数值稳定性

梯度爆炸与梯度消失

对于多层神经网络，损失函数对输入的梯度使用链式法则每一项偏导相乘后呈现指数型增长

- 梯度爆炸：反向传播时，梯度越来越大，导致模型发散
  - 值超出值域inf，对16位浮点数比较严重
  - 对学习率敏感，如果学习率太大就导致更大的参数值导致更大的梯度
  - 学习率过小就会使得训练无法进展
  - 我们可能需要在训练过程中不断调整学习率
- 梯度消失：反向传播时，梯度越来越小，到时模型无法学习
  - 使用sigmoid作为激活函数如果输入过大过小就会导致梯度很小
  - 梯度值变为0，对16位浮点数比较严重
  - 训练无法进展
  - 对于底层尤为严重
    数值过大过小的时候都会导致数值问题，常发生在深度模型中

模型初始化与激活函数

提高模型稳定性
目标是将梯度控制在合理的范围内
将梯度乘法变为加法
梯度归一化压缩
使用合适的权重初始化和激活函数
让每一层的方差是一个常数
将每一层输出和梯度看做随机变量，让它们的均值和方差都保持

权重初始化
- 在合理的区间内随机初始参数
- 训练开始的时候更加容易数值不稳定
- 远离最优解的地方损失函数可能很复杂
- 最优解附近的地方表面可能比较平
- 使用`N(0,0.01)`，来初始

自定义神经网络

层与块

1. 连接网络
   可以使用`nn.Sequential`构造网络，`nn.Sequential`常用于快速构建简单的网络，按照顺序连接多个层
2. 自定义网络结构
   可以通过继承`nn.Module`自定义网络结构
3. 自定义`Sequential`容器类
4. 模型嵌套与融合通过多个block实现网络的嵌套

参数管理与初始化

1. 参数访问
   可以通过`.parameters()/.named_parameters()`来获取模型参数
   使用`.state_dict()`查看参数值
   使用索引访问网络特定层的参数
2. 参数初始化
   参数初始化可以通过`apply()`方法将初始化函数应用到所有的子模块
   常用初始化方法有正态分布，常数，Xavier等
   另外还可以多次使用同一层实现共享权重