A High-Performance Variable Structure Neural Network

Z. Gao*, T. Wang, X. Song, C. Wang
Tuzi Network Technology Shanghai
Shanghai, China
**Email: bottos2050[AT]gmail.com*

Abstract—**The paper is focused on the analysis and application of a variable structure feed-forward neural network. For issues of segmentation and performance optimization, the coding scheme, operational parameters, initiation of training configuration, generation of sub-population, decomposition of network connection are all factors which should be considered. By showing the main code, the proposed method to achieve a high-performance variable structure neural net is developed. In order to verify its performance, a balancing robot based experiment is conducted.**

Keywords - Variable structure; Neural network; Segmentation method; Artificial intelligence

I. INTRODUCTION

As one of the most successful tools for machine learning and deep learning, neural networks have been applied for various scenarios in fields of image recognition, speech recognition, decision marking, output prediction, process approximation and robot control [1-4]. Although these days the importance of training data, including quality and quantity, has been well realized, the development and performance improvement of intelligent and self-adaptive models are still essential.

The main purpose of this work is to introduce a high-performance variable structure neural network. Segmentation of neural network directly affects its overall efficiency. Following figure shows the horizontal segmentation of a representative feed-forward neural network. It can be found that with this configuration, some connections between nodes are difficult to be divided, for example, the connection between nodes 2 and 3, the connection between nodes 3 and 8, and so on. Since these connections cannot be easily dealt with, the performance evaluations of such kind of segmentation are not achievable.

Therefore, other alternative solutions should be considered. One of the best alternative solutions is to separate the entire network based on dividing the connections layer by layer. In this way, the groups of connections in between layers can be coded with both connectivity and weight. Some methods such as genetic algorithm, swarm algorithm can be considered to deal with the connectivity and weight [5-10].

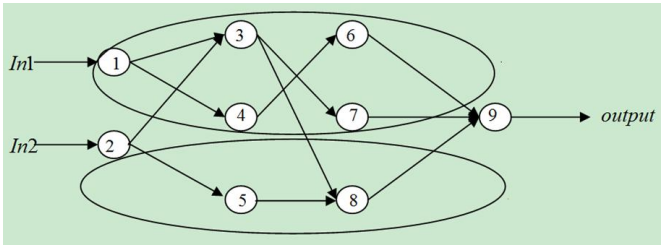

Figure 1. Horizontal segmentation of neural network

II. MOTHOD AND ANALYSIS

In this work, in order to handle the connection and the weight, one issue should be noticed here. Since the population is generated randomly, in some special case, the number of one hidden layer could be zero. Another issue is, after many generation of dynamic adjustment, one hidden layer also has the possibility with zero nodes. To conquer this issue, the aforementioned possibility should be minimized when optimizing the structure and weights.

When error is used as the termination method for model training, there is a tiny chance of unlimited cycling. This problem is possible to be happened especially when using the simple evolutionary algorithm. The reason is the simple evolutionary algorithm usually only optimizes the weight while not the network structure, which can lead to local optimal solution while not the global optima, especially when the setting of approximation accuracy is high.

Besides, in order to reduce the computing resource, the node connectivity and the numerical weights can be considered together. For example, following equation show the connectivity of two layers, each layer has 3 nodes. It seems that there are only 4 real connections.

$$S = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}_{3\times3} \tag{1}$$

However, we have the array of weight showing as follows:

$$0, -0.3, 0.3, 0, 0, -0.1, 0, 0.4, 0 \tag{2}$$

As shown in the following equation which illustrates an example of the integration of node connectivity and the numerical weights, if the value is zero, it means the connection does not exist. However, it is not necessary to remove the node. This is a natural way to combine the two factors. If one element is not zero, the value of this element is the real weight. With this idea, the computing cost can be reduced to half.

$$S = \begin{bmatrix} 0 & -0.3 & 0.3 \\ 0 & 0 & -0.1 \\ 0 & 0.4 & 0 \end{bmatrix}_{3\times 3} \quad (3)$$

III. MAIN CODE

The main code of the introduced method is listed in this section:

```
global POPA POPB      %define sub-population
global POPA1 POPB1
global POPA2 POPB2
global POPA3 POPB3
global FITA    %define individual fitness
global FITB
global T            %number of iteration
global BESTA BESTB
%best individual of sub-population in each generation
global AVEFITA AVEFITB
global PC PM1 PM2
POPSIZE=80;          %size of sub-population
NODE=6;              %node number of hidden layers
T=30;
POPA=zeros(POPSIZE,NODE);
POPA1=zeros(POPSIZE,NODE);
POPA2=zeros(POPSIZE,NODE);
POPA3=zeros(POPSIZE,NODE);
FITA=zeros(1,POPSIZE);
AVEFITA=zeros(1,T);
AVEFITB=zeros(1,T);
POPB=zeros(POPSIZE,NODE);
POPB1=zeros(POPSIZE,NODE);
POPB2=zeros(POPSIZE,NODE);
POPB3=zeros(POPSIZE,NODE);
FITB=zeros(1,POPSIZE);
BESTA=zeros(T+1,7);
BESTB=zeros(T+1,7);
PC=0.9;              % probability of crossover
PM1=0.002;           % probability of structuring mutation
PM2=0.025;           % probability of weight mutation
X=-1:0.1:0.9;
Y=0.5.*(1+cos(X));
t=1;
  for i=1:POPSIZE
    for j=1:NODE
       c=rand;
       if c>0.9
       POPA(i,j)=0;
        else
        POPA(i,j)=rand;
       end
    end
  end
  for i=1:POPSIZE
    for j=1:NODE
       c=rand;
       if c>0.9
       POPB(i,j)=0;
       else
        POPB(i,j)=rand;
      end
    end
  end
[FITA,FITB,AVEFITA,AVEFITB]=calfit(POPSIZE,X,Y);
   %the fitness of first generation
[BESTA,BESTB]=best(POPA,POPB,POPSIZE);
%best individual of first generation and fitness value
dd=1;
for t=2:1:T
  dd=dd+1;

[POPA1,POPB1]=selectoperator(POPA,POPB,POPSIZE);
 %the selection operator

[POPA2,POPB2]=crossoperator(POPA1,POPB1,POPSIZE);
   % the crossover operator

[POPA3,POPB3]=mutation(POPSIZE,dd,POPA2,POPB2);
   %the mutation operator
  POPA=POPA3;
  POPB=POPB3;
[FITA,FITB,AVEFITA,AVEFITB]=calfitness(POPSIZE,X,Y,
BESTA,BESTB,dd);          %fitness after first generation
[BESTA,BESTB]=bestnext(POPA,POPB,POPSIZE,dd);
%record the best individual and fitness after the first generation
End
a0=0;a1=0;
for u=1:20
    a0=abs(last1(u)-Y(u))+a0;
    a1=abs(last2(u)-Y(u))+a1;
end
if a0>a1
    lastbest=last1;
    w1=w1last1;
    w2=w2last1;
else
    lastbest=last2;
    w1=w1last2
    w2=w2last2
end
%function of the representative individual in last generation
Function
[BESTA,BESTB]=perform(POPA,POPB,POPSIZE,dd)
```

```
global FITA FITB
global BESTA BESTB
best_index=0;best=0;
for g=1:POPSIZE
   if  FITA(g)>best
      best=FITA(g);best_index=g;
   end
end
BESTB(dd+1,:)=[POPB(best_index,:),best];
best_index=0;best=0;
for g=1:POPSIZE
   if  FITB(g)>best
      best=FITB(g);best_index=g;
   end
end
BESTA(dd+1,:)=[POPA(best_index,:),best]
```

IV. APPLICATION

Self-balancing is essential for robotic systems since motion intelligence is one of the most basic functions compared with computing intelligence, sensory intelligence and cognitive intelligence [11-12]. Following figure shows one example of a robotic system to be balanced.

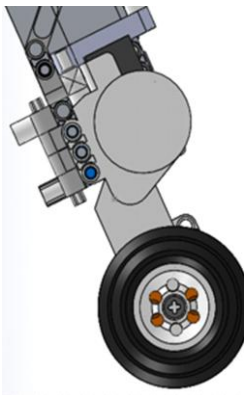

Figure 2. An example of robotic system to be balanced

System balancing can be generally found in many industrial and real world applications where position, speed, toque, etc. must be considered simultaneously. Generally speaking, the PID algorithm is shown as follow [13-15]:

$$u(t) = K_p\left[e(t) + \frac{1}{T_I}\int_0^t e(t)d(t) + \frac{T_D de(t)}{d(t)}\right] \quad (4)$$

Since the closed loop system is high order, it is approximated as a 2nd order whose general expression is,

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0 \quad (5)$$

It can be rewritten as,

$$\frac{d^2x}{dt^2} + 2\zeta w_n\frac{dx}{dt} + w_n^2 x = 0 \quad (6)$$

The damping ratio is approximately derived as,

$$\zeta \cong \ln\frac{previous\ peak\ of\ output}{current\ peak\ of\ output} \quad (7)$$

By applying the introduced method, the PID control method was compared with it and it was found that the overall performance was largely improved.

V. CONCLUSIONS

Neural networks have become the most important and popular technique of artificial intelligence and machine learning. There are many different structure and training methods of neural networks nowadays in the world. This purpose of this paper is to introduce a high-performance variable structure neural network with a segmentation of node connectivity and the numerical weights. In order to save the calculation resource, the integration of node connectivity and the numerical weights is considered to improve the overall efficiency. As a natural way to combine the two factors, this method shows the feasibility and application potentials.

REFERENCES

[1] Edwards, Chris (25 June 2015). "Growing pains for deep learning". Communications of the ACM. 58 (7): 14–16.

[2] Dwarakish, G. S.; Rakshith, Shetty; Natesan, Usha (2013). "Review on Applications of Neural Network in Coastal Engineering". Artificial Intelligent Systems and Machine Learning. 5 (7): 324–331

[3] Griewank, Andreas; Walther, Andrea (2008). Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition. SIAM. ISBN 978-0-89871-776-1.

[4] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks" . NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.

[5] Risi, Togelius. Neuroevolution in Games: State of the Art and Open Challenges. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG) 2017

[6] Tahmasebi, Pejman; Hezarkhani, Ardeshir (21 January 2011). "Application of a Modular Feedforward Neural Network for Grade Estimation". Natural Resources Research. 20 (1): 25–32. doi:10.1007/s11053-011-9135-3.

[7] Roman M. Balabin; Ravilya Z. Safieva; Ekaterina I. Lomakina (2007). "Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction". Chemometr Intell Lab. 88 (2): 183–188. doi:10.1016/j.chemolab.2007.04.006.

[8] García-Pedrajas, N., Hervás-Martínez, C. and Muñoz Pérez, J., COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Transactions on Neural Networks. v14 i3. 575-596.

[9] Risi, Stanley. An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density and Connectivity of Neurons. Artificial Life journal, Cambridge, MA: MIT Press, 2012

[10] Robail Yasrab, Naijie Gu and Xiaoci Zhang, An Encoder-Decoder Based Convolution Neural Network (CNN) for Future Advanced Driver Assistance System (ADAS), Appl. Sci. 2017, 7(4), 312; doi:10.3390/app7040312

[11] Jih-Gau Juang, Yi-Ju Tsai and Yang-Wu Fan, Visual Recognition and Its Application to Robot Arm Control, Appl. Sci. 2015, 5(4), 851-880; doi:10.3390/app5040851

[12] Bingfei Nan, Zhichun Mu, Long Chen and Jian Cheng, A Local Texture-Based Superpixel Feature Coding for Saliency Detection Combined with Global Saliency, Appl. Sci. 2015, 5(4), 1528-1546; doi:10.3390/app5041528

[13] Kebriaei, Reza; Frischkorn, Jan; Reese, Stefanie; Husmann, Tobias; Meier, Horst; Moll, Heiko; Theisen, Werner. "Numerical modelling of powder metallurgical coatings on ring-shaped parts integrated with ring rolling". Material Processing Technology. 213(1): 2015–2032.

[14] Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew (2014). "Going Deeper with Convolutions". Computing Research Repository: doi:10.1109/CVPR.2015.7298594.

[15] Ciresan, Dan; Giusti, Alessandro; Gambardella, Luca M.; Schmidhuber, Juergen (2012). Pereira, F.; Burges, C. J. C.; Bottou, L.; Weinberger, K. Q., eds. Advances in Neural Information Processing Systems 25. Curran Associates, Inc. pp. 2843–2851.