**Experiment 4 Neural Networks and Back Propagation**

**4.5 Results and Report**

**Stage 1**：

1. In this experiment, I choose Wisconsin Cancer from the UCI ML Repository as my data set.

**train data set and test data set**

At First, I code some Functions to generate my train data set(80%) and test data set(20%) from the data set, you can see it in the File “gen\_set.py”. If you want to generate your own train data set and test data set, you can run the file in the terminal like this:



I have generated the two data set, you can see it in the direction “ANN/dataset”.

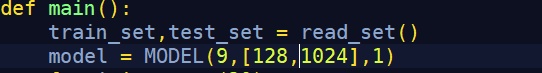
**Normalized Function**

By the way, in order to improve the efficiency of the machine learning model, I have normalized values of all attributes. What’s more, because the first attribute “ID” is useless, I delete this attribute in my two data set. Supposing that there are one value **x** in the “original” data set, and the maximum and minimal in the attribute equal to ɑ and β, so the normalized value of x can be:



**ANN Model**

I have built my ANN Model in the File “ANN.py”. This model contains at least **1** hidden layer. You can decide how many hidden layers the model has and how many neural nodes exist in every hidden layer. You just need to find in the function “main()” of the “ANN.py” File. For example, you want two hidden layers which have 128 and 1024 nodes respectively, you can change it to :



And then, you can run this model like this( Supposing that you let the learning rate be 0.3 ):



In order to avoid the model being over-fitting, I use the test dataset to decide when the model stop learning. The judgement function as follows:



Once that the value of E has no longer decreased, the model stop learning.

**Result**

The Result are saved in the File “dataset/result.csv”. For one line,the **0-9th** value are attributes and real class(**normalized**), and the **10th** value is the predictive value(**normalized**).

1. the NN is better than Perceptron in terms of efficiency and accuracy.

But Perceptron is better than the NN in terms of efficiency.

When I use the same train and test set , let the learning rate be 0.3. We can see the results as follows:

|  |  |  |
| --- | --- | --- |
|  | Number of Iterations | accuracy |
| Neural Network | 51 | 1.0 |
| Perceptron | 15 | 0.96 |

(the Neural Network has 2 hidden layers(128,1024) )

In fact, the Perceptron is very similar to the neural network that has no hidden layer. Therefore, the Perceptron is just a linear function essentially. And the aim of the perceptron is to classify cases, but the aim of NN is to forecast a better value.

1. Yes.

When the initial threshold is far from the input of the node, the NN need more iterations.

When the learning rate is too big or too small, the NN need more iterations.

**Stage 2**：

1. In this experiment, I use 2 activation function: sigmoid and sign function. Because the value of real class is discrete, so when I use sigmoid function, I add a sign function after sigmoid in the output layer.

At same parameter, the former model need 13 iterators when the accuracy of “test” become 1.0 , and the latter model need 19 iterators when the accuracy of “test” become 1.0 . So the sigmoid function is better in terms of efficiency. According to some articles, the sigmoid function is also better in terms of accuracy.

As for hidden layers, I built 2 model( the learning rate = 0.3). The former model has **1** hidden layer(1024) and need 121 iterations of train . The latter model has **2** hidden layer(128, 1024) and need 51 iterations of train. More hidden layer, better efficiency and accuracy of model.

(2)Yes. At First, the both of them are made of many nodes connected. Secondly, the ouput and input all need “activation ” and threshold.

(3) I trained a autoencoder. The dimension of input and output is 5. The dataset can be seen in “dataset/Encoder.csv”, result can be seen in “dataset/result/encoder.csv”. If you want your own dataset, you can run the “gen\_encoder.py” as follow:



Then you can run the model as follow:



(4) more nodes the hidden layer have , better the accuracy and efficiency of the model.

The result as follow :

|  |  |  |
| --- | --- | --- |
| Number of nodes | Number of iterations | Accuracy |
| 20 | 46 | 0.36 |
| 50 | 33 | 0.47 |
| 100 | 22 | 0.57 |

Thinking:

The accuracy is not high, it may be a local optima rather than a global optima. But it really can prove that more nodes can improve the the accuracy and efficiency of the model.