import scipy.optimize as opt

import numpy

from numpy import loadtxt, where

from pylab import scatter, show, legend, xlabel, ylabel

#load the dataset

def plot(X,theta,y):

pos = where(y == 1)

neg = where(y == 0)

plot(y,theta,X)

scatter(X[pos, 0], X[pos, 1], marker='o', c='b')

scatter(X[neg, 0], X[neg, 1], marker='x', c='r')

xlabel('test data')

ylabel('training data')

legend(['Attacker', 'User'])

show()

# prefix an extra column of ones to the feature matrix (for intercept term)

def th(X,y):

theta = 0.1\* numpy.random.randn(3)

X\_1 = numpy.append( numpy.ones((X.shape[0], 1)), X, axis=1)

return theta

def sigmoid(X):

return 1 / (1 + numpy.exp(- X))

def cost(theta, X, y):

p\_1 = sigmoid(numpy.dot(X, theta)) # predicted probability of label 1

log\_l = (-y)\*numpy.log(p\_1) - (1-y)\*numpy.log(1-p\_1) # log-likelihood vector

return log\_l.mean()

def grad(theta, X, y):

p\_1 = sigmoid(numpy.dot(X, theta))

error = p\_1 - y # difference between label and prediction

grad = numpy.dot(error, X\_1) / y.size # gradient vector

return grad

def predict(theta, X, y):

p\_1 = sigmoid(numpy.dot(X, theta))

return p\_1 > 0.5