Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011

EVOLUTIONARY NEURAL NETWORK FOR GHOST IN MS.PAC-MAN

JIA-YUE DAI, YAN LI, JUN-FEN CHEN, FENG ZHANG

Machine Learning Center, Faculty of Mathematics and Computer Science, Hebei University, Baoding 071002, China
E-MAIL: polyuliyan@yahoo.com.cn

Abstract:

Ms. Pac-Man is a popular chasing and evading game and the ghost character in the game is controlled by script. This article evolved an evolutionary neural network for the red ghost to chase Pac-Man. Red ghost' position, Pac-Man's position and Pac-Man's state are considered to be the inputs of the neural network, and the output is the direction of Red ghost to move in the next step. We also proposed a fitness function to raise capture ability in evolution so that the Red ghost learns by itself in simulation. Experimental results show that the agent learns well and plays better in teamwork than the traditional script controlled ghost.

Keywords:

Evolutionary neural network; Game AI; Pac-Man; Chasing and evading game

1. Introduction

Game AI has been studied for many years, which refers to the techniques for controlling NPC (Non-Player Character) in games. These techniques include script [1], finite state machine which have been applied in games and evolutionary algorithm [2] [3], neural network [4], dynamic Bayesian network [5], case-based reasoning [6], fuzzy logic [7], support vector machine [8], etc. By using these techniques, there are good results reported in some applications such as chess game [9] and path finding [10].

Pac-Man is an arcade game developed by Namco in 1980 and has developed lots of versions. In the version of Ms.Pac-Man used here, player controls Pac-Man to avoid NPC ghosts' chase and try to complete its task of eating all pellets. Since ghost is controlled by script which totally relies on game designer's mind, it may suffer from some disadvantages as stiff action and unstable performance in different maps. In this case, we build a neural network to control the ghost, improve its capacity through self-learning in evolutionary process.

Evolutionary neural network is a method of machine learning that uses evolutionary algorithms to train artificial neural network. Stanley [11] has used it to build agent for NPC. There are mainly two kinds of evolutionary neural networks. The first one only evolves the values of the connection weights for a network while the second evolves both the values of the connection and the topology of the network. As the numbers of input and output are fixed in our method, we use the first kind of evolutionary neural network which is simple and efficient.

Previous research on Pac-Man is mostly focused on the role of Pac-Man exploring ways that Pac-Man can eat more pellets [12] [13], rarely about NPC character ghost. Yannakakis [14] proposed an evolutionary neural network to control ghosts, but the Pac-Man game he used is a much simplified version far away from the popular version: the map is simple that only contains a few channels; there is no power pellets; Pac-Man's speed is always double as ghosts' while their speed is almost the same in popular versions. At the same time, his approach ignores the original design which is a famous setting that four ghosts have specific characters leading to different behaviors. In this paper, we use the more popular version of Pac-Man, Ms. Pac-Man, which is more complex and interesting than that used in [14]. Correspondingly, the evolutionary neural network should be different due to the more complex and real environment. The inputs and outputs of the network are changed and a fitness function is proposed to describe the capture ability of the Red ghost in Ms. Pac-Man.

This paper is organized in six main sections. Section Two describes Ms. Pac-Man's simulation environment. Section Three explains about neural network's structure and parameters. Section Four is the evolution process. Section Five presents our experimental results and the conclusion is given in Section Six.

2. Game environment

We take Ms. Pac-Man game engine which is used in WCCI2008 Pac-Man competition as a simulation environment. This engine can not only play a game visually in normal speed, but also play numbers of games in command line form in high speed. A frame takes 40 ms in visual form while takes much less in command line form. It offers a good opportunity to count import parameters in

978-1-4577-0308-9/11/$26.00 ©2011 IEEE

numbers of games. Meanwhile, this engine provides programming interfaces so that we can modify Pac-Man and ghost's behavior freely. In this paper, the map in first mission of Ms.Pac-Man is used for training and test. The screenshot of mission 1 is shown in Figure 1.

Figure 1. The first mission of Ms.Pac-Man

2.1. Basic elements of Ms.Pac-Man

The basic elements of Ms.Pac-Man scenes include map, Pac-Man, ghost, pellet and power pellet. Table 1 is the brief introduction.

Table 1. Game Elements of MS.PAC-Man

Element	Introduction
Pac-Man	The player controlled character; in the train and test process in this paper, it's controlled by script
Ghost	Their task is to catch Pac-Man, in total of 4 and they behave differently from each other.
Power pellet	power pellets provide Pac-Man with the temporary ability to eat ghost
pellet	Pac-Man will win when eat up all pellet and power pellet
map	Include wall and channel, respectively refers to the places characters can move and can't move

Comparatively, Yannakakis's [14] game environment is much simpler. The long narrow channels with less turning and fork decrease the chasing intensity; Pac-Man is not able to eat ghost because there is no power pellet. The number of environment variables is reduced and correspondingly the interesting degree is also decreased. As the speed is such an important setting in chasing and evading games that Pac-Man's high speed in his simulation has a great influence on game and results. It's clear that our environment is more popular and has a more actual effect.

Ms.Pac-Man's Map size is 224 pixels by 251 pixels. The distance between two pellets is 8 pixels. If we take 8 pixels as the basic unit of length and take 8 pixels × 8 pixel areas as a grid, then the map can be divided into 28 × 31 grid. Ghost and Pac-Man's position in the map is defined by the center grid where they are in.

2.2. Analysis on ghosts' behavior

Four ghosts have different colors in Ms.Pacman, namely red, pink, cyan, orange, and have different patterns of behavior. We will call them respectively as Red, Pink, Cyan and Orange later. Red's behaviors conclude pursuit and random movement; it has a long time in the chase state. Cyan and Pink are relatively complex as they have three patterns of behavior. Orange's behavior remains the same, always move randomly, give people the impression that it's stupid and don't known to chase Pac-Man; Table 2 presents the move rules for all four ghosts.

Table 2. Ghost's Behavior

Ghost	Behavior
Red	When far from Pac-Man Red have a probability to move randomly, otherways it would chase
Pink	When far from Pac-Man it have a probability to move randomly. When near Pac-man Pink would check whether Red is near, if Red is near, it would coordinate with Red, else it would chase
Cyan	When far from Pac-Man it have a probability to move randomly. When near Pac-Man it would chase. In other cases it would select the direction that reduce the gap with the abscissa
Orange	Always move randomly

Pac-Man is a classic chasing and evading game so that Ghosts' chasing is an important part. Red is the most aggressive in four ghosts, spares long time to chase, so it can represent ghosts' chasing. At the same time Red is relatively simple that has only two state of behavior: move randomly and chase. Look at other ghosts, it's no sense to talk about Orange's random movement all time and it's more complex to discuss Pink and Cyan. In this paper, we build neural network for red ghost in chasing state. Later we call red ghost controlled by neural network as NRed while call red ghost controlled by traditional script as Red as before.

Ghost change its direction only when a wall or a fork in front of it. If it changes its direction, it would not select the direction just opposite its original direction. [14] did not mention about ghost's moving rules. The ghost in his methods would change direction casually. It's a big modification because ghosts in traditional game wouldn't steer back which would influence game's styles and make player uncomfortable. At the same time, he didn't consider four ghosts' different moving styles which is an excellent and succeeding setting of the game.

3. Neural network controlled ghost

Neural network are suitable for multi-agent environment [15]. A three-layer forward neural network is employed as an agent here to decide NRed's move. The neural network in Yannakakis's [14] method has 4 inputs which fit the simple map. Besides, its teamwork consideration does not fit our map and ideas of every ghost moving relatively independent. In Ms. Pac-Man, there are two main factors influencing Nred's movement: one is NRed and Pac-man's position in the map, the other is whether Pac-man can eat ghost which decides either NRed to go ahead or behind Pac-man. (When Pac-man just eat a power pellet). Two factors can be received from program and expressed in an array of dimension 5 as the input of NRed's neural network. Input array includes Pac-Man's state, correction value of Pac-Man and NRed's coordinate and horizontal. The Network's output is an array of 4 dimensions indicating 4 directions (up, down, left and right) in the simulation with respective value from 0 to 1. When NRed's turn to decide where to move, it will try to move to the direction with the highest value. If the selected direction is banned because of the rules in 2.2, NRed will try to move to the direction with second highest value. Sigmoid function is employed in each neuron as activation function.

There are five neurons in the hidden layer. In addition of 5 in the input layer and 4 in the output layer, there are 14 neurons in the network. Connection weight's number is 45 which will be learned in the evolution. Weight range from -8.192 to 8.191 and precision is 0.001. bias values are all set to 0, and in the process of evolution bias will not change.

In order to make network computing more sensitive to the input, NRed and Pac-man's position Node are fixed as input. As introduced in 2.1 the map is constituted by 28 × 31 grids, we first obtain NRed or Pac-Man's position node on the map, get position node's vertical X and horizontal Y; then the input value is corrected to (X - 14) / S1 and (Y - 16) / S2 where S1 and S2 is constant. It's shown through experiments that when S1 and S2 take value of 10 the outputs are more sensitive, so S1 and S2 are set to 10. Pac-Man's state input value is 1.5 in normal state and is -1.5 in power state.

4. Genetic algorithm

4.1. Encoding and evolution process

Taking into account the computational complexity, 100 neural networks with random weight constitute the initial population. Every chromosome is encoded by 45 connection weights of a neural network. As mentioned in section 3, weight value range from -8.192 to 8.191 and accuracy is 0.001, so it can be encoded by 14 bits. A chromosome's length is 630 (14 × 45) bits.

Since the weight of neural networks is randomly assigned in the initial stages, the behavior of NRed in early generation can't chase Pac-Man well. In order to accelerate the training speed we make all four ghosts in training process controlled by neural network with same chromosome. So the chromosome's performance can be revealed directly in fitness value. The difference between the four ghosts is that they have unique initial position and entering maze time. The position and entering maze time are respectively equal to the ghosts of Red, Pink, Cyan and Orange which help to avid the situation that all neural network controlled ghosts' crow together. Evolution process is shown in Figure 2.

Game is played by mission in simulation. When a mission is completed ghosts and Pac-Man get back to their traditional position and pellets are restored, waiting for a new start. The three situations below in simulation are regarded as a mission's over.

- Pac-Man eats all the pellets
- Ghost catches Pac-Man
- Game is more than 3000 frames. 3000 frames take two minutes in visual form which is a long time for game Pac-Man.

Pac-Man will play against ghosts for many games in simulation. It's impossible manually to play so that Pac-Man script is applied. We select a script called SmartPac in game engine to control Pac-Man. SmartPac works from short-range ghost avoidance. It would always check its surrounding to select the safer rout and it performance normally with traditional ghosts group. It can win 33.1 times in every 100 games in the first mission's map.

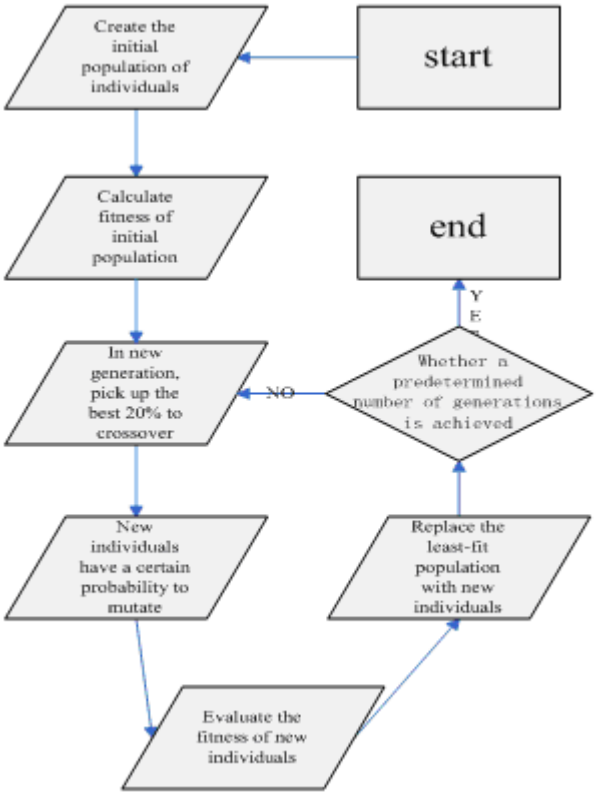

Figure 2. Evolution process

4.2. Fitness function

In Yannakakis's method, fitness function is emphasis on simulation steps (frames) as his map environment is simple and game ends quickly. This function is meaningless in Ms.Pac-Man. In order to train NRed with high performance, out fitness function focuses on two factors which can describe the situation better. One is the times ghost catch Pac-Man in n games, the other is the number of pellets left in the map when Pac-Man is captured.

Formula (1) is capture frequency where d is the number of catching Pac-Man and n is the number of games played.

$$F = d / n \tag{1}$$

Formula (2) is left rate of pellets where p is the number of total pellets and e is the average number of pellets eaten by Pac-Man.

$$L = (p - e) / p \tag{2}$$

We get fitness function (3) by combining formulas above linearly where α and β are constant.

$$C = \frac{\alpha \times F + \beta \times L}{\alpha + \beta} \tag{3}$$

Because the meaning of catching Pac-Man is more than the number of left pellets, in the simulation we set α to 8 and β to 2.

Fitness value cannot reach 1 in simulation. Even if all the Pac-Man is caught in all testing games, it can't be avoidant that some pellets are eaten. With the help of power pellet it is easy for Pac-Man to eat certain number of pellets. We assume the perfect ghost always succeeding in catching Pac-Man and do it efficiently that only keep Pac-Man eat 25% of all pellets. So the fitness of perfect ghost is (8 × 1 + 2 × 0.75) / (8 + 2) = 0.95.

5. Experimental results

This simulation is done using a Core 2 Duo 2.6GHz machine with 2 GB Ram under Windows XP. Visual Studio 2008 is used as the simulation tool.

5.1. Learning result in generations

The number of chromosomes is 100 in the learning phase. Every chromosome controls four ghosts to play 100 games with Pac-Man in every generation. Experiments show that fitness function is nearly converged in generation 100 so the algorithm is determined when 100 generation is achieved. In total the simulation plays 1000000 games in learning phrase which takes about 12 hours. Figure 3 shows the learning result in different generation.

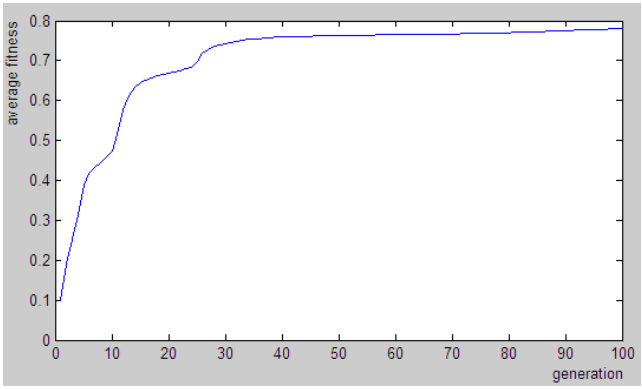

Figure 3. The average fitness of different generation

Ghost's average fitness with a random weight in the first generation only can reach 0.1. It means that ten of hundreds games ghost can win. In the evolutionary process, the ability of capture grows rapidly, with a 10-generation

time to reach the fitness of 0.4, and then evolved a high value of 0.74 in the 40th generation. The average fitness is 0.78 in 100th generation meaning that 89 of 100 games ghosts can catch Pac-Man and the average of left pellets is 74. The result is satisfactory although it is below the performance of the perfect ghost mentioned in 4.2. Then high fitness chromosome is selected to test in 5.2.

5.2. Teamwork Comparison with the traditional group

From 5.1 we get high fitness NRed and let it form a team with Cyan, Orange and Pink. Compare the performance with the traditional team of Red, Cyan, Orange and Pink. In order to get accurate result the number of game played in simulation is 3000. Table 3 is the comparison between two teams in detail.

Table 3. Comparerion of two teams

	Fitness	Capture number	Left pellets
Red team	0.5065	2008	53.2
NRed team	0.5200	2027	58.8

The performance of NRed's team is better than that of the traditional team both in capture number and left pellets.

6. Conclusion

This paper builds an evolutionary neural network for red ghost's chasing behavior in Ms.PacMan. It's an improvement for traditional script controlled ghost based on simple rules. The experiments show neural network ghost can evolve efficiently and learns well by itself through evolution. It works even better in teamwork than traditional ghost team. Future work of our research will focus on two parts: build evolutionary neural network for other ghosts to chase coordinate with NRed and try to use other machine learning method to build agent for ghost.

Acknowledgements

This paper is supported by NSFC (No. 60903088); Natural Science Foundation of Hebei Province (No. F2009000227); 100-Talent Programme of Hebei Province (CPRC002)，the youth natural science foundation of Hebei University (2008Q05), and 2010 Baoding Science Research and Development Project (10ZG008).

References

[1] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma, "Adaptive Game AI with Dynamic Scripting", Machine Learning, Vol 63, No. 3, pp. 217-248, 2006.

[2] Ben Niu, Haibo Wang, Peter H. F. Ng, and Simon C. K. Shiu, "A Neural-Evolutionary Model for Case-based Planning in Real Time Strategy Games", Lecture Notes in Computer Science, Vol 5579, pp. 291-300, 2009.

[3] Jie Zhu, Xiaoping Li and Weiming Shen, "Effective genetic algorithm for resource-constrained project scheduling with limited preemptions", International Journal of Machine Learning and Cybernetics, 2011.

[4] Abdennour EI Rhalibi, and Madjid Merabti, "A Hybrid Fuzzy ANN System for Agent Adaptation in a First Person Shooter", International Journal of Computer Games Technology, Vol 2008, 2008.

[5] S. Yeung, J. Lui, and J. Yan, "Detecting Cheaters for Multiplayer Games: Theory, Design and Implementation", In CCNC, 2006.

[6] David W. Aha, Matthew Molineaux, and Marc Ponsen, "Learning to Win Case-Based Plan Selection in a Real-Time Strategy Game", Lecture Notes in Computer Science, Vol 3620, pp. 5-20, 2005.

[7] Lijuan Wang, "An improved multiple fuzzy NNC system based on mutual information and fuzzy integral", International Journal of Machine Learning and Cybernetics, Vol. 2, pp. 25-36, 2011.

[8] Qiang He and Congxin Wu, "Separating theorem of samples in Banach space for support vector machine learning", International Journal of Machine Learning and Cybernetics, Vol. 2, pp. 49-54, 2011.

[9] Gerald Tesauro, "Temporal Difference Learning and TD-Gammon", Communications of the ACM, Vol. 38, No. 3, pp. 58-68, 1995.

[10] Adi Botea, and Jonathan Schaeffe, "Near Optimal Hierarchical Path-Finding", Journal of game development, 2004.

[11] KO Stanley, BD Bryant, and Miikkulainen, "Real-time neuroevolution in the NERO video game", IEEE Transactions on Evolutionary Computation, Vol 9, pp 653-668, 2005.

[12] SM Lucas, "Evolving a neural network location evaluator to play ms.pac-man," Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 203–210, 2005.

[13] T. Thompson, L. McMillan, J. Levine, and A. Andrew, "An Evaluation of the Benefits of Look-Ahead in Pac-Man", Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 310–315, 2008.

[14] G. Yannakakis, and J. Hallam, "Evolving Opponents for Interesting Interactive Computer Games", Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior, pp. 499-508, 2004.

[15] D. H. Ackley, and M. L. Littman, "Interactions between learning and evolution", In Artificial Life II, pp. 478–507, 1992.