Energy and Power 2012, 2(1): 1-9
DOI: 10.5923/j.ep.20120202.01

Different ANN Models for Short-Term Electricity Price Forecasting

S. V. N. L. Lalitha[1,*], **Maheswarapu Sydulu**[2], **M. Kiran Kumar**[2]

[1]K L University, Guntur, Andhra Pradesh, 521108, India
[2]National Institute of Technology, Warangal, 506004, India

Abstract Application of ANN to the electricity price forecasting is gaining importance in the context of electricity markets. This paper presents four different ANN models used for forecasting the electricity price. They are the classical back propagation neural network model, radial basis function neural network model, genetic algorithm based neural network and a direct non-iterative state estimation based neural network model. A case study is made with the downloaded data of the day-ahead pool market prices of the Iberian Energy Derivatives Exchange using the above four different ANN models and the results are compared.

Keywords ANN, genetic algorithm, state estimation, electricity markets, market clearing price

1. Introduction

In many countries all over the world, the power industry is moving towards a competitive framework, and a market environment is replacing the traditional centralized operation approach, a process that is known as restructuring. The most fundamental characteristic of the restructuring process that is taking place in numerous countries around the world is that market mechanisms have replaced the highly regulated procedures that were used in the decision-making process under traditional regulation[1]. The main objective of an electricity market is to decrease the cost of electricity through competition. The understanding of electric power supply as a public service is being replaced by the notion that a competitive market is a more appropriate mechanism to supply energy to consumers with high reliability and low cost[2]. Most commodities have been traded for many years. However, electrical energy is different from most other commodities, and electrical market has its own complexities. The electrical energy cannot be appreciably stored, and the power system stability requires constant balance between supply and demand. Most users of electricity are, on short timescales, unaware of or indifferent to its price. These two facts drive the extreme price volatility or even price spikes of the electricity market[3,4].

The deregulated power market is an auction market, and energy spot prices are volatile. On the other hand, due to the upheaval of deregulation in electricity markets, price forecasting has become a very valuable tool. The companies that trade in electricity markets make extensive use of price prediction techniques either to bid or to hedge against volatility. In the pool, usually, spot prices are publicly available, as is the case of the day-ahead pool of mainland Spain (www.omip.pt), the Californian pool (www.calpx.com), or the Australian national electricity market (www.nemmco.com.au). Many statistical based techniques are present in the literature for short term load forecasting and the same have been used for price forecasting also[5,6]. Among many existing tools, ANN has received much attention because of its clear model, easy implementation, and good performance. Many ANN based Price forecasting models were proposed in literature[7-12]. A novel non-iterative technique based on weighted least squares state estimation for short term load forecasting of Distribution Systems is proposed in[13]. Usage of Genetic algorithm for electricity price forecasting was proposed in[14,15].

In price forecasting applications, the main function of ANN is to predict price for the next hour, day(s) or week(s). The several reasons why ANN based price forecasting is superior over conventional forecasting techniques are that there is not any such standards and/or rules to describe the relationship between price variations and other parameters such as weather conditions. Also, the data used in the training and testing of the price-forecasting model is usually noisy and uncertain, and the price forecast performance is sensitive to initial conditions such as historical temperature information[16,17].

This paper presents four different ANN models used for Short Term Price Forecasting (STPF). They are the classical Back Propagation Neural Network model (BPNN), Radial Basis Function Neural Network model (RBFNN), Genetic Algorithm based Neural Network (GANN) and a direct non-iterative State Estimation based Neural Network model

* Corresponding author:
svnllalitha@gmail.com (S.V.N.L. Lalitha)
Published online at http://journal.sapub.org/ep

(SENN). In this paper, online price data from the Iberian Energy Derivatives Exchange is considered. Data of three months (April, May and June of 2011) is downloaded from the Iberian Energy Derivatives Exchange (www.omip.pt). A case study with the downloaded data of the day-ahead pool market prices of the Iberian Energy Derivatives Exchange is made with the above four different ANN models and the results are compared.

The paper is organized as follows. Section II briefly discusses the four different models i.e BPNN, RBFNN, GANN and SENN used for STPF. Flow-charts corresponding to the four different models are given in Section III. Section IV presents the results obtained from the case study using the four different models. Observations made from the test results are summarized as conclusions in Section V.

2. Different ANN Models

A. Back Propagation Neural Network Model

The BP network is composed of one input layer and one or more hidden layers and one output layer. The learning process of network includes two courses, one is the input information transmitting in forward direction and another is the error transmitting in backward direction. In the forward action, the input information goes to the hidden layers from input layer and goes to the output layer. If the output of output layer is different from the target value then the error will be calculated. These errors will be transmitted backward direction then the weights between the neurons of every layers will be modified in order to make the error as minimum as possible. Then the network is said to be trained network for the given data or application.

BPNN Architecture:

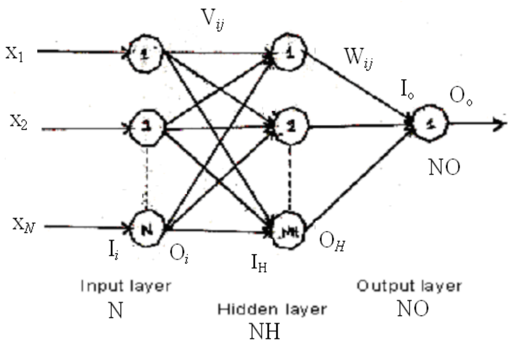

Figure 1. BPNN structure

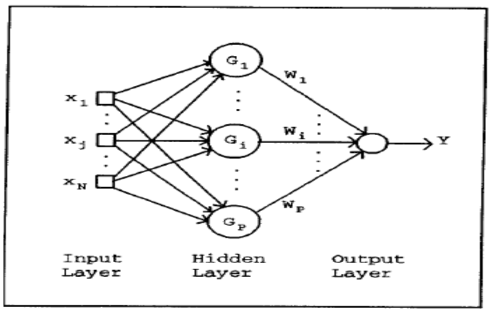

Figure 2. RBFNN structure

B. Radial Basis Function Neural Network Model

The Radial Function is similar to the Gaussian function, which is defined by a centre and a width parameter. The parameters of the RBF units are determined in three steps of the training activity. First, the unit centers are determined by some form of clustering algorithm like K-means clustering algorithm. Then the widths are determined by a nearest neighbor method. Finally, weights connecting the RBF units and the output units are calculated using delta rule. Euclidean distance-based clustering technique has been employed here to select the unit centers. The normalized input and output data are used for training of the RBF neural network. To ensure that the neural network has learned and not memorized, reshuffling of training patterns was performed to get almost equal errors during training and testing.

The location of the centers of the receptive fields is a critical issue. The RBF networks are benefited by clustering the training vectors when there is a large amount of training data. If the data forms a cluster, an entire cluster of training vectors in the training set may get connected to reduce number of hidden nodes. This may substantially decrease the computation time in the reference mode.

The diameter of the receptive region, determined by the value of unit width (σ) can have a profound effect upon the accuracy of the system. The objective is to cover the input space with receptive fields as uniformly as possible. If the spacing between centers is not uniform, it may be necessary for each hidden layer neuron to have its own value of σ .For hidden layer neuron whose centers are widely separated from others. σ must be large enough to cover the gap, whereas those in the centers of a cluster must have a small σ if the shape of the cluster is to be represented accurately.

The typical RBF network structure (Fig. 2) for STPF is a 3-layered network; with the nonlinear radial basis function as the transfer function. The RBFN model used here has 38 neurons in the input layer, 1 neuron in the output layer as utilized for BPA. The number of hidden neurons selected as 600 with Gaussian density function.

Euclidean distance-based clustering technique has been employed to select the number of hidden (RBF) units and unit centers. The optimal learning is achieved at the global minimum of testing error. It was observed that the convergence in this case was faster and also its performance was better as compared to the BPNN model.

C. Genetic Algorithm based Neural Network Model

Genetic algorithm has been used to search the weight space of a multilayer feed forward neural network without the use of any gradient information. The basic concept behind this technique is as follows. A complete set of weights (V_{11}, V_{12}, V_{13}…W_{11}, W_{21}, W_{31}….) are coded in a string, which has an associated fitness finding the optimal weights.

Implementation of Genetic Algorithm Neural Network is discussed below.

Chromosome Encoding: A major obstacle to using genetic algorithms to evolve the weights of a fixed network is the encoding of the weights onto the chromosome.

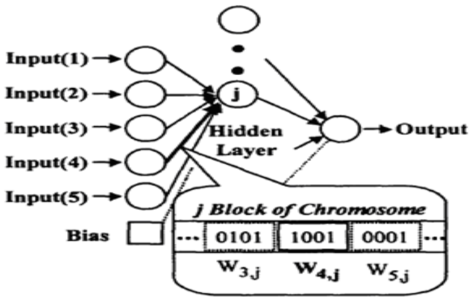

Figure 3. Coding method GANN

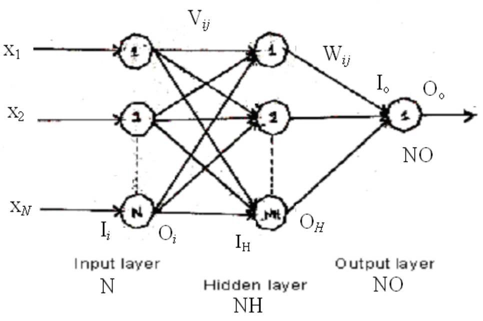

Figure 4. SENN structure

The weights of a neural network are generally real valued and unbounded, whereas a chromosome in a genetic algorithm is generally a string of bits of some arbitrary length (see fig 3). Encoding real values onto such a chromosome presents problems both in the precision of the representation and the resultant length of the chromosome. The length of the chromosome impacts upon the size of the search space of the genetic algorithm, and the efficiency of the search. A sample chromosome representation used in this work is

[V_{11}, V_{12}, ...V_{21},V_{22},...V_{31},V_{32},......W_{11}, W_{21}, W_{31}....]

Evaluation Function: Assign the weights on the chromosome to the links in a network of a given architecture, run the network over the training set of examples, and return the sum of the squares of the errors. Mean square error of each training pattern '*p*' is defined as E^p and is calculated as

$$E^p = \frac{\sqrt{\sum_{j=1}^{No} \left(T_j - O_{oj}\right)^2}}{No}$$

where T*j* is the target value and *Ooj* is the output value and '*No*' is the number of outputs

Error rate and the fitness of the ith chromosome is calculated as

$$Error\ rate(i) = \sqrt{\frac{\sum E^p}{Np}} \quad \text{and} \quad fit(i) = \frac{1}{Error\ rate(i)}$$

where Np is the total no. of training patterns

Genetic operators: Roulette-wheel technique is used for parent selection. Uniform cross over with probability of crossover 0.9 is used in this paper. Probability of *mutation* is taken in this work as 0.03. Probability of *Elitism* is taken in this work as 0.1. Gene Copy Operator (GCO) which selects the worst fitted chromosome and replaces that with best fitted chromosome in order to augment the optimization process is used. This process increases the search speed and allows the GA to get optimal solution.

The construction, scaling, and simulation of NN architecture are similar to that of what we used for BPNN implementation. The population size, bit length for each weight used in this work is 40, 10 respectively. The only difference is that there is no need to initialize the weight matrices; instead we have to generate initial random population.

D: State Estimation based Neural Network Model

The weight matrix between Input and Hidden layer is estimated like the states (x_1, x_2, and x_N) in State Estimation Technique. Similarly the weights between Hidden & Output also estimated like the states in State Estimation Technique. For each training sample, the input data at Input layer and the data at Output layer are known. But there is no relevant data at hidden nodes corresponding to each training sample. So the problem comes with hidden layer data. Different combinations at the hidden layer are tried and finally one set of hidden layer which is good in training the Neural Network Structure is selected.

SENN Architecture:

Construction of SENN (State Estimation based Neural Network) is similar to the BPNN, except the weight initialization[other things like Node structure, Training period, Normalization, De-normalization, etc remain unchanged]. In SENN method there is no need to initialize the weights to random values.

Further, it is notified that that the use of activation functions at the hidden layer nodes are not offering any favor to the training process.[i.e., the results with and without sigmodal activation function are almost the same]. Thus, the relation between all inputs[X], connected weights[V] to the first hidden node output 'y_1' for all training samples 'Ns' can be expressed as

$$\left[X\right]_{(Ns\times N)} \times \left[V\right]_{(N\times 1)} = \left[Y\right]_{(Ns\times 1)} \quad (1)$$

where V_{ij} → Weight matrix b/w Input-Hidden layers.

X_i → Input to each node I of Input layer.

Y_j → Output of each node j of hidden layer.

N → No of Input layer nodes.

NH → No of Hidden layer nodes.

Eq. (1) is somewhat imperfect equation and may be written as

$$\left[X\right]_{(Ns\times N)} \times \left[V\right]_{(N\times 1)} + \left[\eta\right]_{(Ns\times 1)} = \left[Y\right]_{(Ns\times 1)} \quad (2)$$

Since the 'y_1' values for each sample are not known initially, they may be taken as multiples of the mean of input values of that particular training samples.

$$y_i^j = r_t \bar{x}^{(i)} = r_t\text{* Mean value of } (i)^{th} \text{sample } (x_1^{(i)},\ x_2^{(i)}, \ldots, x_N^{(i)})$$

where r_t = 1 for i=1 and j=1

r_t = Random number between (0.85-1.25), which is different for each $y_i^{(j)}$, i ∀ 2 to NH , j ∀ 2 to Ns

Using the state estimation approach, we can evaluate the weight matrix[V] corresponding to the first hidden node.

$$\left[V\right]_{(N\times 1)} = \left[X_{effe}\right]^{-1}_{(N\times N)} \times \left[Y_{effe}\right]_{(N\times 1)} \ldots(3)$$

where $\begin{matrix}[X_{\text{effe}}]=[X^{T}R^{-1}X]\ ; \\ [Y_{effe}]=[X^{T}R^{-1}Y\ ]\end{matrix}$ R^{-1} is chosen as identity matrix whose dimension is Ns X Ns.

Similarly the $[V]_{(N\times j)}$ matrix is evaluated using (3) for each hidden node '*j*', and it is repeated for all hidden nodes to get the final combined total matrix of $[V]_{(N\times NH)}$ between Inputs, and Hidden layer nodes. This completes estimation of weights between Input & Hidden layers.

And now the ouput equation with the weights between (Hidden-Output) nodes can be written as

$$[Y]_{(Ns\times NH)}\times[W]_{(NH\times 1)}=[Out]_{(Ns\times 1)}. \quad (4)$$

W_{ij} → Weight matrix between Hidden-Output layers.

Out → Output of Output layer.

NO → No of Output layer nodes

Equation (4) is some what imperfect equation and it can be written as

$$[Y]_{(Ns\times NH)}\times[W]_{(NH\times 1)}+[\eta]_{(Ns\times 1)}=[Out]_{(Ns\times 1)}. \quad (5)$$

Since target values are known using the state estimation approach we can estimate the weights using

$$[W]_{(NH\times 1)}=[Y_{effe}]^{-1}_{(NH\times NH)}\times[Out_{effe}]_{(NH\times 1)}. \quad (6)$$

where $\begin{matrix}[Y_{\text{effe}}]=[Y^{T}R^{-1}Y]\ ; \\ [Out_{effe}]=[Y^{T}R^{-1}Out]\end{matrix}$

R^{-1} is chosen as identity matrix whose dimension is Ns X Ns.

Then we get the weight matrix $[W]_{(NH\times 1)}$ between (Hidden-Output) layers for the first output. Similarly $[W]_{(NH\times j)}$ needs to be evaluated for j=1 to NO. Now the Network is fully trained with the given samples and is ready for testing[forecasting of new prices]. Now just supply the test input from Input to the Output layer as we did in Neural Networks, we can get the forecasted output which is closer to the actual output.

State Estimation Neural Network- *direct* technique:

As the data pertaining to hidden node is not known initially, instead of generating hidden node, train the network without hidden layer (i.e. eliminating Hidden layer). Then the network structure now becomes very simple with 'N' input nodes and 'NO' output node. The problem now simplifies to estimating only one weight matrix with both input and output data readily available. The input layer outputs are directly propagated to the output layer, which further reduces the computational burden and time of execution.

If the input vector is written as[A] and the target vector is written as[Z] connected by the weight matrix[W] then the first output value for all the training samples is represented as

$$[A]_{(Ns\times N)}\times[W]_{(N\times 1)}=[Z]_{(Ns\times 1)} \quad (7)$$

Equation (7) is some what imperfect equation and it can be written as

$$[A]_{(Ns\times N)}\times[W]_{(N\times 1)}+[\eta]_{(Ns\times 1)}=[Z]_{(Ns\times 1)} \quad (8)$$

Since target values are known using the state estimation approach we can estimate the weights using

$$[W]_{(N\times 1)}=[A_{effe}]^{-1}_{(N\times N)}\times[Z_{effe}]_{(N\times 1)} \quad (9)$$

where $\begin{matrix}[A_{\text{effe}}]=[A^{T}R^{-1}A]\ ; \\ [Z_{effe}]=[A^{T}R^{-1}Z]\end{matrix}$

Similarly $[W]_{(N\times j)}$ needs to be evaluated for j=1 to NO.

Then the Network is fully trained with the given samples and is ready for testing[forecasting of new prices]. The SENN direct technique is used in this paper with 38 inputs and one output.

3. Flow Charts of ANN Models

Flow-charts of the four different ANN models i.e SENN-Direct, BPNN, RBFNN and GANN are given below in Fig.5, 6, 7 and 8 respectively.

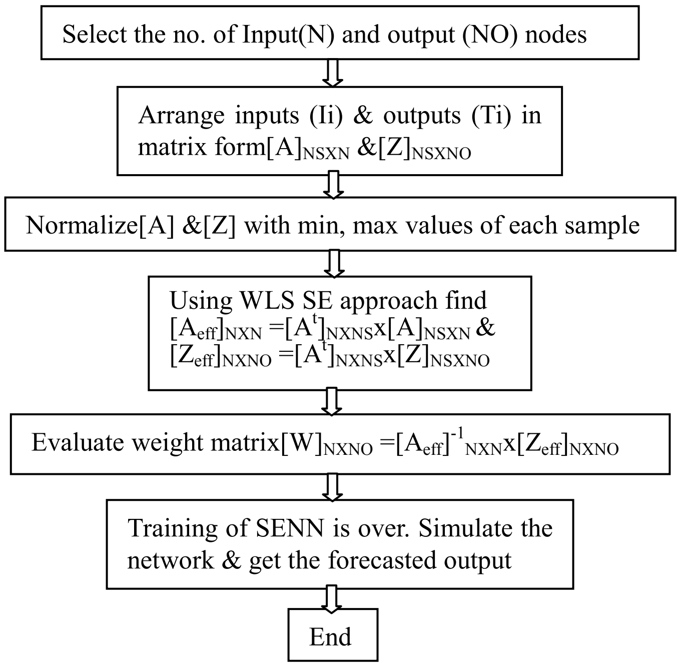
Select the no. of Input(N) and output (NO) nodes
Arrange inputs (Ii) & outputs (Ti) in matrix form[A]NSXN &[Z]NSXNO
Normalize[A] &[Z] with min, max values of each sample
Using WLS SE approach find [Aeff]NXN =[At]NXNSx[A]NSXN & [Zeff]NXNO =[At]NXNSx[Z]NSXNO
Evaluate weight matrix[W]NXNO =[Aeff]-1NXNx[Zeff]NXNO
Training of SENN is over. Simulate the network & get the forecasted output
End

Figure 5. Flow chart for SENN-*direct* algorithm

Error Analysis:

For each of above four models error analysis is done as below.

Daily mean error $DME=\frac{1}{24}\sum_{i=1}^{24}\frac{|Pa(i)-Pf(i)|}{Pa(i)}\times 100;$

Mean Absolute Percentage Error $MAPE=\frac{1}{24}\sum_{i=1}^{24}\frac{|Pa(i)-Pf(i)|}{Pa(i)}\times 100;$

Input Variables selection:

Different sets of lagged Prices have been proposed as input features for the Price prediction in the electricity market. Bearing in mind the daily and weekly periodicity and trend of the Price signal, the set of *38* inputs, is considered in this report as the input features. The selected input features are lagged Prices recommended in several papers. P_h represents the Price at hour h which is the single output of the training samples. P_{h-1} stands for (h-1)th

$$\{P_{h-1}, P_{h-2}, P_{h-3}, P_{h-4}, P_{h-5}, P_{h-6}, P_{h-24}, P_{h-25}, P_{h-26}, P_{h-27}, P_{h-48}, P_{h-49}, P_{h-50}, P_{h-51}, P_{h-72}, P_{h-73}, P_{h-74}, P_{h-75}, P_{h-96}, P_{h-97}, P_{h-98}, P_{h-99}, P_{h-120}, P_{h-121}, P_{h-122}, P_{h-123}, P_{h-144}, P_{h-145}, P_{h-146}, P_{h-147}, P_{h-168}, P_{h-169}, P_{h-170}, P_{h-171}, P_{h-192}, P_{h-193}, P_{h-194}, P_{h-195}\} = 38 \text{ inputs}$$

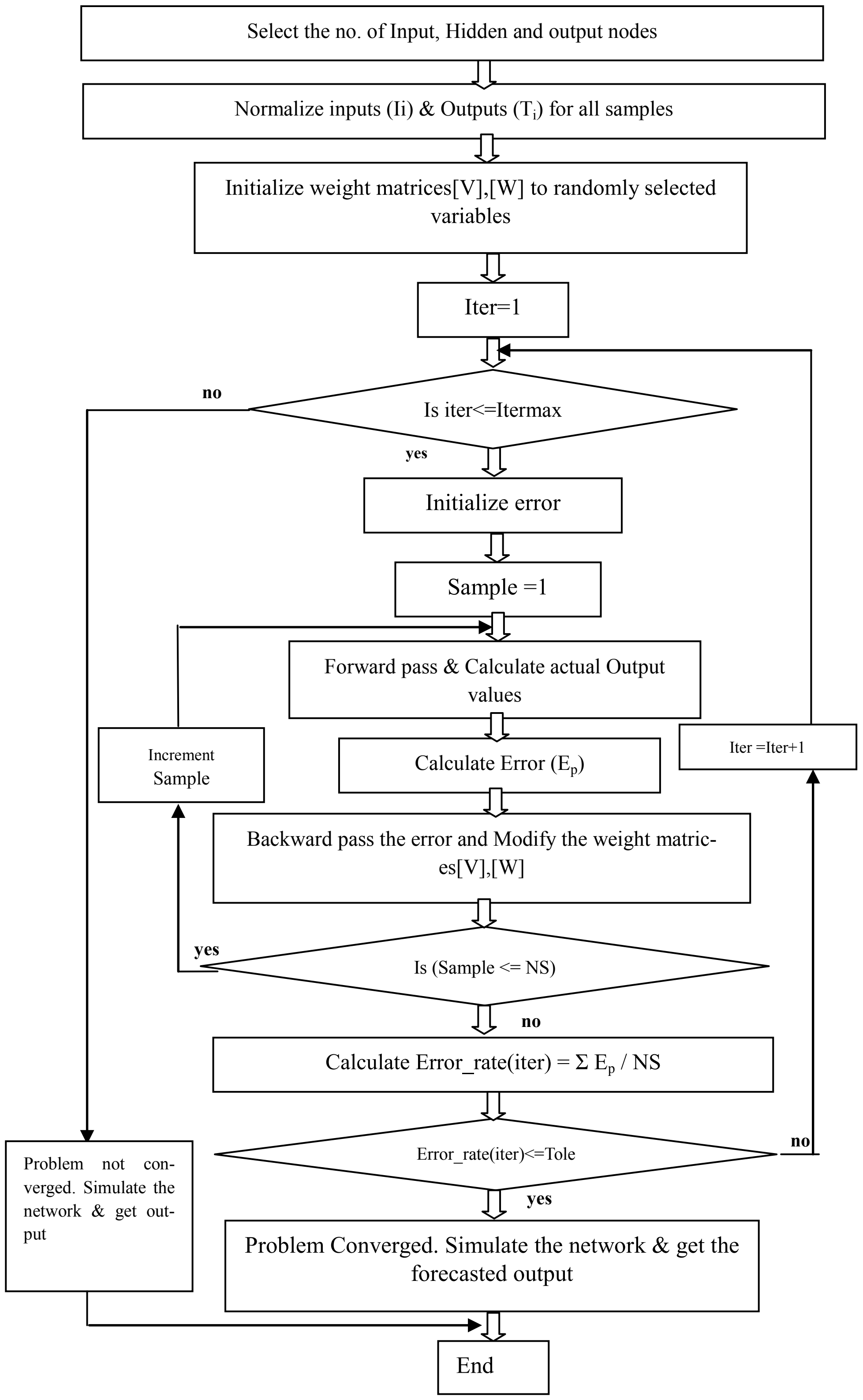
Select the no. of Input, Hidden and output nodes
Normalize inputs (Ii) & Outputs (Ti) for all samples
Initialize weight matrices[V],[W] to randomly selected variables
Iter=1
Is iter<=Itermax
no
yes
Initialize error
Sample =1
Forward pass & Calculate actual Output values
Calculate Error (Ep)
Increment Sample
Iter =Iter+1
Backward pass the error and Modify the weight matric-es[V],[W]
Is (Sample <= NS)
yes
no
Calculate Error_rate(iter) = Σ Ep / NS
Error_rate(iter)<=Tole
no
yes
Problem not converged. Simulate the network & get output
Problem Converged. Simulate the network & get the forecasted output
End

Figure 6. Flow-chart of BPNN Model

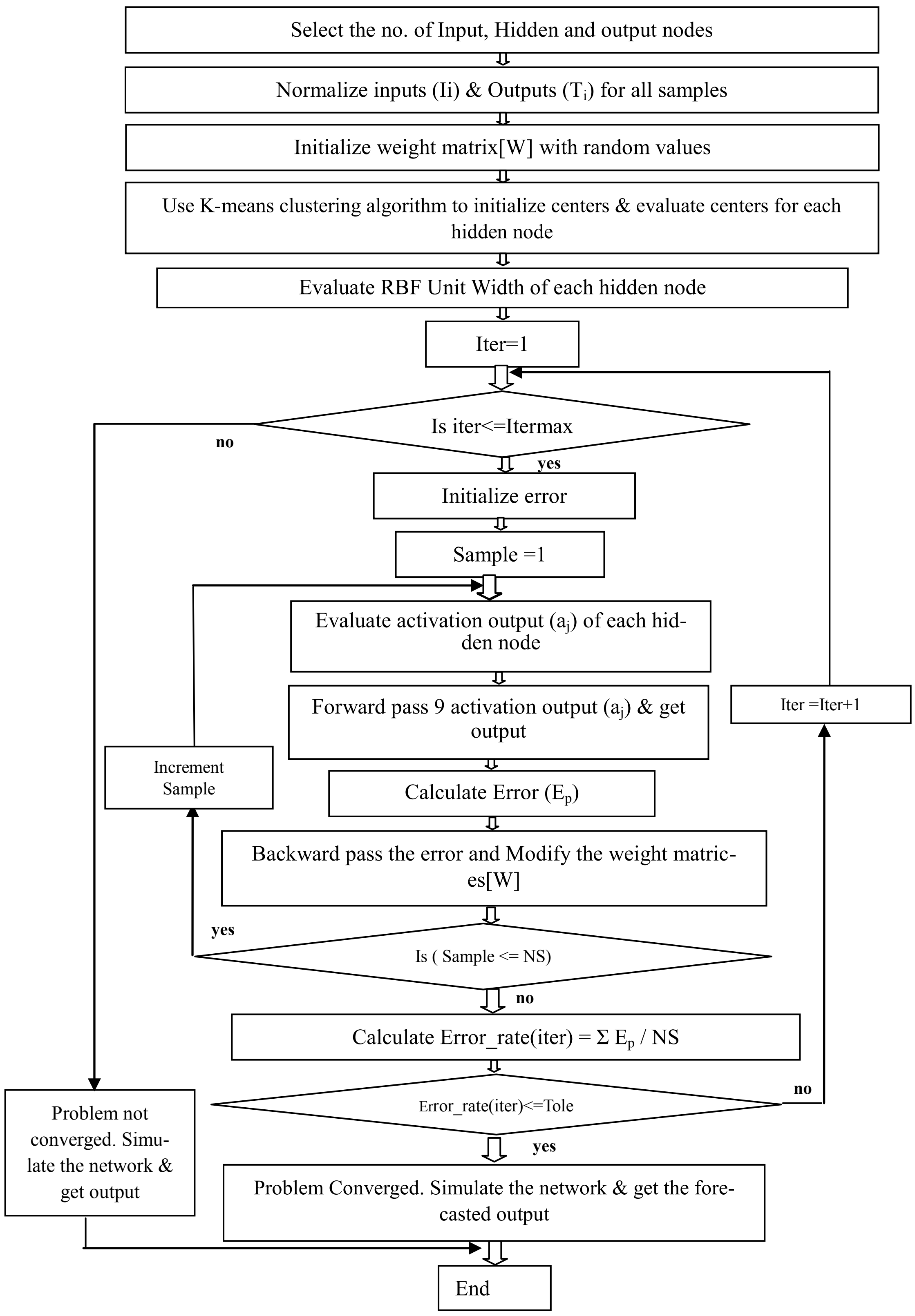
Select the no. of Input, Hidden and output nodes
Normalize inputs (Ii) & Outputs (Ti) for all samples
Initialize weight matrix[W] with random values
Use K-means clustering algorithm to initialize centers & evaluate centers for each hidden node
Evaluate RBF Unit Width of each hidden node
Iter=1
Is iter<=Itermax
no
yes
Initialize error
Sample =1
Evaluate activation output (aj) of each hidden node
Forward pass 9 activation output (aj) & get output
Iter =Iter+1
Increment Sample
Calculate Error (Ep)
Backward pass the error and Modify the weight matrices[W]
yes
Is (Sample <= NS)
no
Calculate Error_rate(iter) = Σ Ep / NS
no
Error_rate(iter)<=Tole
Problem not converged. Simulate the network & get output
yes
Problem Converged. Simulate the network & get the forecasted output
End

Figure 7. Flow-chart of RBFNN Model

Training Period:

Functional relationships of hourly Price, especially in the developed countries, rapidly vary with time. So, the training period of Price prediction is more limited. On the other hand, if the training period is selected to be very short, then the NN can not derive all functional relationships of the Prices due to the small number of training samples. The last *48* days has been proposed as the training period of the NN for the Prices prediction in the electricity market. Then it forecasts the Prices of the next *24* h (one day ahead), which are unseen for the NN. A total of 1152 training data elements are available. Out of these, first *8* days data is set aside as buffer & training is started from 9th day.

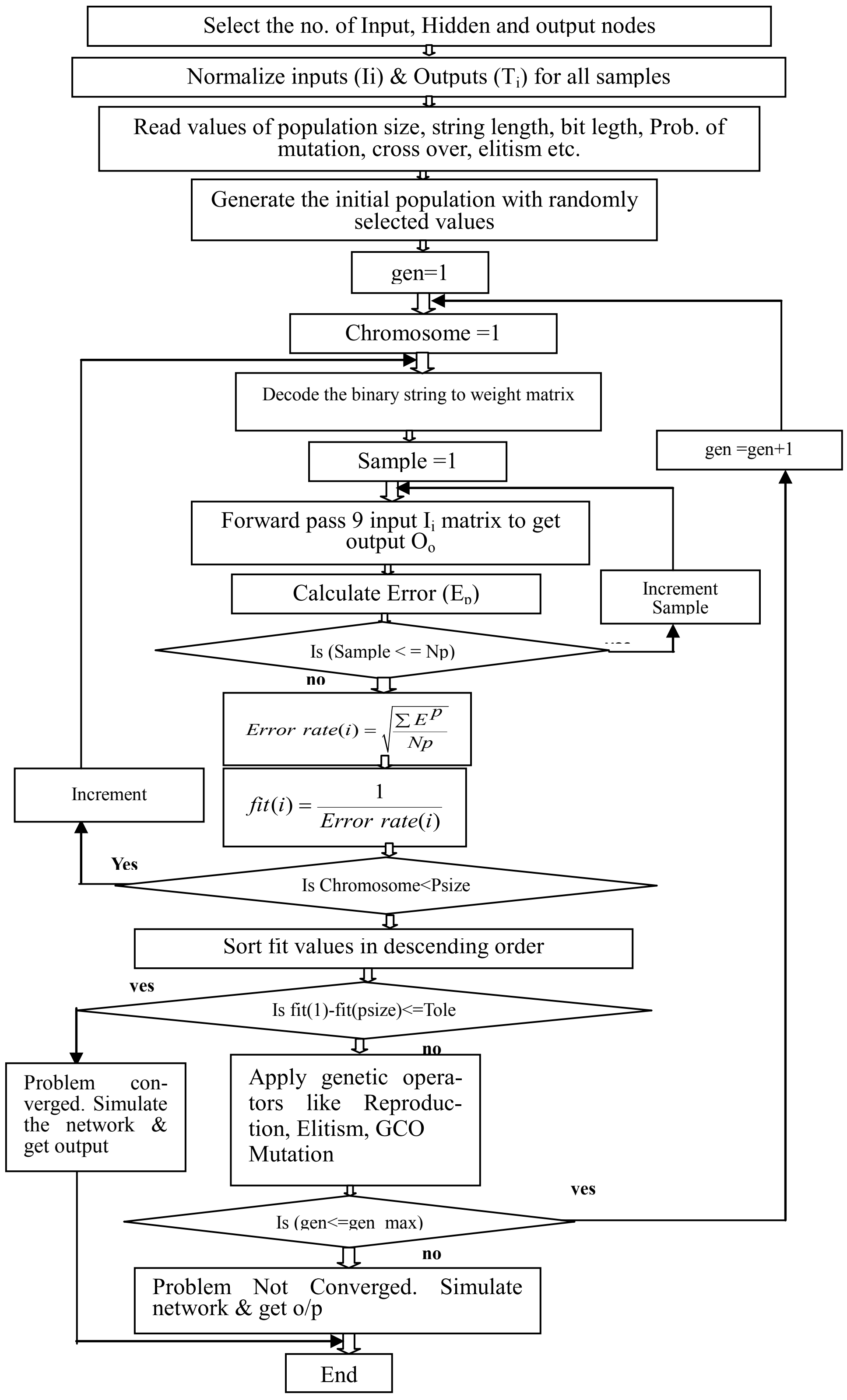
Select the no. of Input, Hidden and output nodes
Normalize inputs (Ii) & Outputs (T_i) for all samples
Read values of population size, string length, bit legth, Prob. of mutation, cross over, elitism etc.
Generate the initial population with randomly selected values
gen=1
Chromosome =1
Decode the binary string to weight matrix
gen =gen+1
Sample =1
Forward pass 9 input I_i matrix to get output O_o
Calculate Error (E_p)
Increment Sample
Is (Sample < = Np)
no
$Error\ rate(i) = \sqrt{\frac{\sum E^p}{Np}}$
Increment
$fit(i) = \frac{1}{Error\ rate(i)}$
Yes
Is Chromosome<Psize
Sort fit values in descending order
yes
Is fit(1)-fit(psize)<=Tole
no
Problem converged. Simulate the network & get output
Apply genetic operators like Reproduction, Elitism, GCO Mutation
yes
Is (gen<=gen max)
no
Problem Not Converged. Simulate network & get o/p
End

Fig 8. Flow-chart of GANN Model

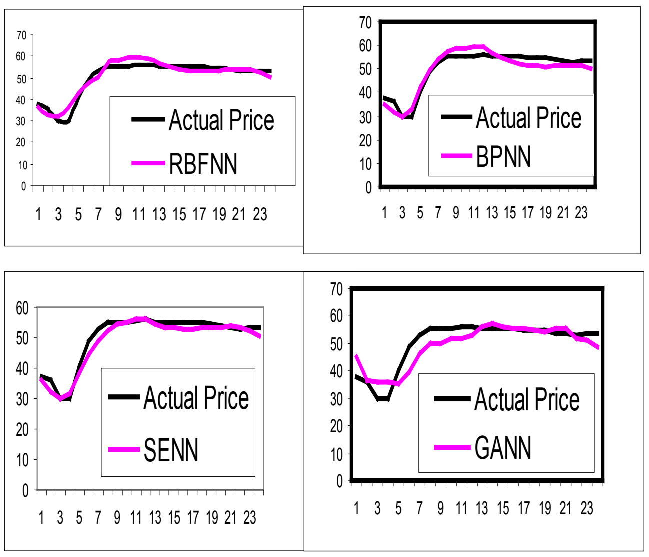

Figure 9. Actual Price and forecasted price using four models

Three months data (April, May and June of 2011) is downloaded from the Iberian Energy Derivatives Exchange (www.omip.pt). A case study with the downloaded data of the day-ahead pool market prices of mainland Spain is made with the above four different ANN models and price for 24 hours (from 3rd hour of 29th June 2011 to 2nd hour of 30th June 2011) is forecasted and the results are shown graphically in Fig.9.

Daily mean error is calculated for the results obtained using the four models. Studies are performed by programming in MATLAB using Pentium IV, 2 GHz, 512 MB RAM. DME and Training time using the four models is given in Table 1.

The actual and forecasted values of prices along with MAPE for the forecasted period are tabulated in Table 1 given below.

Table 1. Daily Mean Error & Training Time of 4 Models

	BPNN	GANN	RBFNN	SENN
Daily mean error	4.9198	7.24679	4.72254	3.449
TrainingTime (sec)	75.172	175.984	85.953	3.563

4. Results

Table 2. Actual, Forecasted Prices and MAPE (29th June 2011)

Actual Price	Forecasted Price				MAPE			
	BPNN	GANN	RBF NN	SENN	BPNN	GANN	RBF NN	SENN
37.5	35.2	44.76	36.08	36.03	6.22	19.36	3.79	3.93
36	31.7	36.74	32.76	32.22	11.8	2.063	8.99	10.5
30	29.9	36.05	31.89	30.07	0.18	20.18	6.31	0.25
30	33.3	36.19	36.25	31.68	10.8	20.64	20.8	5.59
40.42	42.3	35.52	42.86	38.18	4.54	12.12	6.04	5.55
49	49.7	39.28	47.36	44.67	1.46	19.83	3.34	8.83
53.05	54	46.53	50.44	48.91	1.75	12.3	4.91	7.8
55.23	57.2	49.67	57.26	52.18	3.58	10.06	3.68	5.52
55.23	58.6	50.01	58.44	54.54	6.06	9.458	5.81	1.26
55.23	58.7	51.73	59.71	55.2	6.24	6.336	8.11	0.06
55.74	59.3	52.03	59.36	56	6.35	6.664	6.50	0.46
55.96	59.2	52.87	58.81	56.07	5.73	5.516	5.10	0.2
55.23	56.8	56.13	57.03	54.68	2.82	1.624	3.26	0.99
55.23	54.6	57.37	55.51	53.59	1.21	3.88	0.51	2.97
55.23	53.5	55.83	53.94	53.28	3.21	1.089	2.33	3.54
55.23	52.2	55.28	53.18	52.86	5.58	0.099	3.71	4.29
55.03	51.4	55.09	52.97	52.73	6.58	0.116	3.74	4.17
54.92	51.3	54.81	53.10	53.14	6.55	0.193	3.32	3.24
54.68	51	54.28	53.13	53.17	6.71	0.732	2.83	2.77
53.85	51.2	55.33	53.75	53.25	4.92	2.756	0.19	1.12
53.5	51.7	55.37	53.77	53.88	3.32	3.496	0.51	0.71
52.92	51.8	51.89	54.16	53.15	2.16	1.944	2.34	0.43
53.5	51.4	51.15	52.42	51.98	3.9	4.395	2.02	2.84
53.45	50.1	48.59	50.69	50.37	6.34	9.087	5.17	5.77

5. Conclusions

Four different models of ANN are used for short term price forecasting using the downloaded data of the Iberian Energy Derivatives Exchange. The next 24 hours prices are forecasted and the test results are compared. Comparision of Daily mean error and the training time taken for training the network clearly establishes the supremacy of the direct non-iterative state estimation based neural network over the other three methods and may be used as a better alternative for the short term price forecasting solution.

REFERENCES

[1] "Power System Restructuring and Deregulation – Trading, Performance and Information Technology" Edited by Loi Lei Lai, John Wiley & Sons Ltd, Chichester.

[2] Mohammed Shahidehpour, Hatim Yamin, Zuyi Li, "Market Operations in Electric Power System – Forecasting, Scheduling and Risk managements", John Wiley & Sons Ltd, New York

[3] K.Bhattacharya, M.H.J.Bollen and J.E.Daalder, "Operation of Restructured Power Systems", Kluwer Academic Publishers, Boston, 2001

[4] Website:http://www.omip.pt/Downloads/SpotPrices/tabid/296/language// en-GB/Default.aspx

[5] J. Contreras, R. Esp´ınola, F.J. Nogales, A.J. Conejo, "ARIMA models to predict next-day electricity prices", IEEE Transactions on Power Systems. Vol.18, No.3, 2003, pp. 1014–1020.

[6] F.J. Nogales, J. Contreras, A.J. Conejo, R. Esp´ınola, "Forecasting next-day electricity prices by time series models", IEEE Transactions on Power Systems, Vol.17, No.2, 2002, pp. 342–348.

[7] Szkuta B.R, Sanabria L.A,. Dillon T.S, "Electricity price short-term forecasting using artificial neural networks", IEEE Transactions on Power Systems, Vol.14, No.3, Aug. 1999, pp.851-857

[8] Yamin, M. Shahidehpour and Z. Li, "Adaptive short-term Price Forecasting using artificial Neural Networks in the Restructured Power Markets," Electrical Power and Energy Systems, Vol. 26, 2004, pp. 571-581.

[9] Alicia Troncoso Lora, Jesús M. Riquelme Santos, Antonio Gómez Expósito, José Luis Martínez Ramos, and José C. Riquelme Santos, "Electricity Market Price Forecasting Based on Weighted Nearest Neighbors Techniques", IEEE Transactions on Power Systems, Vol. 22, No. 3, Aug. 2007, pp.1294-1301

[10] S N Pandey, S Tapaswi, L Srivastava, "Locational Marginal Price Projection using a Novel RBFNN Approach in Spot Power Market", 2008 Joint International Conference on Power System Technology Powercon and IEEE Power India Conference Vol. 1 and 2 ,2008, pp: 1-7

[11] V. Vahidinasab, S. Jadid, A. Kazemi, "Day-ahead price forecasting in restructured power systems using artificial neural networks", Electric Power Systems Research, Vol. 78, Issue 8, Aug. 2008, pp.1332-1342

[12] J.P.S.Catalao,S.J.P.S. Mariano,V.M.F. Mendes,L.A.F.M. Ferreira , "Short-term electricity prices forecasting in a competitive market: A neural network approach", Electric Power Systems Research, Vol. 77, 2007, pp.1297-1304

[13] Prakash K, Sydulu M, "Non Iterative-State Estimation Based Neural Network for Short Term Load Forecasting of Distribution Systems", Proceedings of Power & Energy Society General Meeting, PES '09, 26-30 July 2009, pp.1 - 8

[14] Chen Yan-Gao, Ma Guangwen, "Electricity Price Forecasting Based on Support Vector Machine Trained by Genetic Algorithm", Third International Symposium on Intelligent Information Technology Application, pp.292 – 295

[15] L.M. Saini, S.K. Aggarwal A. Kumar, "Parameter optimisation using genetic algorithm for support vector machine-based price-forecasting model in National electricity market", IET Generation, Transmission & Distribution, 2010, Vol. 4, Iss. 1, pp. 36–49

[16] Pavlos S. Georgilakis, "Artificial Intelligence Solution to Electricity Price Forecasting Problem", International Journal of Applied Artificial Intelligence , Vol. 21, Iss. 8, 2007

[17] YanBin Xu and Ken Nagasaka, " A research on spike jump of Electricity price in the Deregulated Power markets", International Journal of Electrical and power Engineering, vol. 3 (2), pp. 99-104, 2009