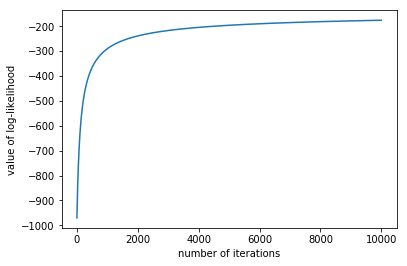
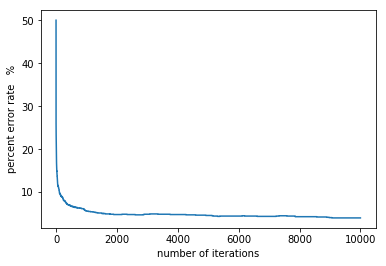
Problem5

**I used gradient ascent.**

(a)training





The matrix for w is:

[ 0.677176110296 0.837988208016 1.02897820256 0.858759026259 0.920163520683 0.0877989644628 -0.843836170376 -1.39076220877 ]

[ -0.16737709961 -0.189154431201 -0.23340269903 0.178262063564 -0.0798308862385 -0.384401624795 0.656451246636 0.484198500979 ]

[ -1.37149185068 -1.04840366919 -0.953427686647 -0.489106940123 -0.104375191115 1.30933008987 2.31012010485 1.95128890051 ]

[ -1.1592823966 -0.727734239566 -0.707899185586 0.236507809017 0.744897694378 0.484555014415 -0.0571002536073 0.190798656182 ]

[ -0.270963089095 -0.218822765155 -0.0332651117464 0.300957979726 0.339240148135 0.380101812828 0.230012996273 0.381375580432 ]

[ -0.678800255071 0.386024639265 -0.159903282624 -0.420027447177 -0.289586811682 0.156444922722 0.118716986102 0.959774875572 ]

[ -0.330075649406 -0.099220399782 -0.547629953998 -0.434140750533 -0.0424865678767 0.0695981502347 -0.302118920825 0.757544937218 ]

[ 0.125916110304 -0.35866248494 -0.499689056858 -0.938555366832 -0.357284974985 -0.477092513256 0.181894386209 0.220750263351 ]

(b)testing

error rate on test set is 4.75 %

**source code:**

import numpy as np

import scipy.optimize

import random

from math import exp

from math import log

def inner(x,y):

return sum([x[i]\*y[i] for i in range(len(x))])

def sigmoid(x):

return 1.0/(1+exp(-x))

#import data train\_3 and train\_5 into train

train = []

with open('hw5\_train3.txt') as inputfile:

for line in inputfile:

train.append(line.strip().split(' '))

with open('hw5\_train5.txt') as inputfile:

for line in inputfile:

train.append(line.strip().split(' '))

train = [list(map(int,x)) for x in train]

y = [1]\*700+[0]\*700

#the first 700 are handwriting for "3"

#the last 700 are handwriting for "5"

#we use 1(positive) example to represents handwriting "3"

#use 0(negarive) example to represents handwriting "5"

def loglikelihood(w):

sum\_likelihood = 0

for i in range(len(train)):

z=inner(w,train[i])

sum\_likelihood += y[i]\*log(sigmoid(z))+(1-y[i])\*log(1-sigmoid(z))

return sum\_likelihood

yita = 0.02/len(train)

def update\_w(w):

derivative = np.zeros(64) #derivative of L to w

for i in range(len(train)):

z=inner(w,train[i])

coef = y[i]-sigmoid(z)

x\_i = np.array(train[i])

derivative += x\_i\*coef

w = w + yita\*derivative

return w

def error\_rate(w):

predict = []

for i in range(len(train)):

predict.append(inner(w,train[i]))

judge = [1 if predict[i]>0 else 0 for i in range(len(train))]

error = [(a!=b) for (a,b) in zip(judge,y)]

error = sum(error)/len(train)\*100#percentage error rate

return error

w=np.zeros(64)

total\_times = 10000 #run 10000 times

deriv\_value=[]

error\_value=[]

deriv\_value.append(loglikelihood(w))

error\_value.append(error\_rate(w))

for j in range(1,total\_times):

w = update\_w(w)

deriv\_value.append(loglikelihood(w))

error\_value.append(error\_rate(w))

w1=[x for x in range(0,63,8)]

for i in w1:

print('[',w[i],w[i+1],w[i+2],w[i+3],w[i+4],w[i+5],w[i+6],w[i+7],']')

import matplotlib.pyplot as plt

x\_label=[i for i in range(total\_times)]

plt.plot(x\_label,deriv\_value)

plt.xlabel('number of iterations')

plt.ylabel('value of log-likelihood')

plt.show()

plt.plot(x\_label,error\_value)

plt.xlabel('number of iterations')

plt.ylabel('percent error rate %')

plt.show()

#testing

#import data test\_3 and test\_5 into train

test = []

with open('hw5\_test3.txt') as inputfile:

for line in inputfile:

test.append(line.strip().split(' '))

with open('hw5\_test5.txt') as inputfile:

for line in inputfile:

test.append(line.strip().split(' '))

test = [list(map(int,x)) for x in test]

y\_test=[1]\*400+[0]\*400

predict = []

for i in range(len(test)):

z=inner(w,test[i])

if z>0:

predict.append(1)

else:

predict.append(0)

error\_test = [(a!=b) for (a,b) in zip(predict,y\_test)]

error\_test = sum(error\_test)/len(test)\*100#percentage error rate

print('error rate on test set is ',error\_test,'%')