$T_{ij}, i = 1, \cdots, 5, j = i, \cdots, 5$. A possible solution, obtained using the network described in [5], is:

$$\boldsymbol{T} = \begin{bmatrix} .5 & 1 & 1 & 1 & -1 \\ 1 & .5 & -.5 & -.5 & .5 \\ 1 & -.5 & .5 & -.5 & .5 \\ 1 & -.5 & -.5 & .5 & .5 \\ -1 & .5 & .5 & .5 & .5 \end{bmatrix}. \tag{14}$$

Step (iii): Using matrix (14), it is:

$$\min\{x_i^{\max}, i = 1, \cdots, 5\} = \min \sum_j |T_{ij}| = 3.$$

Being $\varepsilon(n-1)K = 0.4$, it is sufficient to choose $\alpha > g^{-1}(1-\varepsilon)/0.5$ to fulfill the constraints (10). Since

$$g^{-1}(1-\varepsilon) = (2/\pi)\tan[\pi(1-\varepsilon)/2] \cong 4$$

for $\varepsilon = 0.1$, a slope $\alpha = 8$ is enough to satisfy conditions (5) with the actual network parameters.

IV. Conclusions

The concept of binary-valued invariant set has been introduced for continuous-time feedback neural networks with sigmoidal units, including, as a particular case, the Hopfield model. Using this concept, a theoretical result has been proved which extends to this class of networks a well known property of discrete-time networks with two-state neurons. This result can be exploited to develop a synthesis method based on techniques to solve linear inequalities. The synthesis method has been outlined, in the case of symmetric interconnections, and clarified using a five-neuron example.

References

[1] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Contr. Syst. Mag.*, pp. 6–17, Apr. 1990.

[2] J. A. Farrell and A. N. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 877–884, July 1990.

[3] X. C. Jiang, M. Hedge, and M. Naraghi-Pour, "Neural network design using linear programming and relaxation," in *Proc. IEEE ISCAS '90*, New Orleans, LA, May 1990, pp. 1090–1093.

[4] M. Kam, J. C. Chow, and R. Fischl, "Design of the fully connected binary neural network via linear programming," in *Proc. IEEE ISCAS '90*, New Orleans, LA, May 1990, pp. 1094–1097.

[5] R. Perfetti, "A neural network to design neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1099–1103, September 1991.

[6] F. Zou, S. Schwarz, J. A. Nossek, "Cellular neural network design using a learning algorithm," in *Proc. First Workshop Cellular Neural Networks*, Budapest, 1990, pp. 73–81.

[7] S. Grossberg, "Nonlinear neural networks: principles, mechanisms, and architectures," *Neural Networks*, vol. 1, pp. 17–61, 1988.

[8] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*. New York: Academic Press, 1974.

[9] L. O. Chua and D. N. Green, "A qualitative analysis of the behavior of dynamic nonlinear networks: stability of autonomous networks," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 355–379, June 1976.

[10] F. M. A. Salam, Y. Wang, and M.-R. Choi, "On the analysis of dynamic feedback neural nets," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 196–201, Feb. 1991.

[11] J. H. Li, A. N. Michel, and W. Porod, "Qualitative analysis and synthesis of a class of neural networks," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 976–986, Aug. 1988.

[12] A. Guez, V. Protopopsecu, and J. Barhen, "On the stability, storage capacity, and design of nonlinear continuous neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 80–86, 1988.

Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems

J.-S. Roger Jang and C.-T. Sun

***Abstract*—This letter shows that under some minor restrictions, the functional behavior of radial basis function networks (RBFN's) and fuzzy inference systems are actually equivalent. This functional equivalence enables us to apply what has been discovered (learning rule, representational power, etc.) for one of the models to the other, and vice versa. It is of interest to observe that two models stemming from different origins turn out to be functional equivalent.**

I. Introduction

This paper demonstrates the functional equivalence between radial basis function networks (RBFN's) and a simplified class of fuzzy inference systems. Though these two models are motivated from different origins (RBFN's from physiology and fuzzy inference systems from cognitive science), they share common characteristics not only in their operations on data, but also in their learning process to achieve desired mappings. We show that under some minor restrictions, they are functionally equivalent; the learning algorithms and the theorem on representational power for one model can be applied to the other, and vice versa.

II. Radial Basis Function Networks

The locally-tuned and overlapping receptive field is a well-known structure that has been studied in regions of cerebral cortex, the visual cortex, etc. Based on the biological receptive fields, Moody and Darken [6], [7] proposed a network structure, RBFN, that employs local receptive fields to perform function mappings. Fig. 1 shows the schematic diagram of an RBFN with five receptive field units; the output of ith receptive field unit (or hidden unit) is

$$w_i = R_i(\vec{x}) = \hat{R}_i(\|\vec{x} - \vec{c}_i\|/\sigma_i), \qquad i = 1, 2, \cdots, H \tag{1}$$

where $\vec{x}$ is an N-dimensional input vector, $\vec{c}_i$ is a vector with the same dimension as $\vec{x}$, H is the number of receptive field units, and $R_i(\cdot)$ is the ith receptive field response with a single maximum at the origin. Typically, $R_i(\cdot)$ is chosen as a Gaussian function

$$R_i(\vec{x}) = \exp\left[-\frac{\|\vec{x} - \vec{c}_i\|^2}{\sigma_i^2}\right]. \tag{2}$$

Thus the radial basis function w_i computed by the ith hidden units is maximum when the input vector $\vec{x}$ is near the center $\vec{c}_i$ of that unit.

The output of an RBFN can be computed in two ways. For the simpler one, as shown in Fig. 1, the output is the weighted sum of the function value associated with each receptive field:

$$f(\vec{x}) = \sum_{i=1}^{H} f_i w_i = \sum_{i=1}^{H} f_i R_i(\vec{x}) \tag{3}$$

where f_i is the function value, or strength, of ith receptive field. With the addition of lateral connections (not shown in Fig. 1) between the

Manuscript received April 29, 1992. This was supported in part by NASA Grant NCC-2-275, LLNL Grant ISCR 89-12, Micro State Program Award 90-191, and MICRO Industry Rockwell Grant B02302532.

The authors are with the Department of Electrical and Computer Science, University of California, Berkeley, CA 94720.

IEEE Log Number 9204538.

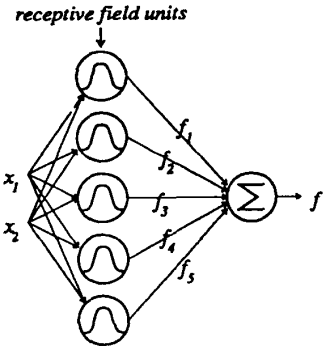


Fig. 1. An RBFN.

receptive field units, the network can produce the normalized response function as the weighted average of the strengths [6]:

$$f(\vec{x}) - \frac{\sum_{i=1}^{H} f_i w_i}{\sum_{i=1}^{H} w_i} = \frac{\sum_{i=1}^{H} f_i R_i(\vec{x})}{\sum_{i=1}^{H} R_i(\vec{x})}. \tag{4}$$

To minimize the square errors between desired output and model output, several learning algorithms have been proposed to identify the parameters ($\vec{c}_i, \sigma_i$, and f_i) of an RBFN. Moody *et al.* [6] use a self-organizing techniques to find the centers ($\vec{c}_i$) and widths (σ_i) of the receptive fields, and then employ the supervised Adaline or LMS learning rule to identify f_i. On the other hand, Chen *et al.* [1] apply orthogonal least squares learning algorithm to determine those parameters.

III. Fuzzy If-Then Rules and Fuzzy Inference Systems

An example of *Fuzzy if-then rules* (or *fuzzy conditional statement*) is

$$If\ pressure\ is\ high, then\ volume\ is\ small.$$

where *pressure* and *volume* are *linguistic variables* [13], *high* and *small* are *linguistic values* (or *linguistic labels*) characterized by appropriate membership functions. Another type of fuzzy if-then rule, proposed by Takagi and Sugeno [10], has fuzzy sets involved only in the premise part. For instance, the dependency of that air resistance (force) on the speed of a moving object can be described as

$$If\ velocity\ is\ high, then\ force = k * (velocity)^2.$$

where *high* is the only linguistic label here, and the consequent part is described by a nonfuzzy equations of the input variable, velocity.

Fuzzy inference systems are also known as *fuzzy rule based systems, fuzzy models, fuzzy associative memories*, or *fuzzy controllers* when used as controllers. A fuzzy inference system is composed of a set of fuzzy if-then rules, a database containing membership functions of linguistic labels, and an inference mechanism called *fuzzy reasoning*. Suppose we have a rule base consisting of two fuzzy if-then rules of Takagi and Sugeno's type:

Rule 1: If x_1 is A_1 and x_2 is B_1, then $f_1 = a_1 x_1 + b_1 x_2 + c_1$.

Rule 2: If x_1 is A_2 and x_2 is B_2, then $f_2 = a_2 x_1 + b_2 x_2 + c_2$.

then the fuzzy reasoning mechanism can be illustrated in Fig. 2(a) where the firing strength (or weight) of ith rule is obtained as the T-norm (usually min. or multiplication operator) of the membership values on the premise part

$$\begin{aligned} w_i &= \mu_{A_i}(x_1)\mu_{B_i}(x_2), \text{ or} \\ &= \min\{\mu_{A_i}(x_1), \mu_{B_i}(x_2)\}. \end{aligned} \tag{5}$$

Note that the overall output can be chosen either as the weighted sum of each rule's output [9], [2]

$$f(\vec{x}) = \sum_{i=1}^{R} w_i f_i \tag{6}$$

or more conventionally, as the weighted average [10] (as shown in Fig. 2 (a))

$$f(\vec{x}) = \frac{\sum_{i=1}^{R} w_i f_i}{\sum_{i=1}^{R} w_i}. \tag{7}$$

where R is the number of fuzzy if-then rules.

Fuzzy modeling concerns the identification of the structure (number of rules, partition pattern, etc.) and parameters of fuzzy inference systems. Various methodologies of fuzzy modeling have been proposed in the past years. Takagi *et al.* [11] and Sugeno *et al.* [8] employ nonlinear programming and heuristic search to identify both the structure and parameters. Hariawa *et al.* [2] and Takagi *et al.* [9] introduce feedforward neural networks into fuzzy inference systems and solve the parameter identification problems through neural network's learning algorithm. Jang [4], [5], [3] propose a more direct method which transforms the fuzzy inference system into a functional equivalent adaptive network (Fig. 2 (b)) and then employ both the back-propagation-type gradient descent to update premise parameters (which determine the shapes and positions of membership functions) and the least square method to identify consequent parameters (which specify the output of each rule). In the proposed adaptive network shown in Fig. 2 (b), there is no weight associated with each link and nodes in different layer can have different functions corresponding to each steps in the fuzzy reasoning mechanism. More specifically, layer 1 calculates membership values, layer 2 perform T-norm operator, layer 3 computes normalized weights, layer 4 derives the product of each rule's output and corresponding normalized weight, and layer 5 sums its inputs as the overall output. For a more in-depth coverage, see [3]–[5].

IV. Functional Equivalence and Its Implication

From (3), (4), (6), and (7), it is obvious that the functional equivalence between an RBFN and a fuzzy inference system can be established if the following is true.

1) The number of receptive field units is equal to the number of fuzzy if-then rules.

2) The output of each fuzzy if-then rule is composed of a constant. (Namely, a_1, b_1, a_2, and b_2 are zeros in Fig. 2 (a).)

3) The membership functions within each rule are chosen as Gaussian functions with the same variance.

4) The T-norm operator used to compute each rule's firing strength is multiplication.

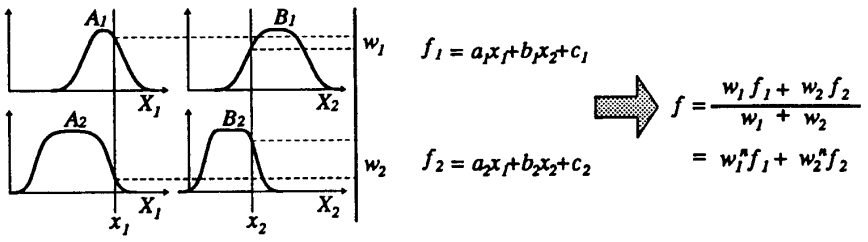
5) Both the RBFN and the fuzzy inference system under consideration use the same method (i.e., either weighted average or weighted sum) to derive their overall outputs.

Under these conditions, the membership functions of linguistic labels A_1 and B_1 in Fig. 2 (a) can be expressed as

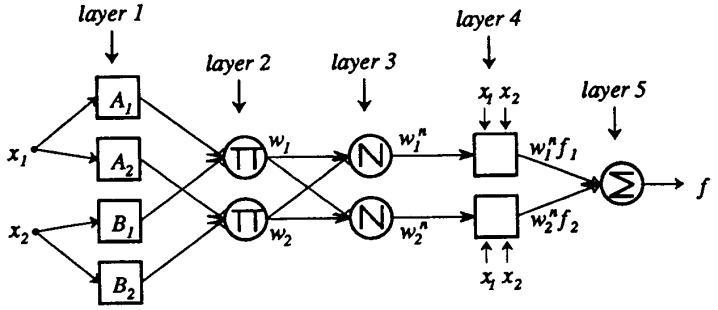
$$\begin{aligned} \mu_{A_1}(x_1) &= \exp\left[-\frac{(x_1 - c_{A_1})^2}{\sigma_1^2}\right], \\ \mu_{B_1}(x_2) &= \exp\left[-\frac{(x_2 - c_{B_1})^2}{\sigma_1^2}\right]. \end{aligned} \tag{8}$$

Hence the firing strength (or weight) of rule 1 (the output of the first node in layer 2) is

$$w_1(x_1, x_2) = \mu_{A_1}(x_1)\mu_{B_1}(x_2) = \exp\left[-\frac{\|\vec{x} - \vec{c}_1\|^2}{\sigma_1^2}\right] = R_1(\vec{x}). \tag{9}$$



(a)



(b)

Fig. 2. (a) fuzzy reasoning; (b) adaptive network representation.

where $\vec{c_1} = (c_{A_1}, c_{B_1})$, the center of the corresponding receptive field. The same argument applies to w_2. Therefore under the above constraints, the output of Fig. 2(a) or (b) is exactly the same as an RBFN (with two receptive field units) where the receptive field units and output units are functionally equivalent to the cascades of layer 1, 2 and layer 3, 4, 5, respectively, in Fig. 2. Without the above constraints, RBFN's are only a special case of fuzzy inference systems.

Because of the functional equivalence shown above, we can apply what is known about one model to the other, and vice versa. In other words, we can apply the learning rules of RBFN's mentioned in Section II to fuzzy inference systems, and the learning rules of fuzzy inference systems in Section III can also be utilized to find the structure (i.e., number of receptive field units) and parameters of RBFN's. Moreover, recently Wang [12] proved that a fuzzy inference system with membership functions of scaled Gaussian functions

$$\mu_A(x) = k * \exp\left[-\frac{(x-c)^2}{\sigma^2}\right] \tag{10}$$

is actually a universal approximator that can approximate any nonlinear input–output data arbitrarily well on a compact set. This argument can be readily applied to RBFN's if the receptive field response in (2) is also scaled by a constant.

V. Concluding Remarks

In this letter, we briefly introduce the structure and learning rules of RBFN's and fuzzy inference systems. Some minor restrictions that renders the functional equivalence of these two models are also discussed. Due to the equivalence of these models, it becomes straightforward to apply one model's learning rules to the other, and vice versa. Furthermore, we can claim both models are universal approximators if the receptive field responses and membership functions are chosen as a scaled version of Gaussian functions. It is of interest to observe that these two models, though derived from different origins and each with different interpretations on its process of data, turn out to be functionally equivalent.

Acknowledgment

The guidance and help of Prof. Lotfi A. Zadeh and other members of the "fuzzy group" at UC Berkeley is gratefully acknowledged.

References

[1] S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, Mar. 1991.

[2] S. Horiawa, T. Furuhashi, S. Ouma, and Y. Uchikawa, "A fuzzy controller using a neural network and its capability to learn expert's control rules," in *Proc. Int. Conf. Fuzzy Logic and Neural Networks*, Japan, 1990, pp. 103–106.

[3] J.-S. Roger Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst. Man, Cybern.* 1991, submitted for publication.

[4] J.-S. Roger Jang, "Fuzzy modeling using generalized neural networks and Kalman filter algorithm," in *Proc. Ninth National Conf. Artificial Intelligence* (AAAI-91), July 1991, pp. 762–767.

[5] J.-S. Roger Jang, "Rule extraction using generalized neural networks," in *Proc. 4th IFSA World Congress*, July 1991.

[6] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., Carnegie Mellon University, Morgan Kaufmann Publishers, 1988.

[7] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[8] M. Sugeno and G.T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, pp. 15–33, 1988.

[9] H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. Approximate Reasoning*," vol. 5, no. 3, pp. 191–212, 1991.

[10] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," *Proc. IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, July 1983, pp. 55–60.

[11] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern*, vol. 15, pp. 116–132, 1985.

[12] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proc. IEEE Int. Conf., Fuzzy Systems*, San Diego, CA, Mar. 1992.
[13] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst. Man, Cybern.*, vol. 3, pp. 28–44, Jan. 1973.

Efficient Implementation of the Boltzmann Machine Algorithm

A. DeGloria, P. Faraboschi, and M. Olivieri

Abstract— **The availability of efficient software implementations of neural network algorithms is a key task in the development phase to evaluate the network results in real cases. In this letter we address the problem of optimizing the sequential algorithm for the Boltzmann Machine (BM). We present a solution which is based on the locality properties of the algorithm and enables a very efficient computation of the cost difference between two configurations. Since the algorithm performance depends on the number of accepted state transitions in the annealing process, we formulate a theoretical procedure to estimate the acceptance probability of a state transition. In addition, we provide experimental data on a well-known optimization problem (TSP) to have a numerical verification of the theory, and to show that the proposed solution obtains a speedup between 3 and 4 in comparison with the traditional algorithm.**

I. Introduction

The Boltzmann Machine (BM) is an interesting neural network which can be used in the fields of combinatorial optimization problems [1], [2], [10], knowledge representation and learning [8]. Rigorous mathematical foundations show convergence properties that can be analyzed by using techniques derived from physics [3].

The huge runtime requirements of the BM have limited its applications in real cases. Long simulation runtime is mainly due to the annealing task performed by the network, whose basic operation is the computation of the cost difference between two configurations.

Although parallel hardware approaches [4], [7] are the best solution for practical applications, efficient implementations of the BM algorithm on conventional workstations are undoubtedly very useful. When approaching a new problem, there is always an algorithm development stage to verify the validity of the BM technique and to tune the methods for weight computation. In this context, a fast sequential BM implementation becomes a primary subject of investigation, and solutions which introduce improvements in the BM algorithm without violating the BM theory can be significantly interesting.

In this letter, we analyze the problem of accelerating the BM algorithm on conventional hardware, and we show a technique to restructure the algorithm which reaches a speedup around three with respect to standard implementations. This is possible by means of the introduction of a different method to compute the cost difference between two configurations based on locality properties of the neurons.

Manuscript received April 29, 1992.
The authors are with the Department of Biophysics and Electrical Engineering (DIBE), University of Genoa, 16145, Genoa, Italy.
IEEE Log Number 9204539.

After a brief presentation of the BM model and the standard algorithm, we show the proposed implementation of the BM, an analytical estimation of the expected performance, and some experimental results on the Traveling Salesman Problem (TSP).

II. The Formal Model of the BM

For reader's convenience, we present an outline of the formal model of the BM, following [3] with minor modifications.

A BM consists of a number N of logical neurons, that can be represented by an undirected graph composed of vertices connected by edges. A number is associated with each vertex, denoting the state of the corresponding logical neuron, i.e., 0 or 1, corresponding to *"off"* or *"on"*, respectively. A configuration k of the BM is uniquely defined by the states of all individual vertices. The state of the ith vertex in configuration k is denoted by $S_i^{(k)}$.

- An edge between vertices i and j defined to be activated in a given configuration k if $S_i^{(k)} S_j^{(k)} = 1$.
- A weight $(W_{ij},)$ is associated with each edge, determining the strength of the connection between the vertices.
- A consensus function (C_k) of a configuration k measures the desirability of all the activated edges in the configuration, and can be defined as:

$$C_k = \sum_{i,j,j \geq i} W_{ij} S_i^{(k)} S_j^{(k)}. \tag{1}$$

From a given configuration k, a neighboring configuration k_i can be obtained by changing the state of the vertex i, so that:

$$S_j^{(k_i)} = \begin{cases} S_j^{(k)} & j \neq i \\ 1 - S_j^{(k)} & j = i \end{cases}. \tag{2}$$

The corresponding difference in the consensus $\Delta C_{kk_i} = C_{k_i} - C_k$ is given by:

$$\Delta C_{kk_i} = \left(1 - 2S_i^{(k)}\right) h_i \tag{3}$$

where

$$h_i = \sum_{i,j,j \neq i} W_{ij} S_j^{(k)} + W_{ii} \tag{4}$$

is the so-called "local field" of neuron i and W_{ii} is the bias of neuron i.

The difference of consensus $\Delta C_{k,k_i}$ is completely determined by the states of the neurons j connected to i and by their corresponding weights, so that it can be computed locally, thus allowing a parallel execution [7].

The probability to accept a transition (k, k_i) with cost $\Delta C_{k,k_i}$ is given by:

$$B_{kk_i} = \frac{1}{1 + e^{-\frac{\Delta C_{kk_i}}{T}}} \tag{5}$$

where T is a cooling parameter, real and positive, whose initial value T_0 is initialized on the basis of the $\sum_{ij} |W_{ij}|$ [1], and $T_{j+1} = \beta \cdot T_j$, $\beta < 1$ and β "close" to 1. The decrement rule for calculating the next value of the cooling parameter T_{j+1} is applied each time the unit has completed a number K of trials.

The annealing process begins with $T = T_0$ and several (K) consecutive trials of transitions (k, k_i) are randomly tossed according

1045–9227/93$03.00 © 1993 IEEE