Evolving a Neural Controller for a Ball-and-Beam System

QING WANG[1], MAN MI[2], GUANGFU MA[3], PIETER SPRONCK[4]

[1-3]Harbin Institute of Technology, 150001, Harbin, P. R. China

[4]IKAT Institute, Maastricht University, The Netherlands

E-MAIL: qingwang@hit.edu.cn

Abstract

This paper presents an evolutionary controller design method for a ball-and-beam system. The method consists of a population of feedforward neural network controllers that evolve towards an optimal controller through the use of a genetic algorithm. The optimal controller is then applied to several different initial positions from which it has to balance the system. From the simulation results, we can conclude that the evolved neural controller can balance the system effectively.

Keywords

Ball-and-beam system; neural network; neural control; genetic algorithm;

1. Introduction

The ball-and-beam balancing system is a well-known standard benchmark for complicated non-linear control methods. Artificial neural networks (ANNs) have the ability to approximate a non-linear function. In the past, ANNs have been employed by several researchers to control the ball-and-beam balancing system. In [1], a feedforward neural network with twelve hidden nodes in a single hidden layer is proposed to regulate the system. In [2], the system is controlled by a two-layer network, developed using error backpropagation (BP) and temporal-difference (TD) learning.

ANNs are commonly designed using the standard BP algorithm or one of the many improved BP algorithms. BP algorithms are gradient-descent learning methods that try to minimize the sum of the errors between the actual output and the target output of an ANN on a training set of typical function samples. A major drawback of BP algorithms is that they can easily get trapped in a local optimum of the search space. They are inefficient in searching for the global optimum of an objective function that is vast, multi-modal and non-differentiable.

Genetic algorithms (GAs), proposed by John Holland in 1975[3], are stochastic search algorithms based on the mechanics of natural selection and natural genetics. They work with a "population" of different potential solutions to a problem. The population "evolves" by generating new solutions, mostly based on the better solutions in the population. GAs are particularly suited for the evolution of controllers for complex systems, since they only require feedback on the performance of a possible solution. In the present research we apply a genetic algorithm to the development of a feedforward neural controller for the ball-and-beam system. We evolve both the weights and the architecture of the network.

2. The ball-and-beam system

The ball-and-beam system, used in [4,5], is illustrated in figure 1. It consists of a beam that pivots at the center point, and a ball that is free to move along the beam in a vertical plane. The task of the system's controller is to apply a sequence of torques to dynamically balance the ball from any initial position with any initial speed (within a $\pm$25-degree beam angle). An optimal controller can get the ball stationary and the beam at a horizontal position within a short period of time.

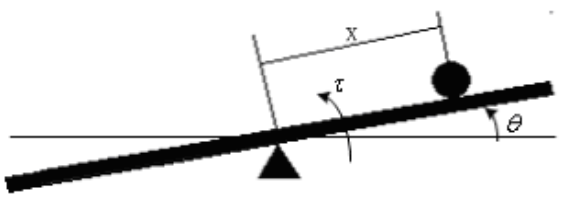


Fig. 1 The ball-and-beam system

Supposing no friction exists in the system, and the ball mass and the beam mass are both uniformly distributed, the ball-and-beam system is described by the following set of Lagrangian equations of motion:

$$\begin{cases} \ddot{x} = \dfrac{\dot{x}\dot{\theta}^2 - g\sin\theta}{1 + I_b / mr^2} \\ \ddot{\theta} = \dfrac{(\tau - mgx\cos\theta - 2m\dot{x}\dot{\theta})}{(mx^2 + I_b + I_a)} \end{cases} \quad (1)$$

In these equations x and θ are generalized position coordinates of the system. The ball position x is measured from the beam center, and is positive if the ball is on the right side of the beam. The beam angle θ is positive if the beam rotates counterclockwise from horizon. The meaning and value of the other parameters are as follows:

g —the gravity acceleration (9.8m/sec^2);

τ —the output of the network, i.e. the action torque applied to the beam center (the torque can take on any value, but in practice is limited to values between -5 and 5 N·m; counterclockwise torque is positive);

m—the mass of the ball (0.01 kg);

r —the rolling radius of the ball (0.01m);

I_a—the moments of the inertia of the beam (0.02 Kg·m^2);

I_b—the moments of the inertia of the ball (2×10^{-6} Kg·m^2);

We set the beam length, which is not part of the equations, to 1.0m.

3. Neural network evolution

To evolve a feedforward neural controller for the ball-and-beam system, we employ a genetic algorithm (GA). In this section we discuss several aspects of the GA, namely encoding, fitness, genetic operators, and selection and replacement.

Encoding

In the genetic algorithm, each neural network is represented by an individual "chromosome." The chromosome consists of an array of "connection genes." Each connection gene represents a single possible connection of the network, and consists of a single bit coupled to a real number. The bit represents the presence or absence of a connection, and the real value specifies the weight of the connection. Even if a connection is absent, the corresponding weight value is maintained in the individual to facilitate the evolution process by functioning as a "memory" for removed weight values.

Fitness function

The fitness function, or objective function, has a critical impact on the quality of results. It provides a measure of success of each evolved individual ANN. The genetic algorithm guides the evolution process to the regions of the solution space that contain the fittest ANNs. For the ball-and-beam system we used the run length until system failure as the fitness measure, where the system is considered to fail when the ball drops off the beam. We specified a target number of time steps T, which is the number of time steps a controller must be able to balance the system without failure to be considered perfect. The fitness f is for the ball-and-beam system is defined as follows:

$$f = \frac{t}{T} \quad (2)$$

where t is the number of time steps the system remains balanced, up to a maximum of T time steps. We used T=50,000, and time steps with a length of 0.002 seconds (so T equals 100 seconds).

Genetic operators

To get new and hopefully better individuals, the GA applies genetic operators to selected individuals. In our research we employ the following genetic operators:

(1) Two point crossover;

(2) Uniform crossover;

(3) Biased weight mutation[6], that adds a random value to selected weights, in a range of [-0.3, +0.3], with each weight of the chromosome having a probability of 5% to be changed;

(4) Unbiased weight mutation[6], that changes selected weights into a random new value, in a range of [-0.3,+0.3], with each weight of the chromosome having a probability of 5% to be changed;

(5) Biased nodes mutation[7], that changes all the

weights representing the input of one of the neural network nodes, within the same range as the biased weight mutation;

(6) Nodes crossover[6], that uses two parents to create two children, that each have half the neural nodes (including their input connections) of each parent;

(7) Node existence mutation[7], that has a probability of 95% to remove a neural node (by disconnecting all incoming and all outgoing connections) and a probability of 5% to completely activate a node (by connecting all possible incoming and all possible outgoing connections);

(8) Connectivity mutation[7], that switches the present-or-absent bit of selected connections, with each connection having a 5% probability to be changed.

Thierens *et al.*'s[8] treatment of competing conventions was used to support the crossover operators.

Selection and replacement

The probability that an individual is selected as a parent is in accordance with its fitness, where the fittest individuals have the greatest chance of being selected. Newly generated individuals replace existing individuals in the population, taking into account elitism. For the selection process tournament selection with a tournament size of 2 was used. As replacement mechanism crowding was used. The population size was set to 100.

4. Experimental results

We performed a large number of experiments applying the method specified in section 3 to the ball-and-beam system. The experiments were performed in the simulation environment Elegance[9], which is a dedicated software environment that realizes genetic neural network controller evolution in a flexible way. Elegance supports a large numbers of genetic algorithm configurations, two types of neural networks (namely feedforward and recurrent neural networks) and several different types of plants to control.

We added the ball-and-beam system as a new plant to Elegance. The outputs of this plant, which are inputs for the controller, represent the state of the plant, consisting of the beam angle θ, the beam velocity $\dot{\theta}$, the ball position x and the ball velocity $\dot{x}$. The input for the plant, which is the output of the controller, is the action torque τ.

The evolution process seeks a neural controller that balances the ball-and-beam system for T=50,000 time steps, starting from a specific initial state (namely an initial ball position of 0.2 and an initial beam angle of 11 degrees). A neural controller that manages to balance the system from the initial state, must then balance the system from a series of different initial states, to guarantee its generalization ability. Specifically, while keeping the initial beam angle of 11 degrees, the controller was tested with initial ball positions ranging from –0.45 to 0.5 with steps of 0.05. Furthermore, while keeping the initial ball position of 0.2, the controller was tested with initial beam angles ranging from 12 to 6 degrees with steps of 0.5.

We tested genetic algorithms with several different subsets of the genetic operators on the ball-and-beam system. We efficiently achieved good ANN results with each of the following three genetic-operator combinations: (i) Connectivity mutation + Unbiased weight mutation + Two point crossover; (ii) Connectivity mutation + Unbiased weight mutation + Uniform crossover; and (iii) Connectivity mutation + Biased weight mutation + Two point crossover. In our tests the third combination was fastest in achieving a high fitness.

Figure 2 shows one of our typical simulation results, where the initial state values are set to θ=10, $\dot{\theta}$=0, x=0.2 and $\dot{x}$=0. Observe that about 30 generations of neural networks (figure 2a) are sufficient to find a ball-and-beam system controller that is very efficient in achieving a good balancing state, consisting of an almost horizontal beam (figure 2b) and the ball almost stationary in the middle (figure 2c).

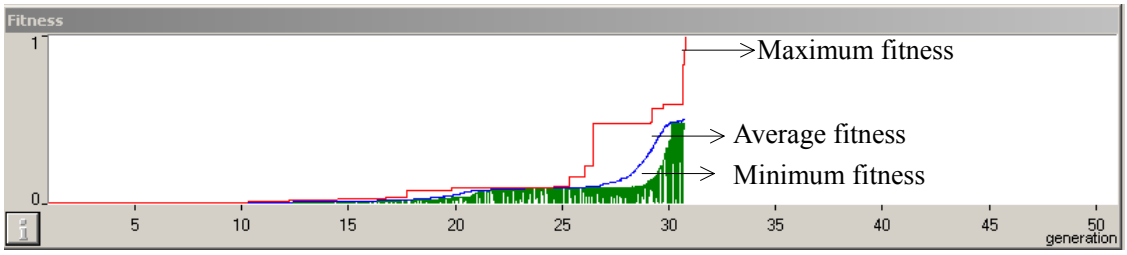
Figure 3 shows the evolved neural controller for the simulation in Figure2. Note that it has a very simple structure, with only two hidden nodes. Compared to the results reported in [1,2], this solution is far more efficient. This confirms the hypothesis offered in [9] that the evolution of neural controllers with genetic algorithms has the benefit of achieving more efficient solutions than regular neural network learning

methods, such as BP.

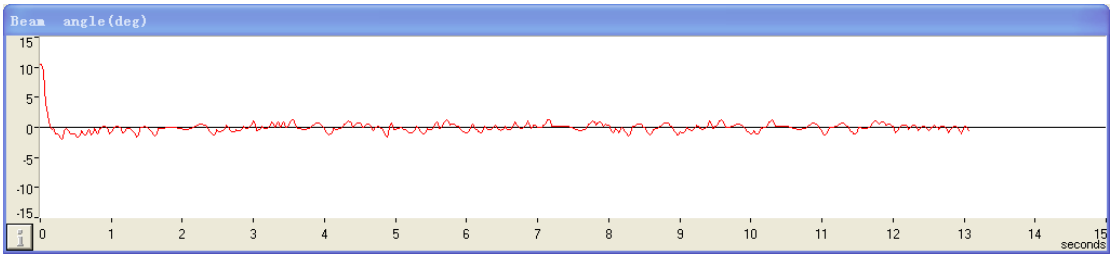
The generalization tests showed that this controller was able to control all ball positions combined with all beam angles of 11 degrees or smaller. An example of a test run, with an initial ball position of 0.5 and an initial beam angle of 11 degrees, is shown in figure 4.

5. Conclusion

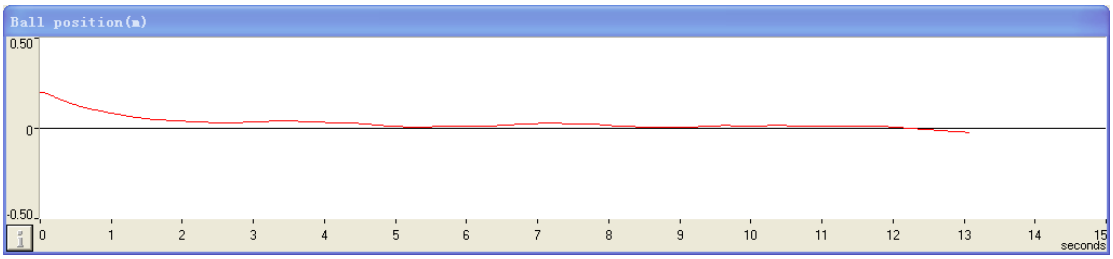
In this paper, we realized a ball-and-beam balancing controller by evolving a feedforward neural network. The simulation results demonstrated that the proposed genetic method solves the ball-and-beam balancing problem efficiently and effectively. Typical solutions found compared favorably with solutions achieved with results achieved with traditional neural network learning methods, as reported in literature[1,2]. The belief is warranted that our genetic approach can outperform traditional neural network learning methods for the control of other classes of non-linear systems as well.



(a) The Fitness



(b) The Beam Angle



(c) The Ball Position

Fig.2 Simulation results for the initial position used for evolution

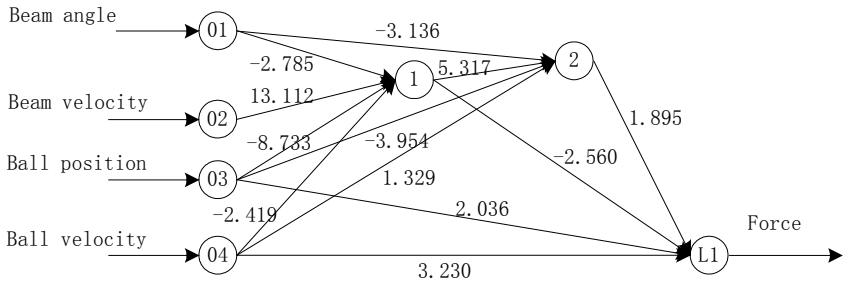
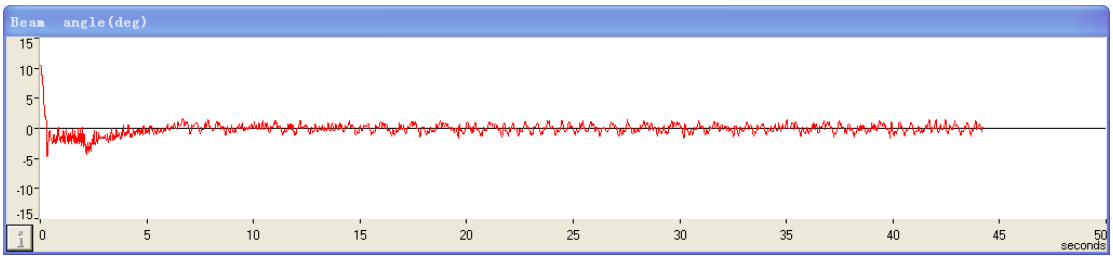
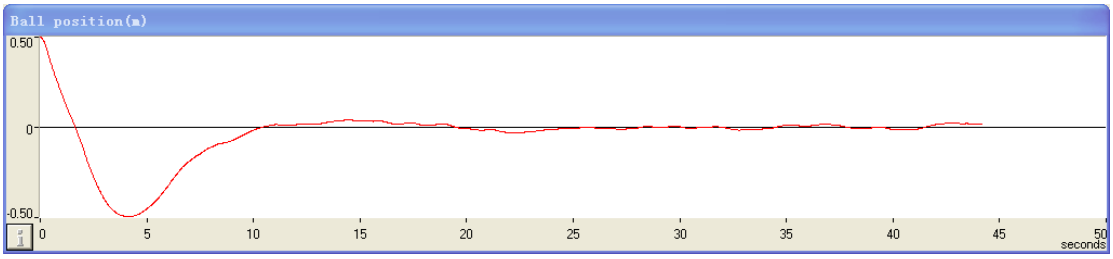


Fig.3 A typical evolved neural Network



(a) Beam angle



(b) Ball position

Fig.4 Results for an extreme initial position, namely a ball position of 0.5 and beam angle of 11 degrees

References

[1] Y. C. Chu and J. Huang. Solving the nonlinear regulator equations by a single layer feedforward neural network. The 23rd international conference on computers and industrial engineering. 1998. 359–362

[2] Y. H. Jiang, C. McCorkell and R. B. Zmood. Application of neural networks for real time control of a ball-beam system. Proceedings of IEEE International Conference on Neural Networks. 1995, Vol.5:2397 - 2402

[3] J. H. Holland. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press. 1975

[4] J. Hauser, S. Sastry and P. Kokotovic. Nonlinear control via approximate input-output linearization: the ball and beam example. IEEE Trans. on Automatic Control, Vol.37, No.3, 1992: 392-398

[5] J. Q. Yi, N. Yubazaki and K. Hirota. Stabilization control of ball and beam systems. Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vol.4 , 2001: 25-28

[6] D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. Proceedings of the 11th International Joint Conference on Artificial Intelligence", Morgan Kaufman, California, 1989: 762-767.

[7] P. Spronck and E. Kerckhoffs. Using genetic algorithms to design neural reinforcement controllers for simulated plants. Proceedings of the 11th European Simulation Conference. A.Kaylan & A. Lehmann, eds.1997: 292-299.

[8] D. Thierens, J. Suykens, J. Vandewalle and B. de Moor. Genetic Weight Optimization of a Feed forward Neural Network Controller. In "Artificial Neural Nets and Genetic Algorithms", R.F. Albrechts, C.R. Reeves and N.C. Steel, eds., Springer-Verlag, New York, 1997: 658-663.

[9] P. H. M. Spronck. Elegance: genetic algorithms in neural reinforcement control. MSc Thesis, Delft University of Technology, Faculty of Technical Mathematics and Information, Delft