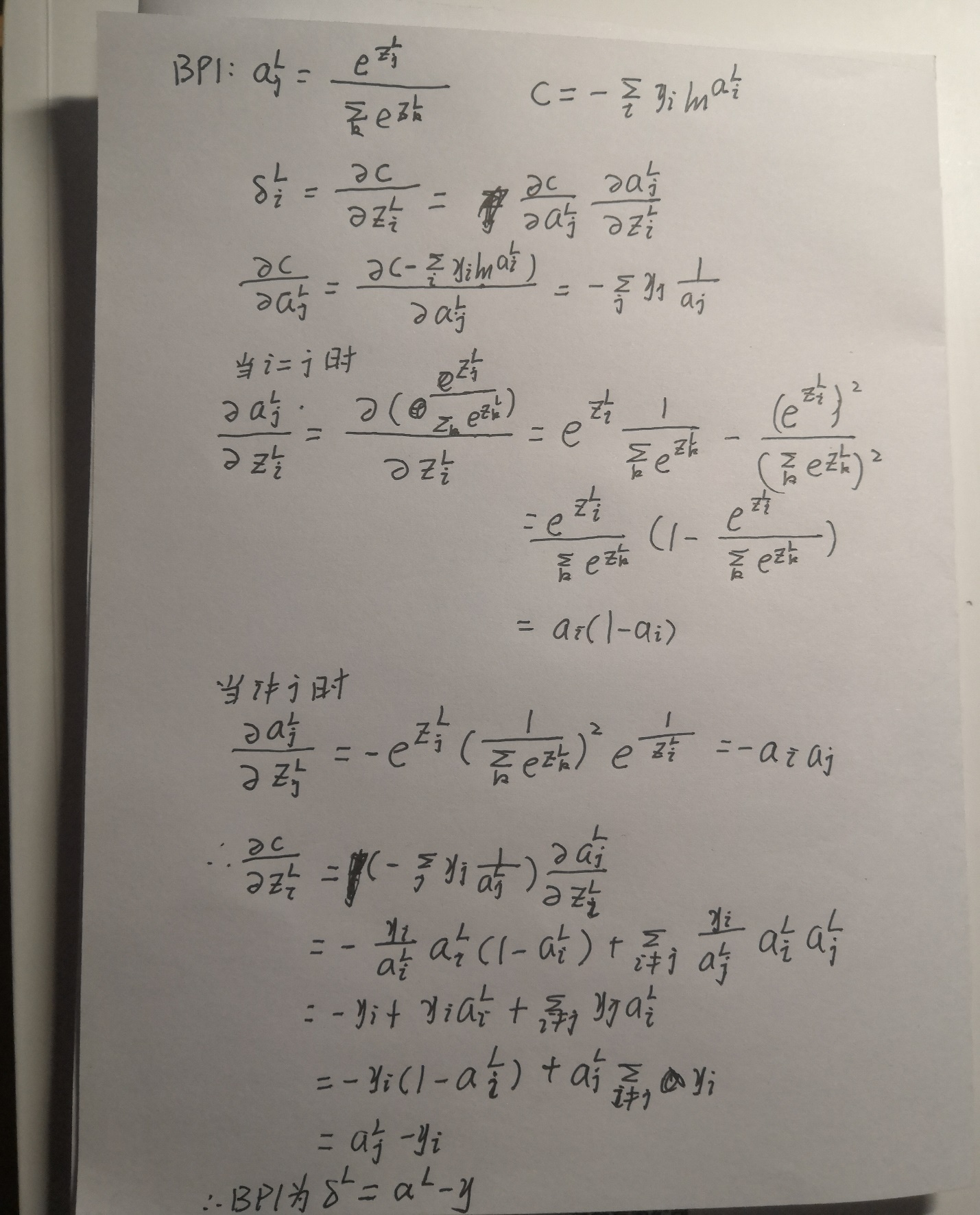
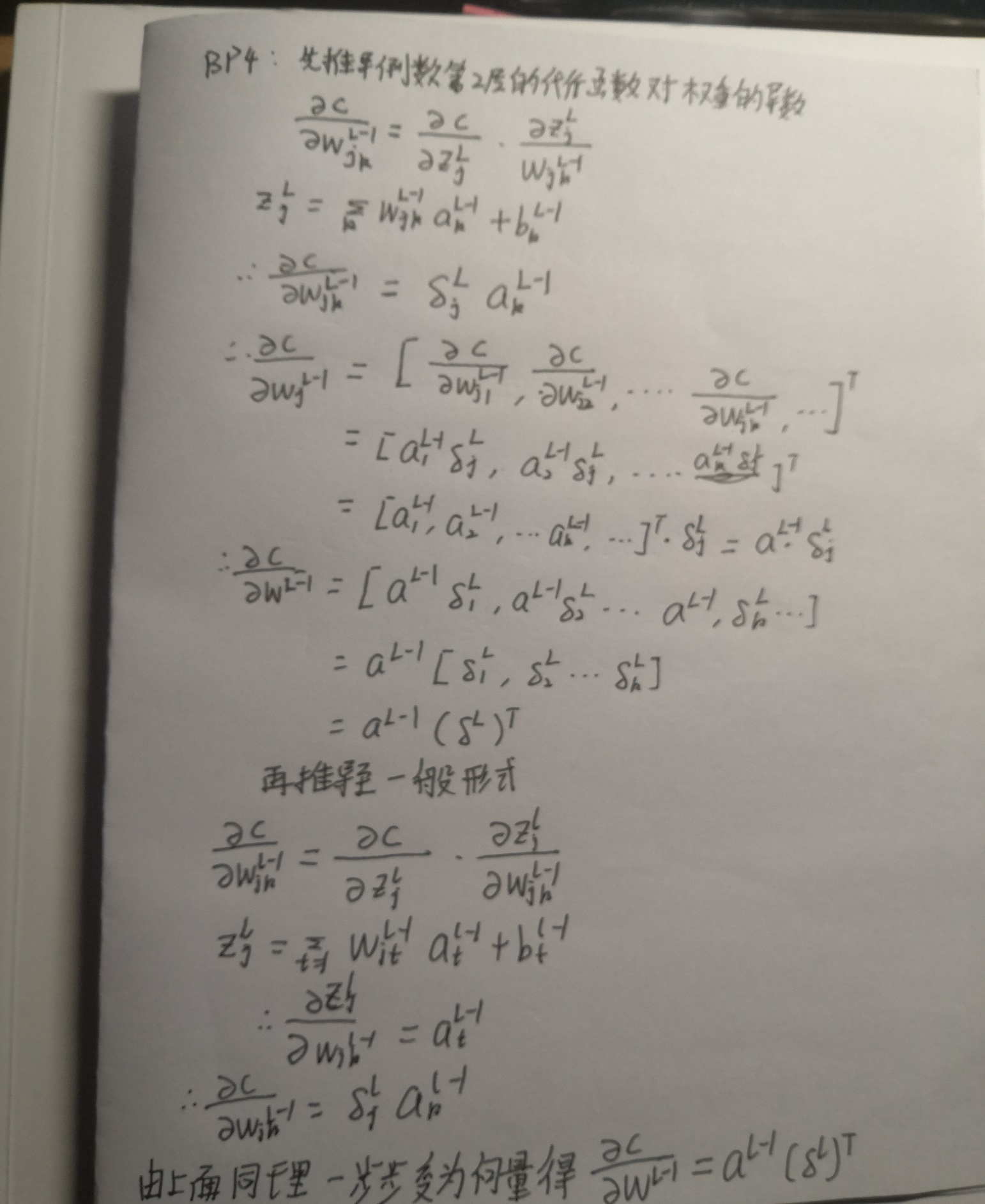
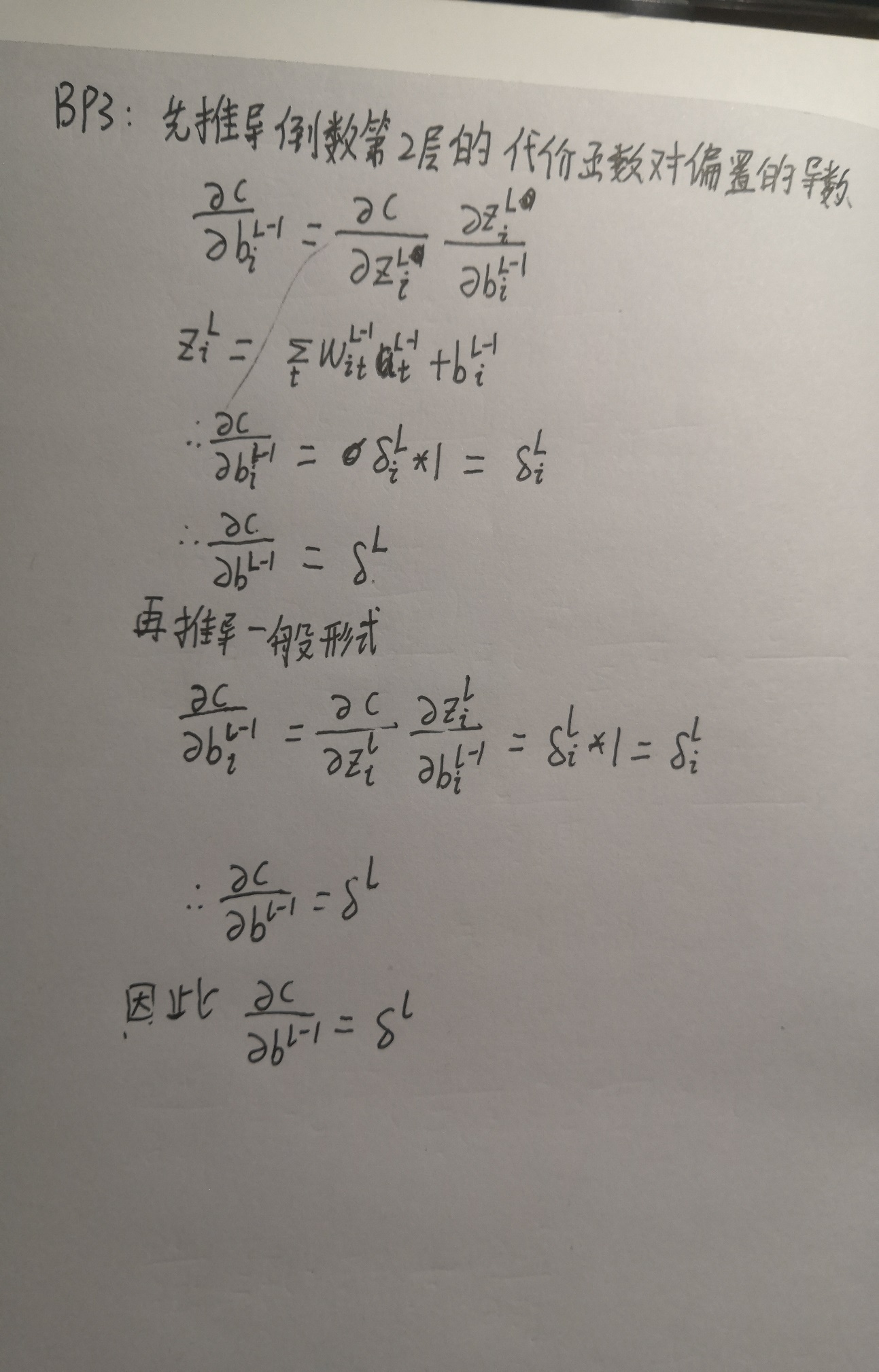
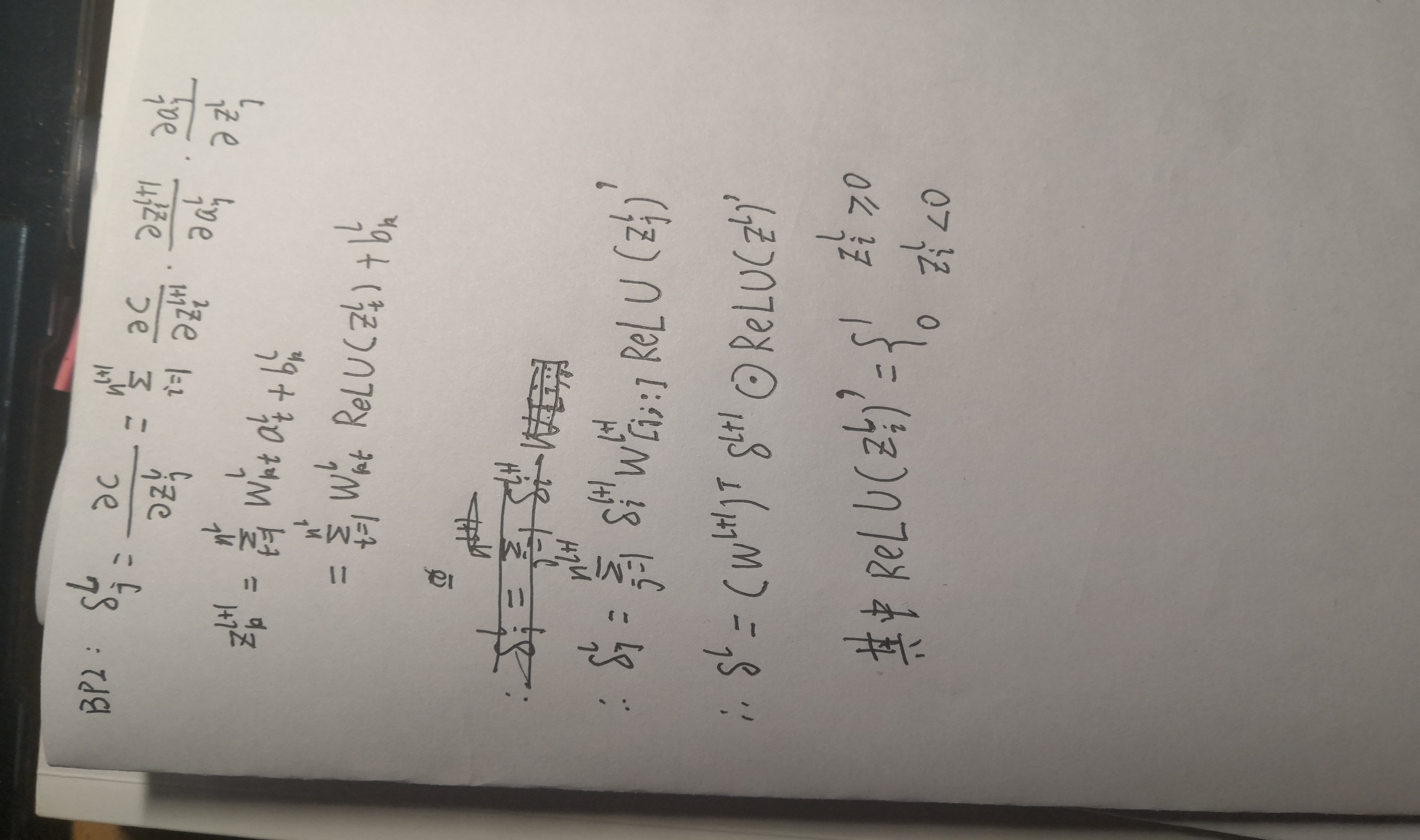
**机器学习第二次作业**

**1、BP公式推导**





**2、编程题**

（1）将sigmoid函数改为ReLU函数

def ReLU(z):  
 *"""ReLU函数"""* return np.where(z < 0, 0, z)  
  
  
def ReLU\_prime(z):  
 *"""ReLU函数的导数"""* return np.where(z < 0, 0, 1)

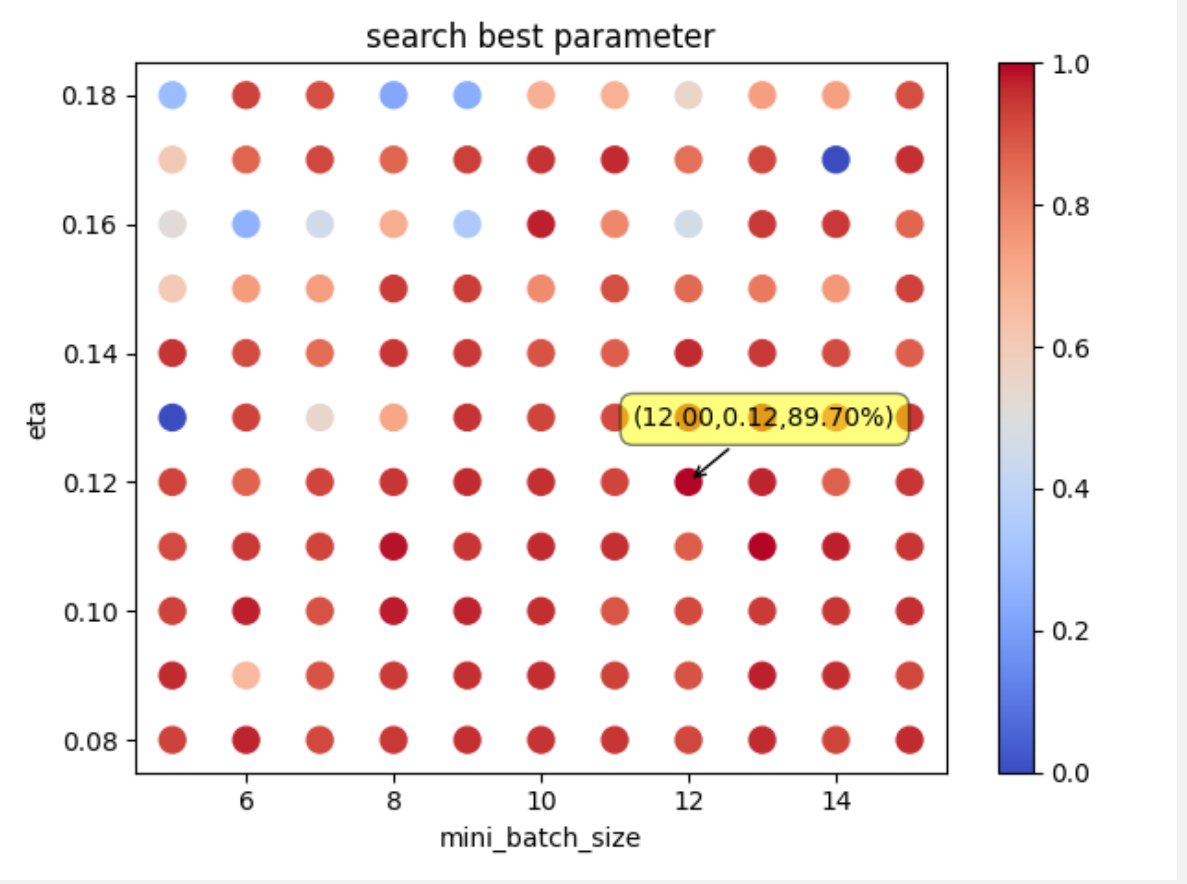
（2）更改损失为交叉熵损失，最后一层用softmax函数表示

def fn(a, y):  
 *"""交叉熵损失函数"""* sums = 0  
 for zi in a:  
 sums += np.exp(zi)  
 sums = np.log(sums)  
  
 return np.sum(-y \* (a - sums))

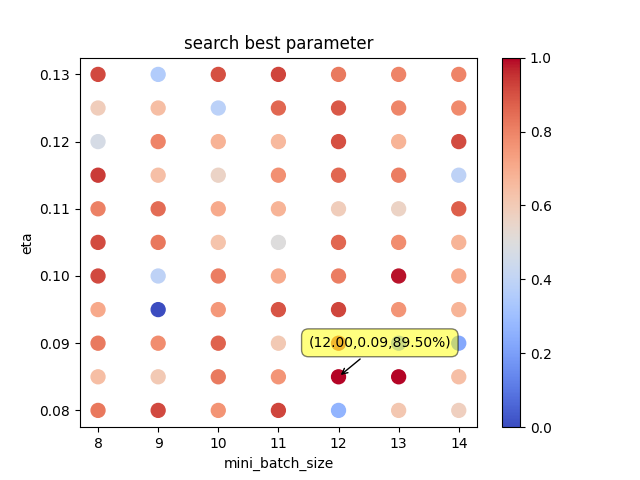
def feedforward(self, a):  
 *"""前向传播"""* for b, w in zip(self.biases[:-1], self.weights[:-1]):  
 a = ReLU(np.dot(w, a) + b)  
 a = np.dot(self.weights[-1], a) + self.biases[-1]  
 return a

（3）采用网格搜索寻找比较好的超参数

*# 网格搜索大致好的超参数*x\_scatter = np.arange(5, 16, 1)  
y\_scatter = np.linspace(0.08, 0.18, 11)  
results = {}  
*# x\_scatter = np.arange(5, 7, 1)  
# y\_scatter = np.linspace(0.08, 0.10, 3)  
# results = {}*for x in x\_scatter:  
 for y in y\_scatter:  
 training\_data, validation\_data, test\_data = load\_data.load\_data\_wrapper()  
 training\_data = training\_data[0:5000]  
 test\_data = test\_data[0:1000]  
 net = network.Network([784, 15, 10], cost=network.CrossEntropyCost)  
 res = net.SGD(training\_data, 10, x, y,  
 evaluation\_data=test\_data,  
 monitor\_training\_accuracy=True,  
 monitor\_evaluation\_accuracy=True)  
 accuracy = max([i / 1000 for i in res[1]])  
 results[(x, y)] = accuracy  
marker\_size = 100 *# default: 20*best\_point = max(results, key=results.get)  
best\_acc = max(results.values())  
worst\_acc = min(results.values())  
colors = [results[x] for x in results.keys()]  
colors = np.array(colors)  
colors = 1-((best\_acc - colors)/(best\_acc - worst\_acc))  
[Y, X] = np.meshgrid(y\_scatter, x\_scatter)  
  
X = X.reshape(len(X.reshape(-1, 1)))  
Y = Y.reshape(len(Y.reshape(-1, 1)))  
plt.figure()  
plt.scatter(X, Y, marker\_size, c=colors, cmap=plt.cm.coolwarm)  
plt.annotate(**'(%.2f,%.2f,%.2f%%)'** % (best\_point[0], best\_point[1], best\_acc\*100),  
 xy=best\_point, xytext=(-30, 30), textcoords=**'offset pixels'**,  
 bbox=dict(boxstyle=**'round,pad=0.5'**, fc=**'yellow'**, alpha=0.5),  
 arrowprops=dict(arrowstyle=**'->'**, connectionstyle=**'arc3,rad=0'**))  
plt.colorbar()  
plt.xlabel(**'mini\_batch\_size'**)  
plt.ylabel(**'eta'**)  
plt.title(**'search best parameter'**)  
plt.show()

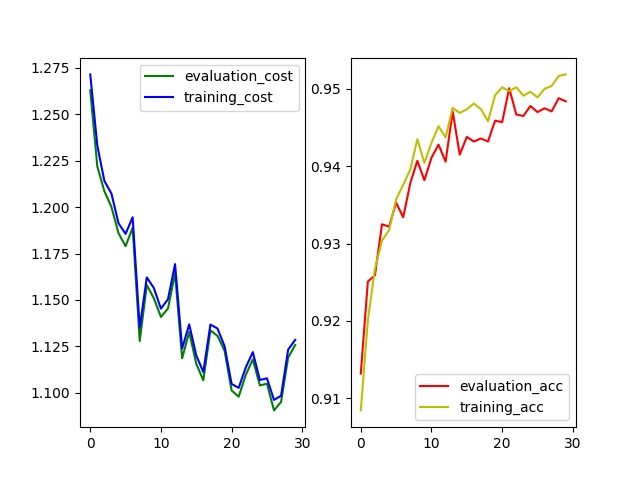


继续缩小超参数区间找到最好的大致区间



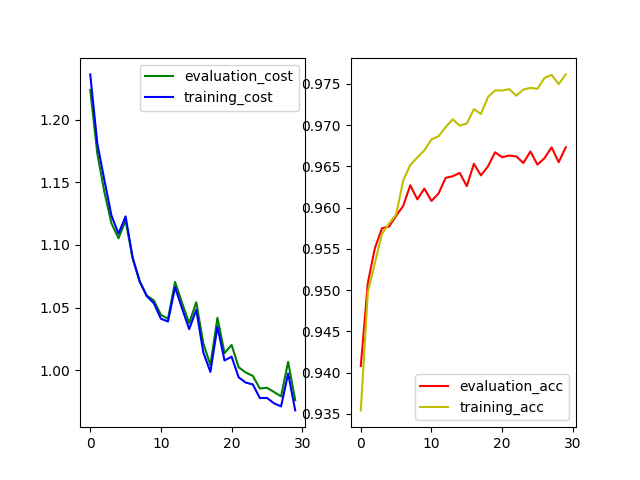
（4）采用隐含层有15个神经元的网络，在大致区间进行试验，找到最好的结果。

training\_data, validation\_data, test\_data = load\_data.load\_data\_wrapper()  
net = network.Network([784, 15, 10], cost=network.CrossEntropyCost)  
y = net.SGD(training\_data, 30, 9, 0.1, lambda1=0.3,  
 evaluation\_data=validation\_data,  
 monitor\_training\_cost=True,  
 monitor\_evaluation\_cost=True,  
 monitor\_training\_accuracy=True,  
 monitor\_evaluation\_accuracy=True)  
x = np.arange(30)  
y0 = [i for i in y[0]]  
y1 = [i/10000 for i in y[1]]  
y2 = [i for i in y[2]]  
y3 = [i/50000 for i in y[3]]  
plt.subplot(1, 2, 1)  
plt.plot(x, y0, **'g'**, label=**'evaluation\_cost'**)  
plt.plot(x, y2, **'b'**, label=**'training\_cost'**)  
plt.title(**"The change of cost"**)  
plt.legend()  
plt.subplot(1, 2, 2)  
plt.plot(x, y1, **'r'**, label=**'evaluation\_acc'**)  
plt.plot(x, y3, **'y'**, label=**'training\_acc'**)  
plt.title(**"The change of accuracy"**)  
plt.legend()

plt.show()  


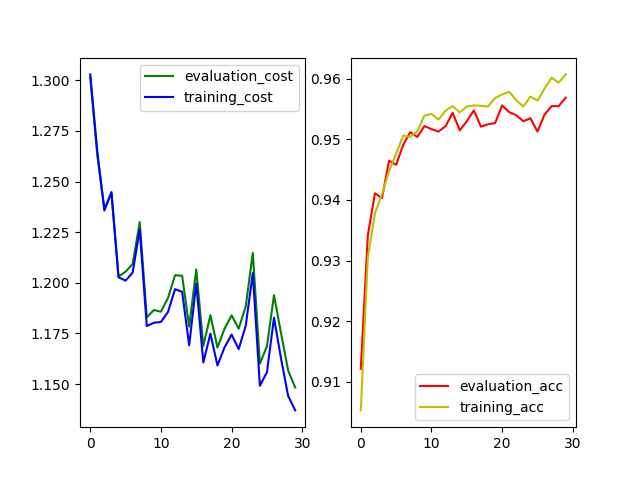
由图可知，损失函数在验证集和训练集上都在下降。并且精度在验证集和训练集上精度都接近95%，结果不错。

（5）将隐含层改为30个神经元



由图可以看出，损失函数得到了进一步的下降，训练集的精度提升到97.5%，但验证集的精度只是提升到了96.5%，出现了过拟合现象。此时可以通过正则化或者减小中间层的神经元数量来解决

（6）正则化



正则化加上降低隐藏层神经元，过拟合现象降低了。但感觉也不是很好。可能没找到更好的超参数