 第4讲  反向传播back propagation 源代码

B站 刘二大人 ，传送门PyTroch 深度学习实践——反向传播

如果需安装PyTorch，传送门 PyTorch深度学习快速入门教程

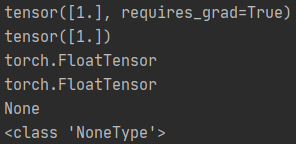
传送门 Tensor和tensor的区别

 代码说明：

1、w是Tensor(张量类型)，Tensor中包含data和grad，data和grad也是Tensorl。grad初始为None，调用l.backward()方法后w.grad为Tensor，故更新w.data时需使用w.grad.data。如果w需要计算梯度，那构建的计算图中，跟w相关的tensor都默认需要计算梯度。

|  |
| --- |
| import torch  a = torch.Tensor([1.0])  a.requires\_grad = True # 或者 a.requires\_grad\_()  print(a)  print(a.data)  print(a.type()) # a的类型是tensor  print(a.data.type()) # a.data的类型是tensor  print(a.grad)  print(type(a.grad)) |

结果为：



2、w是Tensor， forward函数的返回值也是Tensor，loss函数的返回值也是Tensor

3、本算法中反向传播主要体现在，l.backward()。调用该方法后w.grad由None更新为Tensor类型，且w.grad.data的值用于后续w.data的更新。

|  |
| --- |
| import torch  x\_data = [1.0, 2.0, 3.0]  y\_data = [2.0, 4.0, 6.0]    w = torch.Tensor([1.0]) # w的初值为1.0  w.requires\_grad = True # 需要计算梯度    def forward(x):  return x\*w # w是一个Tensor      def loss(x, y):  y\_pred = forward(x)  return (y\_pred - y)\*\*2    print("predict (before training)", 4, forward(4).item())    for epoch in range(100):  for x, y in zip(x\_data, y\_data):  l =loss(x,y) # l是一个张量，tensor主要是在建立计算图 forward, compute the loss  l.backward() # backward,compute grad for Tensor whose requires\_grad set to True  print('\tgrad:', x, y, w.grad.item())  w.data = w.data - 0.01 \* w.grad.data # 权重更新时，需要用到标量，注意grad也是一个tensor    w.grad.data.zero\_() # after update, remember set the grad to zero    print('progress:', epoch, l.item()) # 取出loss使用l.item，不要直接使用l（l是tensor会构建计算图）    print("predict (after training)", 4, forward(4).item()) |

————————————————

版权声明：本文为CSDN博主「错错莫」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/bit452/article/details/109643481>