import csv

import numpy as np

import matplotlib.pyplot as plt

def loadCSV(filename):

'''

function to load dataset

'''

with open(filename,"r") as csvfile:

lines = csv.reader(csvfile)

dataset = list(lines)

for i in range(len(dataset)):

dataset[i] = [float(x) for x in dataset[i]]

return np.array(dataset)

def normalize(X):

'''

function to normalize feature matrix, X

'''

mins = np.min(X, axis = 0)

maxs = np.max(X, axis = 0)

rng = maxs - mins

norm\_X = 1 - ((maxs - X)/rng)

return norm\_X

def logistic\_func(beta, X):

'''

logistic(sigmoid) function

'''

return 1.0/(1 + np.exp(-np.dot(X, beta.T)))

def log\_gradient(beta, X, y):

'''

logistic gradient function

'''

first\_calc = logistic\_func(beta, X) - y.reshape(X.shape[0], -1)

final\_calc = np.dot(first\_calc.T, X)

return final\_calc

def cost\_func(beta, X, y):

'''

cost function, J

'''

log\_func\_v = logistic\_func(beta, X)

y = np.squeeze(y)

step1 = y \* np.log(log\_func\_v)

step2 = (1 - y) \* np.log(1 - log\_func\_v)

final = -step1 - step2

return np.mean(final)

def grad\_desc(X, y, beta, lr=.01, converge\_change=.001):

'''

gradient descent function

'''

cost = cost\_func(beta, X, y)

change\_cost = 1

num\_iter = 1

while(change\_cost > converge\_change):

old\_cost = cost

beta = beta - (lr \* log\_gradient(beta, X, y))

cost = cost\_func(beta, X, y)

change\_cost = old\_cost - cost

num\_iter += 1

return beta, num\_iter

def pred\_values(beta, X):

'''

function to predict labels

'''

pred\_prob = logistic\_func(beta, X)

pred\_value = np.where(pred\_prob >= .5, 1, 0)

return np.squeeze(pred\_value)

def plot\_reg(X, y, beta):

'''

function to plot decision boundary

'''

# labelled observations

x\_0 = X[np.where(y == 0.0)]

x\_1 = X[np.where(y == 1.0)]

# plotting points with diff color for diff label

plt.scatter([x\_0[:, 1]], [x\_0[:, 2]], c='b', label='y = 0')

plt.scatter([x\_1[:, 1]], [x\_1[:, 2]], c='r', label='y = 1')

# plotting decision boundary

x1 = np.arange(0, 1, 0.1)

x2 = -(beta[0,0] + beta[0,1]\*x1)/beta[0,2]

plt.plot(x1, x2, c='k', label='reg line')

plt.xlabel('x1')

plt.ylabel('x2')

plt.legend()

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

# load the dataset

dataset = loadCSV('dataset1.csv')

# normalizing feature matrix

X = normalize(dataset[:, :-1])

# stacking columns wth all ones in feature matrix

X = np.hstack((np.matrix(np.ones(X.shape[0])).T, X))

# response vector

y = dataset[:, -1]

# initial beta values

beta = np.matrix(np.zeros(X.shape[1]))

# beta values after running gradient descent

beta, num\_iter = grad\_desc(X, y, beta)

# estimated beta values and number of iterations

print("Estimated regression coefficients:", beta)

print("No. of iterations:", num\_iter)

# predicted labels

y\_pred = pred\_values(beta, X)

# number of correctly predicted labels

print("Correctly predicted labels:", np.sum(y == y\_pred))

# plotting regression line

plot\_reg(X, y, beta)