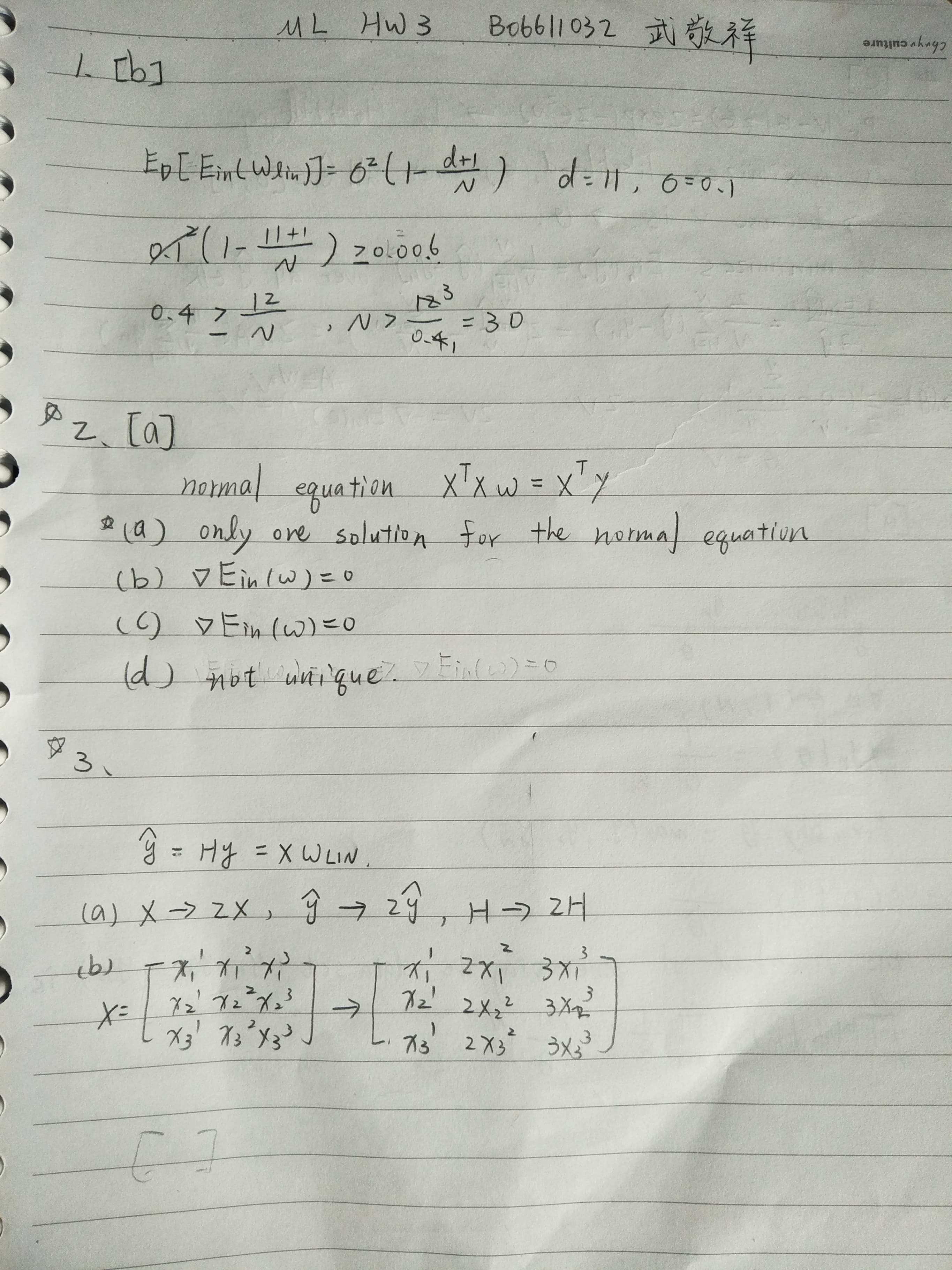
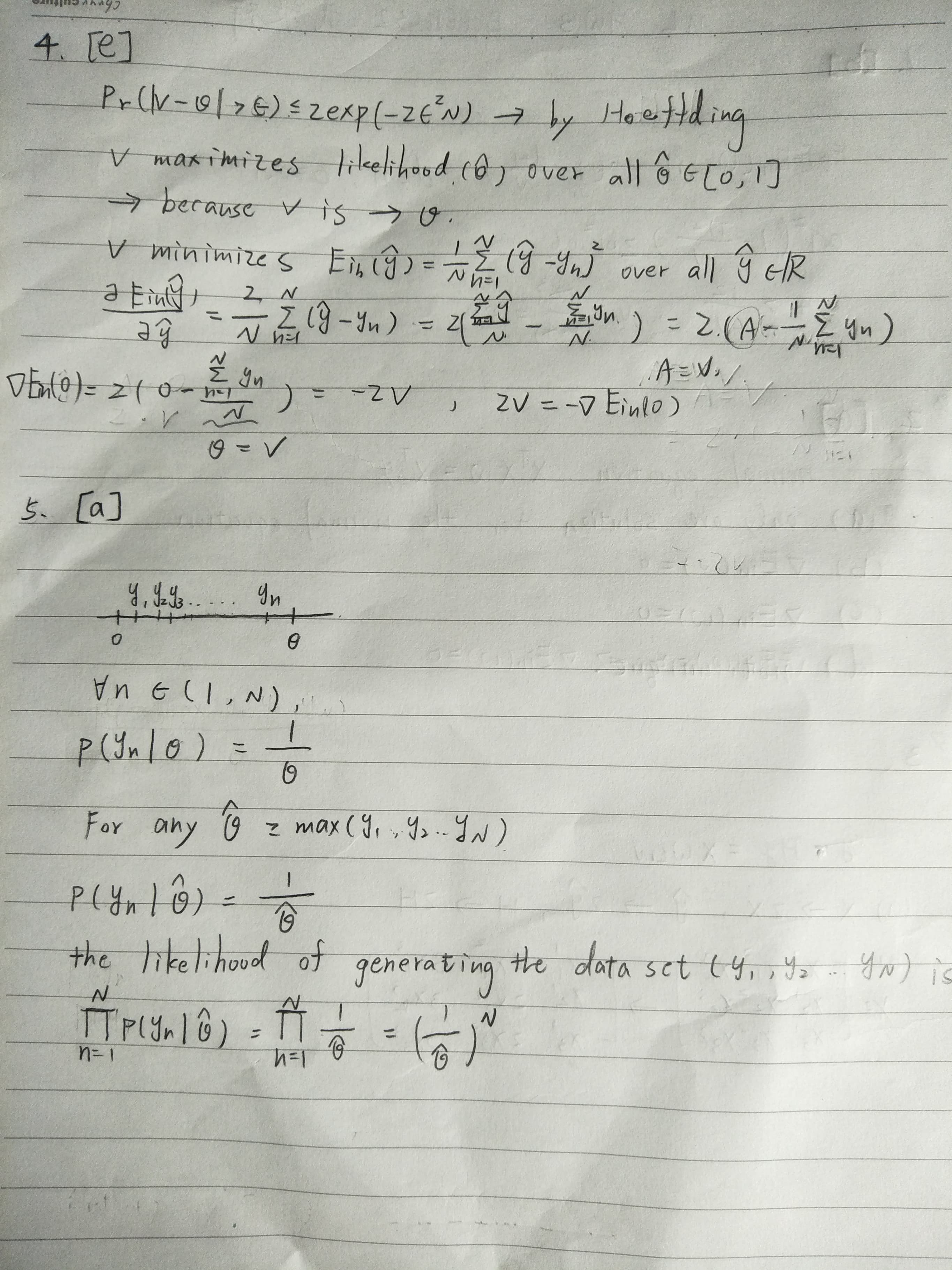
ML HW3

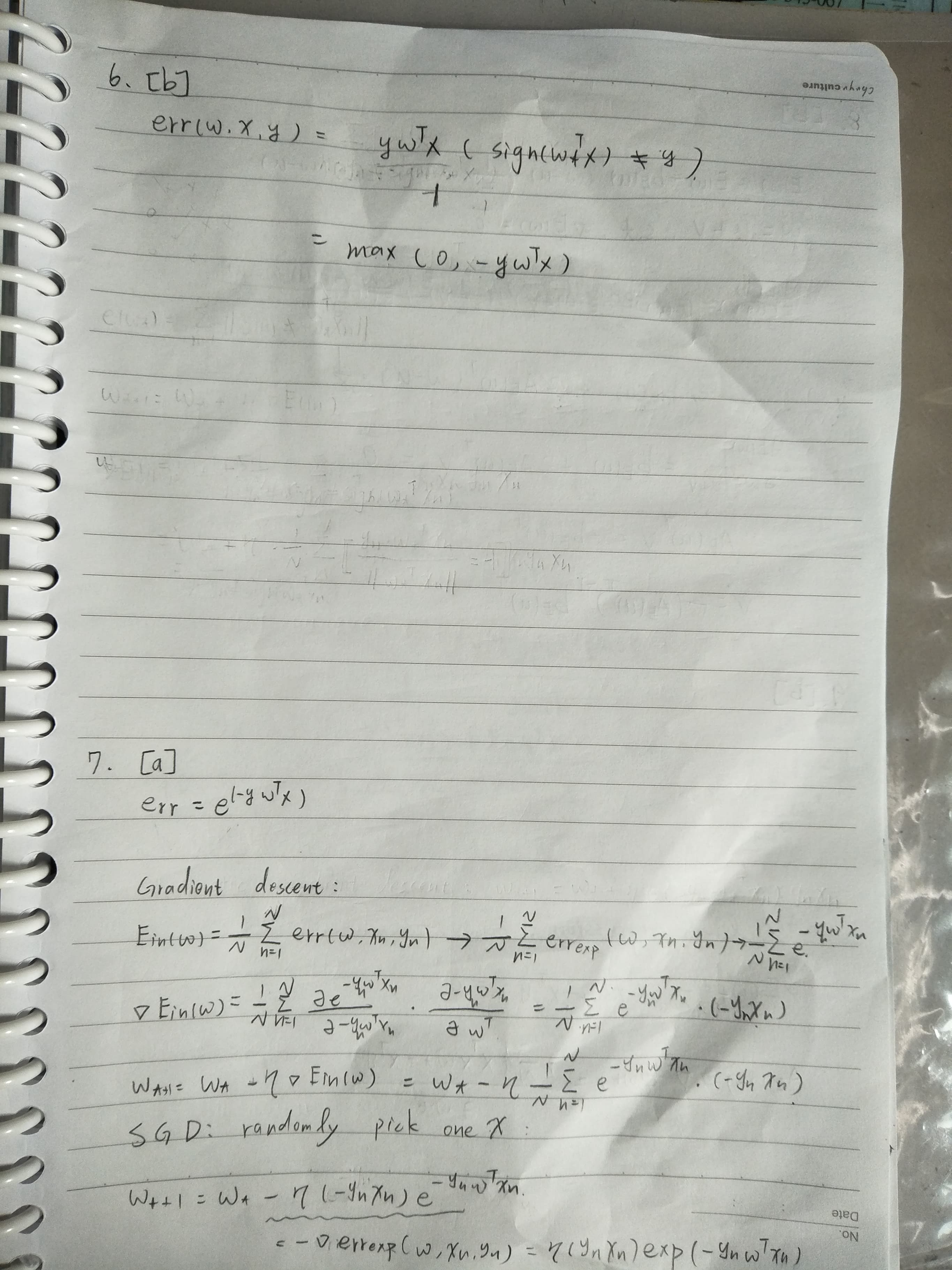
1,2,3:



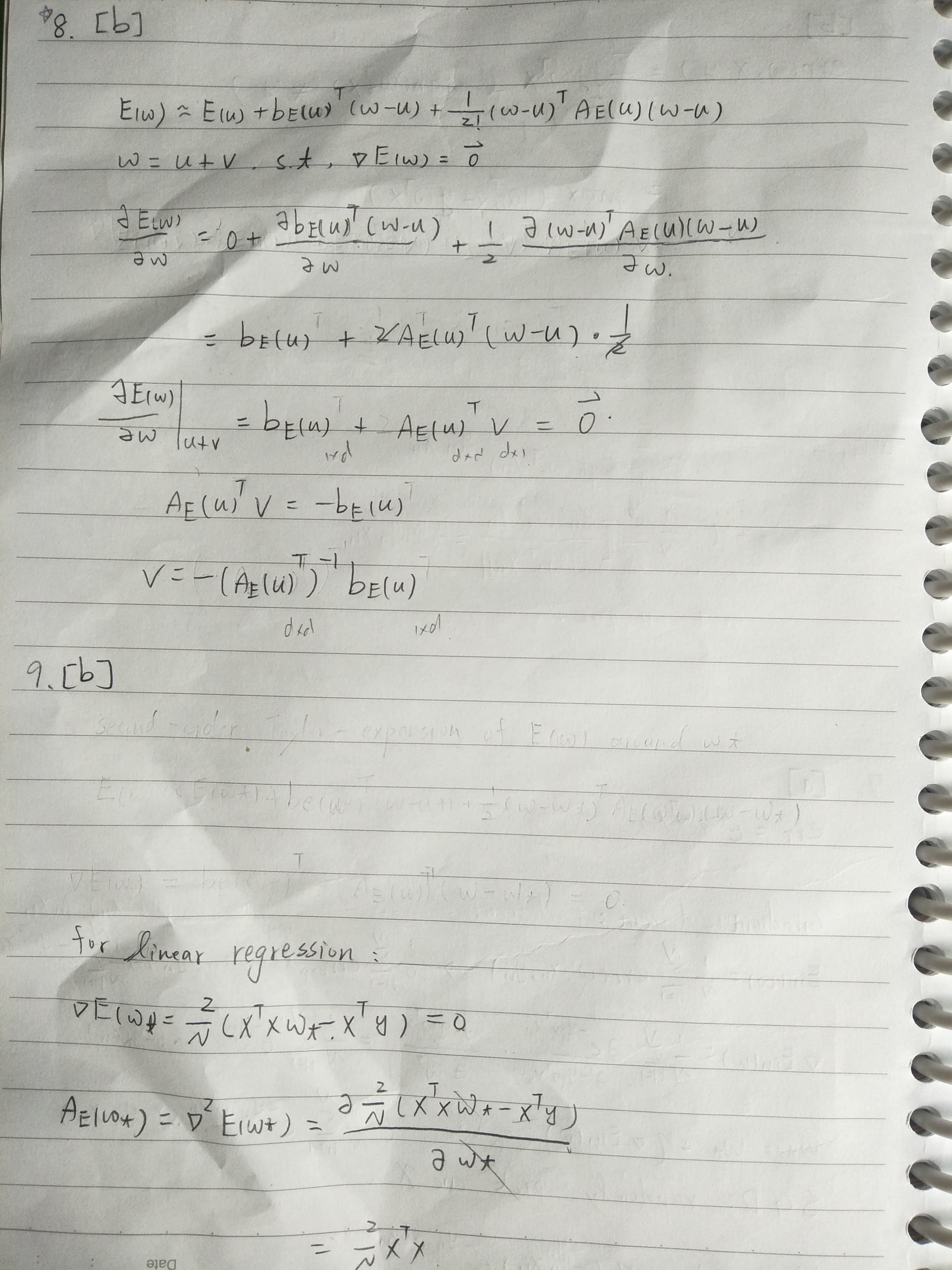
4,5:



6,7:



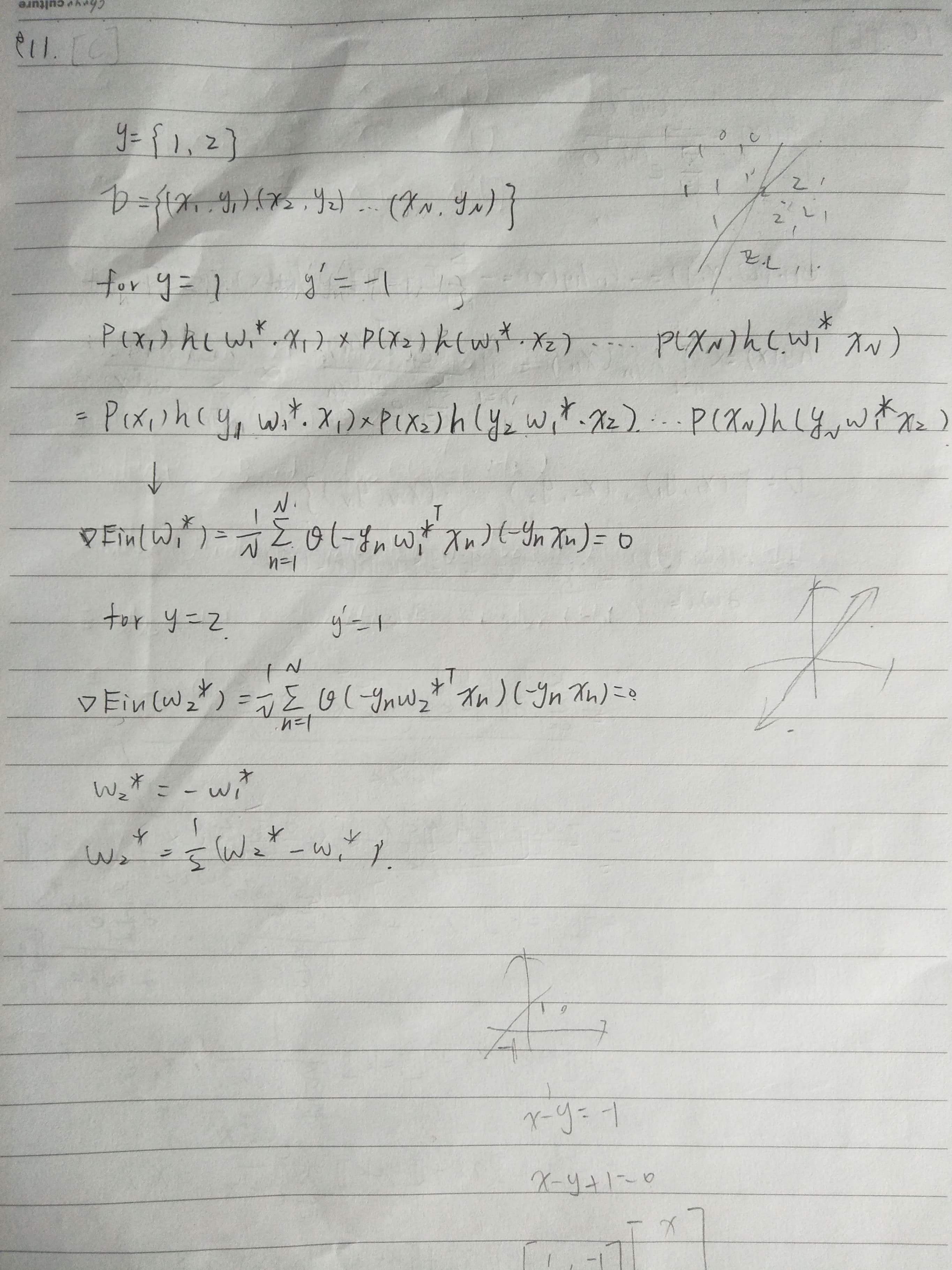
8,9:



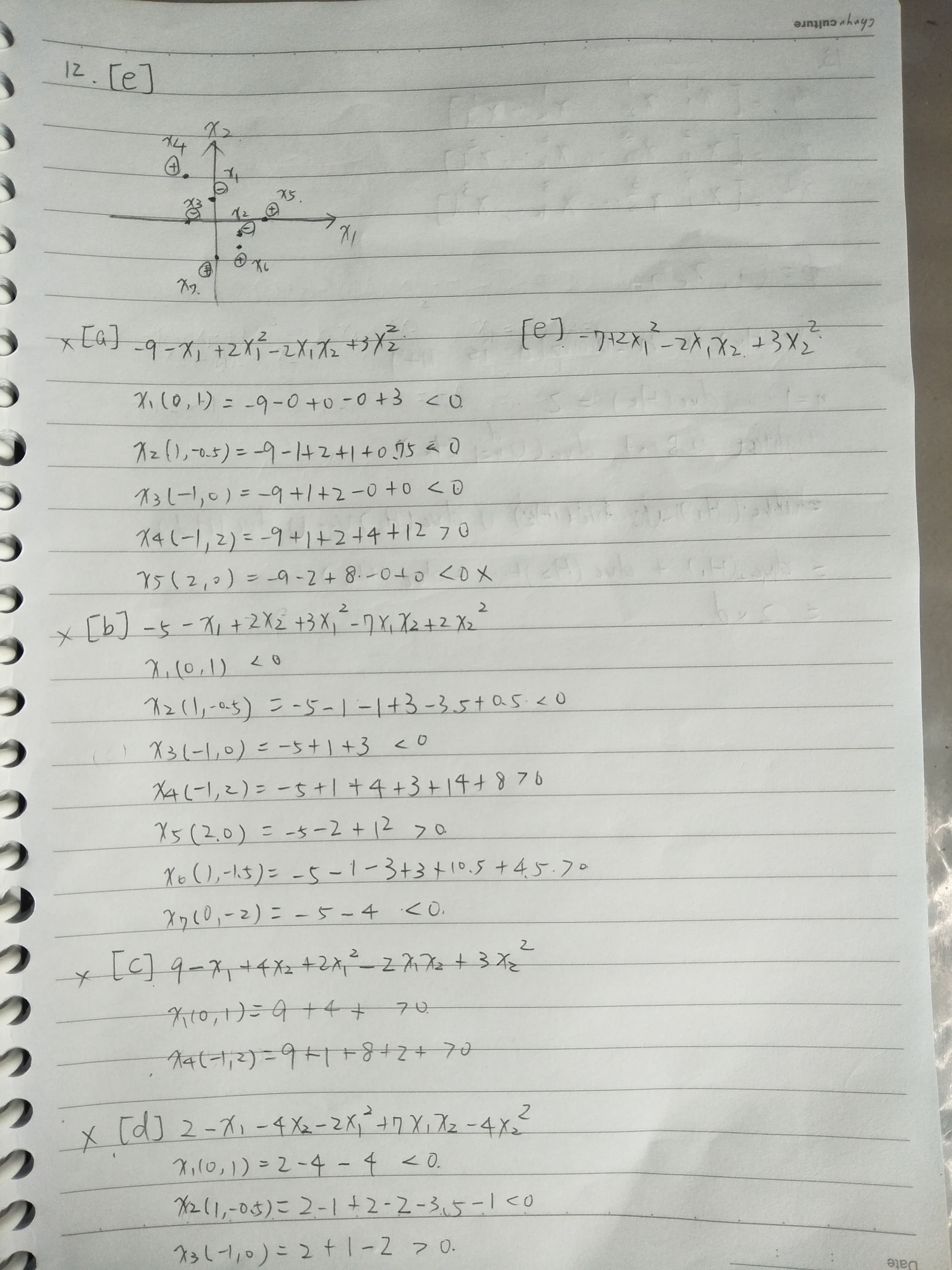
10:



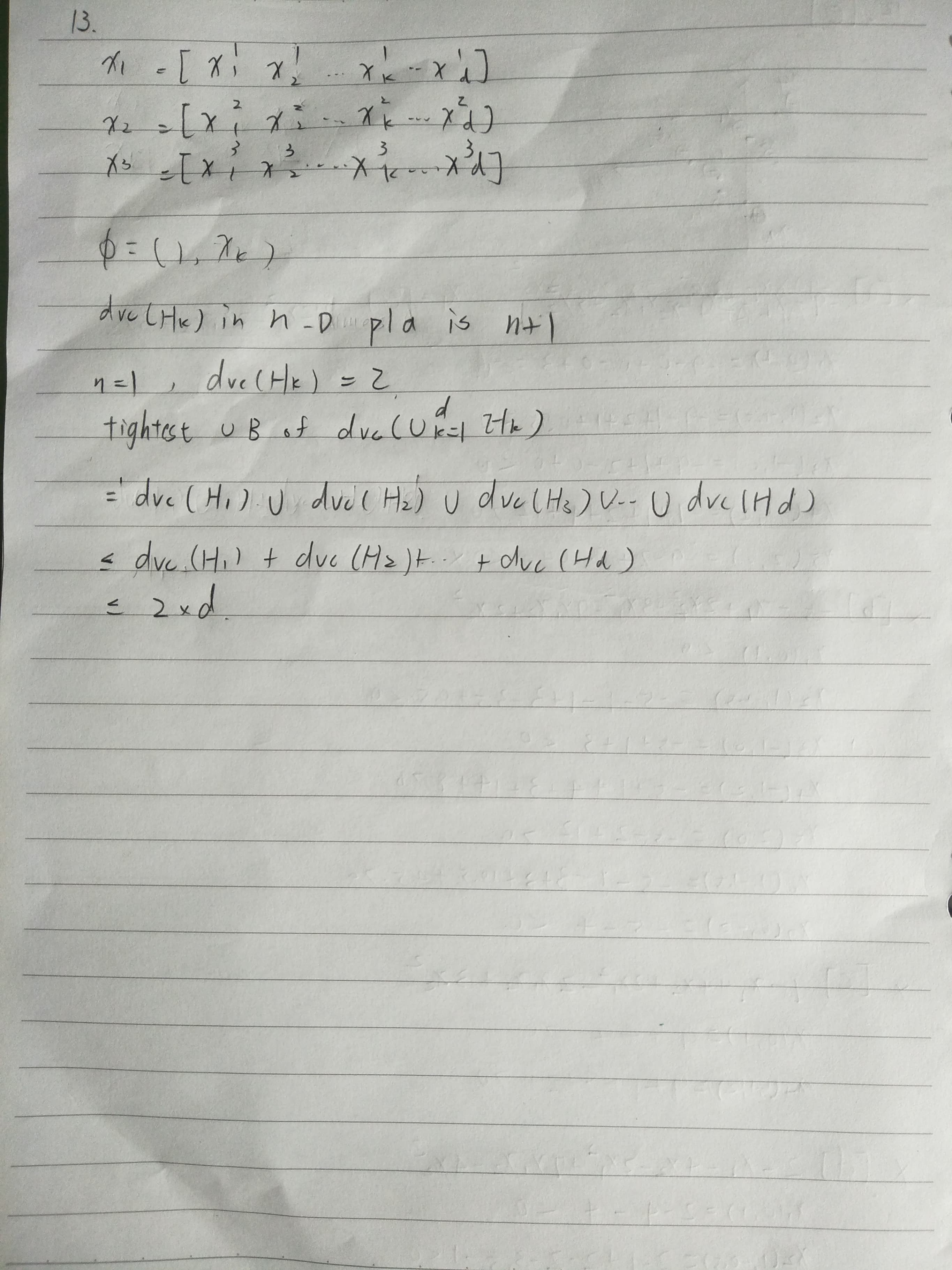
11:



12:



13:



14:

import numpy as np

X0 = np.ones([1000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

train = np.append(X0 , train ,axis=1)

X = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y = np.hsplit(train,[11])[1]

Wlin = np.dot(np.linalg.pinv(X) , Y)

E = np.dot(X , Wlin) - Y

print(E)

Ein = 0

for i in range(len(E)):

    Ein += (E[i][0]) \*\* 2/len(E)

print(Ein)

15:

import numpy as np

from random import randint

X0 = np.ones([1000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

train = np.append(X0 , train ,axis=1)

X = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y = np.hsplit(train,[11])[1]

Wlin = np.dot(np.linalg.pinv(X) , Y)

E = np.dot(X , Wlin) - Y

Einlin = 0

for i in range(len(E)):

    Einlin += (E[i][0]) \*\* 2/len(E)

n = 0.001

count = 0

times\_avg = 0

while count < 1000:

    print(count)

    times = 0

    ratio = 100

    Wt = np.zeros([11,1])

    while ratio > 1.01:

        rand = randint(0,999)

        E = np.dot(X , Wt) - Y

        Einwt = 0

        for i in range(len(E)):

            Einwt += (E[i][0]) \*\* 2/len(E)

        x = X[rand].reshape((1,11))    # what the hell is it????

        Ein\_grad = 2\*(np.dot(x,Wt) - Y[rand][0]) \* np.transpose(x)

        Wt = Wt- n\*Ein\_grad

        ratio = Einwt / Einlin

        times += 1

    times\_avg += times/1000

    count += 1

print(times\_avg)

16:

import numpy as np

from random import randint

X0 = np.ones([1000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

train = np.append(X0 , train ,axis=1)

X = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y = np.hsplit(train,[11])[1]

def sigmoid(h):

    ans = 1/(1+np.exp(-h))

    return ans

n = 0.001

count = 0

Ein\_avg = 0

while count < 1000:

    print(count)

    times = 0

    Wt = np.zeros([11,1])

    while times < 500:

        rand = randint(0,999)

        x = X[rand].reshape((1,11))    # what the hell is it????

        Ein\_grad = sigmoid(-Y[rand][0] \* np.dot(x,Wt)) \* Y[rand][0] \* np.transpose(x)

        Wt = Wt + n\*Ein\_grad

        times += 1

    Einwt = 0

    for i in range(1000):

        x = X[i].reshape((1,11))

        CE\_error = np.log(1+np.exp(-Y[i][0] \* np.dot(x,Wt)[0][0]))

        Einwt += CE\_error/1000

    Ein\_avg += Einwt/1000

    count += 1

print(Ein\_avg)

17:

import numpy as np

from random import randint

X0 = np.ones([1000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

train = np.append(X0 , train ,axis=1)

X = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y = np.hsplit(train,[11])[1]

Wlin = np.dot(np.linalg.pinv(X) , Y)

def sigmoid(h):

    ans = 1/(1+np.exp(-h))

    return ans

n = 0.001

count = 0

Ein\_avg = 0

while count < 1000:

    print(count)

    times = 0

    Wt = Wlin

    while times < 500:

        rand = randint(0,999)

        x = X[rand].reshape((1,11))    # what the hell is it????

        Ein\_grad = sigmoid(-Y[rand][0] \* np.dot(x,Wt)) \* Y[rand][0] \* np.transpose(x)

        Wt = Wt + n\*Ein\_grad

        times += 1

    Einwt = 0

    for i in range(1000):

        x = X[i].reshape((1,11))

        CE\_error = np.log(1+np.exp(-Y[i][0] \* np.dot(x,Wt)[0][0]))

        Einwt += CE\_error/1000

    Ein\_avg += Einwt/1000

    count += 1

print(Ein\_avg)

18:

import numpy as np

X0\_train = np.ones([1000,1])

X0\_test =np.ones([3000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

test = np.genfromtxt('test.txt', delimiter='\t')

train = np.append(X0\_train , train ,axis=1)

test = np.append(X0\_test , test ,axis=1)

X\_train = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y\_train = np.hsplit(train,[11])[1]

X\_test = np.hsplit(test,[11])[0]  ## saparate the array into two part on the base of the nth column

Y\_test = np.hsplit(test,[11])[1]

Wlin = np.dot(np.linalg.pinv(X\_train) , Y\_train)

def sign(s):

    if s >= 0:

        return 1

    else:

        return -1

E\_train = np.dot(X\_train , Wlin) \* Y\_train  #

E\_test = np.dot(X\_test , Wlin) \* Y\_test

Ein = 0

Eout = 0

for i in range(1000):

    if sign(E\_train[i]) == -1:

        Ein += 1/1000

    if sign(E\_test[i]) == -1:

        Eout += 1/1000

print(Ein)

print(Eout)

print(abs(Ein-Eout))

19:

import numpy as np

X0\_train = np.ones([1000,1])

X0\_test =np.ones([3000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

test = np.genfromtxt('test.txt', delimiter='\t')

train = np.append(X0\_train , train ,axis=1)

test = np.append(X0\_test , test ,axis=1)

X\_train = np.hsplit(train,[11])[0]  ## saparate the array into two part on the base of the nth column

Y\_train = np.hsplit(train,[11])[1]

X\_test = np.hsplit(test,[11])[0]  ## saparate the array into two part on the base of the nth column

Y\_test = np.hsplit(test,[11])[1]

Wlin = np.dot(np.linalg.pinv(X\_train) , Y\_train)

def sign(s):

    if s >= 0:

        return 1

    else:

        return -1

E\_train = np.dot(X\_train , Wlin) \* Y\_train  #

E\_test = np.dot(X\_test , Wlin) \* Y\_test

Ein = 0

Eout = 0

for i in range(1000):

    if sign(E\_train[i]) == -1:

        Ein += 1/1000

    if sign(E\_test[i]) == -1:

        Eout += 1/1000

print(Ein)

print(Eout)

print(abs(Ein-Eout))

20:

import numpy as np

X0\_train = np.ones([1000,1])

X0\_test =np.ones([3000,1])

train = np.genfromtxt('train.txt', delimiter='\t')

X\_train = np.hsplit(train,[10])[0]  ## saparate the array into two part on the base of the nth column

Y\_train = np.hsplit(train,[10])[1]

test = np.genfromtxt('test.txt', delimiter='\t')

X\_test = np.hsplit(test,[10])[0]  ## saparate the array into two part on the base of the nth column

Y\_test = np.hsplit(test,[10])[1]

Q = 10

Z\_train = X0\_train

Z\_test = X0\_test

for q in range(1,Q+1):

    Z\_train2 = np.power(X\_train , q)

    Z\_test2 = np.power(X\_test , q)

    Z\_train = np.append(Z\_train , Z\_train2 ,axis = 1)

    Z\_test = np.append(Z\_test , Z\_test2 ,axis = 1)

g = np.dot(np.linalg.pinv(Z\_train) , Y\_train)

def sign(s):

    if s >= 0:

        return 1

    else:

        return -1

E\_train = np.dot(Z\_train , g) \* Y\_train  #

E\_test = np.dot(Z\_test , g) \* Y\_test

Ein = 0

Eout = 0

for i in range(1000):

    if sign(E\_train[i]) == -1:

        Ein += 1/1000

for i in range(3000):

    if sign(E\_test[i]) == -1:

        Eout += 1/3000

print(Ein)

print(Eout)

print(abs(Ein-Eout))