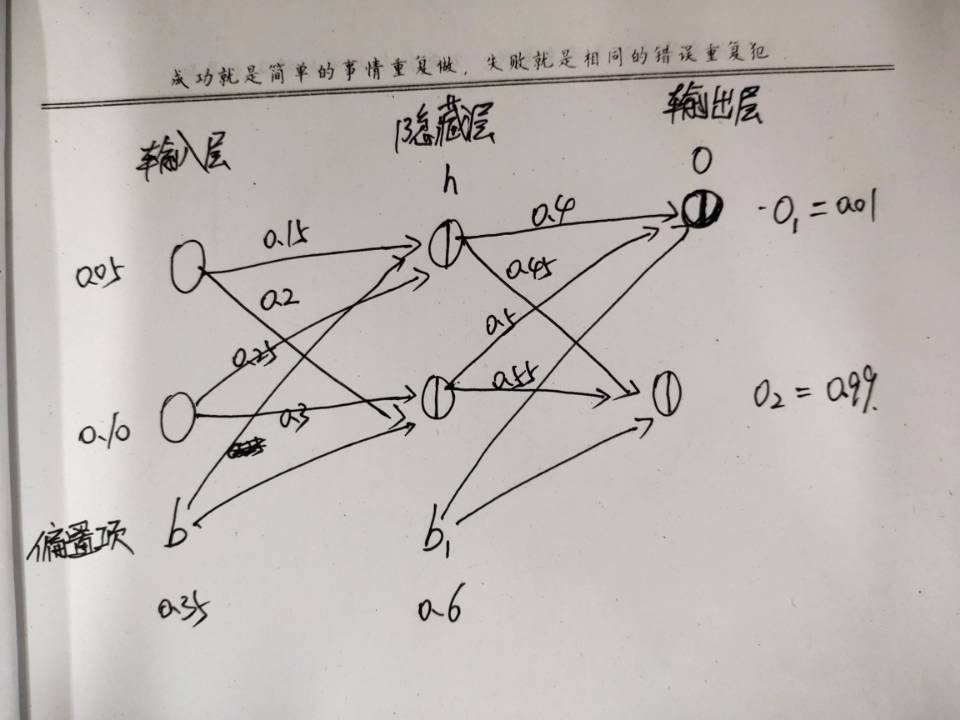
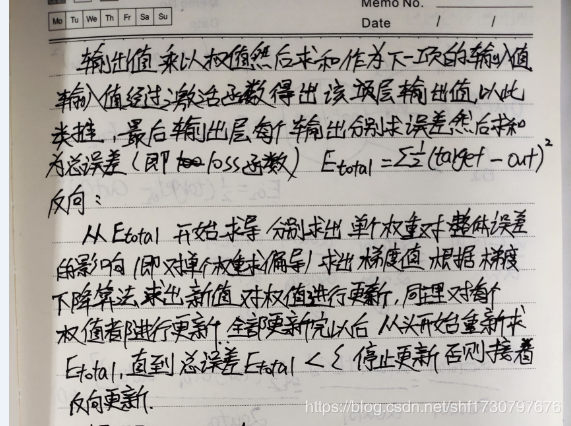
1. 神经网络及反向DP





代码：

import numpy as np

import matplotlib.pyplot as plt

import math

x=np.array([0.05,0.10])#输入层

w1=np.array([[0.15,0.25],[0.2,0.3]])#输入层到隐藏层（一）

w2=np.array([[0.4,0.5],[0.45,0.55]])#隐藏层到输出层（二）

y=np.array([0.01,0.99])#输出层

b1=0.35#（一）偏置

b2=0.6#（二）偏置

q=0.5#学习率

def neth(x,w1,b1):

return np.dot(x,w1)+b1

def outh():

outh=[]

for i in range(len(w1[0])):

a=1/(math.exp(-neth(x,w1,b1)[i])+1)#sigmod

outh.append(a)

return np.array(outh)

def neto():

return np.dot(outh(),w2)+b2

def outo():

outo=[]

for i in range(len(w2[0])):

a=1/(math.exp(-neto()[i])+1)

outo.append(a)

return np.array(outo)

s=0#迭代次数

num=[]#存储迭代次数值

result=[]#存储每次迭代中的损失值

while True:

s=s+1

dif=[[0,0],[0,0]]#初始化

dif1=[[0,0],

[0,0]]

a=0

#更新w5-w8

for i in range (len(w2)):

for j in range(len(w2[i])):

dif1[i][j]=(-(y[j]-outo()[j]))\*(outo()[j]\*(1-outo()[j]))\*(outh()[i])#梯度

w2[i][j]=w2[i][j]-q\*dif1[i][j]#梯度下降

#更新w1-w4

for i in range(len(w1)):

for j in range(len(outo())):

a=a+(-(y[j]-outo()[j]))\*outo()[j]\*(1-outo()[j])\*w2[i][j]

for k in range (len(w1[i])):

dif[i][k]=a\*(outh()[k]\*(1-outh()[k]))\*x[i]

w1[i][k]=w1[i][k]-q\*dif[i][k]

e=np.sum((y-outo())\*\*2/2)#损失值

result.append(e)#记录损失值

num.append(s)#记录迭代次数

if e<0.00001:

break

plt.plot(num,result)

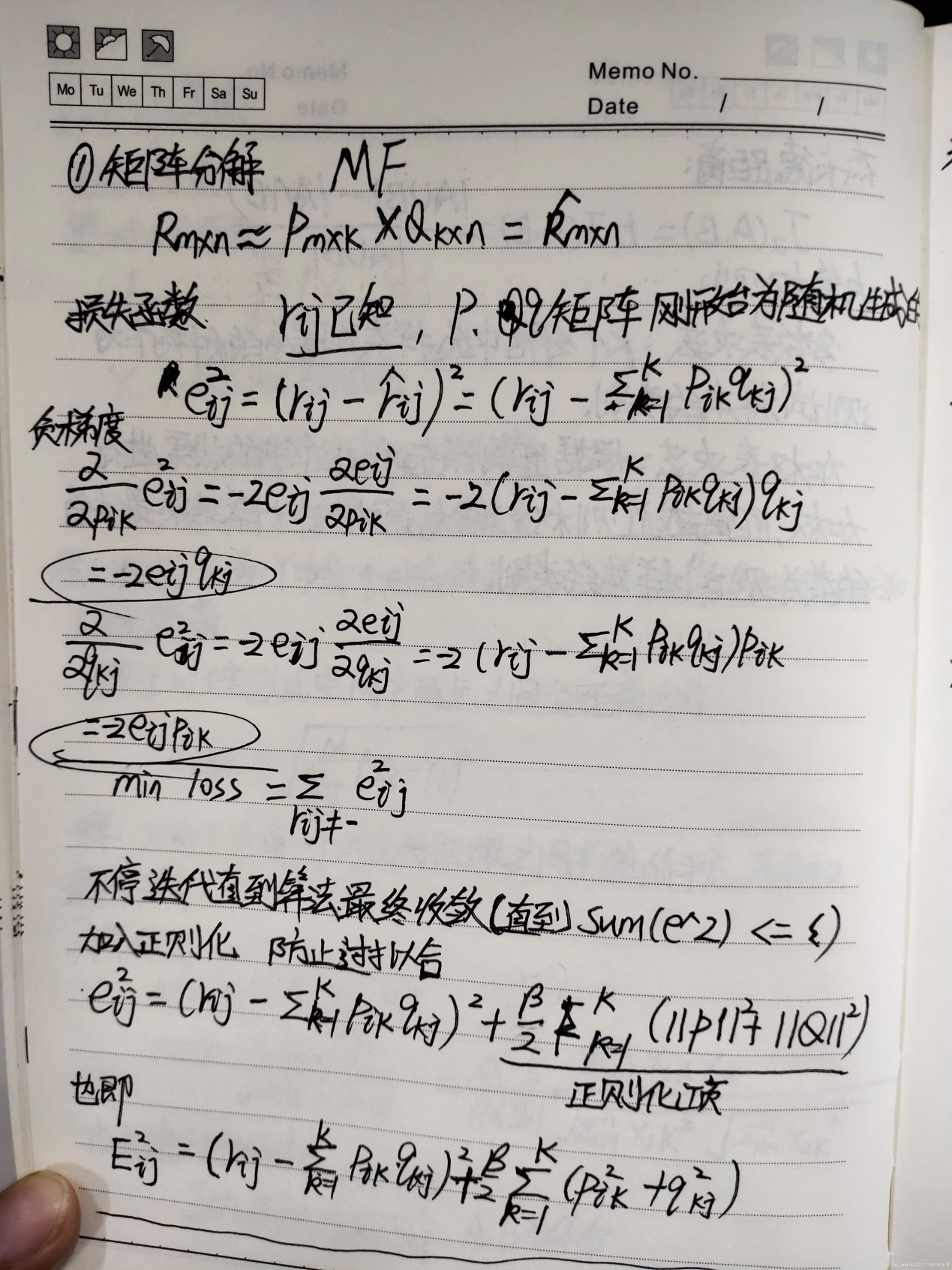
plt.show()

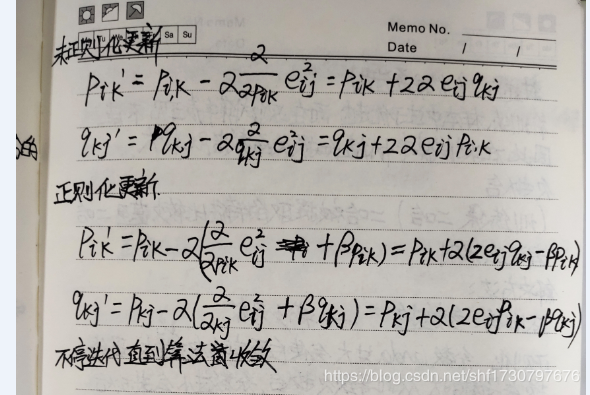
print(outo())

print(s)

print(e)

1. 矩阵分解





import matplotlib.pyplot as plt

from math import pow

import numpy

def matrix\_factorzation(R,P,Q,K,steps=5000,alpha=0.0002,beta=0.02):

Q=Q.T

result=[]

for step in range(steps):

for i in range(len(R)):

for j in range(len(R[i])):

if R[i][j]>0:

eij=R[i][j]-numpy.dot(P[i,:],Q[:,j])

for k in range(K):

P[i][k]=P[i][k]+alpha\*(2\*eij\*Q[k][j]-beta\*P[i][k])

Q[k][j]=Q[k][j]+alpha\*(2\*eij\*P[i][k]-beta\*Q[k][j])

eR=numpy.dot(P,Q)

e=0

for i in range(len(R)):

for j in range(len(R[i])):

if R[i][j]>0:

e=e+pow(R[i][j]-numpy.dot(P[i,:],Q[:,j]),2)

for k in range(K):

e=e+(beta/2)\*(pow(P[i][k],2)+pow(Q[k][j],2))

result.append(e)

if e<0.001:

break

return P,Q.T,result

if \_\_name\_\_=='\_\_main\_\_':

R=[[5,3,0,1],

[4,0,0,1],

[1,1,0,5],

[1,0,0,4],

[0,1,5,4]

]

R=numpy.array(R)

N=len(R)

M=len(R[0])

K=2

P=numpy.random.rand(N,K)

Q=numpy.random.rand(M,K)

nP,nQ,result=matrix\_factorzation(R,P,Q,K)

print("原始的评分矩阵R为：\n",R)

R\_MF=numpy.dot(nP,nQ.T)

print("经过MF算法评分后的矩阵是：\n",R\_MF)

#损失函数收敛曲线图

n=len(result)

x=range(n)

plt.plot(x,result,color='r',linewidth=3)

plt.title("Con")

plt.xlabel("generation")

plt.ylabel("loss")

plt.show()