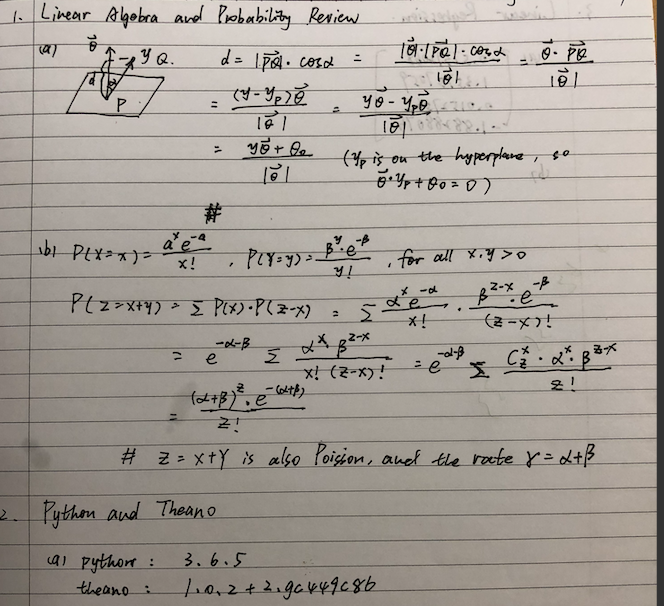
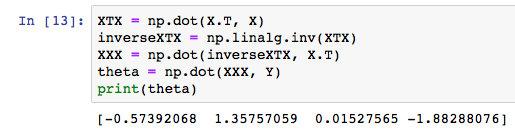
**Li Xingxuan 1002189**



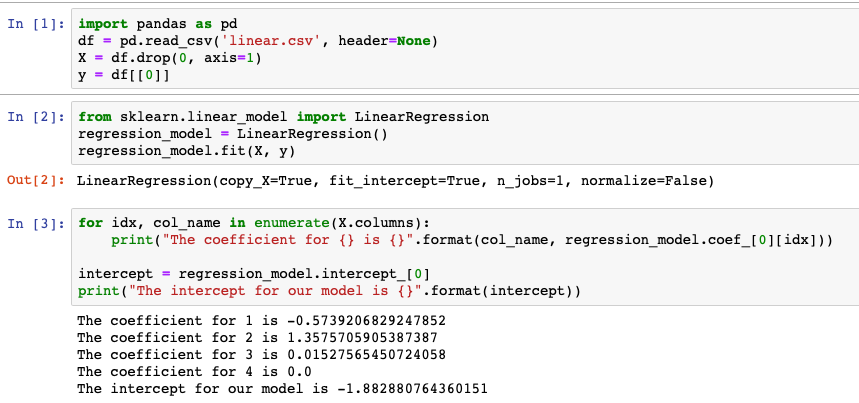
**(a) Original Code Result:**



**(b) Eexact solution:**



**(c) Sklearn:**



**(d) SGD**

%**matplotlib** inline

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **random**

data = np.genfromtxt('linear.csv',delimiter=',')

X = data[:,1:]

Y = data[:,0]

*# print(type(X))*

In [42]:

**def** mini\_batch(X, y):

loc = random.randint(1, 44)

result\_X = X[loc:loc + 5, :]

result\_Y = Y[loc:loc + 5]

**return** result\_X, result\_Y

test\_X, test\_y = mini\_batch(X, Y)

*# print(test\_X, test\_y)*

In [20]:

**import** **theano**

**import** **theano.tensor** **as** **T**

x = T.matrix(name='x') *# feature matrix*

y = T.vector(name='y') *# responese vector*

w = theano.shared(np.zeros((4,1)),name='w') *# model parameteres*

risk = T.sum((T.dot(x,w).T - y)\*\*2)/10 *# empirical risk*

grad\_risk = T.grad(risk, wrt=w) *# gradient of the risk*

In [41]:

n\_steps = 1000

lost\_list = []

w\_dict = {}

**for** i **in** range(n\_steps):

for\_x, for\_y = mini\_batch(X, Y)

train\_model = theano.function(inputs=[],

outputs=risk,

updates=[(w, w-(1/(i+1))\*grad\_risk)],

givens={x:for\_x, y:for\_y})

*# print(train\_model())*

*# print(w.get\_value())*

lost\_list.append(train\_model().item(0))

w\_dict[lost\_list[-1]] = w.get\_value()

print(w.get\_value())

print(lost\_list[-1])

lost\_list.sort()

print(w\_dict[lost\_list[0]])

print(lost\_list[0])

[[-0.57992765]

[ 1.35436854]

[ 0.01884709]

[-1.88086824]]

0.006610305112215585

[[-0.5673264 ]

[ 1.34984663]

[ 0.017143 ]

[-1.87757628]]

0.0007559927287628303