ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 153-158

Iterative Learning Neural Network Control for Nonlinear System Trajectory Tracking

Ping Jiang, Rolf Unbehauen

Lehrstuhl für Allgemeine und Theoretische Elektrotechnik
Universität Erlangen-Nürnberg
Cauerstraße 7, D-91058 Erlangen, Germany

Abstract. This paper presents a neural network controller for nonlinear system trajectory tracking, which works in an iterative learning manner. The controller is composed of many local neural networks and every point along the desired trajectory has its own one for approximating nonlinearity only nearby. This makes that every local neural network can be possessed of a simple structure and less neurons. Because the neural networks are independent from each other, the whole trajectory training can be divided into several segments training, where we train a segment repetitively and extend the trained segment step by step. Stability of the controller is ensured.

1.Introduction

In the last decade, some neural network controllers leading to stable control have been presented[1][2][3]. One of common feature of these approaches appears to be that they try to combine both adaptive control and the robust control technologies for designing neural network controller. Where, the neural networks are used to approximate the uncertainty with only unknown linear weights, the adaptive technologies can then be adopted to update the weights and the residual modeling error is controlled by a robust control scheme. Because the neural networks have to be able to describe the system in a rather large working space around the whole trajectory, the size of the neural networks may become very large for keeping the modeling error within a permitted region. On the other hand, because of correlation of neurons, the network training will face the interference from an other trajectory segment before the ideal weights are achieved. This means that segmental training, during which one segment of the trajectory is repetitively trained, is difficult. Another problem is that the tracking errors may be quite large during the early stages of learning because of poor approximation. In some cases, such as manipulator operating on a complicated workspace or a mobile robot moving within a highly uncertain environment, a large deviation from the desired trajectory may cause collision or even damage.
In this paper, we present a different neural network to control a nonlinear system, where the unknown nonlinearity can be represented by a linearly parameterized approximation model. At first, in contrast to previous global parametric networks or

This work was supported by the AvH of Germany

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 153-158

local networks such as RBF neural networks where the neurons are distributed in state space, we give every point along the desired trajectory an independent neural network where the neurons are distributed along time axis. Then, it is trained from the view point of iterative learning control (ILC)[4][5][6] instead of adaptive control. By iterative learning control, the tracking performance of every point along the trajectory can be improved through tracking the same trajectory many times. It is very suitable for tracking a non-periodic trajectory with a finite time interval, which exists widely in practical applications. On the contrary, the adaptive control based neural networks can only converge to a desired continuous trajectory when the time *t* goes into infinity. Moreover, in our scheme, the training of every local neural network is independent of the others. This makes the segmental training become possible. The ability of segmental training is very useful in many cases. For example, it can be used to ensure uniform boundedness of the tracking error during the whole process of training. We can let the controller always measure the maximum tracking error along the trajectory during the training. if the tracking errors at any trajectory point exceed a predefined bound then the trajectory from this point is divided into two segments. After that, the networks of the first segment are trained repetitively and the next segment can be trained until the tracking of the first one has reached a desired precision. Therefore it works in a step by step manner and finally the whole trajectory tracking can reach the desired precision. In addition, because every local network is only used for approximating a local region around a specific point in the desired trajectory, its structure can be selected to be very simple and with only less neurons. Training this kind of simple neural networks can be easy and fast.

2. Description of iterative learning neural network

We start from a desired trajectory $x_d(t), 0 \le t \le t_f$. Our objective is to force the state vector $x(t,i)$ of the subsequent nonlinear system with affine input to follow the desired trajectory exactly after a series of iterative learning:

$$\dot{x}(t,i) = f(t,x) + G(t,x)u(t,i) \qquad 0 \le t \le t_f \tag{1}$$

where the i expresses the i^{th} iterative training or tracking, $f(t,x) \in R^n$ and $G(t,x) \in R^{n\times m}$ are unknown nonlinear continuous functions, and $x(t,i) \in R^n, u(t,i) \in R^m$ are the state and control input of the i^{th} training at time *t*, respectively.

Because of the approximation capability of a neural network, we employ a linear parametric neural network to express the nonlinear functions *f(t,x)* and *G(t,x)* as follows:

$$f(t,x) = W_f(t)\varphi(t,x) + \varepsilon_1; G(t,x) = G^*(t,x) + \varepsilon_2 = \begin{bmatrix} \varphi(t,x)^T W_1(t) \\ \vdots \\ \varphi(t,x)^T W_n(t) \end{bmatrix} + \varepsilon_2 \tag{2}$$

where $\varphi(t,x) \in R^L$ is a vector composed of basis functions, which depend on what kind of neural network we are using, for example, it may be a radial basis function network [1][3], a high-order neural network[2], etc., it can even be any mathematical approximation model with linear parameters such as a Taylor series, spline functions,

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 153-158

etc.. The $W_f(t) \in R^{n \times L}$ and $W_l(t) \in R^{L \times m}, l = 1 \cdots n,$ are the corresponding unknown optimal weights of the neural networks. The quantities ε_1 and ε_2 are the modeling errors of the approximation and are supposed to be bounded on a compact region Ω.
Note that the above neural networks are different from the adaptive neural networks[1][2][3], where it is assumed that the optimal weights are unknown constants for the total trajectory. In equation (2), the optimal weights $W_f(t)$ and $W_l(t)$ can be time varying. This means that, for every time *t* or more exactly every point along the desired trajectory, one can have different optimal weights. So, instead of a unified neural network for the whole trajectory, equation (2) describes a series of local neural networks for every point along the desired trajectory. From this point, equation (2) is composed of local neural networks for every trajectory point but the adaptive neural networks are global for the whole trajectory. Because every local neural network is only concerned with the uncertainty in a neighborhood around a particular point of the desired trajectory, the local networks are independent from each other and the basis functions can be selected to be very simple.

3. Iterative training of the neural networks

At first, let the i^{th} training error be $e(t,i) = [e_1(t,i), \cdots, e_n(t,i)]^T = x_d(t) - x(t,i)$. In order to deal with the modeling error, we adopt a deadzone scheme [6] and define a modified error vector as

$$e_\Delta(t,i) = e(t,i) - \phi(t,i), \text{ where } \phi(t,i) = [\varepsilon_{f1}\text{sat}(e_1(t,i)/\varepsilon_{f1}), \cdots, \varepsilon_{fn}\text{sat}(e_n(t,i)/\varepsilon_{fn})]^T \quad (3)$$

and $\varepsilon_f = [\varepsilon_{f1}, \cdots, \varepsilon_{fn}]^T$ is an *n*-dimensional width of the deadzone.
Furthermore, we train following neural networks so that their outputs approximate the unknown nonlinear functions at time instant *t*:

$$\hat{f}(t,x) = \hat{W}_f(t,i)\varphi(t,x) \text{ and } \hat{G}(t,x,i) = \begin{bmatrix} \varphi(t,x)^T \hat{W}_1(t,i) \\ \vdots \\ \varphi(t,x)^T \hat{W}_n(t,i) \end{bmatrix} \quad (4)$$

where $\hat{W}_f(t,i)$ and $\hat{W}_l(t,i), l = 1 \cdots n,$ are the i^{th} identification of the optimal weights in equation (2).
Suppose the equation $\hat{G}(t,x,i)x = y$ has a solution on a permitted region around the desired trajectory $x_d(t)$, which can be ensured by the projection algorithm [7]when a rough knowledge about the bound of the parameters is available, then a control law with least norm can be proposed as

$$u(t,i) = \hat{G}(t,x,i)^+ \left(\hat{\dot{x}}_d(t) - \hat{f}(t,x,i) + K_1 e(t,i) + K_2 e(t,i) \right) \quad (5)$$

where $\hat{G}(t,x,i)^+$ denotes the pseudo-inverse matrix of $\hat{G}(t,x,i)$, $\hat{\dot{x}}_d(t)$ is an estimated velocity of the desired trajectory. So, in our scheme, only the state of the desired trajectory should be given but its velocity is not required. This is also a different point from adaptive neural networks. In some cases, when the desired velocity has to be obtained by differentiating a given trajectory, for example, the derivative may be sensitive to the image noise for the feature based visual servoing, this point has benefit.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 153-158

In this control law, we combine neural network compensations for both the velocity and the nonlinearity with robust control (linear feedback). The linear feedback terms of $K_1 e(t,i)$ and $K_2 e(t,i)$ are included to counteract the influence of the modeling errors of ε_1 and ε_2, respectively. Substituting (5) and (2) into equation (1) gives:

$$\dot{x}(t,i) = W_f(t)\varphi(t,x) + \varepsilon_1 + \left(\hat{G}(t,x,i) + \varepsilon_2\right)u(t,i) + \left(G^*(t,x) - \hat{G}(t,x,i)\right)u(t,i)$$
$$= \hat{\dot{x}}_d(t) + \left(W_f(t) - \hat{W}_f(t,i)\right)\varphi(t,x) + \left(G^*(t,x) - \hat{G}(t,x,i)\right)u(t,i) + \left(\varepsilon_1 + K_1 e(t,i)\right) + \left(\varepsilon_2 u(t,i) + K_2 e(t,i)\right) \quad (6)$$

From the criterion in [6], if the width of the deadzone is kept to be constant and the initial error of the system can be controlled within the deadzone for every training, namely $|e_j(0,i)| < \varepsilon_{fj}$, $j = 1 \cdots n$, for $\forall i$, then $\sum_{i=0}^{N} e_\Delta^T(t,i)\dot{e}(t,i) \le \gamma_0^2$ ensures that the tracking error of every point along the trajectory will be less than a specified value described by the deadzone, *i.e.* $\lim_{i\to\infty} |e_j(t,i)| \le \varepsilon_{fj}$, when the system tracks the desired trajectory repetitively. Then , based on equation (6), we want to meet above condition:

$$\sum_{i=0}^{N} e_\Delta^T(t,i)\dot{e}(t,i) = \sum_{i=0}^{N} e_\Delta^T(t,i)\left(\dot{x}_d(t) - \hat{\dot{x}}_d(t,i)\right) + \sum_{i=0}^{N} e_\Delta^T(t,i)\left(\hat{W}_f(t,i) - W_f(t)\right)\varphi(t,x) +$$
$$\sum_{i=0}^{N} e_\Delta^T(t,i)\left(\hat{G}(t,x,i) - G^*(t,x)\right)u(t,i) - \sum_{i=0}^{N} e_\Delta^T(t,i)\left(\varepsilon_1 + K_1 e(t,i)\right) - \sum_{i=0}^{N} e_\Delta^T(t,i)\left(\varepsilon_2 u(t,i) + K_2 e(t,i)\right)$$

For the first term, we know that, although the desired trajectory $\dot{x}_d(t)$ is a function of time *t*, it always remains constant at any specific time *t* for the repetitive tracking. Hence, from the property of a positive real transfer function, if we select the estimation of the desired velocity as

$$\hat{\dot{x}}_d(t,i) = \sum_{j=0}^{i} F(i-j)e_\Delta(t,j) \quad (7)$$

where *F(i-j)* can be any positive definite discrete matrix kernel whose z-transform is a positive real discrete transfer matrix with a pole at *z=1* , then we get

$$\sum_{i=0}^{N} e_\Delta^T(t,i)\left(\dot{x}_d(t) - \hat{\dot{x}}_d(t,i)\right) = \sum_{i=0}^{N} e_\Delta^T(t,i)\left(\dot{x}_d(t) - \sum_{j=0}^{i} F(i-j)e_\Delta(t,i)\right) \le \gamma_1^2 \quad (8)$$

For the same reason, The second term and the third term are upper bounded if the weights of the neural networks are updated by

$$\hat{W}_f(t,i) = -\sum_{j=0}^{i} F_f(i-j)e_\Delta(t,j)\varphi(t,x)^T \quad (9)$$

$$\hat{W}_l(t,i) = -\sum_{j=0}^{i} F_l(i-j)e_{\Delta l}\varphi(t,x)u^T(t,j), \quad l = 1 \cdots n \quad (10)$$

where $F_f(i-j)$ and $F_l(i-j)$ are positive definite discrete matrix kernels as well.

Note that the training laws (7),(9),(10) are updated through accumulations of the repetitive tracking histories at a specific time *t* (a sample period in application). They are different from the training law used by adaptive neural networks where the weights are updated through integration of the state along the time axis *t*.

The next two terms are the robust control terms. The first one for the modeling error ε_1 can be written as

$$-\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(\varepsilon_1 + K_1 e(t,i)\right) = -\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(\varepsilon_1 + K_1 e_{\Delta}(t,i) + K_1\phi(t,i)\right)$$

$$\le \sum_{i=0}^{N}\left(-e_{\Delta}^{T}(t,i)K_1 e_{\Delta}(t,i)\varepsilon_1 + \left|e_{\Delta}(t,i)\right|^{T}\left\|\varepsilon_1\right\|_M - \left|e_{\Delta}(t,i)\right|^{T} K_1\varepsilon_f\right)$$

where $|e_{\Delta}(t,i)| = \left[|e_{\Delta 1}(t,i)|,\cdots,|e_{\Delta n}(t,i)|\right]^T$, $\left\|\varepsilon_1\right\|_M = \left\|\varepsilon_1\right\|_{\infty}[1 \quad \cdots \quad 1]^T$.

If we let $K_1 = diag(K_{11},\cdots,K_{1n}) > 0$ and $K_{1i}\varepsilon_{fi} > \left\|\varepsilon_1\right\|_{\infty}$, which is bounded when we consider the tracking problem on the compact region Ω, then it is less than zero.

For the last term, at first, we rewrite the control law in equation (5) to be $u(t,i) = \hat{G}(t,x,i)^{+}\left(u' + K_2 e(t,i)\right)$ and $E = \varepsilon_2\hat{G}(t,x,i)^{+}$, then

$$-\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(\varepsilon_2 u(t,i) + K_2 e(t,i)\right) = -\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(Eu' + EK_2 e(t,i) + K_2 e(t,i)\right)$$

$$= -\sum_{i=0}^{N}\left(e_{\Delta}^{T}(t,i)EK_2 e_{\Delta} + e_{\Delta}^{T}(t,i)K_2 e_{\Delta}\right) - \sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(Eu' + EK_2\phi(t,i) + K_2\phi(t,i)\right)$$

If we let the control gain K_2 be a positive scalar and further suppose that the maximum norm of the E satisfies $\left\|E\right\|_M < \left|\varepsilon_f\right|_m / \left|\varepsilon_f\right|_M \le 1$, where $\left|\varepsilon_f\right|_m$ and $\left|\varepsilon_f\right|_M$ are the minimum width and the maximum width in the deadzone vector, respectively, then

$$-\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\left(\varepsilon_2 u(t,i) + K_2 e(t,i)\right)$$

$$\le -\sum_{i=0}^{N} K_2\left(1 - \left\|E\right\|_M\right)\left\|e_{\Delta}\right\|^2 + \sum_{i=0}^{N}\left(\left\|e_{\Delta}(t,i)\right\|\left\|E\right\|_M\left\|u'\right\| + K_2\left|e_{\Delta}(t,i)\right|\left\|E\right\|_M\left\|\phi(t,i)\right\|_{\infty} - K_2\left|e_{\Delta}(t,i)\right|^{T}\varepsilon_f\right)$$

$$\le \sum_{i=0}^{N}\left\|e_{\Delta}(t,i)\right\|\left(-K_2\left|\varepsilon_f\right|_m + \left\|E\right\|_M\left\|u'\right\| + K_2\left\|E\right\|_M\left|\varepsilon_f\right|_M\right)$$

$$= \sum_{i=0}^{N}\left\|e_{\Delta}(t,i)\right\|\left(-K_2\left(\left|\varepsilon_f\right|_m - \left\|E\right\|_M\left|\varepsilon_f\right|_M\right) + \left\|E\right\|_M\left\|u'\right\|\right)$$

When we let the control gain satisfy

$$K_2 = \frac{\left\|E\right\|_M}{\left|\varepsilon_f\right|_m - \left\|E\right\|_M\left|\varepsilon_f\right|_M}\left\|u'\right\| > 0 \tag{11}$$

then the last term is less than zero too.

In fact, the matrix *E* reflects the relative error of the approximation for the input matrix *G(t,x)*. So we should select a suitable type of basis functions such that they can approximate the input matrix with a precision less than the ratio between $\left|\varepsilon_f\right|_m$ and $\left|\varepsilon_f\right|_M$. When we make each width of the deadzone for different states equal, this permitted range of the relative error can reach 100%.

Finally, we have that $\sum_{i=0}^{N} e_{\Delta}^{T}(t,i)\dot{e}(t,i)$ is upper bounded and $\lim_{i\to\infty}\left|e_j(t,i)\right| \le \varepsilon_{fj}$.

The proposed neural networks in (2) are the approximations around the desired trajectory for every time instant *t* or every sample period in application. Therefore, they are independent from point to point or from segment to segment. This means that we can divide the whole trajectory training into several segments training. This can help us to avoid large deviation during the training. At the beginning, the system is controlled by a untrained neural network controller. Under the control of this kind

neural network, usually the tracking error will increase along with moving of the desired trajectory. If the tracking errors at any trajectory point exceed a predefined bound then the trajectory from this point is divided into two segments. After that, the networks of the first segment are trained repetitively and the next segment can be trained until the tracking of the first one has reached a desired precision. So it works in a step by step manner, and finally the whole trajectory tracking can reach the desired precision. It makes the tracking error during the training not exceed the predefined bound. We think that step by step is also the learning manner of human being. People prefer to resolve a complicated project into several segments, then they learn all simpler segments repetitively. Before some basic goals are reached, they will never move to the next step. In addition, although we distribute the networks over all trajectory points, it does not mean an increase of the size of the networks because of a simpler structure of every local network for every trajectory point. The training of such a simple local network is faster than a global parametric neural network.

4. Conclusions

We have proposed a neural network controller for nonlinear system trajectory tracking. The proposed method can be used for any approximation model with linear parameters. In contrast to the adaptive neural network control scheme, our training algorithm is derived from the view point of iterative learning control such that every local neural network for a particular point of the trajectory is independent of the others. This makes the repetitive segmental training become possible and every local network can be selected to be very simple. This training law is described by a history accumulation with a gain of positive definite discrete matrix kernel. It guarantees the stability of the tracking system.

References

1. R.M. Sanner, J.J.E. Slotine: Gaussian networks for direct adaptive control. IEEE. Trans. Neural Networks, 3 , 837-863 (1992)
2. E.B. Kosmatopoulos, M.M. Polycarpou, M.A. Christodoulous and P.A. Ioannou: High-order neural network structures for identification of dynamical systems. IEEE. Trans. Neural Networks, 6, 422-431 (1995)
3. G.P. Liu, V. Kadirkamanathan, and S.A. Billings: Variable neural networks for adaptive control of nonlinear systems. IEEE Trans. Systems, Man, and Cybernetics-part C, 29, 34-43 (1999)
4. S. Arimoto, S. Kawamura, F. Miyazaki: Bettering operation of robots by learning. Journal of Robotics System, 1, 123-140 (1984)
5. S.S. Saab: On the P-type learning control. IEEE Trans. Automatic Control, 39, 2298-2302 (1994)
6. P. Jiang, R. Unbehauen: An iterative learning control scheme with deadzone. The 38th IEEE Conf. on Decision and Control, 3816-3817 (1999)
7. K.A. Ossman, E.W. Kmen: Adaptive regulation of MIMO linear discrete-time systems without requiring a persistent exicitation. IEEE Trans. Automatic Control, 32, 397-404 (1987)