Improve Go AI based on BP-Neural Network

Jianming Liao
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
liaojm@uestc.edu.cn

Difei Zou
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
dfzou@uestc.edu.cn

Rui Luo
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
Oioi77@163.com

***Abstract*—Go is a very interest game with complex rules. Since computer came up, there were no very efficient arithmetic yet, because it is very difficult for traditional way to solve problems like this. But things changed since BP-Neural Network came into the world. Neural Network is proper to solve problem. This paper suggests a solution in Go AI program based on BP-Neural Network, and discussed arithmetic in details.**

***Keywords—Go*, AI, BP-Neural Network, game**

I. INTRODUCTION[1]

Go is a two player board game. It is an Oriental game which originated between 2500 and 4000 years ago and is enjoyed a similar status in Japan, China, Taiwan and Korea as Chess does in Western countries. Go is played on a board which consists of grids made by the intersection of horizontal and vertical lines. The size of the board is generally 19 × 19. Intersection points are connected along the horizontal and vertical lines such that the neighbors of any given point are the intersection points that are horizontally and vertically adjacent to it. Two players alternate in placing black and white stones on the intersection points of the board (including edges and corners of the board) with the black player moving first. The aim of Go is to capture more territory and prisoners than your opponent[1].

Go AI program more different to write than Othello[2]. Because it is have more rules and much wider board, it is impossible to give all the possibility to test. Traditional arithmetic can not be contented. As a fact, Neural Network seems to have inherence to give the answer because it developed from human being. This paper suggests a solution based on Back Propagation Neural Network (BP-NN). The organization of this paper is as followed: In Section 2, the principle of BP-NN will be described for reader to better understand. And In Section 3 the solution will be talked. In Section 4, conclusion and future work will be discussed[3].

II. BACK PROPAGATION NEURAL NETWORK

A. BP-Neural Network Overview

The most popular type of neural network is the back propagation neural network (BP). BP-Neural Network works well for pattern matching and for trend analyzing. It is a feed forward network that uses supervised learning to adjust the connection weights. BP-Neural Network is one important kind of Artificial Neural Network. It is a milestone in the history of developing Artificial Neural Network. This kind of Neural Network has the ability to recognize complex pattern and the ability to map multi-dimension functions. It is a strong learning system with simply structure and easy handing.

B. Topology of BP-Neural Network

Neural Network is a network which consists of numbers of interconnected process units (Neurons) with parallel-distributed information process structure. It is similar with human brain in two aspects: One side is to get know ledges from out environments by learning though neural network. And the other side is to store information with inner neuron[4].

Neuron is basic process unit, Fig. 1. It has *n* input entry. Each of them connects neuron with proper weight. Figure 1 describes the relationship between inputs and outputs.

$$I_i = \sum_{j=1}^{n} w_{ij} x_j + \theta_i, y_i = f(I_i) \tag{1}$$

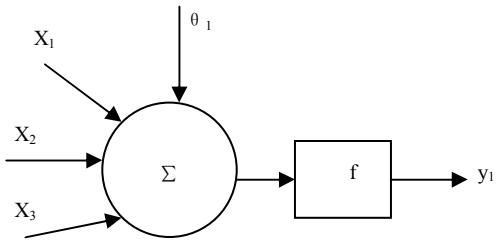

Figure 1. Neuron sturcture

Variable x_j ($j = 1, 2, 3$) is the signal that comes from other neurons. Variable θ_i is the threshold value. Variable w_{ij} presents the connection weight from neuron *i* to neuron *j*.

A layer is composed of many neurons. A conventional BP-Neural Network uses three layers of nodes (See Fig. 2), but it can use more middle layers. The first layer, the input nodes, receives the input data (also called the middle layer or the

[1] This paper is supported by National 863 plan project(NO: 2007AA01Z423)

hidden layer). The results of the first layer are passed to the next layer. This process is repeated for each layer until an output is generated. The difference called error between the generated output and a training set output is calculated. This error is fed back to the network where it is used for connection weight readjustment by iteratively attempting to minimize the difference to within a predefined tolerance. The BP-Neural Network can learn many different output patterns simultaneously with dramatic accuracy[5].

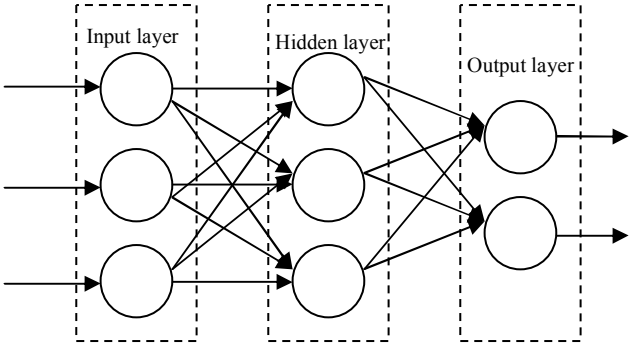

Figure 2. BP-Neural Network topology

In the three layer network, suppose input vector $X = (x_1, x_2, x_3, \ldots, x_i, \ldots x_n)^T$. If let $x_0 = -1$, it can be as the threshold value of hidden layer. Suppose hidden layer's output vector $Y = (y_1, y_2, y_3, \ldots, y_j, \ldots y_m)^T$, if let $y_0 = -1$, it can be as the threshold value of output layer. Suppose output layer's output vector $O = (o_1, o_2, o_3, \ldots, o_k, \ldots o_l)^T$, then expect output vector is $d = (d_1, d_2, d_3, \ldots, d_k, \ldots d_l)^T$. Suppose V presents weight matrix from input layer and hidden layer, then $V = (V_1, V_2, V_3, \ldots, V_j, \ldots V_m)^T$, V_j means weight vector correspond to the j_{th} neuron in the hidden layer. Suppose W presents weight matrix from hidden layer and output layer, then $W = (W_1, W_2, W_3, \ldots, W_k, \ldots W_l)^T$, W_k means weight vector correspond to the k_{th} neuron in the output layer[5].

Next, mathematic relationship will be described in detail.

For output layer,

$$o_k = f(net_k) \qquad k = 1,2,\cdots,l \tag{2}$$

$$net_k = \sum_{j=0}^{m} w_{jk} y_j \qquad k = 1,2,\cdots,l \tag{3}$$

And for hidden layer,

$$y_j = f(net_j) \qquad j = 1,2,\cdots,m \tag{4}$$

$$net_j = \sum_{i=0}^{m} v_{ij} x_i \qquad j = 1,2,\cdots,m \tag{5}$$

Sigmoid function is as followed:

$$f(x) = \frac{1}{1+e^x} \tag{6}$$

Mathematic model of three layers BP-Neural Network is listed from (2) to (6).

C. BP-Neural Network Learning Arithmetic

When actual output is not equal to expect output, we call this output error, represented with E. Definition is[6]

$$E = \frac{1}{2}(d-O)^2 = \frac{1}{2}\sum_{k=1}^{l}(d_k - o_k)^2 \tag{7}$$

Extend definition into hidden layer:

$$E = \frac{1}{2}\sum_{k=1}^{l}[d_k - f(net_k)]^2 = \frac{1}{2}\sum_{k=1}^{l}[d_k - f(\sum_{j=0}^{m} w_{jk} y_j)]^2 \tag{8}$$

And Extend into input layer:

$$\begin{aligned} E &= \frac{1}{2}\sum_{k=1}^{l}\{d_k - f[\sum_{j=0}^{m} w_{jk} f(net_j)]\}^2 \\ &= \frac{1}{2}\sum_{k=1}^{l}\{d_k - f[\sum_{j=0}^{m} w_{jk} f(\sum_{i=0}^{m} v_{ij} x_i)]\}^2 \end{aligned} \tag{9}$$

From (7) to (9), obviously, network input error has something to do with weight of each layer. So changing weight is to change error E.

How to change weight? It is clearly that the less error is the righter weight changes. The conclusions are

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \qquad j = 0,1,2,\cdots,m; k = 1,2,\cdots,l \tag{10}$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} \qquad j = 0,1,2,\cdots,m; i = 1,2,\cdots,n \tag{11}$$

D. Arithmetic Deducibility

We will deduce formula for computing three layers BP arithmetic.

To begin with, we first suppose these condition: for output layer j=0, 1, 2, …, m and k = 1, 2, …, l; for hidden layer j=0, 1, 2, …, m and i = 1, 2, …, n.

For output layer, (10) can be

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k}\frac{\partial net_j}{\partial w_{jk}} \tag{12}$$

For hidden layer, (11) can be

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_k}\frac{\partial net_j}{\partial v_{ij}} \tag{13}$$

Definite an error signal as

$$\delta_k^o = -\frac{\partial E}{\partial net_k} \tag{14}$$

$$\delta_j^y = -\frac{\partial E}{\partial net_j} \tag{15}$$

Considerate (3) and (14), we change (12) into

$$\Delta w_{jk} = \eta \delta_k^o y_j \tag{16}$$

Considerate (5) and (15), we change (13) into

$$\Delta v_{ij} = \eta \delta_j^y x_i \tag{17}$$

$$\delta_k^o = -\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k} = -\frac{\partial E}{\partial o_k} f'(net_k) \tag{18}$$

$$\delta_j^y = -\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_k} = -\frac{\partial E}{\partial y_j} f'(net_j) \tag{19}$$

Because of (7), we get

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \tag{20}$$

And because of (8), we get

$$\frac{\partial E}{\partial y_k} = -\sum_{k=1}^{l} (d_k - o_k) f'(net_k) w_{jk} \tag{21}$$

With (18), (19)

$$\delta_k^o = (d_k - o_k) o_k (1 - o_k) \tag{22}$$

$$\begin{aligned} \delta_j^y &= [\sum_{k=1}^{l} (d_k - o_k) f'(net_j) w_{jk}] f'(net_j) \\ &= (\sum_{k=1}^{l} \delta_k^o w_{jk}) y_j (1 - y_j) \end{aligned} \tag{23}$$

Finally,

$$\begin{cases} \Delta w_{jk} = \eta \delta_k^o y_j = \eta (d_k - o_k) o_k (1 - o_k) y_j \\ \Delta v_{ij} = \eta \delta_j^y x_i = \eta (\sum_{k=1}^{i} \delta_k^o w_{jk}) y_j (1 - y_j) x_i \end{cases} \tag{24}$$

III. ARITHMETIC APPLICATION

A. Arithmetic Description

In the former section, we discussed the BP Neural Network learning arithmetic in detail. In this section, we will talk about Go AI. Because of words limited, we only focus on arithmetic design, and we do not include all the details[7].

As is all known, there are many josekis (formalized series of moves) in game Go. When people start to learn Go, they start to remember josekis. The higher their Go ability is, the more josekis they need remember. Besides, judging to be alive and dead is still a question. It is hard to decide. But the good news is that the chesses needed to be judge usually locate in local blocks. That's where we stand[8].

We dynamic divided the chessboard into several small pieces according to current situation. The principle of divide can be surrounding block or the large blank in the whole chessboard or whatever some clue can be distinguish as a war field. Since the block is smaller than its original, we can suppose the BP-Neural Network can be trained to recognize a particular pattern or feature on the small block. It is obvious that the kink of feature that is important at one place on the board is likely to be important in other places, at other blocks. Therefore, corresponding connections on each unit in a feature map are constrained to have same weights. We have thought that the weight should be larger where the block that the last chess was located. But this is not always the truth. There is a trick in game Go[1]. One in game should try to get forcing movement to gain the most benefit. The one forced to response will not as good as first one. So we give all the block same weight to avoid forced movement. Several feature maps create the hidden layer of the network. All units in the hidden layer are connected to the one output unit. Also, every unit from hidden and output layer has a bias unit connected to it. Finally, every unit from there two layers has a nonlinear activation function (6)[10].

Considering the complex of Go game, we design the network structure as follows.

The input layer to the neural network is the conduit through which the external environment presents a pattern to the neural network. The input layer represents the condition for which we are training the neural network for. Every input neuron should represent some independent variable that has an influence over the output of the neural network. As the size of Go game board usually is generally 19 × 19, and we divided it into small. So, the number of output neurons normal between 80 and 100. The output layer of the neural network is what actually presents a pattern to the external environment. The number of an output neurons should directly related to the type of work that the neural network is to perform. In our application this depends on the known model numbers. In next paragraph we will disucss the arithmetic flow.

B. Arithmetic Flow

- First we initial the weight matrix W and V, and set model counter p and train counter q to 1. Set error E to 0. Learning rate can range from 0 to 1.
- Second, input josekis and compute output of each layer. Use current X^p and d^p to give value to vector array X and d. Equation (2) and (4) are used to compute the elements of Y and O.
- Third, Compute network output error. Suppose we have P pairs of samples, different samples have different error E^p. We use s maximum of them to represent summary error.
- Fourth, Compute error signal of each layer. We can get δ_k^o with (22) and δ_j^y with (23).

- Fifth, adjust weight of each layer. Compute elements of W and V with (24);
- Sixth, check if all the samples have finished training. If $p < P$, p and q increase, return to second. Else jump to seventh.
- Seventh, check if summary error has reached the demand. If $E < E_{min}$, training is over, else $E = 0$, p=1, return back to second.

Totally, above section describes the arithmetic in details. We use tens of josekis as sample to train and the result is content, and the performance also is acceptable[11].

IV. CONCLUSION

Neural Network is not capable of anything. It is wiser if the problem can be solved in traditional way[12][13][14]. But Neural Network can do some things that would otherwise be very difficult. In particular, they can learn a model form training data, such as Go game[2]. Relying on raw board information, this paper showed the solution of using BP-Neural Network. But because of complexity, we have not optimized the arithmetic yet, which will leave to future. We believe that the program will be more intelligent.

REFERENCES

[1] http://www.itee.uq.edu.au/~janetw/Computer%20Go/CS-TR-339.html

[2] Anton Leouski, "Learning of Position Evaluation in the Game of Othello," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[3] http://pubs.acs.org/hotartcl/tcaw/99/nov/simon.html

[4] ZHUANG Xin-tian, YUAN Ying, "Neural Network Modeling and Stock Price Index Forecasting Based on Multifractal Spectrum," Journal of Systems & Management, Vol. 16, No.4 pp. 27-29, Auguest 2007.

[5] LING Cheng-peng, SUN Ya-jun, YANG Lan-he, "Prediction of inrush water of mine with pore water yield base on BP artificial neural network," Water Geography, No. 5, pp. 55-57, July 2007.

[6] Gu Haiyan, Xu Wenke, Yu Lei, "Predictin of Annual Runoff in Songhua River Valley Based on BP Neural Networks," Journal of Northeast Forestry University, Vol. 35, No. 10, pp. 83-85, October 2007.

[7] LI Ji-yao, CHEN Wei, ZHANG Qi, "Intelligent Algorithms in the Simulation of Robotic Soccers," Techniques of Automation & Applications, Vol. 26, pp. 7-9, September 2007.

[8] Shaun-inn Wu, Ruey-Pyng Lu, "Combining Artificial Neural Networks and Statistics for Stock-Market Forecasting," ACM, pp. 257-264, 1993.

[9] SUO Feng-ping, WANG Su-fang, "Application of the manual BP neural network in the traffic forecast," Shanxi Architecture, Vol. 33, No. 23, pp. 359-360, Auguest 2007.

[10] YU Zhi-long, WANG Wei-li, "A new method for rapid prediction of capacity of Li-ion battery," Power Technology, Vol. 31, No.9 pp. 744-746, September 2007.

[11] WANG Jian-ping, GUO Shang, "Improvement Strategies of the BP Neural Network Algorithm Performance," Micro Electronic & Computer, Vol. 24, No. 10 pp. 144-146, 2007.

[12] SHAN Liang, JIN Qing-zhe, ZHOU Shen-li, "Application of back propagation neural network for predicting the film properties and yield of yuba," Food and Fat, No. 8, pp. 19-21, 2007.

[13] Han liqun, Theory of Artificial Nerual Network Design and Application, Chemical Industrial Press, January 2002.

[14] Luo Xiaoshu, Theory of Artificial Nerual Network Design, Guangxi Normal University Press, April 2005.