Extended Radial Basis Function (ERBF) Networks – Linear Extension and Connections

K. Mike Tao
Integrated Systems, Inc.
3260 Jay Street, Santa Clara, CA 95054
e-mail: mtao@isi.com

ABSTRACT

The increasingly popular Radial Basis Function (RBF) networks are *smoothed piecewise-constant* **universal** approximators. The (smoothed) piecewise-constant property, however, limits their effectiveness in extrapolations and in "trend" learning.

This paper *extends* the RBF network model, in a natural manner, to be *smoothed piecewise-linear* approximators, referred to as the Extended Radial Basis Function (ERBF) networks. This extension is significant in (at least) the following respects: 1) it can function as a *global* nonlinear model to *smoothly* link together the various *local* linear models; 2) it extends the RBFs ability to extrapolate and generalize more meaningfully; 3) it serves as a unifying model that brings together the various approximators including splines and CMAC neural network models, and 4) this ERBF extension, makes possible the applications of *statistical modeling* and *experiment design* techniques to the study of general neural network approximation models. Simulations results of learning various response surfaces are included for discussion and comparison.

1 Introduction

The traditional RBF networks are good approximators to arbitrary continuous (nonlinear) functions on a compact set [1]. However, to approximate (or learn) a nonlinear function well, it may require a large number of training samples and RBF memory units. In [2], it was shown that, with a *normalization* architecture, the RBF networks can be made to interpolate much better and extrapolate more sensibly. With this improved generalization ability, the normalized RBF network model also requires less training samples and memory units. Another advantage of the normalization scheme is that it has a nice *fuzzy system* interpretation [2, 3]. Therefore, in this paper, we will *only* consider the *normalization* architecture. For the convenience of discussion, the traditional RBF network *with normalization* will be referred to as the nRBF model (Figure 1). In [2], however, it was also shown that the nRBF model is a smoothed version of a *piecewise-constant* approximator. This piecewise-constant property limits the nRBF's ability to extrapolate and to perform "trend" learning. For example, with adequate training samples the nRBF model can learn *a segment* of a slanted line very well, but it cannot "correctly" extrapolate (generalize) the slanted line beyond this segment. Its extrapolations are more or less "horizontal," due to the nearest-neighbor piecewise-constant property. If it is possible to *extend* the nRBF model to become *piecewise-linear*, then its capacity and usefulness can be greatly enhanced. This paper concerns itself with such an extension leading to the Extended Radial Basis Function (ERBF) model.

In the above example, not only can the ERBF model learn the "trend" and extrapolate more "correctly," but also is the required number of training samples reduced (e.g., the ERBF network needs only two points to learn a line). Since, in many applications, linear models are often successfully used *locally*, the ERBF model could serve as a *global* nonlinear model which can smoothly link together a number of piecewise-linear local models. For example, in electronic device modeling, it was correctly identified in [4] that the traditional RBF model is "incomplete" in that it lacks an *affine* component, whereas the canonical piecewise-linear forms are well accepted in such applications. To compensate for this lacking, [4] adds a *global* affine component to the traditional RBF model and refers to it as Affine Plus Radial Basis Function (ARBF) model. However, this *global* addition is *ad hoc* in nature and does not in general perform better.

In this paper, we describe an ERBF model built with truly piecewise-linear components. Just as in the RBF networks, the ERBF network utilizes a set of RBF kernels (e.g., Gaussian-shaped memory units or *hidden neurons*). Each RBF kernel in the ERBF network corresponds to a *cluster* or a *fuzzy membership*. In the ERBF model, however, a locally linear model (LLM) is associated with each RBF kernel (or cluster). The most appropriate local model, for a given input, should have the strongest influence in the integrated final output. The ERBF network accomplishes this via a weighted *fuzzy* combination of the various LLM outputs.

The rest of the paper is organized as follows. In Section 2, the ERBF architecture is described, and then compared and harmonized with the nRBF architecture. A constructive procedure is described for building an ERBF network. Simulation results of *response surface* learning are presented and compared with that of a nRBF network. The remainder of the paper discusses some important connections between the ERBF model and other nonlinear ap-

1058-6393/95 $4.00 © 1995 IEEE

proximators and with the statistical modeling techniques. Section 3 shows some unifying *spline* connections. Section 4 makes an important connection with the established *statistical modeling* and *experimental design* methodology. It is shown that because of the piecewise-linear extension, the ERBF network is now rich enough in structure to allow the use of systematic model building procedures established in multivariate statistics. Practical issues such as model structure determination, input variable selections can be addressed using the established statistical procedures. This is an important step toward making neural netwoks more practically useful and theoretically sound.

2 The ERBF Network

In this section, we describe the ERBF network architecture; outline a constructive method for determining the network parameters, and present some *response surface* learning results.

Figure 2 depicts the ERBF architecture. Notice that, as discussed before, we only consider *normalized* architectures (whether RBF or ERBF) in this paper. As in the nRBF network, each RBF kernel (Gaussian-shaped memory unit [1] , also referred to as *nodes*) in the ERBF network corresponds to a *cluster* or *fuzzy membership* [2, 3]. In the ERBF model, however, a (locally) linear model (LLM) is associated with each kernel (or cluster). [2] The outputs of these LLMs are *modulated* by their corresponding fuzzy memberships before being combined to produce the final output. Mathematically,

$$\hat{y}(x) = \sum_{j=1}^{M} w_j(\bar{\phi}_j(x)\hat{y}^j(x)) \tag{1}$$

where $\hat{y}(x)$ is the ERBF network output given the network input (pattern) vector, x; M is the number of clusters (member classes); $\bar{\phi}_j(x)$ is the degree to which the input pattern x belongs to class j, and w_j's are the *weights* for output combination. $\hat{y}^j(x)$ is the output of the j-th LLM in which a bias term, b_j, is implicitly included, i.e.,

$$\begin{aligned} \hat{y}^j(x) &= a_j'x + b_j && (2)\\ &= \begin{bmatrix} a_j' & b_j \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} && (3)\\ &= \theta^{j\prime}\bar{x} && (4)\\ &= \bar{x}'\theta^j && (5) \end{aligned}$$

where $\bar{x} = [x, 1]'$ is the augmented pattern vector x, and θ^j is the parameter vector of the j-th LLM. Notice that $\bar{\phi}$'s are the *normalized* (E)RBF kernel outputs, i.e.,

$$\phi_j(x) = exp(-\frac{||x - c_j||^2}{\sigma_j^2}) \tag{6}$$

$$\bar{\phi}_j(x) = \phi_j(x)/s(x) \tag{7}$$

$$s(x) \equiv \sum_{j=1}^{M} \phi_j(x) \tag{8}$$

where ϕ's are the (unnormalized) (E)RBF Gaussian kernel outputs, [3] and $s(x)$ is the *strength* of the RBF kernel responses to the input pattern x which can be used as a (relative) measure of the network's *confidence* in handling this particular input, x. This confidence measure is one of the major strengths of the RBF family of networks in practical applications.

One might notice that because of the affine structure in these local models, the ERBF network would have considerably more parameters to specify and learn than does an nRBF network. Does this necessarily mean that considerably more training samples would be required in order to build an ERBF network? Fortunately, the answer is no. The *key* is to use *proximity weighted* least squares to build each local linear model. Nearby training samples are weighted more heavily than distant samples. The RBF kernels' outputs can naturally serve as the proximity weights. This way, the same set of data can generate multiple LLMs (with *many* parameters). Modulated by the RBF kernel memberships, $\bar{\phi}$s, outputs of these LLMs are combined into a single integrated output (Eq. (1)) via proper weighting, $\{w_j\}$, which can be learned with the same least-squares type of procedure used in training the nRBF network.

A mathematical description of this constructive procedure is in order. Given a set of input-output training pairs: $\{x_i, y_i\}$, $i = 1, 2, ...N$, the first step is to select a set of M (E)RBF kernels ($M \leq N$). For the time being, let us use the self-organizing K-means clustering algorithm [2] to determine the location and size of these M kernels based on the distribution of $\{x_i\}$. Then, the next step is to build an LLM for each of these RBF kernels using weighted least-squares (wls). For example, for the j-th cluster, the LLM produces the following outputs:

$$\hat{Y}^j = \begin{bmatrix} \hat{y}_1^j \\ . \\ . \\ . \\ \hat{y}_N^j \end{bmatrix} \tag{9}$$

$$= \begin{bmatrix} \bar{x}_1' \\ . \\ . \\ . \\ \bar{x}_N' \end{bmatrix} \theta^j \tag{10}$$

$$= \bar{X}'\theta^j \tag{11}$$

Denoting the difference between $\hat{Y}^j$ and the training data $Y = [y_1, ..., y_N]'$ as $\tilde{Y}^j$, the weighted least-squares formulation seeks to minimize $\tilde{Y}^{j\prime}W^j\tilde{Y}^j$ where W^j is a diagonal

[1] Although non-Gaussian RBF kernels [2] are viable alternatives, we concentrate on Gaussian kernels in this paper.

[2] It is noted that if these LLMs are replaced with locally *constant* models, then the ERBF network degenerates to the nRBF network. Hence, the nRBF network can be viewed as a *special case* of the ERBF network.

[3] c_j is the j-th Gaussian RBF kernel's *center* and σ_j the *spread*. In this paper, we use a standardized σ for each kernel, referred to as the *basic* σ, which is the average distance from the center of the kernel to the centers of its two nearest neighbors.

weighting matrix with elements corresponding to memberships of the input patterns, $\{x_1, ..., x_N\}$, with respect to this j-th RBF kernel, i.e., $W^j = diag(\phi_j(x_1), ..., \phi_j(x_N))$. The wls solution (for the j-th LLM) is given as:

$$\hat{\theta}^j = (\bar{X} W^j \bar{X}')^{-1} \bar{X} W^j Y \qquad (12)$$

With these identified $\hat{\theta}^j$s, one can compute the LLM outputs, $\hat{y}_i^j$. Modulating these $\hat{y}^j$s with the fuzzy memberships, ϕ_j, the parenthesized terms on the right hand side of Eq. (1) are determined. This makes the determination of the output combining weights, w_j, a linear least-squares problem.

Some ERBF response surface learning results are compared with those obtained using the nRBF network. Figure 3 shows an example where the original surface is sampled at 16 points evenly on a rectangular grid system, and is given to an nRBF network and an ERBF network to learn. In this example, 16 nodes are used by each network and therefore both can learn these 16 points perfectly. In the right column of Figure 3, the *interpolation* results of these two networks are displayed for comparison. Apparently, for this example, they are quite comparable. What about extrapolation? Figure 4 displays the original surface on an extrapolated scale (upper left), along with the extrapolation result of the nRBF network (lower left), that of the ERBF network (upper right) and another ERBF extrapolation result (lower right). First, notice the *piecewise constant* property of the nRBF extrapolation which does not make much sense for this example. The *piecewise linear* property of the ERBF network allows it to capture of "trend" and makes the extrapolation much more meaningful. The "upper right" result is obtained by using the *basic* σ, whereas the "lower right" result is obtained using twice the basic σ. [4]

Figure 5 shows another example of a (hypothetical) response surface learning. The original surface (with 41 x 41 points) is evenly sampled for 11 x 11 points to form a training set for the nRBF and the ERBF networks. With 11 x 11 nodes (kernels), these networks can certainly learn the training set perfectly, but they can also *interpolate* equally well as Figure 5 indicates. However, their interpolations differ when the training set consists of *randomly* (unevenly) sampled points. Figure 6 shows randomly sampled 121 training data given to the nRBF and the ERBF networks. The interpolation results indicate that the ERBF has produced *smoother* interpolation than the nRBF network's.

Another example shown in Figure 7 also supports the above observation. This is a very difficult surface to learn, if one intends to use only a relatively small number of nodes (i.e., RBF kernels), as the peak of this surface is quite narrow. We use the K-means clustering to determine 30 nodes (kernels) for the nRBF and for the ERBF networks which are then trained with the 41 x 41 points on the original surface. The training results are displayed in Figure 7. The particular random seeding has resulted in "side-lobes" in both learned surfaces. However, the ERBF learning result is smoother and with smaller side-lobe. The smoother interpolation results of the ERBF network may be attributed to its *piecewise linear* property.

With a different random seeding, one could obtain better learning results with 30 nodes. However, as shown in [2, 5], a more systematic node selection procedure which can result in superior performance is via the Orthogonal Least Squares (OLS), a Gram-Schmidt based robust model building procedure also used in statistical multiple regression [10]. In that context, it is known as the *forward selection procedure* for selecting regression variables to be included in the model. Originally applied to the RBF network in [5], this OLS procedure can be extended to the ERBF node selection in a straight forward manner.

3 The Spline Connections

This section discusses some important connections between the ERBF model and other nonlinear approximators and interpolators. First of all, the (piecewise-constant) nRBF model can be viewed as a *fuzzy* generalization of the cardinal spline [6]. Generalizations are made to the handling of multi-dimension and scattered sample data. Likewise, the (piecewise-linear) ERBF network can be viewed as a *fuzzy* generalization of the linear spline [6]. These explicit spline connections also brings the B-spline related cerebellar model articulation controller (CMAC) neural network models [7] into the family. The relationships between the ERBF model, the very recently reported Locally Weighted Regression (LWR) model [8] and the work done in piecewise-linear fuzzy logic control [9] are also noted here. In the latter context, the ERBF model can be further generalized to employ locally linear *dynamic* models (LLDMs), such as ARx or ARMAx models. A common theme found in these various models is to use the power of parametric modeling *locally*, and then pieces these local models together to form a *global* model. It is in that sense, the ERBF architecture provides a unifying view.

4 Statistical Modeling and Experiment Design

Finally, a very important connection with the established *statistical modeling* [10] and *experiment design* [11] methodology is considered. Though restrictive in certain ways, the statistical modeling and regression methodology is very systematic within the assumed setting. For example, in multiple regression, there are statistical procedures to determine, based on a given set of data, which regression variables should be included in the model (and which ones should not). For example, by including mutually correlated variables in a multiple regression model, one may end up with a model which is not good for projection and analysis. This is known as the problem of *multi-collinearity* [10]. These are very important practical issues which are often encountered when dealing with data obtained *not*

[4] Experiences indicate that 2 basic σ seems to work the best in the examples tested.

through controlled experiments, such as operational data of a chemical process or (randomly) collected social economic data. With the traditional RBF or nRBF model, however, some of these statistical concepts do not seem to relate naturally, because piecewise constancy amounts to nothing but locally "averaging" the data. Therefore, using the nRBF model, it is not clear how to determine which variables should be included in the model. [5] Now, with the piecewise linearly structured ERBF model, the door is open for these statistical procedures to be applied. For this regression variable selection problem, the above mentioned OLS-like *forward selection procedure* can now be used to select regression variables for the LLMs in addition to being useful for hidden nodes selection as discussed in Section 2. There is an added flexibility in that variable selections can vary on a *local* basis while maintaining model integrity globally.

On the other hand, when *planned* data collection is possible, the question of how to obtain the most, and the least ambiguous, information possible at the least cost is of great practical importance in applications such as engineering design and process optimization. This is what the science of *experiment design* [11] is about. Some of the statistical experiment design techniques, such as designs for the general study of the importance of and interactions between certain variables, do not depend explicitly on any particular "parametric" model form, and are therefore generally applicable whether *neural network* (NN) is used in the study or not. There are, however, other designs, noteably the so-called *response surface designs* [12], which are based on the assumption of specific *model forms*, e.g., linear or quadratic surface models. Such designs do not seem to have theoretical justifications when NN is used for response surface modeling. Though recognized as an *open issue* in NN research [13], the current practice seems to be using the established experiment design methods such as the central composite design [12, 13] [6] in conjunction with NN modeling. Again, the *piecewise linear* property of the ERBF network seems to provide justification for potential applications of some of the established designs, such as the linear response surface designs.

5 Summary

The Extended Radial Basis Function (ERBF) network is shown to possess the desirable (smoothed) piecewise-linear property. Consequently, it can interpolate more smoothly and extrapolate more meaningfully. In many cases, the required number of training samples may also be reduced. This ERBF architecture provides a universal framework to *smoothly* link together many *locally* linear models, and thereby forming a *global* nonlinear model. Such a global linking is possible for both static and dynamic applications. A constructive procedure is described for effective determination of the ERBF parameters. Unifying connections with various kinds of splines and nonlinear approximators are discussed. Important connections with *statistical modeling* and *experiment design* are pointed out. With the piecewise linear property, the door is open for these statistical procedures to be applied. This is a significant step toward making neural networks more practically useful and theoretically sound.

[5] It should be noted that one might generalize the concept of the *forward selection procedure* to nRBF network as a whole by incrementally selecting one highest payoff variable at a time. The computation is more involved and the selections are global, unlike what can be achieved using the ERBF network where the selections can be made on a local basis.

[6] In some cases, other *ad hoc* "designs" or no design at all !

References

[1] J. Park and I.W. Sandberg, "Universal Approximation Using Radial Basis Function Networks," *Neural Computation*, Vol. 3, Summer 1991, pp. 246 - 257.

[2] K.M. Tao, "A Closer Look at the Radial Basis Function (RBF) Networks," *Record, 27th Asilomar Conf. Signals, Syst., Computers*, Nov. 1993, Pacific Grove, CA, pp. 401 - 405.

[3] L.-X. Wang and J.M. Mendel, "Fuzzy Basis Functions, Universal Approximations, and Orthogonal Least-Squares Learning," *IEEE Trans. Neural Networks*, Vol. 3, No. 5, Sept. 1992, pp. 807 - 814.

[4] A.I. Mees, M.F. Jackson and L.O. Chua, "Device Modeling by Radial Basis Functions," *IEEE Trans. Circuits and Systems, Part I*, Vol. 39, No. 1, January 1992, pp. 19 - 27.

[5] S. Chen, S.F.N. Cowan and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Trans. Neural Networks*, Vol. 2, No. 2, March 1991, pp. 302 - 309.

[6] H. Cohen, *Mathematics for Scientists and Engineers*, Englewood Cliffs, NJ: Prentice-Hall, 1992.

[7] S.H. Lane, D.A. Handelman and J.J. Gelfand, "Theory and Development of Higher-Order CMAC Neural Networks," *IEEE Control Systems Magazine*, April 1992, pp. 23 - 30.

[8] S. Schaal and C.G. Atkeson, "Robot Joggling: Implementation of Memory-Based Learning," *IEEE Control Systems Magazine*, Vol 14, No. 1, February 1994, pp. 57 - 71.

[9] J.-S. R. Jang, "Self-Learning Fuzzy Controllers Based on Temporal Back Propagation," *IEEE Trans. Neural Networks*, Vol. 3, No. 5, Sept. 1992, pp. 714 - 723.

[10] M.S. Younger, *A Handbook for Linear Regression*, North Scituate, MA: Duxbury Press, 1979.

[11] T.B. Barker, *Quality by Experimental Design*, NY: Marcel Dekker, 1985.

[12] A.I. Khuri and J.A. Cornell, *Response Surfaces: Design and Analyses*, NY: Marcel Dekker, 1987.

[13] G.S. May, "Manufacturing ICs the Neural Way," *IEEE Spectrum*, September 1994, pp. 47 - 51.

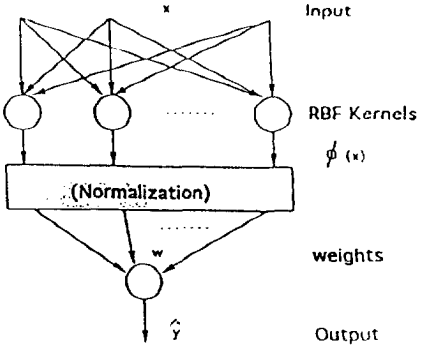

Figure 1: The nRBF Network

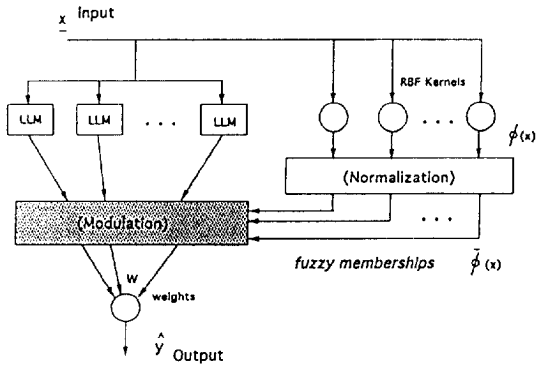

Figure 2: The ERBF Architecture

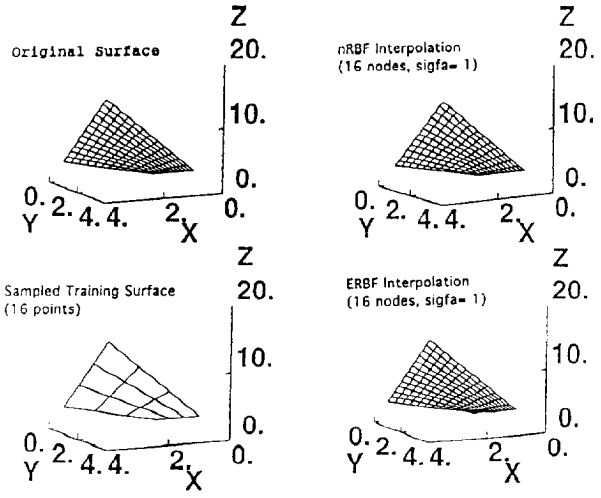

Figure 3: Interpolation: ERBF vs nRBF

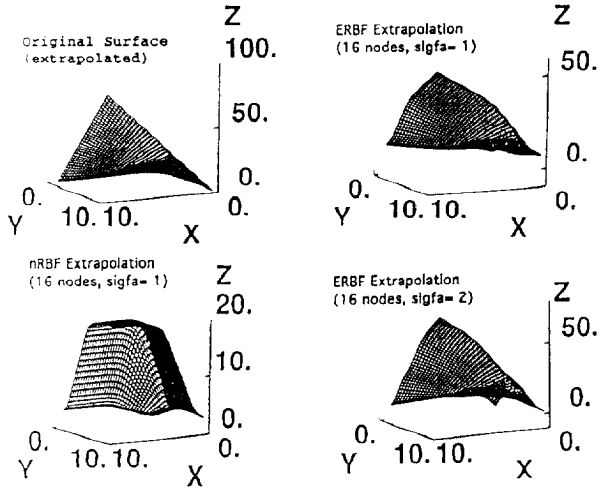

Figure 4: Extrapolation: ERBF vs nRBF

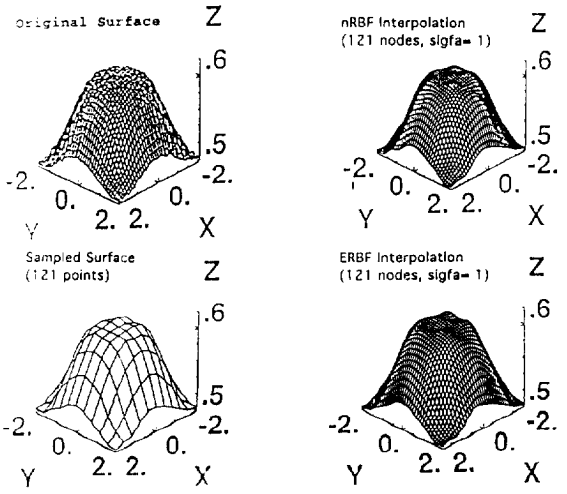

Figure 5: Interpolation: ERBF vs nRBF (even distribution)

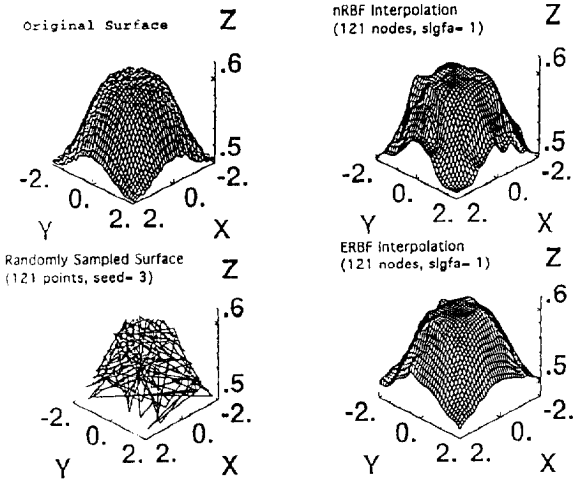

Figure 6: Interpolation: ERBF vs nRBF (uneven distribution)

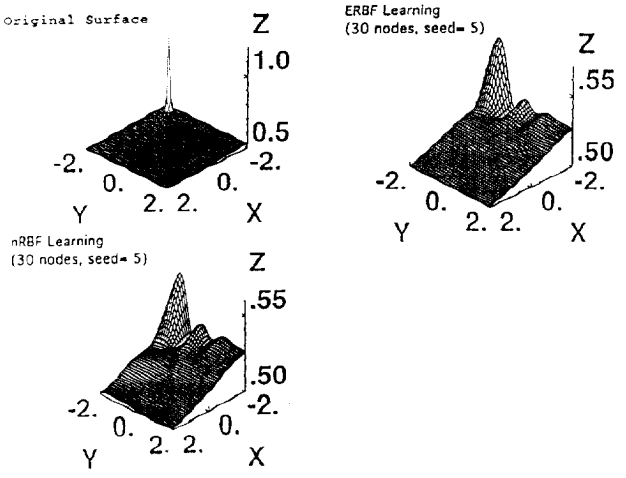

Figure 7: Learning a Difficult Surface with 30 Nodes