正则化(Dropout)

概念与定义

正则化技术是机器学习和深度学习中用来防止模型过拟合的重要方法。过拟合是指模型在训练数据上表现很好，但在未见过的数据(如验证集或测试集)上表现不佳的现象。正则化通过向模型添加额外的信息或约束来解决这个问题。常见的正则化技术有：L1 正则化、L2 正则化和 Dropout。

L1 和 L2 正则化

- L1 正则化：也称为Lasso回归，它在损失函数中加入权重系数的绝对值之和作为惩罚项。L1 正则化倾向于将一些权重变为零，从而实现特征选择。即：

$$L1 = L_{\text{原始}} + \sum_{i=1}^{n} |w_i|$$

- L2 正则化：也称为Ridge回归，它在损失函数中加入权重系数平方和的倍数作为惩罚项。与 L1 正则化不同，L2 正则化不会使权重变为零，而是均匀缩小所有参数，接近于零。即：

$$L2 = L_{\text{原始}} + \sum_{i=1}^{n} w_i^2$$

L1 和 L2 正则化都是通过调整正则化参数的大小来控制对模型复杂度的惩罚力度。

Dropout

Dropout是一种专用于神经网络的正则化技术。在训练过程中，Dropout随机地“丢弃”网络中的部分神经元(即设置其输出为0)，从而打破神经元之间的共适应关系，迫使网络学习更鲁棒的特征表示。在预测阶段，所有神经元都会被保留使用。PyTorch等框架提供了nn.Dropout()方便开发者使用这一技术。

nn.Dropout()的原理

在每次前向传播时，nn.Dropout() 会以一定的概率 p 将部分神经元的输出置为0，被保留下来的神经元的输出会被缩放，以保证期望值不变。

- 丢弃概率 p : 表示每个神经元被置为零的概率。
- 保留概率 $1-p$:表示每个神经元被保留的概率。
- 缩放因子：为了保持输出的期望值不变，在训练时，保留下来的神经元输出会被乘以 $1/(1-p)$。

例如，如果 $p=0.6$，则表示有 60% 的神经元被置为0，而保留下来的神经元输出会被放大 $1/(1-p)=2.5$ 倍。

以下是一个简单的PyTorch示例，展示了如何使用Dropout正则化技术：

```
import torch
import torch.nn as nn

# 设置随机数种子
torch.manual_seed(12)

# 创建一个含有 5 个元素的张量
x = torch.Tensor([1, 2, 3, 4, 5])
# 创建一个 Dropout 层，丢弃概率为 0.6
dropout = nn.Dropout(p=0.6)
# 在张量 x 上应用 Dropout 层，并打印结果
print(dropout(x))
```

输出结果为：tensor([0.0000, 0.0000, 7.5000, 10.0000, 0.0000])。可以看到，由于丢弃概率为 0.6，所以张量 x 的 3 个元素被置为 0，而其他 2 个元素被保留且变为原来的 2.5 倍。