pytorch常见初始化操作

1. torch.nn.init.xavier_uniform_()

语法

torch.nn.init.xavier_uniform_(tensor, gain=1.0)

作用

使用均匀分布 用值填充输入张量

原理

结果张量将具有从 U (− a , a)采样的值，其中

$$a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

也称为Glorot初始化

> Fills the input *Tensor* with values according to the method described in *Understanding the difficulty of training deep feedforward neural networks* - Glorot, X. & Bengio, Y. (2010), using a uniform distribution. The resulting tensor will have values sampled from $\mathcal{U}(-a, a)$ where
>
> $$a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$
>
> Also known as Glorot initialization.
>
> Parameters
>
> - **tensor** – an n-dimensional *torch.Tensor*
> - **gain** – an optional scaling factor
>
> Examples

```
>>> w = torch.empty(3, 5)
>>> nn.init.xavier_uniform_(w, gain=nn.init.calculate_gain('relu'))
```

举例

```
import torch
import torch.nn as nn
w = torch.Tensor([[1.0,2,3],[2.0,3,4]])
print('w : \n', w)
nn.init.xavier_uniform_(w, gain=nn.init.calculate_gain('relu'))
print('w : \n', w)
```

```
w :
 tensor([[1., 2., 3.],
        [2., 3., 4.]])
w :
 tensor([[-0.0365,  0.6977, -1.3484],
        [ 1.0967,  0.2904,  0.7701]])
```

此处要注意：w不能为一维

为什么需要xavier初始化?

简单的说就是，

- 如果初始化值很小，那么随着层数的传递，方差就会趋于0，此时输入值也变得越来越小，在sigmoid上就是在0附近，接近于线性，失去了非线性
- 如果初始值很大，那么随着层数的传递，方差会迅速增加，此时输入值变得很大，而sigmoid在大输入值写倒数趋近于0，反向传播时会遇到梯度消失的问题

所以论文提出，在每一层网络保证输入和输出的方差相同

参考

https://blog.csdn.net/weixin_44225182/article/details/126655294