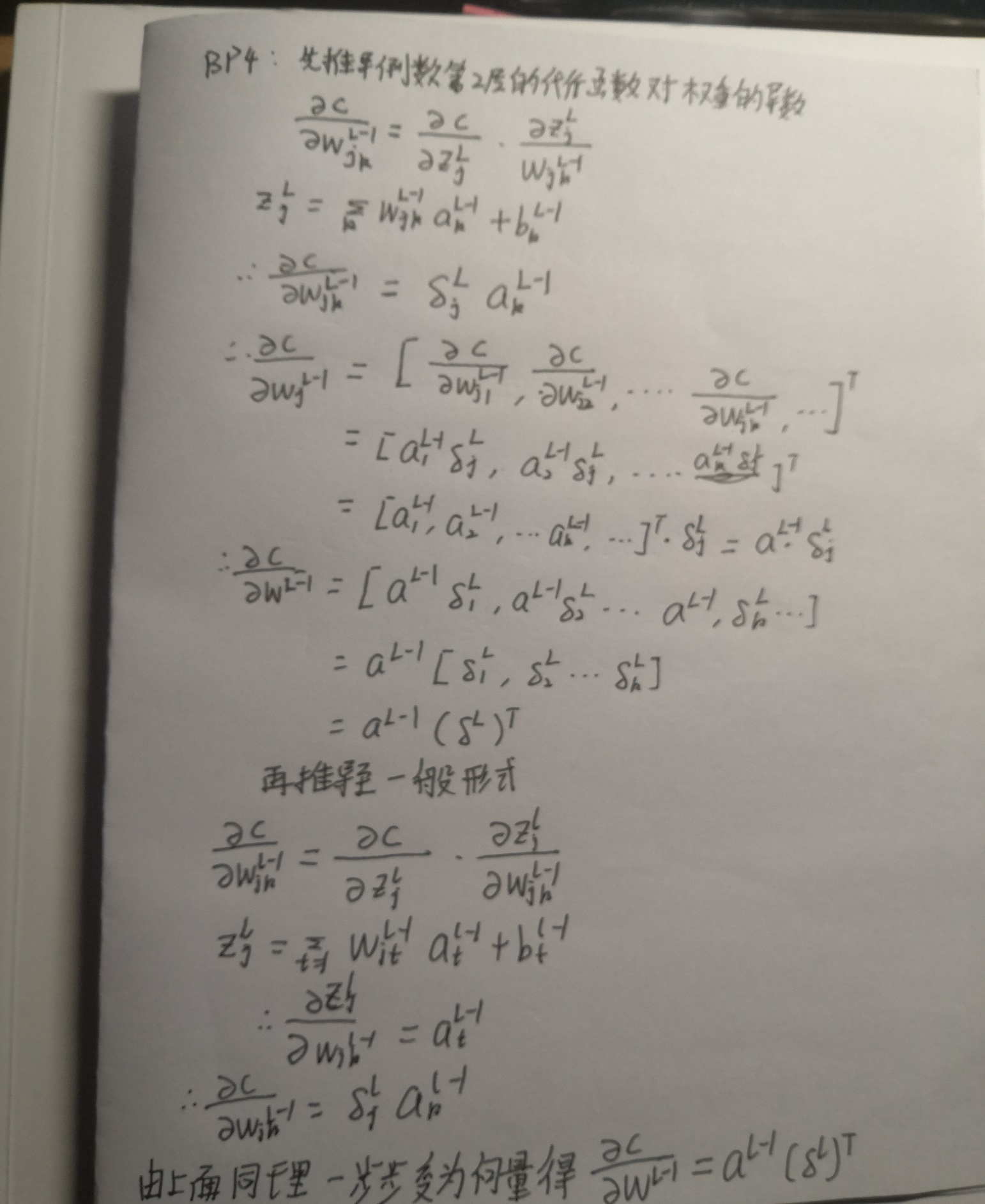
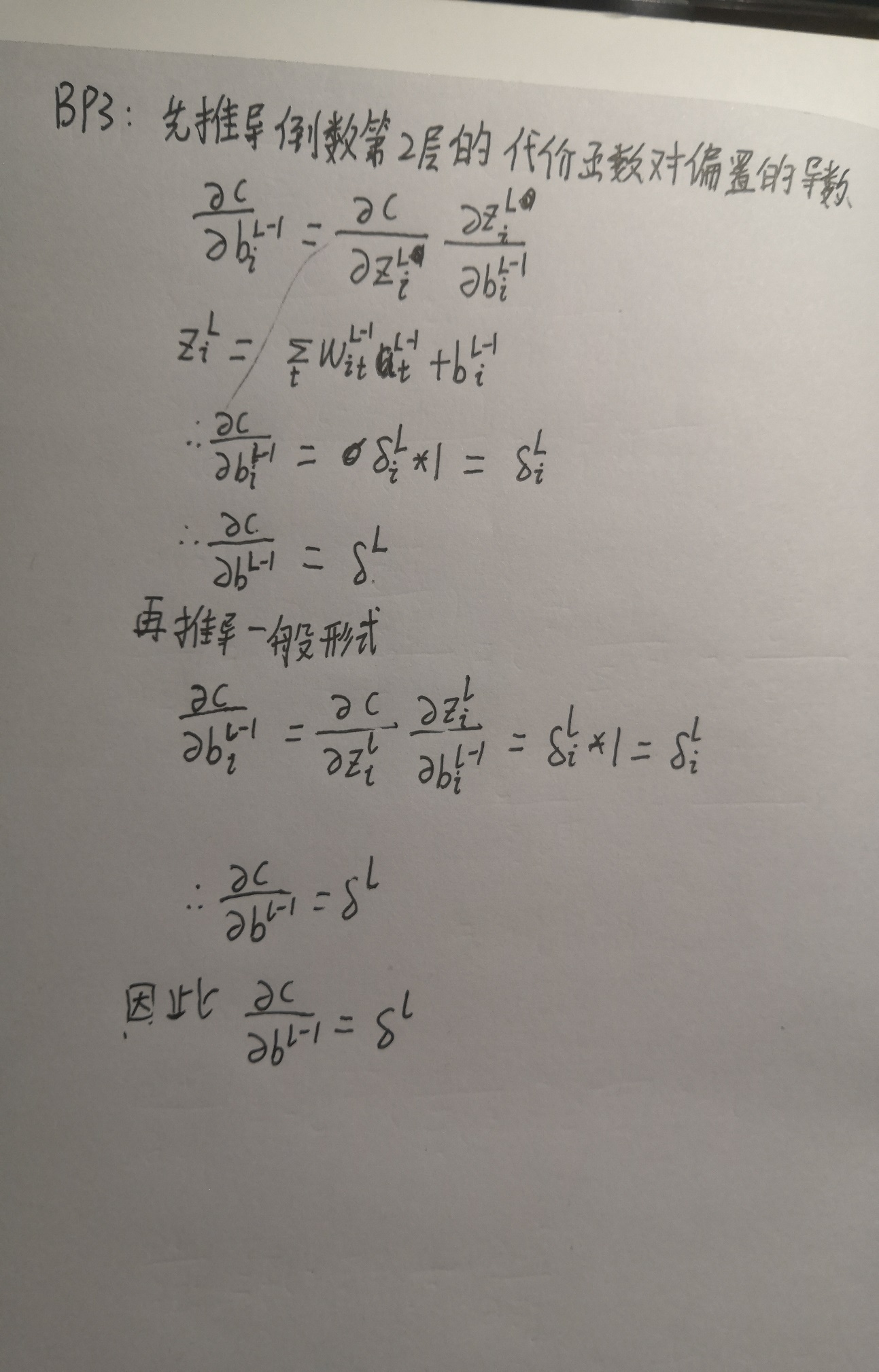
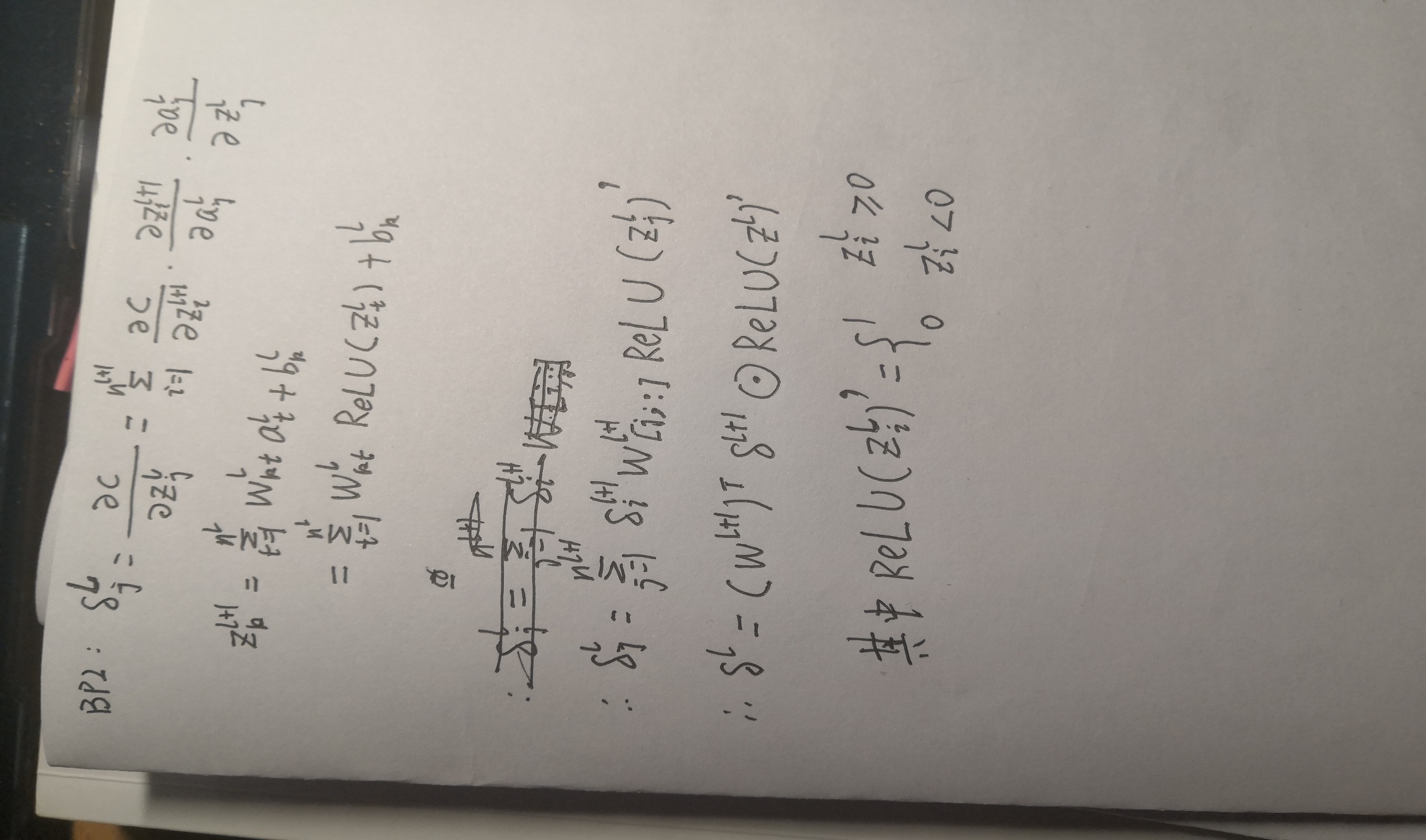
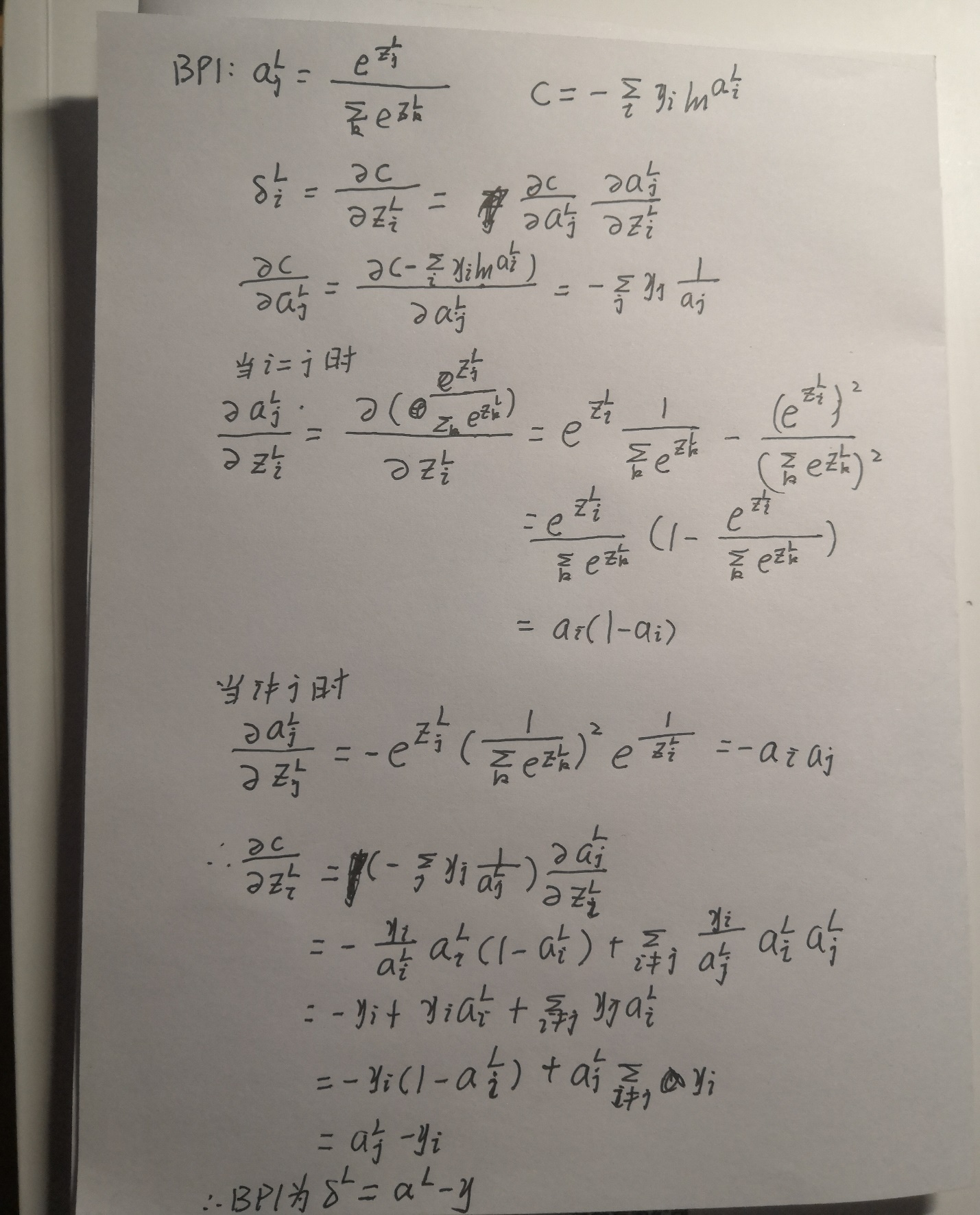
**机器学习第二次作业**

**1、BP公式推导**



**2、编程题**

（1）将sigmoid函数改为ReLU函数

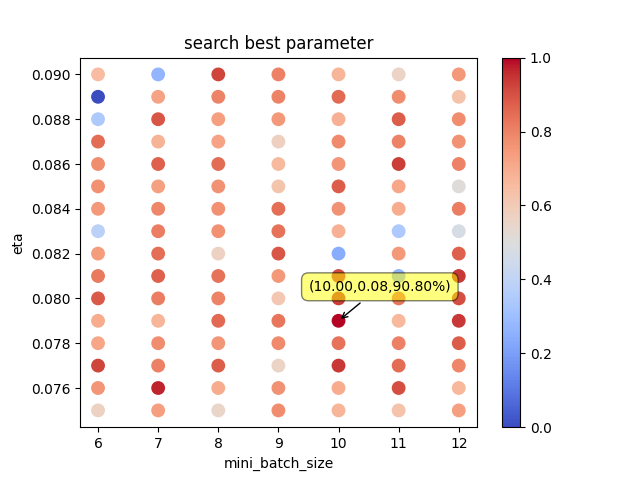
def ReLU(z):  
 *"""ReLU函数"""* return np.where(z < 0, 0, z)  
  
  
def ReLU\_prime(z):  
 *"""ReLU函数的导数"""* return np.where(z < 0, 0, 1)

（2）通过网格搜索找到超参数的大致值

x\_scatter = np.arange(5, 16, 1)  
y\_scatter = np.linspace(0.08, 0.18, 11)  
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, 20, 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 = 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),  
 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()

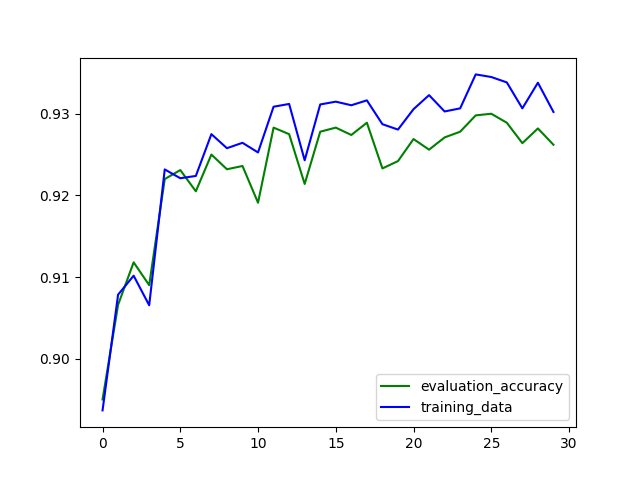


（3）缩小超参数范围，继续搜索更好的值



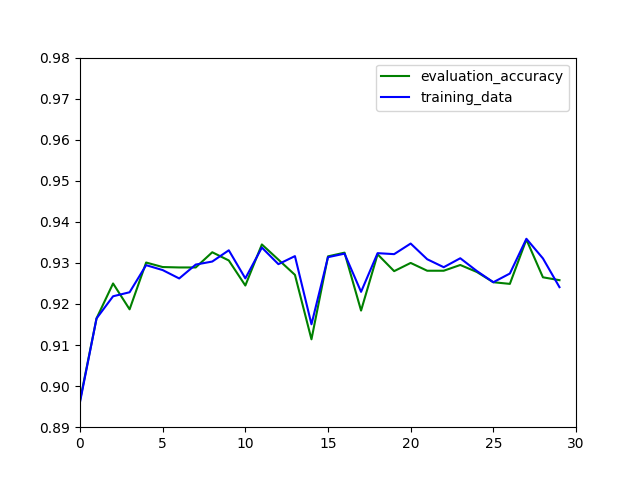
（4）采用隐含层有15个神经元的网络，得出初步结果

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, 0.08,  
 evaluation\_data=test\_data,  
 monitor\_training\_accuracy=True,  
 monitor\_evaluation\_accuracy=True)  
x = np.arange(30)  
y1 = [i/10000 for i in y[1]]  
y2 = [i/50000 for i in y[3]]  
plt.axis([0, 30, 0.89, 0.98])  
plt.plot(x, y1, **'g'**, label=**'evaluation\_accuracy'**)  
plt.plot(x, y2, **'b'**, label=**'training\_data'**)  
plt.legend()  
plt.show()



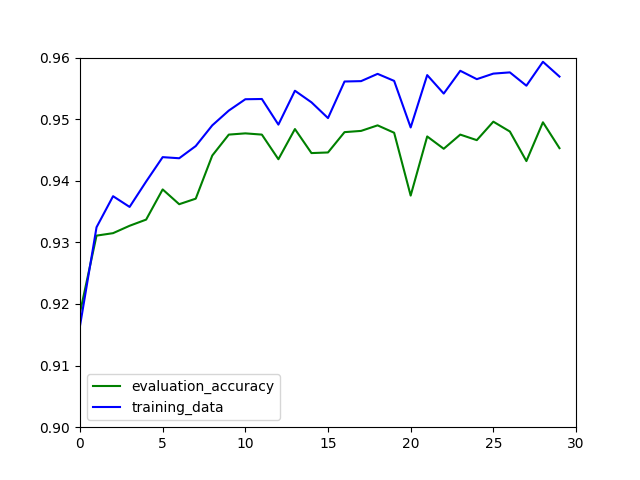
可以看出最后的查准率在93%左右，图像有轻微的过拟合现象

（5）增加正则化项，减少过拟合



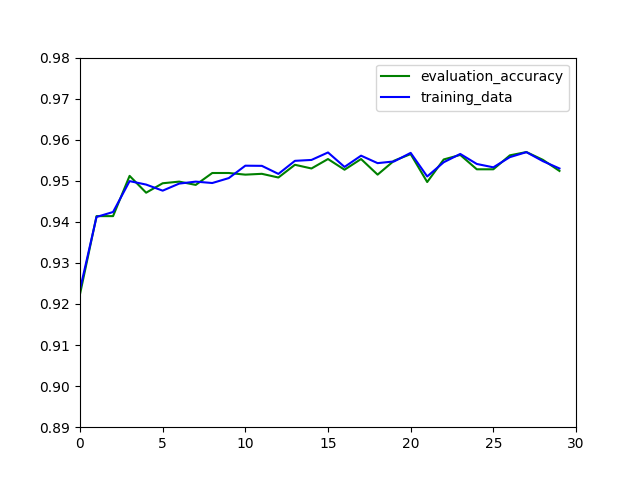
有图可以看出，过拟合现象减轻，查准率在93%左右

（6）将隐含层改为30个神经元



可以看出过拟合现象更严重，但是查准率上升到95%左右

（7）对三十个隐含神经元的网络增加正则化项



完美的解决了过拟合的现象，在验证集上表现不错，查准率提升到95%以上。