ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

Adaptive Simultaneous Perturbation Based Pruning Algorithm for Neural Control Systems

Jie Ni and Qing Song

School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Abstract. It is normally difficult to determine the optimal size of neural networks, particularly, in the sequential training applications such as online control. In this paper, a novel training and pruning algorithm, Adaptive Simultaneous Perturbation Based Pruning Algorithm (ASPBP), is proposed for the online tuning and pruning the neural tracking control system. The conic sector theory is introduced in the design of this robust neural control system, which aims at providing guaranteed boundedness for both the input-output signals and the weights of the neural network.

1 Introduction

Recently, there have been extensive research and significant progress in the area of robust discrete time neural controller designed for nonlinear system with the specific nonlinear functions [1, 2]. However, when a neural system is used to handle unlimited examples, including training data and testing data, an important issue is how well it generalizes to patterns of the testing data, which is known as generalization ability. In this paper, Adaptive Simultaneous Perturbation Based Pruning (ASPBP) algorithm for a generic neural control systems is proposed and a general stability proof for the neural control system is derived. The improved performance of the proposed algorithm can be described in terms of better generalization ability, preventing weight shifting, fast convergence and robustness against system disturbance.

2 NN Tracking Controller and ASPBP Training Algorithm

2.1 Design of the Controller

A dynamic control system can be presented at an input-output form as the following:

$$y_k = f_{k-1} + u_{k-1} + \varepsilon_k + \delta_k \tag{1}$$

where $f_{k-1} \in R^m$ is a dynamic nonlinear function, $\varepsilon_k \in R^m$ denotes the system noise vector and δ_k refers to the artificial error increment caused by pruning, $u_{k-1} \in R^m$ is the control signal vector. The tracking error of the control system can be defined as

$$s_k = y_k - d_k \tag{2}$$

where $d_k \in R^m$ is the command signal.

Define the control signal as

$$u_{k-1} = -\hat{f}_{k-1} + d_k + k_v s_{k-1} \tag{3}$$

where k_v is the gain parameter of the fixed controller and $\hat{f}_{k-1}$ is the estimate of the nonlinear function f_{k-1} by the neural network. Then the error vector can be presented as

$$e_k = f_{k-1} - \hat{f}_{k-1} + \varepsilon_k + \delta_k \tag{4}$$

to train the neural network as shown in Figure 1

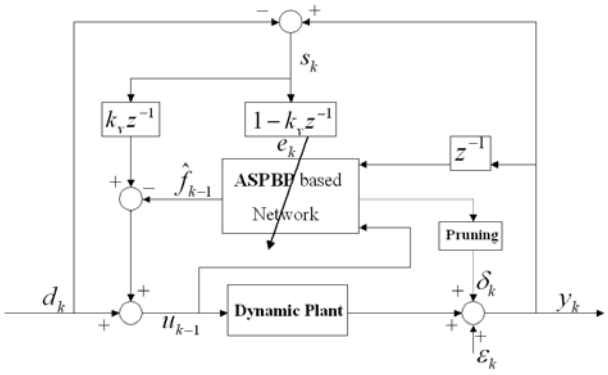


Fig. 1: Structure of the control scheme

The estimation error can be obtained using the closed-loop relationship (1), (3) and (4)

$$e_k = (1 - z^{-1}k_v)s_k \tag{5}$$

2.2 Basic Form of ASP Algorithm

The output of a three-layer neural network can be presented as:

$$\hat{f}_{k-1} = H(\hat{\theta}^w_{k-1}, x_{k-1})\hat{\theta}^v_{k-1} \tag{6}$$

where the input vector $x_{k-1} \in R^{n_i}$ of the neural network is

$$x_{k-1} = [y^T_{k-1}, y^T_{k-2}, \ldots, u^T_{k-2}, u^T_{k-3} \ldots]^T \tag{7}$$

$\hat{\theta}^v_{k-1} \in R^{p_v}$ is the weight vector of the output layer, and $\hat{\theta}^w_{k-1} \in R^{p_w}$ is the weight vector of the hidden layer of the neural network and $H(\hat{\theta}^w_{k-1}, x_{k-1}) \in R^{m \times P_v}$ is the nonlinear activation function matrix with $h_{k-1,i}$ is the nonlinear activation function

$$h_{k-1,i} = h(x^T_{k-1}\hat{\theta}^w_{k-1,i}) = \frac{1}{1 + e^{-4\lambda x^T_{k-1}\hat{\theta}^w_{k-1,i}}} \tag{8}$$

with $\hat{\theta}^w_{k-1,i} \in R^{n_i}$, $\hat{\theta}^w_{k-1} = [\hat{\theta}^{wT}_{k-1,1}, \ldots, \hat{\theta}^{wT}_{k-1,n_h}]^T$ and $4\lambda > 0$, which is the gain parameter of the threshold function.

The Adaptive Simultaneous Perturbation(ASP) approach [3] is composed of two parallel recursions: one for the θ and one for the Hessian of the loss function, $L(\theta)$. The two core recursions are, respectively:

$$\hat{\theta}_k = \hat{\theta}_{k-1} - a_k(\overline{\overline{H}})_k^{-1} G_k(\hat{\theta}_{k-1}) \quad (9)$$

$$\overline{\overline{H}}_k = M_k(\overline{H}_{k-1}) \quad (10)$$

$$\overline{H}_k = \frac{k}{k+1}\overline{H}_{k-1} + \frac{1}{k+1}\hat{H}_k \quad (11)$$

where a_k is a nonnegative scalar gain coefficient, $G_k(\hat{\theta}_{k-1})$ is the input information related to the gradient or the gradient approximation. M_k is a mapping designed to cope with possible non-positive definiteness of $\overline{H}_k$, and $\hat{H}_k$ is a per-iteration estimation of the Hessian matrix.

2.3 Hessian Based Pruning

The main idea of this approach is to choose the minimal saliency [4] caused by the pruning:

$$S_i = \frac{(\hat{\theta}_{ki})^2}{2[H_{i,i}^{-1}]} \quad (12)$$

where H^{-1} is the inverse of the Hessian matrix H, and $[H_{i,i}^{-1}]$ is the ii-th element of the inverse matrix.

So we can define a new term to stand for this error increment(saliency) caused by the delete of weights of both layers

$$\delta_k = \delta_k^w + \delta_k^v \quad (13)$$

$$\delta_k^v = \begin{cases} 0 & \|\delta_k^v\|^2 > \delta \\ \frac{1}{2}\triangle\hat{\theta}_k^v \bar{H}_k^v (\triangle\hat{\theta}_k^v)^T & \|\delta_k^v\|^2 \leq \delta \end{cases}$$

$$\delta_k^w = \begin{cases} 0 & \|\delta_k^w\|^2 > \delta \\ \frac{1}{2}\triangle\hat{\theta}_{ki}^w \bar{H}_k^w (\triangle\hat{\theta}_{ki}^w)^T & \|\delta_k^w\|^2 \leq \delta \end{cases}$$

with δ is a bounded positive constant. This term implies that: before doing pruning, we evaluate the error increment caused by the deletion, if it is less than some criterion, do the pruning; else, abort pruning.i.e.$\delta_k = 0$.

In terms of the definition and prerequisite for the pruning, the error increase $\|\delta_k\|^2 = \|\delta_k^w + \delta_k^v\|^2 \leq 2(\|\delta_k^w\|^2 + \|\delta_k^v\|^2) = 2\delta$. Thus the instant error increase can be regarded as an artificial added noise to the system, which can be absorbed by ε_k.

2.4 The ASPBP Algorithm

Summary of the ASPBP Algorithm for Neural Controller

Step 1.
Initializing: Form the new input vector x_{k-1} of the neural network defined in Eq.(7);
Step 2.
Calculating the output $\hat{f}_{k-1}$ of the neural network: Use the input state x_{k-1} and the existing or initial weights of the network in the first iteration;
Step 3.
Calculating the control input u_{k-1} by using $u_{k-1} = -\hat{f}_{k-1} + d_k + k_v s_{k-1}$;
Step 4.
Evaluating the estimation error e_k by feeding the tracking error signal s_k into a fixed filter;
Step 5.
Calculating the mean squared error of some fixed length: If it is less than the criteria for pruning ξ(where we assume it reaches a local minimum), goto step 6; else, go to step 7;
Step 6.
Calculating the error increment caused by the deletion of the specific weight: If it is less than some constant δ, which is the same criterion for both layers, do pruning using the iterative Hessian matrix, then goto step 2; else go to step 7;
Step 7.
Updating the weights for the output layer and hidden layer respectively;
Step 8.
Go back to step 2 to continue the iteration.

3 The Robustness Analysis of the ASPBP Algorithm

We use $\|.\|$ to denote the Frobenius norm of a matrix and Euclidean norm of a vector in this paper. The stability of the input-output neural control system can be analyzed by the conic sector theory. The main concern is with the discrete time tracking error signal s_k, which is an infinite sequence of real vectors. Consider, the extended space L_{2e}, in which the variable truncations lie in L_2 with

$$\|s\|_{2,t} = \{\sum_{k=1}^{t} s_k^T s_k\}^{1/2} < \infty \tag{14}$$

$\forall t \in Z_+$(the set of positive integer). The conic sector theory and its extension can be found in [6, 5].

The conic conditions for the output layer can be obtained as:

$$\sum_{k=1}^{N}\{e_k^T\Phi_k^v + e_k^T e_k\frac{\breve{\sigma}^v}{2}\} \geq -(\tilde{\theta}_0^v)^2\{\frac{M_k^v\ \breve{\rho}_k^v}{2a_k p_v}\} \quad (15)$$

by selecting a suitable normalized factor $\breve{\rho}_k^v$ to obtain the constant number $\breve{\sigma}^v$ such that

$$1 > \breve{\sigma}^v \geq \frac{a_k^v(M_k^v)^{-1}}{p_v}\|\triangle_k^v\|^2\|r_k^v\|^2(\breve{\rho}_k^v)^{-1} \quad (16)$$

details of which can be found in [7]. The specified normalized factor ρ_k^v plays tow important roles. Firstly, it guarantees $\sigma^v < 1$ to avoid the so-called vanished cone problem [5]; secondly, it guarantees the sector conditions of Theorem 1 to be simultaneously satisfied by both the original feedback system and the normalized equivalent feedback system.

From the definition of δ_k^v, we know that it is bounded too. So this term could be absorbed by ε_k as a system noise, which does not make any impact on the whole system. So we can get the bounded conditions for the pruning of weights in the output layer by taking exactly the same way to the robustness analysis for the learning of the output layer.

And similarly, the conic conditions can be obtained for the hidden layer and the pruning in the hidden layer.

4 Simulation Results

Consider a discrete-time single-link manipulator

$$y(k+2) = 2(1-T)y(k+1) + (2T-1)y(k) + 10T^2 sin(y(k)) + u(k) + d(k) \quad (17)$$

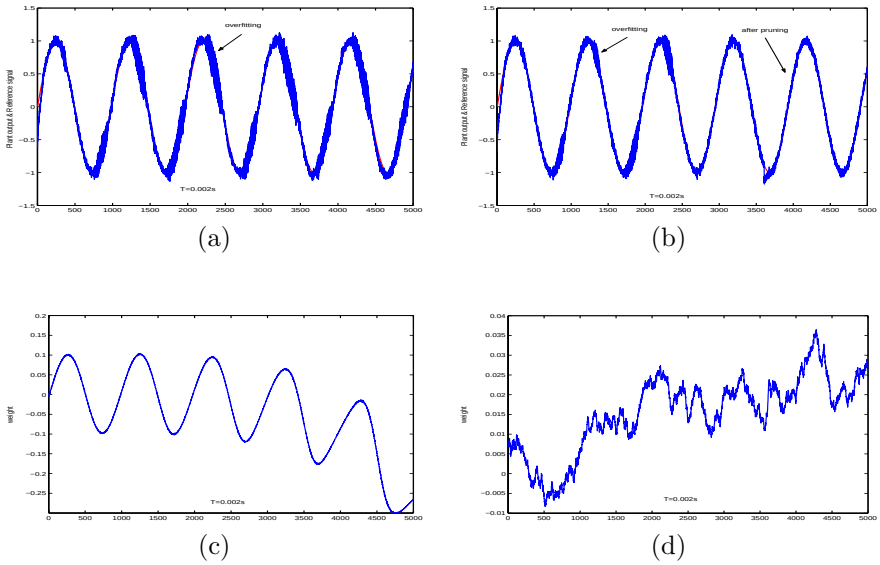
where y is the tracking position signal, u is the control signal $T = 0.002s$ is the sampling time, and d is the disturbance generated from a normally distributed random number with the bound $\|d(k)\| \leq d_m = 0.2$

A three layer feed-forward neural network with 100 hidden neurons is initially used. Figure(a) shows the output of the plant using the standard ASP algorithm without pruning using the command signal $x_{2d} = sin((\pi/5)kT$ and Figure(b) shows the result by using ASPBP algorithm, in which the overfitting has been removed.

Weight drifting in Figure(c) is removed by the perturbation in the ASPBP algorithm as depicted in Figure(d).

5 Conclusion

The ASPBP based pruning method for neural controller has been developed to obtain the guaranteed stability with improved generalization ability and the weight drifting problem is also resolved.



References

[1] M.Lee and H.S.Choi, "A Robust Neural controller for Underwater Robot Manipulators" *IEEE Transactions on Neural Networks*, Vol.11, pp.1465-1470, 2000.

[2] H.D.Patino, R.Carelli and B.R.Kuchen, "Neural Networks for Advanced Control of Robot Manipulators" *IEEE Transactions on Neural Networks*, Vol.13, pp.343-354, 2002.

[3] J.C.Spall, "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method " *IEEE Trans.On Automatic Control*, Vol.45, pp.1839-1853, 2000.

[4] B.Hassibi and D.G.Stork, "Second-order derivatives for network pruning: Optimal brain surgeon" *Advances in Neural Information Processing Systems*, pp.164-171, 1993.

[5] V.R.Cluett, L.Shah and G.Fisher, "Robustness Analysis of Discrete-Time Adaptive Control Systems Using Input-Output Stability Theory: a Tutorial" *IEE Proceedings*, Part D, Vol.135, pp.133-141, 1988.

[6] M.G.Safanov, "Stability and robustness of multivariable feedback systems" *MIT Press*, 1980.

[7] J.Ni and Q.Song, "Pruning Radial Basis Function Neural Network for a Class of Nonlinear Control Systems" revised *IEEE Trans.On Sys. Man. and Cyber.*, Part B, 2004.