American Journal of Applied Sciences 5 (6): 705-713, 2008
ISSN 1546-9239

Vulnerability Assessment of Power System Using Radial Basis Function Neural Network and a New Feature Extraction Method

Ahmed M.A. Haidar, Azah Mohamed and Aini Hussain
Department of Electrical, Electronic and System Engineering,
Universiti Kebangsaan Malaysia UKM 43600 Bangi, Selangor, Malaysia

Abstract: Vulnerability assessment in power systems is important so as to determine how vulnerable a power system in case of any unforeseen catastrophic events. This paper presents the application of Radial Basis Function Neural Network (RBFNN) for vulnerability assessment of power system incorporating a new proposed feature extraction method named as the Neural Network Weight Extraction (NNWE) for dimensionality reduction of input data. The performance of the RBFNN is compared with the Multi Layer Perceptron Neural Network (MLPNN) so as to evaluate the effectiveness of the RBFNN in assessing the vulnerability of a power system based on the indices, power system loss and possible loss of load. In this study, vulnerability analysis simulations were carried out on the IEEE 300 bus test system using the Power System Analysis Toolbox and the development of neural network models were implemented in MATLAB version 7. Test results prove that the RBFNN give better vulnerability assessment performance than the multilayer perceptron neural network in terms of accuracy and training time. The proposed feature extraction method decreases the training time drastically from hours to less than seconds, this bound to influence the vulnerability classification and increase the speed of convergence. It is also concluded that the reduction in error is achieved by using PSL as an output variable of ANN, in all the cases the error of RBFNN output by PSL is less than 4.87% which is well within tolerable limits.

Key words: Vulnerability assessment, vulnerability index, radial basis function, feature extraction

INTRODUCTION

Power system operation has become more complex due to the growing demand for electricity and increase in system interconnection. Such a highly interconnected power system when greatly stressed may be vulnerable to cascading failures occurring due to a series of low probability events. If such events occur, operators have to guarantee the safety of the main parts of a power system and to ensure continuity of power supply to some important infrastructures such as transportation and communication. A power system is considered invulnerable if it can withstand all unpredicted natural disasters such as earthquake and flood as well as disturbances initiated by heavy loading conditions and human errors. On the other hand, a power system is vulnerable if it is susceptible to hazards that substantially reduce its ability to maintain its intended function. The concept of power system vulnerability assessment combines information on the level of system security as well as information on a wide range of scenarios, events and contingencies with regards to which a system is vulnerable so that preventive and emergency control steps can be taken to minimize catastrophic power outages and reduce the associated risk.

Power system vulnerability assessment covers almost all aspects of power system and it requires analysis of the system behavior under a prescribed set of events known as contingencies such as Line Outage (LO), Generator Outage (GO), increase in total load and amount of load disconnected. Conventionally, such analysis is done by simulating all the contingencies which is very time consuming for a large size power system. Hence, vulnerability assessment by contingency analysis is not feasible for real-time application. Methods for vulnerability analysis usually consider either risk analysis or based on performance indicators to assess if the vulnerability of a power delivery system has changed. These indicators are usually refereed to as vulnerability indices[1,2] in which it give a quantitative measure to assess the degree of

Corresponding Author: Ahmed M.A. Haidar, Department of Electrical, Electronic and System Engineering,Universiti Kebangsaan Malaysia UKM 43600 Bangi, Selangor, Malaysia

vulnerability. To overcome the time consuming computation of vulnerability indices using the conventional contingency analysis, artificial intelligent technique is proposed for vulnerability assessment of power systems. It has the potential advantage over conventional techniques in significantly improving the accuracy in pattern recognition. In addition, a trained ANN can quickly map nonlinear relationship between input and output data which is considered suitable for online use.

The first step in applying neural networks to power system vulnerability assessment is the creation of an appropriate training data set. A common approach is to simulate the system in response to various disturbances and then collect a set of pre-disturbance system features along with the corresponding system vulnerability index[3]. The vulnerability indices used in this study are power system loss[4] and Possible Loss of Load[2].

In this study, the Generalized Regression Neural Networks (GRNN) is proposed for assessing vulnerability of power system based on vulnerability indices using power system loss and possible loss of load. A new feature extraction technique NNWE is also proposed for reducing the input features so as to speed up the neural network training process. This work's main objective is to develop a fast and accurate vulnerability assessment method for a large sized power system with a high dimensional datasets using improved computational intelligent technique. Three different artificial neural network architectures have been selected to do this work. The MLPNN and RBFNN architectures were selected because they have been previously used in other works with success[5-10]. GRNN, is considered a developed model of RBFNN architecture, it was chosen because it is indicated to classifying problems and for the fact that it has never been used before to create pattern recognition systems on vulnerability assessment. The paper is organized as such that in section 2, formulation of the vulnerability indices using power system loss and possible loss of load is given. In section 3, background theories of RBF and MLP neural networks architectures and training algorithms are described. Section 4 describes the ANN implementation for vulnerability assessment of power system and the proposed feature extraction technique. Results and conclusions are given in sections 5 and 6, respectively.

VULNERABILITY INDICES

The vulnerability indices considered for assessing vulnerability of power system are based on power system loss (PSL) and possible loss of load (PLL). The formulations of the indices are described as follows:

Vulnerability index based on power system loss: The vulnerability index which is based on PSL considers total system loss, generation loss due to generation outage, power line loss due to line outage, increase in total load and amount of load disconnected. The rational for considering PSL is due to the fact that losses in a power transmission system are a function of not only the system load but also of the generation. In addition, each contingency has an effect not only on the system performance but also on power losses in the system. The outage of transmission line, transformer or generator may result in line overload and causes increased active power loss in lines and reactive power loss in transformers. Similar effect may result if a contingency such as loss of load is said to occur. Therefore, it is important to consider total power system loss as a quantitative measure for assessing vulnerability of power systems[4].

The formulation of the PSL index is given by;

$$PSL=\frac{\sqrt{P_{BCL}^2+Q_{BCL}^2}}{\left(\sqrt{P_{CCL}^2+Q_{CCL}^2}\right)+S_{IL}+S_{LD}+\sum_{i=1}^{n}S_{LGO,i}W_{G,i}+\sum_{i=1}^{m}S_{LLO,i}W_{L,i}} \quad (1)$$

Where:

P_{BCL}, Q_{BCL} = Active and reactive powers of system loss at base case
P_{CCL}, Q_{CCL} = Active and reactive powers of system loss at contingency case
S_{IL} = Total load increase
S_{LD} = Amount of load disconnected
$S_{LGO, i}$ = Loss of generated MVA due to generator outage
$S_{LLO, I}$ = Loss of transported MVA due to line outage
$W_{G,I}$ = Weight of individual generator power output
$W_{L, I}$ = Weight of individual line power influence
n = Number of generators
m = Number of lines

From Eq. 1, it can be noted that the PSL index has values in the range of 1-0. These values can be categorized based on vulnerability boundaries such that if the PSL value is high, for example, in the range of (0.6-1), it indicates that the system is invulnerable, whereas if the PSL value is low, that is, in the range of (0-0.59), it indicates that the system is vulnerable.

These index values can be readjusted by a control operator based on any new system configuration considering the weights of power system components.

Vulnerability index based on possible loss of load: The vulnerability index based on possible loss of load takes into consideration the fact that if unpredicted natural disasters happen which may be due to earthquake or flood, operators will need to shedding some load to guarantee the safety of the main parts of a system and supply power to some important infrastructures. So the structural vulnerability of power grid can be defined as possible loss of load due to the amount of load shed. Thus, the PLL index is the ratio of loss of load in a system which is given by;

$$PLL = \frac{\sum_{i}^{n} S_{shed}}{S_{\Sigma}} \quad (2)$$

Where:

PLL = Possible loss of load
S_{shed} = Load shed at ith substation
$S\Sigma$ = Total system load

PLL index is similar to the anticipated loss of load index which is based on the amount of load shed that may be lost due to a contingency in order to avoid a cascading outage. If more load is shed means that a system becomes more vulnerable and therefore the system is said to be less capable of resisting emergencies[11]. To assess vulnerability of power systems using PLL, it is if the value of PLL is greater than the value at the base case, it indicates that the system is vulnerable[2].

NEURAL NETWORK ARCHITECTURE AND TRAINING ALGORITHM

ANN is a computational tool which attempts to simulate the architecture and internal operational features of the human brain and its nervous system. Two commonly used network architectures for function approximation are RBF and MLP neural networks (NN). These models are considered in this work and will be discussed accordingly. The MLPNN architecture using back-propagation learning is one of the most popular neural networks. It consists of at least three layers of neurons, namely, an input layer, one or more hidden layers and an output layer. The hidden and output layers usually have nonlinear activation function when the MLPNN is used to solve nonlinear regression

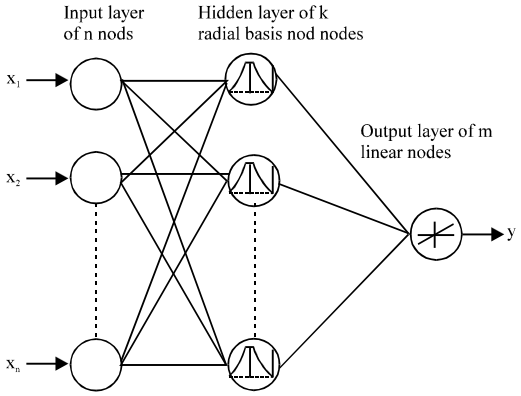

Fig. 1: Structure of typical RBFNN model

problem. The back-propagation is a supervised learning algorithm that uses two passes through the network to calculate the change in network weights. In the forward pass, the weights are fixed and the input vector is propagated through the network to produce an output. An output error is calculated from the difference between actual output and the desired output. This is then propagated backwards through the network, making changes to the weights as required. The back propagation training algorithm using gradient descent with momentum is slow compared to adaptive training algorithm using conjugate gradient which has the advantages of i) smoothing the oscillations across narrow valleys, ii) amplifying the learning rate when all the weights change in the same direction and iii) enables the algorithm to escape from shallow local minima.

An extremely powerful neural network type is the RBFNN which has a simple architecture of three layers known as input, hidden and output layers as shown in Fig. 1. The input layer is made up of n nodes, where n is the dimension of the input vector X. The input of the network passes to the hidden layer by performing a nonlinear mapping from the input space to a new space. The hidden layer is made up of k nodes, with radial activation functions called as radial basis functions. Each of the input components fits forward to the radial functions, whereas the outputs of these functions are linearly combined with a set of weights into the output layer. The hidden and the output layers have biases. A characteristic feature of radial function is that its response decreases or increases monotonically with distance from a central point named as center of the radial function. These functions involve two parameters, the center and distance scales.

The most common choice for this radial basis function is the Gaussian function which has a peak at

the center and decreases monotonically as the distance from the center increases. As such, the output of the kth hidden node, $g_k(x)$ is a radial basis function which is described by;

$$g_k(x) = \exp(-\|x - c_k\|^2 / \sigma_k^2) \quad (3)$$

Where:

c_k = Center of the Gaussian function, g_k which represents the weight between the k^{th} hidden node and the input node

σ_k = Width or spread of g_k

$\| \|$ = Euclidean norm

In other words, each node in the hidden layer has a finite spherical activation region, determined by the Euclidean distance between the input vector X and the center c_k of the function g_k. The width σ_k can be viewed as a distance scaling parameter which determines the region of the input space over which the node has an appreciable response. It is obvious that the radius of the Gaussian function g_k shrinks as the scaling factor spread, σ_k decreases. The scaling factor spread determines the width of an area in the input space to which each radial basis function responds. A large value of spread results in overlapping regions in the input space and therefore leads to higher classification error. However, a small value of spread will show a good classification on the training data, but low performance in generalization.

The output layer is made up of m nodes, but for this case, there is only one node as shown in Fig. 1. The network output is a m-dimensional vector, where the m^{th}component of $y_m(x)$ is given by the following equation:

$$y_m(x) = \sum_{i=1}^{k} w_{mi} g_i(x) \quad (4)$$

where, w_{mi} is the weight between hidden node i and output node m.

Another form of RBFNN architecture is the generalized regression neural network (GRNN) which is a kind of normalized RBFNN. It is similar to the RBFNN in the input and hidden layers, but is slightly different in the output layer. The GRNN implements the Bayesian decision strategy to classify input vectors. A schematic diagram of GRNN is depicted in Fig. 2 in which it consists of four layers, namely, input layer, pattern layer, summation layer and output layer. The number of input units in the first layer is equal to

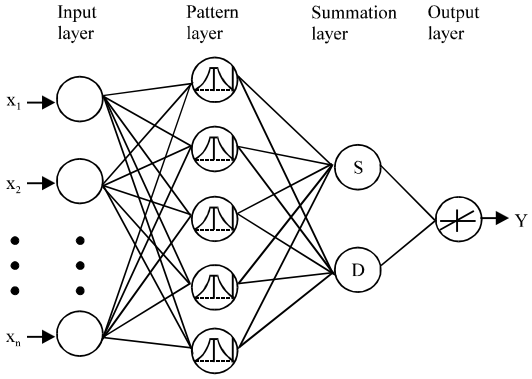

Fig. 2: Structure of GRNN model

independent factors, x_i. Only the hidden layer has biases. The first layer is fully connected to the pattern layer, whose output is a measure of the distance of the input from the stored patterns. Each pattern layer unit is connected to the two neurons in the summation layer, known as S summation neuron and D summation neuron[12]. The S summation neuron computes the sum of the weighted outputs of the pattern layer while the D summation neuron calculates the unweighted outputs of the pattern neurons. For D-summation neuron, the connection weight is set to unity. The output layer merely divides the output of each S-summation neuron by that of each D-summation neuron, yielding the predicted value expressed as:

$$\hat{y}_i = \frac{\sum_{i=1}^{n} y_i \exp[-D(x, x_i)]}{\sum_{i=1}^{n} \exp[-D(x, x_i)]} \quad (5)$$

where, n is the number of independent input variables, y_I is the target output value corresponding to the i^{th} input pattern and the Gaussian D function is defined as:

$$D(x, x_i) = \sum_{j=1}^{p} \left(\frac{x_j - x_{ij}}{\sigma_j} \right)^2 \quad (6)$$

where, p is the total number of training patterns and σ_i is generally referred to as the smoothing parameter (width or spread), whose optimal value is often determined experimentally.

Training of the RBFNN in general can be divided into two stages, that is, training in the hidden layer followed by training in the output layer. Training in the hidden layer is unsupervised and it involves determination of the centers and spread of the Gaussian

functions of the hidden nodes utilizing an appropriate clustering algorithm. On the other hand, training in the output layer uses a supervised method like the Least Mean Square (LMS) algorithm. The centers of the Gaussian functions are determined with the K-means clustering algorithm and the spreads are calculated using the second order nearest neighbor heuristic. The weights between the hidden and output layers are determined by minimizing the square error of the network output with the LMS algorithm[13].

There are variations in the training algorithm implemented in the conventional RBFNN, exact RBFNN and GRNN. In comparison with the conventional RBFNN, the GRNN and Exact RBFNN have a special property in which no iterative training of the weight vectors is required. That is, any input-output mapping is possible, by simply assigning the input vectors to the centroid vectors and fixing the weight vectors between the RBF units and outputs identical to the corresponding target vectors. This training algorithm is much better than the back propagation training which involves long and iterative training as well as facing the problem of local minima. Moreover, the special property of GRNN enables us to flexibly configure the network which is considered to be beneficial to real hardware implementation, with only two parameters, the center, c_k and width, σ_k to be adjusted. Since the radial basis function act as a detector for different input vectors, the weight vectors are computed accordingly and there is no need to train the network. The GRNN and Exact RBFNN are therefore straightforward and do not depend on a training process.

Intensive investigations were carried to select the applicable type of RBFNN for both feature extraction and for vulnerability assessment. Of these types such as conventional RBFNN, Exact RBFNN and GRNN, it was found that Exact RBFNN available to be used for feature extraction (the first ANN) and GRNN effective for using in vulnerability assessment implementation (the second ANN). By using these types of RBFNN the performance increased and the classification error of the NN decreased.

NEURAL NETWORK IMPLEMENTATION TO VULNERABILITY ASSESSMENT OF POWER SYSTEM

The implementation of neural network for vulnerability assessment of a power system is verified on the 300-bus IEEE test system which is characterized by large block of generators. The test system is divided into three areas as shown in Fig. 3 and it consists of

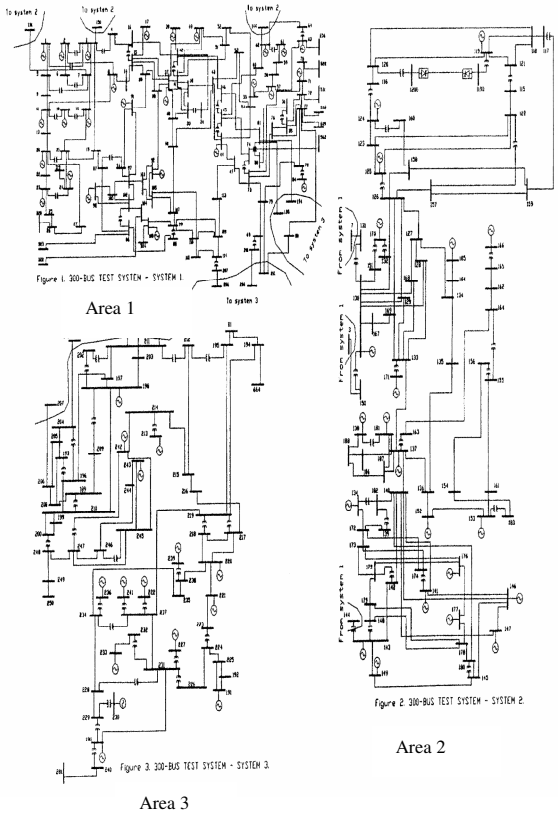

Fig. 3: Single line diagram of the 300-bus IEEE test system

69 generators, 116 transformers, 295 lines and 198 loads. The classes of voltage in the subsystems are 6, 13, 66, 115, 138, 230 and 345 kV.

The first step before applying ANN is to collect as many data as possible from the power system, in which the data are assumed to be of physical interest for vulnerability assessment. The data can be obtained from vulnerability analysis simulations carried out on the test system. The procedures involved in power system vulnerability assessment begin by first analyzing the system behavior at the base case condition. The next step is to analyze the system behavior when subjected to credible system contingencies by considering several test cases such as line outage, generator outage, increase in total load and amount of load disconnected. The vulnerability indices PSL and PLL are then calculated for each test case. In this study, simulations were carried out using the power system analysis and toolbox (PSAT) program and the vulnerability indices were calculated using the MATLAB program. For the calculation of the vulnerability indices, the weights of all the system parameters are set equal to one for

simplicity in calculation. In practice, system operators may assign different weights to represent the varying importance of selected elements in the system.

An important consideration prior to ANN implementation is the creation of an appropriate training data set. A common approach is to simulate the system in response to various disturbances and then collect a set of system features along with the corresponding system vulnerability index. The selection of input features is an important factor to be considered in ANN. The input features selected should be both available and measurable in a real power system.

Input and output variables: The number of input variables and training data sets depend on the size of a power system. For this work, the training data sets were obtained by simulating the system in response to various disturbances. A set of pre-disturbance system features along with the corresponding system vulnerability index are considered as the ANN input and output variables. The total input variables correspond to 413 features which comprise of real and reactive power flows and total real and reactive generations. The calculated vulnerability indices, PSL and PLL in response to various disturbances are separately used as output for each ANN with similar input variables.

The proposed feature extraction method: Feature extraction is the process of mapping all available features into a composite feature set of lower dimension. Here, dimensionality of a feature set is reduced by combining features while retaining the characteristics that allow for accurate classification. The feature extraction method proposed is named as the neural network weight extraction (NNWE) method. The procedure of the proposed NNWE method is described as follows:

- Determine the training data sets of the neural network for vulnerability assessment.
- Use a vulnerability index based on either PSL or PLL to select the critical contingencies. From the training data sets, select sub-data sets whose vulnerability index shows that the system is vulnerable. These sub-data sets are applied as training sets to train the ANN that is meant for feature extraction.
- Obtain the weights matrix which are represented by;

$$\begin{bmatrix} v_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41} + \ldots + x_n w_{n1} \\ v_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42} + \ldots + x_n w_{n2} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ v_m = x_1 w_{1m} + x_2 w_{2m} + x_3 w_{3m} + x_4 w_{4m} + \ldots + x_n w_{nm} \end{bmatrix} \quad (7)$$

where, v_m is the value of hidden neuron and x_n is the input variable.

The weights matrix w extracted from (7) is given as;

$$w = \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} \cdots & w_{n1} \\ w_{12} & w_{22} & w_{32} & w_{42} \cdots & w_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{1m} & w_{2m} & w_{3m} & w_{4m} \cdots & w_{nm} \end{bmatrix} \quad (8)$$

- Using the weights matrix given by Eq. 8, determine the values of the reduced feature sets by using;

$$R = wp = \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} \cdots & w_{n1} \\ w_{12} & w_{22} & w_{32} & w_{42} \cdots & w_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{1m} & w_{2m} & w_{3m} & w_{4m} \cdots & w_{nm} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \\ \vdots \\ x_{n1} \end{bmatrix} \quad (9)$$

R is a set of data that has the number of reduced features which are similar to the number of hidden neurons of the ANN. In other words, the number of reduced features in this case depends on the number of hidden neurons selected in the ANN. After determining these reduced features, it is then used as input features to the ANN developed for assessing vulnerability of power systems.

The NNWE method uses a simple approach to find the optimal number of m-hidden neurons so as to reduce the n-dimension of the original training data sets by comparing the ANN results of multiple runs with different number of hidden neurons and choose the best one according to the accuracy of the ANN developed for vulnerability assessment. An advantage of using NNWE method is that training time of the developed ANN can be reduced significantly by using smaller number of input feature sets.

Normalization: Normalization is a transformation of each feature in the data set. This step is required to preprocess the training data sets by normalizing the inputs and outputs such that that they fall in the interval$^{[-1,1]}$. The following equation is used to normalize the input and output data.

$$x_{i(norm)} = \frac{2(x_i - x_{min})}{(x_{max} - x_{min})} - 1 \quad (10)$$

In the neural network implementation, the Exact RBFNN is used as the first ANN for feature extraction whereas the GRNN is used as the second ANN for vulnerability assessment of the test power system. By using the two types of RBFNN, better neural network performance can be obtained in terms of accuracy.

SIMULATION RESULTS

The development of RBFNN for vulnerability assessment of the test system is implemented using MATLAB version 7 on an Intel Pentium 2.13 GHz with 496 Mb of RAM. The RBFNN coded originally in MATLAB was utilized and modified to suit the vulnerability assessment data. Here, the width parameter was set to 0.8326/spread, resulting in radial basis functions that cross 0.5 at weighted inputs of +/- spread. The spread was experimentally adjusted to optimize model prediction performance.

To evaluate the effectiveness of RBFNN for vulnerability assessment, the results of RBFNN are compared with the results obtained from using the MLPNN. Both neural networks are implemented using similar training and testing data sets obtained from the 467 simulation cases in which a sample of 350 cases (75%) are selected randomly for training and 117 cases (25%) for testing. For the MLPNN implementation, two hidden layers are used since there is no systematic method in MLP of deciding the number of layers and the number of neurons in each layer. The optimum number of hidden neurons is chosen using a pruning strategy. The activation function for hidden and output neurons are linear function or sigmoid nonlinear either of logistic or hyperbolic type, depending on which combination achieves the most accurate result for a given input and output. Training of MLPNN using conjugate gradient algorithm and scaled conjugate gradient algorithm was found to be unacceptable and hence gradient descent with momentum was used.

There are 413 input features selected for each set of data comprising of real and reactive power flows and total generated real and reactive powers. By using the proposed feature extraction method, the extracted features are reduced to 23 and 43 which are 5 and 10 % percent of the original features, respectively.

The training results of RBFNN and MLPNN in terms of vulnerability indices PSL and PLL are as shown in Table 1 and 2, respectively. The results are evaluated in terms of average absolute error and

Table 1: PSL outputs of the RBFNN and MLPNN

Neural network model	Average absolute error (%)	Training time (sec)
Results considering 23 input features		
MLP	5.75	17680.875
RBF	4.8798	0.798
Results considering 43 input features		
MLP	5.714	18348.563
RBF	4.746	0.938

Table 2: PLL outputs of the RBFNN and MLPNN

Neural network model	Average absolute error (%)	Training time (sec)
Results considering 23 input features		
MLP	21.7	19239.4
RBF	6.1	0.735
Results considering 43 input features		
MLP	12.497	21788.3
RBF	6.041	0.891

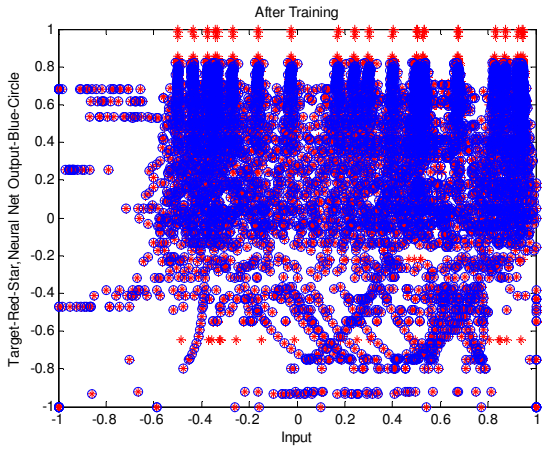

Fig. 4: Training errors of the RBFNN for PSL output

training time. The absolute error is the difference between the desired output and the ANN output. The training results in terms of absolute errors of the PSL and PLL outputs as in Table 1 and 2, respectively, show that the RBFNN is more accurate when considering reduced input features of 43. In this case, the absolute errors of PSL and PLL outputs are 4.746 and 6.041%, respectively. Comparing the training errors of RBFNN with MLPNN, it can be seen graphically as in Fig. 4 and 5 that there are less training errors with RBFNN than that of using MLPNN. However, it can be noted in Fig. 4 that for some cases, the inaccurate estimates of RBFNN in giving PSL outputs may be due to the existence of noncontiguous pockets in the input space.

Referring to Table 2, the training results in terms of absolute errors of the PLL outputs show that the RBFNN is more accurate compared to the MLPNN. In general, comparing the performance of the two neural

Table 3: Neural network testing results

Absolute error of PSL output		Absolute error of PLL output	
RBFNN	MLPNN	RBFNN	MLPNN
0.0045	0.0296	0.02	0.0818
0.0406	0.0429	0.0239	0.1028
0.0424	0.0234	0.0027	0.1225
0.0514	0.0942	0.0211	0.0758
0.0198	0.0261	0.018	0.0374
0.02	0.0277	0.0152	0.0526
0.0303	0.057	0.0194	0.054
0.0163	0.07	0	0.0651
0.0369	0.0834	0.0223	0.0869
0.0036	0.0455	0.0049	0.1245
0.0194	0.0571	0.0161	0.0702
0.0089	0.0593	0.0305	0.1165
0.0272	0.0413	0.0103	0.0743
0.0003	0.0263	0.0017	0.1519
0.025	0.0654	0.0126	0.0901
0.0242	0.0639	0.0196	0.0693
0.0173	0.0784	0.0047	0.0731
0.0211	0.0473	0.0291	0.0667
0.0517	0.0304	0.0237	0.0842
0.0451	0.0906	0.0227	0.0784
0.0432	0.0853	0.0202	0.0805
0.0063	0.0661	0.0191	0.0649
0.0026	0.0614	0.0572	0.0074
0.0064	0.0602	0.0089	0.0293
0.0185	0.0588	0.0213	0.0843
0.0323	0.0745	0.0226	0.0863
0.0389	0.0812	0.025	0.0821
0.0415	0.0872	0.0241	0.0849
0.0184	0.0579	0.0037	0.0775
0.0005	0.0477	0.0006	0.0604
0.0062	0.0271	0.0129	0.0696
0.039	0.08	0.0248	0.0835
0.0033	0.0176	0.0175	0.0607
0.002	0.0247	0.0073	0.1523
0.035	0.0916	0.0369	0.0031
0.0399	0.0794	0.0259	0.0858
0.0275	0.0777	0.0263	0.0913
0.0178	0.0301	0.0521	0.0614
0.0153	0.081	0.0371	0.5118
0.0076	0.1177	0.0164	0.5344
0.0078	0.0999	0.0147	0.5234
0.0096	0.0229	0.0361	0.2918
0.0066	0.0147	0.0067	0.2593
0.0145	0.0066	0.1742	0.2258
0.0382	0.001	0.3787	0.1918
0.0395	0.0796	0.0216	0.0798
0.0235	0.0643	0.0217	0.0856
0.0067	0.032	0.0217	0.1
0.0097	0.1151	0	0.0601
0.0063	0.0809	0	0.1828

network models, the RBFNN gives a better performance than the MLPNN in terms of training time and accuracy. The RBFNN with the proposed feature extraction method for vulnerability assessment decreases the training time drastically from hours to less than a few seconds.

Table 3 shows a summary of RBFNN and MLPNN testing results for the case with 43 input features. The

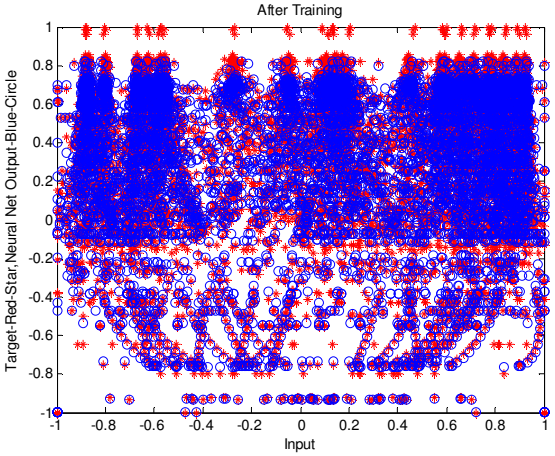

Fig. 5: Training errors of the MLPNN for PSL output

testing results are evaluated in terms of absolute errors. From the testing results, it is noted that in terms of absolute errors, for most cases the RBFNN is more accurate compared to the MLPNN.

CONCLUSION

A comparison study of two neural network models using RBFNN and MLPNN for vulnerability assessment on a large sized power system has been presented. Testing data set has deliberately been chosen outside the region of the training data set so as to test the generalization and extrapolation capability of the ANN models after learning. The use of RBFNN is a new ANN method used for vulnerability assessment of power systems. It has the advantage of taking less computational effort in training compared to the MLPNN due to its very low training time.

A new feature extraction method using the neural network weight extraction has proven to be an effective method in dimensionality reduction. In this application, by using the proposed neural network weight extraction method, the input features are reduced from 413 to 43 and 23. The training results prove that both the neural network performances are better when the input features selected are 43, which is about 10% of the original input features.

The training and testing results prove that the RBFNN perform better than the MLPNN in terms of accuracy and training times. Comparing the ANN results using PSL and PLL as vulnerability indices, it is noted that the PSL is more accurate in terms of evaluating its absolute errors.

REFERENCES

1. Song, H. and M. Kezunovic, 2005. Static Security Analysis based on Vulnerability index (VI) and Network Contribution Factor (NCF) Method. Proceeding of IEEE PES Asia-Pacific Transmission and Distribution Conference and Exposition, China, pp: 1 - 7.
2. Anjia, MAO., Y.U. Jiaxi and G.U.O. Zhizhong, 2006. Electric Power Grid Structural Vulnerability Assessment Proceedings of the IEEE, Power Engineering Society General Meeting <http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=11204>, pp: 1-6.
3. Jensen, C.A., Mohamed A. El-Sharkawi and R.J. Marks, 2001. Power System Security Assessment Using Neural Networks: Feature Selection Using Fisher Discrimination, Proceedings of the IEEE Transaction on Power System, pp: 757-763.
4. Ahmed M.A. Haidar, Azah Mohamed and Aini Hussain, 2007.Vulnerability Assessment of a Large Sized Power System Using a New Index Based on Power System Loss. Eur. J. Sci. Res., 17 (1): 61-72.
5. Sanyal, K.K., 2004. Transient Stability Assessment Using Artificial Neural Network Proceedings of the IEEE International Conference on Electric Utility Deregulation Restructuring and Power Technologies Hong Kong, pp: 633-637.
6. Tan, S.C. and C.P. Lim, 2004. Application of an Adaptive Neural Network with Symbolic Rule Extraction to Fault Detection and Diagnosis in a Power Generation Plant. Proceedings of the IEEE Transaction on Energy Conversion, pp: 369-377.
7. Tetsuya Hoya and Jonathon A. Chambers, 2001. Heuristic Pattern Correction Scheme Using Adaptively Trained Generalized Regression Neural Networks. IEEE Transactions on Neural Networks, pp: 91-100.
8. Dash, P.K., A.K. Pradhan and G. Pandaa, 2001. Application of Minimal Radial Basis Function Neural Network to Distance Protection. IEEE Transactions on Power Delivery, pp: 68-74.
9. Joorabian, M., S.M.A. Taleghani Asl and R.K. Aggarwal, 2004. Accurate Fault Locator For EHV Transmission Lines Based On Radial Basis Function Neural Networks. J. Electric Power Syst. Res., 71: 195-202.
10. Dash, P.K., S. Mishra and G. Panda, 2000. A Radial Basis Function Neural Network Controller for UPFC. IEEE Transactions on Power Systems, pp: 1293-1299.
11. Kim, M., M.A. El-Sharkawi and R.J. Marks, 2005. Vulnerability indices for power system. IEEE Intelligent Systems Application to Power Systems: Proceedings of the 13th International Conference, pp: 335-341.
12. Byungwhan Kim, Junki Bae and Byung Teak Lee, 2006. Modeling of silicon oxynitride etch microtrenching using genetic algorithm and neural Network. J. Microelectronic Eng., 83: 513-519.
13. Soliman, E.A., A.K. Abdelmageed and M.A. El Gamal, 2002. Neural Computation of the MoM Matrix Elements for Planar Configurations. Int. J. Electronics Commun. (AE¨U), 56 (3): 155-162.