Pytorch实现线性回归，利用随机梯度下降法更新参数w和b来最小化损失函数，最终学得w和b的数值

#numpy和pytorch实现线性回归

import torch as t

from matplotlib import pyplot as plt

from IPython import display

device = t.device('cpu')

#设置随机数种子

t.manual\_seed(1000)

def get\_fake\_data(batch\_size=8):

#产生随机数据：y=2x+3，加上一些噪声

x = t.rand(batch\_size,1,device=device)\*5 #x为0到5的随机样本

y = x\*2+3+t.randn(batch\_size,1,device=device)

return x,y

x,y = get\_fake\_data(batch\_size=16)

plt.scatter(x.squeeze().cpu().numpy(), y.squeeze().cpu().numpy())

# 随机初始化参数

w = t.rand(1, 1).to(device)

b = t.zeros(1, 1).to(device)

lr =0.02 # 学习率

for ii in range(500):

x, y = get\_fake\_data(batch\_size=4)

# forward：计算loss

y\_pred = x.mm(w) + b.expand\_as(y) # mm为矩阵相乘，expand\_as扩展矩阵范围，将b（1,1）扩展为（4,1）

loss = 0.5 \* (y\_pred - y) \*\* 2 # 均方误差

loss = loss.mean()

# backward：手动计算梯度

dloss = 1

dy\_pred = dloss \* (y\_pred - y)

dw = x.t().mm(dy\_pred)

db = dy\_pred.sum()

# 更新参数

w.sub\_(lr \* dw)

b.sub\_(lr \* db)

if ii%50 ==0:

# 画图

display.clear\_output(wait=True)

x = t.arange(0, 6).view(-1, 1).float()# t.arange 输出结果为 LongTensor 所以需要转化为float 类型不然会报错

y = x.mm(w) + b.expand\_as(x)

plt.plot(x.cpu().numpy(), y.cpu().numpy()) # predicted

x2, y2 = get\_fake\_data(batch\_size=32)

plt.scatter(x2.numpy(), y2.numpy()) # true data

plt.xlim(0, 5)

plt.ylim(0, 13)

plt.show()

plt.pause(0.5)

print('w: ', w.item(), 'b: ', b.item())