2 inputs only and calculate weights. Of course when a subset of 1 is used we have auto and cross-correlation matrices of 1 X 1, and when subset of 2 then they are of 2 X 2 and 2 X 1 respectively. This way we find linear networks of different sizes for each subset of features/inputs, calculate the validation error and see which subset give the maximum reduction in error from the previous subset size. Keep that input in the subset (a sequence formation begins). Then keep increasing the size of subset one by one and validate the network using all the clusters (whole PLN) till we cover all the inputs and form a sequence which then represents the importance of each input in the network. It will be seen that after the subset crosses certain size there will be hardly any reduction in the error. This would be the size of subset that will be optimal i.e. the best compromise between the network size and validation error/prediction error. A plot of the validation error versus subset size for each type of variable will be shown in chapter 6. This is the PLN feature selection where in we finally get a sequence of inputs in order of their importance in the network, using which we form a new thinner time-domain training file over which we then apply KLT feature selection.

Purpose of Time Domain Feature Selection (PLOFS) - In the forecaster application there is a myth that most of researchers have faced and have tried to avoid. The myth is that in order to be able to predict the values for more hours in future with a good accuracy, we need to use longer window sizes (in figure 5.1 and 5.2), that spans the data for over a few days. Infact ‘more the past data, more better is the prediction’ is a kind of notion that researcher would agree to.

But that logic is flawed for a lot of reasons and we shall in this section try to show that by making an experiment using the PLOFS program.

In order to verify if the more data from past would be helpful of deteriorating, we shall experiment with PLOFS program using some 9000 training patterns formed as in figure 5.1 and 5.2 for single variable, and the pattern dimensions will be 148 inputs and 3 outputs. In the next section we try to predict all 4 variables for upto 51 hours and display results for 3 cases, 1 hr prectiction, 26 hrs prediction and 51 hours prediction. But in this section in order to make an experiment, we shall use the pattern formation as follows-

$$\mathbf{x}_1=[tim_1(1)\ldots.tim_1(4):v_{11}(i-144)\ldots.v_{11}(i)],\ \ \mathbf{t}_1=[v_{11}(i+1):v_{11}(i+26):v_{11}(i+51)]$$

We form 9000 patterns of this form and then apply PLOFS on this pattern file and show the results and interpret them in order to conclude the necessity of the feature selection in time-domain. For ‘v’ first subscript represents the variable number (either of 1 to 4) and second subscript represent the pattern number. For ‘**x**’ and ‘**t**’ the subscript means the pattern number. The results of the feature selection are obtained as a curve of the training error and validation error versus the subset size which later on becomes the inputs to neural network. In order to show the selected input numbers by PLOFS we shall use symbols of ‘x()’, $x_p(1)=tim_p(1)$, …..$x_p(147)=v_{11}(i-1)$, $x_p(148)=v_{11}(i)$ and so on, subscript p is the pattern number. Next we show these results for each of the four variables and request the readers to notice the common trend in them all.

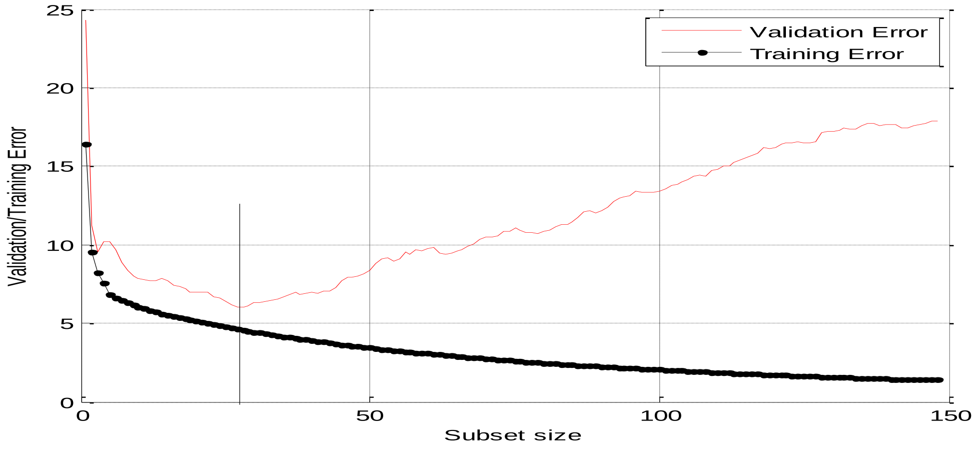


Figure 5.9 Plot of training and validation error versus subset size for Temperature from PLOFS

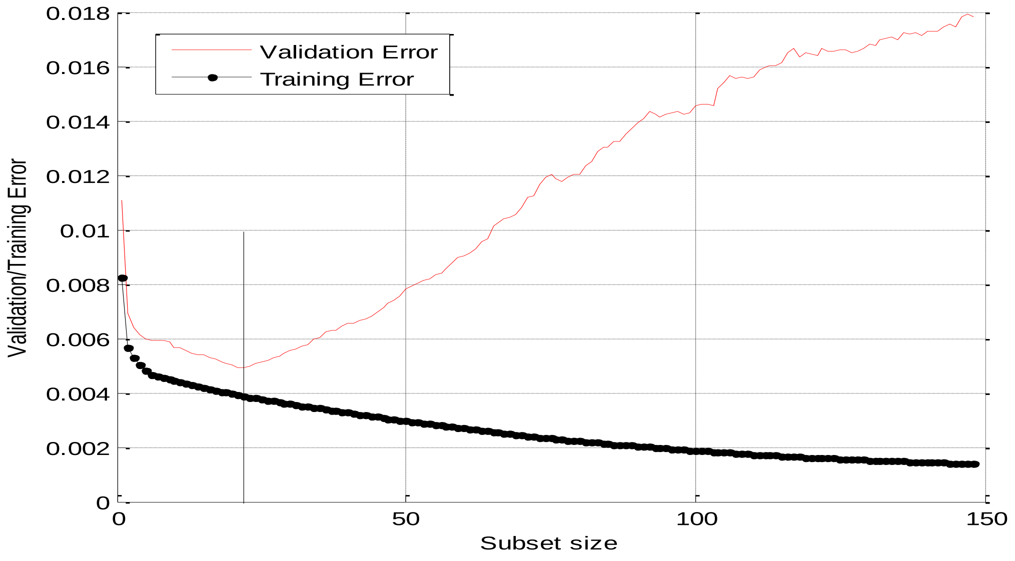


Figure 5.10 Plot of training and validation error versus subset size for Humidity from PLOFS

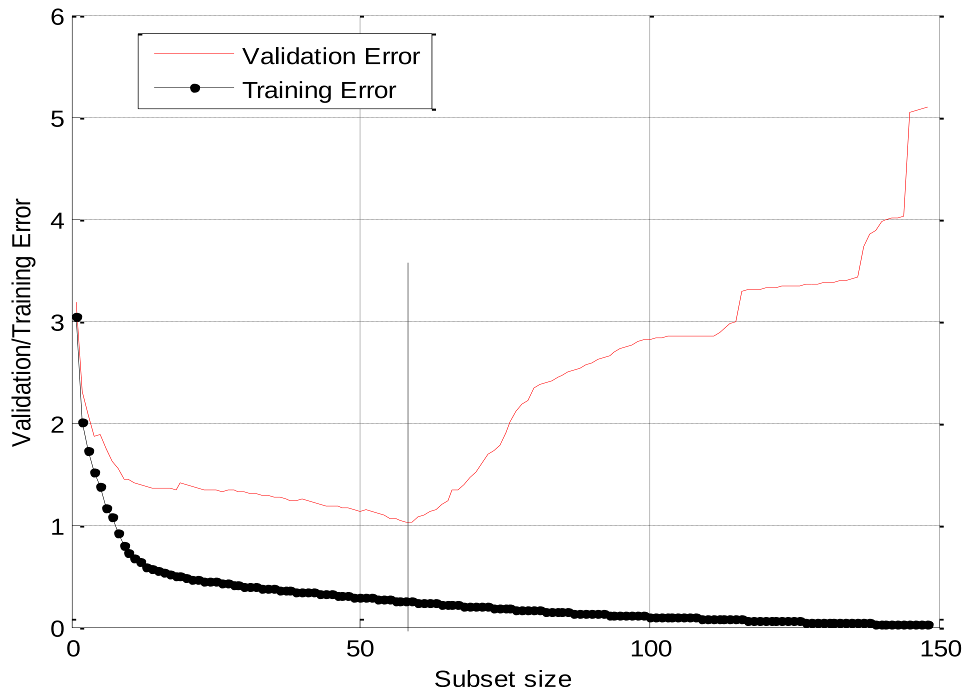


Figure 5.11 Plot of training and validation error versus subset size for Radiation from PLOFS

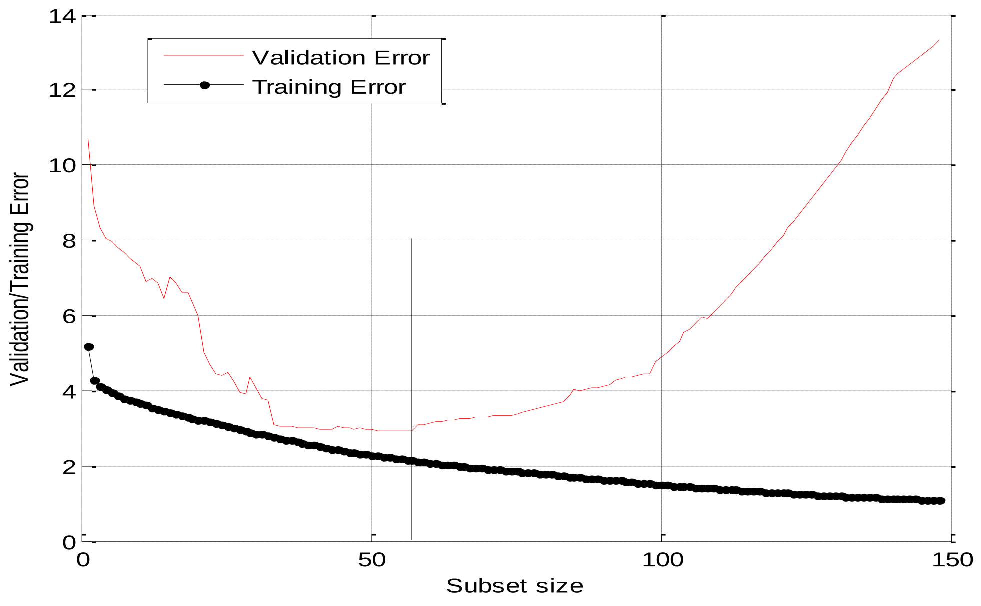


Figure 5.12 Plot of training and validation error versus subset size for Wind Speed from PLOFS

Observed in the plot is firstly that fact that by using more data from past and having a larger pattern would help in reducing the training error(only if trained for larger number of iterations), but the validation error which actually represent the capability of forecaster to predict to alien data, increases after certain subset size is achieved. For record the optimal subset sizes found (shown by vertical line) are 28, 22, 58 and 56 for temperature, humidity, radiation and wind speed respectively. But in order to know whether the important/optimal subsets has recent data in it or more past data in it we shall look at the optimal subsets found out in each case.

Table 5.1 Optimal Subset sequence from PLOFS for temperature

Optimal Subset, Size:28													
1	2	3	4	5	6	7	8	9	10	11	12	13	14
$x_p(148)$	$x_p(147)$	$x_p(138)$	$x_p(4)$	$x_p(2)$	$x_p(124)$	$x_p(126)$	$x_p(132)$	$x_p(144)$	$x_p(143)$	$x_p(130)$	$x_p(141)$	$x_p(127)$	$x_p(1)$
15	16	17	18	19	20	21	22	23	24	25	26	27	28
$x_p(76)$	$x_p(102)$	$x_p(5)$	$x_p(7)$	$x_p(89)$	$x_p(77)$	$x_p(125)$	$x_p(100)$	$x_p(49)$	$x_p(101)$	$x_p(103)$	$x_p(3)$	$x_p(99)$	$x_p(140)$

Table 5.2 Optimal Subset sequence from PLOFS for humidity

Optimal Subset, Size:22										
1	2	3	4	5	6	7	8	9	10	11
$x_p(148)$	$x_p(147)$	$x_p(2)$	$x_p(130)$	$x_p(145)$	$x_p(4)$	$x_p(139)$	$x_p(98)$	$x_p(142)$	$x_p(103)$	$x_p(1)$
12	13	14	15	16	17	18	19	20	21	22
$x_p(3)$	$x_p(123)$	$x_p(6)$	$x_p(124)$	$x_p(100)$	$x_p(122)$	$x_p(146)$	$x_p(125)$	$x_p(101)$	$x_p(144)$	$x_p(126)$

Table 5.3 Optimal Subset sequence from PLOFS for radiation

Optimal Subst, Size:58														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	
$x_p(148)$	$x_p(147)$	$x_p(143)$	$x_p(142)$	$x_p(144)$	$x_p(138)$	$x_p(140)$	$x_p(139)$	$x_p(1)$	$x_p(146)$	$x_p(145)$	$x_p(137)$	$x_p(141)$	$x_p(24)$	
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
$x_p(126)$	$x_p(135)$	$x_p(136)$	$x_p(124)$	$x_p(2)$	$x_p(106)$	$x_p(122)$	$x_p(130)$	$x_p(107)$	$x_p(129)$	$x_p(121)$	$x_p(7)$	$x_p(115)$	$x_p(132)$	$x_p(133)$
30	31	32	33	34	35	36	37	38	39	40	41	42	43	
$x_p(90)$	$x_p(128)$	$x_p(127)$	$x_p(81)$	$x_p(99)$	$x_p(79)$	$x_p(100)$	$x_p(120)$	$x_p(3)$	$x_p(4)$	$x_p(125)$	$x_p(104)$	$x_p(105)$	$x_p(80)$	
44	45	46	47	48	49	50	51	52	53	54	55	56	57	58
$x_p(82)$	$x_p(119)$	$x_p(118)$	$x_p(101)$	$x_p(84)$	$x_p(131)$	$x_p(77)$	$x_p(75)$	$x_p(96)$	$x_p(134)$	$x_p(83)$	$x_p(69)$	$x_p(102)$	$x_p(123)$	$x_p(103)$

Table 5.4 Optimal Subset sequence from PLOFS for wind speed

Optimal Subset, Size:56													
1	2	3	4	5	6	7	8	9	10	11	12	13	14
$x_p(148)$	$x_p(147)$	$x_p(2)$	$x_p(145)$	$x_p(130)$	$x_p(3)$	$x_p(4)$	$x_p(79)$	$x_p(124)$	$x_p(128)$	$x_p(103)$	$x_p(125)$	$x_p(129)$	$x_p(1)$
15	16	17	18	19	20	21	22	23	24	25	26	27	28
$x_p(98)$	$x_p(75)$	$x_p(100)$	$x_p(102)$	$x_p(141)$	$x_p(146)$	$x_p(79)$	$x_p(103)$	$x_p(140)$	$x_p(85)$	$x_p(56)$	$x_p(99)$	$x_p(95)$	$x_p(118)$
29	30	31	32	33	34	35	36	37	38	39	40	41	42
$x_p(122)$	$x_p(121)$	$x_p(117)$	$x_p(91)$	$x_p(138)$	$x_p(92)$	$x_p(107)$	$x_p(108)$	$x_p(131)$	$x_p(126)$	$x_p(101)$	$x_p(135)$	$x_p(89)$	$x_p(127)$
43	44	45	46	47	48	49	50	51	52	53	54	55	56

From the table 5.1 to 5.4, in each case the PLOFS have given out sequences which has most of the features from past 2 or at the most 3 days. The data that was used was from last six days i.e.144 hours. Now, in this case we had tried to predict for 1 hrs ahead, 26 hrs ahead and 51 hours ahead. This would actually mean that intuitionally out of 148 inputs (148th input being the recent past and 105th input being 6 days prior data) 125th ,101st , 77th , 53rd , 29th inputs would be important for predicting the 1 hours future value. Similarly for predicting 26th hour future value 126th 102nd, 78th, 54th, 30th values would be of importance because 26th hour ahead is just 2hrs ahead after 1 day. And similarly 127th, 103rd, 79th, 55th, 31st, values would be important for predicting 51st hour ahead values because 51 hours ahead is just 3 hours ahead after 2 days. But it turns out, that this heuristic is not true. It is not the same hour measurement from past 5 days that is important, but the data of immediate past few hours that will be important in order to predict ahead. Infact even the sinusoidal time related inputs 1, 2, 3, 4 are always more important than previous days data. As such it is seen from the plots of PLOFS that adding more past data would not help the network to generalize better, but instead they will act as noise and deteriorate the networks validation error. Out of 144 at the most 60 inputs would be doing correct work rest of the inputs spoil the network performance. So going in more past to reach out older information is of no use. For this reason having a larger training pattern is also not useful, because as mentioned before it increases the chances of memorization and as such lot of the inputs would act as noise. For this reason having the PLOFS done before the KLT transform is of utmost important in order to know the best capability of the PLN network to approximate the outputs. Of course as we will add the non-linear part (hidden layer) the performance and the validation error reduces from what is shown in the above plots. This curve only shows the best capability of the PLN network to approximate using the given data. Non-Linear networks will obviously give a better result for approximating non-linear function then PLN. For this reason we keep the use of PLN only for feature selection and leave the approximation problem to the MLP training.

Using the above conclusion, in the next chapter we shall display the results of prediction for 3 cases for each variable using the PLOFS and KLT in each case. Case 1 is for 1 hr ahead in future prediction, case 2 is 26 hrs ahead prediction and case 3 is 51 hrs ahead prediction. In each case for each variable we shall initially start with the time domain pattern that is formed out of a window that spans over 2 days i.e. 48 hours of previous data, augmented with 4 time inputs and ‘1’, we shall have initial time-domain file of 53 inputs and 1 output.

CHAPTER 6

RESULTS

This chapter is dedicated to the results obtained using algorithms and conclusion from chapter 3, 4 and chapter 5. We shall divide this chapter in 6 sections with 3 cases in first 5 sections. First section will display the results for the temperature forecast. We shall display the result for temperature forecast for 1hr ahead in future, 26 hrs ahead in future and 51 hours ahead in future in subsection 6.1.1, 6.1.2 and 6.1.3 respectively. In the same format sections 6.2, 6.3, 6.4 and 6.5 will display the results for humidity, solar radiation, wind and combined forecaster respectively. The sequence for forecasting using 2 stage feature selection goes as described now.

- First we form the time domain training file randomly choosing the window size for past data to be used for inputs of forecaster, and for the output of the forecaster depending on how many hours ahead in future do we want to predict. We have gone as far as 51 hours, so we shall show the case for predicting 51 hr in future. This will give us the time-domain file as in section 5.1 case-1. Out of this we form the time-domain file as in section 5.1 case-2 wherein in there is only one output no matter how far in future we want to predict.
- Over this file we apply time-domain feature selection as in section 5.3 using the PLOFS algorithm. From this algorithm we choose an optimal subset size. This is the first advantage of 1st stage feature selection in time-domain. Then over this new time-domain training file we shall apply the KLT or Transform domain feature selection as shown in section 5.2. Over this file we shall apply the algorithm from section 3.2 and then calculate the validation error over all the ordered hidden units or the generalization

capability of the forecaster. It is therefore our claim that by applying this 2 stage feature selection we are capable of reducing the dimension of the input space to 1/4th that of the original time-domain file in order to satisfy the heuristics mentioned in chapter 4. As such the lower the ratio N/N_v, greater will be the generalization ability of the forecaster and lower will be the validation error.

- We shall also show the plot of validation error versus the N_h in order to find the optimal number of N_h to be used for the network. That plot will also be shown in each result along with the training and validation error of the network with the iterations. Finally we shall show the reconstruction for each case in each section.

In each case we use 13000 patterns for training and 4000 patterns for validation. We start with input window or 48 hrs i.e. 4+48 inputs and 1 output. 4 extra inputs i.e. inputs number 1, 2, 3, and 4 are the time related inputs as described in chapter-2. We will assume ‘1’ to be an important input for all the cases in all variables. So time domain feature selection (PLOFS) need be applied on the ‘52 input 1 output’ files leaving ‘1’ aside. From now on we shall refer to PLOFS as time-domain feature selection alternatively TDFS in short. After all the cases are realized for each variable we shall then show the results of combined forecaster wherein we have treated all 4 variables together along with wind direction. The result of this combined version has been found to be better than individual predictors. In the end of this chapter we shall give the case 3 results for prediction in time domain, without any feature selection. A comparative of the advantage in the reduction in the size of network is given in table 5.61. Readers can also compare the results of over smaller forecaster with the bigger time-domain one from table 6.62 and 6.63.

6.1 Results of temperature forecast

Case1- Results for temperature forecast for 1 hr in future.

Table-6.1 Subset size and input sequence from PLOFS

Optimal Subset, Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(4)$	$x_p(27)$	$x_p(30)$	$x_p(52)$	$x_p(49)$	$x_p(2)$	$x_p(51)$	$x_p(3)$	$x_p(6)$	$x_p(29)$	$x_p(28)$	$x_p(10)$	$x_p(41)$	$x_p(31)$	$x_p(40)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(47)$	$x_p(20)$	$x_p(1)$	$x_p(50)$	$x_p(8)$	$x_p(5)$	$x_p(15)$	$x_p(43)$	$x_p(35)$	$x_p(26)$	$x_p(44)$	$x_p(46)$	$x_p(39)$	$x_p(42)$	$x_p(21)$

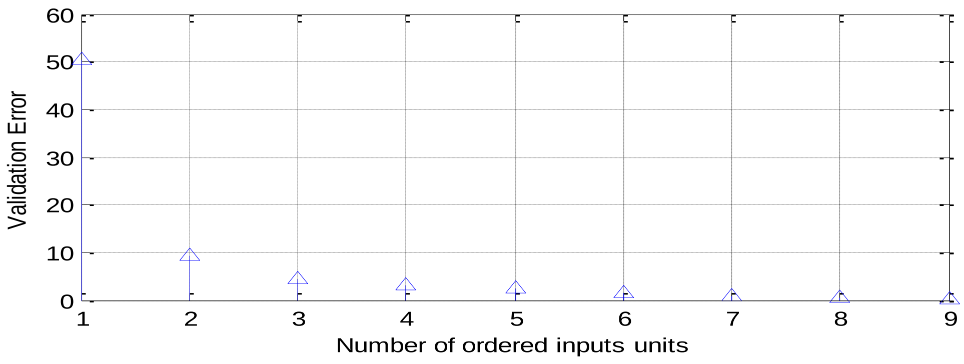


Figure 6.1 Validation error over all the ordered inputs units after training for 20 iterations

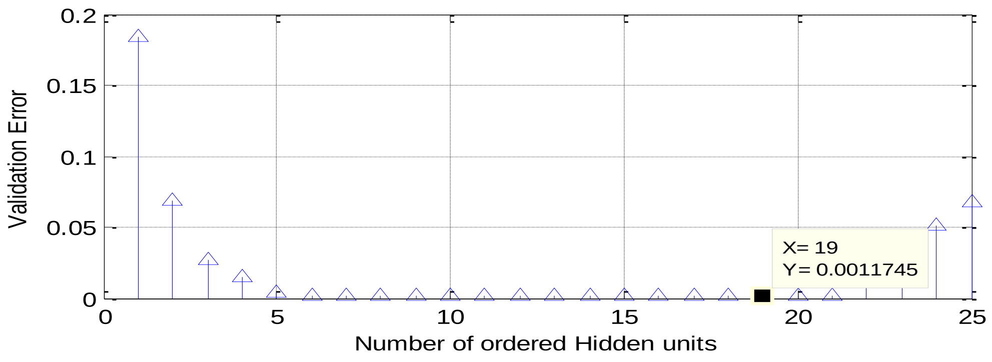


Figure 6.2 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.2 Training error for 20 iteration

Training Error			
It No.	Value	It No.	Value
1	0.286444	11	0.097053
2	0.281605	12	0.09143
3	0.258025	13	0.085889
4	0.237296	14	0.077511
5	0.22315	15	0.069148
6	0.209115	16	0.068942
7	0.180775	17	0.0691
8	0.138941	18	0.068796
9	0.111106	19	0.067172
10	0.097176	20	0.067517

Table 6.3 Validation error for 20 iteration

Validation Error			
It No.	Value	It No.	Value
1	0.31964	11	0.102883
2	0.31296	12	0.097546
3	0.279284	13	0.091707
4	0.260228	14	0.082536
5	0.24295	15	0.074983
6	0.229085	16	0.072667
7	0.194804	17	0.07493
8	0.150495	18	0.074221
9	0.119645	19	0.076184
10	0.103417	20	0.076769

Table 6.4 Network Details

Optimum No. of Hidden units (20 Iterations)	19
Size of subset chosen from table 6.1	30
Number of KLT features used	4
Number of Hidden Units started with	25

From the figure 6.1 it is evident that the PLN networks cannot be used directly for prediction since the validation error is very large. One another thing to note in that figure is that, as we go more towards rights is that there is lesser relative reduction in the validation error with the increase in subset size. To elaborate the first subset of 10 inputs reduces the validation error by over 0.5. While the next subset of 20 features reduces the validation error by 0.5. But the next subset of 22 inputs can hardly reduce the error by 0.25. So as we go towards right, increasing the subset size do not greatly reduce the validation error to that extent. So using the first 20 or 30 inputs will make a proper sense as far as the reduction in the size of network is concerned as well as reduce the validation error. The major reduction in validation error will be caused by training itself.

From the table 6.1 and from the format of the training file discussed in chapter 4, it is possible to find out intuitively what inputs will be more important than others. For example for predicting value 1 hr ahead we will hope that the current hour measurement and the measurement 24 hrs before is important. This is evident from table 6.1. For the PLOFS we have

52 inputs and 1 output. For the output which is 1 hr in future 52nd input is very important , also the 51st is important. These 2 are in the leading numbers. Also the sinusoidal inputs related to time are also falling in the leading 20 inputs. Besides this the value of temperature variations around same time but 24 hrs before is important. So we see 27, 28, 29 and 30th inputs are also important. But as we have mentioned before that the measurement of temperature 1 hr prior is more important than the measurement of same hour of previous day. So input 51, and 52 are ranked before 28 and 29. But it can be questioned why 30 and 27 are before 51 and 52. But these 2 kind of inputs (i.e. 1/2 hrs prior measurement and 24 hrs prior measurement will keep competing in different cases, which we will see in some other case) will always be leading members. This is due to the periodic behavior of these variables (but then what about the wind! That is not periodic.)

Now from the validation error curve over the all the hidden and bypass units show that the inputs and not the hidden units contribute more towards error reduction. Though we see that the hidden units bring the error down to decimal numbers. But not all the hidden units help in error reduction. A few trailing hidden units (after ordering in OLS) end up increasing the error. For this reason we can say that for 20 iterations 19 hidden units will give minimum validation error.

Same can be said also about the training and validation errors over the iterations. The last few iterations reduce the training iteration but the validation error is increased. This means the generalization capability is being reduced while networks keeps memorizing. For this reason early stopping will give a better performance.

But the biggest advantage we gain is the reduction in the size of input space by this 2-stage feature selection. As mentioned in the table 6.4, from 52 inputs we cut down to 30(from the validation error of PLOFS) and then from 30 we cut down to 4 inputs in the KLT domain. This comparatively reduces the chances of memorization many fold time.

Case 2-Results for temperature forecast for 26 hrs ahead in future.

Table 6.5 Subset size and input sequence from PLOFS

Optimal Subset, Size:24											
1	2	3	4	5	6	7	8	9	10	11	12
$x_p(30)$	$x_p(49)$	$x_p(18)$	$x_p(2)$	$x_p(6)$	$x_p(4)$	$x_p(26)$	$x_p(35)$	$x_p(10)$	$x_p(52)$	$x_p(3)$	$x_p(46)$
13	14	15	16	17	18	19	20	21	22	23	24
$x_p(1)$	$x_p(5)$	$x_p(23)$	$x_p(43)$	$x_p(16)$	$x_p(40)$	$x_p(31)$	$x_p(28)$	$x_p(51)$	$x_p(22)$	$x_p(48)$	$x_p(20)$

Table 6.6 Training error for 20 iteration

Training Error			
It No.	Value	It No.	Value
1	2.637513	11	0.563028
2	2.56979	12	0.500928
3	2.315396	13	0.439248
4	2.0948	14	0.347893
5	1.941038	15	0.256901
6	1.786538	16	0.256784
7	1.481133	17	0.226512
8	1.023714	18	0.193786
9	0.718642	19	0.262897
10	0.566025	20	0.193408

Table 6.7 Validation error for 20 iteration

Validation Error			
It No.	Value	It No.	Value
1	2.878797	11	0.585567
2	2.704612	12	0.518851
3	2.432376	13	0.453029
4	2.175161	14	0.357122
5	2.021177	15	0.264349
6	1.862988	16	0.262971
7	1.515443	17	0.269085
8	1.068346	18	0.257106
9	0.739381	19	0.280424
10	0.596775	20	0.291136

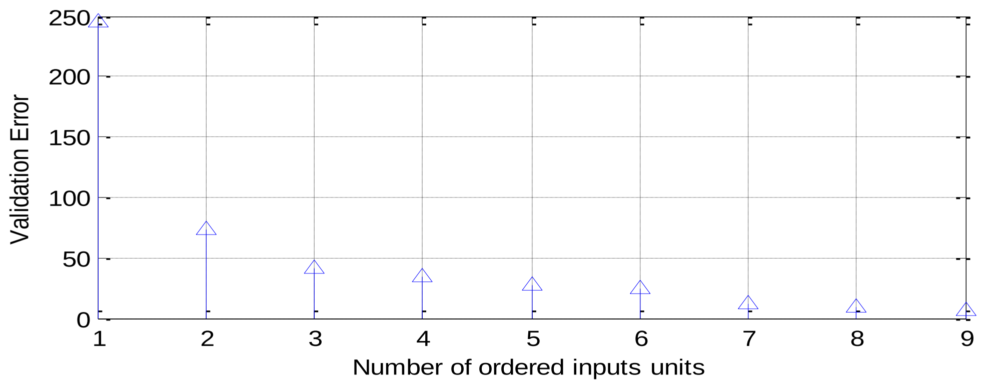


Figure 6.3 Validation Error over all ordered input units after training after 20 iterations

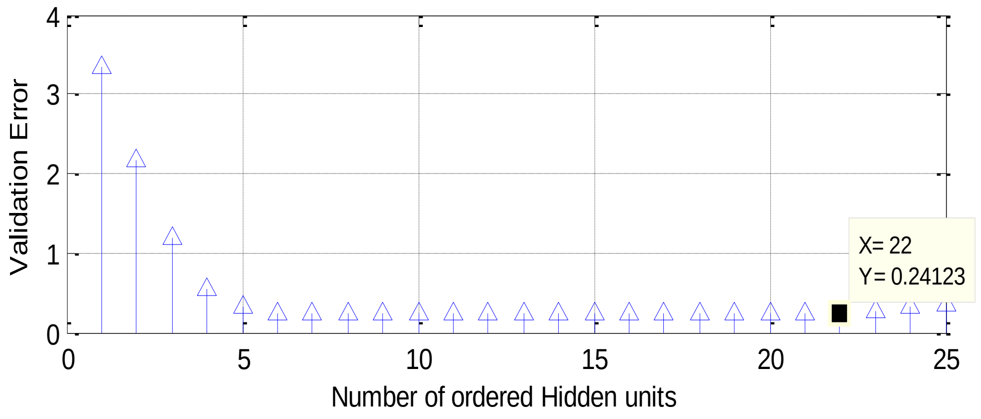


Figure 6.4 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.8 Network Details

Optimum No. of Hidden units (20 Iterations)	22
Size of subset chosen from table 6.5	24
Number of KLT features used	4
Number of Hidden Units started with	25

Same observations as in case 1 can be made in PLOFS. Also the input sequence from the PLOFS show that 30th input which is 24 hrs prior measurement is leading the race, while the 49th input which is a few hours prior measurement is also very close. Sinusoidal time inputs also are in the leading 15. But the more important in this case is the comparative with the case-1. The training and validation errors are more than case-1. Validation curve over the network units displays same behavior as in case-1 thereby concluding that not all hidden units get tuned to a helpful extent , thus helping us to find optimal number of hidden units.

Case-3 Results for temperature forecast for 51 hours ahead in future.

Table 6.9 Subset size and input sequence from PLOFS

Optimal Subset, Size:28													
1	2	3	4	5	6	7	8	9	10	11	12	13	14
$x_p(15)$	$x_p(31)$	$x_p(25)$	$x_p(2)$	$x_p(7)$	$x_p(4)$	$x_p(20)$	$x_p(1)$	$x_p(27)$	$x_p(52)$	$x_p(48)$	$x_p(3)$	$x_p(36)$	$x_p(5)$
15	16	17	18	19	20	21	22	23	24	25	26	27	28
$x_p(12)$	$x_p(23)$	$x_p(50)$	$x_p(10)$	$x_p(45)$	$x_p(17)$	$x_p(34)$	$x_p(43)$	$x_p(39)$	$x_p(33)$	$x_p(28)$	$x_p(41)$	$x_p(47)$	$x_p(30)$

Table 6.10 Training errors for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	16.18777	11	5.16632
2	15.81419	12	4.67981
3	14.68309	13	4.679059
4	12.83242	14	3.869388
5	12.32898	15	3.545465
6	11.50774	16	3.545025
7	9.878537	17	3.374441
8	7.441872	18	3.210794
9	5.656462	19	3.09749
10	4.844659	20	3.2102

Table 6.11 Validation error for 20 iteration

Validation Error			
It No.	Value	It no.	Value
1	17.3312	11	5.67938
2	16.98927	12	5.297705
3	15.68175	13	4.975165
4	14.20516	14	4.449087
5	13.52445	15	3.820206
6	12.65584	16	3.822435
7	10.94618	17	3.880239
8	8.314548	18	3.846425
9	6.553609	19	3.905218
10	5.66928	20	3.981834

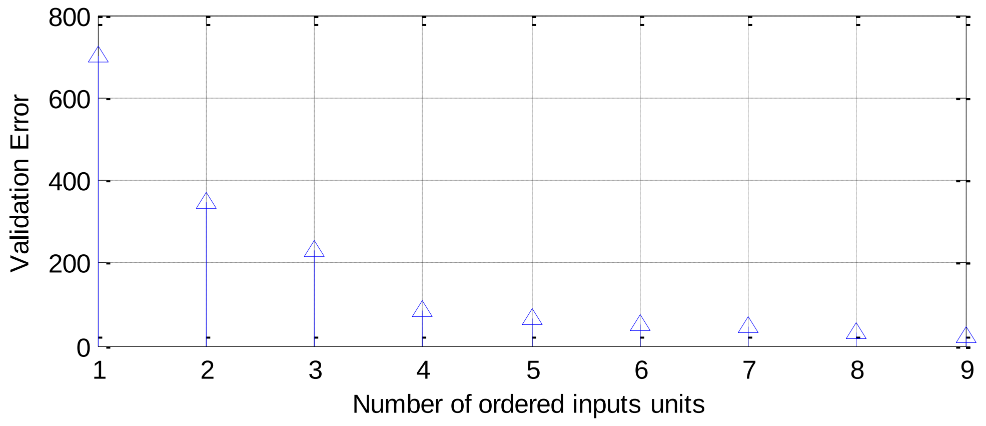


Figure 6.5 Validation Error over all the ordered input units after training for 20 iterations

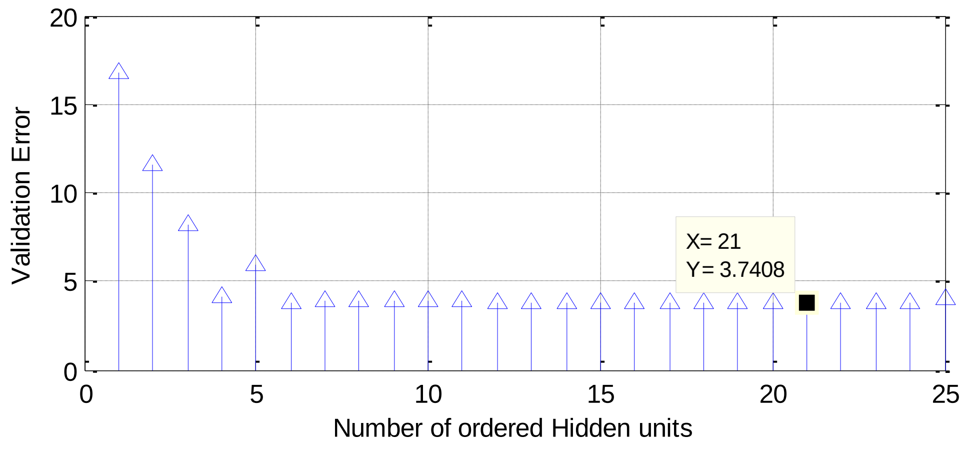


Figure 6.6 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.12 Networks Details

Optimum No. of Hidden units (20 Iterations)	21
Size of subset chosen from table 6.9	28
Number of KLT features used	4
Number of Hidden Units started with	25

Now, this case of predicting 51 hrs in future shows that 24 hour previous measurement is more important than the previous hour measurement to a greater extent and we can see some other number of inputs doing more work that the 51st or the 52nd inputs. Well this is genuine. When we are trying to predict temperature value 51 hrs ahead in future we are skipping 2 days of measurement in between. So immediate past hour measurement is as such not available. The last input number 52 is not the measurement of temperature of previous hour, but it is the measurement of temperature of previous hour 2 days back. So 31st input being in top 5 is a good success for the PLOFS. Also time based inputs are in the leading few. But it is also seen that 7th i.e. (31-24) input which is the measurement of 48 hrs prior is also leading. So in predicting some values we may find that not the prior hour measurement but 24 or 48 or 72 hours prior measurement would be playing important role in the error reduction. Besides even with all the correct feature selection and good number of KLT features (4 in this case) being same as in last case, still the prediction error is going to increase from the previous cases. This is going to be inherent in forecasting i.e. more the future we try to predict more the error will be. This will be clearly visible in the prediction plots shown next.

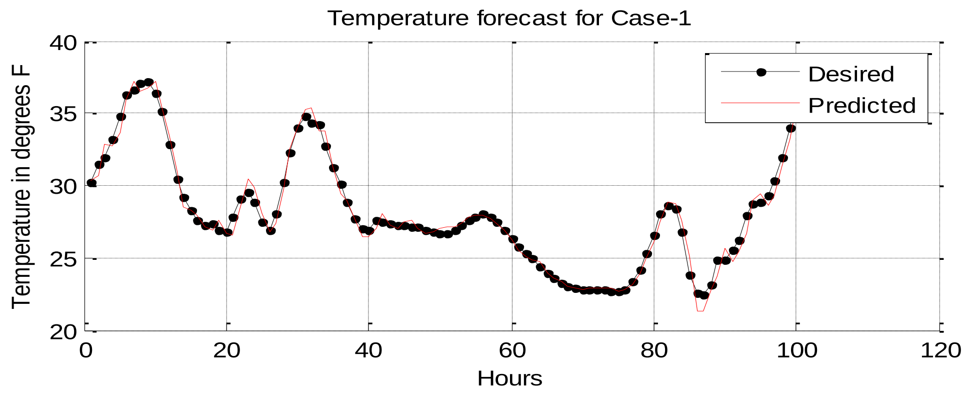


Figure 6.7 Prediction of 1hr ahead temperature (100 samples of Validation data)

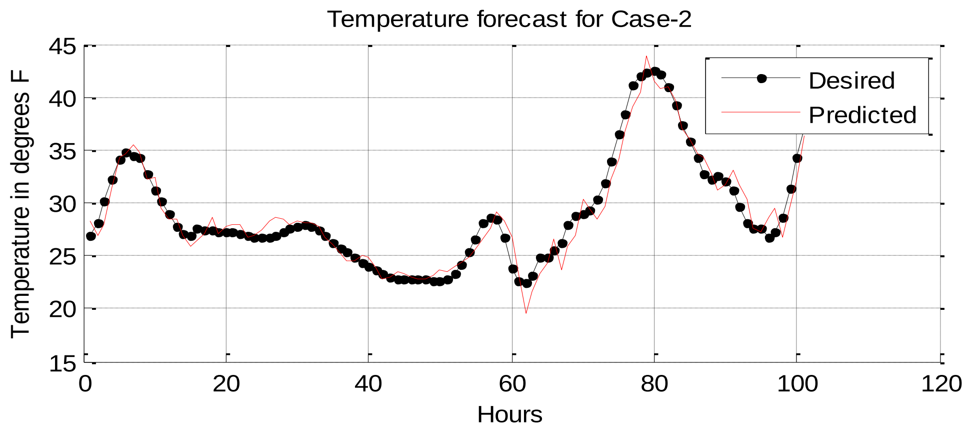


Figure 6.8 Prediction of 26 hrs ahead temperature (100 samples of Validation data)

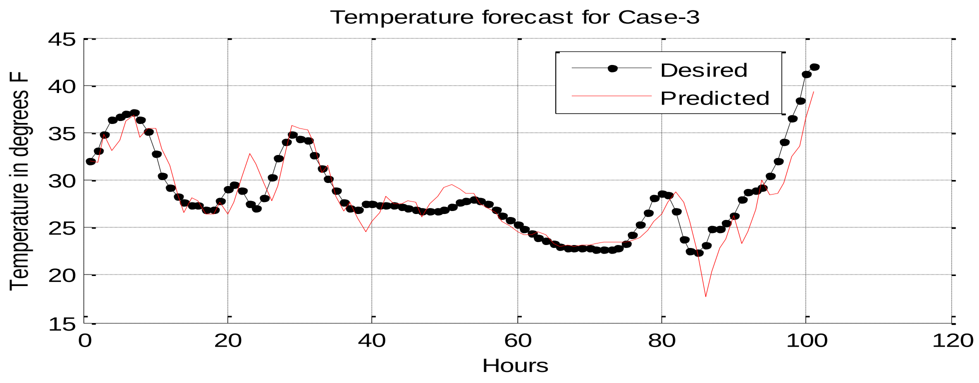


Figure 6.9 Prediction of 51 hrs ahead temperature (100 samples of Validation data)

As already observed, as we try to forecast more hours in future the forecast quality keeps deteriorating. The prediction becomes bumpy and noisy. But the main observation we would later on show in table 6.61 is that the size of the network inputs space dimension is reduced to a very small number due to this 2-stage feature selection, at the same time the prediction error is not so largely affected. Another important observation is that all throughout the thesis we have tried to stress on the fact that a smaller network is much easier to train and that the memorization is avoidable easily if the input space N+1 dimension is small as compared to N_v. For this reason the heuristics mentioned at end of chapter 3 about the ratio N/N_v is a good measure for the possibility of memorization. The lower the ratio the slimmer the training file will be and less will be memorization. Also notice that we have used a network with only one output, figure 5.5 and 5.6. So the KLT compression need not be applied to the output pattern in such cases and so the network training and validation error in KLT domain will be same as the time-domain prediction error after reconstruction.

6.2 Results of humidity forecast

Case 1: Results for prediction of percentage humidity for 1 hr in future.

Table 6.13 Subset size and input sequence from PLOFS

Optimal Subset,Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(27)$	$x_p(52)$	$x_p(4)$	$x_p(30)$	$x_p(49)$	$x_p(3)$	$x_p(51)$	$x_p(50)$	$x_p(41)$	$x_p(26)$	$x_p(45)$	$x_p(31)$	$x_p(47)$	$x_p(5)$	$x_p(2)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(8)$	$x_p(42)$	$x_p(25)$	$x_p(28)$	$x_p(29)$	$x_p(33)$	$x_p(12)$	$x_p(35)$	$x_p(37)$	$x_p(39)$	$x_p(46)$	$x_p(43)$	$x_p(1)$	$x_p(21)$	$x_p(18)$

Table 6.14 Training and Validation errors for 10 iteration

Iteration No	1	2	3	4	5	6	7	8	9	10
Training Error	0.002	0.00189	0.00164	0.00145	0.00123	0.0011	0.001	0.00101	0.001	0.001
Validation Error	0.0036	0.00319	0.00284	0.00241	0.002	0.0021	0.0021	0.0023	0.0024	0.00244

Table 6.15 Network Details

Optimum No. of Hidden units (10 Iterations)	4
Size of subset chosen from table 6.13	30
Number of KLT features used	4
Number of Hidden Units started with	5

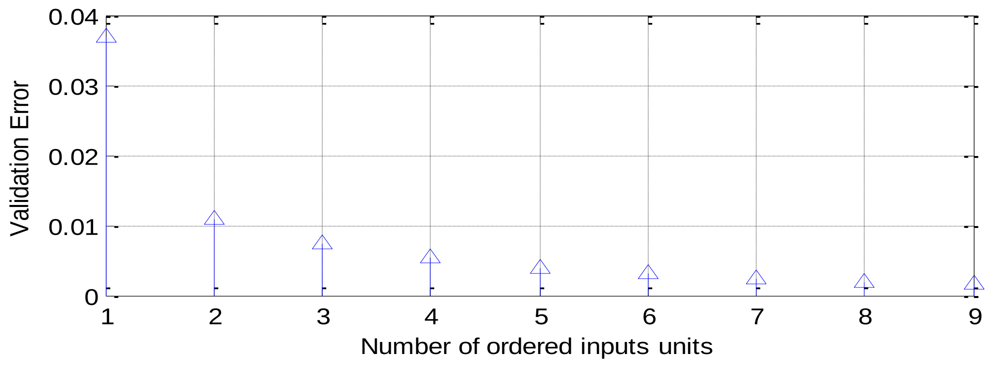


Figure 6.10 Validation Error over all the ordered input units after training for 10 iterations

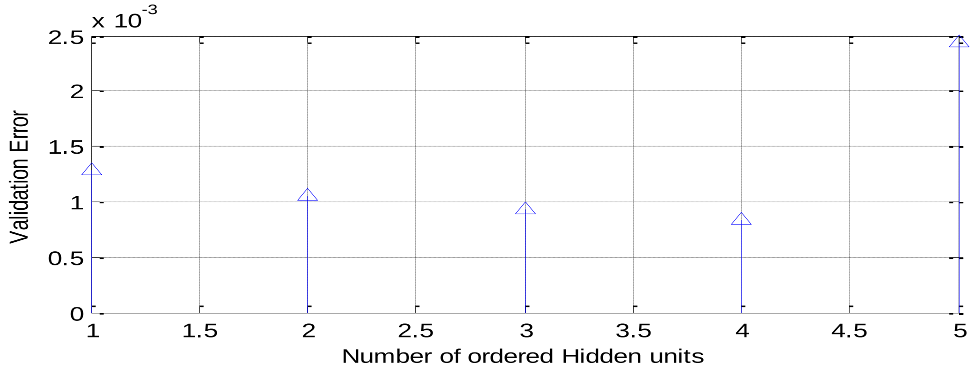


Figure 6.11 Validation Error over all the ordered hidden units after training for 10 iterations

Relative humidity is a type of signal wherein as very small number like 0.001 has a good significance. The variable being at normalized scale between 0 and 1 measured in percentage, predicting this variable with very good accuracy is slightly tough. As such the validation and training error of 0.00246 and 0.0039 for case 1 of this variable looks bad as compared to 0.07 and 0.06 of the case-1 of temperature. This is evident from plots of prediction for humidity. Talking about other things the sequence of features again makes the same sense as in the case-1 of temperature. As can be seen from the PLOFS figure 6.13 that use of PLN directly for prediction will give horribly large errors in prediction. Readers may wonder as to why have we choose such a large subset of 30 inputs out of PLOFS. The way we justify this is by saying that we wish to have all the sinusoidal inputs i.e. 1, 2, 3, 4 related to time in the chosen subset. The reason for that is, we don't want the neural network training to deal with estimating the time. If we choose say first 3 time inputs and leave out the 4th then neural network training and validation error would increase a lot because the training has to account also for the missing time related input i.e. estimate the 4th time related input from the existing 3 that we fed it. For this reason, in order to maintain the synchronization of the data with the time, we make sure that we fed all 4 time related inputs to the neural network during training and even any other time related input that we may have, that would help the network to estimate time. In fact, more the time related sinusoids, better result will it give. As such it is more sensible to chose the subset which corresponds to minimum validation error of the PLN.

Case-2 Result for the forecast of Relative Humidity for 26 hrs in future.

Table 6.16 Subset size and input sequence from PLOFS

Optimal Subset, Size:25												
1	2	3	4	5	6	7	8	9	10	11	12	13
$x_p(48)$	$x_p(52)$	$x_p(2)$	$x_p(21)$	$x_p(10)$	$x_p(16)$	$x_p(5)$	$x_p(30)$	$x_p(1)$	$x_p(36)$	$x_p(7)$	$x_p(27)$	$x_p(25)$
14	15	16	17	18	19	20	21	22	23	24	25	
$x_p(40)$	$x_p(19)$	$x_p(4)$	$x_p(15)$	$x_p(45)$	$x_p(50)$	$x_p(33)$	$x_p(42)$	$x_p(23)$	$x_p(3)$	$x_p(38)$	$x_p(12)$	

Table 6.17 Training and Validation errors for 10 iteration

Iteration No	1	2	3	4	5	6	7	8	9	10
Training Error	0.0063	0.0059	0.0054	0.00505	0.0045	0.0041	0.004	0.0041	0.004	0.0039
Validation Error	0.0076	0.0064	0.006	0.00521	0.005	0.0046	0.0041	0.0035	0.0031	0.00246

Table 6.18 Network Details

Optimum No. of Hidden units (10 Iterations)	4
Size of subset chosen from table 6.16	25
Number of KLT features used	4
Number of Hidden Units started with	5

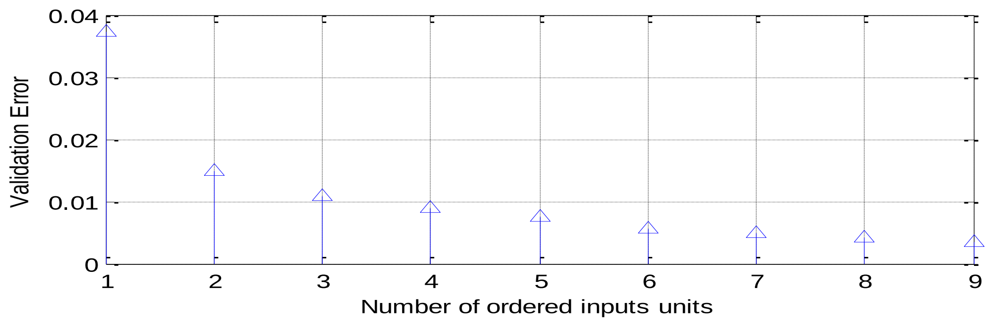


Figure 6.12 Validation Error over all the ordered input units after training for 10 iterations

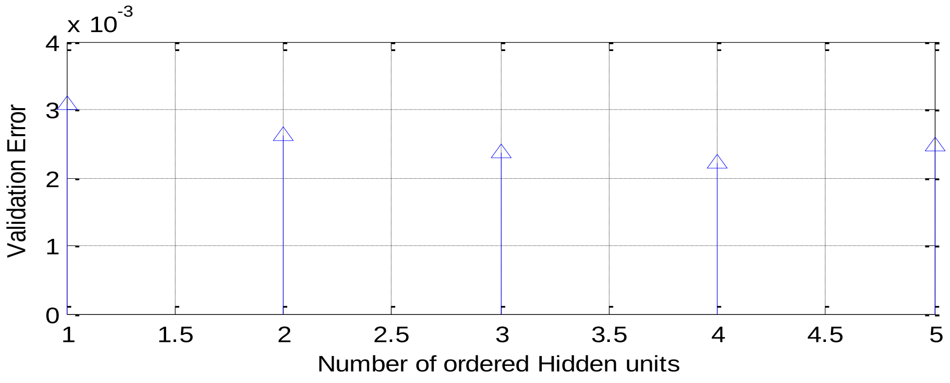


Figure 6.13 Validation Error over all the ordered hidden units after training for 10 iterations

One thing that readers my note in the humidity prediction is the fact that we chose very less number of hidden unit and for that reason we have also used very less number of iterations to train the network. As we had already seen in the temperature prediction we that in the last few iterations the validation error actually started increasing. To avoid the same in this case we reduced the number of iterations. Also we see from figure 6.18, that the last hidden units also spoils the generalization capability of the network, meaning only 4 hidden units were needed.

Case 3- Results for forecast of relative humidity for 51 hrs ahead in future.

Table 6.19 Subset size and input sequence from PLOFS

Optimal Subset, Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(7)$	$x_p(2)$	$x_p(25)$	$x_p(16)$	$x_p(1)$	$x_p(31)$	$x_p(38)$	$x_p(46)$	$x_p(33)$	$x_p(12)$	$x_p(23)$	$x_p(52)$	$x_p(28)$	$x_p(4)$	$x_p(5)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(36)$	$x_p(50)$	$x_p(43)$	$x_p(19)$	$x_p(10)$	$x_p(3)$	$x_p(41)$	$x_p(21)$	$x_p(34)$	$x_p(45)$	$x_p(15)$	$x_p(49)$	$x_p(18)$	$x_p(40)$	$x_p(30)$

Table 6.20 Training and Validation errors for 10 iteration

Iteration No	1	2	3	4	5	6	7	8	9	10
Training Error	0.0098	0.0095	0.0092	0.0084	0.0075	0.007	0.0062	0.0053	0.0049	0.0046
Validation Error	0.0116	0.00993	0.0091	0.0082	0.0075	0.0061	0.0058	0.0057	0.0059	0.0059

Table 6.21 Network Details

Optimum No. of Hidden units (10 Iterations)	4
Size of subset chosen from table 6.19	30
Number of KLT features used	4
Number of Hidden Units started with	5

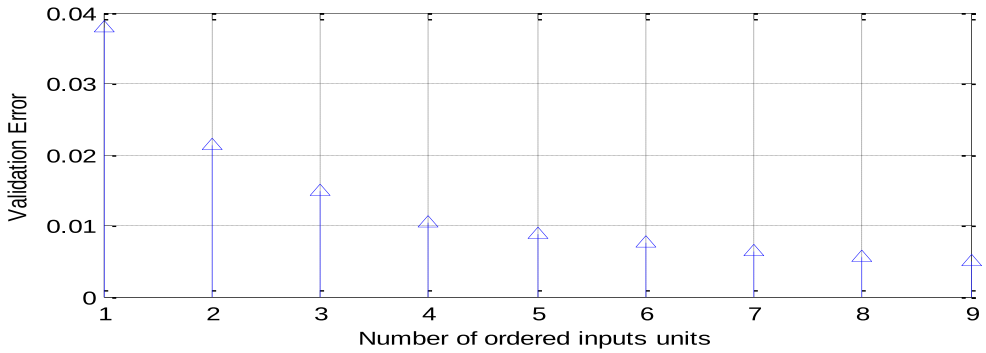


Figure 6.14 Validation Error over all the ordered input units after training for 10 iterations

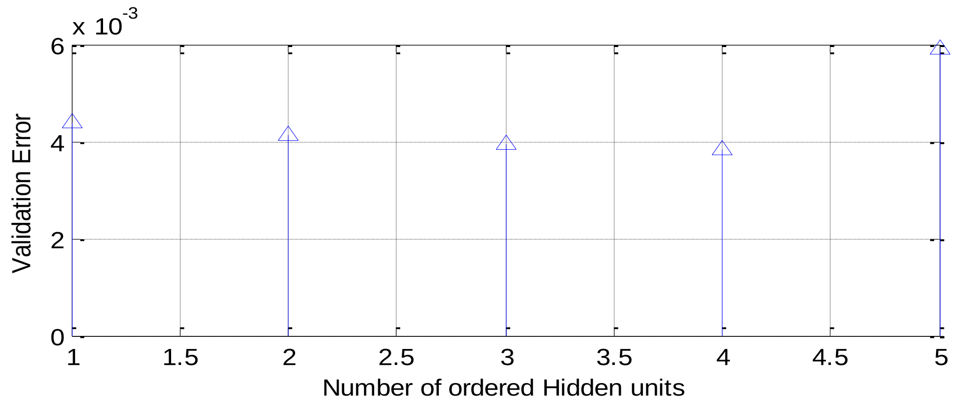


Figure 6.15 Validation Error over all the ordered hidden units after training for 10 iterations

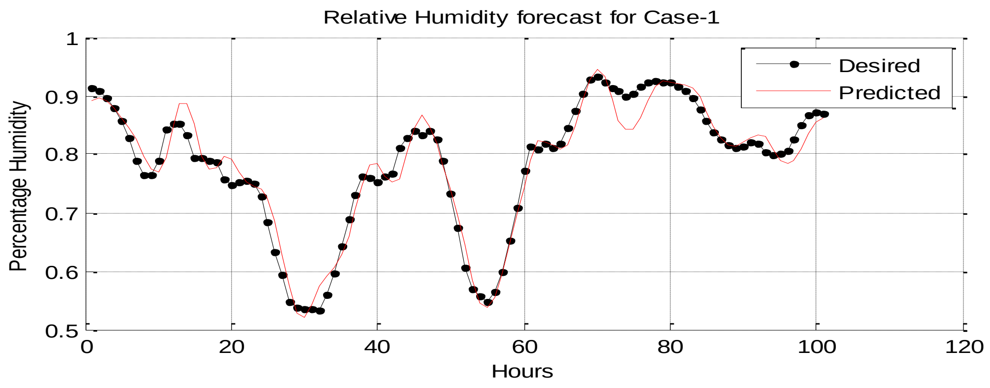


Figure 6.16 Prediction of 1 hr ahead humidity (100 samples of Validation data)

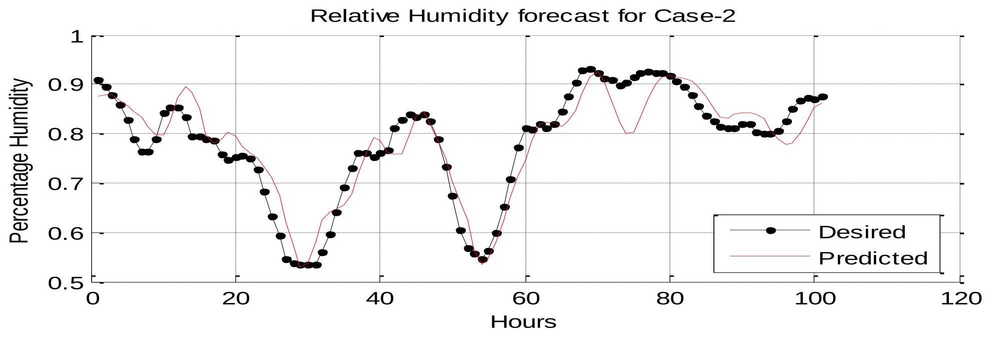


Figure 6.17 Prediction of 26 hrs ahead humidity (100 samples of Validation data)

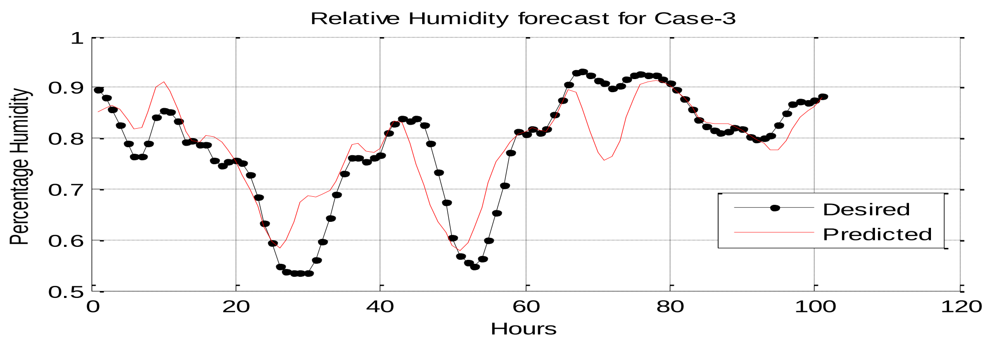


Figure 6.18 Prediction of 51 hrs ahead humidity (100 samples of Validation data)

As, will be observed that only 4 KLT coefficients did a job good enough to forecast for 51 hrs ahead. Thus reduction in number of inputs N is re-asserted. Readers may also ask that why is it we choose the same number of KLT coefficients in all the three cases, even though we are trying to predict more hours in future. The reason as we shall put it is that in chapter 4 we mentioned that the transforms concentrate all the energy of the specific pattern in the top few coefficients. But, having mentioned that, readers must notice from figure 5.6 that, when we try to predict more hours ahead in future, we do not change the input patterns at all. All that we change is the output that we try to predict. Window size being kept the same (i.e. 48, last 2 days of data), in all the 3 cases for a given variable, we get the same covariance and correlation matrix and therefore the SVD and the related KLT will be the same. Only, during the training the cross-correlation matrix is changed, using which we try to solve the equation for weights. For this reason we need to see the SVD and singular values ones only and decide on number of KLT features to be used ones only and use the same for all three cases. As such using lower or trailing KLT vectors of trailing rows of KLT matrix would add nothing but noise to the network, resulting in even more noisy prediction, which we already face in case 3 for all variables.

6.3 Results of solar radiation forecast

Case-1 –Results for forecasting solar radiation 1 hr ahead in future.

Table 6.22 Subset size and input sequence from PLOFS

Optimal Subset, Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(52)$	$x_p(29)$	$x_p(51)$	$x_p(3)$	$x_p(28)$	$x_p(4)$	$x_p(6)$	$x_p(30)$	$x_p(34)$	$x_p(49)$	$x_p(31)$	$x_p(50)$	$x_p(26)$	$x_p(5)$	$x_p(46)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(35)$	$x_p(22)$	$x_p(2)$	$x_p(36)$	$x_p(25)$	$x_p(13)$	$x_p(33)$	$x_p(32)$	$x_p(1)$	$x_p(12)$	$x_p(38)$	$x_p(40)$	$x_p(27)$	$x_p(7)$	$x_p(8)$

Table 6.23 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	0.500209	11	0.431232
2	0.488026	12	0.428533
3	0.476247	13	0.425628
4	0.4691	14	0.422797
5	0.465038	15	0.420355
6	0.463622	16	0.418359
7	0.444094	17	0.415192
8	0.437704	18	0.41201
9	0.435674	19	0.410494
10	0.433603	20	0.409515

Table 6.24 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	0.524021	11	0.454433
2	0.508625	12	0.445002
3	0.496052	13	0.452363
4	0.486233	14	0.446808
5	0.487072	15	0.458722
6	0.496919	16	0.46541
7	0.461248	17	0.460525
8	0.465909	18	0.452897
9	0.459843	19	0.450017
10	0.452154	20	0.448939

Table 6.25 Network Details

Optimum No. of Hidden units (20 Iterations)	18
Size of subset chosen from table 6.22	30
Number of KLT features used	15
Number of Hidden Units started with	20

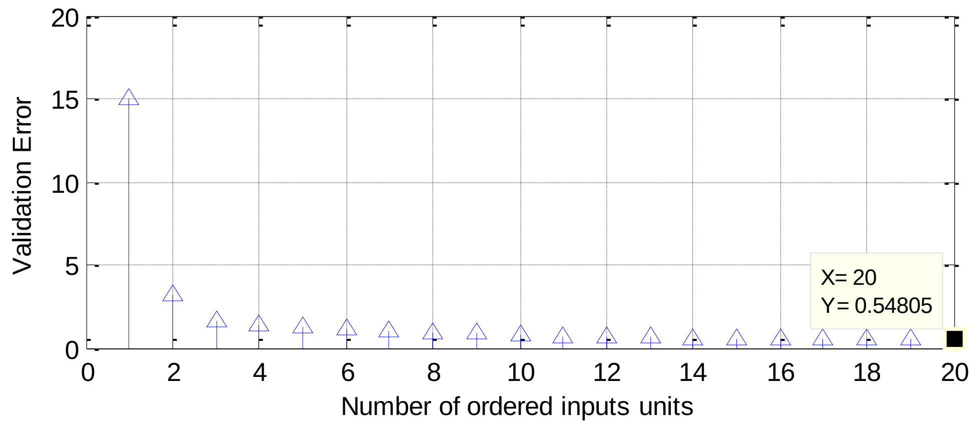


Figure 6.19 Validation Error over all the ordered input units after training for 20 iterations

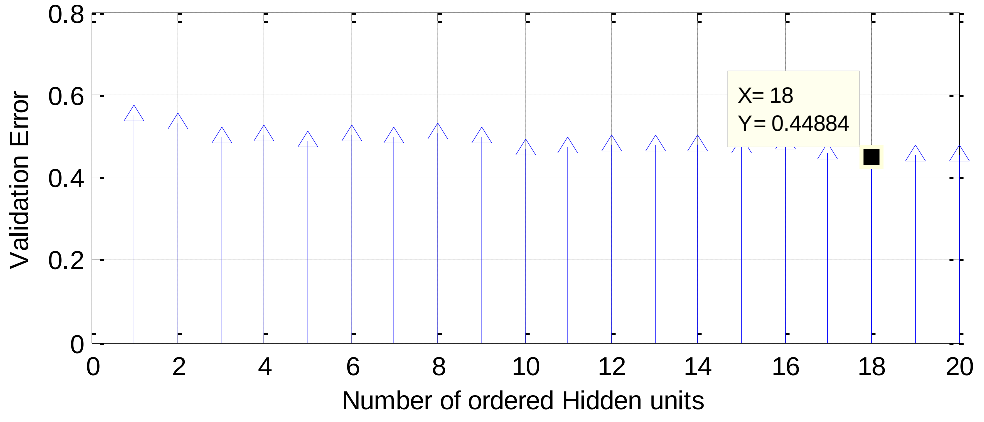


Figure 6.20 Validation Error over all the ordered hidden units after training for 20 iterations

It will noticed by now (from chapter 2 also) that the solar radiation can be considered as a sparse signal. As such there can be other method of dealing with it, but we shall use the same forecaster to treat this variable as well, reason being that we wish to propose the combined forecaster in later sections. The solar radiation is a variable that never takes any negative values. For this reason the pre-rpocessing technique we have used like separating mean is a very very necessary thign to use the data for training. Now we will see in this variable and the other one (wind) coming later how well the hidden non-linear units contribute for reducing the validation error. As was seen in the case of humidity and temperature, more work was being done by the linear part of the network as compared to the hidden units. But in the solar radiation and wind and even more, in the case of combined multivariable case the hidden units take over a lot of non-linearity and reduce a lot of validation error. We in this case also start as usual with some 20 hidden units and train for 20 iterations. But we observe that unlike temperature and humidity the optimal number of hidden units is 18 for this case. It can be also evident from the fact that the PLN validation error (PLOFS) figure 6.25 which uses only linear version of inputs has given out much larger validation error. So if only these linear inputs are used than the validation error will be large. But thanks to the non-linear sigmoid activation functions that the validation error for out neural network turns out to be very less. This proves the role of hidden non-linear units. The PLOFS like before has given impressive results though, keeping all the significant inputs in the top 25 size subset.

Case 2-Results for solar radiation forecast for 26 hrs ahead

Table 6.26 Subset size and input sequence from PLOFS

Optimal Subset, Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(30)$	$x_p(4)$	$x_p(52)$	$x_p(50)$	$x_p(6)$	$x_p(2)$	$x_p(3)$	$x_p(13)$	$x_p(28)$	$x_p(1)$	$x_p(7)$	$x_p(34)$	$x_p(31)$	$x_p(11)$	$x_p(23)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(5)$	$x_p(29)$	$x_p(51)$	$x_p(25)$	$x_p(46)$	$x_p(9)$	$x_p(26)$	$x_p(33)$	$x_p(27)$	$x_p(49)$	$x_p(38)$	$x_p(8)$	$x_p(36)$	$x_p(37)$	$x_p(35)$

Table 6.27 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	1.087428	11	0.968132
2	1.064474	12	0.961778
3	1.04974	13	0.956977
4	1.007749	14	0.949604
5	0.994567	15	0.94614
6	0.988788	16	0.938954
7	0.978217	17	0.932842
8	0.973684	18	0.928331
9	0.972095	19	0.922814
10	0.970131	20	0.918049

Table 6.28 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	1.119587	11	1.029443
2	1.119587	12	1.013127
3	1.119587	13	0.983532
4	1.036982	14	0.968703
5	1.019354	15	0.982738
6	1.023567	16	0.993526
7	1.004882	17	0.989328
8	1.026623	18	0.983823
9	1.02169	19	0.996636
10	1.030312	20	0.972607

Table 6.29 Networks Details

Optimum No. of Hidden units (20 Iterations)	20
Size of subset chosen from table 6.26	30
Number of KLT features used	15
Number of Hidden Units started with	20

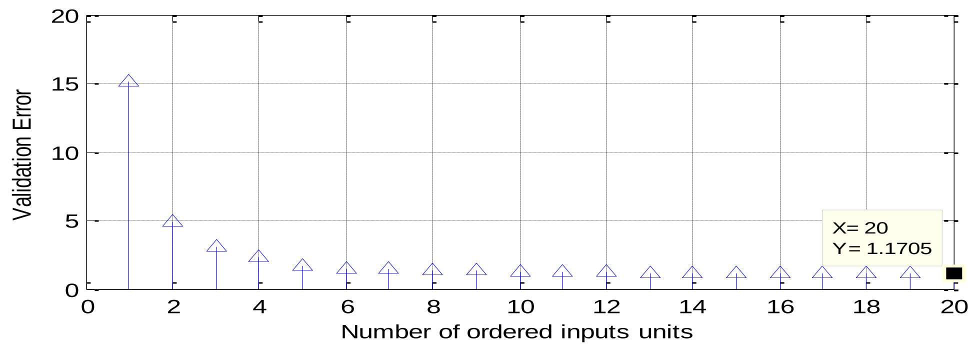


Figure 6.21 Validation Error over all the ordered input units after training for 20 iterations

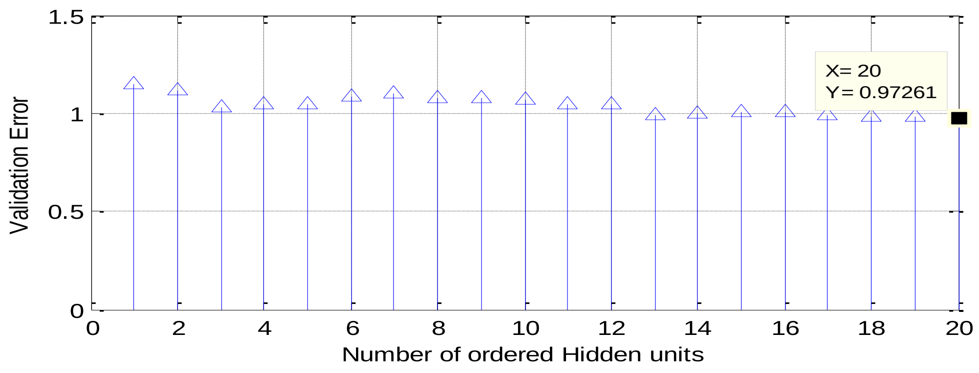


Figure 6.22 Validation Error over all the ordered hidden units after training for 20 iterations

One important fact about predicting the solar radiation is that it can also be considered as a deterministic signal (unless, cloudy weather). For this reason we see that the increase in the validation error from case-1 to case-2 in case of radiation is not so large as compared to increase from case-1 to 2 in the case of temperature. That means, it actually would be possible to predict the solar radiation for more hours in future, more than 51 hrs, without increasing the validation error largely. This is also visible from figures 6.34, 6.35, 6.36

Case-3: Results for forecast of solar radiation for 51 hrs ahead in future.

Table 6.30 Subset size and input sequence from PLOFS

Optimal Subset, Size:25												
1	2	3	4	5	6	7	8	9	10	11	12	13
$x_p(7)$	$x_p(4)$	$x_p(31)$	$x_p(2)$	$x_p(3)$	$x_p(36)$	$x_p(27)$	$x_p(1)$	$x_p(5)$	$x_p(16)$	$x_p(49)$	$x_p(50)$	$x_p(52)$
14	15	16	17	18	19	20	21	22	23	24	25	
$x_p(8)$	$x_p(29)$	$x_p(32)$	$x_p(38)$	$x_p(24)$	$x_p(13)$	$x_p(6)$	$x_p(11)$	$x_p(47)$	$x_p(9)$	$x_p(40)$	$x_p(34)$	

Table 6.31 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	1.616424	11	1.467414
2	1.59558	12	1.463199
3	1.55816	13	1.459774
4	1.544407	14	1.456223
5	1.531335	15	1.451867
6	1.518365	16	1.4482
7	1.506939	17	1.443368
8	1.495856	18	1.440422
9	1.486332	19	1.43774
10	1.474088	20	1.431583

Table 6.32 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	1.528074	11	1.456777
2	1.528074	12	1.457264
3	1.503227	13	1.456583
4	1.493017	14	1.456987
5	1.45999	15	1.459996
6	1.510201	16	1.469464
7	1.465512	17	1.47582
8	1.439538	18	1.478199
9	1.460639	19	1.483273
10	1.461744	20	1.50289

Table 6.33 Networks Details

Optimum No. of Hidden units (20 Iterations)	9
Size of subset chosen from table 6.30	25
Number of KLT features used	15
Number of Hidden Units started with	20

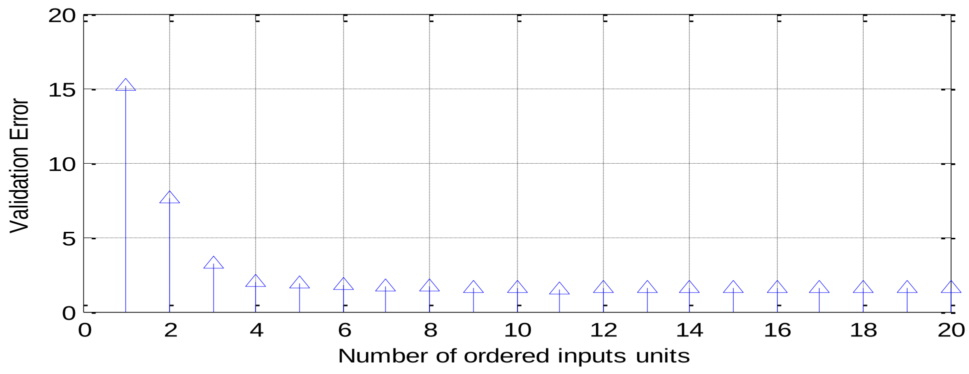


Figure 6.23 Validation Error over all the ordered input units after training for 20 iterations

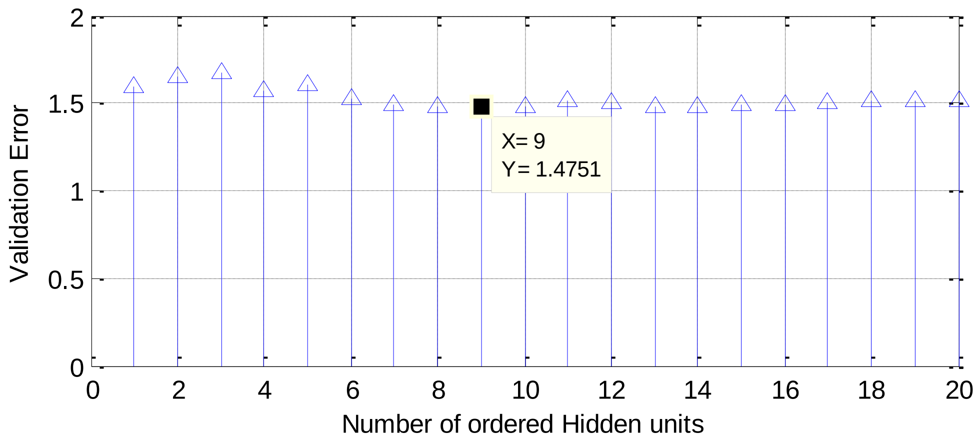


Figure 6.24 Validation Error over all the ordered hidden units after training for 20 iterations

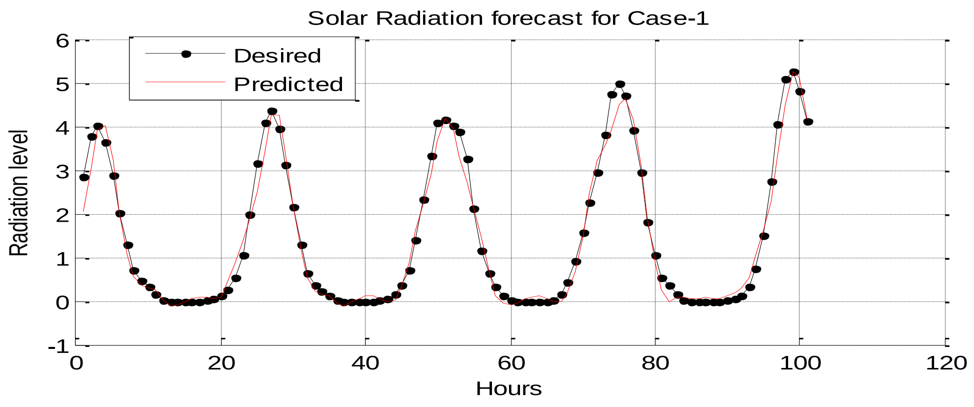


Figure 6.25 Prediction of 1 hr ahead Radiation (100 samples of Validation data)

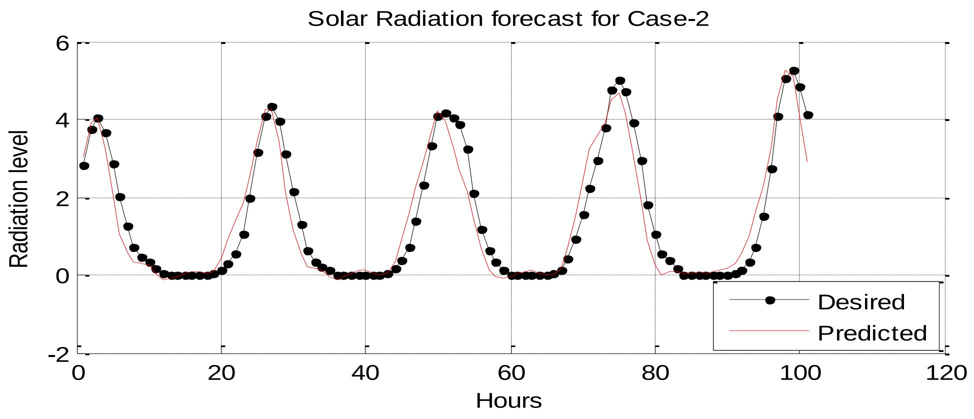


Figure 6.26 Prediction of 26 hrs ahead Radiation (100 samples of Validation data)

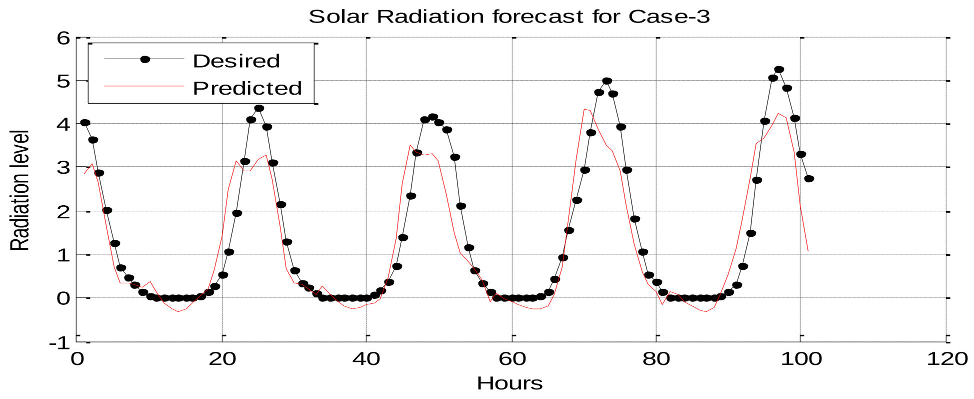


Figure 6.27 Prediction of 51 hrs ahead Radiation (100 samples of Validation data)

Important observation to be made in the case of solar radiation is that the time domain prediction error can be smartly calculated to have a smaller value. For instance, during the dark hours the radiation measurement is zero. It never goes into negative direction. Forecast for 51 hrs ahead show that the predictor does predict negative values. So in order to calculate the error, these negative values can be replaced by '0', and so the time domain error can be reduced at least for a few hours at late night (when sun is not going to be there for sure) by intuition. This kind of smart post processing approaches can really improve the prediction, as will be largely visible in prediction for wind coming next.

6.4 Results of wind forecast

The wind speed and direction are the two variables are the most randomly behaving ones. The reason can be anything from measurement technique to the local effects in the weather itself. But such behavior of a variable makes it almost impossible to predict it for long term ahead. Most of the research [70], [71], [72], has limited themselves for short term prediction for knowing few hours ahead power generation capability. Either they have limited to short term prediction, or they have only predicted speed and not direction, or they have predicted speed after averaging over whole day! These are some ways to ease the problem. But a smarter approach would always be to take the variable as it is. The wind is never measurable absolutely. It is always measured in terms of component of wind in a specific direction. Direction is attributed to the speed or else it losses all its meaning for any application. Therefore we have here combined the speed and direction together as a complex or Cartesian coordinate as shown in chapter 2. As a result of looking at speed and direction together, we eventually add two more sinusoidal inputs to the network which enhances its prediction or generalization capability.

Case 1- Results for predicting wind speed and direction for 1 hr in future.

Table 6.34 Subset size and input sequence from PLOFS

Optimal Subset, Size:26												
1	2	3	4	5	6	7	8	9	10	11	12	13
$x_p(29)$	$x_p(5)$	$x_p(52)$	$x_p(51)$	$x_p(28)$	$x_p(50)$	$x_p(45)$	$x_p(47)$	$x_p(49)$	$x_p(48)$	$x_p(46)$	$x_p(30)$	$x_p(43)$
14	15	16	17	18	19	20	21	22	23	24	25	26
$x_p(3)$	$x_p(27)$	$x_p(6)$	$x_p(4)$	$x_p(38)$	$x_p(33)$	$x_p(32)$	$x_p(37)$	$x_p(35)$	$x_p(36)$	$x_p(34)$	$x_p(1)$	$x_p(2)$

Table 6.35 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	0.645164	11	0.609351
2	0.639537	12	0.606694
3	0.636526	13	0.604414
4	0.632747	14	0.601794
5	0.629626	15	0.599241
6	0.626406	16	0.59603
7	0.621639	17	0.594304
8	0.6177	18	0.592876
9	0.613854	19	0.591232
10	0.611268	20	0.5892

Table 6.36 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	0.459965	11	0.454861
2	0.453467	12	0.451558
3	0.451086	13	0.450167
4	0.449623	14	0.452128
5	0.450346	15	0.454803
6	0.450725	16	0.460785
7	0.448589	17	0.460492
8	0.44922	18	0.463426
9	0.452291	19	0.463292
10	0.453667	20	0.464422

Table 6.37 Networks Details

Optimum No. of Hidden units (20 Iterations)	26
Size of subset chosen from table 6.34	[26;26;26]
Number of KLT features used	[13;12;12]
Number of Hidden Units started with	50

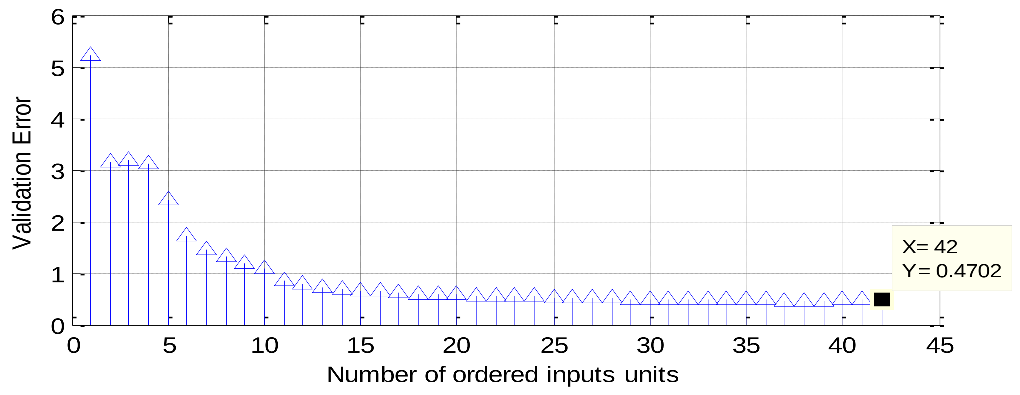


Figure 6.28 Validation Error over all the ordered input units after training for 20 iterations

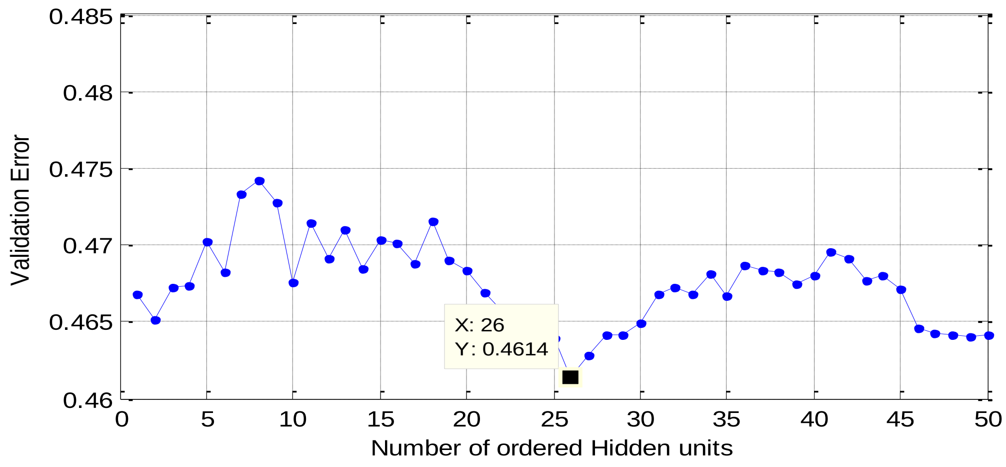


Figure 6.29 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.38 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	0.576333	0.5303
2	0.038129	0.02646
3	0.04237	0.03244

Unlike the variables like temperature and radiation, we don't expect the wind speed or direction to be a periodic signal, unless we take consideration of the fact that wind mostly blows from one specific direction in one season and that pattern is repeated every year. But such facts are known to us only and hard to represent or detect by a neural network especially if we use data only for 2 years. So the PLOFS becomes the most important part for predicting wind. The

PLOFS result of table 6.34 shows that the measurement of wind speed and direction of previous day would hardly have any effect in predicting next hour values. We see in that table that 24 hours prior measurement (i.e. 52-24) i.e. except for 27th 28th and 29th input, all other inputs in the first subset of 25 inputs come from the sequence of inputs from 52 to 30 and time related inputs. This shows us that using data of more past days would not help. But immediate past measurement would really be the determining factor for predicting future values.

Besides this, out of all 4 variables we treated till now, **wind is the only variable that has validation error less than the training error!** This comes as an approval of the fact that the wind speed and direction are together capable of reducing the randomness in individual one of them and due to this the network is able to generalize better. This also gives us a heads up to try to use more number of variables in the network that can reduce each other's randomness. For example temperature to a certain extent can be considered as random and periodic. But when used along with solar radiation measurement, its randomness is reduced because of the fact that if solar radiation is low then temperature is ought to be low in near future. This encourages us to use multi-variable model of forecaster.

Case-2: Result of wind speed and direction forecast for 26 ahead in future.

Table 6.39 Subset size and input sequence from PLOFS

Optimal Subset, Size:30														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_p(5)$	$x_p(3)$	$x_p(45)$	$x_p(4)$	$x_p(29)$	$x_p(1)$	$x_p(17)$	$x_p(21)$	$x_p(6)$	$x_p(46)$	$x_p(47)$	$x_p(25)$	$x_p(52)$	$x_p(27)$	$x_p(18)$
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$x_p(15)$	$x_p(39)$	$x_p(2)$	$x_p(43)$	$x_p(49)$	$x_p(22)$	$x_p(26)$	$x_p(16)$	$x_p(36)$	$x_p(23)$	$x_p(19)$	$x_p(41)$	$x_p(38)$	$x_p(11)$	$x_p(10)$

Table 6.40 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	1.940171	11	1.795932
2	1.919454	12	1.786014
3	1.899669	13	1.776924
4	1.886827	14	1.76929
5	1.869635	15	1.7642
6	1.857883	16	1.760392
7	1.84255	17	1.755157
8	1.828436	18	1.748657
9	1.817593	19	1.742374
10	1.807099	20	1.737772

Table 6.41 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	1.400552	11	1.366433
2	1.372843	12	1.364474
3	1.355348	13	1.365644
4	1.353551	14	1.369983
5	1.350327	15	1.373265
6	1.351293	16	1.376814
7	1.356641	17	1.380661
8	1.368408	18	1.394443
9	1.366202	19	1.399002
10	1.366783	20	1.407314

Table 6.42 Network Details

Optimum No. of Hidden units (20 Iterations)	7
Size of subset chosen from table 6.39	[30;30;30]
Number of KLT features used	[13;12;12]
Number of Hidden Units started with	50

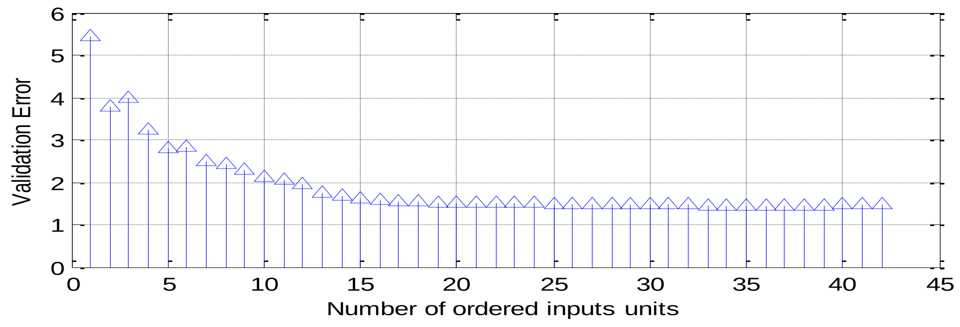


Figure 6.30 Validation Error over all the ordered input units after training for 20 iterations

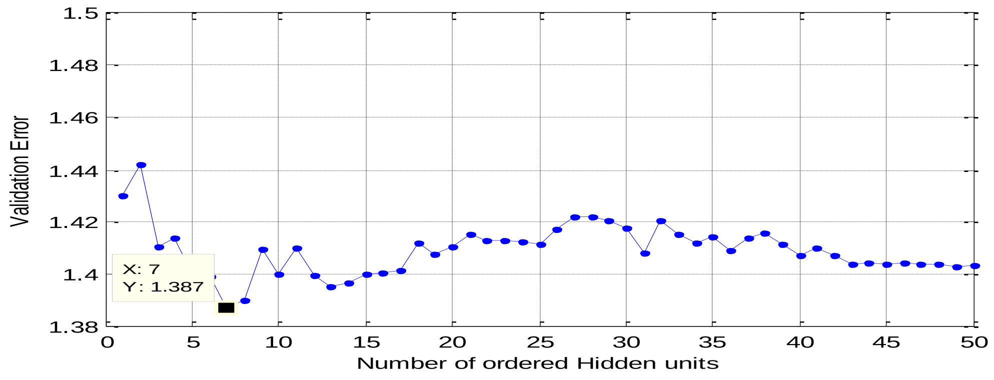


Figure 6.31 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.43 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	1.796515	1.6322
2	0.087127	0.07833
3	0.112907	0.0262

Predicting the wind for a long term ahead is the most challenging task. As mentioned before hardly any researcher has tried this before. Most of the wind predictors are based on specific Raleigh distribution that is used in areas of communication. But we shall prefer to use correlations to predict wind. There are sinusoids in the time related inputs and there is a sinusoid of direction in the data. The PLOFS results show that, these sinusoids are the most important signals in predicting the wind speed and direction ahead in time. Remembering that we don’t have the data for immediate past i.e. we are trying to predict 26 hrs ahead (solely on which wind prediction is dependent as discussed in case-1), the most important inputs that PLOFS finds is the sinusoids, in order to extrapolate the signal non-linearly.

Also it is important to understand that PLOFS algorithm has to be applied only 1 time even though the wind as a variable as 3 time series (and time-domain pattern is formed by augmenting/concatenating 3 patterns), as in figure 5.3. The reason being the fact that the same feature sequence or subset that we use for wind magnitude have to be used for the sin(direction) and cosine(direction) time series also. If we wish to use the 2 time series representation (mag·sin(direction), mag·cosine(direction)), then again the PLOFS need be applied on only one of the 2 time-series and use the same subset for the other. It is not right to modulate the sin(direction) of one time sample with the magnitude from some other time sample!

Case 3: Results of wind speed and direction prediction for 51 hrs ahead in future.

Table 6.43 Subset size and input sequence from PLOFS

Optimal Subset, Size:25												
1	2	3	4	5	6	7	8	9	10	11	12	13
$x_p(29)$	$x_p(1)$	$x_p(48)$	$x_p(27)$	$x_p(42)$	$x_p(20)$	$x_p(15)$	$x_p(44)$	$x_p(2)$	$x_p(3)$	$x_p(43)$	$x_p(24)$	$x_p(30)$
14	15	16	17	18	19	20	21	22	23	24	25	
$x_p(47)$	$x_p(49)$	$x_p(5)$	$x_p(19)$	$x_p(18)$	$x_p(41)$	$x_p(51)$	$x_p(50)$	$x_p(52)$	$x_p(4)$	$x_p(26)$	$x_p(21)$	

Table 6.45 Training error for 20 iteration

Training Error			
It No.	Value	It no.	Value
1	3.596836	11	3.337271
2	3.543535	12	3.330078
3	3.501623	13	3.318601
4	3.464042	14	3.308755
5	3.432554	15	3.296113
6	3.409379	16	3.283435
7	3.380853	17	3.275259
8	3.363957	18	3.266166
9	3.354772	19	3.254363
10	3.345871	20	3.244602

Table 6.46 Validation error for 20 iteration

Validation Error			
It no.	Value	It No.	Value
1	2.511213	11	2.401503
2	2.448337	12	2.409194
3	2.405601	13	2.452782
4	2.419438	14	2.457951
5	2.39525	15	2.466362
6	2.388655	16	2.468488
7	2.383429	17	2.500549
8	2.412824	18	2.50461
9	2.395397	19	2.593049
10	2.395802	20	2.645984

Table 6.47 Network Details

Optimum No. of Hidden units (20 Iterations)	11
Size of subset chosen from table 6.43	[25;25;25]
Number of KLT features used	[13;12;12]
Number of Hidden Units started with	50

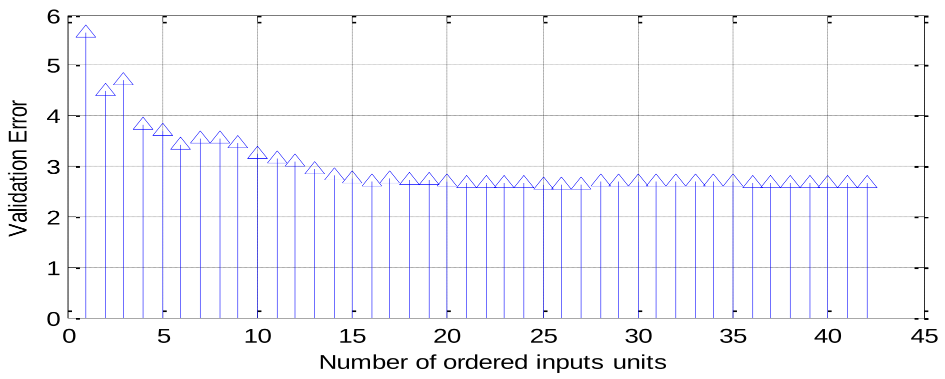


Figure 6.32 Validation Error over all the ordered input units after training for 20 iterations

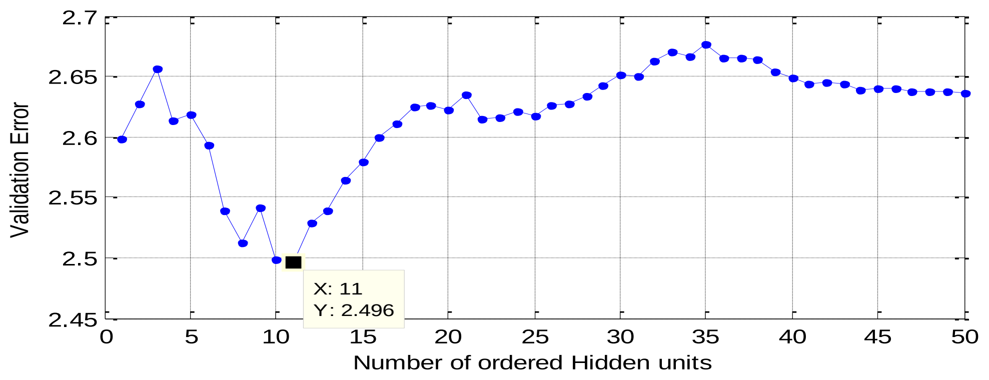


Figure 6.33 Validation Error over all the ordered hidden units after training for 20 iterations

Table 6.48 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	3.403952	3.1243
2	0.121025	0.0832
3	0.178491	0.03705

The most important observation as we mentioned before is the fact that validation error is less than training error. Elaborating this, it must be observed that the PLOFS gives us the most important inputs based on only linear capability of the inputs, without knowing about the non-linear capability of the sigmoid activation units which represent the non-linear part of the network. The tuned non-linear activation units are actually the true units that are responsible for such a good training and validation errors. It is visible from the figures 6.39, 6.42, 6.45 that the hidden units are really very actively participating as compared to the other 3 variables. It will be seen that this capability is exploited even more in the combined multivariable forecaster.

One another important observation made during wind prediction is in the KLT part. It was observed that the KLT transformed coefficients in the first 3 variables could somehow were capable to concentrate the maximum energy in the top few coefficients thereby allowing us to use less number of coefficients and less number of leading rows of KLT matrix for compression. But in the case of wind this was not possible. It was observed that in the case of wind the magnitude of the singular values did not drop drastically from first 1 to 5 or 10 values, but it gradually fell to small values at around 15th or 16th value forcing us to use more KLT features, resulting in a slightly larger network. This also implies that the energy was distributed uniformly even after transformation. This encourages us to propose a 3-stage feature selection wherein we can use PLOFS over the KLT transformed patterns to find out most significant ones (future work).

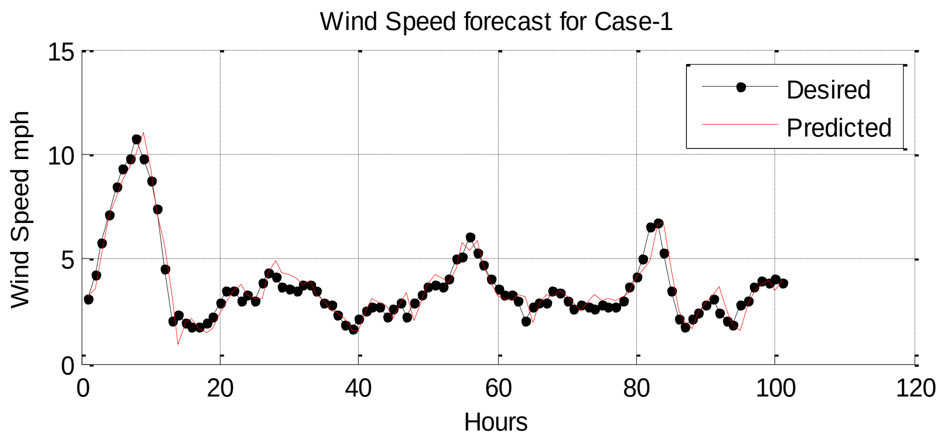


Figure 6.34 Prediction of 1 hr ahead wind speed (100 samples of Validation data)

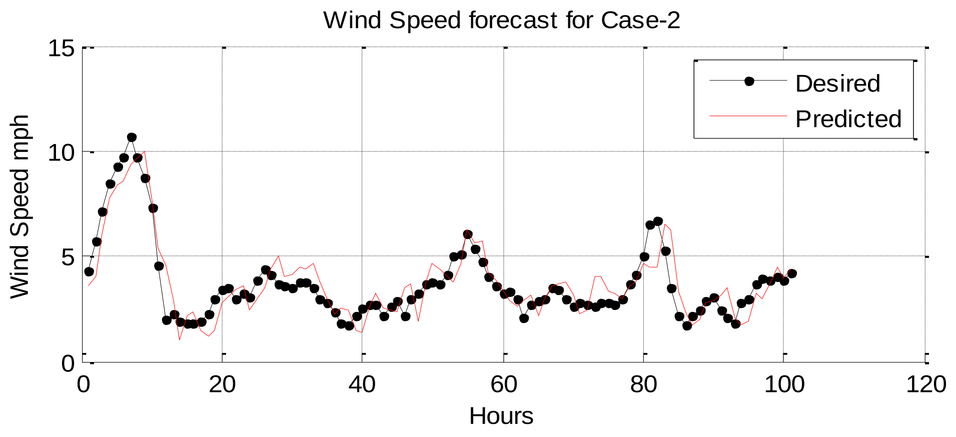


Figure 6.35 Prediction of 26 hrs ahead wind speed (100 samples of Validation data)

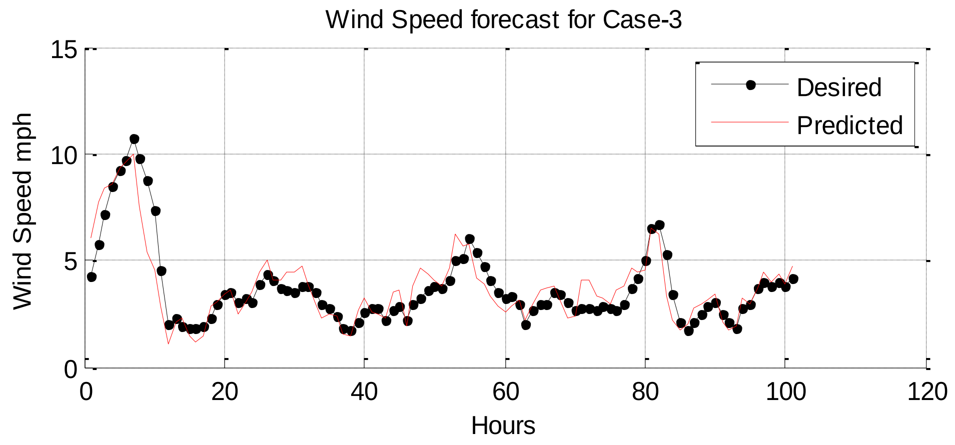


Figure 6.36 Prediction of 51 hrs ahead wind speed (100 samples of Validation data)

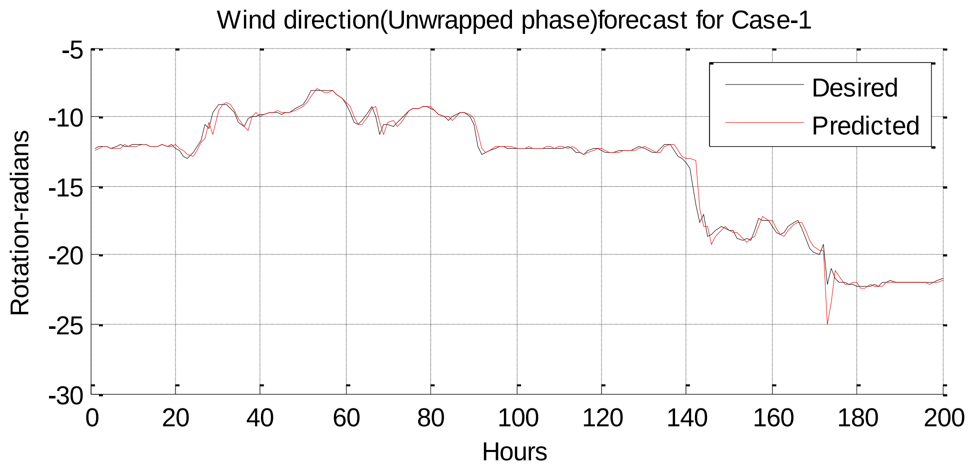


Figure 6.37 Prediction of 1 hr ahead wind direction (200 samples of Validation data)

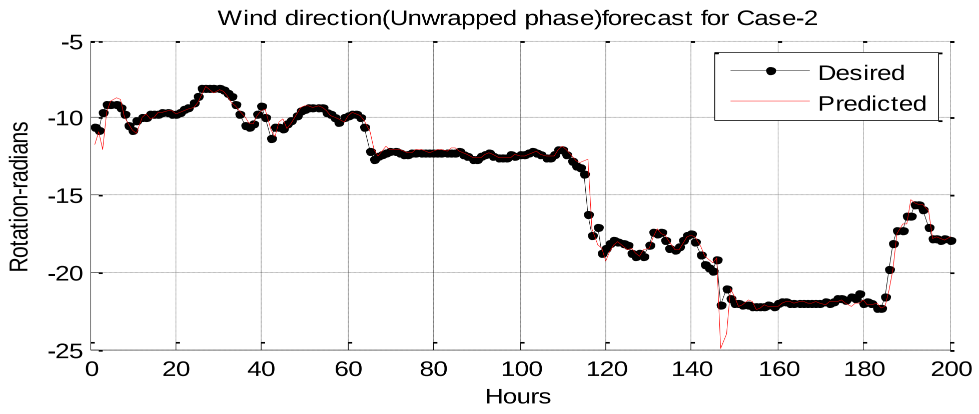


Figure 6.38 Prediction of 26 hrs ahead wind direction (200 samples of Validation data)

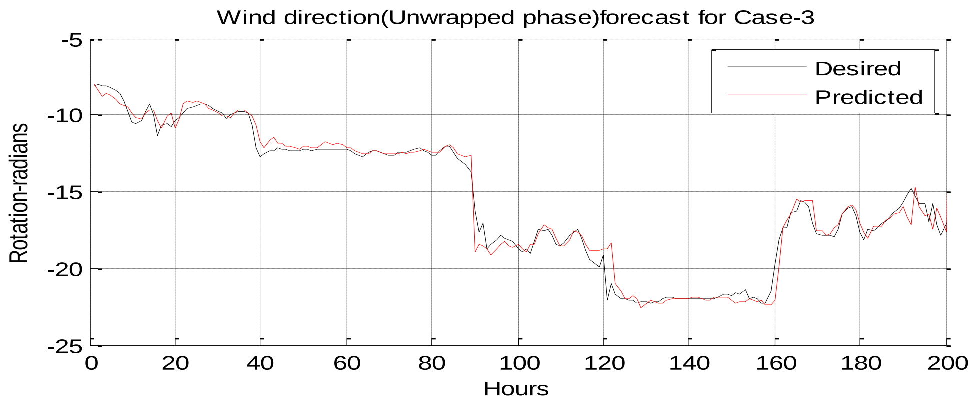


Figure 6.39 Prediction of 51 hrs ahead wind direction (200 samples of Validation data)

One most important post-processing after the prediction of the modulated signal is using the equations (11) – (16) for calculating the phase/direction from the predicted signal. More importantly is the unwrapping operation from chapter 2(following equation (16)). The wrapped version of signal has 2π discontinuities and messes up the time-domain error. So after the phase calculation we first unwrap the signal which then look like the above shown signals in figure 6.37-6.39. The fact is that the shape of the signal is predicted correctly but there are separation of multiples of 2π in the signal. As such these separations would give large errors but the error calculated need to go through the 'modulo' operation as in equation (17). This gives us true error in predicting the direction. **Error of 2π means zero error as such**. Calculating this way we get prediction error **MSE** of **0.032, 0.0734, and 0.0967** radians for case 1, 2, 3 respectively over validation data of 4000 patterns. This is the true error in predicting wind direction in time-domain. 2π separation virtually means nothing in case of phase.

6.5 Results of combined forecast

Case-1 Results of combined forecaster for 1 hrs ahead prediction

Table 6.49 Training error for 30 iteration

Training Error			
It No.	Value	It no.	Value
1	2.017251	16	1.707593
2	1.969116	17	1.697271
3	1.943677	18	1.687138
4	1.901094	19	1.677077
5	1.88552	20	1.678306
6	1.862801	21	1.640283
7	1.842501	22	1.602476
8	1.798376	23	1.588245
9	1.758308	24	1.580185
10	1.738245	25	1.574058
11	1.750168	26	1.561799
12	1.74812	27	1.537528
13	1.718101	28	1.533399
14	1.69805	29	1.509369
15	1.697906	30	1.509272

Table 6.50 Validation error for 30 iteration

Validation Error			
It no.	Value	It No.	Value
1	1.409059	16	1.170657
2	1.4061	17	1.163767
3	1.384846	18	1.15973
4	1.346546	19	1.150289
5	1.324651	20	1.138683
6	1.321546	21	1.129264
7	1.320648	22	1.123616
8	1.320326	23	1.119297
9	1.275375	24	1.109654
10	1.261466	25	1.115918
11	1.228254	26	1.124391
12	1.227268	27	1.137159
13	1.20626	28	1.13886
14	1.192423	29	1.146086
15	1.184611	30	1.153105

Table 6.51 Networks Details

Optimum No. of Hidden units (30 Iterations)	28
Subset size chosen (sections 1 to 4,case-1)	[30;30;30;26;26]
Number of KLT features used	[14;14;14;14;14]
Number of Hidden Units started with	75

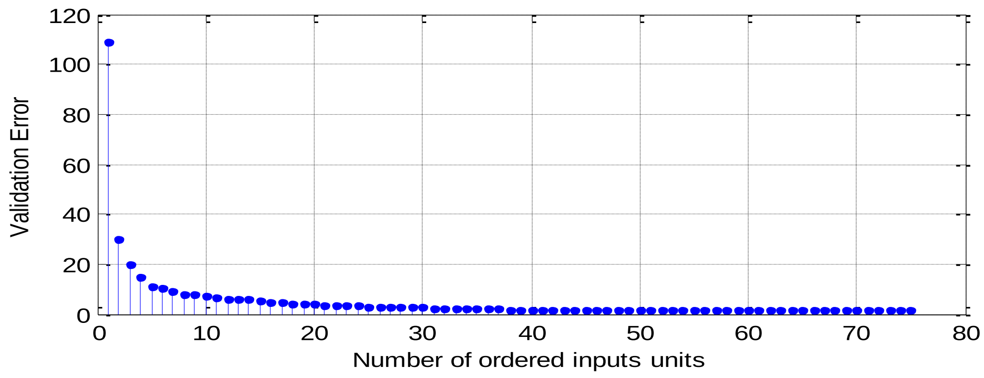


Figure 6.40 Validation Error over all the ordered input units after training for 30 iterations

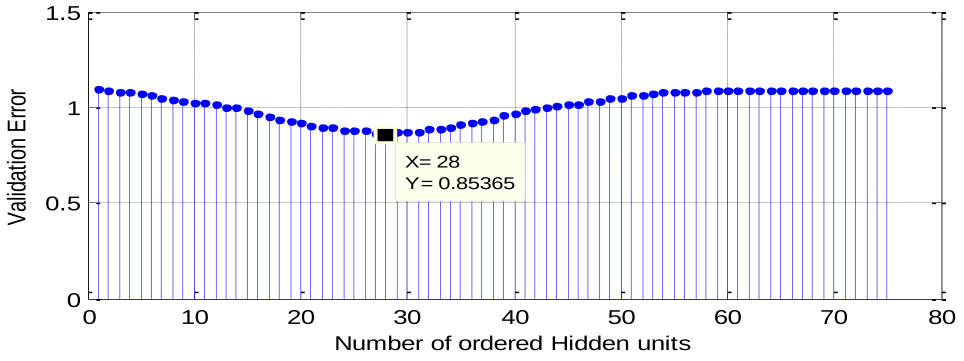


Figure 6.41 Validation Error over all the ordered hidden units after training for 30 iterations

Table 6.52 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	0.056313	0.03992
2	0.000319	0.000154
3	0.906608	0.722303
4	0.791784	0.59342
5	0.282213	0.15343

Case-2 Results of combined forecaster for 26 hrs ahead prediction.

Table 6.53 Training error for 30 iteration

Training Error			
It No.	Value	It no.	Value
1	5.914351	16	4.988424
2	5.759603	17	4.95901
3	5.683046	18	4.929673
4	5.558998	19	4.900225
5	5.514361	20	4.905591
6	5.447578	21	4.794052
7	5.387406	22	4.68238
8	5.257535	23	4.640487
9	5.198242	24	4.616698
10	5.13919	25	4.598218
11	5.115213	26	4.562512
12	5.107945	27	4.492279
13	5.019897	28	4.48046
14	4.961054	29	4.410202
15	4.960007	30	4.409783

Table 6.54 Validation error for 30 iteration

Validation Error			
It no.	Value	It No.	Value
1	4.070107	16	3.480357
2	4.072644	17	3.416742
3	3.999412	18	3.403819
4	3.914358	19	3.335639
5	3.863862	20	3.296633
6	3.842987	21	3.321941
7	3.825295	22	3.294202
8	3.82704	23	3.238772
9	3.793542	24	3.257085
10	3.753552	25	3.241282
11	3.71981	26	3.298763
12	3.65323	27	3.258297
13	3.589841	28	3.314865
14	3.568098	29	3.323453
15	3.538535	30	3.331891

Table 6.55 Network Details

Optimum No. of Hidden units (30 Iterations)	53
Subset size chosen (sections 1 to 4,case-2)	[24;25;30;30;30]
Number of KLT features used	[14;14;14;14;14]
Number of Hidden Units started with	75

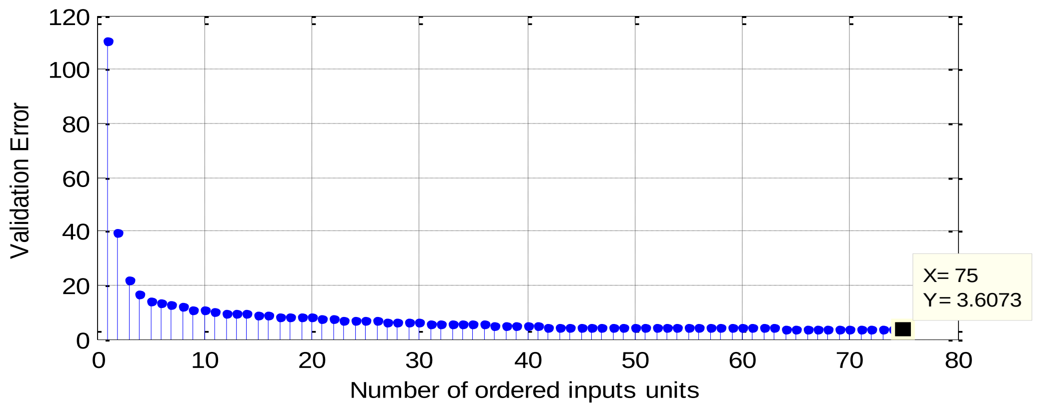


Figure 6.42 Validation Error over all the ordered input units after training for 30 iterations

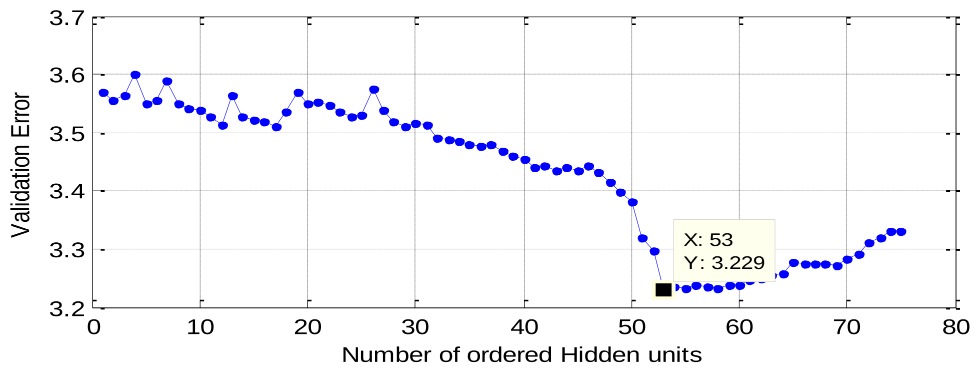


Figure 6.43 Validation Error over all the ordered hidden units after training for 30 iterations

Table 6.56 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	0.250068	0.08823
2	0.001029	0.0005023
3	2.767848	2.093476
4	2.204547	1.678387
5	0.785255	0.54946

Case 3-Results of Combined forecast for 51 hrs ahead.

Table 6.57 Training error for 30 iteration

Training Error			
It No.	Value	It no.	Value
1	10.81109	16	9.118341
2	10.54936	17	9.063071
3	10.42161	18	9.008381
4	10.18918	19	8.95404
5	10.09962	20	8.965915
6	9.967983	21	8.762045
7	9.850657	22	8.559443
8	9.610548	23	8.482797
9	9.501173	24	8.439018
10	9.392863	25	8.405575
11	9.349028	26	8.339816
12	9.337381	27	8.210076
13	9.176359	28	8.187499
14	9.068477	29	8.058636
15	9.066927	30	8.057926

Table 6.58 Validation error for 30 iteration

Validation Error			
It no.	Value	It No.	Value
1	7.303202	16	6.224528
2	7.284315	17	6.10941
3	7.220953	18	6.082698
4	7.086033	19	5.92344
5	6.96228	20	5.892548
6	6.899563	21	5.929072
7	6.828868	22	5.884271
8	6.844404	23	5.797383
9	6.772019	24	5.808396
10	6.709266	25	5.786824
11	6.639197	26	5.899751
12	6.525585	27	5.826096
13	6.425796	28	5.923734
14	6.363029	29	5.9018
15	6.317525	30	5.918126

Table 6.59 Networks Details

Optimum No. of Hidden units (30 Iterations)	58
Subset size chosen (sections 1 to 4,case-3)	[28;30;25;25;25]
Number of KLT features used	[14;14;14;14;14]
Number of Hidden Units started with	75

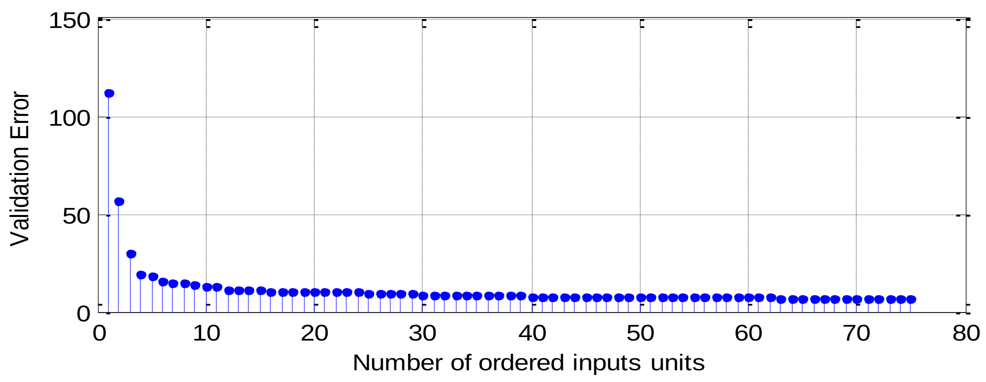


Figure 6.44 Validation Error over all the ordered input units after training for 30 iterations

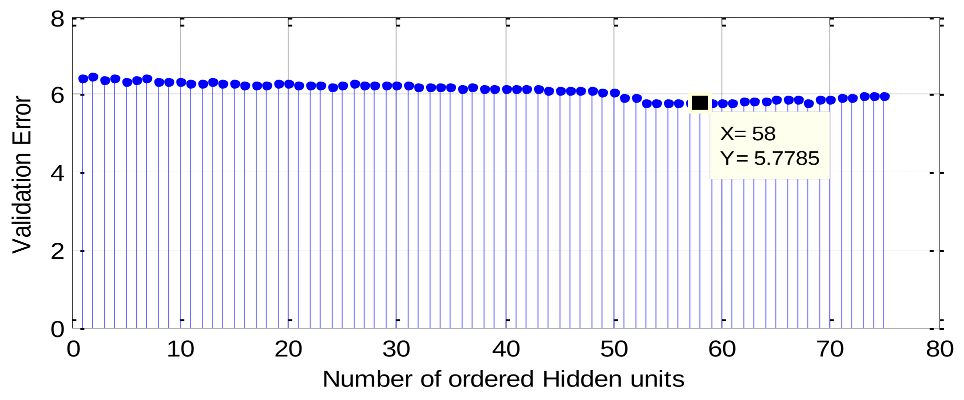


Figure 6.45 Validation Error over all the ordered hidden units after training for 30 iterations

Table 6.60 Error at individual Outputs

Error at Outputs	Before Training	After Training
1	0.669325	0.46454
2	0.002088	0.001230
3	5.14582	4.2570
4	3.823486	2.4980
5	1.387493	0.8363

Table 6.61 Reduction in number of inputs due to 2-Stage Feature Selection

Variable Number		No Feat Selection		After TDFS		After TDFS and KLT	
		N	N/N_v	N	N/N_v	N	N/N_v
Temperature	Case 1	53	0.0041	35	0.0027	9	0.00069
	Case 2	53	0.0041	29	0.0022	9	0.00069
	Case 3	53	0.0041	33	0.0025	9	0.00069
Relative Humidity	Case 1	53	0.0041	35	0.0027	9	0.00069
	Case 2	53	0.0041	30	0.0023	9	0.00069
	Case 3	53	0.0041	35	0.0027	9	0.00069
Solar Radiation	Case 1	53	0.0041	35	0.0027	20	0.0015
	Case 2	53	0.0041	35	0.0027	20	0.0015
	Case 3	53	0.0041	30	0.0023	20	0.0015
Wind Speed and Direction	Case 1	149	0.0115	83	0.0064	42	0.0032
	Case 2	149	0.0115	95	0.0073	42	0.0032
	Case 3	149	0.0115	80	0.0062	42	0.0032
Combined Forecaster	Case 1	245	0.0188	147	0.0113	75	0.0058
	Case 2	245	0.0188	144	0.0111	75	0.0058
	Case 3	245	0.0188	138	0.0106	75	0.0058

.

Table 6.62 Temperature results

Temperature	Training	Validation
case1	0.067517	0.076769
case2	0.193408	0.291136
case3	3.2102	3.981834

Table 6.63 Humidity results

Humidity	Training	Validation
case1	0.001	0.0024
case2	0.0039	0.00246
case3	0.0046	0.0059

Table 6.63 Radiation results

Radiation	Training	Validation
case1	0.409515	0.448939
case2	0.918049	0.972607
case3	1.4315	1.50289

Table 6.64 Wind results

Wind	Training	Validation
case1	0.5892	0.464422
case2	1.7377	1.4073
case3	3.2446	2.6459

Table 6.65 Sum of Error of all 4 variables

Total of top 4 Tables	Training	Validation
case1	1.067232	0.99253
case2	2.853057	2.673503
case3	7.8909	8.136524

Table 6.66 Results of Combined Forecaster

Combined Forecaster Results	Training	Validation
case1	1.509	1.1531
case2	4.4097	3.3318
case3	8.057	5.9181

Tables 6.61, 6.62, 6.63, 6.64, 6.65, 6.66 gives the comparative of individual forecasts with the results of combined forecaster. The sole reason of proposing the combined forecaster is to stress on the fact that the variables can use the correlations between each other as mention in chapter 2. For this reason **it is seen that the sum of error out of the combined forecaster is much less than the sum of individual errors** from section 6.1 to 6.4. Of course the only disadvantage is that number of iterations and number of hidden units needed are larger, giving a larger network and 5 KLT matrices need be calculated. But the advantage obtained is the fact that the validation error is less than the training error. In the cases where we have used more than one time-series for designing the forecaster i.e. in case of wind where we used 2 time-series and in section 6.5 where we used 5 time series, the validation error has been seen to be smaller than training error. Meaning that the network is able to generalize better when it has more information about more number of variable rather than when it has information only about one variable.

As mentioned in the beginning of the chapter, we have now given the prediction results of time-domain training with no feature selection. These are training results for each variable for predicting 51 hrs ahead in time. As seen, the training and validation errors are very close to the training and validation results in the KLT domain. But these networks have 53 inputs and 1 output and 50 hidden units in each network. For these large networks we have trained each network for 25 iterations. The biggest advantage obtained therefore by 2-stage feature selection is less iteration required for training and super smaller size of network as mentioned in table 6.61. The N values in the table also include ‘1’ and 4 time related inputs.

Table 6.68 Results of time domain training of temperature and humidity 51 hrs ahead prediction

Iteration	Results of training in Time-domain for Temperature 51hrs ahead		Results of training in Time-domain for Humidity 51hrs ahead	
	Training Error	Validation Error	Training Error	Validation Error
1	5.014588	5.445902	0.008587	0.007826
2	4.911325	5.304677	0.008513	0.00777
3	4.822566	5.180696	0.00847	0.00788
4	4.704014	5.413733	0.00841	0.007847
5	4.616312	5.407483	0.008368	0.007712
6	4.530208	5.183719	0.008332	0.007706
7	4.413082	5.070025	0.008302	0.007778
8	4.328221	4.901329	0.008274	0.007762
9	4.261662	4.965095	0.008249	0.007728
10	4.213797	5.027613	0.008222	0.007708
11	4.161136	4.995138	0.008198	0.00772
12	4.111347	5.023372	0.008177	0.007698
13	4.068044	5.097714	0.008157	0.007709
14	4.025581	5.046147	0.008135	0.007683
15	3.985717	5.051059	0.008112	0.007649
16	3.950987	5.070594	0.008089	0.007731
17	3.916131	5.112813	0.00807	0.00782
18	3.883218	5.090287	0.00805	0.007855
19	3.853229	5.156363	0.008024	0.007888
20	3.82012	5.207608	0.008005	0.007799
21	3.774925	5.20928	0.007987	0.007822
22	3.747402	5.097325	0.007971	0.007848
23	3.70301	5.24799	0.007953	0.007756
24	3.670579	5.328998	0.007936	0.007711
25	3.631556	5.237458	0.007919	0.007673

Table 6.69 Results of time domain training of radiation and wind speed 51 hrs ahead prediction

Iteration	Results of training in Time-domain for Radiation 51hrs ahead		Results of training in Time-domain for Wind 51hrs ahead	
	Training Error	Validation Error	Training Error	Validation Error
1	2.227062	2.014768	4.29271	3.421428
2	2.145387	1.965081	4.203846	3.341564
3	2.058502	1.930811	4.149413	3.256577
4	2.008074	1.898366	4.116617	3.252703
5	1.983001	1.955049	4.082218	3.221455
6	1.954633	1.914677	4.054211	3.211156
7	1.914488	1.871976	4.027665	3.237341
8	1.892002	1.871101	4.001767	3.240865
9	1.872907	1.875286	3.977877	3.260351
10	1.854263	1.883222	3.943098	3.302107
11	1.836385	1.884633	3.918982	3.306254
12	1.821602	1.876033	3.898319	3.307019
13	1.808233	1.872071	3.874616	3.28295
14	1.791363	1.870378	3.842655	3.322512
15	1.767766	1.828065	3.821074	3.323292
16	1.754873	1.866893	3.799424	3.36025
17	1.739033	1.855812	3.778294	3.373889
18	1.72716	1.834734	3.756669	3.379913
19	1.712663	1.81391	3.736912	3.368982
20	1.70338	1.827711	3.709551	3.382065
21	1.694652	1.838442	3.688966	3.38282
22	1.684796	1.841739	3.667017	3.425506
23	1.676353	1.883657	3.647547	3.457403
24	1.667408	1.91145	3.625133	3.446824
25	1.656075	1.902184	3.607127	3.507686

CHAPTER 7

CONCLUSION

For forecasting the weather related variables each variable needs a special treatment as pre-processing and as post-processing. The time-domain training patterns need to undergo special treatments like mean –separation and mean removal in order to make data zero-mean. The time domain training asks for larger networks and larger number of hidden units. Therefore the PLOFS enables us to get a smaller time-domain pattern which has lesser noise. This patterned can yet be de-noised by KLT compression process. Training with such a pattern in KLT domain using advanced algorithm of multiple learning factors makes the hidden space adaptable to variations in the data and thereby reduce the validation error better. Last but not the least the size of networks input space and hidden space plays a very critical role in the possibility of memorization. Smaller networks trained with larger number of smaller training patterns reduced the chances of memorization to very large extent thereby by giving us a optimal forecaster/predictor which has good accuracy of prediction and even better capability of predicting more hours in future. As such the accuracy of prediction keeps deteriorating as we try and predict more hours in future.

APPENDIX A

THEORY ON THE GRAM-SCHMIDT ORTHONORMALIZATION

Vector Space- A set is called a vector space or a linear space if it satisfies axioms as follows-

$$\mathbf{a}+\mathbf{b}=\mathbf{b}+\mathbf{a}\text{(commutative law)}$$

$$\mathbf{a}+(\mathbf{b}+\mathbf{c})=(\mathbf{a}+\mathbf{b})+\mathbf{c}\text{ (associative law)}$$

$$\mathbf{a}+\mathbf{0}=\mathbf{0}+\mathbf{a}=\mathbf{a}$$

$$\mathbf{a}+(-\mathbf{a})=\mathbf{0}$$

And for any two members of the set $\mathbf{x}$ and $\mathbf{y}$ and scalars α and β the sum- $\alpha\cdot\mathbf{x}+\beta\cdot\mathbf{y}$ also belongs to the set.

Inner products- An operator between two vectors is called inner product if it satisfies

$$(\mathbf{x},\mathbf{y})=(\mathbf{y},\mathbf{x})^{*}\text{ is the complex conjugate}$$

$$(\mathbf{x}_1+\mathbf{x}_2,\mathbf{y})=(\mathbf{x}_1,\mathbf{y})+(\mathbf{x}_2,\mathbf{y})$$

$$(c\cdot\mathbf{x},\mathbf{y})=c(\mathbf{x},\mathbf{y}),\text{ c is a scalar}$$

$$(\mathbf{x},\mathbf{x})\geq 0\text{ and}=0\text{ if and only if }\mathbf{x}=\mathbf{0}$$

In 2-dimensional vector-space $\langle\mathbf{x},\mathbf{y}\rangle=|\mathbf{x}||\mathbf{y}|\cdot\cos\theta$, and also as $\langle\mathbf{x},\mathbf{y}\rangle=\mathbf{x}_1\cdot\mathbf{y}_1+\mathbf{x}_2\cdot\mathbf{y}_1$ which is called dot product.

Norm- Norm represents the length of the given vector and is defined as-

$$\|\alpha\mathbf{x}\|=|\alpha|\|\mathbf{x}\|,$$

$$\|\mathbf{x}\|\geq 0\text{ and}=0\text{ if and only if }\mathbf{x}=\mathbf{0},$$

$$\|\mathbf{x}+\mathbf{y}\|\leq\|\mathbf{x}\|+\|\mathbf{y}\|\text{ i.e.triangular ineguality}$$

Distance-The distance d between two vectors is defined as-

$$d(\mathbf{x},\mathbf{y})=\|\mathbf{x}-\mathbf{y}\|$$

Norm of a Matrix-(Magnitude of Matrix)-The most useful way of defining the norm of matrix is to use the length of a vector associated with the matrix. Since the matrices are interpreted as the transformation from the vector **x** to another vector **y**=**A*****x** , it is natural to compare the length of **y** to **x** and define the ratio as-

$$||\mathbf{A}|| = \max\left(\frac{||\mathbf{Ax}||}{||\mathbf{x}||}\right)$$

If the vector **x** is expressed as **x** = **x**.**e**, where bold **x** is the magnitude and the **e** is the unit vector then above expression becomes-

$$||\mathbf{A}|| = \max\left(\frac{||\mathbf{Axe}||}{||\mathbf{xe}||}\right) = \max_{||\mathbf{e}||=1} ||\mathbf{Ae}||$$

Linear Independence of Base Vectors- If a set of n vectors $\mathbf{x_1},\mathbf{x_2},\mathbf{x_3},\ldots\mathbf{x_n}$ satisfies-

$$a_1\mathbf{x_1}+a_2\mathbf{x_2}+a_3\mathbf{x_3}\ldots..+a_n\mathbf{x_n}=0 \text{ if and only if } a_1=a_2=a_3=..=a_n=0,$$

then $\mathbf{x_1},\mathbf{x_2},\mathbf{x_3},\ldots.\mathbf{x_n}$ are linearly independent. Or else they are linearly dependent.

- Base Vectors-If a set of vectors $\mathbf{e_1},\mathbf{e_2},\mathbf{e_3}\ldots\ldots\mathbf{e_n}$ satisfies –

1. $\mathbf{e_1},\mathbf{e_2},\mathbf{e_3}\ldots\mathbf{e_n}$ are linearly independent
2. any arbitrary vector **v** in the set can be expressed as

$$\mathbf{v}= a_1\mathbf{e_1}+a_2\mathbf{e_2}+a_3\mathbf{e_3}\ldots..+a_n\mathbf{e_n}$$

then they form the basis or base vectors.

Best Approximation-

An approximation of the arbitrary vector **k** by a linear combination of the linearly independent vectors {$\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}\ldots\mathbf{e_n}$} as-

$$\mathbf{k} \sim \sum_{i=1}^{n} c_i\mathbf{e_i}$$

The unknown coefficients {c_1, c_2, $c_3\ldots..c_n$} can be calculated so that the distance between k and the summation is minimized-

$$\left\| \mathbf{k}-\sum_{i=1}^{n} c_i \mathbf{e_i} \right\|^2 \rightarrow \min$$

It means that –

$$\left\| \mathbf{k}-\sum_{i=1}^{n} c_i \mathbf{e_i} \right\|^2 = \langle \mathbf{k}-\sum_{i=1}^{n} c_i \mathbf{e_i}\, , \mathbf{k}-\sum_{j=1}^{n} c_j \mathbf{e_j} \rangle$$

$$= \left((\mathbf{k},\mathbf{k}) - 2\sum_{i=1}^{n} c_i (\mathbf{k},\mathbf{e_i}) + \sum_{i=1}^{n}\sum_{j=1}^{n} c_i\, c_j (\mathbf{e}_i \mathbf{e}_j) \right)$$

Differentiation with respect to the coefficients c_m-

$$0-2\sum_{i=1}^{n} \delta_{im} \langle \mathbf{k},\mathbf{e}_i \rangle + \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{im}\, c_j \langle \mathbf{e_i e_j} \rangle + \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{im}\, c_j \langle \mathbf{e_i e_j} \rangle$$

$$= -2\langle \mathbf{k},\mathbf{e_m} \rangle + \sum_{j=1}^{n} c_j \langle \mathbf{e_m},\mathbf{e_j} \rangle + \sum_{i=1}^{n} c_i \langle \mathbf{e_i},\mathbf{e_m} \rangle$$

$$= -2\langle \mathbf{k},\mathbf{e_m} \rangle + 2\sum_{j=1}^{n} c_j \langle \mathbf{e_m},\mathbf{e_j} \rangle = 0$$

$$\sum_{j=1}^{n} c_j \langle \mathbf{e_m},\mathbf{e_j} \rangle = \langle \mathbf{k},\mathbf{e_m} \rangle$$

$$\begin{bmatrix} \langle \mathbf{e_1},\mathbf{e_1} \rangle & \langle \mathbf{e_1},\mathbf{e_2} \rangle & \cdots & \langle \mathbf{e_1},\mathbf{e_n} \rangle \\ \langle \mathbf{e_2},\mathbf{e_1} \rangle & \langle \mathbf{e_2},\mathbf{e_2} \rangle & & \langle \mathbf{e_2},\mathbf{e_n} \rangle \\ \vdots & & \ddots & \vdots \\ \langle \mathbf{e_n},\mathbf{e_1} \rangle & \langle \mathbf{e_n},\mathbf{e_2} \rangle & \cdots & \langle \mathbf{e_n},\mathbf{e_n} \rangle \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle \mathbf{k},\mathbf{e_1} \rangle \\ \langle \mathbf{k},\mathbf{e_2} \rangle \\ \vdots \\ \langle \mathbf{k},\mathbf{e_n} \rangle \end{bmatrix}$$

This is a set of simultaneous equation that can be solved for vector $(c_1, c_2 \ldots . c_n)^T$. This is where the Gram-Schmidt Ortho-normalization is usable.

- Gram Schmidt Ortho-normalization- One needs to solve the above simultaneous equations by first computing the components of the matrix. The matrix is symmetrical so an efficient inverse technique like LU or Cholesky decomposition would work well.

However if each of the base vector is orthogonal and normalized as $\langle \mathbf{e_i},\mathbf{e_j}\rangle=\delta_{ij}$ i.e. is kronecker delta then the above matrix equation is simplified as-

$$\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & & 0\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix}\cdot\begin{bmatrix}c_1\\ c_2\\ \vdots\\ c_n\end{bmatrix}=\begin{bmatrix}\langle \mathbf{k},\mathbf{e_1}\rangle\\ \langle \mathbf{k},\mathbf{e_2}\rangle\\ \vdots\\ \langle \mathbf{k},\mathbf{e_n}\rangle\end{bmatrix}, \text{ i.e.} c_n=\langle \mathbf{k},\mathbf{e_n}\rangle$$

This will be the most desirable base vectors from which the coefficients can be easily calculated.

For a simplified understanding, the functions like $\sqrt{\frac{1}{(2\pi)}}, \left(\frac{1}{\sqrt{\pi}}\right)\cdot\sin x$, $\left(\frac{1}{\sqrt{\pi}}\right)\cdot\cos x$,

$\left(\frac{1}{\sqrt{\pi}}\right)\cdot\sin 2x, \left(\frac{1}{\sqrt{\pi}}\right)\cdot\cos 2x$, $\left(\frac{1}{\sqrt{\pi}}\right)\cdot\sin 3x, \left(\frac{1}{\sqrt{\pi}}\right)\cdot\cos 3x$…….. form an ortho-normal set in

$[-\pi,\pi]$. Such function expansion of a periodic function is what is Fourier series.

The steps for Gram-Schmidt ortho-normalization method for generating the ortho-normal vectos $\{\mathbf{e_1}, \mathbf{e_2}, \ldots\mathbf{e_n}\}$ from linearly independent vectors $\{\mathbf{a_1}, \mathbf{a_2}, \ldots\mathbf{a_n}\}$ are as upto certain steps.

$$\mathbf{e_1}=\left(\frac{1}{||\mathbf{a_1}||}\right)\cdot\mathbf{a_1}$$

$$\mathbf{e_2}'=\mathbf{a_2}-\langle \mathbf{a_2},\mathbf{e_1}\rangle\cdot\mathbf{e_1}$$

$$\mathbf{e_2}=\left(\frac{1}{||\mathbf{e_2}'||}\right)\cdot\mathbf{e_2}'$$

$$\mathbf{e_3}'=\mathbf{a_3}-\langle \mathbf{a_3},\mathbf{e_1}\rangle\cdot\mathbf{e}_1-\langle \mathbf{a_3},\mathbf{e_2}\rangle\cdot\mathbf{e_2}$$

$$\mathbf{e_3}=\left(\frac{1}{||\mathbf{e_3}'||}\right)\cdot\mathbf{e_3}'$$

$$\vdots$$

$$\mathbf{e_n}'=\mathbf{a_n}-\langle \mathbf{a}_n,\mathbf{e_1}\rangle\cdot\mathbf{e_1}-\langle \mathbf{a_n},\mathbf{e_2}\rangle\cdot\mathbf{e_2}\ldots\ldots\langle \mathbf{a_n},\mathbf{e_{n-1}}\rangle\mathbf{e_{n-1}}$$

$$\mathbf{e_n} = \left(\frac{1}{||\mathbf{e_n}'||}\right) \cdot \mathbf{e_n}'$$

This process of Gram-Schmidt Ortho-normalization is used in a recursive manner for solving for the output weights of the neural networks, but in a modified way. The modified version of the process is basically where we order the basis in the order of their contribution towards the function approximation, the biggest contributor being at the top and the least contributor hanging at the bottom. The basis functions for us are the tuned hidden units. So basically we try to order the hidden units using a volatile vector called ‘oe’ which has the index of hidden units in the descending order of their importance, thereby allowing us to know which one can be pruned off or removed in order to get a smaller or rather an optimal network.

REFERENCES

[1] Simon Haykin, “Neural networks-A Comprehensive Foundation,” Second Edition.

[2] P L Narasimha “Sequences of Near-Optimal Feedforward neural networks” ,Ph.D Dissertation, University of Texas at Arlington,2007

[3] Praveen Jesudhas, “Analysis and Improvement of Multiple Optimal Learning Factor for Feedforward neural network, M.S. Thesis, University of Texas at Arlington,2010

[4] S Malalur, “A Family of robust second order training algorithms” , Ph.D Dissertation, University of Texas at Arlington, 2009

[5] Kanishka Tyagi,” Second Order training Algorithms for Radial basis function Networks”, M.S Thesis, University of Texas at Arlington,2011

[6] Rohit Rawat, “An Efficient Peicewise Linear Network”, M.S Thesis, University of Texas at Arlington, 2009.

[7] Aoife M Foley, Paul G Leahy, Antonino Marvuglia, Eamon J McKeogh, “Current methods in advances In forecasting of wind power generation”, *Elsevier, Renewable Energy*, 2011,pp.1-8

[8] M Ghiassi, Stanley Nangoy, “ A dynamic artificial neural network for forecasting non-linear processes”, *Elsevier , Computer and Industrial Engineering*,2009 pp.287-297

[9] Giorgio Corani, “Air quality prediction in Milan:feedforward neural networks, pruned neural networks and lazy learning”, *Elsevier, Science Direct, Ecological Modelling 185*, 2005, pp.513-529.

[10] Guoqiang Zhang, B. Eddy Patuwo, Micheal Y Hu, “Forecasting with artificial neural networks: The state of the art. *Elsevier, Int Journal of Forecasting-14*, 1998, pp.35-62

[11] Ilaria Bertini, Francesco Ceravolo, Marco Citterio, Matteo DeFelice, Biagio Di Pietra, Francesca Margiotta, Stefano Pizzuti, Giovanni Puglisi, “ Ambient Temparature modeling

with soft Computing Techniques," *Elsevier, Science Direct, Solar Energy 84*,(2010),pp.1264-1272.

[12] Mehdi Khashei, Mehdi Bijari, "An Artificial neural network (p,d,q) model for time-series forecasting," *Elsevier, Expert systems with application 37* (2010), pp. 479-489

[13] Jiang Li, Micheal T Manry, Pramod Narasimha, Changhua Yu, "Feature Selection Using a Piecewise Linear Network," *IEEE Trans on Neural Networks*, Vol.17, No.5, September 2006, pp.1101-1115.

[14] Charytoniuk, W.; Chen, M.S, "Neural network design for short-term load forecasting," *Electric Utility Deregulation and Restructuring and Power Technologies*, 2000. Proceedings. DRPT 2000. International Conference on , vol., no., pp.554-561, 2000 doi: 10.1109/DRPT.2000.855725

[15] http://www-ee.uta.edu/EEweb/ip/new_software.html

[16] http://www.usbr.gov/pn/agrimet/webaghrread.html

[17] D E Rummelhart, G.E Hinton, R.j Williams, "Learning internal representations by error propagation," in D.E. Rummelhart and J L McClelland (Eds), *Parallel Distributed Processing*, Vol.I, Cambridge, Massachussets: The MIT Press 1986.

[18] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, "Improving the Convergence of the Backpropagation Algorithm Using Learning Rate Adaptation Methods," *Neural Computation*, Vol. 11, No. 7, Pages 1769-1796, October 1, 1999.

[19] R. Battiti, "First and Second order methods-Between steepest Descent and Newton's Method," *Neural Computation*, Vol 4, pp. 141-166,1992

[20] Sanjeev Malalur, M.T Manry, "Multiple optimal Learning factors for feed-forward networks," accepted by *SPIE Defense, Security, and Sensing(DSS) Conference*, Orlando, FL, April 2010.

[21] F .J Maldonaldo and M T Manry , “Optimal pruning of feed-forward neural networks Using the Schmidt Procedure”, *Conference Record of the Thirty Sixth Annual Asilomar Conference on Signals, Systems and Computers*., November 2002,pp 1024-1028

[22] Hema Chandrasekaran, Hung-Han Chen, Michael Manry, “Pruning of basis functions in non-linear approximators,” *Neurocomputing*, Vol.34, No1-4, 2000, pp.29-53

[23] Methaprayoon, K.; Lee, W.J.; Didsayabutra, P.; Liao, J.; Ross, R.; , "Neural network-based short term load forecasting for unit commitment scheduling," *Industrial and Commercial Power Systems*, *2003.* 2003 IEEE Technical Conference , vol., no., pp. 138- 143, 4-8 May 2003

[24] K Y lee, Y T Cha, C C Ku, “A study on neural networks for short term load forecasting,”. *Proc. 1st International forum on application of neural networks to power systems*,pp.26-30,June 23-1991, Seattle,WA.

[25] Charytoniuk, W.; Chen, M.S., "Neural network design for short-term load forecasting," *Electric Utility Deregulation and Restructuring and Power Technologies*, 2000. *Proceedings. DRPT 2000. International Conference* , vol., no., pp.554-561, 2000

[26] Hughes, G.; , "On the mean accuracy of statistical pattern recognizers," *Information Theory, IEEE Transactions* , vol.14, no.1, pp. 55- 63, Jan 1968

[27] Barron ,A.R.1992, “Neural net Approximation,” in Proceedings of Seventh Yale workshop on Adaptive and Learning Systems, pp,69-72, New Haven, CT:Yale University.

[28] M A Sartori and P J Antsaklis, “A simple method to derive bounds on the size and to train multilayer neural networks,” *IEEE Trans on Neural networks*, Vol.2, No.4, July 1991, pp.467-471.

[29] Narasimha, P.L.; Manry, M.T., Maldonado, F., "Upper Bound on Pattern Storage in Feedforward Networks," *IJCNN 2007. International Joint Conference on Neural Networks*, vol., no., pp.1714-1719, 12-17 Aug. 2007

[30] Vladimir N Vapnik, “An Overview of Statistical Learning Theory,” *IEEE trans on Neural Networks*,VOL.10, No.5, September 1999, pp.988-999

[31] Koiran,P. and E.D .Sontag,1996,”Neural networks with quadratic VC Dimension,” *Advances in Neural Information Processing Systems*,vol.8,pp.197-203,Cambridge, MA:MIT PRess

[32] Funahashi, K 1989. “On the Approximate realization of Continuous mappings by neural networks,” *Neural Networks*, vol.2, pp.183-192.

[33] Chester, D.L., 1990, “Why two hidden layers are better than one,” *International Joint Conference on Neural Networks*, vol.1,pp265-268, Washington,D.C.

[34] Jiang Li, Micheal T Manry, Pramod Narasimha, Changhua Yu, “Feature Selection Using a Piecewise Linear Network,” *IEEE Trans on Neural Networks*, Vol.17, No.5, September 2006, pp.1101-1115.

[35] Douglas F.Elliott, Kamisetty Ramamohan Rao, “Fast Transforms: Algorithms, Analyses, Applications. Academic Press.

[36] William H P, Saul A. Teukolsky, William T.Vetterling, Brian P.Flannery, “Numerical Recipes in C:Art of Scientific Computing” , Second Edition, Cambridge University Press

[37] A K Jain, “Fundamental of Digital Image Processing”, Prentice Hall Information and System Science Series.

[38] P M Narendra, Fukunaga, “A Branch and Bound Algorithm for feature subset selection,” *IEEE Trans. on Computers*. Vol. C-26, no.9, pp 917-922, Sep.1977.

[39] A D Back and T P Trappenberg, “Selecting inputs for modelling using normalized higher order statistics and independent component analysis,” *IEEE Tran. Neural Network*s, vol. 12, no.3, pp.612-617, May 2001.

[40] N Kambhatla and T L Leen, “Dimension reduction by local principal Component analysis,” *Neural Computation*. Vol. 9, no.7, pp.1493-1516,1997.

[41] J T Kwok and I W Tsang, “The pre-image problem in kernel methods.” *IEEE Trans Neural Networks*, vol. 15, no.6, pp 1517-1525. Nov.2004.

[42] M D Plumbey and E. Oja, “A non-negative PCA algorithm for independent component analysis.” *IEEE Trans Neural Networks*, vol. 15, no.1, pp. 66-76, Jan.2004.

[43] Ponnapalli, “A formal Selection and pruning algorithm for Feedforward artificial neural network optimization,” *IEEE Trans. Neural Networks*. Vol.10, no.4, pp.964-968, Jul.1999.

[44] L K Kansen and C E Rasmussan, “Pruning from adaptive regularization,” *Neural Computation*, vol. 6, no.6, pp.1223-1232, 1994.

[45] R Reed, ”Pruning algorithms-A survey,” *IEEE Trans Neural Networks*, vol. 4,no.5,pp.740-747,Sep.1993.

[46] S E Fahlman and C Lebiere. “The cascade correlation learning architecture,” in *Advances in Neural Information Processing Systems2*, 1993, pp.524-532, San Mateo, CA: Morgan Kaufmann.

[47] T Y Kwok and D Y Yeung, “Constuctive Algorithms for structure learning in Feedforward neural networks for regression problems,” *IEEE Trans. Neural Network*, Vol.8, no.3, pp.630-645, May1997.

[48] I Rivals and L Personnaz, “Neural Networks Construction and selection in Non-linear modeling*,” IEEE Trans Neural Netw*orks, vol.14, no.4, pp 804-819, Jul. 2003.

[49] G B Huang, P Saratchandran and N Sundararajan,” A generalized growing and pruning RBF (GGAP-RBF) Neural network for function approximation,” *IEEE Trans. Neural Network*,.vol.16, no.1, pp.57-67, Jan.2005.

[50] P Pudil, J. Novovicova and J. Kittler,” Floating Search methods in Feature Selection,” *Pattern Recognition Letters*, vol.15,pp 1119-1125, 1994.

[51] S D Stearns, “On Selecting Features for pattern classifiers,” *3rd Int. Conf. Pattern Recognition* , pp.71-75,1976.

[52] J. Olvera, X. Guan and M.T. Manry, "Theory of Monomial Networks," *Proceedings of SINS'92*, Automation and Robotics Research Institute, Fort Worth, Texas, pp. 96-101, December 1992.

[53] A. Azadeh, M. Sheikhalishahi, M. Tabesh, A. Negahban, “The Effects of Pre-Processing Methods on Forecasting Improvement of Artificial Neural Networks,” *Australian Journal of Basic and Applied Sciences*, 5(6):570-580,2011, pp. 570-580, ISSN 1991-8178

[54] Methaprayoon, K.; Lee, W.J.; Rasmiddatta, S.; Liao, J.; Ross, R.; , "Multi-Stage Artificial Neural Network Short-term Load Forecasting Engine with Front-End Weather Forecast," *Industrial and Commercial Power Systems Technical Conference*, 2006 IEEE , vol., no., pp.1-7, 0-0 0, doi: 10.1109/ICPS.2006.1677297.

[55] Narendra, K.S.; Parthasarathy, K.; , "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on* , vol.1, no.1, pp.4-27, Mar 1990 doi: 10.1109/72.80202

[56] P.H. Sørensen, M. Nørgaard, O. Ravn, N.K. Poulsen, “Implementation of neural network based non-linear predictive control,” *Neurocomputing*, Volume 28, Issues 1–3, October 1999, Pages 37-51, ISSN 0925-2312, 10.1016/S0925-2312(98)00114-3.

[57] Roth, M.W.; , "Survey of neural network technology for automatic target recognition," *Neural Networks, IEEE Transactions on* , vol.1, no.1, pp.28-43, Mar 1990 doi: 10.1109/72.80203

[58] Lin-Cheng Wang; Der, S.Z.; Nasrabadi, N.M.; , "Automatic target recognition using a feature-decomposition and data-decomposition modular neural network," *Image Processing, IEEE Transactions on* , vol.7, no.8, pp.1113-1121, Aug 1998

[59] John A. Bullinaria, “Modeling Reading, Spelling, and Past Tense Learning with Artificial Neural Networks,” *Brain and Language,* Volume 59, Issue 2, September 1997, Pages 236-266, ISSN 0093-934X, 10.1006/brln.1997.1818.

[60] “Understanding normal and impaired word reading: Computational principles in quasi-regular domains”. Plaut, David C.; McClelland, James L.; Seidenberg, Mark S.; Patterson, Karalyn *Psychological Review*, Vol 103(1), Jan 1996, 56-115.

[61] Benediktsson, J.A.; Swain, P.H.; Ersoy, O.K.; , "Neural Network Approaches Versus Statistical Methods In Classification Of Multisource Remote Sensing Data," *Geoscience and Remote Sensing, IEEE Transactions on* , vol.28, no.4, pp. 540- 552, Jul 1990

[62] Heermann, P.D.; Khazenie, N.; , "Classification of multispectral remote sensing data using a back-propagation neural network," *Geoscience and Remote Sensing, IEEE Transactions on* , vol.30, no.1, pp.81-88, Jan 1992

[63] Kamijo, M.; , "Classifying fingerprint images using neural network: deriving the classification state," *Neural Networks, 1993., IEEE International Conference on , vol., no*., pp.1932-1937 vol.3, 1993

[64] Jain, A.K.; Prabhakar, S.; Lin Hong; , "A multichannel approach to fingerprint classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.21, no.4, pp.348-359, Apr 1999

[65] Rowley, H.A.; Baluja, S.; Kanade, T.; , "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.20, no.1, pp.23-38, Jan 1998 doi: 10.1109/34.655647

[66] Lawrence, S.; Giles, C.L.; Ah Chung Tsoi; Back, A.D.; , "Face recognition: a convolutional neural-network approach," *Neural Networks, IEEE Transactions on* , vol.8, no.1, pp.98-113, Jan 1997

[67] Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K.J.; , "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions* on , vol.37, no.3, pp.328-339, Mar 1989

[68] Hansen, J.H.L.; Womack, B.D.; , "Feature analysis and neural network-based classification of speech under stress," *Speech and Audio Processing, IEEE Transactions* on , vol.4, no.4, pp.307-313, Jul 1996

[69] Stensrund, D. J., H. E. Brooks, J. Du, M. S. Tracton, and E. Rogers, 1999: Using ensembles for short-range forecasting. *Mon. Wea. Rev*., 127, 433–446

[70] K. Sreelakshmi, and P. Ramakanthkumar, “Neural Networks for Short Term Wind Speed Prediction” *PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY* VOLUME 32 AUGUST 2008 ISSN 2070-3740, pp.863-867.

[71] C. Pérez-Llera, M.C. Fernández-Baizan, J.L. Feito and V. González del Valle, Local Short-Term of Wind Speed: A Neural Network Analysis, *iEMSs’02*, Lugano, Switzerland, July 2002, pp.124-129

[72] C. Potter, M. Negnevitsky, and K. Jacka, "Short-term wind forecasting application using an Adaptive Neural-Fuzzy Inference System(ANFIS)", *Artificial Intelligence in Science and Technology*, 2004, pp.175-180.

BIOGRAPHICAL INFORMATION

Kunal C Vora holds a Bachelors in Electrical Engineering from the Sardar Patel University. He is currently finishing his course for M.S in Electrical Engineering at University of Texas At Arlington. Prior to his student career in U.S.A he was working in the area of power systems in India and carries his interest for signal and system analysis from that exposure.

He is currently working in the area of neural networks and weather patterns analysis. His major interests are non-linear analysis and power system quality analysis using signal processing systems and application of signal processing in control design.