Evolutionary Optimization of RBF Networks

E. G. M. de Lacerda
Center of Informatics
Pernambuco Federal University
Recife, PE, Brazil

A. C. P. L. F. de Carvalho
Dept. of Comp. and Inf. Science
University of Guelph
Guelph, ON, Canada

T. B. Ludermir
Center of Informatics
Pernambuco Federal University
Recife, PE, Brazil

Abstract

One of the main obstacles to the widespread use of artificial neural networks is the difficulty of adequately define values for their free parameters. This article discusses how Radial Basis Function, RBF, networks can have their parameters defined by genetic algorithms. For such, it presents an overall view of the problems involved and the different approaches used to genetically optimize RBF networks. Finally, a model is proposed which includes representation, crossover operator and multiobjective optimization criteria. Experimental results using this model are presented.

1. Introduction

Although artificial neural networks, ANNs, have usually achieved good performances when applied to a large number of application domains, these performances are directly influenced by the appropriate choice of architecture and learning parameters. Several alternative approaches have been proposed to select the network parameters. These approaches may be grouped in four different categories:

- Trial and error;
- Pruning techniques;
- Constructive training algorithms;
- Evolutionary design.

When trial and error is employed, different values for the network parameters must be selected, trained and compared before the choice of an ultimate network. This disadvantage becomes more apparent if, after the choice of the best values, the patterns set is changed, making necessary to re-start the design process. This search can be made more efficient if heuristics are used to guide it. The use of pruning techniques optimizes trained networks by removing neurons and connections that are irrelevant or redundant. In the constructive approach, a network starts its training with a minimal topology and, according to the problem complexity, new neurons and connections are inserted, aiming to improve the network performance. The evolutionary approach uses genetic algorithms, GAs, to generate several networks variations and combine the features of those with the best performance, thus generating new networks with improved performances through a number of generations. A good survey on genetic algorithms can be found in [9].

RBF networks are briefly described in Section 2. Section 3 discusses the evolutionary optimization of RBF neural networks. Section 4 explains the proposed approach. Experimental studies comparing the proposed approach with different approaches are shown in the Section 5. Finally, Section 6 presents the conclusions.

2. Radial Basis Function Networks

RBF networks have their origin in the solution of the multivariate interpolation problem [20, 4]. These networks have traditionally only one hidden layer. Properly trained, they can approximate an arbitrary function $f : \Re^n \rightarrow \Re$ by mapping:

$$f(\mathbf{x}) \approx w_0 + \sum_{j=1}^{m} w_j z_j(\mathbf{x}) \qquad (1)$$

where, $\mathbf{x} \in \Re^n$, $\{w_i;\ i = 1, \ldots, m\}$ denotes the *weights* coefficients, w_0 is the *bias* and $z_j(\mathbf{x})$ represents the *activation function* (also known as radial basis function), which is

1522-4899/00 $10.00 © 2000 IEEE

given by:

$$z_j(\mathbf{x}) = \phi\left(\frac{\|\mathbf{x}-\mathbf{c}_j\|}{\sigma_j}\right) \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm, $\mathbf{c}_j = [c_{j1}, c_{j2}, \ldots, c_{jn}]^{\mathsf{T}}$ is the *center vector*, σ_j is the *width*, which is a scaling factor for the radius $\|\mathbf{x}-\mathbf{c}_j\|$, and $\phi(\cdot)$ is a non-linear function that monotonically decreases (or increases) as $\mathbf{x}$ moves away from $\mathbf{c}_j$. A common example of a radial basis function is the Gaussian function $\phi(v) = \exp(-v^2/2)$. Others examples are: $\phi(v) = v$ (linear); $\phi(v) = v^3$ (cubic); $\phi(v) = v^2 \log v$ (thin plate spline); $\phi(v) = \sqrt{v^2+1}$ (multiquadratic); $\phi(v) = 1/\sqrt{v^2+1}$ (inverse multiquadratic).

2.1 Hybrid Learning of RBF networks

Several training techniques have been proposed to train RBF networks. A well known RBF training technique [16] employs a hybrid approach that combines unsupervised and supervised learning. In order to see how it works, let

$$\{(\mathbf{x}_i, t_i);\ i = 1, \ldots, p\} \tag{3}$$

be the set of the training patterns where $\mathbf{x}_i$ is a *input vector* and t_i its *desired output.*

The unsupervised learning stage defines the center and width of the radial basis functions. Simple methods make $\mathbf{c}_i = \mathbf{x}_{\alpha_i}$, for $i = 1, \ldots, m$, where $\alpha_i \in \{1, \ldots, p\}$ is randomly chosen. Usually, the number of centers, m, is determined by trial and error. Nevertheless, this approach is prone to generate large networks, overfitting, and numerical problems (mainly when the data set is noisy) [18].

A more efficient approach employs a clustering algorithm, such as K-means [2] or self-organizing feature map [12]. Roughly, the K-means starts by randomly assigning the p input vectors $\mathbf{x}_j$ to K sets $S_1, \ldots, S_K$. Next, it computes the mean vectors of each set as:

$$\mathbf{m}_i = \frac{1}{|S_i|}\sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j \tag{4}$$

In the following steps, it re-assigns all input vectors $\mathbf{x}_j$ to the nearest cluster S_i (i.e. nearest mean vector) and recalculates the mean vector for each cluster. This two steps procedure is repeated until there is no further change in the mean vectors. The mean vectors become the centers (i.e. $\mathbf{c}_i = \mathbf{m}_i$ for $i = 1, \ldots, K$). Another alternative is to partition the input space in regions using a decision-tree and fix the centers on strategic positions inside these regions [13].

The widths are usually defined through heuristics. Some use a single value σ for all basis functions. In [16], the use of $\sigma = \langle\|\mathbf{c}_i - \mathbf{c}_j\|\rangle$ is suggested, where $\mathbf{c}_j$ is the nearest center from $\mathbf{c}_i$ and $\langle\cdot\rangle$ indicates the average over all such pairs. Other methods use a different value σ_i for each basis function. As an example, let Ψ_i be the set of the N training patterns nearest to $\mathbf{c}_i$. The local width σ_i is thus given by:

$$\sigma_i^2 = \alpha\frac{1}{N}\sum_{\mathbf{x}_j \in \Psi_i} \|\mathbf{c}_i - \mathbf{x}_j\|^2 \tag{5}$$

where α is an overlap factor.

In the supervised learning stage, the RBF network with fixed centers and widths can be interpreted as a case of linear regression on the training set:

$$\mathbf{t} = \mathbf{Z}\mathbf{w} + \mathbf{e} \tag{6}$$

where $\mathbf{t} = [t_1, t_2, \ldots, t_p]^{\mathsf{T}}$, is the desired output, $\mathbf{Z}$ is the *transformed input*, which is a matrix with the j th column $[z_j(\mathbf{x}_1), z_j(\mathbf{x}_1), \ldots, z_j(\mathbf{x}_p)]^{\mathsf{T}}$, $\mathbf{w} = [w_1, w_2, \ldots, w_m]^{\mathsf{T}}$ is the output layer weight vector and $\mathbf{e}$ is the error. The vector $\mathbf{w}$ is determined minimizing the sum of squared errors SSE $= \mathbf{e}^{\mathsf{T}}\mathbf{e}$. The solution to this least square problem can be obtained solving the well-known linear system:

$$(\mathbf{Z}^{\mathsf{T}}\mathbf{Z})\,\mathbf{w} = \mathbf{Z}^{\mathsf{T}}\mathbf{t} \tag{7}$$

In order to avoid possible numerical problems (ill-conditioning) in solving (7), the use of Singular Value Decomposition, SVD, has been recommended [21]. SVD computes the pseudo-inverse matrix $\mathbf{Z}^{+}$. Thus, the weight vector $\mathbf{w}$ is given by $\mathbf{w} = \mathbf{Z}^{+}\mathbf{t}$ where $\mathbf{Z}^{+} = (\mathbf{Z}^{\mathsf{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathsf{T}}$. The LMS algorithm [24] can also be used to determinate the weight vector $\mathbf{w}$ or to fine-tune the solution of (6) found by SVD.

3. Evolutionary Optimization of Neural Networks

The evolutionary optimization of ANNs may occur, roughly, in three different ways:

- Optimization of topology and training parameters: this parameters may include the number of layers, the number of hidden units, the activation function, the learning rate, etc;
- ANN training: a GA may be used as a training algorithm to optimize the values of the network weights;
- Learning rule optimization: given an ANN, this method looks for an efficient learning rule.

This article is concerned with the optimization of the topology and training parameters, named here evolutionary design. In the evolutionary design of ANNs, each chromosome usually corresponds to a network architecture. The evolutionary process starts with a population of chromosomes randomly generated. All the chromosomes are trained with the same training set. Next, a fitness function is used to evaluate the population, establishing the fitness for each chromosome according to its performance on the training set. The best networks are selected to the next generation, where genetic operators, like crossover and mutation, produce a new population. The selection process drives the population to better networks. Crossover and Mutation drive it to explore unknown regions of the search space. Eventually, the population converges to the best network architecture.

Although most of the evolutionary approaches for neural networks design have been focused on MLP networks, their long training time is a strong negative factor concerning the design efficiency. RBF networks are known for requiring a much shorter training period. To take advantage of this feature, a few methods have also been proposed to optimize the parameters of RBF networks.

The optimization of the RBF network topology has been pursued by through other approaches, like OLS (Orthogonal Least Square) [6] and RAN (Resource Allocating Network) [19]. Despite being very fast, these methods do local search; thus they can easily fall in local minima and produce suboptimal solutions. GAs, on the other hand, are global search methods. Thus, they can provide an efficient alternative for the optimization of RBF networks.

Next, a few methods that have been proposed in the literature to optimize RBF networks using GAs are discussed.

3.1. Selecting Centers from Patterns

In [3] was addressed the combinatorial aspect of the RBF network optimization whose aim is to extract a subset of patterns to set centers vectors. The chromosome represents a subset of patterns by means of a variable length list of labels which each label represents a pattern. Per example, the chromosome

$$(100, 7, 411, 286)$$

represents four centers drawn from patterns labeled as 100, 7, 411 and 286. The genetic operators are the same used to solve the so-called *subset selection problem* [14]. This simple representation reduced significantly the number of centers compared with traditional approaches. To evaluate the performance of their approach, the authors calculated the individuals' fitness using the AIC (Akaike's Information Criterion) [1] over the training and validation set with a Multiobjective GA [9].

In [15], it was modified the model proposed in [3] by allowing the centers to be fixed not only on the training input vectors. In this model was also investigated the use of mixed radial basis functions in the same network. According to authors, the networks with different radial basis functions presented a smaller number of hidden nodes and achieved lower error rates than those with only Gaussian functions.

3.2. Crossing Hypervolumes

In [5] was proposed a method to genetically optimize the centers and widths of a RBF network. In their representation, the chromosome is represented by a list de genes where each gene represents a hidden unit. It was used a different Gaussian basis function which may have a width for each component of the center vector. This basis function is given by

$$z_j(\mathbf{x}) = \prod_{k=1}^{n} \exp\left(-\frac{(x_k - c_{jk})^2}{\sigma_{jk}^2}\right) \tag{8}$$

Each gene of chromossome is represented by tuple:

$$\mathbf{p}_j = (c_{1j}, \sigma_{1j}, c_{2j}, \sigma_{2j}, \ldots, c_{mj}, \sigma_{mj}) \,. \tag{9}$$

The authors also proposed a modified 2-point crossover, which exchanges hypervolumes of the input space instead of chunks of the chromosome structure. This hypervolume is determined by two crosspoint vectors $\mathbf{a}, \mathbf{b} \in \Re^n$, whose elements are given by:

$$\begin{aligned} a_j &= \min_j + (\max_j - \min_j)\, r_1 & (10) \\ b_j &= a_j + (\max_j - \min_j)\, r_2^{1/n} & (11) \end{aligned}$$

where r_1 and r_2 are randomly selected from the range $[0, 1]$ with uniform probability density and $[\min_j, \max_j]$ is the allowed range for the component x_j of the input vector $\mathbf{x}$.

The main advantage of this crossover operator is that, by crossing hypervolumes of the input space, most of the *functional equivalence problem* can be avoided.

3.3. Functional Equivalence

A model proposed by [17] tackles the functional equivalence problem (also called competing conventions problem). Two chromosomes are functionally equivalent if they code RBF networks performing the same input-output mapping. This can happen for two networks whose hidden nodes are the same, but located at different positions. Figure 1 illustrates this situation. This problem substantially increases the search space by generating, without need, different chromosomes for networks with the same functionality.

To deal with this problem, the author created a unique representation for a class of functionally equivalent RBF networks. Such representation was based on a lexicographic ordering of the genes defined by the author. It was also proposed genetic operators specially suited to work with this representation. However, in the article consulted, the author did not present any experimental results for his model.

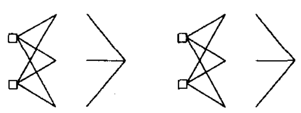

Figure 1. Functionally equivalent RBF networks.

3.4. Others Models

Additional models have been suggested for the evolutionary design of RBF networks. In [22], it was proposed a genetic representation that evolves space-filling curves to define the centers of the RBFs. The basic idea behind their representation is the mapping of the centers m-dimensional space, situated along the space filling curves, to a unidimensional space in which the chromosome is encoded. In another model proposed by same authors [23], the centers and widths of the RBFs evolve through an elegant cooperative-competitive genetic algorithm. In this model, each individual encodes only one hidden unit. The whole population represents just one RBF network. In [7], RBF networks are trained using the OLS algorithm with regularization. GAs are employed to evolve the widths and regularization parameters, after the OLS algorithm defines automatically the number and position of the RBF centers.

4. Proposed Model

This section presents the evolutionary approach proposed in this article by describing the chromosome representation, genetic operators and objective function employed.

4.1. Representation

The solutions representation is one of the key aspects to be considered when using GAs. In the proposed representation, the ith chromosome of the population is a list of genes given by:

$$\mathbf{P}_i = (\sigma, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K) \tag{12}$$

where σ is the width for all basis function and $\mathbf{p}_j$ encodes a hidden unit which is given by

$$\mathbf{p}_j = \{c_{j1}, c_{j2}, \ldots, c_{jn}\} \tag{13}$$

which represents the jth basis function of the network whose center is $\mathbf{c}_j = [c_{j1}, \ldots, c_{jn}]^{\mathsf{T}}$. Each $\mathbf{p}_j$ can be empty ($\mathbf{p}_j = \emptyset$) or non empty ($\mathbf{p}_j \neq \emptyset$). Thus chromosomes $\mathbf{P}_i$ can represent variable size networks topologies. The parameters are encoded as floating-point values.

The index j of $\mathbf{p}_j$ also indicates that the center $\mathbf{c}_j$ is inside a region R_j of the input space (this region is discussed in the next section).

```
Let $\mathbf{P}_i$, $\mathbf{Q}_i$ be two parents and $\mathbf{C}$ their child.
for $j = 1$ to $K$ then
    if $\mathbf{p}_j \neq \emptyset$ and $\mathbf{q}_j \neq \emptyset$ then
        $\mathbf{c}_j = \text{crossover}(\mathbf{p}_j, \mathbf{q}_j)$
    else
        $\mathbf{c}_j = \mathbf{p}_j$ or $\mathbf{q}_j$ with equal probability.
    endif
endfor
```

Figure 2. New crossover operator.

4.2. Partitioning the input space

The partitioning of the input space creates a set of K regions $\{R_1, \ldots, R_K\}$. The regions are placed in the areas of higher density of training patterns. Each region R_i is obtained from the cluster S_i generated by the K-means algorithm. Each region R_i has the shape of a hypercube whose length l_i of its edge is given by

$$l_i = 2 \max_{\mathbf{x}_j \in S_i} \|\mathbf{m}_i - \mathbf{x}_j\| \tag{14}$$

where $\mathbf{m}_i$ is the mean of cluster S_i. Thus the region R_i embraces all points of the cluster S_i.

4.3. Genetic Operators

A new crossover operator is used by the authors. Figure 2 illustrates this operator where the function `crossover()` represents any traditional crossover for floating-point values. This crossover is an uniform crossover that crosses regions R_i instead of structural chunks of chromosomes. This modification avoids most of the duplication of hidden units in the chromosomes and, consequently, most of the functional equivalence problems.

The traditional mutation operator and a few variations are also used. Creep mutation adds a Gaussian noise with a Normal distribution to widths and centers. The added noise is small, so creep mutation plays the role of a local search. Addition and removal operators add and delete hidden units randomly chosen.

4.4. The choice of the objective function

After decoding the chromosome, the output-layer weights are determined by a pseudoinverse matrix. Next, the chromossome is evaluated. Since the topology will be optimized, such evaluation should consider not only the network performance, but stimulate a good balance between its performance and complexity.

A good criterion for the objective function would be the use of crossvalidation. However, it is computationally intensive. For RBF networks with fixed centers and widths there are more computationally efficient criteria for their evaluation. A good candidate for the objective function is the Generalized Crossvalidation (GCV) [10]. The GCV is defined as:

$$f_1 = \mathrm{GCV} = \frac{p\,\mathrm{SSE}}{(p-m)^2} \tag{15}$$

where SSE denotes sum of squared errors on the training set, m is the number of weights (free parameters) and p is the number of the training set patterns.

In previous studies, the objective function employed has been based on the training set error rates. As this may result in overfitting, the experiments reported in this article used a validation set too. As a result, a second objective function must be optimized:

$$f_2 = \mathrm{SSE}_{valid} \tag{16}$$

where SSE_{valid} denotes the sum of squared errors over validation set. This result into a *multiobjective optimization problem*. The Multiobjective optimization using GAs can be found in [8]

Experimental studies showed the following procedure performed in each evaluation of chromossome improve the generalization:

1. join both training and validation set into a unique large dataset.
2. shuffle the large dataset.
3. sample a new training set and validation set from the large dataset.

If a chromossome pass to next generation, it should be evaluated again using the above procedure.

Next Section presents the results achieved using different techniques to set the RBF networks parameters for a Hermite Polynomial approximation task.

5. Experimental Studies

In this section the proposed GA is applied to a benchmark problem: a Hermite polynomial approximation, which is given by:

$$f(x) = 1.1(1 - x - 2x^2)\exp\left(-\frac{x^2}{2}\right) \tag{17}$$

Table 1. GA parameters

Population	100
Generations	200
No. of regions	15

Crossover rate	0.60
Mutation rate	0.01
Creep rate	0.5
Creep std dev.	0.001
Addition rate	0.3
Deletion rate	0.3

The genetic algorithm ran with the parameters from Table 1. The results obtained were compared to two constructive algorithms used to determinate the RBF network topology. Namely, RAN-EKF (Resource Allocating Network with Extended Kalmon Filter) [11] and ROLS (Regularized Orthogonal Least Square) [18]. Both RAN-EKF and ROLS used training sets with 40 patterns randomly chosen in range [-4, +4] and noise added. As GA needs validation set, then the genetics RBF networks was trained with 90% (36 patterns) of the original training set and remainder 10% was left to the validation set.

The results was performed over a test set with 200 uniformly spaced noiseless patterns in the range [-4, +4]. Results of GA were averaged over 10 runs for each different noise added to the training set. In Figures 3 and 4 is showed how the number of centers and the root mean squared error for the test set varies with the noise variance. The RAN-EKF and ROLS results were extracted from [18]. As showed in Figure 4, GA generated RBF networks with a very small number of centers in all levels of noise whereas their generalization capacities, Figure 3, was compatible with the ROLS algorithm.

6. Conclusions

This paper investigated the performance of RBF networks optimized by a Genetic Algorithm using two objective functions. GA was compared to two approaches to optimize RBF networks. The GA approach was as accurate as the best of the others approaches and yields networks with significantly less number of hidden units in all experiments this work. The final conclusion is that GA is able to generate parsimonious networks and still keep good generalization. Nevertheless, GA took a long training time (which is orders of magnitude greater than other approaches) to achieved these results. But for a large number of applications, where recognition performance is more important than the training time, the results obtained suggest that the genetic approach is an attractive solution for the design of efficient ANNs

Acknowledgment: The authors would like to thank FAPESP and CNPq for their support.

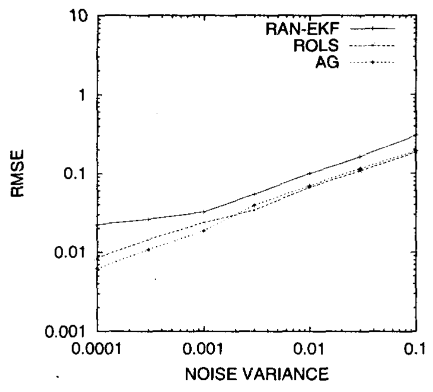

Figure 3. Performance over several noise levels

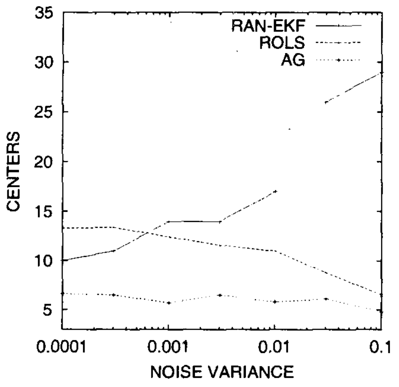

Figure 4. Variation of the number of centers over several noise levels

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[2] M. R. Anderberg. *Cluster Analisys for Applications*. Academic Press, New York, 1973.

[3] S. A. Billings and G. L. Zheng. Radial basis function network configuration using genetic algorithms. *Neural Networks*, 8(6):877–890, 1995.

[4] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[5] B. Carse and T. C. Fogarty. Fast evolucionary learning of minimal radial basis function neural networks using a genetic algorithm. In T. Forgaty, editor, *AISB Workshop on Evolucionary Computing*, Lectures Notes in Computer Science N. 1143, pages 1–22. Springer-Verlag, 1996.

[6] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, 2(2):302–309, 1991.

[7] S. Chen, Y. Wu, and K. Alkadhimi. A two-layer learning method for radial basis function networks using combined genetic and regularised ols algorithms. In *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 245–249, 1995.

[8] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Mateo, 1993. Morgan Kaufmann Publishers, Inc.

[9] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[10] G. H. Golub, M. Heath, and G. Wahba. Generalised cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[11] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.

[12] T. Kohonen. Self-organized formation of topologically correct feacture maps. *Biological Cybernetics*, 43:59–69, 1982.

[13] M. Kubat. Decision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998.

[14] C. Lucasius and G. Kateman. Towards solving subset selection problems with the aid of the genetic algorithm. In *Parallel Problem Solving from Nature Vol. 2*. Elsevier Science Publishers, 1992.

[15] E. P. Maillard and D. Gueriot. RBF neural network, basis functions and genetic algorithm. In *Proceedings of International Conference on Neural Networks Vol. 4*, pages 2187–2192, 1997.

[16] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[17] R. Neruda. Functional equivalence and genetic learning of rbf networks. In D. W. Pearson, N. C. Steele, and R. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 53–56. Springer-Verlag, 1995.

[18] M. J. L. Orr. Regularisation in the selection of radial basis function centers. *Neural Computation*, 7(3):606–623, 1995.

[19] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.

[20] M. Powell. The theory of radial basis function approximation in 1990. In W. Light, editor, *Advances in Numerical Analysis, Vol.3*, pages 105–210. Clarendon, Oxford, 1992.

[21] W. H. Press, S. A. T. B. P. Flannery, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[22] B. A. Whitehead and T. D. Choate. Evolving space-filing curves to distribute radial basis functions over an input space. *IEEE Transactions on Neural Networks*, 5(1):15–23, 1994.

[23] B. A. Whitehead and T. D. Choate. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, 7(4):869–880, 1996.

[24] B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE-WESCON Convention Record*, 4:96–104, 1960.