Homework
神经网络与应用

Name:Li Shibo Student ID:119033910046 Email: ShiboLi@sjtu.edu.cn

Problem 1 Let $X = [w^T, b]^T, Z_q = [P_q^T, 1]^T$, where P_q is the input. Then $n = w^T p + b = X^T Z$. Let η denote learning rate.The percepton learning rule can now be written $X^{new} = X^{old} + e\eta Z$. We will assume that a weight vector exists that can be correctly categorize all Q input vectors and this solution will be denoted X^*. For the solution we will assume if $t_q = 1$,${X^*}^T Z_Q > \delta > 0$ (1) and if $t_q = 0$,${X^*}^T Z_Q < \delta < 0$ (2).

If we count only those iterations for which the weight vector is changed,the learning rule becomes $X(k) = X(k-1) + \eta Z'(k-1)$ (3),where $Z'(k-1)$ is the appropriate member of the set $\{Z_1, Z_2, \ldots Z_Q, -Z_1, -Z_2, \cdots - Z_Q\}$.

Assume without generality that the learning algorithm is initialized with $X(0) = 0$ (4).We can get $X(k) = \eta(Z'(0) + Z'(1) + \cdots + Z'(k-1))$ (5) from (3).From (1) and (2),we can get ${X^*}^T Z'(i) > \delta$ (6). From (5) and (6),we can get ${X^*}^T X(k) > k\eta\delta$ (7).

Using Cauthy-Schwartz inequality on (7), $({X^*}^T X(k))^2 \le ||X^*||^2 ||X(k)||^2$ (8) where $||X||^2 = X^T X$. With (7) and (8), $||X(k)||^2 \ge ({X^*}^T X(k))^2 / ||X^*||^2 > (k\eta\delta)^2 / ||X^*||^2$ (9).

We can easily get $X(k-1)^T Z'(k-1) \le 0$ (10) since the weights would not be updated unless the previous input vector had been misclassified. We also know $||X(k)||^2 = X(k)^T X(k) = [X(k-1) + \eta Z'(k-1)]^T [X(k-1) + \eta Z'(k-1)] = X(k-1)^T X(k-1) + 2\eta Z'(k-1)^T X(k-1) + \eta^2 Z'(k-1)^T Z'(k-1)$ (11). with (10), (11) can be simplified to $||X(k)||^2 \ge X(k-1)^T X(k-1) + \eta^2 Z'(k-1)^T Z'(k-1)$ (12).When we repeat this process, we will get $||X(k)||^2 \le \eta^2(||Z'(o)||^2 + ||Z'(1)||^2 + \ldots ||Z'(k-1)||^2)$ (13).Let $M = max||Z'(i)||^2$,Then $||X(k)||^2 \le k\eta M$. (14)

With (9) and (14), we cann get $\eta^2 kM \ge ||X(k)||^2 > (k\eta\delta)^2 / ||X^*||^2$ (15).By simplify, $k < M||X^*||^2/\delta^2$. Obviously,the maximum number of iterations has nothing to do with the learning rate.

Problem 2 Let C denote cost function,$z^(l)$ denote the input vector of $l - th$ layer,$a^(l)$ denote the active vector of $l - th$ layer.

According to the network,we can get $z^{(l)} = (a^{(l-1)})^T U^{(l)} a^{(l-1)} + V^{(l)} a^{(l-1)} + b^{(l)}$ (1) and $a^(l) = f(z^{(l)})$ (2).

Define cost function as $C_{(i)} = 1/m \sum_{r=1}^{m} ||a_r^{(L)} - y_r||$ and $C = 1/N^{(L)} \sum_{i=1}^{N^{(L)}} C_{(i)}$, where y_r is the label of $r - th$ sample and $N^{(L)}$ is the number of layers in the output layer.

Case 1: Neuron j is an output Node.

We define the local gradient $\delta_j(n)$ as:

$$\delta_j^{(L)} = \frac{\partial C}{\partial z_j^{(L)}} = \frac{\partial C}{\partial a_j^{(L)}}\frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} = \frac{\partial C}{\partial a_j^{(L)}}\frac{\partial f(z_j^{(L)})}{\partial z_j^{(L)}} = \frac{\partial C}{\partial a_j^{(L)}} f'(z^{(L)_j}) \tag{3}$$

Case 2: Neuron j is a Hidden Node.

$$\begin{aligned}
\delta_j^{(l)} &= \frac{\partial C}{\partial z_j^{(l)}} \\
&= \sum_{k=1}^{N^{(l+1)}} \frac{\partial C}{\partial z_k^{(l+1)}}\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \\
&= \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \sum_{k=1}^{N^{(l+1)}} \frac{\partial C}{\partial z_k^{(l+1)}}\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \\
&= f'(z_j^{(l)}) \sum_{k=1}^{N^{(l+1)}} \delta_k^{(l+1)} \frac{\partial \sum_{s=1}^{N^{(l)}} (u_{ks}^{(l+1)}(a^{(l)})_s^2 + v_{ks}^{(l+1)} a_s^{(l)}) + b_k^{(l+1)}}{\partial a_j^{(l)}} \\
&= f'(z_j^{(l)}) \sum_{k=1}^{N^{(l+1)}} \delta_k^{(l+1)} (2u_{kj}^{(l+1)} a_j^{(l)} + v_{kj}^{(l+1)})
\end{aligned} \tag{4}$$

The partial derivative for cost function:

$$\begin{aligned}
\frac{\partial C}{\partial u_{jk}^{(l)}} &= \frac{\partial C}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial u_{jk}^{(l)}} \\
&= \delta_j^{(l)} \frac{\partial \sum_{s=1}^{N^{(l)}} (u_{js}^{(l)}(a^{(l-1)})_s^2 + v_{js}^{(l)} a_s^{(l-1)}) + b_j^{(l)}}{\partial u_{jk}^{(l)}} \\
&= \delta_j^{(l)} (a^{(l-1)})_k^2
\end{aligned} \tag{5}$$

$$\begin{aligned}
\frac{\partial C}{\partial v_{jk}^{(l)}} &= \frac{\partial C}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial v_{jk}^{(l)}} \\
&= \delta_j^{(l)} \frac{\partial \sum_{s=1}^{N^{(l)}} (u_{js}^{(l)}(a^{(l-1)})_s^2 + v_{js}^{(l)} a_s^{(l-1)}) + b_j^{(l)}}{\partial v_{jk}^{(l)}} \\
&= \delta_j^{(l)} a_k^{(l-1)}
\end{aligned} \tag{6}$$

$$
\begin{aligned}
\frac{\partial C}{\partial b_j^{(l)}} &= \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} \\
&= \delta_j^{(l)} \frac{\partial \sum_{s=1}^{N^{(l)}} (u_{js}^{(l)} (a^{(l-1)})_s^2 + v_{js}^{(l)} a_s^{(l-1)}) + b_j^{(l)}}{\partial b_j^{(l)}} \\
&= \delta_j^{(l)} \qquad (7)
\end{aligned}
$$

Then we can update the parameter

$$u_{jk}^{(l)} = u_{jk}^{(l)} - \eta \frac{\partial C}{\partial u_{jk}^{(l)}} \qquad (8)$$

$$v_{jk}^{(l)} = v_{jk}^{(l)} - \eta \frac{\partial C}{\partial v_{jk}^{(l)}} \qquad (9)$$

$$b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial C}{\partial b_j^{(l)}} \qquad (10)$$

Then the learning algorithm is:
The forward pass: compute $z^{(l)}, a^{(l)}$ in each layer with equation (1) and (2).
The backward pass: compute the local gradient in output layer with (3); compute the local gradient in each hidden layer with (4) and update the weight u, v and the bias b with (8),(9),(10).
In the on-line learning,the weight and bias will be updated with each sample;But the batch learning will update the weight and bias with a number of samples;in this way,the equation will be altered as:

$$u_{jk}^{(l)} = u_{jk}^{(l)} - \frac{\eta}{N_s} \sum_{i=1}^{N_s} \frac{\partial C_i}{\partial u_{ijk}^{(l)}} \qquad (11)$$

$$v_{jk}^{(l)} = v_{jk}^{(l)} - \frac{\eta}{N_s} \sum_{i=1}^{N_s} \frac{\partial C_i}{\partial v_{ijk}^{(l)}} \qquad (12)$$

$$b_j^{(l)} = b_j^{(l)} - \frac{\eta}{N_s} \sum_{i=1}^{N_s} \frac{\partial C_i}{\partial b_{ij}^{(l)}} \qquad (13)$$

where N_s is the number of samples in each epoch.

Problem 3 I built a 3 layer perceptron network with the sigmoid active function both in hidden layer and output layer. But I found that when I setted different value of the learning rate and the number of hiddern layer the accurate rate of the result would be different.Even when I trained several times on the same number of hidden layers and learning rate,I would get different result too. I thought the reason is that the result did not converge to a global optimum; Maybe some time it did not converge to a local optimum; So the results are different each time. Here are some result of diferent learning rate and different number of hidden layers.

```
cuda:0
epoch num : 100 -- Loss : 1.0678400993347168 -- Acc : 0.564
epoch num : 200 -- Loss : 1.0564826726913452 -- Acc : 0.564
epoch num : 300 -- Loss : 1.0452615022659302 -- Acc : 0.564
epoch num : 400 -- Loss : 1.032552719116211 -- Acc : 0.564
epoch num : 500 -- Loss : 1.022934913635254 -- Acc : 0.564
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.5131195335276968
```

图 1: The result when the number of hidden layers is 300 and the learning rate is 0.001; The value of "acc" in the figuer is the matching rate in the training process in each epoc; The value in the bottom of the picture is the matching rate in the test process. The same below.

```
cuda:0
epoch num : 100 -- Loss : 1.080365538597107 -- Acc : 0.682
epoch num : 200 -- Loss : 1.0700528621673584 -- Acc : 0.462
epoch num : 300 -- Loss : 1.0624767541885376 -- Acc : 0.564
epoch num : 400 -- Loss : 1.056030035018921 -- Acc : 0.564
epoch num : 500 -- Loss : 1.0489943027496338 -- Acc : 0.656
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.7725947521865889
```

图 2: The result when the number of hidden layers is 310 and the learning rate is 0.001.

We can see easily,with different hidden layers,the result is different in the same learning rate.

```
cuda:0
epoch num : 100 -- Loss : 1.0846432447433472 -- Acc : 0.344
epoch num : 200 -- Loss : 1.0929690599441528 -- Acc : 0.436
epoch num : 300 -- Loss : 1.037062168121338 -- Acc : 0.66
epoch num : 400 -- Loss : 1.00754976272583 -- Acc : 0.436
epoch num : 500 -- Loss : 1.0584733486175537 -- Acc : 0.462
```

图 3: The result when the number of hidden layers is 320 and the learning rate is 0.01.

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.34402332361516036
```

图 4: The result when the number of hidden layers is 320 and the learning rate is 0.01.

```
cuda:0
epoch num : 100 -- Loss : 1.0792206525802612 -- Acc : 0.564
epoch num : 200 -- Loss : 1.0715432167053223 -- Acc : 0.564
epoch num : 300 -- Loss : 1.0645995140075684 -- Acc : 0.564
epoch num : 400 -- Loss : 1.0573903322219849 -- Acc : 0.564
epoch num : 500 -- Loss : 1.0504868030548096 -- Acc : 0.564
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.6822157434402333
```

图 5: The result when the number of hidden layers is 320 and the learning rate is 0.001.

We can see easily from figure 4 and figure 5,with different learning rate,the result is different in the same hidden layer.

```
cuda:0
epoch num : 100 -- Loss : 1.0820626020431519 -- Acc : 0.564
epoch num : 200 -- Loss : 1.07456374168396 -- Acc : 0.564
epoch num : 300 -- Loss : 1.0672413110733032 -- Acc : 0.564
epoch num : 400 -- Loss : 1.0615028142929077 -- Acc : 0.564
epoch num : 500 -- Loss : 1.0536856651306152 -- Acc : 0.564
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.5131195335276968
```

图 6: The result when the number of hidden layers is 315 and the learning rate is 0.001.

```
cuda:0
epoch num : 100 -- Loss : 1.0671052932739258 -- Acc : 0.564
epoch num : 200 -- Loss : 1.0552523136138916 -- Acc : 0.682
epoch num : 300 -- Loss : 1.0455245971679688 -- Acc : 0.682
epoch num : 400 -- Loss : 1.0342292785644531 -- Acc : 0.564
epoch num : 500 -- Loss : 1.0220879316329956 -- Acc : 0.564
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2]
----
0.6822157434402333
```

图 7: The result when the number of hidden layers is 315 and the learning rate is 0.001.

```
cuda:0
epoch num : 100 -- Loss : 1.0709600448608398 -- Acc : 0.58
epoch num : 200 -- Loss : 1.0609092712402344 -- Acc : 0.58
epoch num : 300 -- Loss : 1.0534906387329102 -- Acc : 0.682
epoch num : 400 -- Loss : 1.0454679727554321 -- Acc : 0.79
epoch num : 500 -- Loss : 1.0361320972442627 -- Acc : 0.79
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1]
----
0.8513119533527697
```

图 8: The result when the number of hidden layers is 315 and the learning rate is 0.001.

We can see easily from figure 6, figure 7 and figure 8,even with the same learning rate and the same hidden layer,the result is different in several training times.
Since the dataset is small, so I tried to use some loops to get a better result on different parameters. And saved the output with diffent parameters and the corresponding results in the file out.txt; Because I run the loop several times, there are several out.txt, such as out.txt,out0.txt. And I figured a graph on different parameters in the file.

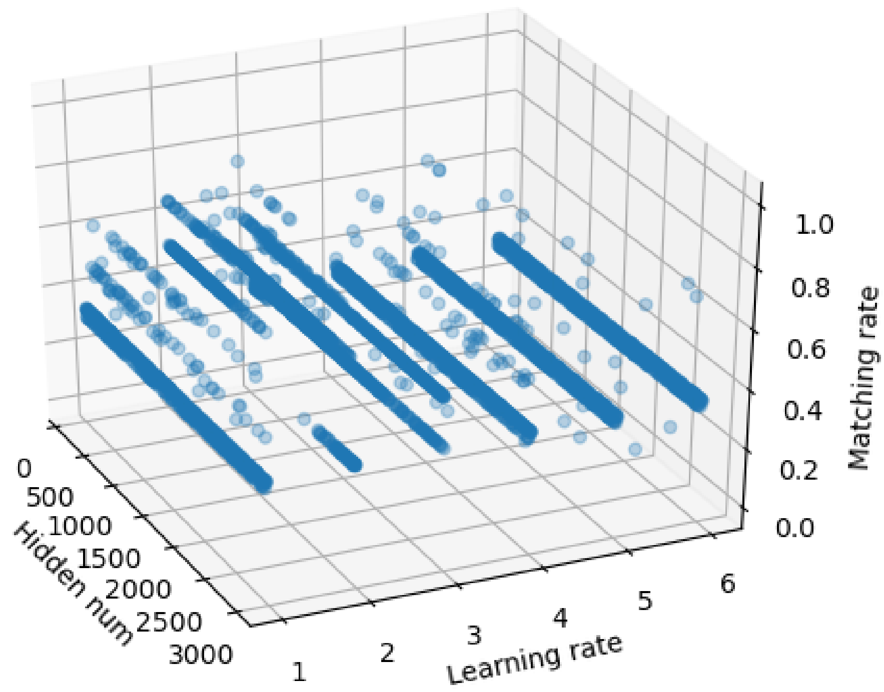

图 9: The result with different number of hidden layers is different learning rate. We represent 10^{-i} with i in the learning rate.

Obviously, with different number of hidden layers and different learning rate,we will get different result. Then I find the different result maybe caused by small number of iterations of the training procese; So I set the number of iterations to 500000 rather than 500,set the number of hidden layers 100 and set the learning rate with a small number 0.00001 because the large number of iterations. Then I got a better result.

```
epoch num : 499000 -- Loss : 0.549227237701416 -- Acc : 0.998
epoch num : 499100 -- Loss : 0.5491548776626587 -- Acc : 0.998
epoch num : 499200 -- Loss : 0.5490825772285461 -- Acc : 0.998
epoch num : 499300 -- Loss : 0.5490104556083679 -- Acc : 0.998
epoch num : 499400 -- Loss : 0.5489380955696106 -- Acc : 0.998
epoch num : 499500 -- Loss : 0.5488659143447876 -- Acc : 0.998
epoch num : 499600 -- Loss : 0.5487937927246094 -- Acc : 0.998
epoch num : 499700 -- Loss : 0.5487216114997864 -- Acc : 0.998
epoch num : 499800 -- Loss : 0.548649311065673 -- Acc : 0.998
epoch num : 499900 -- Loss : 0.5485771894454956 -- Acc : 0.998
epoch num : 500000 -- Loss : 0.5485048890113831 -- Acc : 0.998
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0]
----
0.8309037900874635
```

图 10: The number of iterations is 500000;The number of hidden layers is 100; The learning rate is 0.00001.