12-17-2010

Weather Radar image Based Forecasting using Joint Series Prediction

Sravanthi Kattekola
University of New Orleans

Weather radar image based forecasting using joint series prediction

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Engineering
Electrical Engineering

by

Sravanthi Kattekola

B.Tech Jawaharlal Nehru Technological University, 2007
M.S University of New Orleans, 2010

December 2010

Acknowledgement

I owe my deepest gratitude to my supervisor Dr. Dimitrios Charalampidis for his patience guidance, excellent advising, invaluable assistance, and support throughout this study. I am indebted to advisory committee member Professor Kim Jovanovich for his guidance and direction. Deepest gratitude is also due to the advisory committee member Dr. Edit Bourgeois whose knowledge and assistance made this study successful.

Special thanks to my parents for their unconditional support. I would like to take this opportunity to express my profound gratitude to my brother for his understanding and encouragement.

Table of Contents

List of Figures

List of Tables

Abstract

Accurate rainfall forecasting using weather radar imagery has always been a crucial and predominant task in the field of meteorology [1], [2], [3] and [4]. Competitive Radial Basis Function Neural Networks (CRBFNN) [5] is one of the methods used for weather radar image based forecasting.

Recently, an alternative CRBFNN based approach [6] was introduced to model the precipitation events. The difference between the techniques presented in [5] and [6] is in the approach used to model the rainfall image. Overall, it was shown that the modified CRBFNN approach [6] is more computationally efficient compared to the CRBFNN approach [5]. However, both techniques [5] and [6] share the same prediction stage.

In this thesis, a different GRBFNN approach is presented for forecasting Gaussian envelope parameters. The proposed method investigates the concept of parameter dependency among Gaussian envelopes. Experimental results are also presented to illustrate the advantage of parameters prediction over the independent series prediction.

Key words: Radial Basis Function Neural Network, Forecasting, Time Series Prediction.

Chapter 1: Introduction

The nature of the atmosphere is chaotic, thus rainfall forecasting is always a challenging issue. The rainfall forecasting system is intended to assist an assortment of fields that includes marine [7], forestry [8], private sector [9] and military applications [10].

In ancient times, forecasting was mostly based on weather pattern observation. Over the years, the study of weather patterns has resulted in various techniques for rainfall forecasting. Present rainfall forecasting embodies a combination of computer models, interpretation, and an acquaintance of weather patterns. Some of the very commonly used methods are the persistence method [11], the use of a barometer [12], nowcasting [13], and use of forecast models [14], and [15]. All these methods result in reasonably accurate prediction.

Several radars were developed to observe weather patterns [16]. The development of artificial Neural Networks established an innovative trend to understand the intricate weather patterns, and therefore improve the forecasting accuracy. In this work, the weather radar data used are obtained from the NASA website [16]. The data have been collected by the WSR-88D radar located in Houston, Texas.

The thesis is organized as follows. The rest of Chapter 1 presents an overview of Radial Basis Function Networks [5], and how they are used to model the rainfall events [6]. Chapter 2 discusses rainfall forecasting using GRBFNN [17]. Chapter 3 presents the proposed work, and discusses the adjustments made to the existing forecasting algorithms. Chapter 4 presents the experimental results using simulations and actual weather precipitation events [22]. Finally, Chapter 5 concludes the thesis suggesting future work.

1.1 *Overview of Radial Basis Function Neural Networks*

In general, Radial Basis Function Neural Networks (RBFNN) consist of three layers. Each network layer consists of certain nodes. The input data are presented to the network as vectors. Each node of the hidden layer holds a suitable centroid or prototype, and each hidden node performs a nonlinear function. The response of each hidden node is multiplied with its corresponding connecting weight, and presented to the output layer. In this layer, all responses obtained from the hidden layer are combined accordingly to get the network output. The RBFNN may be trained following an iterative process in order to get the desired output. Training includes adaption of the network parameters. These parameters depend on the type of the non linear function associated to the hidden nodes. The general architecture of RBFNN is shown in Figure 1.

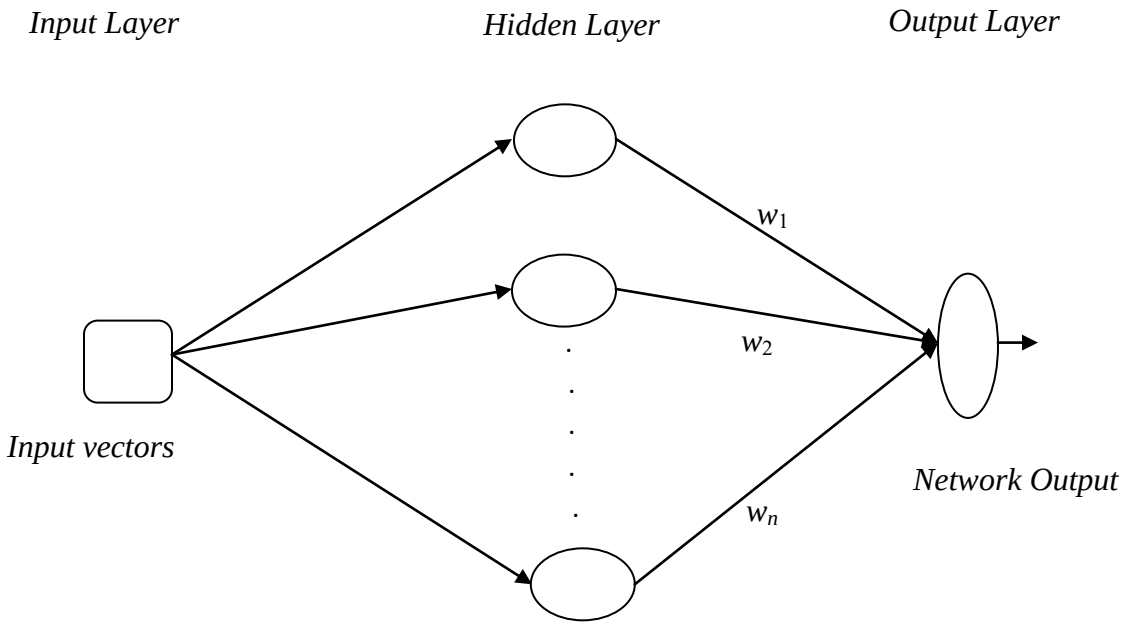


Figure 1. Radial Basis Function Neural Network architecture

1.2 *Competitive Radial Basis Function Neural Networks*

In the particular application of weather forecasting, the CRBFNN has been used to forecast the rainfall map in 2-D imaging [5]. In this approach, forecasting is performed in two steps. The first step is to approximate the rainfall events using localized functions, and in particular, Gaussian envelopes. The second step is rainfall forecasting by using the approximated Gaussian envelope parameter. For forecasting purposes, a different Neural Network called GRBFNN [17] has been used.

The CRBFNN network consists of three layers, namely, the input layer, the hidden layer, and the output layer. The network architecture is shown in Figure 2. In this approach, a rainfall image of size $N \times N$ is represented as a set of vectors $\{x_1, x_2, \ldots, x_n\}$. The vector x_i represents the pixel value $f(x_i', y_i')$ at its corresponding coordinates (x_i', y_i') of rainfall image. In order to model the rainfall image, all these vectors x_i, $1 \leq i \leq n$ are used to train the network.

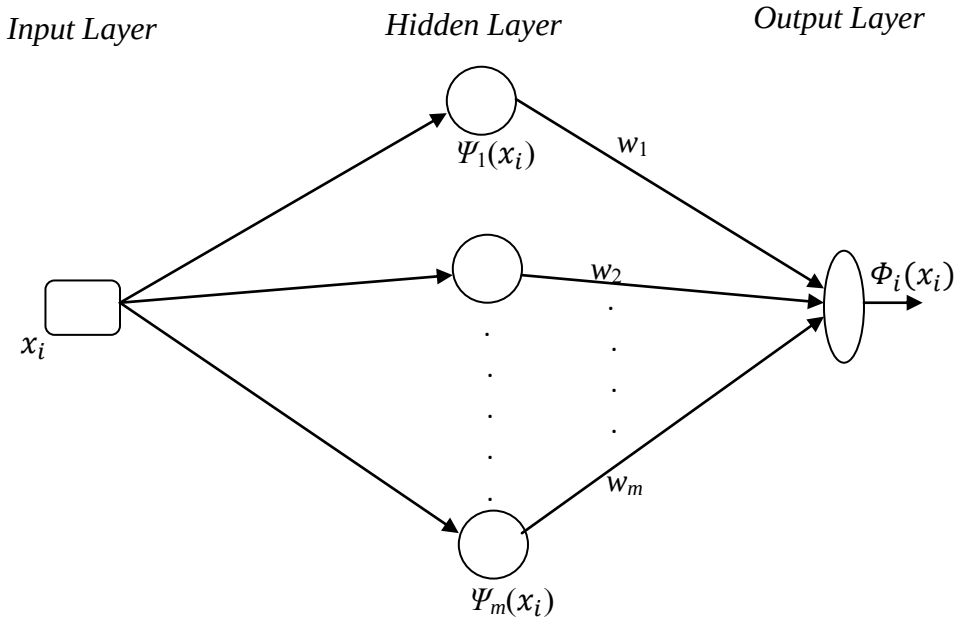


Figure 2. *Competitive Radial Basis Function Neural Network*

Assuming that the vector x_i is applied as input to the network, the response of the j^{th} hidden node for i^{th} input vector x_i is given by following equation:

$$\Psi_j(x_i) = exp\left(\frac{-d_j(x_i, p_j)}{2}\right), \quad 1 \leq j \leq m \tag{1}$$

where

$$d_j(x_i, p_j) = (x_i - p_j)^T K (x_i - p_j), \ 1 \leq j \leq m \tag{2}$$

The above equation (2) represents the Mahalanobis distance between the i^{th} input vector x_i, and centroid associated to the j^{th} node of hidden layer. Moreover, K is the inverse covariance matrix, thus, equation (1) represents a directional Gaussian function. For modeling rain events, the width of the Gaussian function is made independent along each direction using Mahalanobis distance. The CRBFNN output is the maximum of the individual hidden node responses $\Psi_1(x_i)$, $\Psi_2(x_i)$, ..., $\Psi_n(x_i)$ multiplied with their connection weights $w_1, w_2, w_3, \ldots\ldots, w_n$. In other words, only those hidden nodes whose output is close to the Gaussian centroid, in the mahanalobis sense will contribute significantly to the network output $\Phi_i(x_i)$. The network output is expressed as follows

$$\Phi_i(x_i) = max(\Psi_j(x_i) w_j + \theta_j, 1 \leq j \leq m) \tag{3}$$

Equation (3) represents the approximated rain rate i.e. i^{th} pixel value at coordinates (x_i', y_i') of the rainfall fall image. Also, n is the total number of the nodes in the hidden layer, and θ_j is the bias term associated to the j^{th} hidden node.

Based on the above discussion, the parameters required for modeling the precipitation events are the weights w_j, the inverse covariance matrix K, and the centroid p_j. From equation (3) it is also clear that the node that gives the maximum output for a point in rainfall image will be activated and trained. The training of the parameters is performed in an iterative manner using the learning rules presented in [18]. During this process, if none of the nodes in the hidden layer gives an output that exceeds a particular threshold, a new hidden node will be initialized at that point. In addition, hidden nodes which are not producing a significantly large output for a given number iterations are deleted. However, the deletion is performed only after examining all input vectors extracted from the rainfall image. The Mean Square Error (MSE) between the network's output $\Phi_i(x_i)$, and the actual output $f(x_i', y_i')$, i.e. the pixel value at coordinates (x_i', y_i') is computed. This MSE is fed back to the network to reduce the error and adopt the parameters of the network for the purpose of reducing the MSE. The iterative training process will be terminated once a specified maximum number of iterations is reached, [5] or when a specified minimum MSE is achieved [5].

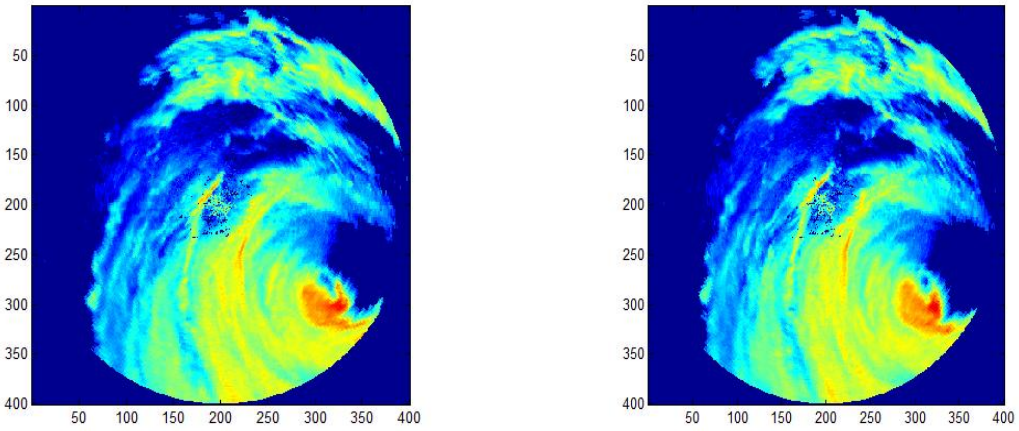
Once the network training is complete, the output image produced by the network will contain the modeled rain events. In other words, the output image is approximated as mixture of Gaussian envelopes. The Gaussian envelope parameters, including the weights, the covariance matrix, and the centroids, are used in rainfall forecasting [17].

However, the CRBFNN approach in [5] used the full resolution rainfall image size of $N \times N$ to train the network, following a pyramidal approach. Even though the pyramidal approach speeds up the process, the overall training stage is still time consuming. In order to solve this problem, an alternative CRBFNN technique [6] was proposed, which is computationally efficient. In the technique presented in [6], instead of using the full resolution weather radar image, a down sampled version of the image is used to train the network. The high resolution image is simply obtained by an extrapolation of the Gaussian parameter for higher resolution. As a result, the computational time is considerably reduced while achieving the same MSE as for CRBFNN [5].

Apart from that, in technique [6], only non-zero value pixels and a zone of zero-value pixels around the non-zero value pixels are used in the training process. Considering the zone of zero-value pixels is important because the Gaussian envelopes may assume some arbitrary values beyond the rainfall event boundaries. This selection during training of the network ensures the Gaussian function represented by the hidden nodes is restricted to the boundaries determined by the zero-value pixels zone. At the same time, the majority of zeros in the weather image are not used in the training process.

In general, as the consecutive weather images are similar, the Gaussian envelope parameters of the preceding weather image are used as starting points to model its consecutive weather image. As a result, computation time is reduced. The technique in [6] also uses the same learning rules [18] for training, but the network parameters such as weights, covariance matrix, and centers of hidden layer nodes are updated differently [6]. Once the training of the network is complete, the output image of the network will be represented as a mixture of Gaussian envelopes. The obtained envelope parameters including centers, covariance matrix, and weights are used in GRBFNN [17] for rainfall forecasting, as presented in Chapter 2.

Figure 1 illustrates a sequence of radar rainfall maps of hurricane Rita, and Figure 3 illustrates the rainfall events approximated as a mixture of Gaussian envelopes.



(a) (b)

Figure 3. *Weather radar precipitation events in consecutive two images* (*starting from* (a) *to* (b))

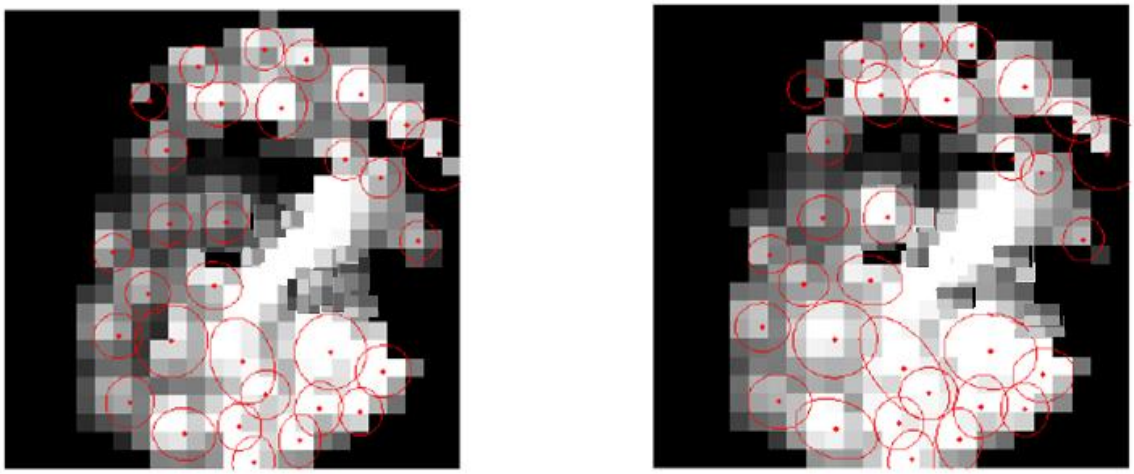


Figure 4. *Weather images approximated as mixture of Gaussian envelopes in two consecutive images* (*starting from* (a) to (b))

Chapter 2: Non-linear and non-stationary time series

Time series is defined as a sequence of successive samples of an event, which are temporally equidistant from one another without any missing observation [19]. In particular, non-linear and non-stationary time series have a mean that varies with respective time [20]. These series have an important role in real time applications such as interest rate dynamics [24], macroeconomic applications [23], and heart rate dynamics analysis [25]. Figure 6 illustrates a general time series.

2.1 *Gradient Radial Basis Function Neural Network*

GRBFNN is a three layer feed-forward Neural Network [17]. The three layers are input layer, hidden layer, and output layer. In this network, the nonlinear function associated to the hidden layer nodes is a static Gaussian Function Φ_{ij}. Each node of the hidden layer holds a suitable center c_j. In order to process the input data, the input vector z_i is compared to each node's center c_j. The GRBF network output y_i, is a sum of the individual hidden node responses multiplied with their corresponding connection weights h_j.

As mentioned in Chapter 1.2, the technique in [6] represents the rainfall events image as a mixture of Gaussian envelopes. Thus, rainfall forecasting can be done by tracking the parameters associated to the Gaussian envelopes. The Gaussian parameters include envelope centroids, the covariance matrices and the weights.

In general, the envelope parameters obtained from consecutive images can be represented as time series. Thus, forecasting can be performed by predicting the series' next value. The GRBFNN approach presented in [17] is used for this time series prediction. One reason of

employing this approach is that the hidden layer nodes of GRBFNN [17] can cope with the series which are characterized by a non-linear and non-stationary behavior.

GRBFNN [17] considers the successive sample differences of the time series as the input vector presented to the network. It analyzes the non-linear and non-stationary behavior of the series to make an accurate prediction. The network architecture is shown in Figure 5.

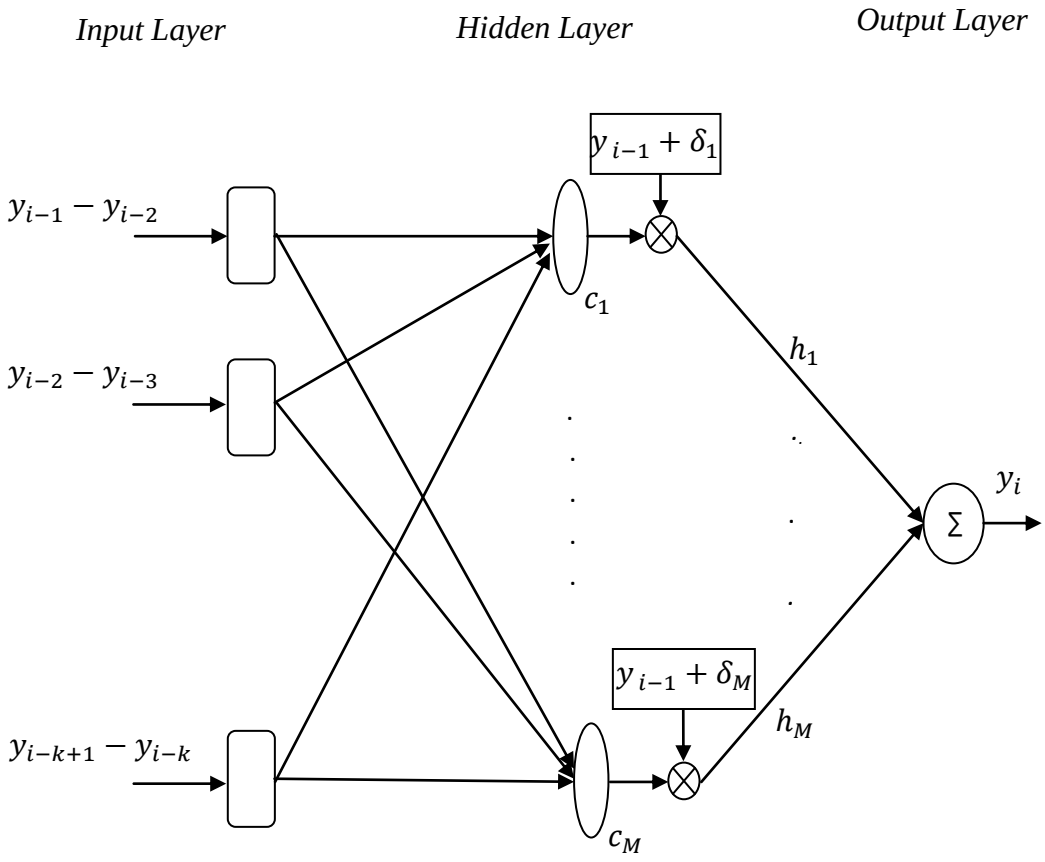


Figure 5. *Topology of the first order GRBFNN*

A general time series is shown in Figure 6. The input vector of GRBFNN, x_i, consists of *k* successive sample differences of the time series at a particular time *i* as follows:

$$x_i = [y_{i-1} - y_{i-2}\ ,\ y_{i-2} - y_{i-3}\ , \ldots\ldots\ y_{i-k+1} - y_{i-k}]^T \qquad (4)$$

where $y_{i-1}\ , y_{i-2}\ ,\ y_{i-3}\ \ldots\ , y_{i-k+1}\ ,\ y_{i-k}$ are the successive samples of the input time series. Thus, vector x_i represents the rate of change in the time series for the past *k* samples. This vector is used to predict the time series future value y_i.

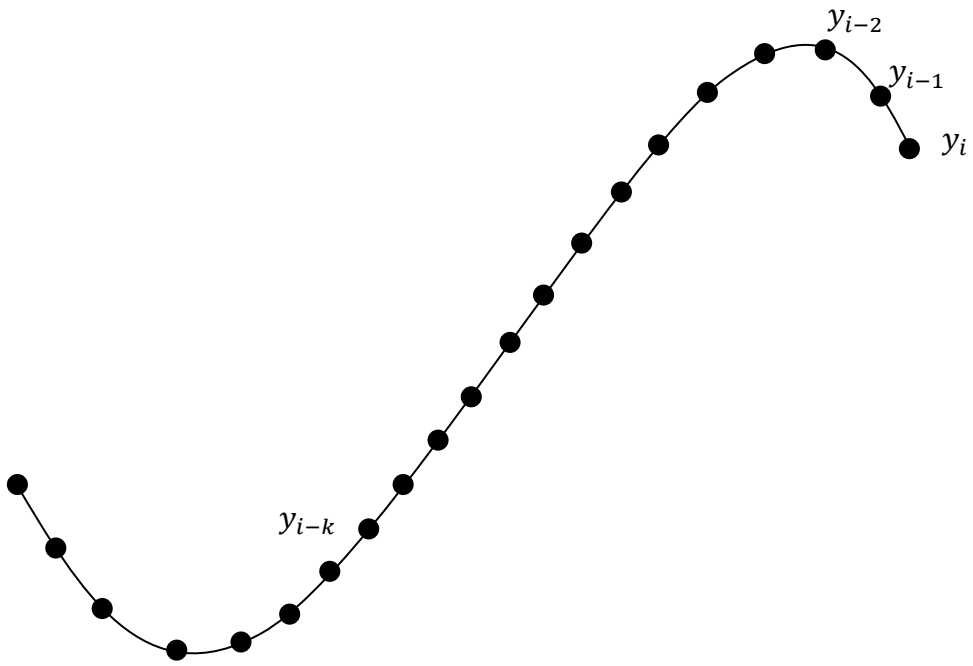


Figure 6. *Time series*

The input vector x_i is applied to the hidden layer of the GRBFNN. The j^{th} hidden node response of the network, P_j, for i^{th} input vector x_i is given by following equation

$$P_j(i) = \ \Phi_{ij} = exp\ (-\alpha\ ||x_i - c_j\ ||^2) \times \left(y_{\ i-1} + \delta_j\right), \qquad 1 \le j \le M \qquad (5)$$

In equation (5), $exp\ (-\alpha||x_i - c_j||^2)$ represents a static k-dimensional Gaussian function, and α is a scaling parameter. This function compares the similarity of the input vector x_i to the hidden node's center c_j. Here c_j is an k-dimensional center vector of j^{th} hidden node.

The additional term $(y_{i-1} + \delta_j)$ is local single step prediction of y_i by the j^{th} hidden node and δ_j is a constant value associated with the center c_j. Thus, if the input vector is very similar to the j^{th} hidden node center c_j the value of Gaussian function in equation (5) will be close to 1.0, and the predictor $(y_{i-1} + \delta_j)$ becomes active.

The response of each hidden node of GRBFNN is multiplied with its corresponding connection weight h_j, $1 \leq j \leq M$ and the weighted sum of these responses will give the network output as follows:

$$d(i) = y_i = \sum_{j=1}^{M} \Phi_{ij} h_j \tag{6}$$

The centers c_j ,and prediction terms δ_j , $1 \leq j \leq M$ can be chosen from the training data $\{x_l\}_{l=1}^{n}$. using OLS algorithm [21]. During training, for each training input vector a sample difference is defined as follows:

$$t_l = y_l - y_{l-1} \tag{7}$$

If input x_l is selected as the j^{th} center c_j, we set $\delta_j = t_l$ to ensure that the j^{th} hidden node is a perfect prediction of y_l. A detailed discussion of the OLS algorithm is presented in chapter 2.2.

2.2 *Orthogonal Least Squares algorithm (OLS)*

In order to obtain the GRBFNN centers and prediction terms, initially all the training input vectors $\{x_l\}_{l=1}^{n}$ are considered as centers of the GRBFNN. The GRBFNN output equation (6) for the training input $\{x_l\}_{l=1}^{n}$ can be expressed in the following form:

$$d(i) = y_i = \sum_{j=1}^{M} \Phi_{ij} h_j = \sum_{j=1}^{M} P_j(i)\, \theta_j \tag{8}$$

$$d = P\theta + E \tag{9}$$

where

$$d = [d(1)\ d(2) \ldots . d(n)]^T \tag{10}$$

$$\theta = [\theta_1\ \theta_2\ \theta_3 \ldots . . . \theta_M]^T \tag{11}$$

$$P_j = [p_1\ p_2 \ldots . . . . p_M] \tag{12}$$

$$p_i = [p_i(1)\ p_i(2)\ p_i(3) \ldots . . . . . p_i(n)]^T,\ 1 \le i \le M \tag{13}$$

$$E = [E(1)\ E(2) \ldots . . E(n)]^T \tag{14}$$

In equation (8) the matrix P consists of regressor vectors, p_i. The OLS algorithm transforms these regressors $p_i, 1 \le i \le M$ into orthogonal basis vectors. Thus, each regressor contribution to the output d can be determined. The regressor matrix P is can be decomposed as follows:

$$P = WA \tag{15}$$

where

$$A = \begin{pmatrix} 1 & \alpha_{12} & \alpha_{13} & \alpha_{14} & \ldots\ldots & \alpha_{1M} \\ 0 & 1 & \alpha_{23} & \alpha_{24} & \ldots\ldots & \alpha_{2M} \\ . & . & . & . & \ldots\ldots & \alpha_{M-1M} \\ 0 & 0 & 0 & 0 & \ldots\ldots & 1 \end{pmatrix} \quad (16)$$

and

$$W = [w_1\ w_2\ w_3 \ldots\ldots\ldots w_M] \quad (17)$$

In equation (16), W is an $N \times M$ matrix with orthogonal columns such that it will satisfy the following condition

$$W^T W = H \quad (18)$$

where H is the diagonal matrix with elements

$$h_i = w_i^T w_i = \sum_{t=1}^{N} w_i(t) w_i(t)\ ,\ 1 \le i \le M \quad (19)$$

Using Gram-Schmith orthogonalization [22], matrices A and W can be computed based on the procedure is described below

$w_1 = p_1$

$$\left.\begin{aligned} \alpha_{ik} &= w_i^T p_k / (w_i^T w_i),\ 1 \le i < k \\ w_k &= p_k - \sum_{i=1}^{k-1} \alpha_{jk} w_i \end{aligned}\right\} \quad k=2, \ldots\ldots., M$$

From equations (13) and (8)

$$d = Wg + E \quad (20)$$

The *OLS* solution for the non linear equation (8) is

$$g = A\theta \tag{21}$$

Based on linear regression, the g is given by

$$g_i = w_i^T d/(w_i^T w_i), \quad 1 \leq i \leq M \tag{22}$$

From the above steps, once g and A are computed, the weights of the GRBF network $\theta_1, \theta_2, \ldots \theta_M$ are obtained using equation (20).

However, in Neural Networks like GRBFNN the training data set is usually very large, and during training initially all input vectors are considered as centers of the network. In such a case, the regressor matrix P is very large consisting of M regressors. In actuality, a satisfactory model requires only M_s regressors, which is a smaller number than M ($M_s << M$). These significant regressors are selected using the OLS subset selection process [21].

The matrix W has orthogonal columns i.e. $W^T W = H$, and thus w_i and w_j are orthogonal for $i \neq j$. Therefore, the sum of squares of the GRBFFNN output d is expressed as

$$d^T d = \sum_{i=1}^{M} g^2 w_i^T w_i + E^T E \tag{23}$$

The variance of GRBFNN output d is given by

$$N^{-1} d^T d = N^{-1} \sum_{i=1}^{M} g^2 w_i^T w_i + N^{-1} E^T E \tag{24}$$

In equation (23), $N^{-1} \sum_{i=1}^{M} g^2 w_i^T w_i$ is the variance of actual output, while $N^{-1} E^T E$ is the variance of error. The error reduction ratio for GRBFNN output d is defined as:

$$[err]_i = g_i^2 w_i^T w_i / d^T d \text{ , } 1 \leq i \leq M \quad (25)$$

Using equation (24), the error reduction ratio is computed for all regressors $p_{1.\ldots}\, p_m$. The significant regressors $\{p_{1.\ldots}\, p_s\}$ are selected from the total regressors $\{p_{1.\ldots}\, p_m\}$ based on their computed error reduction ratio. This subset selection is an iterative process and is terminated once the sufficient number of regressors are found, as determined by OLS's [21] specified threshold value. The selection procedure is summarized as follows

Step 1 :

First step $k = 1$, for $1 \leq i \leq M$ compute

$$\left.\begin{aligned} w_1^{(i)} &= p_i \\ g_1^{(i)} &= (w_1^{(i)} d / ((w_1^{(i)})^T w_1^{(i)}) \\ [err]_1^{(i)} &= (g_1^{(i)})^2 w_1^{(i)})^T w_1^{(i)} / d^T d \end{aligned}\right\}$$

select

$$\left.\begin{aligned} [err]_1^{(i1)} &= \{\max [err]_1^{(i)}, 1 \leq i \leq M \} \\ w_1 &= p_{i1} \end{aligned}\right\}$$

Step 2:

At the k^{th} step where $k > 1$ for $1 \leq j \leq M$, $i \neq i_1$, …… $i \neq i_{k-1}$

$$\left.\begin{aligned} \alpha_{jk}^{(i)} &= w_j^T p_i / (w_j^T w_j) \text{ , } \quad 1 \leq j < k \\ w_k^{(i)} &= p_i - \textstyle\sum_{j=1}^{k-1} a_{jk}^{(i)} w_j \\ g_k^{(i)} &= (w_k^{(i)})^T d / ((w_k^{(i)})^T w_k^{(i)}) \\ [err]_k^{(i)} &= (g_k^{(i)})^2 w_k^{(i)})^T w_k^{(i)} / d^T d \end{aligned}\right\}$$

Find

$[err]_k^{(i_k)} = \{\max[err]_k^{(i)}, 1 \leq i \leq M, i \neq i_1, \ldots\ldots i \neq i_{k-1}\}$

select

$w_k = w_k^{(i_k)} = p_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} w_j$

$\alpha_{jk} = \alpha_{jk}^{(i_k)},\ 1 \leq j < k$

Step 3 :

The procedure is terminated at M_s step where i_k

$1 - \sum_{j=1}^{M_s} [err]_j < \rho$ where $0 < \rho < 1$ is a chosen tolerance (26)

Using this subset selection process, the significant number of regressors $\{p_1, \ldots., p_s\}$, as well as matrices g and A are determined. The weights of the GRBFNN are obtained using g and A in equation (20).

As mentioned in Chapter 2.1, In the GRBFNN approach, each approximated Gaussian envelope parameters are considered as individual time series. As a result rainfall forecasting is made once all parameters path are predicted.

However, the technique [5] has some drawbacks as discussed in Chapter 3. In order to overcome these, a new approach called joint series prediction [22] is presented in this thesis.

Chapter 3: Joint series prediction

The study of several cyclones and storms reveled that most of the times their path can be represented as a time varying non-linear and non-stationary series [5]. This kind of storm behavior is always chaotic and needs to be forecasted immediately. For such events, even though the GRBFNN [17] performance is satisfactory, an enhanced performance may be required.

The hurricane or cyclone event forecasting has always being a complex issue to understand, and uncertain to forecast an accurate rainfall. Sometimes these events' motion is fast, which needs almost an immediate prediction. There wouldn't be sufficient time to analyze the motion of events, and then to forecast their path. In such situations, the only solution would be to train and test the network in an online mode.

As discussed in Chapter 1.2, the technique in [5] represents the rainfall map as a mixture of Gaussian envelopes. This method has some shortcomings discussed below.

Primarily, the technique [5] assumes that there is no dependency among the Gaussian envelope parameters. In actuality, Gaussian motion is directly associated with the change of the envelope parameters with respect to time. Thus, there should be a dependency among all these parameters. Apart from that, it considers each Gaussian envelope parameters as individual time series and forecasts the parameters separately using GRBNN. Then, if Gaussian envelope has *n* parameters, *n* number of GRBFNN should be trained, which is an extremely time consuming process. This also results in additional computational complexity. In case of hurricanes or cyclone events, their motion has to be predicted right away, which may not be possible in this case.

In order to overcome all these shortcomings, a new concept called joint series prediction is proposed. This proposed method investigates the dependency among the envelope parameters. All envelope parameters are predicted simultaneously, so computational time is reduced.

3.1 *Proposed approach*

Assume that each approximated Gaussian envelope has *N* number of parameters. The input vector contains the successive sample difference of all *N* parameters and is given by z_i:

$$z_i = [x_i^{(1)}\ x_i^{(2)} \ldots\ldots\ldots\ x_i^{(N)}] \tag{27}$$

where

$$x_i^{(j)} = [y_{i-1}^{(j)} - y_{i-2}^{(j)}\ y_{i-2}^{(j)} - y_{i-3}^{(j)} \ldots\ldots\ldots\ y_{i-k+1}^{(j)} - y_{i-k}^{(j)}],\ 1 \le j \le N$$

This input is applied to the hidden layer of the GRBFNN. The j^{th} hidden node response of the GRBF network for the i^{th} input is given by

$$\Phi_{ij}^{(n)} = exp\ (-\alpha||z_i - c_j\ ||^2) \times \left(y_{i-1}^{(n)} + \delta_j^{(n)}\right), 1 \le n \le M \tag{28}$$

Here c_j is an K-dimensional center vector and $\delta_j^{\wedge} = [\delta_j^{(1)}\ \delta_j^{(2)}\ \delta_j^{(3)}\ \ldots\ \delta_j^{(N)}], 1 \le j \le M$ is a constant vector associated with the center c_j .The response of each hidden node is multiplied with its connecting weights

$$h_j^{\wedge} = [h_j^{(1)}\ h_j^{(2)}\ h_j^{(3)} \ldots\ldots\ldots h_j^{(N)}],\ 1 \le j \le M \tag{29}$$

The weighted sum of these responses provides the network output as follows:

$$\hat{y_j} = [y_i^{(1)}\ y_i^{(2)}\ y_i^{(3)} \ldots\ldots y_i^{(N)}]$$

$$y_i^{(n)} = \sum_{j=1}^{N} \Phi_{ij}^{(n)} h_j^{(n)}\ ,\ 1 \le n \le N \tag{30}$$

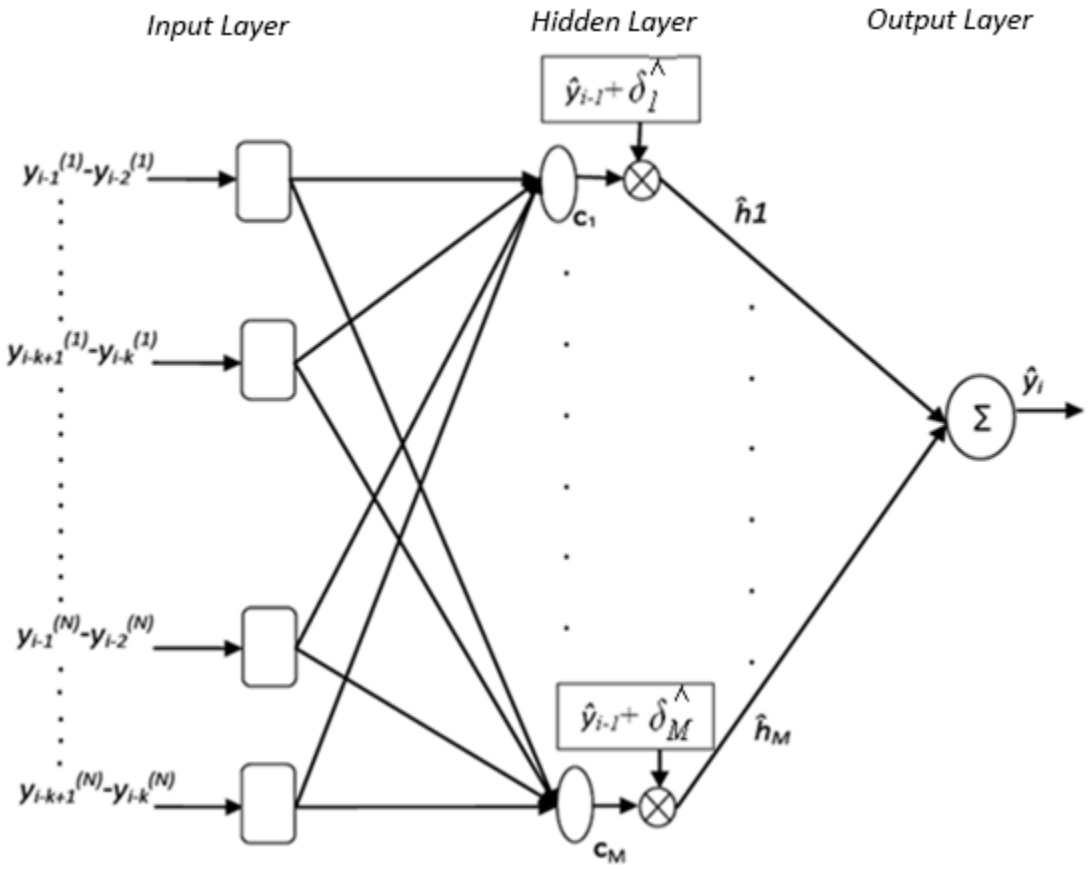


Figure 7. *Joint series prediction using GRBFNN*

The centers, c_j, and the prediction terms, δ_j, $1 \le j \le M$ are chosen using the OLS subset selection process from the training data set $\{z_l\}_{l=1}^{n}$. During the subset selection process, initially all input vectors of the training dataset $\{z_l\}_{l=1}^{n}$ are considered as GRBFNN centers. However, during the subset selection process $\{z_l\}_{l=1}^{n} - M\}$ centroids are eliminated based on their error reduction ratio, and finally, the algorithm provides the optimal number of centers *M* that are used

for prediction. For each training input vector $\{z_l\}_{l=1}^{n}$, a sample difference is defined and is given by:

$$t_i^{(n)} = y_i^{(n)} - y_{i-1}^{(n)}, 1 \leq n \leq N \tag{31}$$

If input z_i is selected as the j^{th} center c_j for the n^{th} time series , we set $\delta_j^{(n)} = y_i^{(n)} - y_{i-1}^{(n))}$ to ensure that the j^{th} hidden node is a perfect predictor of $y_k^{(n)}$. Therefore, the value of $\delta_j^{(n)}$ is equal to the value of $\delta_j^{(n)}$ if vector z_i is not eliminated from the list of centroids, and is selected as the j^{th} centroid, c_j.

Although the proposed method uses the same non-linear function $exp\ (-\alpha\ ||z_i - c_j\ ||^2)$ which was also used in the previous method [17], the weights $h_j^{(n)}$ and the prediction term $\left(y_{i-1}^{(n)} + \delta_j^{(n)}\right)$ used are different for different time series. This is one of the two adjustments made in the proposed joint series prediction. This modification contributed a better performance, because the GRBFNN Gaussian functions are close to 1 if an only if all N time series vectors $x_i^{(n)},\ 1 \leq n \leq N$ are closely located to the centroid. Thus, it can be argued that if the rainfall events are moving on a particular pattern, then there might a dependency among the position of the event (represented by Gaussian envelope centroids), and the size and directionality of the event (represented by the Gaussian covariance matrix).

The second adjustment of the joint series prediction is the center selection process. Although the same OLS algorithm is used to select the centers, during the selection process a center is selected based on its error reduction ratio associated to all N parameters i.e.

$$ERR_{total} = \sum_{j=1}^{N} ERR^{(j)} \tag{32}$$

It can be argued that the selected center will allows the GRBFNN to have a better overall performance if all N Gaussian envelope parameters are used instead of a just single parameter, in the case where there is some dependence between the envelope parameters.

3.2 *Discussion*

As mentioned earlier, the input vector of the GRBFNN in the joint series prediction contains the successive sample differences of all parameters. During the training of GRBFNN, this merging plays a crucial role, because centers have knowledge of the whole motion of Gaussian envelopes as they can see the dependency among all N parameters of Gaussian envelopes.

The scaling parameter α is used in the non-linear function of GRBFNN centers. This parameter is inversely proportional to the variance, and its value is assumed to be same for all centers. However, this parameter plays a significant role in the prediction process.

Parameter α should be directly dependent to the number of GRBFNN centers. If the value of α is large, it implies that the Gaussian envelopes have a small variance, which in turn implies that the Gaussian function is narrow. Therefore, if a large α value is used, then a larger number of centers should be considered. Since each Gaussian function is the main representative function around its own centroid, if small α value is used, then it results in a significant overlap among the closely located Gaussian functions.

The value of the parameter α is also related to the value of successive sample differences in the input vector. If successive sample differences are large, then they can afford a large Gaussian variance and thus a small α.

The value of α is also depends on the size of the Gaussians of the GRBFNN. If the GRBFNN center contains a large number of elements, a mismatch between all the elements of the successive difference vector and the centroid will result in a large Euclidean distance, and thus a small Gaussian function value. For this reason, the joint series prediction is producing a larger signal to noise ratio for smaller α value compared to the previously used method [6].

Chapter 4: Simulations

In this chapter, the experimentation results are presented for simulated and also actual weather image data.

The joint series prediction and the independent series prediction were compared using simulations in order to examine a specific motion pattern that was not available in actual weather-image based data. While experimenting with simulations, it's not consistent to include all envelope parameters without knowledge of their dependency. Thus, simulations are concentrated only on the joint prediction of the Gaussian envelope centroid coordinates (horizontal and vertical). Based on these simulations the following conclusions were drawn.

4.1 *Motion Pattern*

The joint series and independent series predictor network (predicting parameters separately using separate GRBFNN) were tested with different kind of motion patterns. Initially, when they were tested with linear moving patterns, it was found that the performance of both joint series predictor and independent series predictor was mostly similar. The reason could be the lack of significant variations in the motion pattern.

Moreover, the joint series predictor network and independent series predictor network were tested with circular motion patterns which have a significant variation in their motion pattern. The kind of motion was considered to be a counter clock wise rotation, coupled with an almost linear overall movement of the overall circulations, which is similar to the motion of cyclones and hurricane events.

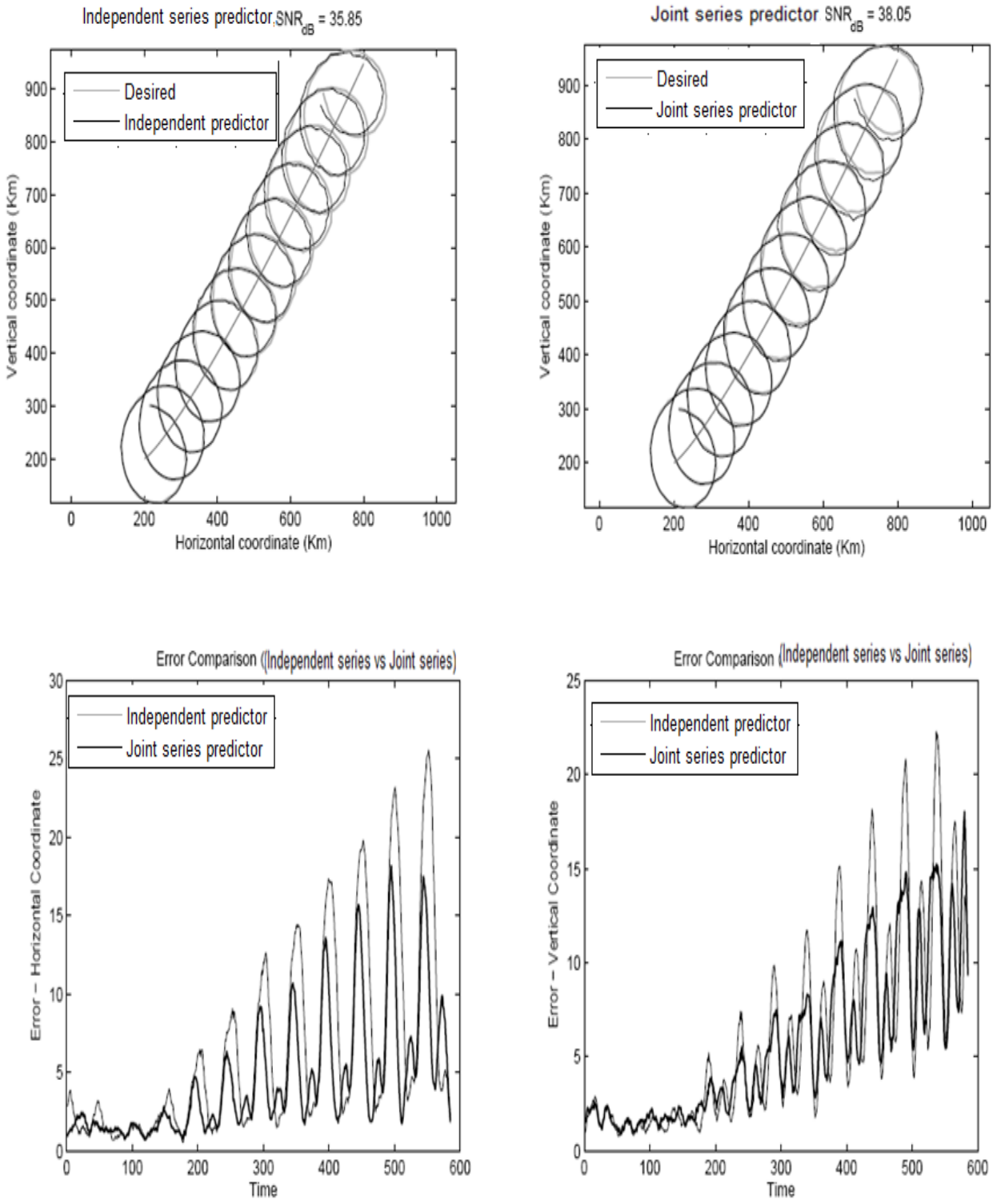


Figure 8. Simulation presenting the performance of joint series predictor and independent series predictor

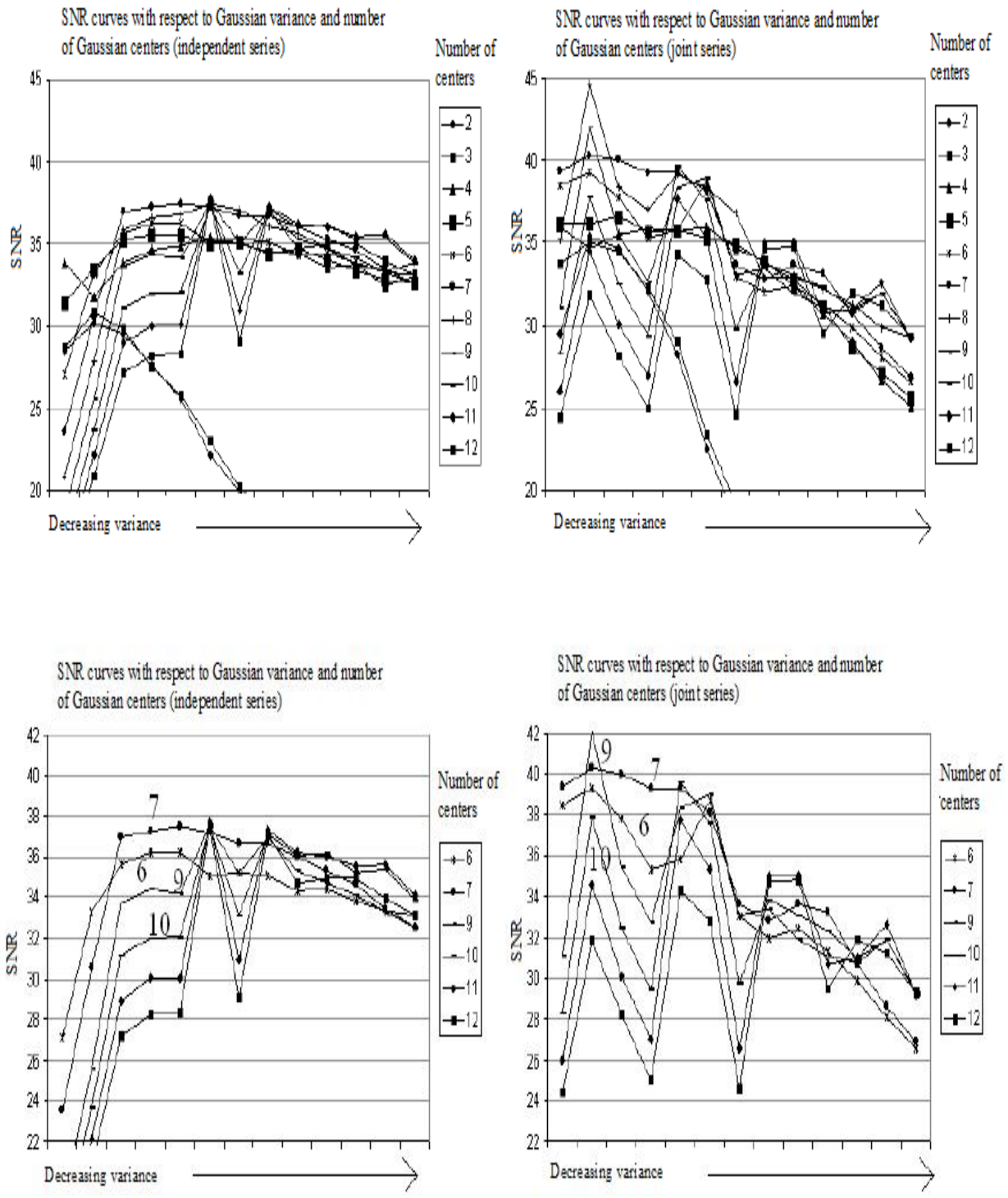


Figure 9. SNR results with respect to the variance and with respect to the number of centers for joint series predictor and independent series predictor

The joint series predictor and independent series predictor networks were trained using 200 samples of the generated circular motion series, while 600 samples of the circular motion series were used to test both of these networks. The testing samples of the series that were used are the further evolvements of the circular motion series, corrupted by a small amount of additive random noise. Thus, only a short, noiseless version of the testing series was seen by the network during the training phase.

One of the reasons of this simulation was to study the concept of parameter dependency. The second reason was to analyze the performance of two networks with a limited training data set, so that training and testing can be performed online. The simulations are presented in Figure 9. The scaling parameter α value is 0.5 x 10^{-4} and number of centers employed is 8 for both predictor networks. The need of using small value for α has already been discussed in section 3.2. From the simulations, it can be observed that joint series predictor network is more effective in predicting the particular motion pattern presented.

4.2 *Dependence on the number of basis function and on scaling parameter α*

As it is not easy to draw conclusions based on only a single simulation, several simulations were conducted which are presented in Figure 8. These simulations give a better idea of the performance of joint series predictor network and independent series predictor network in correspondence to the number of basis functions and different values of variance (which as a reminder is equal to $\sigma^2 = 1/2\alpha$).

The results are presented in Figure 9. The top two figures present all results, while the bottom two figures present selected results in order to be able to provide a closer look to the SNR performance curves.

Form the simulations in Figure 9, it can be easily understood that the joint series predictor network produces higher SNR values compared to the independent series predictor network. Moreover, it can be seen that as the number of basis functions (centers) increases, the SNR performance of the two predictor networks is not respectively increasing. The reason for this can be the overlapping of basis functions, because with such a small variance only few centers are sufficient to approximate the series. Therefore, when a large number of centers are used, there is an overlap which in turn causes the network to lose its generalization capabilities.

The joint series predictor network and independent series predictor network may have their own range of variance values and number of basis function for which they may be performing well. In the simulation as the value of the parameter α used is in the order of 10^{-4}. It can be argued that if a small numbers of basic functions are employed in the network, it is preferable to choose a small α for better SNR performance.

One of the main goals was to find a computationally efficient predictor network that will support a computationally efficient weather image modeling, so that the networks are tested with a small set of training data. However, if the training data set is large, then the value of the variance and number of basis function required may be different than the one used in these experiments.

4.3 *Experimentation with Real weather-image Data*

It was clearly understood from the simulations that the joint series predictor provides higher SNR than the independent series predictor. In order to ensure that the concept dependency among Gaussian envelope parameters holds, experimentation was performed with real weather-image data.

Figure 3 shows actual weather-image data in consecutive frames. Figure 4 illustrates these weather images approximations as a mixture of Gaussian envelopes using the technique in [6]. The weather-image data was approximated with a total number of 32 Gaussian envelopes in each frame.

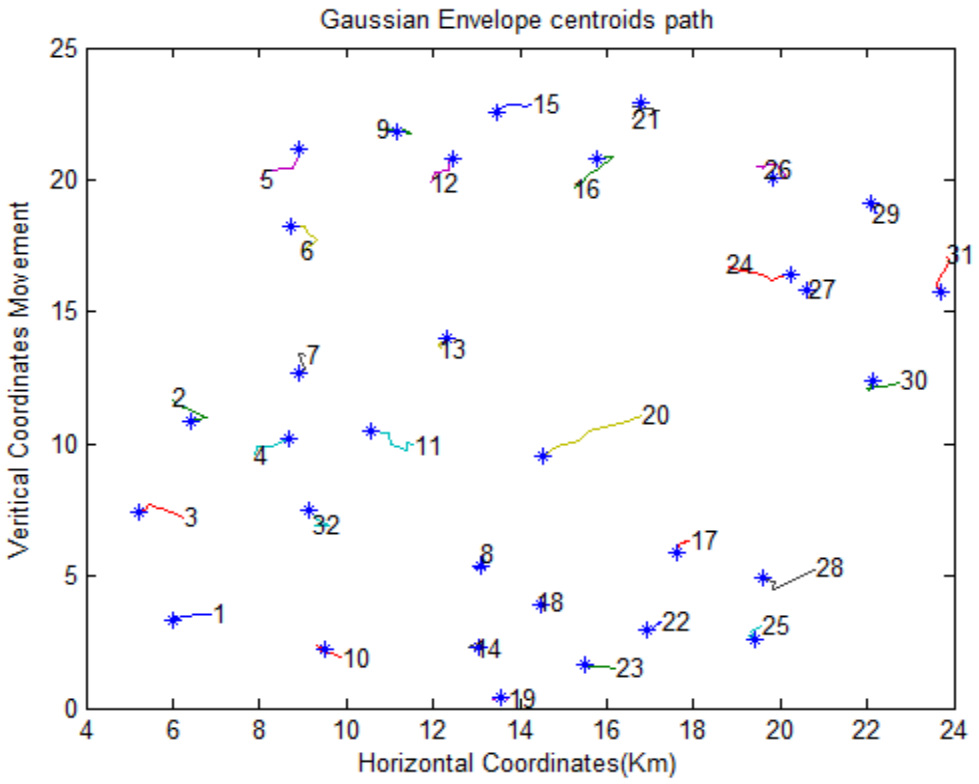


Figure 10. *The Gaussian envelope centers path for four consecutive weather images*

After the approximation is completed, the center coordinates (horizontal coordinate, vertical coordinate) of the Gaussian envelopes in each image are considered for four consecutive frames. Considering these two coordinates values in successive frames as a time series, they are presented to the joint series predictor network and the individual series predictor network. The Figure 10 shows the motion pattern of Gaussians envelope centers for four consecutive images. The response of both predictors is shown in Figure 11.

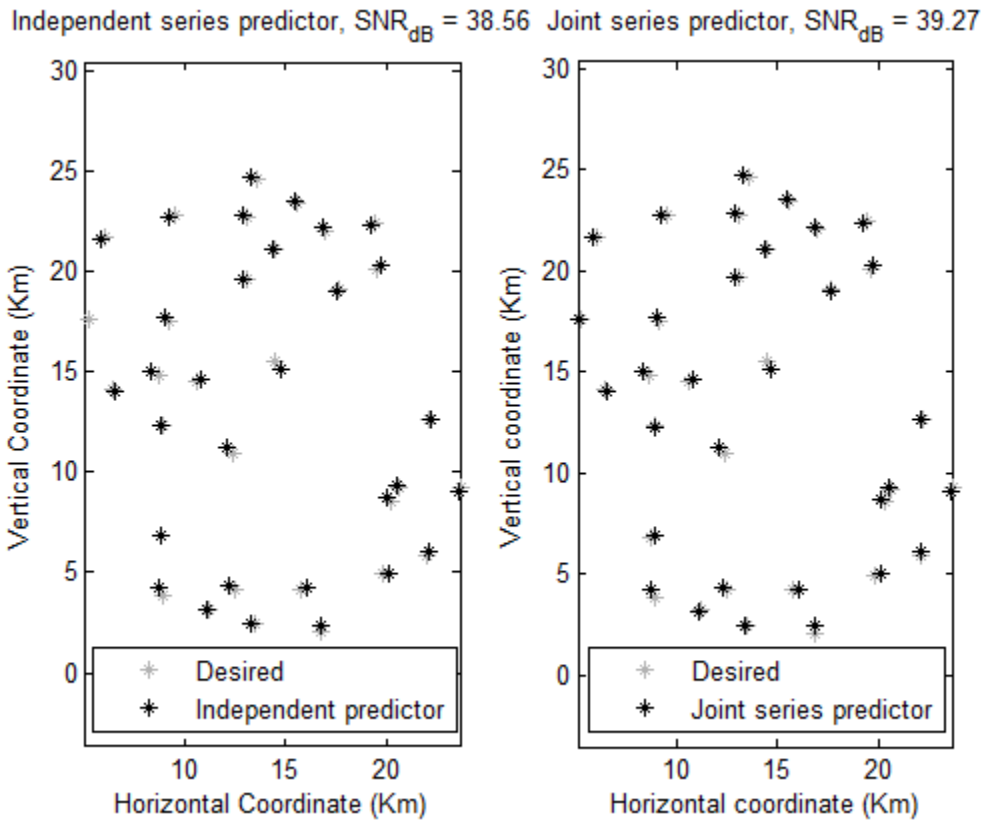


Figure 11. *Response of independent series predictor and joint series predictor*

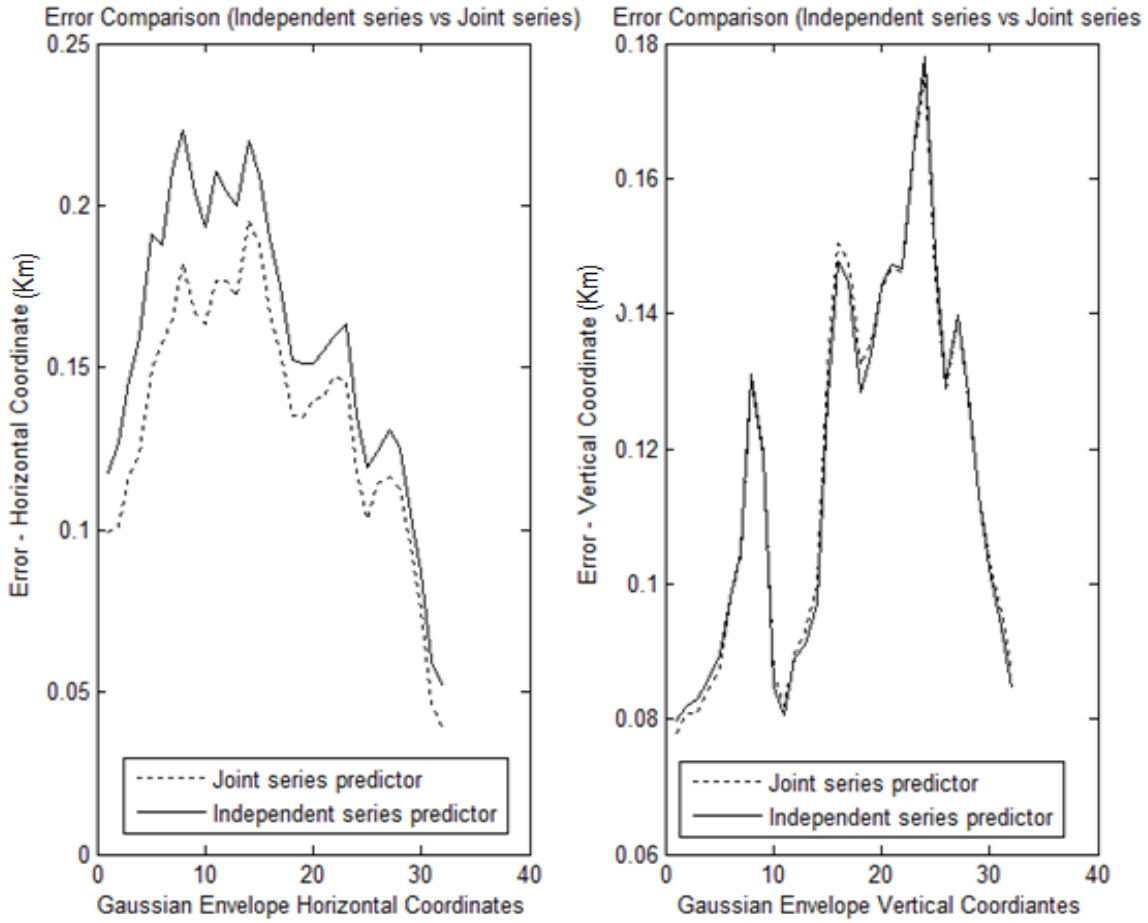


Figure 12.*Error comparison of independent predictor and joint series predictor*

It is clear from Figure 12 that in overall joint series network prediction is better than the independent series network prediction, in terms of the prediction error. The error in Figure 12 is defined as the absolute difference between the actual coordinate value, and the predicted coordinate value.

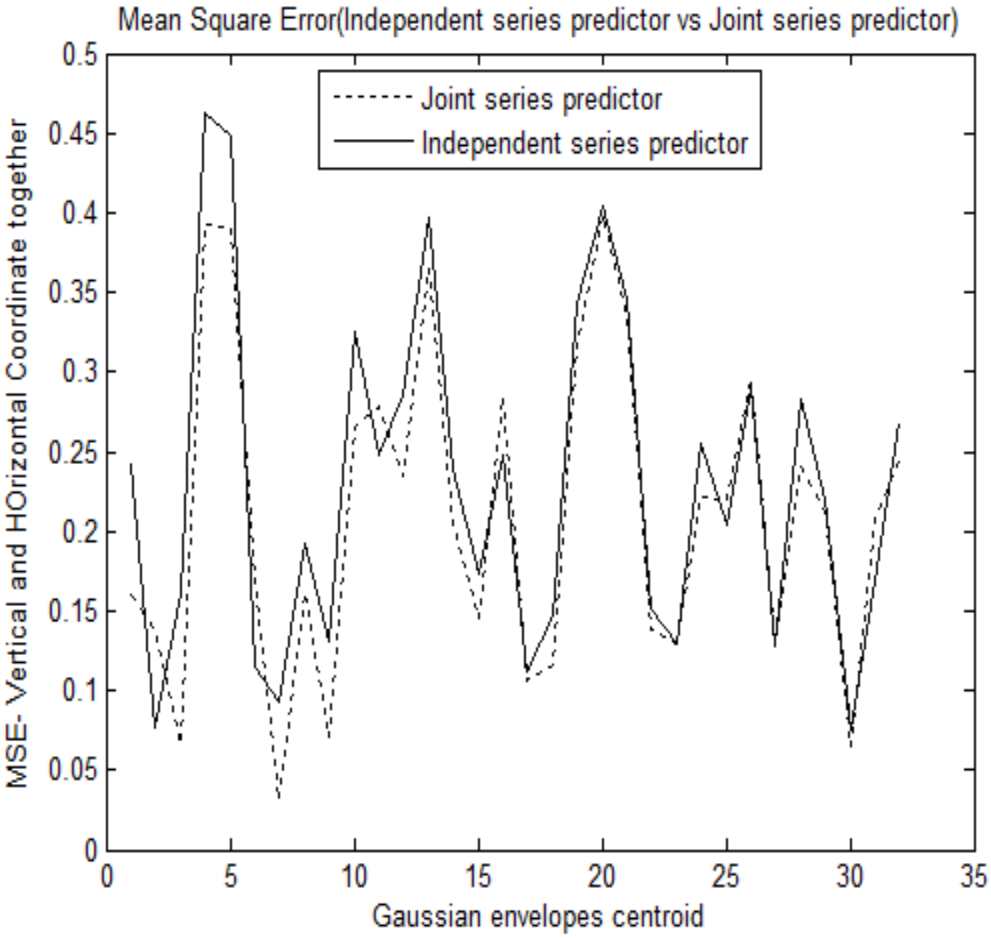


Figure 13. MSE of independent series predictor and joint series predictor

Figure 13 shows the Mean Square Error (MSE) of independent series predictor and joint series predictor. From this figure, it is clear that the joint series prediction has a smaller MSE value compared to the independent series prediction. The experiment is conducted with a total of 6 different data sets. The obtained values of MSE and SNR values of two predictors are given in Table 1 and Table 2, respectively. Here, it should be reminded that the actual weather image was down sampled by 16, thus the image size of 25×25 was approximated with Gaussian envelopes. In other words, obtained MSE and SNR values of the two predictors are 16 times the values presented in Table 1 and Table 2, respectively.

Data Set No.	Independent series predictor	Joint series predictor
Data set 1	$0.0321(km)$	$0.0272(km)$
Data set 2	$0.0272(km)$	$0.0367(km)$
Data set 3	$0.1195(km)$	$0.1093(km)$
Data set 4	$0.0318(km)$	$0.0290(km)$
Data set 5	$0.0393(km)$	$0.0374(km)$
Data set 6	$0.0278(km)$	$0.0214(km)$

Table 1.*Comparison in terms of MSE*

Data Set No.	Independent series predictor	Joint series predictor
Data set 1	$38.56(db)$	$39.27(db)$
Data set 2	$37.32(db)$	$38.17(db)$
Data set 3	$32.79(db)$	$33.18(db)$
Data set 4	$39.71(db)$	$40.11(db)$
Data set 5	$38.96(db)$	$38.75(db)$
Data set 6	$39.62(db)$	$40.76(db)$

Table 2. *Comparison in terms of SNR*

From the above tables it can be argued that the joint series predictor performance is better compared to independent series predictor, in terms of MSE and SNR.

Chapter 5: Conclusion and Future work

In this work, a modification to the existing technique was proposed and implemented. Assuming that weather radar data can be represented as a mixture of Gaussian envelopes, the main goal of the thesis is to investigate the parameter dependency of Gaussian envelope parameters.

With the introduction of joint series prediction, the computational complexity and algorithm runtime are reduced. By using the proposed concept of parameter dependency it was shown that parameters can be predicted simultaneously. At the same time, a better SNR performance is achieved compared to predicting parameters independently, at least based on the experimental results performed in this work.

Future work includes studying the dependency of all Gaussian envelope parameters including envelope heights and covariance matrices. For this study, interesting motion patterns which change rapidly such as hurricane events, cyclones, storms, and other significantly moving tropical systems will be collected. Although the forecasting of these events is a critical issue, it was proved that once the events are modeled, the forecasting can be done quickly by tracking the modeled parameters.

References

[1] Einfalt, T., Denoeux, T., Jacquet, G., "A radar rainfall forecasting method designed for hydrological purposes," *J Hydrology* , vol.114, pp.229-244, 1990.

[2] Austin, GL.,Belllon, A., "The used digital weather radar records for short term precipitations forecasting," *Quartz J Roy Meteorological Soc,* vol.100, pp.658-664, 1974.

[3] Frech, MN., Krajewski, WF., Cuykendall, RR., "Rainfall forecasting in space and time using a neural network," *J Hydrology,* vol.137, pp.1-31, 1992.

[4] Broomhead, D.S., "Multivariable functional interpolation and Adaptive Networks," *complex systems*, vol.2, pp.321-355, 1988.

[5] Denoeux, T., and Rizand, P., "Analysis of radar images for rainfall forecasting using neural networks," *Neural Computing and Applications*, vol.3, pp.50-61, 1995.

[6] Charalampidis, D., Paduru, .A, "Tracking of storm fronts in weather radar imagery ", *spie Defense security and sensing*, 2009.

[7] Borrell, V.E., Dartus, D., Alquier, M., "Forecasting falsh floods in unengaged basins with satellite data," *IAHS Publ* , vol.309, pp.20-22, 2007.

[8] Paul, J.T., "A Real Time weather system for forestry." *Bul. Ame.Metro. Soc.,* vol.62, pp.1466-1466, 1981.

[9] Hudlow, M.D.," Technological developments in real-time operational hydrologic forecasting in the United States," *Journal of Hydrology ,* Vol.102, pp.69-92, 1988.

[10] Jorgeson, J., Julien, P., "Peak Flow Forecasting with Radar Precipitation and the Distributed Model CASC2D," *Water International*, Vol.30, pp.40 – 49, 2005.

[11] http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/fcst/mth/prst.rxml

[12] Jack W.," The Weather Book", *Vintage Books*, ISBN 0-679-77665-6, 1997.

[13] Roberts, R. D., Rutledge, S.," Nowcasting storm initiation and growth using GOES-8 and WSR-88D data," *Wea.Forecasting*, vol.18, pp.562-584, 2003.

[14] Hamill, T.M.," Interpretation of rank histograms for verifying ensemble forecasts," *Mon. Wea. Rev*, vol.129, pp.550-560, 2001.

[15] Reggiani, P., Weerts, A. H.," Probabilistic Quantitative Precipitation Forecast for Flood Prediction," *Journal of Hydrometeorology*, pp.76–95, 2008.

[16] http://www.roc.noaa.gov/WSR88D/Level_II/Level2Info.aspx

[17] Chang, E. S. , Chen, S., and Mulgrew, B. , "Gradient radial basis function networks for nonlinear and nonstationary time series prediction," *IEEE Trans. Neural Networks*, vol.7, pp.190-194, 1996.

[18] Lee, S., "Supervised learning with Gaussian potentials, "*Neural Networks for Signal Processing*, pp. 189-227, 1992.

[19] Potts, M. A. S., Broomhead, D. S., "Time series prediction with a radial basis function neural network," *SPIE Adaptive Signal Processing*, vol. 1565, pp. 255-266, 1991.

[20] Casdagli., M., "Nonlinear prediction of chaotic time-series," *Physica D*, vol. 35, pp.335-356, 1989.

.[21] Chen S., Cowan, C. F. N., Grant, P. M., "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol 2, No 2, 1991.

[22] Charalampidis, D., Sravanthi, K., "Computationally efficient radar image based forecasting using RBF neural networks", *SPIE defence security and sensing*, vol.7701, 2010.

[23] James, S.H,"Business cycle fluctuations in us macroeconomic time series," *Hand book of Macroeconomics,* vol 1, pp 3-64, 1999.

[24] Chan, K.C., Karolyi, G.A., Longstaff, F.A., "An empirical comparison of alternative models of the short-term interest rate," *Journal of science*, vol 3, 1992.

[25] Makikallio, T.H., Seppanen, T., "Dynamic analysis of heart rate may predict subsequent ventricular tachycardia after myocardial infarction," *Scand J Soc,* 1989.

Appendix
Code

```
clc;
clear all;
close all;

var=1/1000; %%% variance of gaussian for original and modified
mc=3;    %%%modified OLS centers
oc=6;    %%%original OLS centers

ts=0.5;  %%%threshold %Note:value must be and 0< ts <1

nd=2;    %No.of parameters (x,y, kx, ky, wx, wy)
n1=4;   %%No.of past samples considered(length of each input vector)
%%%%% Parameters for testing the n/w%%%
nd2=2;    %no.of parameters (x,y, kx, ky, wx, wy) in each gaussian,should be
equal to (nd)
n2=4;    %%length of each input vector;always should be equal to (n1)
hours1=1;hours2=1;params=2;
paramst=2;

[mxi,mde1,mD1,mI,mC,mDel,mE1,mth,mA1,mG1,mw]=modified_training(hours1,hours2,
params,nd,n1,ts,mc,var);

%%%%%%%%%%%%%%%%%tetsing Modified_Ols%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% followed (testing) is the main function which will carry out the whole-
%-testing part of Modified_OLS
%mde2=desired output of testing series
% mD2=predicted output of tested series using modified_ols
%mE2=Differece b/w the actual series and predicted series values
%de2=desired output of testing series
% mC=dlmread('mC.txt');mDel=dlmread('mDel.txt');mth=dlmread('mth.txt');

hours3=18;
hours4=18;
[input2,mxi2,mde2,mD2,mE2]=modified_testing(hours3,hours4,paramst,nd2,n2,var)
;
% r);
input1=input2(:,:,1);
input3=input2(:,:,2);
%%%%%%%%%%%%%%%%%%%%Trainign OLS %%%%%%%%%%%%%%%%%%%%%%
% followed (ols_tr) is the main function which will carry out the whole-
%-training part of OLS
%send all input parameters defined so far
%this will return the training pattern and responce of n/w for training
%de1=desired output training series
%D1=predicted output of trained series using ols
%I Index of the selected centers using OLS
%C Centers selected using OLS
%Del prediction terms selected using OLS
%E1 error
%th weights computed using OLS
```

```
[xi1,de1,D1,I,C,Del,E1,th,A1,G1,w]=ols_tr(hours1,hours2,nd,n1,ts,oc,var) ;

[xi2,E2,D2,de2]=ols_te(th,nd2,n2,hours3,hours4,var) ;

figure
plot(D1{:,1},'-g');
hold on
plot(de1{:,1},'-r');
legend('Predicted(x,y)','Desired(x,y)');
title('Training pattern OLS');
figure
plot(D1{:,2},'-g');
hold on
plot(de1{:,2},'-r');
legend('Predicted(x,y)','Desired(x,y)');
title('Training pattern OLS');

figure
plot(mD1(:,1),'-b');
hold on
% figure
plot(mde1(:,1),'-r');
legend('Predicted(x)','Desired(x)');
title('Training pattern of centers (M_OLS)');

%plot the testing series coordinates(x,y)
figure
plot(mD1(:,2),'-b');
hold on
% figure
plot(mde1(:,2),'-r');
legend('Predicted(y)','Desired(y)');
title('Training pattern of  (M_OLS)');

figure
% subplot(3,1,1)
plot(D2{:,1},'-g');
hold on
plot(de2{:,1},'-r');
legend('Predicted(x,y)','Desired(x,y)');
title('Testing pattern OLS');

figure
% subplot(3,1,2)
plot(D2{:,2},'-g');
hold on
plot(de2{:,2},'-r');
legend('Predicted(x,y)','Desired(x,y)');
title('Testing pattern OLS');

figure
% subplot(3,1,1)
plot(mD2(:,1),'-b');
hold on
plot(mde2(:,1),'-r');
```

```
legend('Predicted(x)','Desired(x)');
title('Testing pattern of centers (M_OLS)');
%
figure
% subplot(3,1,2)
plot(mD2(:,2),'-b');
hold on
plot(mde2(:,2),'-r');
legend('Predicted(y)','Desired(y)');
title('Testing pattern of centers (M_OLS)');
%

% ols_me=[(mse(de1{1}-D1{1})+mse(de1{2}-D1{2})+mse(de1{3}-D1{3})+mse(de1{4}-
D1{4})+mse(de1{5}-D1{5}))/2 (mse(de2{1}-D2{1})+mse(de2{2}-D2{2})+mse(de2{3}-
D2{3})+mse(de2{4}-D2{4})+mse(de2{5}-D2{5}))/2]
mols_me=[mse(mde1-mD1) mse(mde2-mD2)]
ols_me=[(mse(de1{1}-D1{1})+mse(de1{2}-D1{2}))/2 (mse(de2{1}-
D2{1})+mse(de2{2}-D2{2}))/2]
mols_me1=sqrt((mde2(:,1)-mD2(:,1)).^2+(mde2(:,2)-mD2(:,2)).^2);
ols_me1=sqrt((de2{1}-D2{1}).^2+(de2{2}-D2{2}).^2);
% pers_me=mse(test(:,1:end-1,:)-test(:,2:end,:))
error_ov1=(mde2(:,1)-mD2(:,1))-(de2{1}-D2{1});
error_ov2=(mde2(:,2)-mD2(:,2))-(de2{2}-D2{2});
dB_SIGNAL=10*log10(mse(mde2));
dB_ERROR_OLS=10*log10(ols_me(2));
dB_ERROR_mOLS=10*log10(mols_me(2));
% dB_ERROR_pers=10*log10(pers_me);
dB_SNR_OLS=round((dB_SIGNAL-dB_ERROR_OLS)*100)/100
dB_SNR_mOLS=round((dB_SIGNAL-dB_ERROR_mOLS)*100)/100
% dB_SNR_pers=round((dB_SIGNAL-dB_ERROR_pers)*100)/100

figure
subplot(1,2,1),
plot(de2{1},de2{2},'*k','Color',[0.7 0.7 0.7]);hold
on;plot(D2{1},D2{2},'*k');hold off;
% if(strcmp(model_used,'Circular')==1)
%     hold on, plot(xcc,ycc,'Color',[.4 0.4 0.4]);hold off
% end
legend('Desired','Independent prdictor','Location','best');
xlabel('Horizontal Coordinate (Km)')
ylabel('Vertical Coordinate (Km)')
title(['Testing Series (OLS), SNR_{dB} = ' num2str(dB_SNR_OLS)]);
axis equal

subplot(1,2,2),
plot(mde2(:,1),mde2(:,2),'*k','Color',[0.7 0.7 0.7]); hold
on;plot(mD2(:,1),mD2(:,2),'*k');hold off
% if(strcmp(model_used,'Circular')==1)
%     hold on,plot(xcc,ycc,'Color',[.4 0.4 0.4]), hold off;
% end
legend('Desired','Joint series predictor','Location','best');
xlabel('Horizontal coordinate (Km)')
ylabel('Vertical coordinate (Km)')
title(['Testing Series (Joint OLS), SNR_{dB} = ' num2str(dB_SNR_mOLS)]);
axis equal
```

```
figure
subplot(1,2,1),
plot(1:length(mD2(:,1)),filter2(ones(10,1)/10,abs(mD2(:,1)-
mde2(:,1))),':k',1:length(mD2(:,1)),filter2(ones(10,1)/10,abs(D2{1}-
mde2(:,1))),'-k');
xlabel('Gaussian Envelope Horizontal Coordinates')
ylabel('Error - Horizontal Coordinate')
title('Error Comparison (Independent series vs Joint series)');
legend('Joint series predictor','Independent series
predictor','Location','best');

subplot(1,2,2),
plot(1:length(mD2(:,1)),filter2(ones(10,1)/10,abs(mD2(:,2)-
mde2(:,2))),':k',1:length(mD2(:,1)),filter2(ones(10,1)/10,abs(D2{2}-
mde2(:,2))),'-k');
xlabel('Gaussian Envelope Vertical Coordiantes')
ylabel('Error - Vertical Coordinate')
title('Error Comparison (Independent series vs Joint series)');
legend('Joint series predictor','Independent series
predictor','Location','best');

set(gcf,'Position',[101 101 800 800])
%
figure
plot(error_ov1,':k');hold on;plot(error_ov2,'-k');hold off;
xlabel('Gaussian envelopes centroids')
ylabel('Error - Verticaland horizontal Coordinate')
title('Error Comparison (Independent series vs Joint series)');
legend('Error-vertical','Error-horizontal','Location','best');

figure
plot(mols_me1,':k');hold on;plot(ols_me1,'-k');
xlabel('Gaussian envelopes centroid')
ylabel('MSE- Vertical and HOrizontal Coordinate together')
title('Mean Sqare Error(Independent series predictor vs Joint series
predictor)');
legend('Joint series predictor','Independent series
predictor','Location','best');

figure
plot(input1(:,:),25-input3(:,:));
hold on
plot(input1(1,:),25-input3(1,:),'*');
for i=1:length(input1(1,:))
    text(input1(end,i),25-input3(end,i),num2str(i));
end
xlabel('Horizontal Coordinates(Km)')
ylabel('Veritical Coordinates Movement')
title('Gaussian Envelope centroids path');

% plot(1:length(mD2(:,3)),filter2(ones(10,1)/10,abs(mD2(:,3)-mde2(:,3))),'-
k',1:length(mD2(:,3)),filter2(ones(10,1)/10,abs(D2{3}-mde2(:,3))),':k');
% xlabel('Time')
% ylabel('Error - Horizontal Coordinate')
% title('Error Comparison (OLS vs Joint OLS)');
% legend('Predicted Joint OLS','Predicted OLS','Location','best');
```

```
%
% subplot(2,2,3),
% plot(1:length(mD2(:,4)),filter2(ones(10,1)/10,abs(mD2(:,4)-mde2(:,4))),'-
k',1:length(mD2(:,4)),filter2(ones(10,1)/10,abs(D2{4}-mde2(:,4))),':k');
% xlabel('Time')
% ylabel('Error - Horizontal Coordinate')
% title('Error Comparison (OLS vs Joint OLS)');
% legend('Predicted Joint OLS','Predicted OLS','Location','best');
%
% subplot(2,2,4),
% plot(1:length(mD2(:,5)),filter2(ones(10,1)/10,abs(mD2(:,5)-mde2(:,5))),'-
k',1:length(mD2(:,5)),filter2(ones(10,1)/10,abs(D2{5}-mde2(:,5))),':k');
% xlabel('Time')
% ylabel('Error - Vertical Coordinate')
% title('Error Comparison (OLS vs Joint OLS)');
% legend('Predicted Joint OLS','Predicted OLS','Location','best');
%
% set(gcf,'Position',[101 101 800 800])
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```
function [input2,xi de d2 del]=real_input(hours1,hours2,params,s)
% clc;clear all;close all;
% hours1=1;hours2=1;s=5;
% params=2;
sam=s+1;
location{1}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\cx\';
location{2}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\cy\';
location{3}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k1\';
location{4}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k2\';
location{5}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k3\';

% loc=strcat(location{params},num2str(hours1),'.txt');
%         temp=load(loc);

for j=1:params
        irw=1;
        for i=hours1:hours2
        loc=strcat(location{j},num2str(i),'.txt');
        temp=load(loc);
        [rwt cwt]=size(temp);
        for rw=1:rwt
            input(:,irw,j)=temp(rw,:);
            irw=irw+1;
        end
    end;

    end
%     input2=input(1:end,:,:);
%     xi=input2(2:end-sam,:,:)-input2(3:end-sam-1,:,:);
```

```
%     de=input2(1,:);
%     d2=input2(:,2);
%     del=de-d2;

[rw col di]=size(input);
% input1=input(end:-1:1,:,:);
temp=1;
ir3=1;sf3=sam;ir4=1;sf4=sam;
input2=input(end:-1:1,:,:);
for k=1:di
    ic=1;

    sf2=sam;
    input1=input(end:-1:1,:,k);
    for j=1:col
        ir=1;
        st=1;
        sf1=sam;
        for i=1
            temp(ir:sf2,ic)=input1(st:sf1,j);
            st=st+1;
            sf1=sf1+1;
            ic=ic+1;
        end


    end

    de(:,k)=temp(1,:);
    d2(:,k)=temp(2,:);
    input3(ir3:sf3,:)=temp;
xi(ir4:sf4-2,:)=temp(2:end-1,:)-temp(3:end,:);
    ir3=ir3+sam;
    sf3=sf3+sam;
    ir4=ir4+sam-2;
    sf4=sf4+sam-2;
end
del=de-d2;

% %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p]=modified_gaus(xg,cg,d2g,delg,no,var,hours,params)
%xg sucessive difference samples of input
%d2g past samples of time series
%delg Initial Predication Terms
%cg Initial centers
%p=guassian terms
%no is no.of parameters of guassian
% clc;clear all; close all;
% hours=1;
% params=5;
% no=5;
% s=5;
% var=1/2000;
```

```
% [xg,deg,d2g,delg]=real_input(hours,params,s);
% % for prs=1:params
%     cg=xg;
%To calculate the Guassian terms

[n2 N]=size(xg);
[m2 M]=size(cg);
%To calculate the Guassian terms
for k=1:no
for i=1:N  %to repeat the hidden node loop for every input(N number of
inputs)
     temp=xg(:,i);
  for j=1:M  % to repeat M no.of centers or hidden nodes
       nor=norm(temp-cg(:,j));
       gu=exp(-var*nor^2);
       p(i,j,k)=gu*(d2g(i,k)+delg(j,k));
 end;
 end;
 %%%%%%%%%%%end of calculating guassian term%%%%%%%%%%
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [I,th,T,A1,G1,w]=modified_subset(d,ts,mc)
%%%%%feb03su is function determine the actual centers and prediction terms
%%I variable that contains the Index of the selected centers using OLS
algorithm
% th weight vector and each value is the weight associated with specific
% center
%%T is the thresold error value at which the subset selection process is
%%terminated
%p guassian terms
%d is the normalized desired output desried
%ts is tolerable threshold value
[n,M,nd]=size(p(:,:,:));
% %%%%%%%first step of the subset selection algorithm %%%%%%%%
for i=1:M
    for k=1:nd
        w1(:,i,k)=p(:,i,k);
        g1(:,i,k)=(w1(:,i,k)'*d(:,k))/(w1(:,i,k)'*w1(:,i,k));
        e1(:,i,k)=g1(:,i,k)^2*w1(:,i,k)'*w1(:,i,k)/(d(:,k)'*d(:,k));
    end
    e1f(:,i)=sum(e1(:,i,1:nd));
end
[e(:,1) I(:,1)]=max(e1f(:,:),[],2);
w(:,1,1:nd)=p(:,I,1:nd);
for i=1:nd
    G1(1,:,i)=g1(:,I,i);
end
%%%%%%%%%end of sub set celction algoritm first step%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%ts defines the tolerable error value that we can choose
% ts=0.5;
T=ts;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%second step of algorithm for subset selction%%%%%%
for ms=2:M
   if T>=ts
%     if ms<=mc
        for k=2:ms
            for i=1:M %%no.of centers
                for N=1:nd %%nd.no.of parameters predicted at a time
                    if(i~=I(:,:))

                        s=0;
                        for j=1:k-1

a(j,k,N)=w(:,j,N)'*p(:,i,N)/(w(:,j,N)'*w(:,j,N)+0.0001);
                            s=s+(a(j,k,N)*w(:,j,N));

                        end
                        wk(:,i,N)=p(:,i,N)-s;

gk(:,i,N)=wk(:,i,N)'*d(:,N)/(wk(:,i,N)'*wk(:,i,N)+0.0001);

ek(:,i,N)=gk(:,i,N)^2*wk(:,i,N)'*wk(:,i,N)/(d(:,N)'*d(:,N)+0.0001);

                    else ek(:,i,N)=0;
                    end
                end

                ekf(:,i)=mean(ek(:,i,1:nd));
            end
            [e(:,k) I(:,k)]=max(ekf(:,:),[],2);
            w(:,k,1:nd)=wk(:,I(:,k),1:nd);
            G1(k,:,1:nd)=gk(:,I(:,k),1:nd);

            for N=1:nd
                for j=1:k-1

A1(j,k,N)=w(:,j,N)'*p(:,I(:,k),N)/(w(:,j,N)'*w(:,j,N)+0.0001);
                    A1(j,j,N)=1;
                end
            end
            %   w(:,k,2)=wk(:,I(:,k),2);
            %  G1(k,:,2)=gk(:,I(:,k),2);
        end

        A1(ms,ms,1:nd)=1;
        % A1(ms,ms,2)=1;

        %%%%finding total error to terminate the sub set selection
process%%%%%%%

        t=0;
        for i=1:ms
            t=t+ e(:,i);
        end
```

```
        T=nd-t;
    end
end
for i=1:nd
    th(:,i)=inv(A1(:,:,i))*G1(:,:,i);
end
% th(:,2)=inv(A1(:,:,2))*G1(:,:,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
[input2,xi2,de2,mD2,E2]=modified_testing(hours3,hours4,params,nd,n2,var)
% C centers selected by OLS algorithm while training n/w
%Del prediction terms calculated by OLS algorithm while training n/w
% th , weights calculated by OLS algorithm while training n/w
%test is series to be tested

% %%%%% Parameters for testing the n/w%%%
% nd;%no.of parameters (x,y, kx, ky, wx, wy) in each guassian,should be equal
to (no)
% n2;%%length of each input vector;always should be equal to (n1)
%%%%%%%%%%Testing the N/W for a different series(testing series)%%%%
%%%%%%Generating sucessive sample differences for testing series%%%%%%%%

[input2,xi2,de2,dp2,del2]=real_input(hours3,hours4,params,n2);

%%%%%%%%%%%%%computing guassian terms of testing series%%%%%%%%
%p3=guassian terms

[p3]=modified_gaus(xi2,C,dp2,Del,nd,var);
%%%%%%%%%%calculating normalized predicted output of testing series %%
for i=1:nd
mD2(:,i)=p3(:,:,i)*th(:,i);
end
% E2 is the Error b/w the desired series de2 and Actual predicted series  ytt
E2=de2-mD2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p]=ols_guas(xg,cg,d2g,delg,var)
%xg sucessive difference samples of input
%d2g past samples of time series
%delg Initial Predication Terms
%cg Initial centers
%p=guassian terms

[n2 N]=size(xg);
[m2 M]=size(cg);
%To calculate the Guassian terms
for i=1:N  %to repeat the hidden node loop for every input(N number of
inputs)
     temp=xg(:,i);
  for j=1:M  % to repeat M no.of centers or hidden nodes
       no=norm(temp-cg(:,j));
       gu=exp(-var*no^2);
    p(i,j)=gu*(d2g(i,:)+delg(j,:));
```

```
    %figure(100+j),plot(i,gu,'*'),hold on
    %j
 end;
 end;
 %%%%%%%%%%%end of calculating guassian term%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xi,d2,del,de]=ols_input(hours1,hours2,jp,s)
%jp===specific file (1.txt,2.txt)
% function [xi,d2,del,d,de,Su,sfi,indices]=ols10i(j,nd,npi,sfi,input)
%xi sucessive difference samples of input
%d2 past samples of time series (going to be used in subset selection
%function)
%del Initial Predication Terms
%de actual desire output of time series
%nd no.of parameters we are prediciting at a time(no.on o/p arguments)
%npi no.of past samples to be considered(length of each input vector)
% clc;clear all;close all;
% hours1=1;hours2=1;
% jp=5;s=2;
sam=s+1;
location{1}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\cx\';
location{2}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\cy\';
location{3}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k1\';
location{4}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k2\';
location{5}='C:\Users\Hariprakash Reddy\Documents\SRAVANTHI\REFERENCE
PAPERS\data\sep16_2004_ivan\rita_parameters\k3\';

    for j=jp
        irw=1;
        for i=hours1:hours2
        loc=strcat(location{j},num2str(i),'.txt');
        temp=load(loc);
        rwt=size(temp);
        for rw=1:rwt
            input(:,irw)=temp(rw,:);
            irw=irw+1;
        end
    end;

    end

input1=input(end:-1:1,:);
[rw col]=size(input);
temp=1;
ir3=1;sf3=sam;ir4=1;sf4=sam;

for k=1
    ic=1;
```

```
    sf2=sam;

    for j=1:col
        ir=1;
        st=1;
        sf1=sam;
        for i=1
            temp(ir:sf2,ic)=input1(st:sf1,j);
            st=st+1;
            sf1=sf1+1;
            ic=ic+1;
        end

    end
    de(:,k)=temp(1,:);
    d2(:,k)=temp(2,:);
    input3(ir3:sf3,:)=temp;
xi(ir4:sf4-2,:)=temp(2:end-1,:)-temp(3:end,:);

    ir3=ir3+sam;
    sf3=sf3+sam;
    ir4=ir4+sam-2;
    sf4=sf4+sam-2;
end
del=de-d2;

% %     input2=input1(1:end-1,:,:)-input1(2:end,:,:);
%     xi=input2(2:end-1,:,:)-input2(3:end,:,:);
%     de=input2(1,:);
%     d2=input2(:,2);
%     del=de-d2;


% %     input2=input1(1:end-1,:,:)-input1(2:end,:,:);
%     xi=input1(2:end-1,:)-input1(3:end,:);
%     de=input1(1,:)';
%     d2=input1(2,:)';
%     del=de-d2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [I,th,T,A1,G1,w]=ols_subset(p,d,ts,oc)

%%%%%nov23su is function determine the actual centers and prediction terms
%%I variable that contains the Index of the selected centers using OLS
algorithm
% th weight vector and each value is the weight associated with specific
% center
%%T is the thresold error value at which the subset selection process is
%%terminated
%p guassian terms
%d is the normalized desired output desried
%ts is tolerable threshold value
%%%%%%%first step of the subset selection algorithm %%%%%%%%
```

```
%M define the number of centers
% clc;
% clear all;
% close all;
% p=rand(
[n M]=size(p);
for i=1:M
    w1(:,i)=p(:,i);
    g1(:,i)=(w1(:,i)'*d)/(w1(:,i)'*w1(:,i));
    e1(:,i)=g1(:,i)^2*w1(:,i)'*w1(:,i)/(d'*d);
end
[e(:,1) I(:,1)]=max(e1(:,:),[],2);
w(:,1)=p(:,I);
G1(1,:)=g1(:,I);
%%%%%%%%%end of sub set selection algorithm first step%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%ts defines the tolerable error value that we can choose
% ts=0.5;
T=ts;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%second step of algorithm for subset selction%%%%%%
for ms=2:M
     if T>=ts
%       if ms<=oc
        for k=2:ms
            for i=1:M
                if(i~=I(:,:))
                    s=0;
                    for j=1:k-1
                        a(j,k)=w(:,j)'*p(:,i)/(w(:,j)'*w(:,j)+0.0001);
                        s=s+(a(j,k)*w(:,j));
                        A1(j,k)=a(j,k);
                        A1(j,j)=1;
                    end
                    wk(:,i)=p(:,i)-s;
                    gk(:,i)=wk(:,i)'*d/(wk(:,i)'*wk(:,i)+0.0001);
                    ek(:,i)=gk(:,i)^2*wk(:,i)'*wk(:,i)/(d'*d+0.0001);
                else ek(:,i)=0;
                end

            end

            [e(:,k) I(:,k)]=max(ek(:,:),[],2);
            w(:,k)=wk(:,I(:,k));
            G1(k,:)=gk(:,I(:,k));
            for j=1:k-1
                A1(j,k)=w(:,j)'*p(:,I(:,k))/(w(:,j)'*w(:,j)+0.0001);
                A1(j,j)=1;
            end
        end
        A1(ms,ms)=1;
```

```
        %%%%finding total error to terminate the sub set selection
process%%%%%%%

        t=0;
        for i=1:ms
            t=t+ e(:,i);
        end
        T=1-t;
    end
end
%%%%%%%%%%%%%%%%%%%%%%end of the subset selection algorithm second step%%
th=inv(A1)*G1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

function
[xi3,o_E2,o_DD2,o_de2]=ols_te(o_C,o_Del,o_th,nd,n2,hours3,hours4,var)
% clear all;
% close all
% clc;
for i=1:nd
  C=o_C{:,i};
  Del=o_Del{:,i};
  th=o_th{:,i};
%%%%%%%%%%%%%%%%%%%%%%%To generate time series%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [xi2,dp2,del2,d2,de2,su2]=ols_ip(i,n2,sf2,test);
[xi2,dp2,del2,de2]=ols_input(hours3,hours4,i,n2);
xi3(:,:,i)=xi2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
[p3]=ols_guas(xi2,C,dp2,Del,var);
D2=p3*th;
E2=de2-D2;
[m n]=size(D2);

o_E2(:,i)={E2};
o_DD2(:,i)={D2};
o_de2(:,i)={de2};
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
[xi1t,o_de1,o_DD1,o_I,o_C,o_Del,o_E1,o_th,A1,G1,w]=ols_tr(hours1,hours2,nd,n1
,ts,oc,var)
% o_de1 desired output of training series
% o_DD1 predicted output of training series
% o_I Index of selected centers
% o_C selected centers
% o_Del selected prediction terms using OLS
% o_E1 Erroe b/w actual desired o.p and predicted output
% o_th computed weights using OLS
% train is the training i/p series
% nd no.of parametrs of (x,y,kx,ky,wx,wy)
% n1 no.of past sample(length of each i/p vector)
% sf1 no.of total i/p vectors each of length n1
```

```
% ts tolerable threshold

% clc;
%
% clear all;
% close all;hours1=1;hours2=2;nd=5;n1=5;ts=0.5;oc=4;var=1/2000;
i=1;
% train=train1(:,:,:);
for j=1:nd

%%%%%%%%%%%%%%%%%%%%%%To generate input data of time
series%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[xi1,dp1,del1,de1]=ols_input(hours1,hours2,j,n1);
xi1t(:,:,j)=xi1;
c1=xi1;
%%%%%%%%%%nov4 is function that determine guassian terms%%%%%%%%%
%%p > guassian terms
[p]=ols_guas(xi1,c1,dp1,del1,var);

%%%%%nov4sub is function determine the actual centers and predication terms
%%I variable that contains the centers index,th is variables that
%%represents the weights and Del prediction terms
[I,th,T,A1,G1,w]=ols_subset(p,de1,ts,oc);
%%now assigning the actual ceners and prediction terms obatined from
algorithm to vairables
C=c1(:,I);%C--centers
Del=del1(I,:);%del--prediction terms
%%%%%%%% %Testing the trained network for th same series
P=p(:,I);
D1=P*th;
E1=de1-D1;
DD1=D1;
o_de1(:,j)={de1};
o_DD1(:,j)={DD1};
o_I(:,j)={I};
o_C(:,j)={C};
o_Del(:,j)={Del};
o_E1(:,j)={E1};
o_th(:,j)={th};
i=i+1;
end
```

Vita

Sravanthi Kattekola was born in Karimnagar, India. She received her undergraduate degree in Electronics and Communication Engineering from J.N.T University, India in May 2007. From Summer-2008 to Fall-2010 she was with Electrical Engineering department at UNO where she worked with Dr.Dimitrios Charalampidis as a Research assistant and pursued her Master's degree in Electrical Engineering. Her research interests include Digital Image Processing and Signal Processing.