用pytorch实现线性回归

笔记本:	pytorch		
创建时间:	2023/3/25 11:24	**更新时间:**	2023/3/26 20:34
作者:	22qdnlbn		
URL:	about:blank		

1、pytorch fashion:

(1) parpare dataset
(2) construct model using class（inherit nn.moduel）
(3) construct loss and optimizer
(4) traing cycle

2、代码实例

```
# 时间：2023/3/25 21:52
# cky
import torch
import matplotlib.pyplot as plt
from torch import nn as nn
#prepare dataset
x_data=torch.Tensor([[1.0],[2.0],[3.0]])
y_data=torch.Tensor([[2.0],[4.0],[6.0]])


#design modeul using class
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel,self).__init__() #just do it
        self.linear=nn.Linear(1,1)  #in_put_size(int),out_put_size(int),bais(bool)
        #y^=wx+b
        #nn.Linear 也是继承自modeul，其对象被调用时，如forward中的第一句，就会执行
nn.linear的forward,
        #一个计算图将生成,
    def forward(self,x):
        y_hat=self.linear(x)
        return y_hat


#继承自modeul 都有__call__ 即当继承自nn.Modeul的对象被调用时会自动执行forward函数
model=LinearModel() #实例化一个对象


#construct loss and optimizer
l=nn.MSELoss(reduction='mean')  #size_average=None, reduce=None
reduction.py中提示的，我们不再使用 size和reduce 直接使用reduction即可
if size_average is None:
    size_average = True
if reduce is None:
    reduce = True


if size_average and reduce:
    ret = 'mean'
elif reduce:
    ret = 'sum'
else:
    ret = 'none'
```

```
#size_average 即我们求得的损失是否需要除以样本个数 是否求平均损失
#reduce 即我们求得的loss 其实是一个张量向量，如果我们需要将其转为一个值，即求sum，就需要为true
#也是继承自nn.modeul
optim=torch.optim.SGD(model.parameters(),lr=0.01)
#params: _params_t, lr: float, momentum: float=..., dampening: float=...,
weight_decay:float=..., nesterov:bool=...) -> None


#traing cycle
epoch_list=[]
loss_list=[]
for epoch in range(101):
    ys_hat=model(x_data) #对象被调用，会自动执行forward
    loss=l(ys_hat,y_data)
    epoch_list.append(epoch)
    loss_list.append(loss.item())
    print(epoch,loss.item())
    optim.zero_grad()  #如之前讲的，要将梯度清零
    loss.backward() #反向传播，计算梯度
    optim.step() #梯度更新
print('w:',model.linear.weight.item())
print('b:',model.linear.bias.item())
x_test=torch.Tensor([4.0])
y_test=model(x_test)
print("y_predict:",y_test.item())
plt.plot(epoch_list,loss_list)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

我们改变通过改变迭代次数和学习率 来改变我们的准确率