2003 정보 및 제어 학술회의 (2003.11.21-22)

Stabilization of Inverted Pendulum Using Neural Network with Genetic Algorithm

Jin Dan, Kab-Il Kim, Young I. Son
Department of Electrical Engineering, Myongji University
E-mail:jindan1127@hotmail.com

Abstract : In this paper, the stabilization of an inverted pendulum system is studied. Here, the PID control method is adopted to make the system stable. In order to adjust the PID gains, a three-layer neural network, which is based on the back propagation method, is used. Meanwhile, the time for training the neural network depends on the initial values of PID gains and connection weights. Hence, the genetic algorithm is considered to shorten the time to find the desired values. Simulation results show the effectiveness of the proposed approach.

Keywords : PID, BP neural network, genetic algorithm, inverted pendulum

Ⅰ. Introduction

The inverted pendulum is a typical experimental equipment in the auto-control theory and also is a typical physical model in the teaching and scientific research in control theory [1].The inverted pendulum, itself, is a natural unstable object. It can reflect many pivotal problems in the control process, such as the nonlinear, the robust character of the system. The ultimate goal is to make such an unstable object into a stable one by using proper control method.

Traditional control method, which is based on models, designs the controller in terms of the mathematical model of the object and the performance guideline of the system requirement and uses the mathematic analysis to describe the control rules [2]. Back propagation (BP) neural network, which has a strong ability to approach nonlinear function, can study and adapt the dynamic characteristic of uncertain system [3]. All the quantitative and qualitative information are saved in the each neuron of the neural network and accordingly, it has strong robustness. But at the same time, traditional BP neural network has some problems: first, it is a nonlinear optimization problem, so it unavoidably has local minimum; second, the time of convergence in the learning algorithm is very slow, it needs thousands of steps or much more. Comparatively, genetic algorithm (GA), during the time of searching optimal solution, in the searching space, it synchronously processes a group of points not only one point, this can avoid local optimal solution and lead fast convergence [4].

Hence, in this paper, we propose a method, which combines BP neural network with genetic algorithm, to design a PID controller and use such a controller to make the inverted pendulum stable.

Ⅱ. Inverted pendulum

We consider the inverted pendulum in Figure 1.

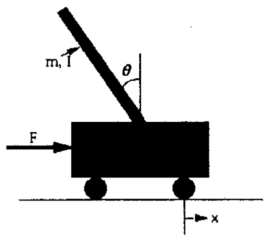

Fig. 1. The inverted pendulum system

M	mass of the cart
m	mass of the pendulum
l	length to pendulum center of mass
I	inertia of the pendulum
F	force applied to the cart
x	cart position coordinate
θ	pendulum angle from vertical

The system consists of a cart whose mass is M and an inverted pendulum whose length is 2L. Under the action of force F, the cart moves along x axes, and makes the inverted pendulum stable in the vertical plane.

Our goal is to make the inverted pendulum stable at the upright position. Considering the characteristic of the physical structure of the system, the goal can also be expressed like this: $x \to 0,\ \theta \to 0,\ \theta' \to 0$

III. Design of PID controller

1. Tuning principle of PID based on BP neural network

The control algorithm of classical increment digital PID is [5]:

$$\begin{aligned} u(k) = u(k-1) + k_p(error(k) - \\ error(k-1)) + k_i error(k) + \\ k_d(error(k) - 2error(k-1) + \\ error(k-2)) \end{aligned} \quad (1)$$

Here, k_p, k_i, k_d are respective proportional (P), integral (I) and derivative (D) gains. And we can use neural network by training and learning to find a three-layer BP neural network, which structure is shown as following:

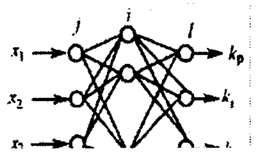

Fig. 2. The structure of a BP neural network

The input of the input layer is

$$O_j^{(1)} = x(j) \quad j = 1,2,\Lambda\ N \quad (2)$$

Here, the number of the input variables M depends on the complexity of the system.

The input and output of the hidden layer are:

$$\begin{aligned} net_i^{(2)}(k) &= \sum_{j=0}^{N} w_{ij}^{(2)} O_j^{(1)} \\ O_i^{(2)}(k) &= f(net_i^{(2)}(k)) \quad (i = 1,\Lambda\ Q) \end{aligned} \quad (3)$$

Here, $w_{ij}^{(2)}$ is the weight coefficient; the superscript (1), (2), (3) represent input layer, hidden layer and output layer.

The activation function of the neuron in the hidden layer is symmetrical sigmoid function:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

The input and output of the output layer are:

$$\begin{aligned} net_l^{(3)}(k) &= \sum_{i=0}^{Q} w_{li}^{(3)} O_i^{(2)}(k) \\ O_l^{(3)}(k) &= g(net_l^{(3)}(k)) \quad (l = 1,2,3) \\ O_1^{(3)}(k) &= k_p \\ O_2^{(3)}(k) &= k_i \\ O_3^{(3)}(k) &= k_d \end{aligned} \quad (5)$$

The output nodes in the output layer are respectively corresponding three adjustable parameters: k_p, k_i, k_d. Because their value cannot be negative, then the activation function of the neuron in the output layer is nonnegative sigmoid function

$$g(x) = \frac{1}{2}(1 + \tanh(x)) = \frac{e^x}{e^x + e^{-x}} \quad (6)$$

The cost function is:

$$E(k) = \frac{1}{2}(rin(k) - yout(k))^2 \quad (7)$$

Where, $rin(k)$ is the input and $yout(k)$ is the output.

The back propagation algorithm employs gradient descent algorithm which modifies the weight to the negative direction of the gradient [3]:

$$\Delta w_{li}^{(3)}(k) = -\eta \frac{\partial E(k)}{\partial w_{li}^{(3)}} + \alpha \Delta w_{li}^{(3)}(k-1) \quad (8)$$

Here, η is learning rate and α is inertia coefficient.

$$\begin{aligned} \frac{\partial E(k)}{\partial w_{li}^{(3)}} = \frac{\partial E(k)}{\partial y(k)} \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial O_l^{(3)}(k)} \cdot \\ \frac{\partial O_l^{(3)}(k)}{\partial net_l^{(3)}(k)} \cdot \frac{\partial net_l^{(3)}(k)}{\partial w_{li}^{(3)}} \end{aligned} \quad (9)$$

$$\frac{\partial net_l^{(3)}(k)}{\partial w_{li}^{(3)}} = O_i^{(2)}(k) \quad (10)$$

Because $\frac{\partial y(k)}{\partial u(k)}$ is unknown, then we use sign function $\operatorname{sgn}\frac{\partial y(k)}{\partial u(k)}$ instead of it, and we can adjust learning rate η to compensate the effect because of inexact calculation.

From formula (1) and formula (5), we can get:

$$\frac{\partial u(k)}{\partial O_1^{(3)}(k)} = error(k) - error(k-1) \quad (11)$$

$$\frac{\partial u(k)}{\partial O_2^{(3)}(k)} = error(k) \quad (12)$$

$$\begin{aligned} \frac{\partial u(k)}{\partial O_3^{(3)}(k)} = error(k) - 2error(k-1) + \\ error(k-2) \end{aligned} \quad (13)$$

From the analysis above, we can get the learning algorithm of the weight of the output layer, that is

$$\begin{aligned} \Delta w_{li}^{(3)}(k) = \eta \delta_l^{(3)} O_i^{(2)}(k) + \\ \alpha \Delta w_{li}^{(3)}(k-1) \end{aligned} \quad (14)$$

$$\begin{aligned} \delta_l^{(3)} = error(k) \cdot \operatorname{sgn}(\frac{\partial y(k)}{\partial u(k)}) \cdot \\ \frac{\partial u(k)}{\partial O_l^{(3)}(k)} \cdot g'(net_l^{(3)}(k)) \end{aligned} \quad (15)$$

$$(l = 1,2,3)$$

With the same principle, we can also get the learning algorithm of the weight of the hidden layer, that is:

$$\Delta w_{ij}^{(2)}(k) = \eta \delta_i^{(2)} O_j^{(1)}(k) + \alpha \Delta w_{ij}^{(2)}(k-1) \qquad (16)$$

$$\delta_i^{(3)} = f'(net_i^{(2)}(k)) \sum_{l=1}^{3} \delta_l^{(3)} w_{li}^{(3)}(k)$$

$$(i = 1,2,\Lambda\ Q) \qquad (17)$$

Here,

$$g'(\bullet) = g(x)(1-g(x)) \text{ and } f'(\bullet) = (1-f^2(x))/2$$

The structure of PID controller based on BP neural network is shown as follows:

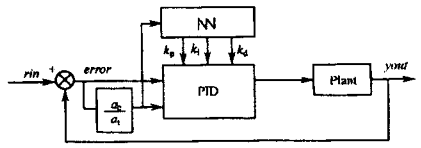

Fig. 3. The structure of a PID controller based on BP neural network

The control algorithm of the controller is concluded as follows

(1) Confirm the structure of BP neural network, which means to make certain the numbers of the nodes of the input layer and hidden layer, and make choice of the learning rate η and inertia coefficient α

(2) Get $rin(k)$ and $yout(k)$, and count the error

$error(k) = rin(k) - yout(k)$

(3) Count the input and output of each layer, and the output of the output layer is three adjustable parameters: k_p, k_i, k_d

(4) Count the output $u(k)$ of the PID controller by formula (1)

(5) Make neural network process learning and adjust weight coefficient $w_{ij}^1(k)$ and $w_{ij}^2(k)$, realize adaptive tune of the parameter of the PID controller

(6) Let $k = k+1$, return (1)

Using the scheme and algorithm above, we can obtain good tuning of the PID gains. But it takes thousands of steps to get convergence results while using genetic algorithm, it only spends hundreds of steps. Therefore, to shorten the computational time to find the fine values, we try to use genetic algorithm.

2. Tuning principle of neural network based on genetic algorithm

In order to apply GA to find the initial values of the PID gains, we propose the following procedure [4]:

(1) Encoding parameter

There are many kinds of coding methods of genetic algorithm [4].Here, we adopt the binary coding method. The relationship between the denotations values and real values is:

$$a = a_{\min} + bin(a_{\max} - a_{\min})/(2^n - 1)$$

Here, bin is the binary coding of the parameter a and $[a_{\min}, a_{\max}]$ is the range of parameter a; n is the length of the binary coding. Let

$$k_p \in [k_{p\min}, k_{p\max}]$$

$$k_i \in [k_{i\min}, k_{i\max}]$$

$$k_d \in [k_{d\min}, k_{d\max}]$$

First, we encode each parameter into binary coding to get three sub-strings and then link them into a whole chromosome in order to make up of the individual of the genetic space. The corresponding form of the parameter is as follows:

$$\underbrace{01010110}_{k_p}\ \underbrace{01000011}_{k_i}\ \underbrace{11010101}_{k_d}$$

(2) Confirming the fitness function

The fitness function should reflect the individual's performance on the actual problem [4]. In this paper, an appropriate fitness function is as following,

$$fitness = e(\max) - e$$

Where, e is the difference between the input and output and $e(\max)$ is the maximum of each population.

(3) Choosing the control parameters

The control parameters include population sizes (S), crossover probability (P_c), and mutation probability (P_m). In this paper, we adopt S=20, P_c=0.85, P_m=0.05.

(4) Generating initial population

(5) Counting fitness

(6) Selection

Selection is the key of genetic algorithm. In this paper, we use roulette wheel method,

(7) Crossover

Here, we use double crossover.

(8) Mutation

(9) Generating the new population

The initial values of weights of the neural network are also obtained by GA. The population size is taken as 15. Selection is based on the fitness function given by above. And unequal crossover is chosen. Crossover probability is 0.7 and mutation probability is 0.04.

Ⅳ. Simulation results

Using Matlab, we simulate the inverted pendulum system with the proposed method. From the pictures, we can see that the proposed method can control the inverted pendulum on the vertical line in a stable way.

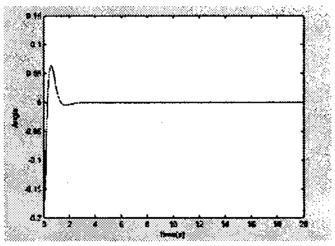

Fig. 4. The pendulum's angle

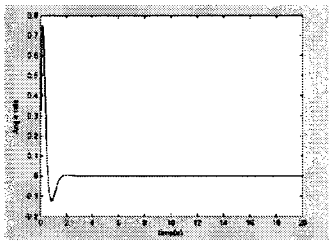

Fig. 5. The angle's rate

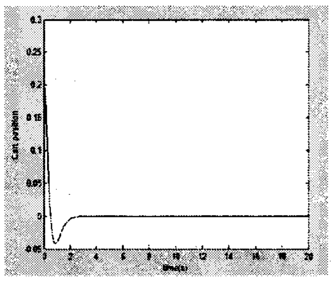

Fig. 6. The car's position

V. Conclusions

This paper designed a PID controller based on the BP neural network and genetic algorithm. And apply the controller to the inverted pendulum. From the paper, we can see that:

(1) The PID controller based on BP network and genetic algorithm can make the inverted pendulum stable

(2) Training the neural network with genetic algorithm, can avoid the local optimization and can get better optimized results.

Although this method is not mature as classical method, it provides a new trend of designing the controller of the inverted pendulum.

The future work is to apply this controller to the more complicated inverted pendulum.

Acknowledgement

The authors would like to thank the Korea Ministry of Science and Technology and the Korea Science and Engineering Foundation for their support through the ERC program.

References

[1] Yu fun-xiu, Duan hai-bin, "Research for Inverted Pendulum Control Based on BP of ANN", *Measurement & Control Technique*, vol.22, pp.41-44, 2003.

[2] Pan shi-zhu, Yang guo-liang, "The application of genetic neural network in PID controller", *Microprocessor and application*, vol.12, pp59-61, 2002.

[3] Dong C. Park, *"Artificial Neural Networks"*, Korean, 1998.

[4] A J F van, Rooij, L C Jain, *"Neural network training using genetic algorithm"*, Singapore: World Scientific Pub, 1996.

[5] Liu in-kun, *"Advanced PID control"*, China: Publishing House of Electronics Industry, 2003.