Power System Security Margin Prediction Using Radial Basis Function Networks

TR #97-10

Guozhong Zhou, James D. McCalley and Vasant Honavar

June 27, 1997

ACM Computing Classification System Categories (1991):
I.2.6 [*Artificial Intelligence*] Learning — connectionism and neural nets

Keywords:
power systems, security margin
feature subset selection, genetic algorithms, radial basis function networks

Dr. McCalley's research is partially supported through grants from National Science Foundation and Pacific Gas and Electric Company. Dr. Honavar's research is partially supported through grants from National Science Foundation and the John Deere Foundation.

This paper will appear in: Proceedings of the 29th Annual North American Power Symposium, Oct. 13-14. 1997, Laramie, Wyoming.

Artificial Intelligence Research Group
Department of Computer Science
226 Atanasoff Hall
Iowa Sate University
Ames, Iowa 50011-1040, USA

Power System Security Margin Prediction Using Radial Basis Function Networks

Guozhong Zhou
gzhou@iastate.edu

James D. McCalley
jdm@iastate.edu

Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011

Vasant Honavar
honavar@cs.iastate.edu
Department of Computer Science
Iowa State University
Ames, Iowa 50011

Abstract

This paper presents a method to predict the postcontingency security margin using radial basis function networks with a fast training method. A genetic-based feature selection tool is developed to obtain the most predictive attributes for use in RBF networks. The proposed method is applied to a thermal overload problem for demonstration. The simulation results show that the proposed method gives satisfactory results and the running time decreases by a factor of 10 compared with using multilayer perceptrons.

1 Introduction

1.1 Power system security margin prediction

Security is a very important problem for power system operations. Power system states can be identified as secure or insecure. In addition, it is very useful for operators to know the margin, i. e., how far the state is from the security boundary. Knowledge of security margins is becoming more and more essential as transmission lines are operating closer and closer to their capacities in today's deregulated environment. The security margin prediction problem can be expressed as follows:

Given an operating condition characterized by a set of critical parameters chosen with respect to a particular contingency and postcontingency performance measure, find the "distance" between the operating condition and the boundary defined by a threshold on the postcontingency performance measure.

Fig. 1 shows the procedure for security margin prediction.

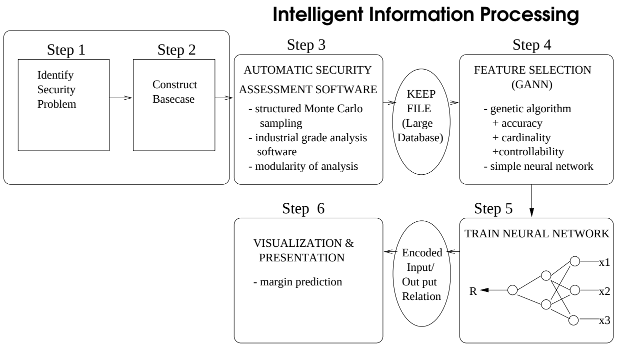

Figure 1: Procedure for security margin prediction

There are various security problems in power systems. We consider only thermal overload problem here. In this case, the postcontingency performance is the flow on the line at risk of overload and the threshold is the emergency rating of this line. Let $\underline{x}$ denote the critical parameter candidates, vector $\underline{x}^s$ denote the selected critical parameters, and R denote the performance measure, our goal is to obtain a relationship $R = f(\underline{x}^s)$ that predicts the contingency effects using knowledge of only precontingency conditions. Neural networks are used for this purpose because of their capability to handle nonlinear functions of many variables.

A sample system is shown in Fig. 2. The load in the area, during high loading conditions, is greater than the generation capacities in this area. Therefore, a significant amount of power must be imported into the area to meet the demand. There are several ties between this area and the remaining part of the system. Thermal overload occurs on tie line 5 when tie line 3 is outaged. So the postcontingency performance measure is the flow on tie line 5.

The emergency rating of tie line 5 is $I_0 = 600$A. Therefore, the threshold value for this constraint is $I_0 = 600$A, and when the performance measure is normalized according to

$$R = \frac{I - I_0}{I_0}$$

we have $R_0 = 0$ on the boundary, i. e., the margin is zero. Thus, the value of R represents the security margin. The more negative the margin, the more secure the system.

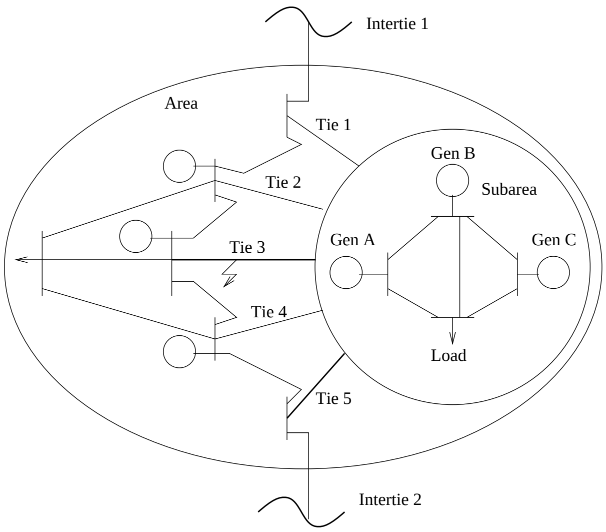

Figure 2: A sample system

1.2 Feature Selection for Security Margin Prediction

Security assessment studies have traditionally depended on engineering judgment to select the critical parameters, i.e., to select the parameters to be used in predicting the margin for each security problem. Our goal here is to develop an automatic approach to critical parameter selection, i.e., feature selection. We do not intend that automatic feature selection replace engineering judgment, but rather enrich it by confirming and extending physical understanding.

Feature selection is a search over the space of all combinations of features where a function is used to evaluate what is best. So two components of feature selection are evaluation function and search approach.

For the purpose of feature selection in power systems, operating parameters can be classified as *independent* or *dependent*. A parameter is independent if it is included in the input data to a power flow program; examples include MW injection or voltage magnitude at a generator bus or load level (MW or MVAR injection) at a load bus. A parameter is dependent if it is computed as a result of a power flow program solution; examples include bus voltages at a load bus or line flows. Independent critical parameters can be further divided according to operator controllability. MW injections and voltage magnitudes at generator buses are controllable independent parameters; load levels at load buses are noncontrollable independent parameters.

The problem for feature selection in power system security assessment can be described as follows:

Given a database having columns consisting of a number of features and a single performance measure, determine a subset of the total attribute space that can be used to train a neural network that predicts the postcontingency performance measure such that the following criteria are satisfied:

- *Set Sufficiency (accuracy):* The feature set must contain sufficient information to allow prediction of the postcontingency performance measure within a desired accuracy for all operating conditions within the study scope.
- *Set Cardinality:* The feature set should be chosen as the set of minimum size that satisfies the set sufficiency criterion.
- *Controllability Constraint:* At least one feature within the set must be controllable by the operator so that the operating point can be adjusted, with respect to the boundary, using preventive actions.

Researchers in the statistics and pattern recognition communities have investigated feature subset selection problems for decades [10]-[12]. Many of these methods are based only on the ability to predict the output. There are no efficient way to account for cardinality constraints on the feature set although there are several techniques that have attempted to find minimum feature subsets (see [16] for a discussion). Furthermore, there is no capability to preselect some parameters into the solution. Some of the models select the feature sets that satisfy the sufficiency condition, but no mechanism has been proposed to ensure that the chosen subset will satisfy the cardinality and the controllability conditions. To satisfy all three parts of the stated criteria, we have developed a genetic algorithm approach that searches only portions of the solution space of a specified cardinality level containing certain attributes. This localized search technique ensures that the cardinality and controllability criteria are satisfied. We used MLPs in our early work, but it took too long to obtain the results. Therefore, we turn to radial basis function networks to speed up the solution.

2 Radial basis function networks

Neural networks can be classified into supervised and unsupervised networks according to their learning strategies. The most widely used supervised neural network is the multilayer perceptron (MLP) trained with back-propagation algorithm. The design of this kind of neural network may be thought of as an application of an optimization method known in statistics as stochastic approximation. However, the design of a supervised neural network may be pursued in different ways. One important approach is to view this design problem as a curve fitting problem in a high-dimensional space. From this perspective, learning is equivalent to finding a surface in a multidimensional space that best fits the training data in the sense of statistics. Generalization is then equivalent to using this multidimensional surface to interpolate the test data. Such a viewpoint is indeed the motivation behind the method of radial basis functions (RBFs). Broomhead and Lowe [2] first explored the use of RBFs in neural networks. Moody and Darken [3], Renals and Rohwer [4], and Poggio and Girosi [5] among others made major contributions to the theory, design, and application of RBF networks.

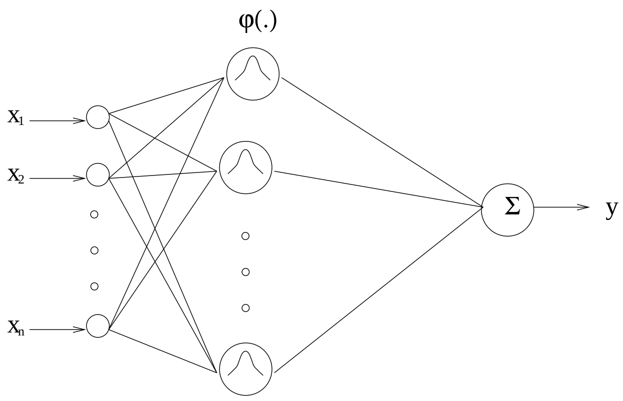

Figure 3: A radial basis function network

The basic architecture of a RBF network is shown in Fig. 3. It includes three entirely different layers. The first layer is an input layer of which each node corresponds to an attribute of an input pattern. The second layer is a hidden layer that serves a different purpose from that of the output layer. The third layer is an output layer responding to the input patterns. The transformation from the input layer to the hidden layer is nonlinear, whereas the transformation from the hidden layer to the output layer is linear. An activation function for a hidden layer node is a locally radially symmetric function (typically Gaussian function), whose output decays to zero as the distance (usually Euclidean distance) between the input vector and its center increases, see Fig. 4. There are two important parameters, center and width, associated with an activation function. A center is a vector with the same dimension as the input pattern, which represents a cluster center of the input space. A width is used to control the spread of the RBF so that its output decays more slowly or more rapidly as the distance between the input vector and the center increases. Each RBF $\varphi(\cdot)$ responds to a small convex region of the feature space. A large number of these functions cover the entire feature space so that the output layer neurons can join them with different weights to accomplish the function approximation or classification. Fig. 5 shows a portion of two-dimensional feature space covered by RBFs.

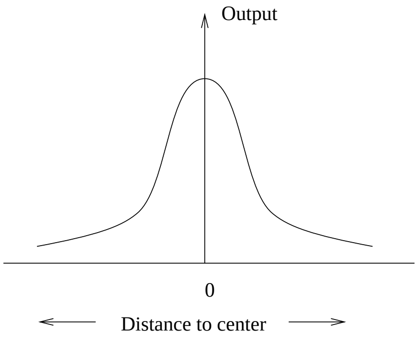

Figure 4: A radial basis function in one-dimensional space

Without loss of generality, we consider a RBF network with only one output. The output of such a network can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{M} w_i \varphi_i(\|\mathbf{x} - \mathbf{c}_i\|) \tag{1}$$

where $\mathbf{x}$ is an input pattern, $\mathbf{c}_i$ is the center for hidden node i, w_i is the weight between hidden node i and the output node, and w_0 is a bias weight. M is the number of hidden nodes, $\varphi(\cdot)$ is the activation function for the hidden layer. The norm $\|\cdot\|$ can be Euclidian or Mahalanobis distance when the densities

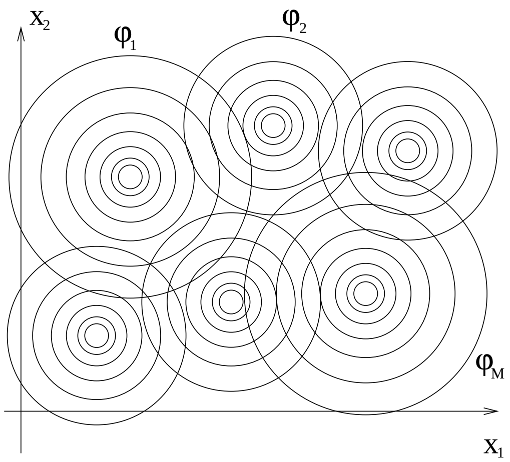

Figure 5: A portion of two-dimensional feature space covered by RBFs

have a common covariance matrix. We consider RBF approximation using Gaussian functions.

When $\varphi(\cdot)$ is a Gaussian function, (1) can be seen as approximating a probability density by a mixture of Gaussian functions. The Gaussian function can be expressed as

$$\varphi(u) = e^{-\frac{u}{2\sigma^2}} \tag{2}$$

The output can then be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{M} w_i e^{-\frac{\|\mathbf{x}-\mathbf{c}_i\|}{2\sigma^2}} \tag{3}$$

RBF networks provide an attractive approach for function approximation because of their flexible structure, fast training, powerful generalization capability, and conceptual elegance. Park and Sandberg [7] have shown that RBF networks with Gaussian basis functions are universal function approximators. Girosi and Poggio [8] have shown pointwise convergence property of a class of RBF networks.

3 Radial Basis Function Network Training Strategies

There are different training strategies for RBF networks [1]. Since the linear weights associated with the output node tend to evolve more slowly than the nonlinear activation functions of the hidden nodes, it is reasonable to train the network layer by layer to get fast training speed. The training strategies can be divided into the following three classes depending on how the centers of the RBFs are specified.

3.1 Fixed centers selected randomly

The simplest and fastest approach is to randomly select centers from the training data set and keep them constant throughout the training. This is reasonable provided that the training data are well representative of the problem [6]. The widths for all RBFs are also fixed and are the same. This width can be taken as the standard deviation of the Gaussian function, expressed as

$$\sigma = \frac{d}{\sqrt{2M}} \tag{4}$$

where d is the maximum distance between the selected centers. Such a choice for the standard deviation σ is to ensure that RBFs are neither too peaked to cover the whole input space nor too flat to distinguish between dissimilar input patterns.

Then, the only parameters that need to be trained are the weights between the hidden and output layer, which can be computed directly by solving linear equations. Usually the number of training patterns is much larger than the number of selected centers, so the resulting linear equations are overdetermined. A straightforward procedure for solving such equations is to use the pseudoinverse method [2] to obtain a solution with the minimum least square error.

The linear weights can also be solved by iteration using gradient-descent technique. This approach is much faster than the backpropagation (BP) algorithm for MLPs because it adjusts only the weights between the hidden and output layer. The generalization capability and accuracy level depends on the number of centers and the representativeness of the training data set to the problem that is being studied.

3.2 Unsupervised selection of centers

In principle, any unsupervised or clustering algorithm can be used to specify the centers. For example, we may use k-means clustering algorithm [3], hierarchical cluster analysis, or self-organizing map. After we determine the centers, we may obtain the linear weights either by directly solving the linear equations or by iteration.

3.3 Supervised selection of centers

This is the most flexible but most time-consuming training approach among the three strategies. The centers, widths, and linear weights are all adjusted through a supervised training process. A natural candidate for such a process is error correction iteration algorithm using gradient-descent technique.

The squared error for the output can be written as

$$E = \frac{1}{2}\sum_{k=1}^{N}(d_k - y_k) \tag{5}$$

The update equations for the linear weights, centers, and widths are given as follows [1]

$$w_i(t+1) = w_i(t) + \eta_1 \sum_{k=1}^{N}(d_k - y_k)e^{-\frac{\|\mathbf{x}_k - \mathbf{c}_i\|^2}{2\sigma_i^2}} \tag{6}$$

$$\begin{aligned} c_{ij}(t+1) &= c_{ij}(t) + \frac{\eta_2}{\sigma_i^2}\sum_{k=1}^{N}(d_k - y_k)w_i(t)e^{-\frac{\|\mathbf{x}_j - \mathbf{c}_i\|^2}{2\sigma_i^2}} \\ &\quad \cdot(x_{kj} - c_{ij}(t)) \end{aligned} \tag{7}$$

$$\begin{aligned} \sigma_i^2(t+1) &= \sigma_i^2(t) + \eta_3 \sum_{k=1}^{N}(d_k - y_k)w_i z_{ik} \\ &\quad \cdot\frac{\|\mathbf{x}_k - \mathbf{c}_i\|^2}{2\sigma_i^4(t)} \end{aligned} \tag{8}$$

4 Data Generation

Quality of the data is a key factor that affects the performance of a neural network. If the data do not accurately reflect the actual behaviors of the power system operations, we can not expect the neural network to predict the margin correctly. Automatic security assessment software (ASAS) [17, 18] is used to generate the data for the use of the RBF network. The resulting data set consists of 1005 patterns, with each pattern corresponding to a simulation of tie line 3 outage (see Fig. 2) under various operating conditions. So each pattern contains 32 precontingency parameters (feature candidates) and the corresponding postcontingency performance (margin for tie line 5).

5 Design of Genetic-Based Radial Basis Function Network Feature Selection Tool

5.1 Overall structure

Suppose the number of candidate attributes is n, the solution space will consist of 2^n possible solutions. For our problem, the solution space is very large so that an exhaustive search is not practical. The genetic algorithm (GA) is very effective in solving combinatorial problems for which a performance measure, i.e., an evaluation or "fitness" function, can be defined for each combination under consideration. We therefore turn to GA for an efficient search. In our implementation, the user specifies parameters that should be forced into the solution to satisfy the controllability requirement, and the user also specifies a cardinality range. The GA then identifies the most accurate solution containing the forced parameters for each cardinality level in the specified range. It is then an easy matter for the user to select the solution that offers the best tradeoff between accuracy and cardinality. Since our goal is to obtain a neural network that can accurately predict postcontingency performance given precontingency information, the evaluation function includes the accuracy of a neural network trained with data that correspond to the parameters composing the particular subset. The evaluation function also includes a component that depends on cardinality. The weight for this component is much higher if the cardinality of a subset is equal to or smaller than the desired cardinality. On the other hand, the cardinality component is weighted much higher than the accuracy component and therefore dominates the early generations of the evolution so that members in later generations are all of the desired cardinality and evolution proceeds to maximize accuracy.

Fig. 6 illustrates the basic functional structure of the developed software.

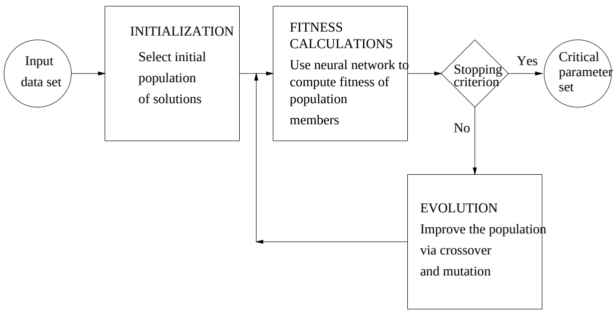

Figure 6: Basic functional structure of GANN

The GA follows the following steps

1. *Population initialization:* Generate the initial population consisting of M solutions randomly.

2. Compute the fitness of each individual solution in the current population using RBF networks.

3. *Elitist strategy implementation:* Select best N solution in terms of fitness values to be directly placed into the next generation.

4. *Evolution (crossover and mutation):* We do this in two steps. First select $M - N$ solutions using roulette wheel selection to be placed into a mating pool. Then, apply the following procedure to put the resulting solutions to the new generation.

   (a) Pick a pair of parents with probability proportional to fitness.

   (b) Perform crossover to create offspring and do mutation for each new solution.

5. *Stopping criteria check:* We use a given maximum number of generations as the stopping criterion. If the stopping criterion is satisfied, stop. Otherwise, go to Step 2.

5.2 Implementation details

The following gives the parameter settings for the genetic algorithm and the neural network in implementation.

- Genetic algorithm
  - Population size for each generation: 50
  - Total number of generations: 50
  - Probability of crossover: 0.9
  - Probability of mutation: 0.01
- Neural Network
  - Network type: RBF network
  - Number of centers: 80
  - Training method: Using fixed centers and directly solving for weights

6 Feature Selection Result

We ran GANN for 5 different cardinality levels, from 5 to 10, with parameters 1-4 forced (preselected) into the solution, meaning that all solutions provided by GANN will have GenA, AreaLoad, GenB, and GenC, in addition to some other parameters.

The results of GANN runs are given in Table 1. The level 8 cardinality was chosen because it represented the best tradeoff between cardinality and accuracy.

The results generally agree with engineering judgment. For example, parameters 9 and 18 are chosen the most often; parameter 9 represents the flow on the overloaded circuit and parameter 18 is a bus voltage close to the outaged circuit.

Table 1: GANN output for thermal overload on tie line 5 for cardinality levels 5-10

Parameter		Cardinality Level					
No.	Name	5	6	7	8	9	10
1	GenA	1	1	1	1	1	1
2	AreaLoad	1	1	1	1	1	1
3	GenB	1	1	1	1	1	1
4	GenC	1	1	1	1	1	1
5	TieFlow1	0	0	0	0	0	0
6	TieFlow2	0	0	0	0	0	0
7	TieFlow3	0	0	0	0	0	0
8	TieFlow4	0	0	0	0	0	0
9	L-NFlow	0	1	0	1	1	0
10	P-HFlow	0	0	0	0	0	0
11	H-MFlow	0	0	0	0	0	0
12	P-MFlow	0	0	0	0	0	0
13	HNTrFlow	0	0	1	0	1	0
14	KNTrFlow	0	0	0	0	0	0
15	A-GFlow	0	0	0	0	0	0
16	HTTrFlow	0	0	0	0	0	1
17	W-GFlow	1	0	1	0	0	1
18	G230kV	0	1	1	1	0	1
19	A230kV	0	0	0	0	1	0
20	W230kV	0	0	0	1	0	0
21	P230kV	0	0	0	0	0	0
22	KN230kV	0	0	0	0	0	0
23	HN230kV	0	0	0	0	0	1
24	GG230kV	0	0	0	0	1	0
25	HT230kV	0	0	0	0	0	1
26	H230kV	0	0	0	0	0	0
27	M230kV	0	0	0	0	0	1
28	G500kV	0	0	0	0	0	0
29	M500kV	0	0	0	0	0	0
30	SL70kV	0	0	0	0	0	0
31	SG115kV	0	0	0	0	1	0
32	G-HT2	0	0	0	1	0	0
Average Error		.0222	.0161	.0095	.0087	.0083	.0061

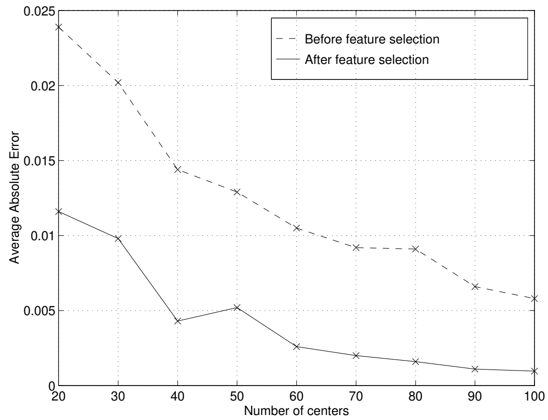

Figure 7: Test errors versus number of centers (random selection of centers)

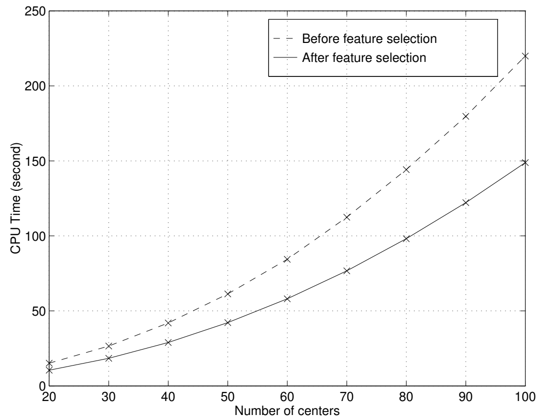

Figure 8: CPU time versus number of centers (random selection of centers)

For comparison, the test results with 8 selected attributes and with all 32 attributes are shown in Fig. 7. We can see that the test error is much smaller using selected attributes. Furthermore, as Fig. 8 shows, training with fewer selected attributes takes shorter time than with all 32 attributes although CPU time is increasing with the number of centers in both cases.

7 Network Training and Test

The data set was split into two parts according to a given ratio 6:4 for training and test, respectively. Centers were randomly selected from the training data set and fixed. The number of centers varied from 50 to 100. Widths were set to the same values for all centers. Weights between the hidden and output layer were computed directly using pseudoinverse technique.

Fig. 9 shows the training and test errors versus number of centers that were randomly chosen from the training data, using features selected by backpropagation neural networks in our previous work and by radial basis networks, respectively.

We can observe that both training and test error decrease as the number of centers increases. This is because more RBFs make more accurate interpolation of the data. When the number is 80 (which is used in feature selection), the average absolute test error is 0.0015 when using features selected by radial basis function networks, and 0.0017 when using features selected by backpropagation neural networks in our previous work. They are very close. Thus, for the purpose of training and test, we may use RBF networks with fewer centers in feature selection and use RBF networks with more centers or another type networks with higher accuracy for training, test, and use in prediction.

Our experience also shows that GANN with RBFNs using the learning algorithm of fixed centers runs faster than with MLPs using the BP learning algorithm, given the same accuracy. For example, at cardinality level of 8, it takes almost 10 times faster for the former than the latter (8903 seconds with RBFNs versus 88886 seconds with MLPs).

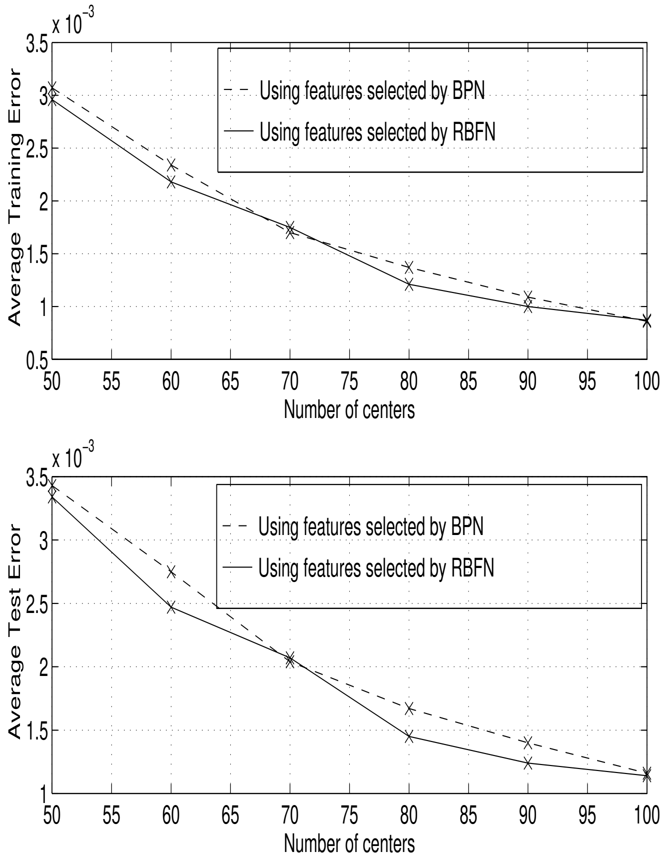

Figure 9: Training and test errors versus number of centers (random selection of centers)

8 Conclusions

Postcontingency security margin prediction can be solved using RBF networks with fast training strategy. A genetic-based feature selection approach using RBF networks is presented that chooses critical parameters from the data set generated by simulation. The feature selection provides the accuracy in building neural networks from the feature set for security margin prediction. On the other hand, the selected feature subset also satisfies controllability constraints.

To further increase the speed, we can use fewer centers for RBF networks during feature selection period. After the best features are selected, we can use an RBF network with more centers to obtain higher accuracy for margin prediction.

Acknowledgments

The authors would like to acknowledge helpful discussions from Mr. Khor Chin.

References

[1] S. Haykin, *Neural Networks*, New York: Macmillan College Publishing Company, 1994.

[2] D. S. Broomhead, D. Lowe, “Multivariable functional interpolation and adaptive networks,” *Complex Systems*, vol. 2, 1988, pp. 321-355.

[3] J. Moody, C. J. Darken, “Fast learning in networks of locally tuned processing units,” *Neural Computation*, vol. 1, 1989, pp. 281-294.

[4] S. Renals, “Radial basis function network for speech pattern classification,” *Electronics Letters*, vol. 25, 1989, pp. 437-439.

[5] T. Poggio and F. Girosi, “Networks for approximation and learning,” *Proceedings of the IEEE*, vol. 78, 1990, pp. 1481-1497.

[6] D. Lowe, “Adaptive radial basis function nonlinearities, and the problem of generalization,” *1st IEE International Conference on Artificial Neural Networks*, London, UK, 1989, pp. 171-175.

[7] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, 1991, pp. 246-257.

[8] F. Girosi and T. Poggio, "Networks and the best approximation property," *Biological Cybernetics*, vol. 63, 1990, pp. 169-176.

[9] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, 1992, pp. 595-603.

[10] N. R. Draper and H. Smith, *Applied Regression Analysis*, 2nd ed., New York: John Wiley & Sons, 1981.

[11] A. J. Miller, *Subset Selection in Regression*, New York: Chapman and Hall, 1990.

[12] J. Neter, W. Wasserman, and M. H. Kutner, *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Designs*, 3rd ed., Homewood, IL: Irwin, 1990.

[13] L. Chambers, ed., *Practical Handbook of Genetic Algorithms: Applications*, Volume I, Boca Raton, FL: CRC Press, 1995.

[14] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Englewood Cliffs, NJ: Prentice-Hall, 1982.

[15] H. Vafaie and K. D. Jong, "Genetic algorithms as a tool for feature selection in machine learning," *Fourth International Conference on Tools with Artificial Intelligence*, Arlington, VA, November 1992, pp. 200-203.

[16] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," to appear in *Proceedings of International Conference on Genetic Programming*, Stanford, CA, July 1997.

[17] V. Van Acker, S. Wang, J. D. McCalley, G. Zhou, and M. Mitchell, "Data generation using automated security assessment for neural network training," to appear in *Proceedings of the 29th North American Power Symposium*, Laramie, WY, October 1997.

[18] J. D. McCalley, S. Wang, Q. Zhao, G. Zhou, R. T. Treinen, and A. Papalexopoulos, "Security boundary visualization for system operation

using automated assessment and neural networks," *IEEE Transactions on Power Systems*, vol. 12, no. 2, May 1997, pp. 940-947.

Biographies

Guozhong Zhou received the B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, P. R. China, in 1985 and 1988, respectively. He worked as a teacher and researcher at Tianjin University from 1988 to 1992, as an engineer at Northeast China Electric Power Company from 1992 to 1994, and as a visiting researcher at the University of Porto in Portugal from 1994 to 1995. He is currently working towards his Ph.D. degree at Iowa State University.

James D. McCalley is Assistant Professor in Electrical and Computer Engineering at Iowa State University, where he has been employed since 1992. He worked for Pacific Gas and Electric Company from 1986 to 1990. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, in 1982, 1986, and 1992, respectively. Dr. McCalley is a registered professional engineer in California and a member of the IEEE.

Vasant Honavar is Associate Professor of Computer Science and Neuroscience at Iowa State University where he started working as an Assistant Professor in 1990. He received a B.E. in Electronics Engineering from Bangalore University (India) in 1982, an M.S. in Electrical and Computer Engineering from Drexel University in 1984, an M.S. in Computer Science and a Ph.D. in Computer Science and Cognitive Science from the University of Wisconsin-Madison in 1989 and 1990 respectively. Dr. Honavar is a member of IEEE, ACM, and AAAI. He has co-edited one book, and published over 70 research articles in refereed journals, books, and conferences. His current research interests include machine learning, neural networks, evolutionary computing, intelligent agent architectures, neural and cognitive modeling, data mining and knowledge discovery, and applied artificial intelligence.