**Mini Batch Gradient Descent**

**Code:**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.metrics import r2\_score,accuracy\_score

import random

from sklearn.linear\_model import LinearRegression

x,y=load\_iris(return\_X\_y=True)

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=2)

lr=LinearRegression()

lr.fit(x\_train,y\_train)

y\_pred=lr.predict(x\_test)

r2\_score(y\_test,y\_pred)

lr.coef\_

lr.intercept\_

class MBRegeressor:

def \_\_init\_\_(self,batch\_size,LR,epoch):

self.batch\_size=batch\_size

self.LR=LR

self.epoch=epoch

self.coff\_=None

self.intercept\_=None

def fit(self,x,y):

self.intercept\_=0

self.coff\_=np.ones(np.array(x.shape[1]))

print(self.coff\_)

for i in range(self.epoch):

for j in range(int(x.shape[0]/self.batch\_size)):

index=random.sample(range(x.shape[0]),self.batch\_size)

y\_pred=np.dot(x[index],self.coff\_)+self.intercept\_

# print('prediction',y\_pred)

inter\_der=-2\*np.mean(y[index]-y\_pred)

self.intercept\_=self.intercept\_-(self.LR\*inter\_der)

coff\_der=-2\*np.dot((y[index]-y\_pred),x[index])/(x.shape[0]/self.batch\_size)

self.coff\_=self.coff\_-(self.LR\*coff\_der)

print(self.coff\_)

print(self.intercept\_)

def predict(self,x\_test):

return np.dot(x\_test,self.coff\_)+self.intercept\_

ApnaMreg=MBRegeressor(batch\_size=int(x.shape[0]/10),LR=0.009,epoch=2)

ApnaMreg.fit(x\_train,y\_train)

# y\_pred1=ApnaMreg.predict(x\_test)

# r2\_score(y\_test,y\_pred1)

**Visualization:**







