91 KACC 1991. 10. 22~24

Construction of Coordinate Transformation Map Using Neural Network

WONCHANG LEE AND KWANGHEE NAM

Department of Electrical Engineering
POSTECH, Pohang, P.O.Box 125, 790-600

Abstract

In general, it is not easy to find the linearizing coordinate transformation map for a class of sysetms which are state equivalent to linear systems, because it is required to solve a set of partial differential equations. It is possible to construct an arbitrary nonlinear fuction with a backpropagation (BP) net. Uitilizing this property of BP neural net, we construct a desired linearizing coordinate transformation map. That is, we implement a unknown coordinate transformation map through the training of neural weights. We have shown an example which supports this idea.

1. Introduction

Linearization of nonlnear systems has been one of the most interesting research topics in nonlinear system analysis. There has been considerable development in the study of continuous-time sytems [1],[2],[3]. In the case of a single input continuous-time affine system $\dot{x} = f(x) + ug(x)$, the characterization of linearizabibity is given by the relationship between vector fields f and g. It is necessary to find the coordinate transformation map in the linear equivalence of nonlinear systems. However, a necessary and sufficient condition for the existence of the corresponding coordinate transformation map is given but it is not easy to find it since a set of partial differential equations must be solved. With a multilayered BP neural net it is possible to construct an arbitrary nonlinear function [4]. In other words, BP neural net has the power of general mapping implementation. Therefore, it is meaningful to construct the linearizing coordinate transformation map using BP neural net.

2. Preliminaries and Problem Statement

Let M is a smooth n-dimensional manifold. By smooth, we mean infinite differentiability. We denote by $\mathbf{R}$ the real line. Since the problems addressed in this paper are local in nature, we identify manifold M with an open neighborhood of the origin in $\mathbb{R}^n$.

Consider a single input nonlinear system

$$\dot{x} = f(x) + ug(x), \tag{1}$$

where $x \in M$, f, g are vector field and u is a scalar input from a suitable function space. We assume that there exists $x_e \in M$ such that $f(x_e) = 0$.

Definition. The nonlinear system (1) is said to be *(locally) state equivalent to a linear system* if there exists a C^∞ diffeomorphism $T : M \to \mathbf{R}^n$ which transforms the system (1) to a linear controllable system

$$\dot{z} = Az + bu, \tag{2}$$

where

$$A = \begin{bmatrix} a_n & 1 & 0 & \cdots & 0 \\ a_{n-1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & 0 & 0 & \cdots & 1 \\ a_1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

in $Im(T)$. If T is onto $\mathbb{R}^n$, we say that (1) is *globally state equivalent to a linear system.*

A necessary and sufficient condition for system (1) to be state equivalent to a linear controllable system (2) is given in the following theorem.

Theorem 1. *[1] System (1) is locally state equivalent to a linear controllable system (2) if and only if in an open neighborhood of* x_e

(i) $\{g, \mathrm{ad}_f g, \cdots, \mathrm{ad}_f^{n-1} g\}$ *are linearly independent,*

(ii) $[\mathrm{ad}_f^i g,\ \mathrm{ad}_f^j g] = 0, \qquad 0 \le i, j \le n.$

Corollary 1. *System (1) is globally state equivalent to a linear controllable system (2) if and only if*

(i) $\{g, \mathrm{ad}_f g, \cdots, \mathrm{ad}_f^{n-1} g\}$ *are linearly independent,*

(ii) $[\mathrm{ad}_f^i g,\ \mathrm{ad}_f^j g] = 0, \qquad 0 \le i, j \le n,$

(iii) $\{g, \mathrm{ad}_f g, \cdots, \mathrm{ad}_f^{n-1} g\}$ *are complete,*

(iv) M is simply-connected.

A necessary and sufficient condition for the existence of coordinate transformation map is given in Theorem 1

but in general it is not easy to obtain the corresponding coordinate transformation map since it is required to solve a set of partial differential equations. Therefore, this problem is closely related to the general mapping implementation problem. It is known that some neural networks are capable of solving this mapping implementation problem, and the capabibilities offered by these networks are substantially different from those offered by regression approaches. Thus, in this paper, since BP neural net is one of those networks, we construct the coordiante transformation map which transforms system (1) to the linear controllable system (2) using it.

3. Construction of Neural Network

BP neural net is the most popular one among many neural network paradigms that satisfy the requirements for the general mapping implementation problem.

It is well known that the number of nodes is one of the most critical factors in determining the training time of BP neural network, i.e., training time grows exponentially as the number of nodes increase. Hence for the effective use of the neural network, it is reasonable to bound the input nodes, as well as the nodes in the hidden layers. In our experiment we choose three output nodes. We construct a BP neural net as shown in Figure 1. It has two hidden layers: The first hidden layer has 10 nodes while the second one 10. Each processing element (neuron or node) is fully connected between the adjacent layers.

We denote by (i,n) the i^{th} node of the n^{th} layer. Also we let

N_n : the total number of nodes in the n^{th} layer;

x_{in} : output of the node (i,n);

w_{ij}^{n-1} : link weight from the node $(j,n-1)$ to the node (i,n);

θ_{in} : offset (bias) of the node (i,n).

t_{ip} : the i^{th} element of target vector, p. The operation of the node (i,n) is characterized by

$$z_{in} = \sum_{j=1}^{N_{n-1}} w_{ij}^{n-1} x_{j\,n-1} + \theta_{in}, \tag{3}$$

$$x_{in} = f_n(z_{in}), \tag{4}$$

where x_{i1} is the i^{th} element of the input vector. Function $f_n : \mathbf{R} \to \mathbf{R}$ is chosen as

$$f(x) = \begin{cases} x, & \text{if } n = 1 \quad \text{(input layer);} \\ \dfrac{1-e^{-x}}{1+e^{-x}}, & \text{if } n = 2,3,4 \\ & \text{(hidden layer, output layer).} \end{cases} \tag{5}$$

We choose as an error function

$$E_p = \frac{1}{2}\sum_{i=1}^{N_4}(t_{ip} - x_{i4})^2, \tag{6}$$

where x_{i4} represents the i^{th} element of the actual output pattern produced by the presentation of an input. Applying the steepest decent (gradient) rule, we obtain

$$w_{ij}^n(t+1) = w_{ij}^n(t) - \eta\frac{\partial E_p}{\partial w_{ij}^n}, \tag{7}$$

where η is a positive real constant. Constant η determines the learning rate and δ_{in} represents the i^{th} element of the error in the n^{th} layer.

But, instead of (7) we utilize the following modified rule as our weight update algorithm:

$$w_{ij}^n(t+1) = w_{ij}^n(t) - \eta\frac{\partial E_p}{\partial w_{ij}^n} + \alpha\Delta w_{ij}^n(t), \tag{8}$$

where α is a positive real constant which determines the effect of the past weight and $\Delta w_{ij}^n(t) \equiv w_{ij}^n(t) - w_{ij}^n(t-1)$ is often called *momentum term* [5]. It is known, in general, that the addition of the momentum term increases the learning rate without oscillation. Concerning the rule for choosing offset values, θ_{in}'s we have followed the method suggested in [5].

4. A Case Study

We consider a single input system

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 2x_1^2 - 2x_2 \\ 4x_1^3 + x_1^2 + x_1 - 4x_1x_2 - x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2x_1 \end{bmatrix} u. \tag{9}$$

System (9) satisfies conditions *(i),(ii),(iii)* of Theorem 1 and so it is state equivalent to a linear controllable system

$$\begin{bmatrix} \dot{z_1} \\ \dot{z_2} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{10}$$

via a linearizing coordinate transformation map $z = T(x) = \begin{bmatrix} -x_1^2 + x_2 \\ x_1 \end{bmatrix}$.

In this case study we construct the coodrdinate transformation map $z = T(x)$ using BP neural net, which transforms system (9) into the linear controllable system (10). For the training of BP neural net

we uitilize $u(t) = sin(\pi t) + 2.0\ (sin(3\pi t) - cos(3\pi t))$ as an input function. If the mean square error is less than 0.5 percent, then we stop training the neural network. In this case we have $N_1 = 2, N_2 = 10, N_3 = 10, N_4 = 2$ and also $\eta = 0.3, \alpha = 0.3$. Figure 2 shows how a BP neural net weights are trained. Figure 3 shows the target trajectory and the BP neural net outputs after training. But, the training result looks not good. More work needs to be done concerning scaling of the BP neural net input and output.

References

[1] W. Dayawansa, W. M. Boothby, and D. L. Elliot, "Global state and feedback equivalence of nonlinear systems," *Syst. Contr. Lett.*, vol. 6, pp. 229-234, 1985.

[2] L. R. Hunt, R. Su, and G. Meyer, "Global transformation of nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. AC-28, no. 1, pp. 24-31, 1983.

[3] B. Jakubczyk, and W. Respondek, "On linearization of control systems," *Bull. Acad. Pul. Sci. Ser. Math. Astron. Phys.*, vol. 28, pp. 517-522, 1980.

[4] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley. 1990.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel Distributed Processing*, vol. 1, MIT Press, 1986.

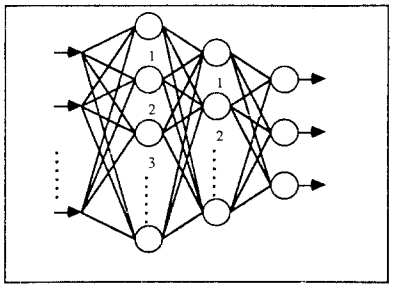

Fig. 1 Structure of two hidden layered BP neural netowrk

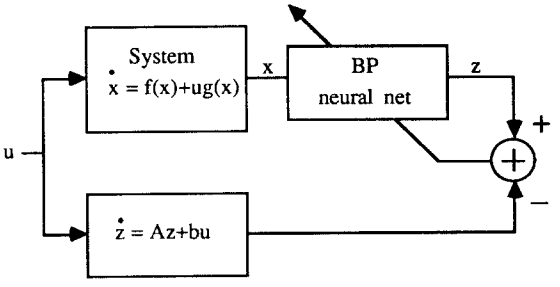

Fig. 2 Block diagram of a BP neural net training for constructing a linearizing coordinate transformation map

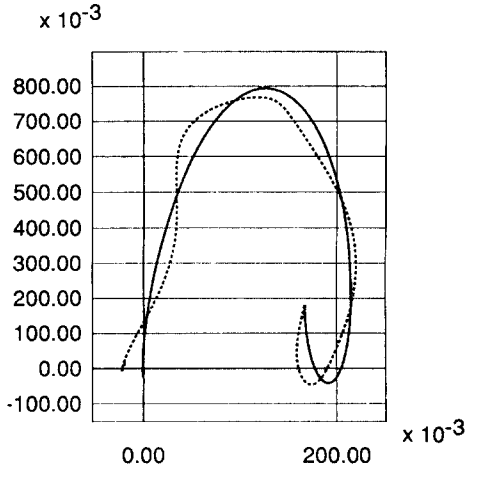

Fig. 3 Phase plots of target trajectory and BP net output
(solid line : target trajectory)
(dotted line : BP net output)