深度学习快速入门

06nn.Module 模块使用

POET

2024 年 2 月 11 日

1 nn.Module 使用

- nn.Module 是对所有神经网络提供一个基本的类。
- 我们的神经网络是继承 nn.Module 这个类，即 nn.Module 为父类，nn.Module 为所有神经网络提供一个模板，对其中一些我们不满意的部分进行修改。

```
import torch
from torch import nn

class Tudui(nn.Module):
    def __init__(self):
        super(Tudui, self).__init__()  # 继承父类的初始化

    def forward(self, input):          # 将forward函数进行重写
        output = input + 1
        return output

tudui = Tudui()
x = torch.tensor(1.0)  # 创建一个值为 1.0 的tensor
output = tudui(x)
print(output)
```

```
super(Myclass, self).__init__():
1.简单理解就是子类把父类的__init__()放到自己的__init__()当中，这样子类就有了父类的__init__()的那些东西。
2.Myclass类继承nn.Module，super(Myclass, self).__init__()就是对继承自父类nn.Module的属性进行初始化。而且是用nn.Module的初始化方法来初始化继承的属性。
```

3.super().__init()__()来通过初始化父类属性以初始化自身继承了父类的那部分属性；这样一来，作为nn.Module的子类就无需再初始化那一部分属性了，只需初始化新加的元素。

4.子类继承了父类的所有属性和方法，父类属性自然会用父类方法来进行初始化。

2 forword 函数

- 使用 pytorch 的时候，不需要手动调用 forward 函数，只要在实例化一个对象中传入对应的参数就可以自动调用 forward 函数
- 因为 PyTorch 中的大部分方法都继承自 Module，而 torch.nn.Module 的 call(self) 函数中会返回 forward() 函数的结果，因此 PyTroch 中的 forward() 函数等于是被嵌套在了 call(self) 函数中；因此 forward() 函数可以直接通过类名被调用，而不用实例化对象。

```
class A():
def __call__(self, param):
    print('i can called like a function')
    print('传入参数的类型是：{}    值为：  {}'.
       format(type(param), param))
    res = self.forward(param)
    return res

def forward(self, input_):
    print('forward 函数被调用了')
    print('in  forward, 传入参数类型是：{}  值为:
       {}'.format( type(input_), input_))
    return input_

```

```
13      a = A()
14      input_param = a('i')
15      print("对象a传入的参数是：", input_param)
```