**Report-Adaline-question 2:**

In this part I separate the results according to the number of the iterations of the loop.

I pick 0.1 for the learning rate(alpha=0.1)- because it was the larger impact from all the values I try to find for the learning rate.

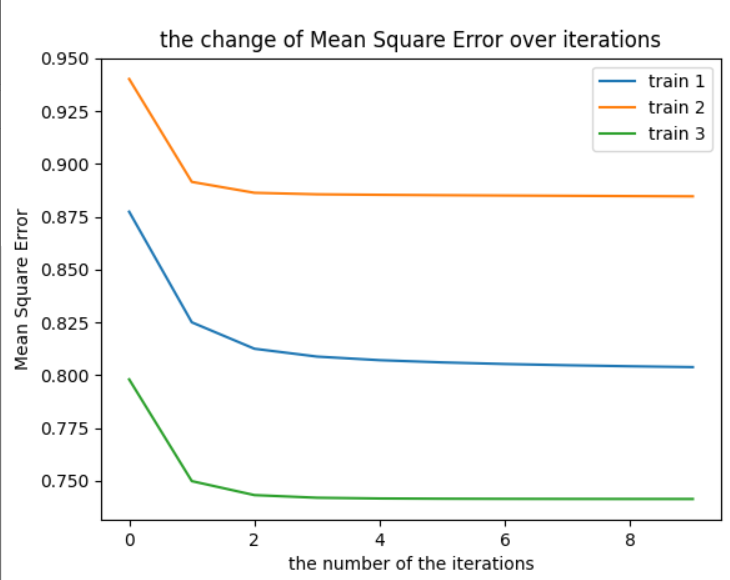
The bias input was 1.I initialized all the weights to zero- and where was ? in the data I change to 0 in order that will not impact my results(because we don’t know the input!).in train 1 I chose the first 66% of the data and in the test the last 33% of the data.in train 2 I chose the 66% of the data in the middle and in the test I took the 33% of the data in the start and in the end.in train 3 I chose the last 66% of the data and in the test I took the first 33% of the data. In the balance way I chose(in the reason that the positive data were few) to pick according to the order because they are proportion on the train and in the test(according to the order).

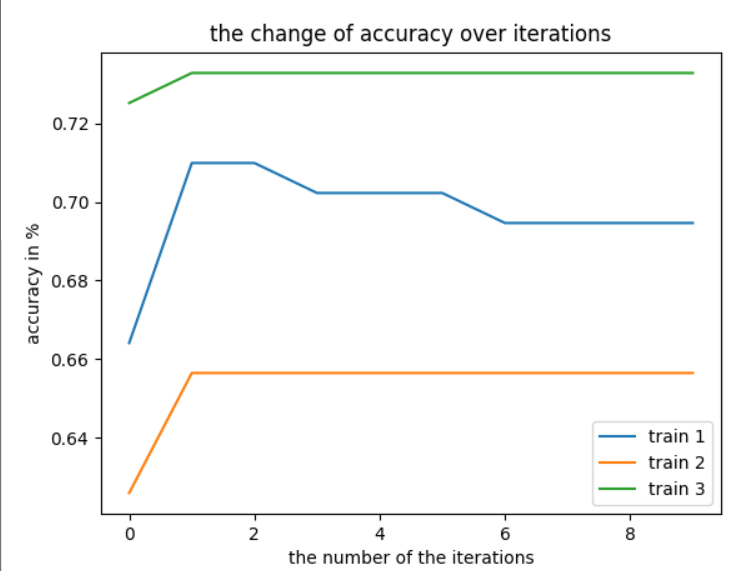
The data were impact so large on weights and more parameters-by increasing so fast their numbers to very big numbers that cause to overflow(in python-because the numbers were so big).so there was trouble with that. In the end I took all the data and diverse on 1500- in order that my numbers will be small in the calculations of the program.

my outputs are 1(positive-R) or -1(negative-N).

the architecture very simple-all the data are(one row)inputs that go to one neuron.

**Iterations=10:**



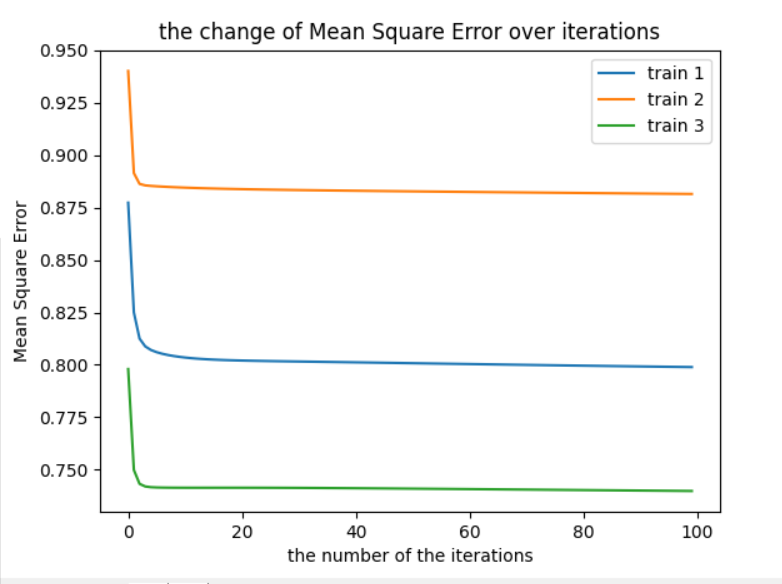


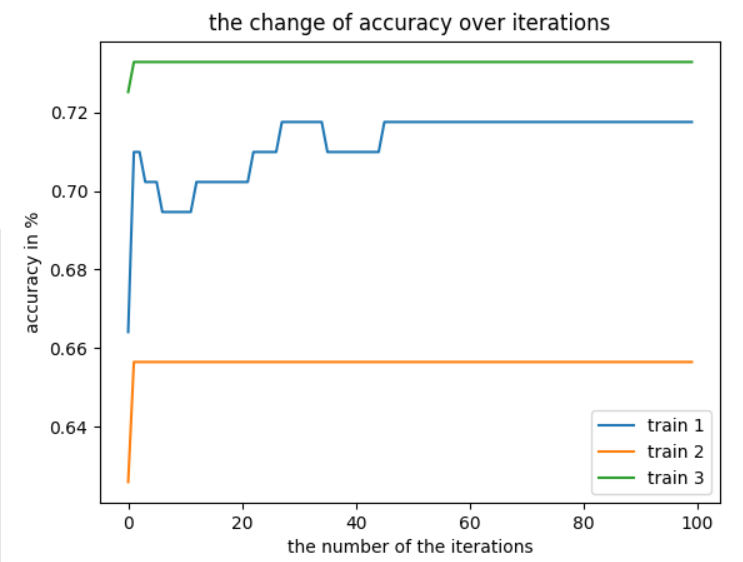
the accuracy of test 1 is(in %): 0.5454545454545454 Mean Square Error of test 1 is: 1.0485644848459152 the accuracy of test 2 is(in %): 0.8636363636363636 Mean Square Error of test 2 is: 0.529223266171007 the accuracy of test 3 is(in %): 0.7272727272727273 Mean Square Error of test 3 is: 0.7644839536273443 the average accuracy(3 tests): 0.712121212121212 the std accuracy(3 tests): 0.130338261621858

|  |  |  |
| --- | --- | --- |
| Predict: no | Predict: yes | **The 3 tests** |
| 0.1717(ave)/0.0724(std) | 0.0353(ave)/0.5(std) | Actual: yes |
| 0.6767(ave)/0.1768(std) | 0.1161(ave)/0.1642(std) | Actual: no |

the program run in seconds: 1.5792241096496582

**Iterations=100:**



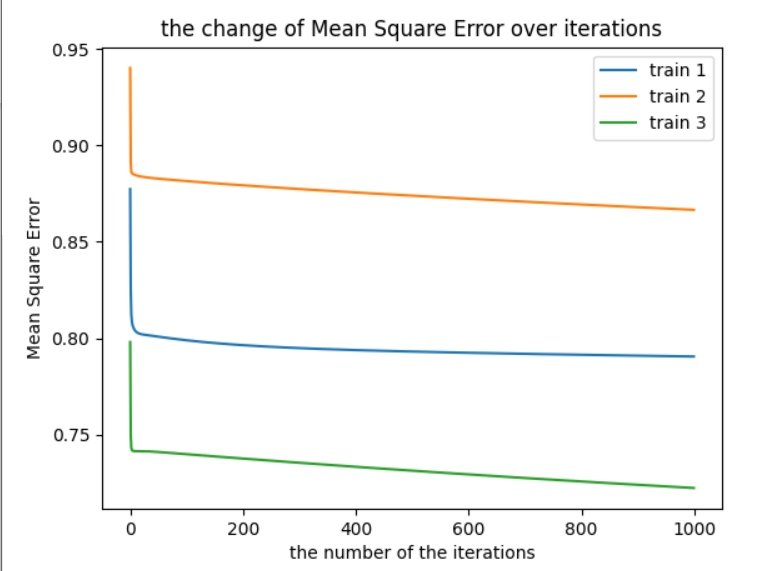


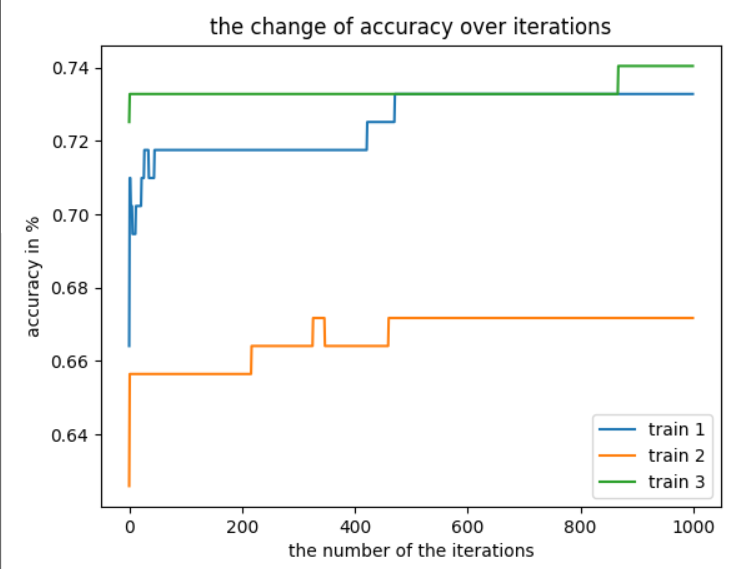
the accuracy of test 1 is(in %): 0.5303030303030303 Mean Square Error of test 1 is: 1.06757674349536 the accuracy of test 2 is(in %): 0.8484848484848485 Mean Square Error of test 2 is: 0.5341105042817393 the accuracy of test 3 is(in %): 0.7272727272727273 Mean Square Error of test 3 is: 0.7649573900177001 the average accuracy(3 tests): 0.702020202020202 the std accuracy(3 tests): 0.1311187372297694

|  |  |  |
| --- | --- | --- |
| Predict: no | Predict: yes | **The 3 tests** |
| 0.1717(ave)/0.0724(std) | 0.0353(ave)/0.5(std) | Actual: yes |
| 0.6666(ave)/0.1784(std) | 0.126262(ave)/0.1679(std) | Actual: no |

the program run in seconds: 2.8314764499664307

**Iterations=1000:**





the accuracy of test 1 is(in %): 0.5757575757575758 Mean Square Error of test 1 is: 1.0192478629330892 the accuracy of test 2 is(in %): 0.8484848484848485 Mean Square Error of test 2 is: 0.5294754254367217 the accuracy of test 3 is(in %): 0.7272727272727273 Mean Square Error of test 3 is: 0.763586755083950 the average accuracy(3 tests): 0.7171717171717171 the std accuracy(3 tests): 0.1115693032039117

|  |  |  |
| --- | --- | --- |
| Predict: no | Predict: yes | **The 3 tests** |
| 0.1616(ave)/0.08237(std) | 0.04545(ave)/0.06428(std) | Actual: yes |
| 0.6717(ave)/0.1715(std) | 0.1212(ave)/0.1608(std) | Actual: no |

the program run in seconds: 14.655659914016724

between the 101 iterations until 999 I don’t treat to in the reason that their result are similar with the iterations=100 and 1000(but with little bit difference).

**Conclusions:**

As we can see- for every kind of iteration we see that we have almost the same results. we can see we have high learning in the start for everyone and after that they become a little bit constants(I talk about of course the iterations between 1-1000).

**Code(python):**

import math  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib.colors import ListedColormap  
import time  
indication=0  
#a=alpha  
#Sampels\_X-inputs matrix  
#Sampels\_y-outputs array  
#num\_iter-number of iterations  
#W-weights  
def fit(num\_iter,a,Sampels\_X,Sampels\_Y):  
 X=[]  
 Y=[]  
 W=np.zeros(len(Sampels\_X[0])+1)  
 listACC = []  
 listERSQ = []  
 listITER = []  
 for it in range(0,num\_iter):  
 AERQ = 0  
 ACC = 0  
 for i in range(0,len(Sampels\_X)):  
 sum=0  
 Error=0  
 for j in range (0,len(Sampels\_X[i])):  
  
 sum=(W[j]\*Sampels\_X[i][j]+sum)  
 sum=W[len(W)-1]\*1+sum  
  
 Error=((Sampels\_Y[i]-sum))  
 AERQ=AERQ+((Error)\*\*2)  
 if sum >= 0:  
 tf = 1  
 else:  
 tf = -1  
 if tf == Sampels\_Y[i]:  
 ACC = ACC + 1  
 for j in range(0,len(Sampels\_X[i])):  
 W[j]=W[j]+(a\*Error\*Sampels\_X[i][j])  
 W[len(W)-1]=W[len(W)-1]+(a\*Error)  
 ACC=(ACC/len(Sampels\_Y))  
 AERQ = (AERQ / len(Sampels\_Y))  
 if indication == 0:  
 listACC.append(ACC)  
 listERSQ.append(AERQ)  
 listITER.append(it)  
 elif indication == 1:  
 listACC.append(ACC)  
 listERSQ.append(AERQ)  
 listITER.append(it)  
 elif indication == 2:  
 listACC.append(ACC)  
 listERSQ.append(AERQ)  
 listITER.append(it)  
  
 return [listACC,listERSQ,listITER],W  
  
  
def test(W,Sampels\_X,Sampels\_Y):  
  
 sumT = 0  
 AERQ = 0  
 pa1 = 0  
 pa2 = 0  
 pa3 = 0  
 pa4 = 0  
 for i in range(len(Sampels\_X)):  
 sum=0  
 tf=0  
 for j in range(len(Sampels\_X[i])):  
 sum=Sampels\_X[i][j]\*W[j]+sum  
 sum=(W[len(W)-1]\*1)+sum  
  
 if sum>=0:  
 py=1  
 if py==Sampels\_Y[i]:  
 sumT=sumT+1  
 pa1=pa1+1  
 else:  
 pa3=pa3+1  
 else:  
 py=-1  
 if py==Sampels\_Y[i]:  
 sumT=sumT+1  
 pa4=pa4+1  
 else:  
 pa2=pa2+1  
 AERQ = AERQ + ((Sampels\_Y[i] - sum) \* (Sampels\_Y[i]-sum))  
 pa1=(pa1/len(Sampels\_Y))  
 pa2 = (pa2 / len(Sampels\_Y))  
 pa3 = (pa3 / len(Sampels\_Y))  
 pa4 = (pa4 / len(Sampels\_Y))  
 a= (sumT/(len(Sampels\_Y)))  
 b=(AERQ/len(Sampels\_Y))  
 c=[pa1,pa2,pa3,pa4]  
 return a,b,c  
  
  
  
  
  
start\_time=time.time()  
file=pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc.data')  
len\_rows\_train=(int)(0.666666\*len(file))  
line\_y=file.iloc[0:len(file),[1]]  
  
  
line\_y=np.where(line\_y=='N',-1,1)  
temp=[]  
  
for i in range(len(line\_y)):  
 temp.append(line\_y[i][0])  
  
  
  
line\_y=np.array(temp)  
  
  
  
  
  
  
  
  
  
ave1=[]  
ave2=[]  
ave3=[]  
len\_col\_end=len(file.columns)  
len\_col\_start=3  
time\_of\_train\_test=0  
while(time\_of\_train\_test<3):  
  
 a=0.1  
 iter=100  
 if time\_of\_train\_test==0:  
 x = np.array(file.iloc[0:len\_rows\_train, len\_col\_start:len\_col\_end])  
 y = np.zeros(len\_rows\_train)  
 for i in range(0, len\_rows\_train):  
 y[i] = line\_y[i]  
  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x[i][j] = float(x[i][j])  
 if x[i][j] == '?':  
 x[i][j] = 0  
 x[i][j] = float(x[i][j])/1500  
  
 ave1,W = fit(iter, a, x, y)  
  
 x = np.array(file.iloc[len\_rows\_train:len(file), len\_col\_start:len\_col\_end])  
 y = np.zeros(len(file)-len\_rows\_train)  
 for i in range(0,len(file)-len\_rows\_train):  
 y[i] = line\_y[i+len\_rows\_train]  
  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x[i][j] = float(x[i][j])  
 if x[i][j] == '?':  
 x[i][j] = 0  
 x[i][j] = float(x[i][j])/1500  
 acc, erq,ta1 = test(W, x, y)  
 print("the accuracy of test 1 is(in %): ", acc)  
 print("Mean Square Error of test 1 is: ", erq)  
 finalav1 = acc  
  
 elif time\_of\_train\_test==1:  
  
 x = np.array(file.iloc[int((len(file) - len\_rows\_train)/2):len\_rows\_train+(int((len(file) - len\_rows\_train)/2)), len\_col\_start:len\_col\_end])  
 y = np.zeros(len\_rows\_train)  
 for i in range(0, len\_rows\_train):  
 y[i] = line\_y[i+int((len(file) - len\_rows\_train)/2)]  
  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x[i][j] = float(x[i][j])  
 if x[i][j] == '?':  
 x[i][j] = 0  
 x[i][j] = float(x[i][j])/1500  
  
  
 ave2,W = fit(iter, a, x, y)  
  
 x1 = np.array(file.iloc[0:int((len(file) - len\_rows\_train)/2), len\_col\_start:len\_col\_end])  
 y1 = np.zeros(int((len(file) - len\_rows\_train)/2))  
 for i in range(0, len(y1)):  
 y1[i] = line\_y[i]  
  
 for i in range(len(x1)):  
 for j in range(len(x1[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x1[i][j] = float(x1[i][j])  
 if x1[i][j] == '?':  
 x1[i][j] = 0  
 x1[i][j] = float(x1[i][j])/1500  
  
 x2 = np.array(file.iloc[len\_rows\_train+int((len(file) - len\_rows\_train) / 2):len(file), len\_col\_start:len\_col\_end])  
 y2 = np.zeros(int((len(file) - len\_rows\_train) / 2))  
 for i in range(0, len(y2)):  
 y2[i] = line\_y[i+len\_rows\_train+int((len(file) - len\_rows\_train) / 2)]  
  
 for i in range(len(x2)):  
 for j in range(len(x2[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x2[i][j] = float(x2[i][j])  
 if x2[i][j] == '?':  
 x2[i][j] = 0  
 x2[i][j] = float(x2[i][j])/1500  
  
  
 x=np.zeros([len(file)-len\_rows\_train,len(x1[1])])  
 y=np.zeros(len(file)-len\_rows\_train)  
 for i in range(len(x1)):  
 for j in range(len(x1[i])):  
 x[i][j]=x1[i][j]  
 y[i]=y1[i]  
  
 for i in range(len(x2)):  
 for j in range(len(x2[i])):  
 x[i+len(x1)][j] = x2[i][j]  
 y[i+len(y1)] = y2[i]  
  
  
  
  
 acc,erq,ta2 = test(W, x, y)  
 print("the accuracy of test 2 is(in %): ",acc)  
 print("Mean Square Error of test 2 is: ",erq)  
 finalav2 = acc  
  
  
 elif time\_of\_train\_test==2:  
  
 x = np.array(file.iloc[len(file)-len\_rows\_train:len(file), len\_col\_start:len\_col\_end])  
 y = np.zeros(len\_rows\_train)  
 for i in range(0, len\_rows\_train):  
 y[i] = line\_y[i+len(file)-len\_rows\_train]  
  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x[i][j] = float(x[i][j])  
 if x[i][j] == '?':  
 x[i][j] = 0  
 x[i][j] = float(x[i][j])/1500  
  
  
 ave3,W = fit(iter, a, x, y)  
  
 x = np.array(file.iloc[0:len(file)-len\_rows\_train, len\_col\_start:len\_col\_end])  
 y = np.zeros(len(file) - len\_rows\_train)  
 for i in range(0, len(file) - len\_rows\_train):  
 y[i] = line\_y[i]  
  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if j == 2 or j == 3 or j == 22 or j == 23:  
 x[i][j] = float(x[i][j])  
 if x[i][j] == '?':  
 x[i][j] = 0  
 x[i][j] = float(x[i][j])/1500  
 acc,erq,ta3 = test(W, x, y)  
 finalav3=acc  
 print("the accuracy of test 3 is(in %): ", acc)  
 print("Mean Square Error of test 3 is: ", erq)  
 print("the average accuracy(3 tests): ", ((finalav3+finalav2+finalav1)/3))  
 arr=[finalav1,finalav2,finalav3]  
 print("the std accuracy(3 tests): " ,(np.std(arr)))  
 vv1=((ta1[0]+ta2[0]+ta3[0])/3)  
 sta1=math.sqrt((((ta1[0]-vv1)\*\*2)+((ta2[0]-vv1)\*\*2)+((ta3[0]-vv1)\*\*2))/3)  
 vv2=((ta1[1] + ta2[1] + ta3[1]) / 3)  
 sta2 = math.sqrt((((ta1[1] - vv2) \*\* 2) + ((ta2[1] - vv2) \*\* 2) + ((ta3[1] - vv2) \*\* 2)) / 3)  
 vv3=((ta1[2] + ta2[2] + ta3[2]) / 3)  
 sta3 = math.sqrt((((ta1[2] - vv3) \*\* 2) + ((ta2[2] - vv3) \*\* 2) + ((ta3[2] - vv3) \*\* 2)) / 3)  
 vv4=((ta1[3] + ta2[3] + ta3[3]) / 3)  
 sta4=math.sqrt((((ta1[3] - vv4) \*\* 2) + ((ta2[3] - vv4) \*\* 2) + ((ta3[3] - vv4) \*\* 2)) / 3)  
 print("the average of actual-yes and predict-yes between the 3 tests is: ",(vv1))  
 print("Standard deviation of actual-yes and predict-yes between the 3 tests is: ",sta1)  
 print("the average of actual-yes and predict-no between the 3 tests is: ", vv2)  
 print("Standard deviation of actual-yes and predict-no between the 3 tests is: ", sta2)  
 print("the average of actual-no and predict-yes between the 3 tests is: ", vv3)  
 print("Standard deviation of actual-no and predict-yes between the 3 tests is: ", sta3)  
 print("the average of actual-no and predict-no between the 3 tests is:: ",vv4)  
 print("Standard deviation of actual-no and predict-no between the 3 tests is: ", sta4)  
 indication = indication + 1  
 time\_of\_train\_test=time\_of\_train\_test+1  
  
  
  
end\_time=time.time()  
print("the program run in seconds: ",end\_time-start\_time)  
plt.plot(ave1[2], ave1[0], label="train 1")  
plt.plot(ave2[2], ave2[0], label="train 2")  
plt.plot(ave3[2], ave3[0], label="train 3")  
plt.xlabel('the number of the iterations')  
plt.ylabel('accuracy in %')  
plt.title('the change of accuracy over iterations')  
plt.legend()  
plt.show()  
  
plt.plot(ave1[2], ave1[1], label="train 1")  
plt.plot(ave2[2], ave2[1], label="train 2")  
plt.plot(ave3[2], ave3[1], label="train 3")  
plt.xlabel('the number of the iterations')  
plt.ylabel('Mean Square Error')  
plt.title('the change of Mean Square Error over iterations')  
plt.legend()  
plt.show()