Xavier 初始化【简单易懂】

superjie13

目录 收起

1. Xavier 初始化简介
2. 数学原理
3. 作用
4. 示例
5. 代码

1. Xavier 初始化简介

Xavier 初始化是一种在训练深度学习模型时常用的权重初始化方法。它是 Xavier Glorot 和 Yoshua Bengio 在 2010 年提出的，原文为 Understanding the difficulty of training deep feedforward neural networks。该初始化方法旨在保持激活函数的方差在前向传播和反向传播过程中大致相同，从而避免梯度消失或梯度爆炸的问题。

2. 数学原理

考虑一个简单的全连接层，该层接受 n_{in} 个输入并产生 n_{out} 个输出。每个输出 o_i 可以表示为：

$$o_i = \text{activation}(\sum_{j=1}^{n_{in}} w_{ij}x_j + b_i)$$

其中 w_{ij} 是输入 x_j 到输出 o_i 的权重， b_i 是输出 o_i 的偏置， activation 是激活函数。

如果输入 x 的方差为 $Var(x)$，则线性函数 $\sum_{j=1}^{n_{in}} w_{ij}x_j$ 的方差将是 $n_{in} \times Var(w) \times Var(x)$（忽略偏执和激活函数）。

Xavier 初始化试图使得每一层的输出的方差接近于其输入的方差。具体地，它设置权重 w 的初始方差为：

$$Var(w) = \frac{2}{n_{in} + n_{out}}$$

这样，无论 n_{in} 和 n_{out} 的大小如何，这一层的输出方差都接近于其输入方差。

3. 作用

1. **梯度消失和爆炸**：在深度网络中，梯度消失和梯度爆炸是一个常见的问题。如果每一层都将方差

失）。Xavier 初始化试图使得每一层的输出的方差接近于其输入的方差，从而避免梯度消失或梯度爆炸的问题。

2. **加速收敛**：Xavier 初始化使得每一层的输出的方差接近于其输入的方差，从而使得每一层的梯度的方差接近于 1。这样，每一层的参数更新的幅度就不会相差太大，从而加速收敛。

4. 示例

考虑一个有256个输入节点和512个输出节点的全连接层。我们使用 Xavier 初始化，权重 w 的方差将被设置为：

$$Var(w) = \frac{2}{256 + 512} \approx 0.0026$$

然后，权重 w 将会从一个均值为 0，方差为 0.0026 的正态分布中随机抽取，或者从 $[-v, v]$ 的均匀分布中随机抽取，其中 $v = \sqrt{0.0026 \times 3} \approx 0.09$ 。(因为均匀分布的方差为 $\frac{1}{12}(b-a)^2$ ，而正态分布的方差为 σ^2，因此我们令 $\frac{1}{12}(b-a)^2 = \sigma^2$，且 $a = -b$，则有 $b = \sqrt{3\sigma^2}$。)

5. 代码

PyTorch 中的 `torch.nn.init` 模块提供了 Xavier 初始化的实现。我们可以使用 `torch.nn.init.xavier_uniform_` 或 `torch.nn.init.xavier_normal_` 函数来对模型的权重进行 Xavier 初始化。

`torch.nn.init.xavier_uniform_` 函数从均匀分布 $U(-v, v)$ 中抽取权重，其中

$$v = \sqrt{3 \times Var(w)} = \sqrt{\frac{6}{n_{in} + n_{out}}}$$

`torch.nn.init.xavier_normal_` 函数从正态分布 $N(0, \sigma^2)$ 中抽取权重，其中

$$\sigma = \sqrt{Var(w)} = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

n_{in} 和 n_{out} 分别是权重的输入节点数和输出节点数。

发布于 2023-09-01 16:17 · IP 属地奥地利