北京航空航天大学
BEIHANG UNIVERSITY

Neural Networks and Back Propagation

Experiment Report

School	School of Automatic Science
Major	Pattern Recognition and Intelligent System
Student	Tianxiang Lan
Student ID	15231087
Teacher	Zengchang Qin

May 11,2018

1 Introduction

The learning model of Artificial Neural Networks (ANN) (or just a neural network (NN)) is an approach inspired by biological neural systems that perform extraordinarily complex computations in the real world without recourse to explicit quantitative operations. The original inspiration for the technique was from examination of bioelectrical networks in the brain formed by neurons and their synapses. In a neural network model, simple nodes (called variously “neurons” or “units”) are connected together to form a network of nodes, hence the term “neural network”.

Each node has a set of input lines which are analogous to input synapses in a biological neuron. Each node also has an “activation function” that tells the node when to fire, similar to a biological neuron. In its simplest form, this activation function can just be to generate a ‘1’ if the summed input is greater than some value, or a ‘0’ otherwise. Activation functions, however, do not have to be this simple - in fact to create networks that can do useful things, they almost always have to be more complex, for at least some of the nodes in the network. Typically there are at least three layers to a feed-forward network - an input layer, a hidden layer, and an output layer. The input layer does no processing - it is simply where the data vector is fed into the network. The input layer then feeds into the hidden layer. The hidden layer, in turn, feeds into the output layer. The actual processing in the network occurs in the nodes of the hidden layer and the output layer.

2 Principle and Theory

The goal of any supervised learning algorithm is to find a function that best maps a set of inputs to its correct output. An example would be a simple classification task, where the input is an image of an animal, and the correct output would be the name of the animal. For an intuitive example, the first layer of a Neural Network may be responsible for learning the orientations of lines using the inputs from the individual pixels in the image. The second layer may combine the features learned in the first layer and learn to identify simple shapes such as circles. Each higher layer learns more and more abstract features such as those mentioned above that can be used to classify the image. Each layer finds patterns in the layer below it and it is this ability to create internal representations that are independent of outside input that gives multi-layered networks their power. The goal and motivation for developing the back-propagation algorithm was to find a way to train a multi-layered neural network such that it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output.

Mathematically, a neuron's network function *f(x)* is defined as a composition of other functions $g_i(x)$ which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the nonlinear weighted sum, where:

$$f(x)=\sum_i w_i g_i(x)$$

where K (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.Back-propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method.

The squared error function is:

$$E=\frac{1}{2}(t-y)^2$$

where E is the squared error, t is the target output for a training sample, and y is the actual output of the output neuron. For each neuron j, its output o_j is defined as

$$o_j=\varphi(net_j)=\varphi(\sum_{k=1}^{n} w_{kj}x_k)$$

The input net to a neuron is the weighted sum of outputs o_k of previous neurons.If the neuron is in the first layer after the input layer, the o_k of the input layer are simply the inputs x_k to the network. The number of input units to the neuron is n. The variable w_{ij} denotes the weight between neurons i and j.

The activation function φ is in general non-linear and differentiable. A commonly used activation function is the logistic function, e.g.:

$$\varphi(z)=\frac{1}{1+e^{-z}}$$

which has a nice derivative of:

$$\frac{\partial\varphi}{\partial z}=\varphi(1-\varphi)$$

Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial \mathrm{w}_{ij}}=\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

We can finally yield:

$$\frac{\partial E}{\partial \mathrm{w}_{ij}}=\delta_j x_i$$

With

$$\delta_j=\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}=\begin{cases}(o_j-t_j)\varphi(net_j)(1-\varphi(net_j)) & if.j.is.an.output.neuron\\ (\sum_{l\in L}\delta_l w_{jl})\varphi(net_j)(1-\varphi(net_j)) & if.j.is.an.inner.neuron\end{cases}$$

3 Objective

The goals of the experiment are as follows:

(1) To understand how to build a neural network for a classification problem.

(2) To understand how the back-propagation algorithm is used for training a given a neural network.

(3) To understand the limitation of the neural network model (e.g., the local minimum).

(4) To understand how to use back-propagation in Autoencoder.

4 Contents and Procedure

Stage 1：

(1) Given a dataset for classification, (E.g., Iris, Pima Indian and Wisconsin Cancer from the UCI ML Repository). Build a multi-layer neural network (NN) and train the network using the BP algorithm. We can start with constructing a NN with only one hidden layer.

(2) Compare the NN results to the results of Perceptron, which model is better in terms of efficiency and accuracy?

(3) Whether the performance of the model is heavily influenced by different parameters settings (e.g., the learning rate α)?

Stage 2：

(1) How the efficiency and accuracy will be influenced by different activation functions and more hidden layers.

(2) Do you think that biological neural networks work in the same way as our NN model? Can you provide any discussions to support your opinions?

(3)Using the BP algorithm to train an autoencoder.

(We will train the simplest autoencoder which is a feedforward, non-recurrent neural net with an input layer, an output layer and one or more hidden layers connecting them. The difference is that, for an autoencoder, the output layer has equally many nodes as the input layer, and instead of training it to predict some target value y given inputs x, an autoencoder is trained to reconstruct its own inputs x.

(4) For an autoencoder with only one hidden layer, how the number of nodes in the hidden layer influence the model performance?

Stage 3：

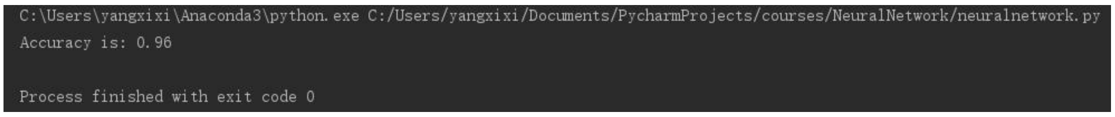
Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences about this experiment study.

5 Results and Analyzation

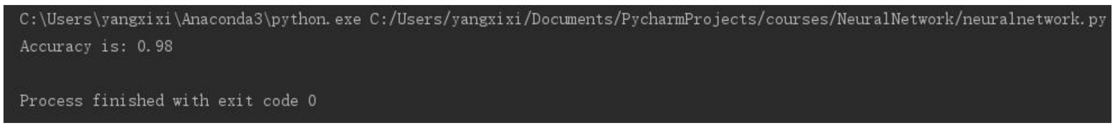
Stage 1:

(1)Given a dataset for classification.Build a multi-layer neural network (NN) and train the network using the BP algorithm.

I choose the Iris Dataset from the UCI Machine Learning Repository to do experiment.The accuracy of 3 layers(1 input layer,2 hidden layers,1 output layer) is 0.98, 0.96 or some other number because the partition of dataset is random every time.

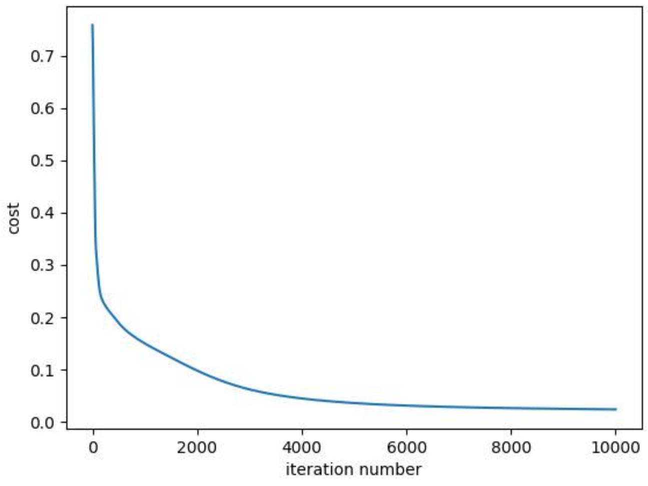


```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.96

Process finished with exit code 0
```



```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.98

Process finished with exit code 0
```

As we can see,the cost goes down when iteration increase.The cost function converge at about 4,000 times iteration when learning rate is 0.1.

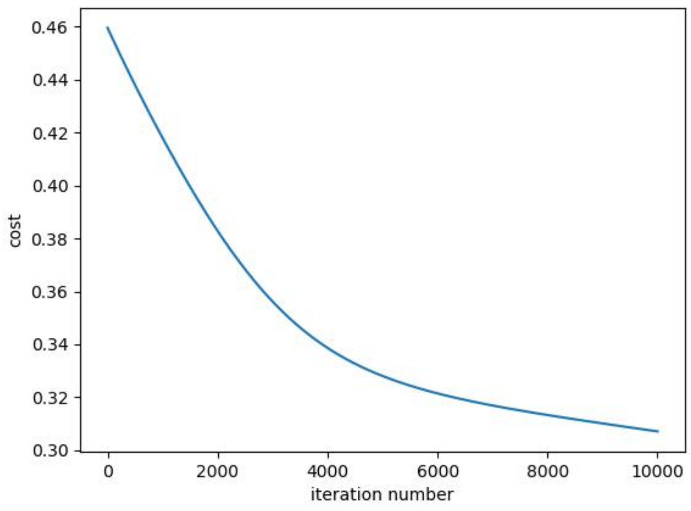


(2)Compare the NN results to the results of Perceptron, which model is better in terms of efficiency and accuracy?

Neural network performs better in terms of efficiency and accuracy.

(3) Whether the performance of the model is heavily influenced by different parameters settings?

Yes. For example,I change the learning rate from 0.1 to 0.001,and the cost curve looks like the following picture:



As you can see in the picture,the cost function don’t converge even when the iteration number is 10,000.Therefore,learning rate affect the performance of the model,especially for the speed of convergence.

Stage 2:

(1)How the efficiency and accuracy will be influenced by different activation functions and more hidden layers?

In fact,the selection of activation functions is very important.We hardly take sigmoid function as activation function,because performances of sigmoid function are weaker than hyperbolic tangent function in almost every situations,unless we finish a classification assignment with output layer outputting ‘0’ or ‘1’.In most case,we usually use ReLU function,especially in the field of computer vision(CV).In field of nature language processing(NLP),hyperbolic tangent function is more commonly used.

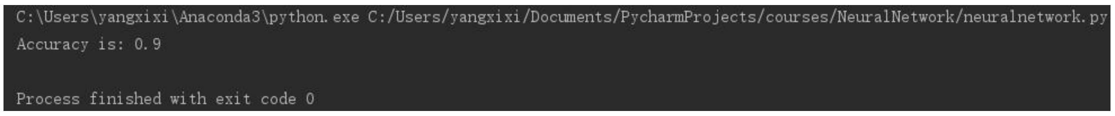
As for deep neural network,it usually performs much better than shallow neural network in efficiency and accuracy because of large scale of modal.

(2)Do you think that biological neural networks work in the same way as our NN model? Can you provide any discussions to support your opinions?

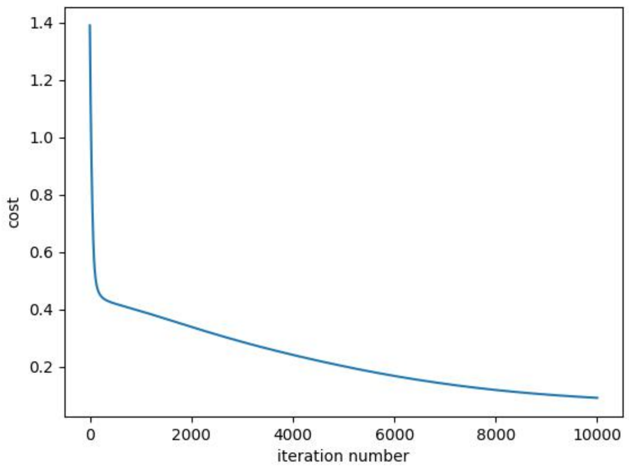
I think the way biological neural networks work is different from the artificial neural networks.In fact,artificial neural networks are realized by calculation.While biological neural networks are realized by establishing synapses,which is a physical process.But they are both work in a group of units.That’s way they looks have the same effect.

(3)Using the BP algorithm to train an autoencoder.

I generate an autoencoder neural network,consist of 1 input layer,1 hidden layer and 1 output layer.For example,cell number in input and output layer is 10,cell number in hidden layer is 5.The result are following pictures:



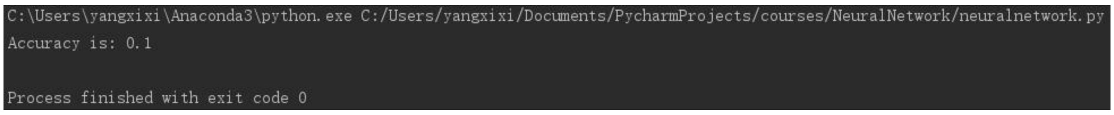
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.9

Process finished with exit code 0
```



(4)For an autoencoder with only one hidden layer, how the number of nodes in the hidden layer influence the model performance?

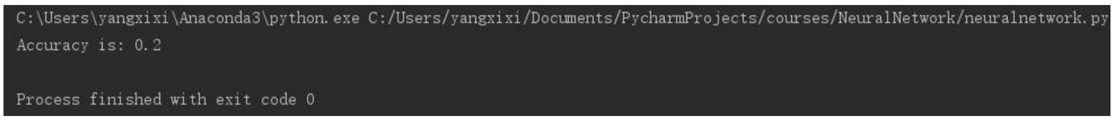
Change the cell number of hidden layer.

Cell number layer is 1:



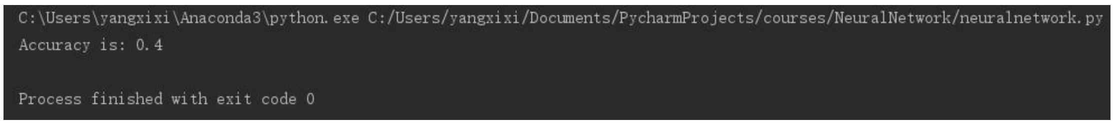
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.1

Process finished with exit code 0
```

Cell number layer is 2:



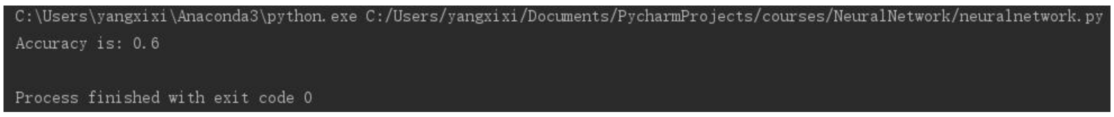
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.2

Process finished with exit code 0
```

Cell number layer is 3:



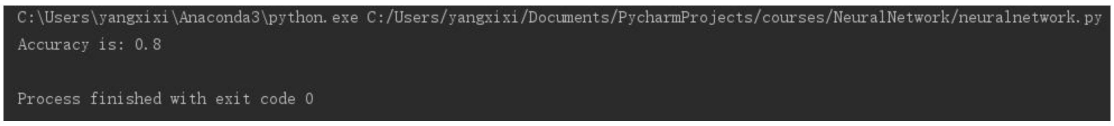
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.4

Process finished with exit code 0
```

Cell number layer is 4:



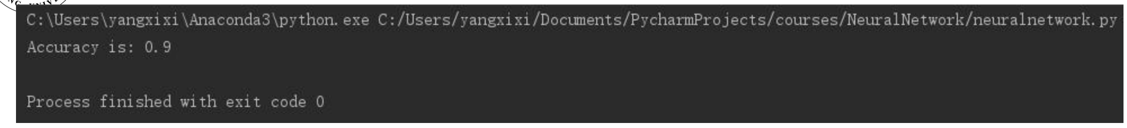
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.6

Process finished with exit code 0
```

Cell number layer is 5:



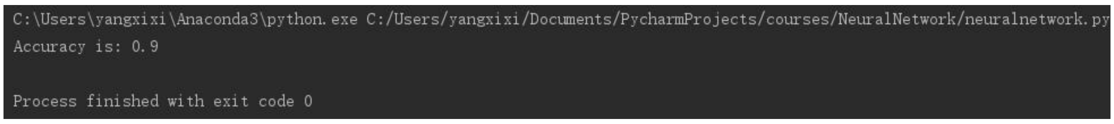
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.8

Process finished with exit code 0
```

Cell number layer is 6:



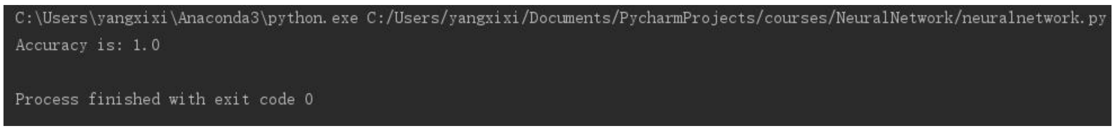
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.9

Process finished with exit code 0
```

Cell number layer is 7:



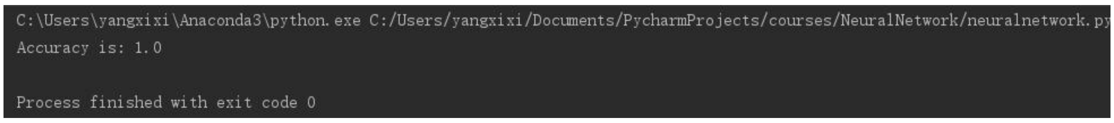
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 0.9

Process finished with exit code 0
```

Cell number layer is 8:



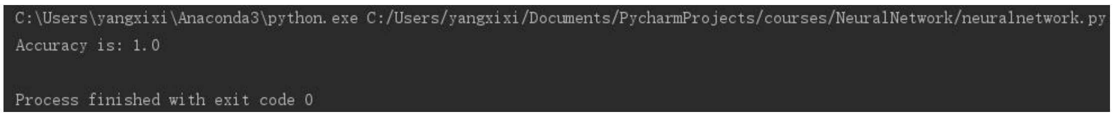
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 1.0

Process finished with exit code 0
```

Cell number layer is 9:



```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 1.0

Process finished with exit code 0
```

Cell number layer is 10:



```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/NeuralNetwork/neuralnetwork.py
Accuracy is: 1.0

Process finished with exit code 0
```

As you can see,cell number of hidden layer is larger,the performance is better.

All the code and instruction files are on my github.You can run them if you want:
https://github.com/Deep-Lan/Neural-Network.git