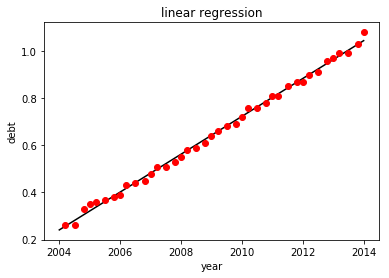
**HW2 Huaiyu Wang**

3.1



Output: [3.93601056]

w= 8.03244175e-02

b=-1.60729045e+02

In 2050, the student loan debt will be 3.93601056

Code here:

import csv

from pylab import \*

def main():

rfile = 'student\_debt.csv'

csvfile = open(rfile, 'rt')

data = csv.reader(csvfile, delimiter = ',')

X = []

Y = []

for i, row in enumerate(data):

X.append(float(row[0]))

Y.append(float(row[1]))

X = array(X)

X\_temp = X[:]

Y = array(Y)

one = ones((len(X)))

X = row\_stack((one,X))

X = X.T

Y = reshape(Y, (len(Y),1))

weight = linearfit(X,Y)

print (weight)

it = np.arange(2004, 2015, 1)

plt.ylabel('debt')

plt.xlabel('year')

plt.title('linear regression')

g = weight[1] \* it + weight[0]

plt.plot(it, g, 'k')

plt.plot(X\_temp, Y, 'ro')

plt.show()

plt.close()

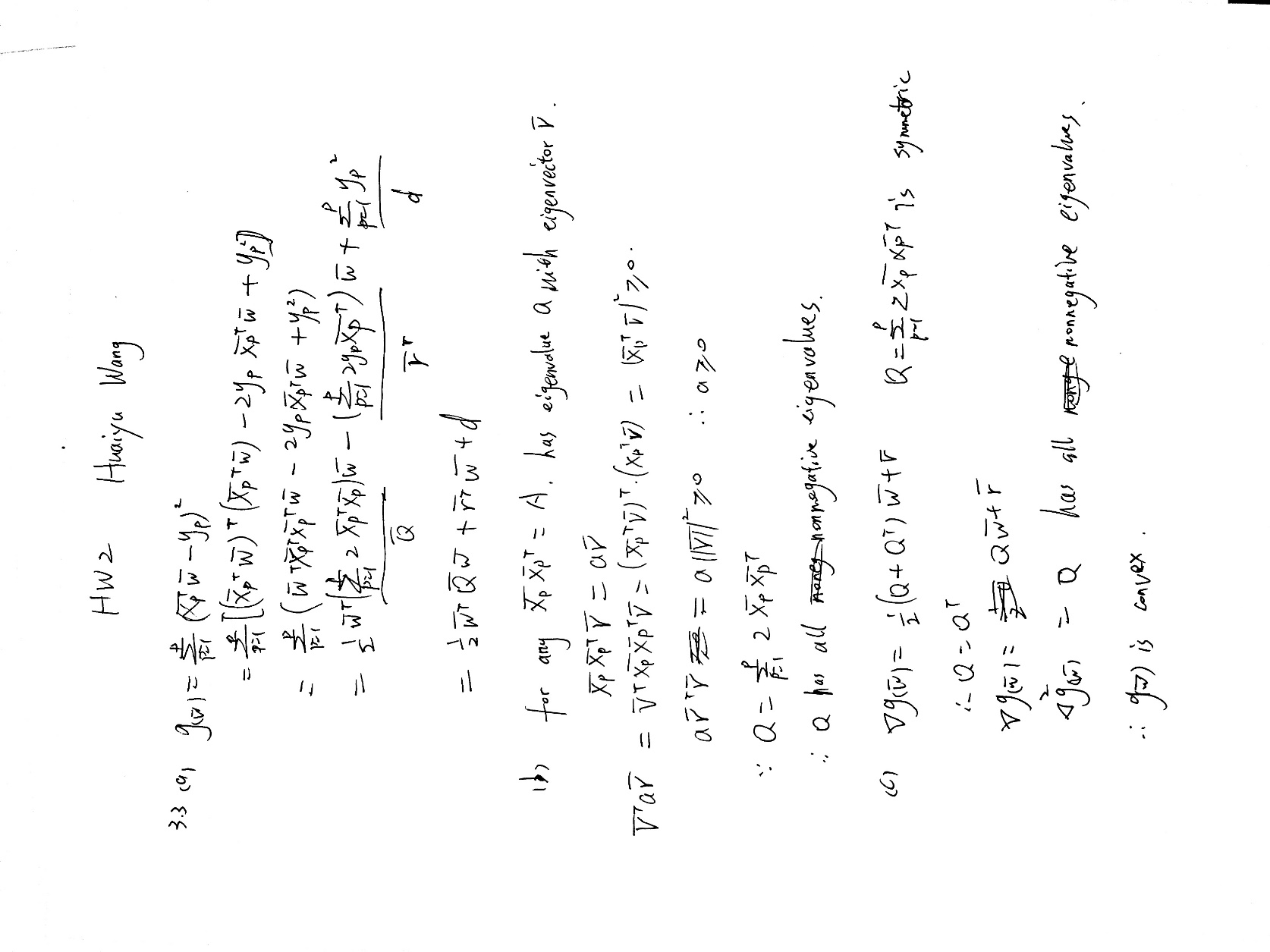
print (weight[1] \* 2050 + weight[0])

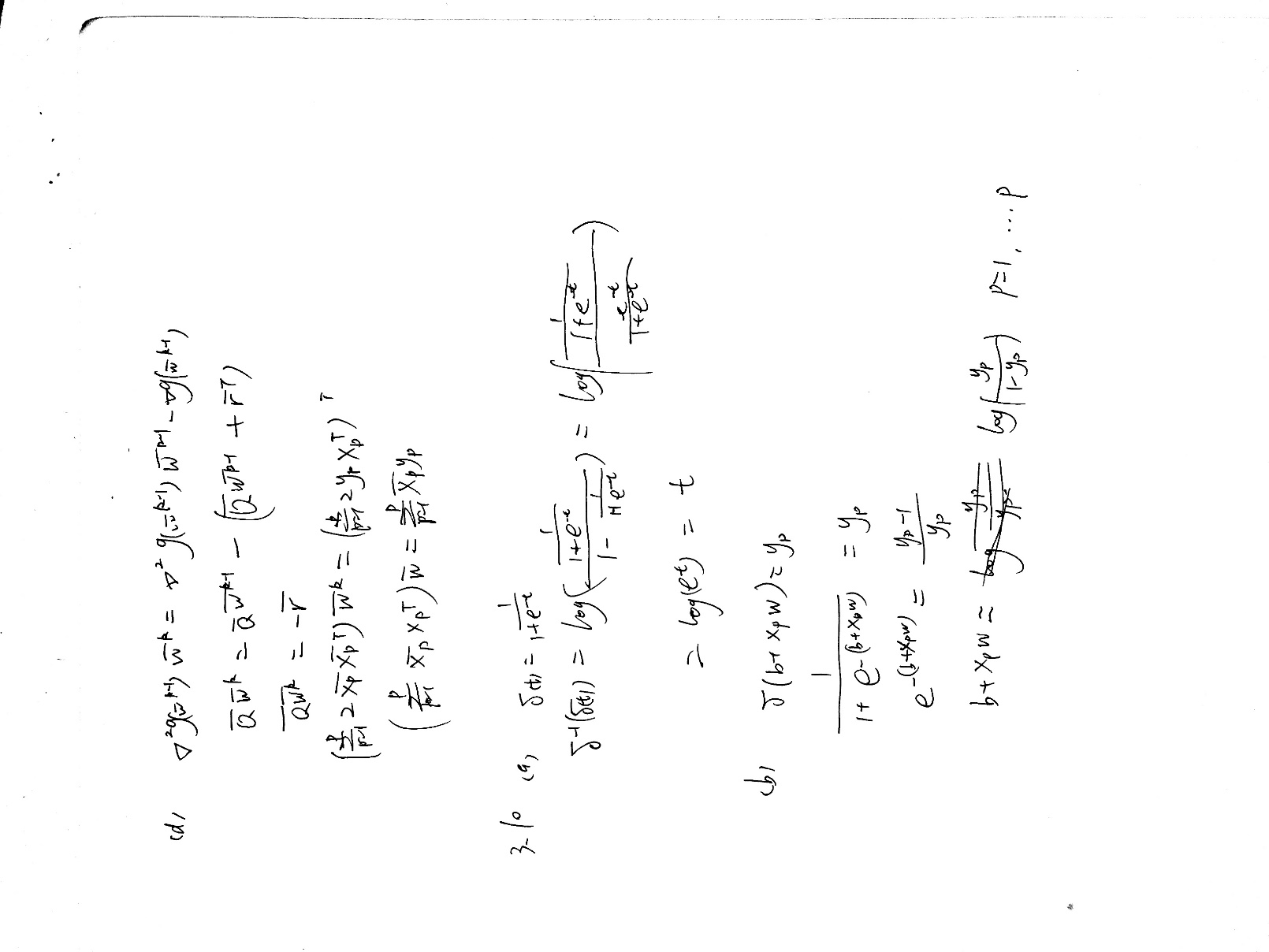
def linearfit(X, Y):

W =dot(inv(dot(X.T, X)),dot(X.T,Y))

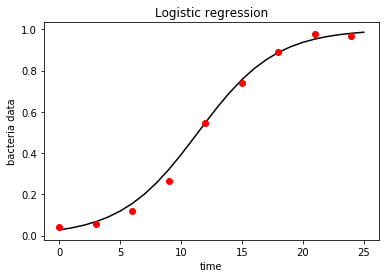
return W

main()

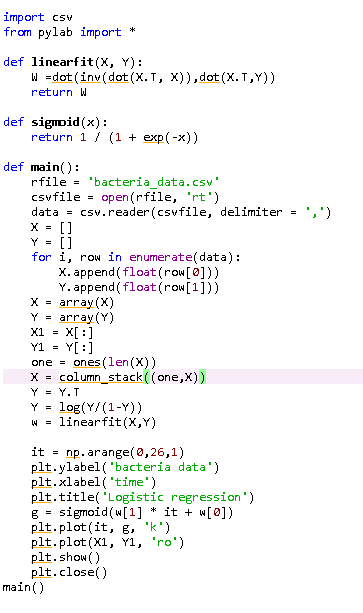


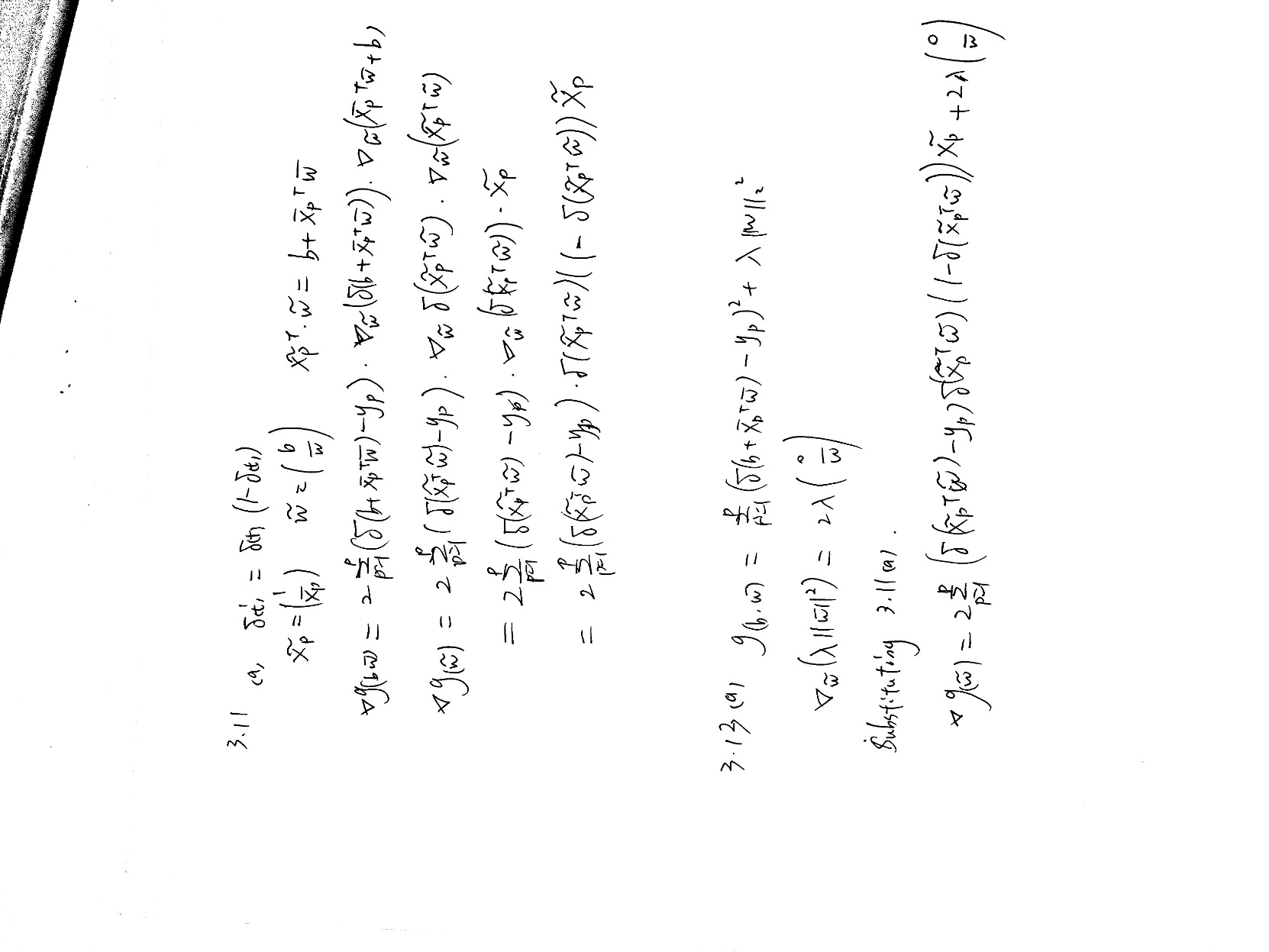


（c）

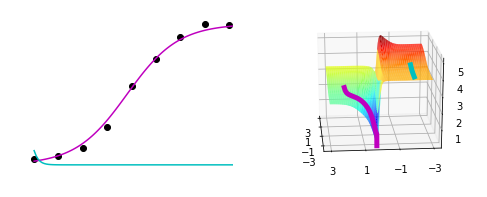


Code here：





（b）



# YOUR CODE GOES HERE - compute gradient

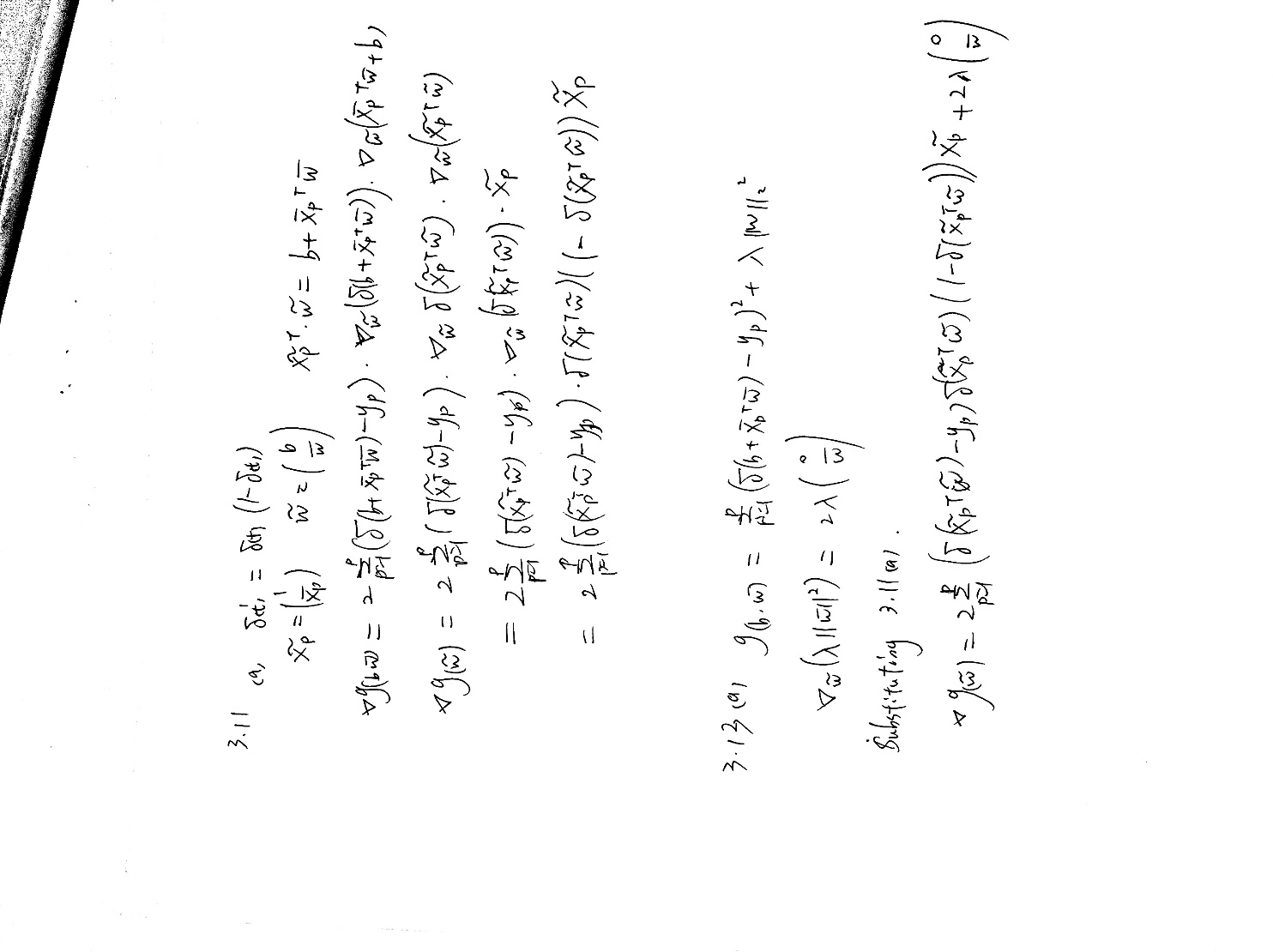
grad=0

for p in range(0,np.shape(X)[0]):

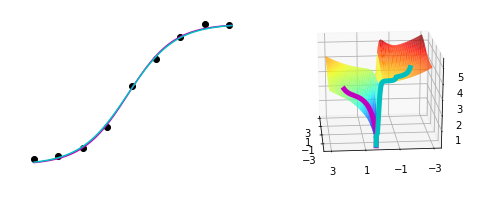
grad+=2\*(s(np.dot(X[p].T,w))-y[p])\*s(np.dot(X[p].T,w))\*(1-s(np.dot(X[p].T,w)))\*X[p]

grad=np.asarray(grad)

grad.shape=(2,1)



（b）



Code here：

def gradient\_descent(X,y,w0,lam):

w\_path = [] # container for weights learned at each iteration

cost\_path = [] # container for associated objective values at each iteration

w\_path.append(w0)

cost = compute\_cost(w0)

cost\_path.append(cost)

w= w0

w1= np.array([0,0])

w1.shape = (2,1)

# start gradient descent loop

max\_its =20000

alpha = 10\*\*(-2)

for k in range(max\_its):

# YOUR CODE GOES HERE - compute gradient

grad=0

for p in range(0,np.shape(X)[0]):

grad+=2\*(s(np.dot(X[p].T,w))-y[p])\*s(np.dot(X[p].T,w))\*(1-s(np.dot(X[p].T,w)))\*X[p]

grad=np.asarray(grad)

grad.shape=(2,1)

w1[1]=w[1]

grad=grad+0.05\*w1

# take gradient step

w = w - alpha\*grad

# update path containers

w\_path.append(w)

cost = compute\_cost(w)

cost\_path.append(cost)

# reshape containers for use in plotting in 3d

w\_path = np.asarray(w\_path)

w\_path.shape = (np.shape(w\_path)[0],2)

cost\_path = np.asarray(cost\_path)

cost\_path.shape = (np.size(cost\_path),1)

return w\_path,cost\_path