# -\*- coding: utf-8 -\*-

"""

Created on Wed Sep 28 18:36:08 2016

@author: Akash Rastogi

"""

import numpy as np

import scipy.io as sio

import matplotlib.pyplot as plt

import os

os.getcwd()

mnist\_49\_3000 = sio.loadmat('mnist\_49\_3000.mat')

x = mnist\_49\_3000['x']

y = mnist\_49\_3000['y']

d,n = x.shape

i = 2000 #Index of the image to be visualized

plt.imshow( np.reshape(x[:,i], (int(np.sqrt(d)),int(np.sqrt(d)))))

plt.show()

A = np.ones(n)

A = A[None,:]

xNew = np.vstack((A, x))

xTrain = xNew[:,:2000]

yTrain = y[:,:2000]

dTrain ,nTrain = xTrain.shape

#sigmoid as per our defination as 1/(1+exp(-(yi) \* theta.transpose() \* (xi))

def sigmoid(y, x, theta):

aMat = (np.matrix(theta)) \* np.matrix(x)

aArray = aMat.A1

b = aArray[None,:]

c = np.exp(- y \* b)

oneArray = np.ones(x.shape[1])[None,:]

sig = oneArray/(oneArray - c)

return sig

k = sigmoid(yTrain, xTrain, theta)

def gradient(y, x, theta, lamda):

var1 = (1-sigmoid(y, x, theta))

var2 = var1 \* (-y)

var3 = var2 \* x

term1 = var3.sum(axis = 1)

term2 = 2\*lamda \* np.ones(x.shape[0])

grad = term1 + term2

return grad

def hessian(y, x, theta, lamda):

hVar1 = np.matrix(xTrain) \* np.matrix(np.transpose(xTrain))

hVar2 = np.squeeze(np.asarray(hVar1))

return hess

lamda = 10

theta = np.ones(d)