A New Method for Inverting Nonlinear Multilayer Feedforward Networks

Bao-Liang Lu, Hajime Kita, and Yoshikazu Nishikawa

Dept. of Electrical Engineering, Kyoto University
Yoshida–Honmachi, Sakyo, Kyoto 606, Japan

Abstract: *A novel method for inverting a mapping of the multilayer feedforward network is proposed in this paper. This method is based upon recursive constrained linear equations which is constructed by a given desired output, weights, an activation function and unknown inputs. By solving these recursive constrained linear equations by nonlinear and linear programming techniques, we can obtain some typical network inversions from a given output, i.e., some unseen typical inputs corresponding to the given output. Therefore, we can obtain some new relations between the unseen inputs and the given output. The importance of this method lies in the fact that it provides us a promising approach to examining generalization capability of the network, extracting rules from the network, and realizing backward reasoning in an expert system based on the neural network. Two algorithms based on this method for inverting the mapping of the three-layer feedforward network are derived and some examples are presented.*

Introduction

In a multilayer neural network, once the network is trained with a set of training examples, the mapping from the input space to the output space is fixed, and therefore the generalization capability is determined. In general, this mapping is a many-to-one mapping, i.e., many inputs correspond to one output. In most cases, we only know part of relations between the training inputs and outputs. Hence, we don't know whether the trained network would have the capability to classify correctly untested inputs of the problem. Obviously, if all or some typical relations between the inputs and the outputs are brought to light, then the generalization capability is made more clear. It has been observed that, some method of network inversion would yield an effective tool for this task.[1]

In the last few years, some algorithms for inverting the mapping of the multilayer feedforward network have been proposed.[1–4] However, these algorithms have followed a same path which is based on the error back-propagation. They suffer from some deficiences such as follows:

- The inversion depends on a starting point in the input space, and only one network inversion can be found for a starting point.
- The analytical relations among elements of each network inversion are unable to be found.

The purpose of this paper is to present a new method for inverting the mapping of the multilayer feedforward networks. This method can overcome the above shortcomings.

One advantage of this method is that, to a given output, a set of typical unseen inputs corresponding to it can be found. By use of these typical unseen inputs, some new relations between the input space and the given output can be clarified. Another advantage of this method is that it provides us with the analytical relations among elements of each network inversion.

Two algorithms based on this method for inverting the mapping of the three-layer feedforward network are derived. The first algorithm, called IVG algorithm, is to invert the mapping into a global input space. The second one, called IVL algorithm, is developed to invert the mapping into a local input space.

Network Inversion Problem

Before we discuss the network inversion problem, let us adopt the following notation:

L	number of layers,
N_k	number of units in the layer k, $1 \le k \le L$,
x_{kj}	output of the jth unit in the layer k where x_{1j} represents the input of the jth unit,
b_{kj}	total input to the jth unit in the layer k,
d_j	desired output of the jth unit in the output layer,
w_{kij}	weight connecting the jth unit in the layer $(k-1)$ to the ith unit in the layer k,
$bias_{kj}$	bias weight of the jth unit in the layer k,
f	nonlinear activation function,
f^{-1}	inverse of the activation function,
$[\gamma_{kj}, \theta_{kj}]$	activation range of the jth unit in the layer k.

A multilayer feedforward network with L layers consists of the input layer, the output layer, and $L-2$ hidden layers. The units in the input layer serve only as distribution points, and they perform no input summation. Each unit in the layer k, $1<k<L$, receives its input from the layer $(k-1)$, and sents its output to the layer $(k+1)$.

In the multilayer feed-forward network, the inputs are fed forward through the connections according to

$$x_{ki} = f\left(\sum_{j=1}^{N_{k-1}} w_{kij}\, x_{k-1,j} + bias_{ki}\right), \quad k=2,3,\ldots,L, \; i=1,2,\ldots,N_k \qquad (1)$$

We regard the feedforward network as a mapping from an input space to an output space. In general, this mapping is a many-to-one mapping. It can be written as the following Eqs.(2)

$$\begin{cases} x_{Li} = f\left(\sum_{j=1}^{N_{L-1}} w_{Lij}\, x_{L-1,j} + bias_{Li} \right), & i=1,2,\dots,N_L \\ x_{L-1,j} = f\left(\sum_{l=1}^{N_{L-2}} w_{L-1,j,l}\, x_{L-2,l} + bias_{L-1,j} \right), & j=1,2,\dots,N_{L-1} \\ \cdot \\ \cdot \\ \cdot \\ x_{2n} = f\left(\sum_{m=1}^{N_1} w_{2nm}\, x_{1m} + bias_{2n} \right), & n=1,2,\dots,N_2 \\ \gamma_{pq} \le x_{pq} \le \theta_{pq}, & p=1,2,\dots,L,\ q=1,2,\dots,N_p \end{cases} \tag{2}$$

where $[\gamma_{pq}, \theta_{pq}]$ is the activation range of x_{pq}. Note that, for practical numerical computation, in order to protect arithmetic operation from overflow and attain an effective computing, the activation range selected is narrower than the theoretical one. For example, for a symmetric activation function of Eq.(23) shown later, the activation range is selected as [-0.499999, 0.499999] instead of [-0.5, 0.5].

In terms of matrix notation, we obtain

$$\begin{cases} \mathbf{x}_L = F_L\left(W_L\, \mathbf{x}_{L-1} + \mathbf{bias}_L \right) \\ \mathbf{x}_{L-1} = F_{L-1}\left(W_{L-1}\, \mathbf{x}_{L-2} + \mathbf{bias}_{L-1} \right) \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}_2 = F_2\left(W_2\, \mathbf{x}_1 + \mathbf{bias}_2 \right) \\ \Gamma_p \le \mathbf{x}_p \le \Theta_p, \quad p=1,2,\dots,L \end{cases} \tag{3}$$

where

$$\mathbf{x}_p = \begin{bmatrix} x_{p1} \\ x_{p2} \\ \cdot \\ \cdot \\ x_{pN_p} \end{bmatrix}, \Gamma_p = \begin{bmatrix} \gamma_{p1} \\ \gamma_{p2} \\ \cdot \\ \cdot \\ \gamma_{pN_p} \end{bmatrix}, \Theta_p = \begin{bmatrix} \theta_{p1} \\ \theta_{p2} \\ \cdot \\ \cdot \\ \theta_{pN_p} \end{bmatrix}, \quad p=1,2,\dots,L,$$

$$W_k = \begin{bmatrix} w_{k11} & w_{k12} & \cdots & w_{k,1,N_{k-1}} \\ w_{k21} & w_{k22} & \cdots & w_{k,2,N_{k-1}} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ w_{k,N_k,1} & w_{k,N_k,2} & \cdots & w_{k,N_k,N_{k-1}} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_{k1} \\ \mathbf{w}_{k2} \\ \cdot \\ \cdot \\ \mathbf{w}_{kN_k} \end{bmatrix},$$

$$F_k = \begin{bmatrix} f\left(\sum_{j=1}^{N_{k-1}} w_{k1j}\, x_{k-1,j} + bias_{k1}\right) \\ f\left(\sum_{j=1}^{N_{k-1}} w_{k2j}\, x_{k-1,j} + bias_{k2}\right) \\ \cdot \\ \cdot \\ \cdot \\ f\left(\sum_{j=1}^{N_{k-1}} w_{k,N_k,j}\, x_{k-1,j} + bias_{kN_k}\right) \end{bmatrix} = \begin{bmatrix} f(\mathbf{w}_{k1}\, \mathbf{x}_{k-1} + bias_{k1}) \\ f(\mathbf{w}_{k2}\, \mathbf{x}_{k-1} + bias_{k2}) \\ \cdot \\ \cdot \\ f(\mathbf{w}_{kN_k}\, \mathbf{x}_{k-1} + bias_{kN_k}) \end{bmatrix},$$

$$\mathbf{bias}_k = [bias_{k1}, bias_{k2}, \dots, bias_{kN_k}]^T, \quad k=2,3,\dots,L$$

From Eqs. (3), by use of the inverse of the activation function, we obtain the following recursive constrained linear equations

$$\begin{cases} W_L\, \mathbf{x}_{L-1} = \mathbf{b}_L \\ W_{L-1}\mathbf{x}_{L-2} = \mathbf{b}_{L-1} \\ \cdot \\ \cdot \\ \cdot \\ W_2\, \mathbf{x}_1 = \mathbf{b}_2 \\ \Gamma_p \le \mathbf{x}_p \le \Theta_p, \quad p=1,2,\dots,L-1 \end{cases} \tag{4}$$

where

$\mathbf{b}_k = F^{-1}(\mathbf{x}_k) - \mathbf{bias}_k, \quad F^{-1} = [f^{-1}(x_{k1}), f^{-1}(x_{k2}), \dots, f^{-1}(x_{kN_k})]^T,$

$k=2,3,\dots,L.$

The importance of Eqs. (4) lies in the fact that it provides us with an analytical formula for finding network inversions and indicates the following property of the multilayer feedforward network:

- The relations among elements, $x_{k1}, x_{k2}, \dots, x_{kN_k}$, of each $\mathbf{x}_k (k=1,2,\dots,L)$ are linear.

From Eqs. (4), the difficult nonlinear network inversion problem in the multilayer feedforward network can be partitioned into some subproblems:

- Compute $\mathbf{b}_k$ from $\mathbf{x}_k$ by

$$\mathbf{b}_k = F^{-1}(\mathbf{x}_k) - \mathbf{bias}_k, \quad k=L, L-1, \dots, 3, 2$$

- Solve the constrained linear equation

$$\begin{cases} W_k\, \mathbf{x}_{k-1} = \mathbf{b}_k \\ \Gamma_{k-1} \le \mathbf{x}_{k-1} \le \Theta_{k-1} \end{cases} \tag{5}$$

Thus, by calculating $\mathbf{b}_k$ and finding the solution of Eq.(5) recursively from the output layer to the input layer, we can solve the network inversion problem in the multilayer feedforward network.

Inverting Algorithms

In this section, limiting to the case of three-layer network, we shall derive two algorithms for inverting the mapping, and present a computing procedure for determining boundary of the network inversions.

The numbers of units in the input, the hidden and the output layers differ for variety of application problems. Here we consider a typical case in which $N_1 > N_2 > N_3$, where N_1, N_2, and N_3 are numbers of the input, the hidden and the output units, respectively.

Inverting into a Global Input Space

Suppose that a desired output $\mathbf{d} = [d_1, d_2, \dots, d_{N_3}]^T$ is given. Since $\mathbf{x}_3 = \mathbf{d}$, the following equations are obtained from Eq.(4):

$$\mathbf{b}_3 = F^{-1}(\mathbf{d}) - \mathbf{bias}_3 \tag{6}$$

$$\begin{cases} W_3\, \mathbf{x}_2 = \mathbf{b}_3 \\ \Gamma_2 \le \mathbf{x}_2 \le \Theta_2 \end{cases} \tag{7}$$

$$\mathbf{b}_2 = F^{-1}(\mathbf{x}_2) - \mathbf{bias}_2 \tag{8}$$

$$\begin{cases} W_2\, \mathbf{x}_1 = \mathbf{b}_2 \\ \Gamma_1 \le \mathbf{x}_1 \le \Theta_1 \end{cases} \tag{9}$$

The objective of the inverting algorithms is to find a set of typical $\mathbf{x}_1$ from the above equations as large as possible.

If Eq.(7) have an infinity of solutions, there exist an infinity of $\mathbf{b}_2$. The problem is how to select $\mathbf{b}_2$ from Eqs.(7) and (8) to make Eq.(9) consistent. In general, the number of $\bar{\mathbf{b}}_2$ which make Eq. (9) consistent is infinity. Obviously, we can not find all the $\bar{\mathbf{b}}_2$ in practical computation. We propose a computing procedure, called IVG(InVerting into Global input space) algorithm, to deal with this problem. The principal idea is to use a nonlinear programming technique to select typical $\mathbf{b}_2$.

The IVG algorithm proceeds as follows:

1) Suppose that $b_{21}, b_{22}, \ldots, b_{2N_2}$ are variable.

2) Compute W_2^-, a generalized inverse of W_2.

3) Compute $I_{N_1}-W_2^- W_2$, and obtain the general solution[5] of $W_2\mathbf{x}_1=\mathbf{b}_2$ by

$$\mathbf{x}_1=W_2^-\mathbf{b}_2+(I_{N_1}-W_2^- W_2)\mathbf{s},\quad I_{N_1}\in\mathbf{R}^{N_1\times N_1} \tag{10}$$

where I_{N_1} is the N_1 dimensional identity matrix, $\mathbf{s}$ is an arbitrary column vector, $\mathbf{s}\in\mathbf{R}^{N_1}$.

Let $\mathbf{b}_2$ and $\mathbf{s}$ be denoted by $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively. Then, Eq.(10) is rewritten as

$$\mathbf{x}_1=[W_2^- \quad (I_{N_1}-W_2^- W_2)]\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix} \tag{11}$$

4) Solve the following nonlinear programming problem:

$$\text{Maximize}(\mathbf{a}_i\mathbf{y}) \text{ or } \text{Minimize}(\mathbf{a}_i\mathbf{y}),\ i=1,2,\ldots,N_1 \tag{12}$$

Subject to

$$\begin{cases} W_3F_2(\mathbf{y}_1)=\mathbf{b}_3\\ \Gamma_1\le A\mathbf{y}\le\Theta_1\\ \Psi_2\le\mathbf{y}_1\le\Omega_2\end{cases} \tag{13}$$

where

$$F_2(\mathbf{y}_1)=[f(y_{11}+bias_{21}),\ f(y_{12}+bias_{22}),\ldots,\ f(y_{1N_2}+bias_{2N_2})]^T,$$

$$y_{1i}=b_{2i},\ i=1,2,\ldots,N_2,$$

$$A=[W_2^- \quad (I_{N_1}-W_2^- W_2)]=[\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_{N_1}]^T,\quad A\in\mathbf{R}^{N_1\times(N_2+N_1)},$$

$$\mathbf{y}=\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix},\quad \mathbf{y}\in\mathbf{R}^{N_2+N_1},$$

$$\Omega_2=[\omega_{21},\omega_{22},\ldots,\omega_{2N_2}]^T$$
$$=[f^{-1}(\theta_{21})-bias_{21},\ f^{-1}(\theta_{22})-bais_{22},\ldots,\ f^{-1}(\theta_{2N_2})-bias_{2N_2}]^T,$$

$$\Psi_2=[\psi_{21},\psi_{22},\ldots,\psi_{2N_2}]^T$$
$$=[f^{-1}(\gamma_{21})-bias_{21},\ f^{-1}(\gamma_{22})-bias_{22},\ldots,\ f^{-1}(\gamma_{2N_2})-bias_{2N_2}]^T$$

Here we suppose that the inverse of activation function is monotone increasing in the activation range.

5) If the above nonlinear programming(NLP) problem has an optimum or a feasible solution, there exits such a $\bar{\mathbf{b}}_2$ that makes Eq.(9) consistent,

$$\bar{\mathbf{b}}_2=\mathbf{y}_1^*,\ \mathbf{y}^*=[y_{11}^*\ y_{12}^*,\ldots,\ y_{1N_2}^*,\ y_{21}^*,\ y_{22}^*,\ldots,\ y_{2N_1}^*]^T$$

where $\mathbf{y}^*$ is the optimum or a feasible solution of the NLP problem. From $\mathbf{y}^*$, a network inversion corresponding to the given output $\mathbf{d}$ is obtained by

$$\bar{\mathbf{x}}_1=[W_2^- \quad (I_{N_1}-W_2^- W_2)]\begin{bmatrix}\mathbf{y}_1^*\\ \mathbf{y}_2^*\end{bmatrix} \tag{14}$$

6) If the NLP problem has no solution, it indicates that Eq. (9) has no solution corresponding to the given output $\mathbf{d}$.

7) Go to 4), to select another $\bar{\mathbf{b}}_2$.

The algorithm continues until all the maximum and the minimum solutions of the objective function $\mathbf{a}_i\mathbf{y}$, $i=1,2,\ldots,N_1$, are obtained.

The IVG algorithm has two features as follows:

- Besides the parameters of the network, only a given output is needed for selecting $\bar{\mathbf{b}}_2$ from Eqs.(7) and (8).
- A set of $\bar{\mathbf{b}}_2$, i.e., $\{\mathbf{b}_2\}_G$, can be selected. From these $\bar{\mathbf{b}}_2$, $\bar{\mathbf{b}}_2\in\{\mathbf{b}_2\}_G$, a set of typical network inversions can be obtained, and the range of each element of network inversions corresponding to the given output is known. With this range, the desired network inversions can be searched effectively in the whole input space.

It should be noted that if $\mathbf{y}^*$ is a feasible solution instead of a minimum or a maximum solution of the NLP problem, the range of network inversions corresponding to the given output is unable to be obtained.

Inverting into a Local Input Space

From the above discussion, we see that, for a given output, if we expect to obtain some typical network inversions of which $\bar{x}_{1i}$, $i=1,2,\ldots,N_1$, is minimum or maximum in all the network inversions, we can use the IVG algorithm to select a set of $\bar{\mathbf{b}}_2$. However, in some cases, for a given output, associated $\bar{\mathbf{x}}_2$, $\bar{\mathbf{b}}_2$ and $\bar{\mathbf{x}}_1$ are known. For example, for each of training example pairs, $<\bar{\mathbf{x}}_1, \mathbf{d}>$, associated $\bar{\mathbf{x}}_2$ and $\bar{\mathbf{b}}_2$ are known. Then, the problem is that, for the given output, and the associated $\bar{\mathbf{x}}_2$ and $\bar{\mathbf{b}}_2$, we want to find other $\mathbf{x}_1$ as many as possible besides $\bar{\mathbf{x}}_1$. Here we propose a computing procedure, called IVL(InVerting into Local input space) algorithm, to solve this problem.

In general, corresponding to a given output, there exist an infinity of network inversions. For $N_1>N_2>N_3$, in the neighborhood of $\bar{\mathbf{x}}_2$, there exits a set of $\mathbf{x}_2$, $\{\mathbf{x}_2\}_H$, which makes Eq. (9) consistent, where $\{\mathbf{x}_2\}_H\subset\{\mathbf{x}_2\}$, and

$$\{\mathbf{x}_2\}=\{\mathbf{x}_2 \mid W_3\mathbf{x}_2=\mathbf{b}_3,\ \Gamma_2\le\mathbf{x}_2\le\Theta_2,\ \mathbf{b}_3=F^{-1}(\mathbf{d})-\mathbf{bias}_3\} \tag{15}$$

Suppose $rank(W_3)<N_2$, then the general solution of Eq.(7) contains r, $r=N_2-rank(W_3)$, arbitrary parameters. The general solution of Eq.(7) can be written as

$$\begin{cases}\mathbf{x}_2=\mathbf{x}_{2h}+\mathbf{x}_{2p}\\ \Gamma_2\le\mathbf{x}_2\le\Theta_2\end{cases} \tag{16}$$

where $\mathbf{x}_{2h}$ is the general solution of the homogeneous equation of Eq.(7) and therefore contains no constant terms, and $\mathbf{x}_{2p}$ is any particular solution of Eq.(7) and therefore is independent of parameters.

When a $\bar{\mathbf{x}}_2$ is given, how can we find the neighborhood of $\bar{\mathbf{x}}_2$, i.e., $\{\mathbf{x}_2\}_H$, which makes Eq.(9) consistent. In the IVL algorithm, a direct search strategy is used to search the elements of $\{\mathbf{x}_2\}_H$. The direct search strategy operates in the following manner: The given $\bar{\mathbf{x}}_2$ is set as a starting point, then one independent variable is changed at a time while keeping all the others unchanged until the search for this variable is completed. Each independent variable is changed by turns in the similar way.

The principal steps in the IVL algorithm are as follows:

1) Let $x_{2t1}, x_{2t2}, \ldots, x_{2tr}$, $1 \le ti \le N_2$, $i=1,2,..,r$, correspond to r arbitrary parameters of Eq.(20), and put $i=1$.

2) Let $\mathbf{x}_2^0 = \overline{\mathbf{x}}_2$ and $\Delta x_{2ti} = -\Delta x$, $i=1,2,\ldots,r$, Δx is a selected step size.

3) Let $x_{2ti} = x_{2ti}^0 + \Delta x_{2ti}$ and $x_{2tj} = x_{2tj}^0$, $j=1,2,\ldots,r$, $j \ne i$, and compute x_{2k} according to Eq. (16), $k=1,2,\ldots,N_2$, $k \ne ti$, $i=1,2,\ldots,r$.

4) From $\mathbf{x}_2$, obtain $\mathbf{b}_2$ by

$$\mathbf{b}_2 = F^{-1}(\mathbf{x}_2) - \mathbf{bias}_2$$

5) Solve the linear programming problem

$$\text{Minimize } x_{1i} \text{ or Maximize } x_{1i}, \quad i=1,2,\ldots,N_1 \tag{17}$$

Subject to

$$\begin{cases} W_2 \mathbf{x}_1 = \mathbf{b}_2 \\ \Gamma_1 \le \mathbf{x}_1 \le \Theta_1 \end{cases} \tag{18}$$

6) If Eq.(17) has an optimum or a feasible solution, $\mathbf{x}_2$, i.e., $\mathbf{b}_2$, makes Eq.(9) consistent. Let $\mathbf{x}_2^0 = \mathbf{x}_2$, and go back to 3) to search another $\mathbf{x}_2$.

7) If Eq.(17) has no solution, it indicates that, corresponding to $\mathbf{x}_2$, Eq.(9) is inconsistent. If $\Delta x_{2ti} < 0$, let $\mathbf{x}^0 = \overline{\mathbf{x}}_2$ and $\Delta x_{2ti} = \Delta x$, go back to 3). Otherwise, the search for x_{2ti} is completed. Set $i=i+1$, and go back to 3) until all the independent variables have been changed to complete the search.

Note that, in order to simplify the description, we use a direct search strategy in the IVL algorithm. For practical computation, we can replace the direct search strategy by some more effective one to improve the performance of the search.

Comparing the IVL algorithm with the IVG algorithm, we see the following features of the IVL algorithm:

- The procedure for selecting $\overline{\mathbf{b}}_2$ is simpler than that in the IVG algorithm, because $\overline{\mathbf{b}}_2$ is obtained by solving the linear programming problem.
- The IVL algorithm can complement the IVG algorithm for finding the neighborhoods of $\overline{\mathbf{x}}_2$, i.e., $\{\mathbf{x}_2\}_H$. Therefore, it is possible to find the network inversions as many as possible.

Boundary of the Network Inversions

For a given output, if the corresponding network inversions exist, by use of the IVG and the IVL algorithms, we can obtain a set of $\overline{\mathbf{b}}_2$, $\{\mathbf{b}_2\}_{IV}$. For each $\overline{\mathbf{b}}_2$, Eq.(9) may have an infinity of solutions (network inversions) or a unique solution. If Eq. (9) has an infinity of solutions, we can not obtain all the solutions in detail. In this case, we expect to know the boundary of the solutions, i.e., the boundary of the network inversions. The following procedure, called DBI(Determining the Boundary of the network Inversions) algorithm, will meet our expectation.

For each $\overline{\mathbf{b}}_2$, $\overline{\mathbf{b}}_2 \in \{\mathbf{b}_2\}_{IV}$, depending upon the activation range Γ_1 and Θ_1, we can determine the boundary of the solutions as follows:

1) Solve the linear programming problem

$$\text{Minimize } x_{1i}, \quad i=1,2,\ldots,N_1 \tag{19}$$

Subject to

$$\begin{cases} W_2 \mathbf{x}_1 = \mathbf{b}_2 \\ \Gamma_1 \le \mathbf{x}_1 \le \Theta_1 \end{cases} \tag{20}$$

From the minimum solution $\mathbf{x}_1^* = [x_{11}^*, x_{12}^*, \ldots, x_{1N_1}^*]^T$ of the above problem, λ_{1i}, the minimum value of x_{1i}, on the boundary of the solutions is obtained: $\lambda_{1i} = x_{1i}^*$.

2) Solve the linear programming problem

$$\text{Maximize } x_{1i}, \quad i=1,2,\ldots,N_1 \tag{21}$$

Subject to

$$\begin{cases} W_2 \mathbf{x}_1 = \mathbf{b}_2 \\ \Gamma_1 \le \mathbf{x}_1 \le \Theta_1 \end{cases}$$

From the maximum solution $\mathbf{x}_1^* = [x_{11}^*, x_{12}, \ldots, x_{1N_1}^*]^T$ of the problem, ϕ_{1i}, the maximum value of x_{1i}, on the boundary of the solutions is obtained: $\phi_{1i} = x_{1i}^*$.

Repeating 1) and 2) until all of λ_{1i} and ϕ_{1i}, $i=1,2,\ldots,N_1$, are obtained, we can obtain the boundary of the network inversions as follows:

$$\begin{cases} W_2 \mathbf{x}_1 = \mathbf{b}_2 \\ \Lambda_1 \le \mathbf{x}_1 \le \Phi_1 \end{cases} \tag{22}$$

where, $\Lambda_1 = [\lambda_{11}, \lambda_{12}, \ldots, \lambda_{1N_1}]^T$, $\Phi_1 = [\phi_{11}, \phi_{12}, \ldots, \phi_{1N_1}]^T$.

Obviously, if Eq. (9) has a unique solution, Λ_1 is identical with Φ_1, i.e., $\Lambda_1 = \Phi_1$.

Examples

We now present some examples to illustrate the IVG, the IVL, and the DBI algorithms.

In our examples, we use the well known back-propagation algorithm[6] to train the network. The activation function is a symmetric sigmoid function,[7] i.e.,

$$x_{pq} = \frac{1}{1+\exp\{-(\sum_{j=1}^{N_{p-1}} w_{pqj}\, x_{p-1,j} + bias_{pq})\}} - \frac{1}{2} \tag{23}$$

where $p=2,3$, and $q=1,2,\ldots,N_p$. For numerical computation, the activation range is selected as $[-0.499999, 0.499999]$. That is, for $k=1,2,3$, $\Gamma_k = -\Theta_k$, $\Theta_k = [0.499999, 0.499999, \ldots, 0.499999]^T$.

We treat the "two-or-more clumps" problem.[8] The desired output is 0.5 if there are two or more clumps of 0.5's in the input and −0.5 otherwise. We use a three-layer network, with 5 input units and one output unit. The hidden layer contains 3 units, which are fully connected to the input and the output layers. The training inputs, desired outputs, and the actual outputs are shown in Table 1.

The parameters of the network after training are as follows:

$$W_2 = \begin{bmatrix} 0.757632 & 0.301185 & 0.866417 & 2.731681 & 4.010859 \\ 0.359501 & 0.548825 & 0.304895 & 2.091348 & 4.418867 \\ 0.335742 & -0.152064 & 0.379951 & 1.299941 & 5.575029 \end{bmatrix} \tag{24}$$

$$W_3 = [4.227584 \;\; 4.870978 \;\; 7.014191] \tag{25}$$

$$\mathbf{bias}_2 = \begin{bmatrix} 1.536189 \\ 1.065805 \\ 0.786672 \end{bmatrix}, \quad \mathbf{bias}_3 = [-0.047802] \tag{26}$$

Table 1 Training Examples and Actual Outputs

No.	Training Inputs					Desired Output	Actual Output
	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	d_1	x_{31}
1	−	−	−	−	−	−	-0.499273
2	−	−	+	−	−	−	-0.498758
3	−	−	+	−	+	+	0.498634
4	−	+	−	−	−	−	-0.499098
5	−	+	−	−	+	+	0.497997
6	−	+	+	−	−	−	-0.498275
7	−	+	+	−	+	+	0.498990
8	+	−	−	−	−	−	-0.498818
9	+	−	−	−	+	+	0.498589
10	+	−	+	−	+	+	0.499167
11	+	+	−	−	−	−	-0.498357
12	+	+	+	−	+	+	0.499318

Note that − and + represent -0.5 and +0.5 respectively.

In the following numerical experiments, we suppose that the given output d is $[0.498634]$. The known input corresponding to 0.498634 is $[-0.5, -0.5, 0.5, -0.5, 0.5]$ as shown in the row No.3 of Table 1.

Inverting into the Global Input Space

For the given output $d=[0.498634]$, b_3 is obtained as, $b_3=6.642304$ from Eq.(6). Then, $W_3 x_2 = b_3$ is written as

$$4.227584x_{21}+4.870978x_{22}+7.014191x_{23}=6.642304$$

From Eq.(24), W_2^-, a generalized inverse of W_2, and $I_{N_1}-W_2^- W_2$ are obtained to be

$$W_2^- = \begin{bmatrix} 0.446109 & -0.567151 & 0.125170 \\ -0.608144 & 1.285744 & -0.587999 \\ 0.682845 & -0.955666 & 0.260403 \\ 0.342056 & 0.184567 & -0.388010 \\ -0.169749 & 0.091320 & 0.228521 \end{bmatrix} \tag{27}$$

$$(I_{N_1}-W_2^- W_2)= \begin{bmatrix} 0.823880 & 0.195939 & -0.261211 & -0.195233 & 0.019053 \\ 0.195939 & 0.388102 & 0.358429 & -0.263319 & 0.035757 \\ -0.261211 & 0.358429 & 0.600713 & -0.205193 & 0.032413 \\ -0.195233 & -0.263319 & -0.205193 & 0.184009 & -0.024346 \\ 0.019053 & 0.035757 & 0.032413 & -0.024346 & 0.003296 \end{bmatrix} \tag{28}$$

From Γ_2, Θ_2, $bias_2$, and the inverse of activation function, Ψ_2 and Ω_2 are obtained as

$$\Psi_2=\begin{bmatrix} -15.351700 \\ -14.881316 \\ -14.602183 \end{bmatrix}, \ \Omega_2=\begin{bmatrix} 12.279322 \\ 12.749706 \\ 13.028839 \end{bmatrix} \tag{29}$$

Let $y_1=b_2$ and $y_2=s$, then $W_3F_2(y_1)=b_3$ becomes

$$\frac{4.227584}{1+e^{-(y_{11}+1.536189)}}+\frac{4.870978}{1+e^{-(y_{12}+1.065805)}}+\frac{7.014191}{1+e^{-(y_{13}+0.786672)}} = 14.69868 \tag{30}$$

By use of the revised Powell's method, we solve the nonlinear programming problem of Eqs.(12) and (13) with the above data, and obtain the minimum and the maximum solutions of each objective function $a_i y$, $i=1,2,...,5$, as shown in Table 2 and Table 3. From y_1^* and y_2^*, i.e., $\bar{b}_2$ and $\bar{s}$, we obtain a set of network inversions, as shown in Table 4. From Table 4, we see that these network inversions possess the following character:

If the inversion x_1 is computed from the minimum (maximum) solution of the objective function $a_i y$, x_{1i} is minimum (maximum) in the whole input space corresponding to the given output.

Table 2 A Set of y_1^*, i.e., $\{\bar{b}_2\}_G$, Selected from the Global Input Space

No.	y_{11}^*	y_{12}^*	y_{13}^*
Min (a_1 y)	1.232224	1.111698	1.464073
Max (a_1 y)	1.429272	0.930166	1.541359
Min (a_2 y)	1.391840	0.954746	1.530420
Max (a_2 y)	1.193994	1.355389	1.329803
Min (a_3 y)	1.140926	1.256493	1.409663
Max (a_3 y)	1.403899	0.948670	1.539202
Min (a_4 y)	0.919280	0.891100	1.815839
Max (a_4 y)	1.621460	1.208073	1.280225
Min (a_5 y)	2.347908	1.673666	0.958825
Max (a_5 y)	0.380774	1.135954	1.989710

Table 3 A Set of y_2^*, i.e., $\{\bar{s}\}_G$, Selected from the Global Input Space

No.	y_{21}^*	y_{22}^*	y_{23}^*	y_{24}^*	y_{25}^*
Min (a_1 y)	-0.602464	-0.143281	0.191011	0.142765	-0.013933
Max (a_1 y)	0.196999	0.079566	-0.022171	-0.067459	0.007547
Min (a_2 y)	0.009474	0.018766	0.017331	-0.012732	0.001729
Max (a_2 y)	0.133970	0.265358	0.245069	-0.180040	0.024448
Min (a_3 y)	0.193661	-0.265738	-0.445367	0.152130	-0.024030
Max (a_3 y)	0.001587	0.039076	0.047526	-0.024953	0.003575
Min (a_4 y)	0.362780	0.382894	0.250254	-0.274347	0.035512
Max (a_4 y)	-0.247255	-0.311595	-0.232915	0.219138	-0.028833
Min (a_5 y)	0.387798	0.178388	0.405260	-0.126183	0.448785
Max (a_5 y)	0.412297	-0.034961	0.010439	0.029739	-0.003318

Table 4 Network Inversions and Their Actual Outputs

No.	Network Inversion in the Global Input Space					Output
	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{31}
x_{11}^{Min}	-0.499999	-0.324164	0.351266	0.201362	0.212990	0.498634
x_{11}^{Max}	0.499999	-0.499999	0.466246	-0.004953	0.202106	0.498636
x_{12}^{Min}	0.280465	-0.499999	0.453849	0.045751	0.202387	0.498634
x_{12}^{Max}	0.064363	0.499999	0.111367	-0.037444	0.249431	0.498639
x_{13}^{Min}	0.166466	-0.172936	-0.499999	0.227333	0.219179	0.498641
x_{13}^{Max}	0.282503	-0.499999	0.499999	0.033126	0.203638	0.498640
x_{14}^{Min}	0.494781	-0.098145	0.499235	-0.499999	0.375797	0.498625
x_{14}^{Max}	-0.048819	-0.497174	0.053151	0.499999	0.098806	0.498633
x_{15}^{Min}	0.499999	0.499999	0.499999	0.499999	0.004851	0.498621
x_{15}^{Max}	0.098899	0.122043	-0.427623	-0.499999	0.499999	0.498601

Inverting into the Local Input Space

Suppose that a given $\bar{b}_2$ is [1.232224, 1.111698, 1.464073] as shown in the first row of Table2. Then, the resulting $\bar{x}_2$ is [0.440945, 0.398211, 0.404715].

According to the IVL algorithm, we obtain some neighborhoods of $\bar{x}_2$, i.e., $\{x_2\}_H$, as shown in Table 5. Here, the step size Δx is selected as 0.02.

Boundary of the Network Inversions

For each of x_2 in Table 5, which makes Eq.(9) consistent, by use of the DBI algorithm, we obtain the minimum and the maximum values of x_{15} as shown in Table 6. For example, corresponding to $x_2=[0.420945,$

0.398211, 0.416734], the minimum and the maximum values of x_{15} in all the network inversions are 0.274614 and 0.367117, respectively.

Table 5 Neighborhoods of $\overline{\mathbf{x}}_2$, i.e., $\{\mathbf{x}_2\}_H$

No.	Independent variables		Dependent variable	Is Eq. (9) consistent?
	x_{21}	x_{22}	x_{23}	
1	0.420945	0.398211	0.416734	Yes
2	0.400945	0.398211	0.428789	Yes
3	0.380945	0.398211	0.440843	Yes
*	0.360945	0.398211	0.452897	No
4	0.460945	0.398211	0.392625	Yes
*	0.480945	0.398211	0.380571	No
5	0.440945	0.378211	0.418569	Yes
*	0.440945	0.358211	0.432458	No
6	0.440945	0.418211	0.390791	Yes
*	0.440945	0.438211	0.376902	No

Table 6 Network Inversions and Their Outputs

No.	Network Inversion in the Local Input Space					Output
	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{31}
1_{Min}	-0.499999	-0.499999	-0.241800	0.203737	0.274614	0.498634
1_{Max}	0.301973	0.499571	0.499999	-0.499999	0.367117	0.498634
2_{Min}	-0.499999	-0.499999	-0.483721	0.077980	0.350818	0.498634
2_{Max}	-0.323584	0.386195	0.499999	-0.499999	0.432092	0.498634
3_{Min}	-0.499999	-0.398571	-0.499999	-0.134133	0.439731	0.498634
3_{Max}	-0.499999	0.177507	0.209424	-0.499999	0.492406	0.498634
4_{Min}	-0.116145	-0.499999	0.499999	0.377282	0.110084	0.498634
4_{Max}	0.499999	-0.194834	0.499999	0.130529	0.138838	0.498634
5_{Min}	0.155850	-0.499999	0.499999	-0.064969	0.251571	0.498634
5_{Max}	0.499999	-0.329548	0.499999	-0.202794	0.267632	0.498634
6_{Min}	-0.394146	-0.268077	-0.499999	0.499999	0.169283	0.498634
6_{Max}	0.499999	0.499999	-0.099503	-0.064636	0.240747	0.498634

Comparing the Table 1 with the Table 6, we see that the network trained with the examples shown in Table 1 may generalize mistakenly on some novel inputs. For example, in Table 6, the inputs at the rows of 1_{Min}, 2_{Min}, 3_{Min}, 4_{Min}, and 6_{Min} generate the same output, 0.498634. Obviously, from the definition of the "two-or-more clumps" problem, the network generalize mistakenly on these novel inputs.

From the preceding examples, it may be clear that it is possible to obtain some typical network inversions from a given output by use of the IVG and the IVL algorithms, and determine the boundary of the network inversions by the DBI algorithm.

Discussion

We have proposed a novel method for inverting the mapping of the multilayer feedforward network, and have derived two inverting algorithms based on this method. The results of the numerical experiments indicate that our inverting algorithms can overcome the shortcomings of the existing inverting algorithms. We have shown that, combining the IVG algorithm and the IVL algorithm, we can obtain a set of typical network inversions associated with a given output. The relations between these network inversions and the given output outline the mapping from the input space to the given output. Hence, by use of these network inversions, we can examine generalization capability, extract rules from networks and realize backward reasoning in expert systems based on the neural network.

The present results show that the multilayer feedforward network may generalize mistakenly on some novel inputs even for a simple problem. Therefore, when we apply a neural network to a practical problem, especially in case of real-time problem, we have to examine carefully generalization capability of the network.

Let us consider the following question: Can all feedforward multilayer networks be inverted ? The answer seems to depend upon what activation function is being used. If the symmetric sigmoid function, as shown in Eq. (23), is being used, then we should take care of saturating nonlinearity of the activation function. In other words, there is no limit of input magnitude to the hidden and the output units. In such situations, numerical computations for inversion would become almost impossible. Thus we have to put a certain limit on the input magnitude to every unit.

As to applicability of the present method to recurrent networks, the authors intend to develop their study in future work.

Acknowledgement

The authors are thankful to Dr. T. Tezuka of Kyoto University for his useful discussions and suggestions.

References

[1] J. Kindermann and A. Linden, "Inversion of neural networks by gradient decent", *Parallel Computing,* vol. 14, 1990.

[2] R. J. Williams, "Inverting a Connectionist Network Mapping by Backpropagation of Error", *Proc. of 8th Annual Conference of the Cognitive Science Society,* Lawrence-Erlbaum, 1986.

[3] A. Linden and J. Kindermann, "Inversion of Multilayer Nets", *Proc. of First International Joint Conference on Neural Networks,* Washington, 1989.

[4] S. Thrun and A. Linden, "Inversion in time", *Proc. of EURASIP Workshop Neural Networks,* Sesimbra, Portugal, 1990.

[5] B. Stephen, "Matrices Methods and Applications", Clarendon Press, Oxford, 1990.

[6] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing,* Vol. 1, MIT Press, 1986.

[7] W.S. Stornetta and B.A. Huberman, "An improved three-layer backpropagation algorithm", *Proc. of IEEE First International Conference on Neural Networks,* San Diego, 1987.

[8] T. Maxwell, C. L. Giles and Y. C. Lee, "Generalization in neural networks: the contiguity problem", *Proc. of IEEE First International Conference on Neural Networks,* San Diego, 1987.