Behaviormetrika, Vol. 26, No. 1, 5-28
1999

ANALYSIS OF KNOWLEDGE REPRESENTATIONS IN CASCADE CORRELATION NETWORKS

Yoshio Takane*, Yuriko Oshima-Takane* and Thomas R. Shultz*

Feed-forward neural network models approximate nonlinear functions connecting inputs to outputs. The cascade correlation (CC) learning algorithm allows networks to grow dynamically starting from the simplest network topology to solve increasingly more difficult problems. It has been demonstrated that the CC network can solve a wide range of problems including those for which other kinds of networks (e.g., back-propagation networks) have been found to fail. In this paper we show the mechanism and characteristics of nonlinear function learning and representations in CC networks, their generalization capabilities, the effects of environmental bias, etc., using a variety of knowledge representation analysis tools.

1. Introduction

Feed-forward neural network (NN) models approximate functions that connect input to output variables. It is essential that a network is of the right size to achieve good performance; the network should be able to approximate the required functions sufficiently well while preserving good generalization performance (Geman, Bienenstock & Doursat, 1992). Too small a network is not flexible enough to represent the required functions, whereas too large a network is too sensitive to local features of the functions (over-fitting), resulting in poor generalization performance. However, the complexity of the functions to be approximated is usually not known in advance.

There are at least two representative strategies to deal with the unknown complexity of the required functions. One approach is to start with a very large network that is flexible enough to approximate a wide range of functions, and after the training is done, prune unnecessary units and/or connections to enhance network's generalization performance. This class of algorithm is called "pruning". Various pruning and regularization techniques have been developed for this purpose (Reed, 1993). The second approach proceeds in the opposite direction. It starts with a small network, to which hidden units are added and trained to reduce error until a satisfactory degree of performance is reached. This class of algorithm is called "constructive". Since the constructive methods are incremental, training

Key Words and Phrases: nonlinear function approximations, activations, contributions, learning, generalization, environmental bias, lesioning of connections, role history of units.

* Department of Psychology, McGill University, Dr. Penfield Avenue, Montreal, Quebec H3A 1B1, Canada. E-mail: takane@takane2.psych.mcgill.ca

The work reported in this paper has been supported by a team grant from Fonds pour la Formation de Chercheurs et l'Aide a la Recherche to the authors. We thank David Buckingham, Heungsun Hwang, Francois Rivest and Sylvain Sirois for their helpful comments on an earlier draft of this paper.

often proceeds much more quickly, leading to more efficient learning (Prechelt, 1996). There is also some evidence suggesting that the constructive approach is more in line with the nature of human learning and development (Shultz & Mareschal, 1997).

Cascade correlation (CC) learning (Fahlman & Lebiere, 1990) is a versatile NN construction technique, which allows the network to grow dynamically according to the complexity of the task to be solved, starting from a network consisting of only the input and output units. Each input unit is directly connected to the output units by adjustable weights. Initial weights are selected randomly, and are adjusted based on target activations given in the training patterns. When performance cannot be improved any further by weight adjustments, a hidden unit with sigmoid activation functions is recruited, producting nonlinear and interaction effects in the mapping of inputs to outputs. Incoming weights to this new unit are determined by maximizing the "correlation" between the unit's activation and network's current error, and are fixed throughout the remainder of the training period. Thus error is not propagated back across different levels of the network, resulting in quicker, more stable convergence. After the hidden unit has been recruited, output weights are readjusted to optimize performance. This cycle of error reduction is repeated until an acceptable range is reached.

No network topology has to be prescribed in CC networks except input and output. An "optimal" network configuration is automatically determined, tailored to the level of difficulty of the task. In CC networks, units are connected in a cascaded manner; the input units and all previously recruited hidden units are connected to more recently recruited hidden units as well as to the output units. This helps capture higher order nonliearities and interaction effects among the input variables with a minimal network configuration. The input units are directly connected to the output units (cross connections). The cross connections capture linear effects of the input variables, which often account for major portions of the variability in the function to be approximated. They can also prevent the network from fitting unnecessary nonlinear functions when only linear relationships exist between the input and output variables.

The CC algorithm has proven to be a powerful nonlinear function approximator. It has been demonstrated that it can solve a wide range of problems including those for which other kinds of networks have been found to fail. For example, CC networks could discriminate between the two interlocking spirals that no previous back-propagation networks could solve (Fahlman & Lebiere, 1990). They could also handle personal pronoun acquisition quite easily (Oshima-Takne, Takane, & Shultz, 1995), a problem with which nonlinear regression and discriminant analysis procedures such as MARS (Multivariate Adaptive Regression Splines; Friedman, 1991) have been found to have considerable difficulties. The CC algorithm has been used to implement a wide variety of other psychological models, particularly in congitive and language development (Shultz, Schmidt,

Buckingham & Mareschal, 1994). Many empirical phenomena such as the balance scale problem (Shultz, Mareschal & Schmidt, 1994), seriation (Mareschal & Shultz, in press), conservation (Shultz, 1998), learning of personal pronouns (Oshima-Takane, Takane & Shultz, 1995; Oshima-Takane, Takane & Takane, 1998; Shultz, Buckingham & Oshima-Takane, 1994), development of velocity, time, and distance concepts (Buckingham & Shultz, 1994), and discrimination shifts (Sirois & Shultz, in press) have been successfully simulated with CC.

Despite these clear demonstrations of its successes, the CC algorithm has not been widely used. Part of the reason is a lack of understanding of how the CC networks function. In this paper we systematically investigate the mechanism and characteristics of nonlinear function learning and approximations in CC networks, their generalization capabilities, the effects of environmenal bias, etc., using a variety of tools for the analysis of network knowledge representations (Takane, Oshima-Takane & Shultz, 1994).

2. The CC Network algorithm

Let $\mathbf{X}$ denote an N by p matrix of input patterns. Its rows represent N input patterns, and its columns p input variables describing the input patterns. Let $\mathbf{Y}$ denote the corresponding matrix (N by K) of output patterns, $\mathbf{Y}=[\mathbf{y}_1, \cdots, \mathbf{y}_K]$, where K indicates the number of output variables. For simplicity, all output variables are assumed to be binary (either $-.5$ or $.5$) in the following exposition.

Three kinds of units are distinguished in multi-layered feed-forward networks: input, output, and hidden units. The input units only send out contributions to other units, while the output units only receive contributions from other units. The hidden units both receive and send contributions. The contributions of a unit are defined as the products of the unit's activations and the weights representing the strengths of sending connections it has with other units. The activations at the input units are the input patterns themselves. Activations at other units are derived from the contributions they receive. The contributions received by a unit are summed and transformed by a sigmoid transformation to obtain activations at the unit. Activations at the output units are called the network's output predictions. The learning algorithm attempts to make the network's output predictions as close as possible to the target output patterns.

As has been noted, the CC network starts from a minimal network configuration consisting of only the input and the output units. The CC network initially attempts to predict $\mathbf{Y}$ based on $\mathbf{X}$ alone. Let $\mathbf{Z}^{(1)}$ denote the matrix of activations at the input units. That is, $\mathbf{Z}^{(1)}=\mathbf{X}$. Let $\hat{\mathbf{Y}}^{(1)}$ represent the matrix of output predictions at this stage. This is obtained by

$$\hat{\mathbf{Y}}^{(1)}=\sigma(\mathbf{Z}^{(1)}\mathbf{W}^{(1)}), \tag{1}$$

where $\mathbf{W}^{(1)}$ is the p by K matrix of weights (called output weights) associated with

the connections from the input units to the output units, and σ is the sigmoidal transformation, i.e., $\sigma(t)=1/\{1+\exp(-t)\}-.5$. (The .5 is subtracted from the usual sigmoidal function to make the range of the output values between $-.5$ and .5.) It should also be understood that when the sigmoidal function takes a matrix (vector) argument, the transformation is applied to each element of the matrix (vector) argument. The weight matrix $\mathbf{W}^{(1)}$ is determined in such a way that

$$\phi^{(1)}(\mathbf{W}^{(1)})=\mathrm{SS}(\mathbf{Y}-\hat{\mathbf{Y}}^{(1)})=\mathrm{tr}(\mathbf{Y}-\hat{\mathbf{Y}}^{(1)})'(\mathbf{Y}-\hat{\mathbf{Y}}^{(1)}) \tag{2}$$

is minimized. (Note that at this stage the CC network is essentially the same as the linear logistic discrimination method.)

If the value of $\min\phi^{(1)}$ is not sufficiently small according to a prescribed criterion, the network recruits a hidden unit whose activations capture as much of the residual variation as possible of the output units left unaccounted for by $\hat{\mathbf{Y}}^{(1)}$, ie., $\mathbf{e}_k^{(1)}=\mathbf{y}_k-\hat{\mathbf{y}}_k^{(1)}$ for $k=1, \cdots, K$. Let $\mathbf{z}^{(1)}$ denote the vector of activations at this hidden unit. This is obtained by

$$\mathbf{z}^{(1)}=\sigma(\mathbf{Z}^{(1)}\mathbf{v}^{(1)}), \tag{3}$$

where $\mathbf{v}^{(1)}$ is a vector of weights (called input weights), determined so as to maximize

$$\phi^2(\mathbf{v}^{(1)})=\sum_{k=1}^{K} | \mathbf{e}_k^{(1)\prime}\mathbf{J}\mathbf{z}^{(1)} |, \tag{4}$$

where $\boldsymbol{e}_k^{(1)}$ is as defined above, and $\mathbf{J}$ is the centering matrix, i.e., $\mathbf{J}=\mathbf{I}_N-\mathbf{1}_N\mathbf{1}_N/N$ where $\mathbf{I}_N$ is the identity matrix of order N, and $\mathbf{1}_N$ is the N-element vector ones. The activations at the first hidden unit recruited (h_1) are ones that "correlate" most highly with the residuals from the previous step. In defining $\boldsymbol{\phi}^{(2)}$ above, the absolute values of the "correlations" between $\mathbf{z}^{(1)}$ and the residuals are taken before they are added across K output units. This is because $\mathbf{z}^{(1)}$ could be positively "correlated" with the residuals from some output units, but negatively with those from other output units. The sign of activations is essentially arbitrary; large negative "correlations" are as good as positive "correlations".

Now the output weights are modified in the light of the new hidden unit introduced. The matrix of activations is redefined by appending $\mathbf{z}^{(1)}$, the vector of activations at h_1, to $\mathbf{Z}^{(1)}$, the matrix of activations in the previous stage, i.e., $\mathbf{Z}^{(2)}=[\mathbf{Z}^{(1)}, \mathbf{z}^{(1)}]$. The matrix of new output predictions is then obtained by

$$\hat{\mathbf{Y}}^{(2)}=\sigma(\mathbf{Z}^{(2)}\mathbf{W}^{(2)}), \tag{5}$$

where $\mathbf{W}^{(2)}$ is the matrix of new output weights, determined in such a way that

$$\phi^{(2)}(\mathbf{W}^{(2)})=\mathrm{SS}(\mathbf{Y}-\hat{\mathbf{Y}}^{(2)}) \tag{6}$$

is minimized. If $\min\ \phi^{(2)}$ is still unsatisfactory, this process is repeated as many times as necessary until the error becomes smaller than the prescribed level. This generates a sequence of $\mathbf{Z}$'s and $\hat{\mathbf{Y}}$'s updated alternately, as in $\mathbf{Z}^{(1)}$, $\hat{\mathbf{Y}}^{(1)}$, $\mathbf{Z}^{(2)}$, $\hat{\mathbf{Y}}^{(2)}$, $\cdots$,

and so on.

Each cycle of optimization defines a distinct stage consisting of two phases: The input training phase in which a hidden unit is recruited and its activation function derived, and the output training phase in which adjustments are made on output weights. At stage g, in general, the vector of input weights $\mathbf{v}^{(g-1)}$ in

$$\mathbf{z}^{(g-1)}=\sigma(\mathbf{Z}^{(g-1)}\mathbf{v}^{(g-1)}) \tag{7}$$

is updated so as to maximize

$$\phi^{(g)}(\mathbf{v}^{(g-1)})=\sum_{k=1}^{K} \mid \mathbf{e}_k^{(g-1)\prime}\mathbf{J}\mathbf{z}^{(g-1)} \mid. \tag{8}$$

The matrix of activations at stage g is then defined by $\mathbf{Z}^{(g)}=[\mathbf{Z}^{(g-1)}, \mathbf{z}^{(g-1)}]$. When $g=1, \mathbf{Z}^{(1)}$ is set to $\mathbf{X}$ without going through any optimization step. The matrix of output weights $\mathbf{W}^{(g)}$

$$\hat{\mathbf{Y}}^{(g)}=\sigma(\mathbf{Z}^{(g)}\mathbf{W}^{(g)}) \tag{9}$$

is then updated in such a way as to minimize

$$\psi^{(g)}(\mathbf{W}^{(g)})=\mathrm{SS}(\mathbf{Y}-\hat{\mathbf{Y}}^{(g)}) \tag{10}$$

In the CC algorithm (Fahlman & Lebiere, 1990) optimizations involved in input and output training are performed by an iterative procedure called Quickprop (Fahlman, 1989). This is a kind of quasi-Newton method in which the matrix of the second order derivatives (the Hessian matrix) is approximated by a numerically estimated diagonal matrix. The optimization in the input training is apparently very difficult; it is prone to convergence to suboptimal solutions. In an attempt to alleviate this problem, several candidate hidden units (eight by default) with different random initial weights are trained concurrently (Fahlman and Lebiere, 1990). The candidate that achieves the largest value of $\phi^{(g)}$ is subsequently incorporated into the network. Quickprop uses a host of other heuristics, which we will not discuss in this paper. See Fahlman (1989) for details.

3. Learning and Representations by CC networks

We now illustrate the CC algorithm by turning to graphic representations of the knowledge learned by CC networks. The knowledge learned by the network is stored in the estimates of connection weights used to approximate the function governing input-output relations. We demonstrate the mechanism of function approximation by showing how the input data and the connection weights are combined into output predictions that approximate functions.

For illustration we use the continuous *xor* problem. This problem is simple, yet complicated enough to require a network solution. It is simple because the target function is known and can be depicted graphically (see Figure 1). It is complicated because the solution requires an interaction effect among input variables. Capturing the interaction effects among the input variables without explicit

instructions to do so is one of the major advantages of NN models. It is interesting to see how the network "perceives" the necessity of interaction effects, and how it actually creates them based exclusively on the examples of input-output relations.

The continuous *xor* problem has two input variables, x_1 and x_2, each ranging from .1 to 1, and one binary output variable, y. The problem is to discriminate between two groups of input patterns. When both x_1 and x_2 are larger than or both smaller than .55, y takes the value of $-.5$; when only one of the two input variables is larger (but the other smaller) than .55, y takes the value of .5. The target function for this problem is given by

$$y=f\{-c(x_1-.55)(x_2-.55)\}-.5, \tag{11}$$

where f is a sigmoid function, and $c\to\infty$. This target function is depicted in Figure 1. A CC network learns to approximate this function from a set of training patterns. It is obvious that this function has an interaction term between x_1 and x_2. The center figure in the bottom row of Figure 8 depicts an approximation of this function by a CC network. The approximation is excellent; it is virtually indistinguishable from the original target function displayed in Figure 1.

The CC algorithm constructs a network and estimates connection weights based on a sample of training patterns. For the continuous *xor* problem 100 triplets of the form, (x_1, x_2, y), were used, both x_1 and x_2 ranging from .1 to 1.0 with a step of .1, and $y=-.5$ or .5. The CC algorithm constructed a network with two hidden units for the solution presented in Figure 8, starting from a network with only a bias unit (the constant term), two input units and one output unit. The final network is presented in Figure 2. A CC network with only two hidden units solving the continuous *xor* problem is somewhat atypical, however. The CC algorithm typically requires four to seven hidden units to solve the problem. The derived network is the result of our effort to get a network with as minimal a configuration as possible. (This was done by running the simulation many times, and choosing the "best" solution among those obtained. In these simulations it seems important to set the number of candidate units in input training equal to one to avoid convergence to a same suboptimal solution all the time.) This simplifies our presentation because network analyses are much easier wth a smaller number of hidden units. It should be emphasized, however, that essentially the same knowledge representation analysis techniques can apply with a larger number of hidden units.

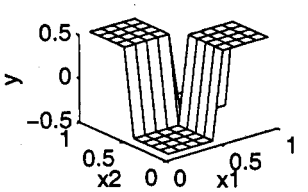

Fig. 1 The target function for the continuous *xor* problem

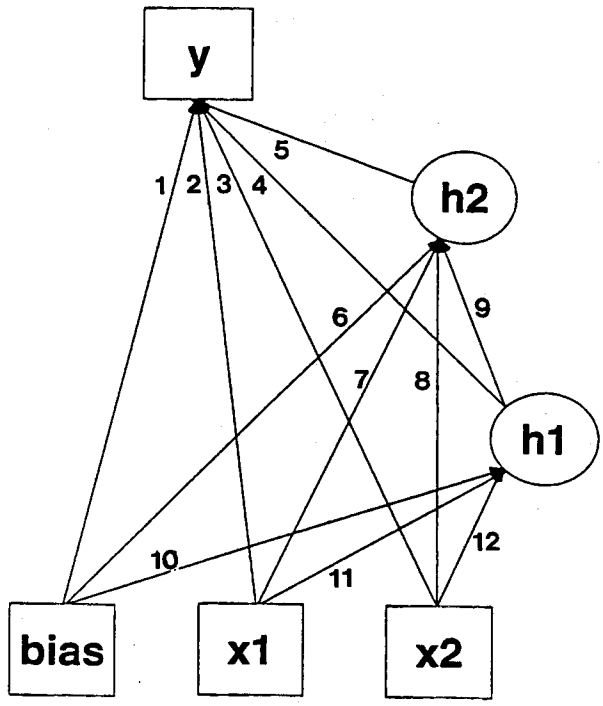

Fig. 2 A CC network configuration dervied for the continuous *xor* problem

The initial network consisted of only connections 1, 2, and 3. To capture part of the interaction effect needed to solve the problem, the hidden unit (h_1) with connections 10, 11, and 12 from the input units and connection 4 to the output unit was added. Weights associated with the input connections define the activation function at this unit. One hidden unit was found insufficient to approximate the target function. To improve network's performance further, a second hidden unit (h_2) was recruited with connections 6, 7, 8, and 9 from the input units and the first hidden unit, and connection 5 to the output unit. Weights associated with the input connections are used to derive activations at this hidden unit, which, in turn, in used to approximate the target function.

We illustrate the above process in more detail. The activation functions at the input units are input data themselves, i.e., $\mathbf{Z}^{(1)}=\mathbf{X}$, where $\mathbf{X}$ is the 100 by 3 matrix of $(1, x_{i1}, x_{i2})$ in the i^{th} row. The three activation functions at the input units are depicted in Figure 3. The bias function is constant (=1) irrespective of the values of x_1 and x_2. The x_1 function is linearly increasing in x_1, but is constant over x_2. The x_2 function, on the other hand, is linearly increasing in x_2, but constant with respect to x_1.

Now a linear combination of $\mathbf{Z}^{(1)}$ is formed and sigmoid-transformed to obtain output predictions at this stage. Let $\mathbf{w}^{(1)}$ denote the weight vector in this linear combination. (Since there is only one output unit in the continuous *xor* problem, the output weights are always vectors.) The $\mathbf{w}^{(1)}$ is determined in such a way that

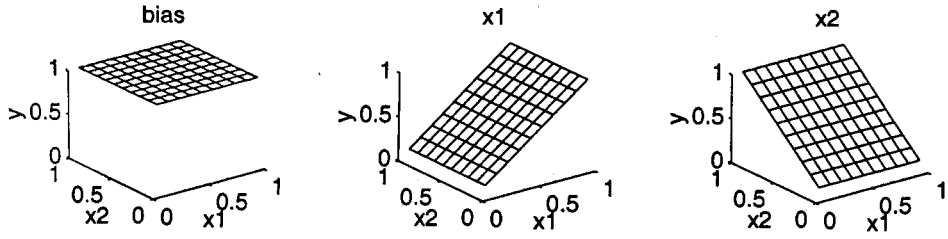

Fig. 3 The activation functions at the input units

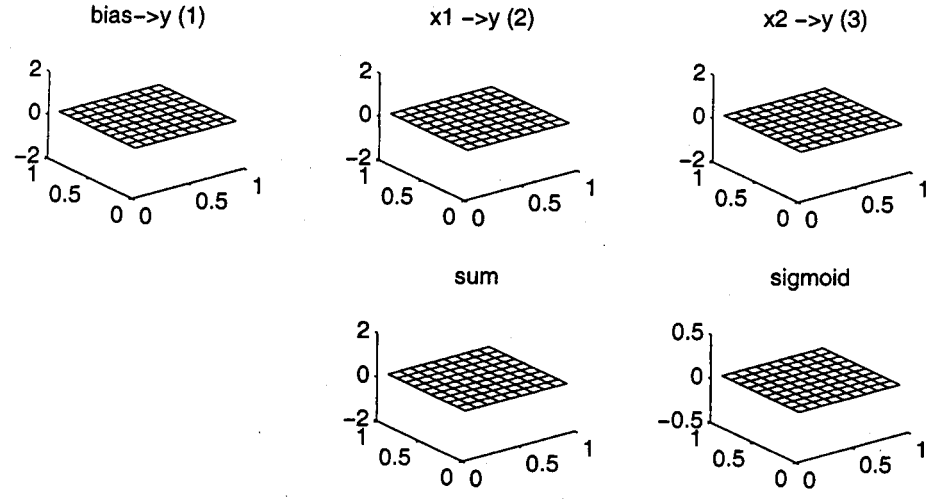

Fig. 4 The contribution functions, the summed contribution function, and the output prediction function at Stage 1. (Numbers in parentheses refer to the connection numbers in Figure 2.)

$\phi^{(1)}$ in (2) is minimized. Let $\mathbf{w}^{(1)}=(w_0^{(1)}, w_1^{(1)}, w_2^{(1)})'$. The product of each element of $\mathbf{w}^{(1)}$ wth the corresponding activation function defines a set of contributions. Three sets of contributions, $bias \cdot w_0^{(1)}$, $x_1 w_1^{(1)}$, and $x_2 w_2^{(1)}$, can be defined. The three contribution functions are depicted in the upper panel of Figure 4. (The numbers in parentheses in the labels correspond with connection numbers in Figure 2.) All of them are nearly constant at zero. In the continuous *xor* problem the crucial variable in the solution is the interaction effect, and consequently the weights for the linear effects of the input variables are all nearly zero, as expected, which explains why the three contributiton functions are all nearly constrant at zero. These contribution functions are added and sigmoid-transformed to obtain the output prediction function. The summed contribution and the output prediction functions are depicted in the lower panel of Figure 4. These functions are also constant near zero, indicating the inability of the linear network to solve the continuous *xor* problem. The SS discrepancy between the target and approximated functions at this stage is nearly 25, indicating that the prediction is no better than at a chance level.

The prediction in the previous stage is not at all satisfactory. So a hidden unit,

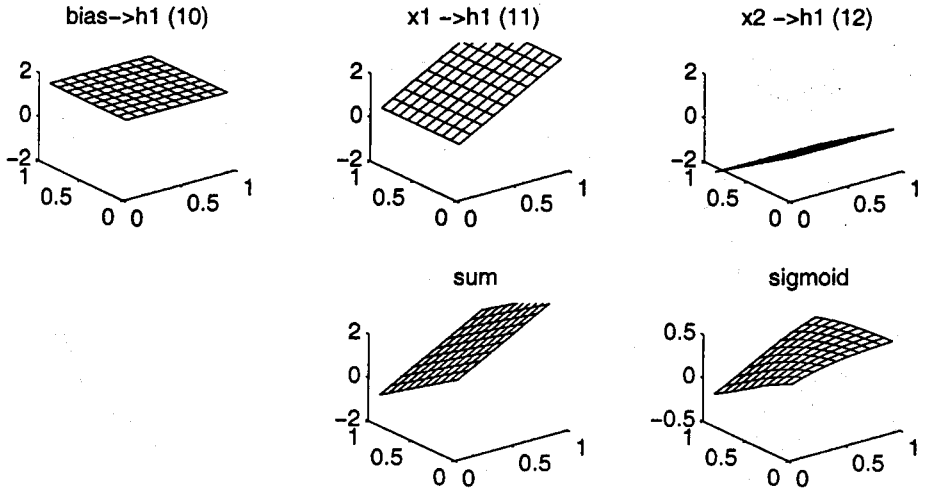

Fig. 5 The contribution functions, the summed contribution function, and the activation function at h_1. (Nunmers in parentheses refer to the connection numbers in Figure 2.)

h_1, is recruited and trained. The activation function at this unit is derived by (3), where $\mathbf{v}_{(1)}$ is determined in such a way that $\phi^{(2)}$ in (4) is maximized. The terms in the linear combination to obtain the activation function, $\mathbf{z}^{(1)}=\sigma(\mathbf{Z}^{(1)}\mathbf{v}^{(1)})$, are the contribution functions from the input units, which are depicted in the upper panel of Figure 5. Again the parenthesized numbers in the labels indicate connection numbers in Figure 2. These contribution functions are summed and sigmoid-transformed to define the activation function at h_1. This activation function, once derived, remains intact throughout the remainder of the learning process. The summed contribution and the h_1 activation functions are depicted in the lower panel of Figure 5. At a first glance this activation function does not seem to be of much use in predicting the output variable. However, it turns out that it helps create an important activation function at the next hidden unit, which plays a crucial role in the final solution.

The matrix of previous activation functions is appended by the new activation function to form a new matrix of activation functions, $\mathbf{Z}^{(2)}=[\mathbf{Z}^{(1)}, \mathbf{z}^{(1)}]$. Output weights are trained and a new set of predictions are derived by (5). The contributions of the bias, the two input units and h_1 are depicted in the upper and the middle rows of Figure 6. Numbers in parentheses in the labels indicate the connection numbers in Figure 2. These contributions are summed and sigmoid-transformed to obtain the output predictions at this stage. The summed contributions and the output predictions are displayed in the bottom row of Figure 6. It may be seen that the derived output prediction function has improved substantially. It captures, albeit imperfectly, the rapid rises in the function value toward both ends of the domain of the target function. The SS discrepancy between the target and the approximated functions was reduced from 25 to 11.30.

The predictions are still less than satisfactory, and so another hidden unit (h_2)

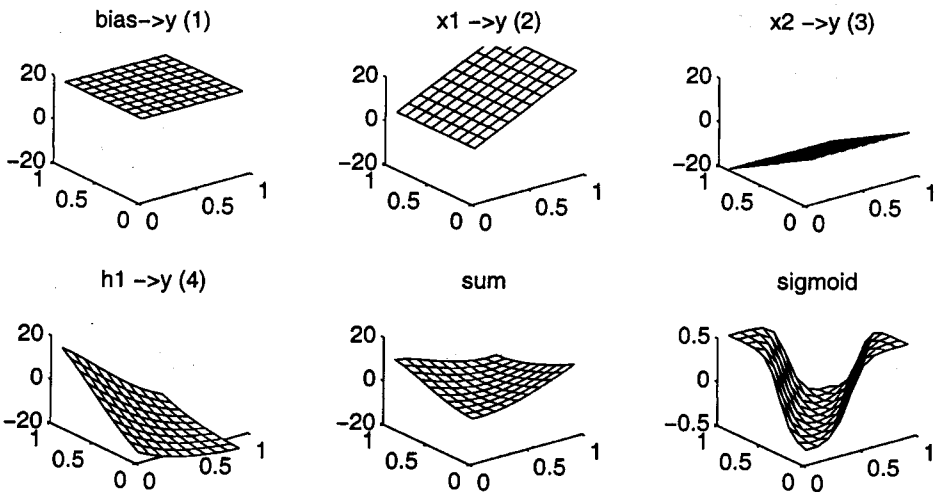

Fig. 6 The contribution functions, the summed contribution function, and the output prediction function at Stage 2. (Numbers in parentheses refer to the connection numbers in Figure 2.)

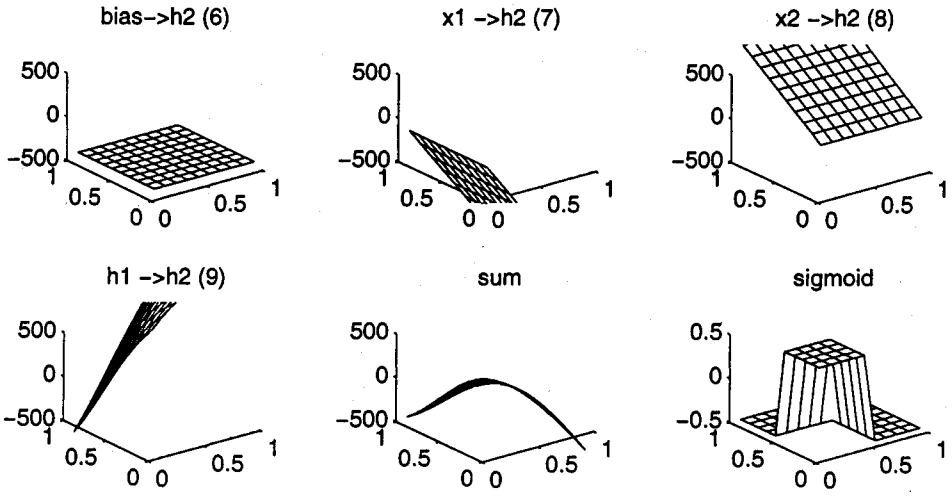

Fig. 7 The contribution functions, the summed contribution function, and the activation function at h_2. (Numbers in parentheses refer to the connection numbers in Figure 2.)

was recruited. The activation function at h_2 is derived by $\mathbf{z}^{(2)}=\sigma(\mathbf{Z}^{(2)}\mathbf{v}^{(2)})$, where the weight vector $\mathbf{v}^{(2)}$ is determined in such a way that $\phi^{(3)}$ is maximized. The terms in $\mathbf{Z}^{(2)}\mathbf{v}^{(2)}$ (the contribution functions) are depicted in the upper panel of Figue 7. The numbers in parentheses refer to connection numbers in Figure 2. These functions are added together, then sigmoid-transformed to obtain the activation function at this unit. The summed contribution and the actvation function at h_2 are displayed in the lower panel of Figure 7. It is remarkable to see the h_2 activation function capture one of the cliffs in the target function.

All the necessary ingredients are now ready, and the final output predictions can be derived. First, the matrix of activation functions is redefined, $\mathbf{Z}^{(3)}=[\mathbf{Z}^{(2)},$

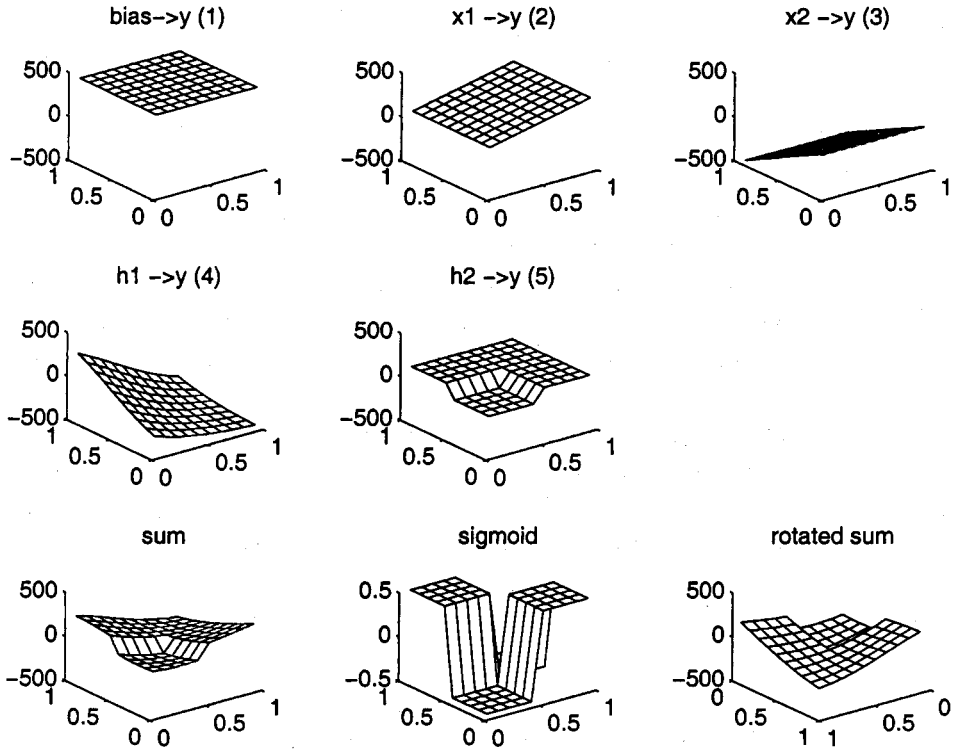

Fig. 8 The contribution functions, the summed contribution function, and the final output prediction function. (Numbers in parentheses refer to the connection numbers in Figure 2.)

$\mathbf{z}^{(2)}$], and the output prediction function is obtained by $\hat{\mathbf{y}}^{(3)}=\sigma(\mathbf{Z}^{(3)}\mathbf{w}^{(3)})$, where $\mathbf{w}^{(3)}$ is determined so as to minimize $\psi^{(3)}$. The contribution functions at this stage are depicted in the first two rows of Figure 8. (The numbers in parentheses indicate connection numbers in Figure 2.) As before, they are summed and sigmoid-transformed to obtain the output predictions. The summed contribution and the output prediction functions are displayed in the first two panels of the bottom row of Figure 8. The graph of the summed contribution function is somewhat misleading. There should be a dip in the function value toward the far end of the function domain, but this may be difficult to see. This is due to the inadequacy of the MATLAB routine used to draw this three-dimensional perspective graph. To show there is indeed a dip toward the far end, the figure was rotated 180°. The rotated graph is presented in the last panel of the bottom row of Figure 8. The far end in the original summed contribution function comes to the front of the graph. It can be clearly seen that there is indeed a dip in the function value. This dip is turned into a deep cliff by the sigmoid transformation. The output prediction function displayed in the center of the bottom row of Figure 8 is very similar to the target function. The SS discrepancy between the target and the approximated functions is now near zero. It is remarkable that the target function with such

sharp edges can be approximated so well by adding only two new activation functions obtained through sigmoid transformations of linear combinations of the input activations.

4. Generalizations

The nonlinear function approximation by the CC network was rather impressive in the example presented above. The output prediction function depicted in Figure 8 was, however, evaluated only for the training patterns. What about the function approximated at patterns not included in the training sample? This is called a generalization problem. There are two aspects to the generalization problem; one is interpolation and the other is extrapolation. The former refers to predictions for untrained patterns within the domain of the function defined by the training sample, and the latter for those outside the domain. In the present case both x_1 and x_2 extend from .1 to 1 in the step of .1, so that predicting a function value, say, at $x_1=.38$ and $x_2=.73$ is an instance of interpolation, while predicting the same at $x_1=-.1$ and $x_2=1.2$, $x_1=.20$ and $x_2=-.3$, or $x_1=1.3$ and $x_1=.65$ involves extrapolation.

Figure 9(a) shows the CC network's interpolation performance for the continuous *xor* problem. The function approximated by the CC network was evaluated at all combinations of the value of x_1 and x_2 in the same range as in the training sample (i.e., between .1 and 1 inclusive), but with a step of one half in size. Thus, every other grid point in the graph represents the prediction at an untrained pattern. The graph indicates that CC network's interpolation performance is excellent; correct output predictions were made for all untrained patterns (as well as for all the training patterns) within the range of x_1 and x_2 given in the training sample. Minor irregularities are observed on the boundaries of the two output classes. However, this is due to the fact that input patterns with $x_1=.55$ and/or $x_2=.55$ are right on the boundaries. It is natural that the network does not know what to do in these cases.

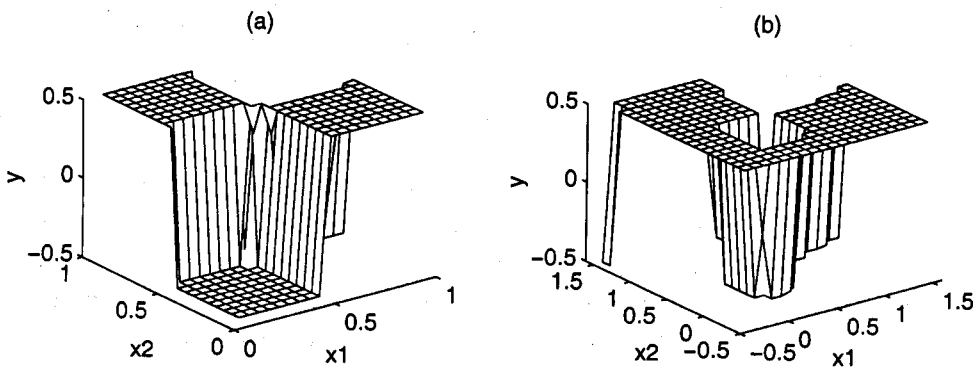

Fig. 9 Generalization performance by the CC network for the continuous *xor* problem: (a) Interpolation. (b) Extrapolation.

Figure 9(b), on the other hand, shows CC network's extrapolation performance for the continuous *xor* problem. The x_1 and x_2 were both varied from $-.4$ to 1.5 with a step of .1. In contrast to the interpolation performance, network's extrapolation performance in rather poor; incorrect predictions are made for some untrained patterns outside the range of the training sample. It can be observed that the value of output predictions becomes .5 as soon as x_1 and/or x_2 get smaller than .1. The output predictions remain relatively stable in other parts of the outside domain. However, it is impossible to predict where the extrapolation succeeds and where it fails.

Extrapolation seems difficult not only with the CC network but also with other kinds of NN models. NN models offer flexible nonlinear function approximation capabilities. However, the very flexibility of the NN models could work adversely in extrapolation. This suggests that a certain kind of knowledge transfer, the kind that involves extrapolation as in analogical reasoning, is difficult in CC networks. Work is in progress, however, to develop a CC network procedure that can accommodate networks representing prior knowledge as part of a new CC network being constructed for solving new tasks.

5. The effects of environmental bias

Typically the approximated function obtained by a CC network varies with the training sample. If a random sample of possible training patterns is used, we may be able to obtain a less biased function approximation. If, on the other hand, a special subset of possible training patterns is used, the resultant function could be quite disparate from the one desired. The effects of the training sample on the approximated function are generally called the problem of environmental bias.

Small numerical experiments were conducted to investigate the effects of environmental bias in the continuous *xor* problem. The specific question we ask is: What happens to the approximated function if all the training patterns used are far away from, or all close to the boundaries of the classes to be discriminated? Two training samples were created. In one case the training sample consisted of only those patterns far away from the boundaries. Specifically, both x_1 and x_2 were varied from .1 to .3 and from .8 to 1 in the step of .05. The values of x_1 were factorially combined with the same set of values of x_2, creating 100 training patterns. In the second case the training sample consisted of patterns near the boundaries of the two classes. Specifically, one of the two input variables was fixed at either .5 or .6, while the other varied from .1 to 1 in the step of .05. That is, when $x_1 = .5$ or .6, x_2 was systematically varied from. 1 to 1 in the step of .05, and when $x_2 = .5$ or .6, x_1 was varied in the same way x_2 was varied before, giving rise to 68 training patterns in total.

Figure 10(a) shows the function approximated by the CC network in the first case where the training patterns are all far away from the boudaries. The network

discriminates the training patterns very well. However, the approximated function is distinctly different from the target function (see Figure 1). It takes a single value (of $-.5$) in the entire central region of the function domain, where no training patterns existed, resulting in incorrect output predictions for about half of the patterns in this region. To remedy this situation, the training sample was augmented by four additional training patterns: (.6, .6, $-.5$), (.5, .5, $-.5$), (.6, .5, .5) and (.5, .6, .5), where the first number in parentheses indicates the value of x_1, the second that of x_2, and the third that of y, near the center of the function domain. The approximated function obtained from the augmented sample is presented in Figure 10(b). (The training with the augmented training sample started from scratch rather than retraining the previous network with the augmented sample.) A fair amount of improvement can be observed in the approximation. However, it still falls short of capturing primary features of the target function.

Figure 10(c) displays the approximated function in the second case where the training patterns are all near the boundaries. The approximation is fairly good, although an unexpected rise in the function value can be clearly seen toward the front end of the function domain, where no training patterns existed. Four training patterns were added to the training sample: (1, 1, $-.5$), (.1, .1, -5), (1, .1, .5) and (.1, 1, .5). They are all located at the far ends of the function domain. Again the

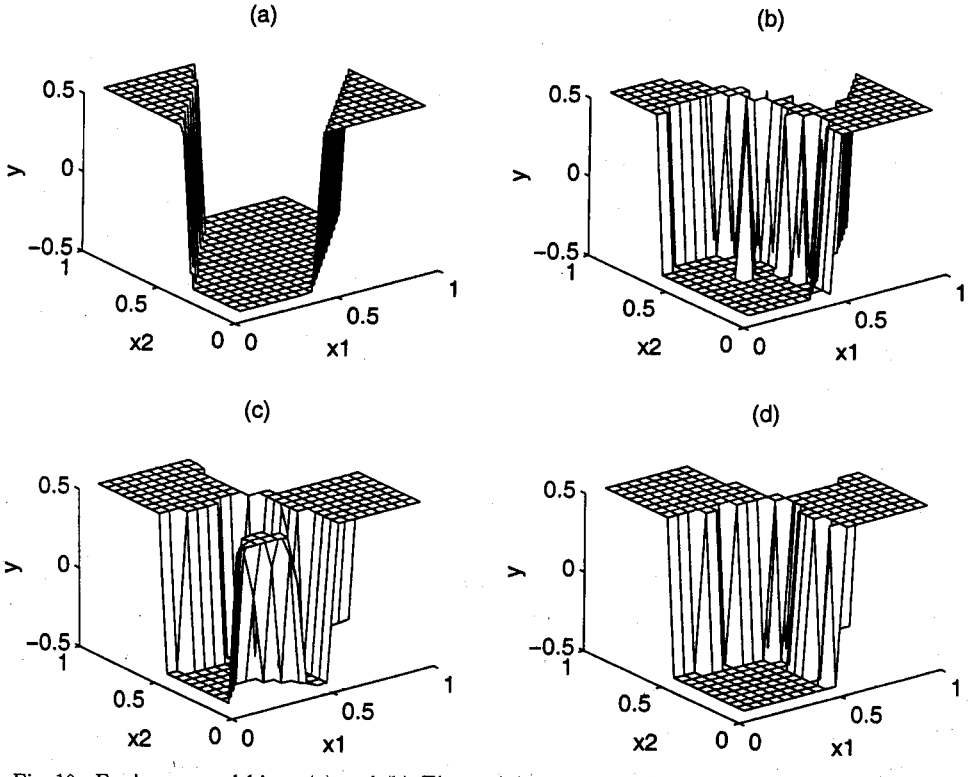

Fig. 10 Environmental bias: (a) and (b) The training patterns far away from the boundaries. (c) and (d) The training patterns near the boundaries.

training with the augmented training sample started from scratch rather than retraining the network derived from the unaugmented data. Figure 10(d) indicates the approximated function with the augmented training sample. The approximation is quite good; no rise in the function values can be observed any longer toward the ends of the function domain. The four added patterns played an important role in defining the extent of the regions in the x_1-x_2 plane associated with the two distinct classes of the output variable. Training patterns bearing crucial information on boundaries are far more important than others which are "buried" in the middle range of the regions. This observation may have an important implication in education. Examples used in teaching a new concept must define clear boundaries of the concept, and the boudaries covered should be far from as well as near to important distinctions.

Investigating the effects of environmental bias is important in a broader perspective, where we may ask more general questions such as: What is the optimal design of the training sample? Or what are the psychological implications of the effects of environmental bias? It has been a long standing problem in statistics to find an optimal set of training patterns for a particular task. In the context of NN models, this problem has been investigated by Cohn and his collaborators (Cohn, 1994; Cohn, Ghahramani & Jordan, 1995). In psychology, changes in environment are considered to be a major influence on learning. Dramatic effects of environmental bias were found in acquisition of personal pronouns (Oshima-Takane, 1988, 1992; Oshima-Takane & Benaroya, 1989; Oshima-Takane, Takane & Shultz, 1995; Oshima-Takane, Takane & Takane, 1998; Shultz, Buckingham & Oshima-Takane, 1994; Takane, Oshima-Takane & Shultz, 1994) as well as in a number of other learning situations (e.g., Tetewsky, Shultz & Takane, 1995).

We illustrate the case of personal pronoun acquisition in a little more detail. Learning of first and second person pronouns presents a psychologically interesting problem (Oshima-Takane, 1988, 1992; Oshima-Takane, Goodz & Derevensky, 1996). When the mother talks to her child, *me* refers to the mother, and *you* refers to the child. However, when the child talks to the mother, *me* refers to the child, and *you* refers to the mother. Learning of the shifting reference of these pronouns can be viewed as a special kind of nonlinear function learning, where the function to be learned stipulates *me* if the speaker and the referent agree, and *you* if the addressee and the referent agree. The effects of environmental bias were investigated under two conditions: an addressee condition in which the addressee is always the child, and a nonaddressee condition in which the child is neither the speaker nor the addressee but an observer of a conversation between others (Oshima-Takane, Takane & Shultz, 1995; Oshima-Takane, Takane & Takane, 1998; Shultz, Buckingham & Takane, 1994; Takane, Oshima-Takane & Shultz, 1995). It was found that exposures to nonaddressee patterns were essential for network's learning of the target function underlying the correct use of the pronouns. It was also found that a large variety of nonaddressee patterns facilitated the

learning of the correct use of the pronouns.

6. Lesioning of connections

It may appear that the effect of a unit on another unit can simply be assessed by the contributions associated with the connection between the two units. The contributions of the twelve connections in the derived network have been presented in Figures 4 through 8. The output connections may be characterized by more than one set of contributions, because the weights associated with these connections are updated every time a new hidden unit is recruited, and their contributions change as the training proceeds. We may focus our attention on a particular output connection and look at the change in the contributions associated with it over time. For example, by connecting the leftmost graphs in the top panel of Figures 4, 6, and 8 in a sequence we can trace the history of contributions of the bias to the output unit. The same can be done for other units (x_1, x_2 & h_1) as well. It looks like that the contributions of these units increase uniformly as the training proceeds. (Note the difference in scale on the vertical axis across these figures.)

These unit contributions, however, represent only linear effects. They are subsequently added and transformed nonlinearly to derive activations of a unit receiving the contributions. We may define another kind of contributions, which might be called nonlinear contributions, by the change in the activations produced by the elimination of a connection or connections. Connections can, in effect, be "eliminated" by setting the weights associated with them equal to zero. Investigating the behavior of the network under lesioning of connections is analogous to observing the behavior of human subjects with brain damage (e.g., Hinton & Shallice, 1991).

Figure 11 depicts function approximations obtained by eliminating one of the twelve connections, in turn, described in Figure 2. Numbers in parentheses give the SS discrepancies between the full approximation using all the twelve connections (the middle graph in the bottom row of Figure 8) and the degraded approximations (Figure 11). They indicate the amount of deterioration in function approximations due to the elimination of a connection. The consequence of the elimination is drastic in all cases ; even the least damaging one (due to the elimination of Connection 8) results in a sizable jump (0 to 18) in the value of SS discrepancy. Elimination of any one of the other connections makes the network performance worse than a chance level. (The chance level here refers to the SS discrepancy value of 25 obtained when all the predictions are 0.)

Figure 12 depicts the nonlinear contributions of the twelve connections. They are obtained by subtracting the deteriorated function approximations (due to a removal of a connection) in Figure 11 from the full approximation. The nonlinear contribution functions look quite different from their linear counterparts; the nonlinear effects of Connections 3($x_2 \rightarrow y$), 4($h_1 \rightarrow y$), 10($bias \rightarrow h_1$), and 11($x_1 \rightarrow h_1$) are

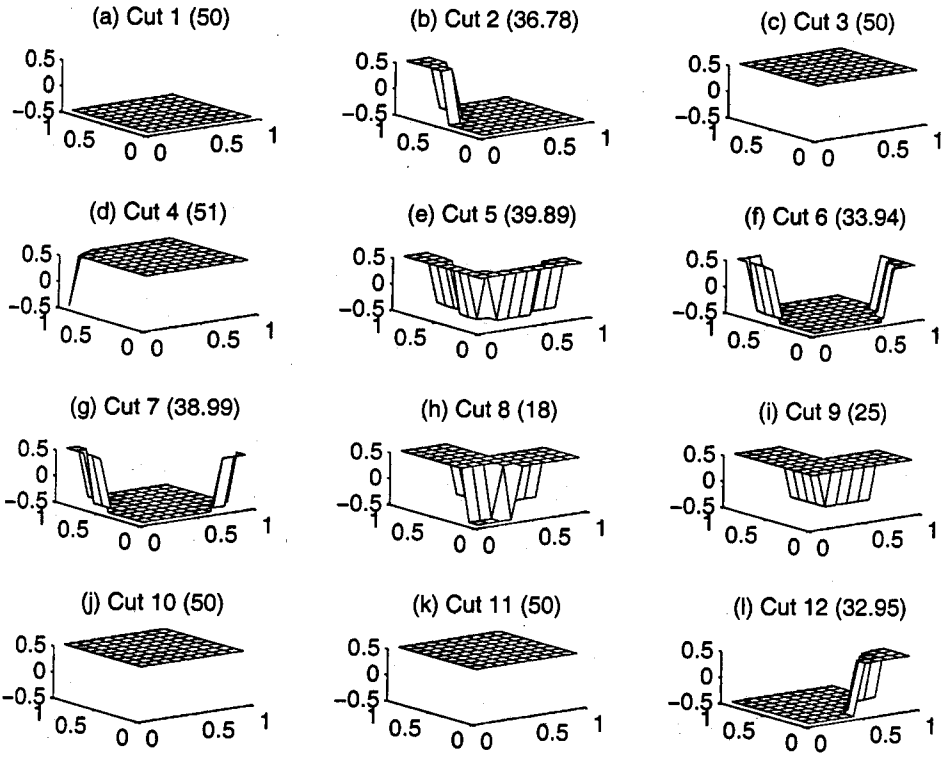

Fig. 11 The reduced function approximations obtained by cutting each one of the twelve connections. (Numbers in parentheses refer to increase in SS.)

similar, despite significant differences in the corresponding linear effects of these connections (compare the rightmost graph in the top row and the leftmost graph in the middle row of Figure 8, and the first two graphs in the top row of Figure 5). A surface that remains constant at zero indicates no nonlinear contributions of the unit associated with it. The surface that comes closest to this description is that of Connection 8. This corresponds with our earlier observation that the effect of eliminating this connection is least damaging in the approximation of the target function.

Although this discussion is only concerned with changes in the activations at the output unit (i.e., changes in output predictions), similar analyses can also be performed on changes in activation functions at the hidden units.

In this analysis only one connection was eliminated at a time. We may eliminate more than one connection to study their joint effects. The number of possible combinations of connections that might be eliminated simultaneously can be enormous even in a small network like the one depicted in Figure 2. There are 2^{11} different ways of eliminating connections in the network with 12 connections. Obviously, it is impossible to examine all of them. However, there are subsets of connections whose joint effects are worth examining. Elimination of a set of connections may entail the total effect of a unit, only the direct effect of the unit or

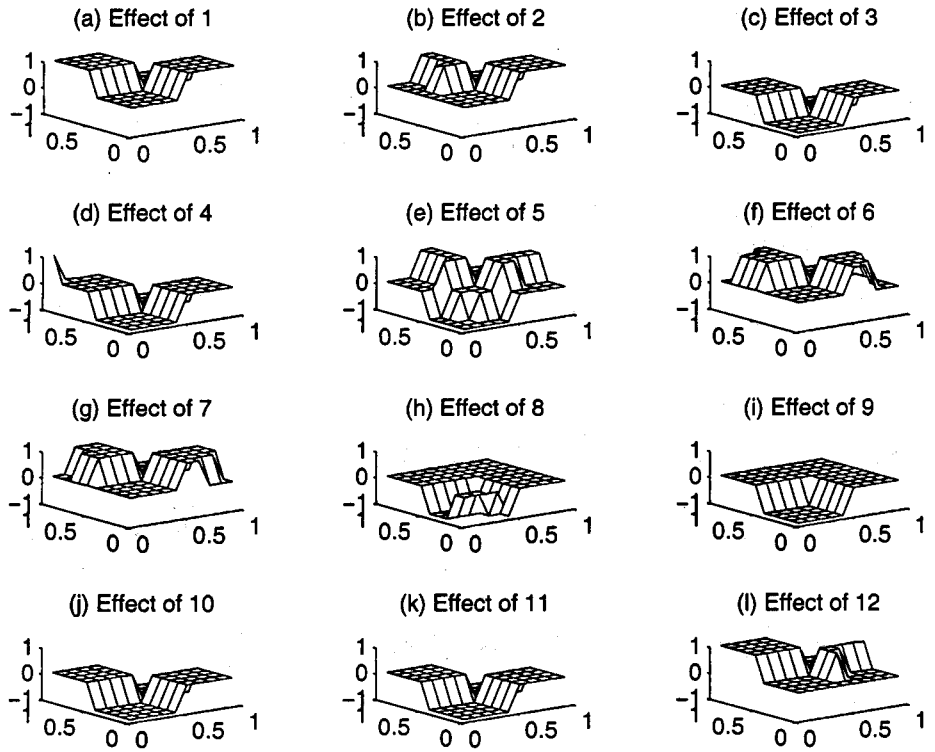

Fig. 12 The nonlinear contribution functions of the twelve connections in the network.

the indirect effects of the unit. For example, the joint effect of Connections 1, 6 and 10 represent the total effect of the bias on the output, of which Connection 1 pertains to the direct effect, while the latter two pertain to the indirect effects of the bias. Figure 13 shows the total and the indirect effects of the bias, x_1, and x_2, and the total effect of h_1. The direct effect of the bias (the effect of Connection 1), and the two partial indirect effects of the bias (separate effects of Connections 6 and 10) were already given in Figure 12. Similarly, the direct effects (the effect of Connection 2 for x_1 and that of Connection 3 for x_2) as well as their partial indirect effects (the separate effects of Connections 7 and 11 for x_1, and those of Connections 8 and 12 for x_2) were given in Figure 12. The direct and the indirect effects of h_1 (the effects of Connections 4 and 9, respectively) were also given in Figure 12. The h_2 unit has only the direct effect (the effect of Connection 5) on the output unit, which was also presented in Figure 12. It is interesting to note that the size of the total effect is not necessarily larger than the size of either the direct or the indirect effect, as measured by the SS discrepancy. That is, simultaneous eliminations of more than one connection may induce smaller overall damage than elimination of anyone of them. For example, the size of the total effect of the bias is 48.03, while the direct and the indirect effects of the bias are both 50. Similarly, the total indirect effect could be smaller than either one of the partial indirect effects. These somewhat counter-intuitive phenomena are entirely due to the nonlinear transformations

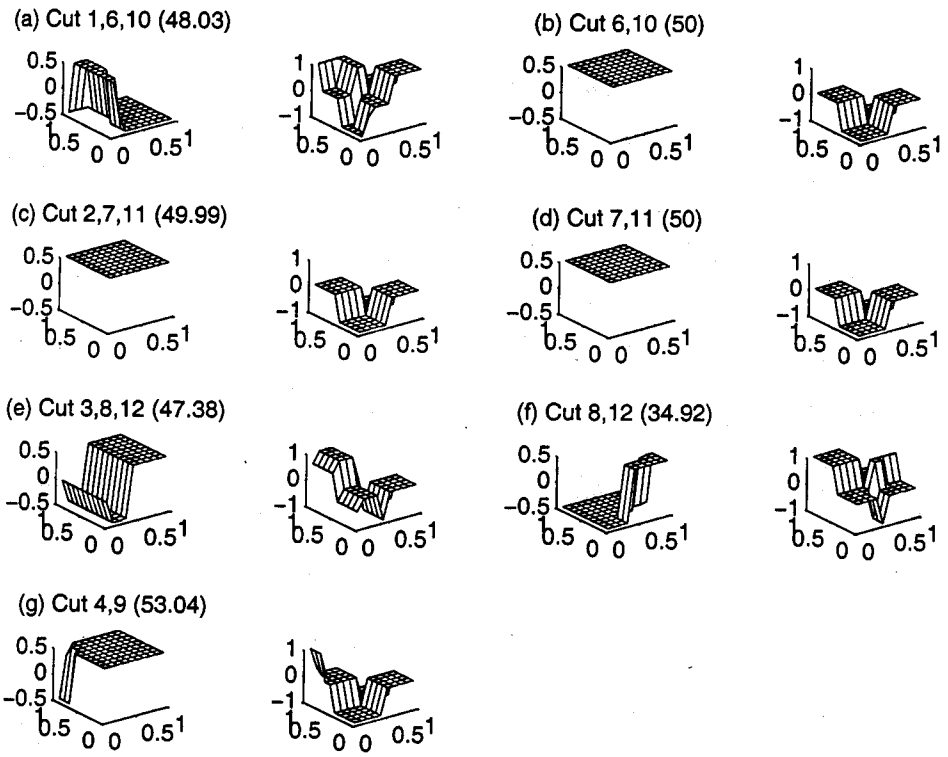

Fig. 13 The total and the indirect effects of units: (a) & (b) for bias, (c) & (d) for x_1, (e) & (f) for x_2, and (g) for h_1. (Numbers in parentheses refer to increase in SS.)

applied to the summed contributions to obtain activation functions in NN models. Although the distinction among the total, the direct and the indirect effects were made above only in terms of the effects on the output unit, similar distinctions can also be made for the effects on hidden units.

Just as the linear contributions evolved over time, the nonlinear contributions also evolve during the course of network training. The nonlinear contributions in earlier stages of network development can be calculated in the same way as in the final stage. We can look at the history of nonlinear contributions of a unit by tracing their changes over time. Figure 14 displays the direct effect of the input variables (the nonlinear contributions of Connections 1, 2 and 3) in stages 1 and 2, and that of h_1 (the nonlinear contributions of Connection 4) in stage 2. The direct effects of the five non-output units (the nonlinear contributions of Connections 1 through 5) in the final stage were given in Figure 12(a)-(e).

In the first stage no units could contribute effectively toward the solution. The output prediction function as well as the nonlinear contribution function of each input unit was completely flat at zero. In the second stage the output prediction was improved substantially. Elimination of any one of the four non-output units (the bias, x_1, x_2, and h_1) flattens out the prediction function at either $-.5$ or .5, indicating that all of them are important for prediction at this stage. The non-

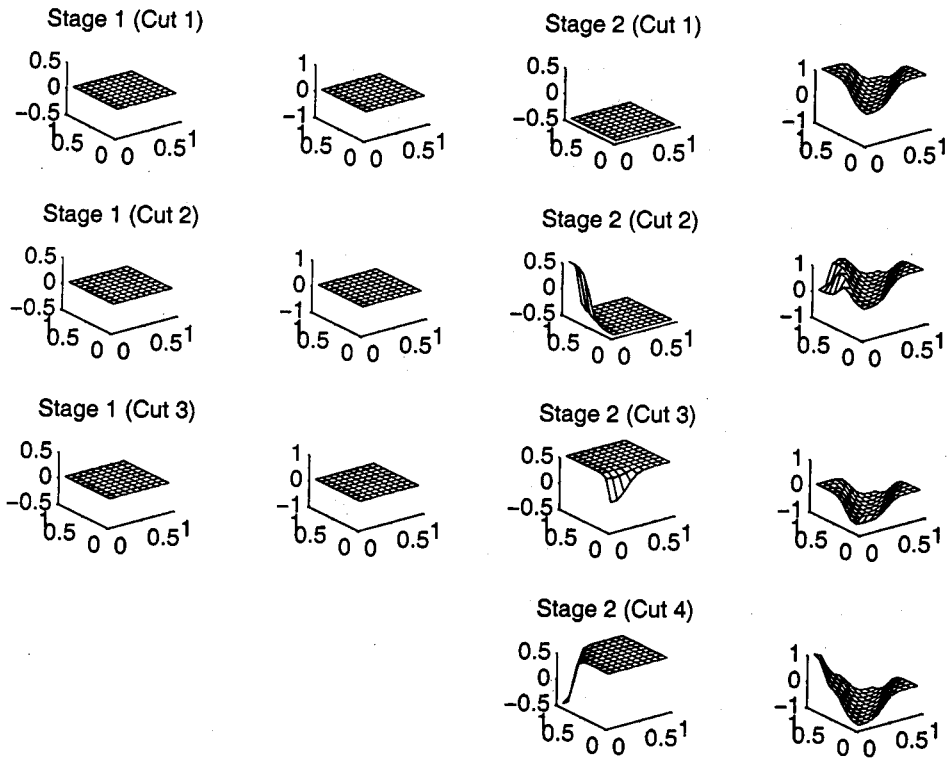

Fig. 14 The change in the nonlinear contributions of connections over different stages.

linear contribution functions of the four units all look similar in shape to the output prediction function at this stage; only their elevations differ. This tendency remains essentially the same in the final stage (see Figure 11(a-e) and Figure 12(a-e)).

7. Discussion

In this paper we have literally seen the mechanism of learning and function approximations in a CC network. We have emphasized the roles that units are playing through their connections with other units. We have analyzed how a new hidden unit is recruited, how activations at hidden units are produced, how contributions are derived and combined to yield network predictions, and how connection strengths are adjusted to give optimal network predictions. We have also analyzed network's sensitivities to untrained patterns (generalizations), to particuar sets of training patterns used (environmental bias), and to deletions of some connections in the derived network. Although the situation (the continuous *xor* problem) taken up in this paper is somewhat limited in scope, we have done similar analyses and demonstrated their usefulness in other contexts (Takane, 1995, 1998) as well as in real empirical situations (pronoun acquisition) with much success (Oshima-Takane,

Takane & Shultz, 1995; Oshima-Takane, Takane & Takane, 1998; Takane, Oshima-Takane & Shultz, 1994).

In all of the above analyses, visualizations of approximated functions play a particularly important role. The visualization strategy works best when there are only up to two input variables (excluding the bias). It still works when the number of input variables in three or perhaps four by slicing the functions; that is, by drawing a series of bivariate functions at several values of the third and the fourth variables. Examples of slicing functions can be found in Moseholm, Taudorf and Frosig (1993), Oshima-Takane, Takane and Takane (1998) and Takane (1988). Beyond that, however, more sophisticated visualization techniques seem to be in order. We refer to Roosen (1995) and Plate, Band, Bert and Grace (1997) for some of the techniques for visualizing high dimensional functions. Roosen proposed a functional ANOVA approach that decomposes approximated functions into the sum of basic functions pertaining to the main and the lower-order interaction effects among the input variables. Plate et al., on the other hand, proposed a special technique for visualizing departures of the approximated functions from the generalized additive models (GAM: Hastie & Tibshirani, 1990). See also the paper by Plate in this volume.

There is one useful technique for the analysis of knowledge representations we deliberately did not discuss in this paper. Linear contributions associated with different connections in the network are usually correlated. This implies that the contributions are partially redundant. Principal component analysis (PCA) may be applied to the matrices of contributions to eliminate this redundancy. The reduced-rank approximations (afforded by PCA) may be graphically presented in a manner similar to the above. The reduced-rank approximations may also improve network's generalization performance (Takane, Oshima-Takane & Shultz, 1994). In the example discussed in this paper, the derived network was already fairly minimal, and there was not much room for further rank reduction. PCA of unstandardized contributions in CC networks has proven to be useful for understanding the nature of the contributions from different units (Shultz, Oshima-Takane & Takane, 1995). Plotting component scores (possibly after a rotation like VARIMAX that improves interpretability) is particularly useful for this purpose. PCA has been applied to the analysis of knowledge representations in CC networks simulating a variety of cognitive-developmental phenomena (e.g., Shultz, 1998; Sirois & Shultz, in press; Tetewsky, Shultz & Takane, 1995).

The matrix of contributions may contain known components. It may, for example, contain linear effects of input variables and the bias. These known effects may be eliminated before PCA is applied to the residuals. This will highlight more interesting aspects of contributions such as interactions among input variables. The summed contributions play a crucial role in network performance. We may eliminate the effect of the summed contributions from the matrix of contributions and analyze the rest. More generally, the matrix of contributions

may be decomposed into several components before each component is subjected to PCA (Takane & Shibayama, 1991). In general, partialing out known or trivial effects from the matrix of contributions is an effective way of extracting unique contributions of particular units.

When there are more than one output unit, contributions to different output units are related in a special way. They are obtained by applying different weights to a same set of activations. We may form a single matrix of contributions by arranging matrices of contributions to different output units side by side and apply PCA to this super matrix. However, it may be more interesting to apply a three-way PCA such as PARAFAC (Harshman, 1972). It may lead to a more parsimonious representations of contributions. PARAFAC solutions are also not rotatable, unlike usual two-way PCA solutions. Unique orientation of axes may facilitate interpretation.

More recently, DCP (Discriminant Component Pruning), a method based on PCA of summed contributions in multi-layered back-propagation networks, has been proposed (Koene & Takane, 1998) to enhance interpretability of NN models. DCP may also be extended to cover CC networks.

References

Buckingham, D. & Shultz, T.R. (1994). A connectionist model of the development of velocity, time, and distance concepts. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 507-511). Hillsdale, NJ: Lawrence Earlbaum Associates.

Cohn, D.A. (1994). Neural network exploration using optimal experiment design. In J.D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems 6* (pp. 679-686). San Mateo, CA: Morgan Kaufmann.

Cohn, D.A., Ghahramani, Z. & Jordan, M.I. (1995). Active learning with statistical models. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 705-712). Cambridge, MA: The MIT Press.

Fahlman, S.E. (1989). Faster-learning variations on back-propagation: An empirical study. In D.S. Touretzky, G.E. Hinton, & T.J. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 33-51). San Mateo, CA: Morgan Kaufmann.

Fahlman, S.E. & Lebiere, C. (1990). The cascade correlation learning architecture. In D.S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524-532). San Mateo, CA: Morgan Kaufmann.

Friedman, J.H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics,* **19**, 1-141.

Geman, S., Bienenstock, E. & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation,* **4**, 1-58.

Harshman, R.A. (1972). Foundations of parafac procedure: Models and conditions for an "exploratory" multi-model factor analysis. UCLA Phonetic Lab Working Paper #16.

Hastie, T.J. & Tibshirani, R.J. (1990). *Generalized additive models.* London: Chapman and Hall.

Hinton, G.E. & Shallice, T. (1991). Lesioning an attractor network: Investigations of acquired dyslexia. *Psychological Review,* **98**, 74-95.

Koene, R. & Takne, Y. (1998). Discriminant component pruning: Regularization and interpretation of multi-layered back-propagation networks. *Neural Computation,* in press.

Mareschal, D. & Shultz, T.R. (in press). Development of children's seriation: a connectionist approach. *Connection Science.*

Moseholm, L., Taudorf, E. & Frosig, A. (1993). Pulmonary function changes in asthmatics associated with low-level SO_2 and NO_2 air pollution, weather, and medicine intake. *Allergy,* **48**, 334-

344.

Oshima-Takane, Y. (1988). Children learn from speech not addressed to them: The case of personal pronouns. *Journal of Child Language,* **15**, 94-108.

Oshima-Takane, Y. (1992). Analysis of pronomial errors: A case study. *Journal of Child Language,* **19**, 111-131.

Oshima-Takane, Y. & Benaroya, S. (1989). An alternative view of pronomial errors in autistic children. *Journal of Autism and Developmental Disorders,* **19**, 73-89.

Oshima-Takane, Y., Goodz, E. & Derevensky, J.L. (1996). Birth order effects on early language development: Do secondborn children learn from overheard speech? *Child Development,* **67**, 621-634.

Oshima-Takane, Y., Takane, Y. & Shultz, T.R. (1995). Learning of personal pronouns: Network models and analysis. Technical Report 2095, McGill Papers in Cognitive Science, McGill University, Montreal.

Oshima-Takane, Y., Takane, M. & Takane, Y. (1998). Learning of first, second, and third person pronouns. *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 800-805). Mahwar, NJ: Lawrence Earlbaum Associates.

Plate, T., Band, P., Bert, J. & Grace, J. (1997). Visualizing the function computed by a feedforward neural network. In Kosabov, N., Kozma, R., Ko, K., O'Shea, R., Coghill, G. & Gedeon, T. (Eds.) *Progress in connectionist-based information systems* (pp. 306-309). Berlin: Springer Verlag.

Prechelt, L. (1996). Investigation of the CasCor family of learning algorithms. *Neural Neuworks,* in press.

Reed, R. (1993). Pruning algorithms—a survey. *IEEE Transactions on Neural Networks,* **4**, 740-746.

Roosen, C.B. (1995). Visualization and exploration of high-dimensional functions through functional ANOVA decomposition. Unpublished Doctoral Dissertation, Stanford University.

Shultz, T.R. (1998). A computational analysis of conservation. *Developmental Science,* **1**, 103-126.

Shultz, T.R., Buckingham, D. & Oshima-Takane, Y. (1994). A connectionist model of learning personal pronouns in English. In Hanson, S.J., Petche, T., Kearn, M., & Rivest, R.J. (Eds.), *Computational learning theory and natural learning systems, Vol. 2 : Intesection between theory and experiment* (pp. 347-362). Cambridge, MA: MIT Press.

Shultz, T.R. & Mareschal, D. (1997). Rethinking innateness, learning, and constructivism: Connectionist perspective on development. *Cognitive Development,* **12**, 563-586.

Shultz, T.R., Mareschal, D. & Schmidt, W.C. (1994). Modeling cognitive development on balance scale phenomena. *Machine Learning,* **16**, 57-86.

Shultz, T.R., Oshima-Takane, Y. & Takane, Y. (1995). Analysis of unstandardized contributions in cross connected networks. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 601-608). Cambridge, MA: MIT Press.

Shultz, T.R., Schmidt, W.C., Buckingham, D. & Mareschal, D. (1994). Modeling cognitive development with a generative connectionist algorithm. In Simon, T.J. & Halford, G.S. (Ed.), *Developing cognitive competence : New approaches to process modeling* (pp. 205-261). Hillsdale, NJ: Lawrence Earlbaum Associates.

Sirois, S. & Shultz, T.R. (1998). Neural network modeling of developmental effects in discrimination shifts. *Journal of Experimental Child Psychology,* in press.

Takane, Y. (1995). Nonlinear PCA by neural network models. *Proceedings of the Sixty-third Annual Meeting of the Japan Statistical Society,* 258-260.

Takane, Y. (1998). Nonlinear multivariate analysis by neural network models. In C. Hayashi, N. Ohsumi, K. Yajima, Y. Tanaka, H.H. Bock & Y. Baba (Eds.), *Data science, classification, and related methods: Studies in classification, data analysis, and knowledge organization* (pp. 527-538). Tokyo: Springer Verlag.

Takne, Y., Oshima-Takane, Y. & Shultz, T.R. (1994). Approximations of nonlinear functions by feed-forward neural networks. *Proceedings of the Eleventh Annual Meeting of the Japan Classification Society,* 26-33.

Takane, Y., Oshima-Takane, Y. & Shultz, T.R. (1995). Network analyses: The case of first and second person pronouns. *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics.* 3594-3599.

Takane, Y. & Shibayama, T. (1991). Principal component analysis with external information on both subjects and variables. *Psychometrika,* **56**, 97-120.

Tetewsky, S.J., Shultz, T.R. & Takane, T. (1995). Training regimens and function compatibility: Implications for understanding the effects of knowledge on concept learning. *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society* (pp. 304-309), Hillsdale, NJ : Lawrence Earlbaum Associates.

(Received February 1998, Revised August 1998)