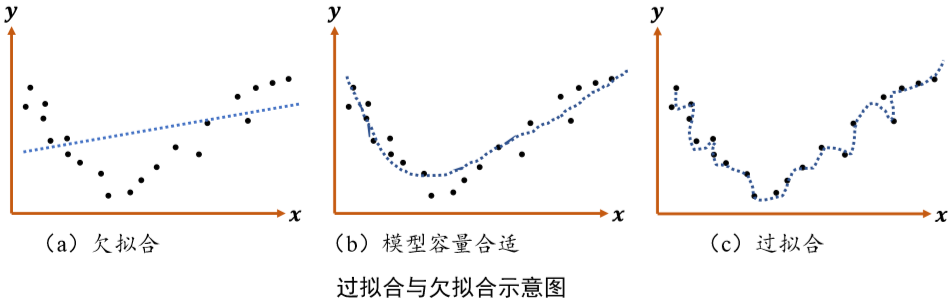
欠拟合与过拟合

机器学习的主要目的是从训练集上学习到数据的真实模型，从而能够在未见过的测试 集上也能够表现良好，我们把这种能力叫做泛化能力。通常来说，训练集和测试集都采样自某个相同的数据分布𝑝(𝑥)。采样到的样本是相互独立的，但是又来自于相同的分布，这种假设叫做独立同分布假设(Independent Identical Distribution assumption，简称 i.i.d.)。

模型的表达能力是指模型拟合复杂函数的能力。当模型的表达能力偏弱时，比如单一线性层，它只能学习到线性模型，无法良好地逼近非线性模型；但模型的表达能力过强时，它就有可能把训练集的噪声模态也学到，导致在测试集上面表现不佳的现象(泛化能力偏弱)。因此针对不同的任务，设计合适容量的模型算法才能取得较好的泛化性能。

当模型的容量过大时，网络模型除了学习到训练集数据的模态之外，还把额外的观测误差也学习进来，导致学习的模型在训练集上面表现较好，但是在未见的样本上表现不佳，也就是模型泛化能力偏弱，这种现象叫作过拟合(Overfitting)。当模型的容量过小时，模型不能够很好地学习到训练集数据的模态，导致训练集上表现不佳，同时在未见的样本上表现也不佳，这种现象叫作欠拟合(Underfitting)。



# 如何解决欠拟合

增加神经网络的层数、增加网络参数量

# 如何防止过拟合

1. 加正则化项
2. 降低模型的复杂度
3. 减少训练次数
4. 加大学习步幅
5. 连续特征分箱
6. 特征降维
7. 正负样本调整均衡
8. 采用多个弱分类器做集成学习
9. 数据清洗
10. 剔除离群值
11. 交叉训练和验证
12. 加大数据集
13. 修改激活函数
14. 引入batch norm
15. Dropout
16. 提前停止
17. 降低学习率

减少神经网络层数，增加训练样本数（也叫数据增强（Data Augmentation）），降低网络的参数量，添加正则化，提前停止（对于同一网络结构，在训练初期，训练集和测试集准确率都是提升的，但随着训练epoch的增加，训练集的准确率持续提升，而测试集的准确率反而下降，此时发生了过拟合的现象，可以采用提前停止（early stopping）的方式防止过拟合。如何选择在何时停止呢？可以记录模型的验证准确率，并监控验证集准确率的变化，当发现验证集准确率连续𝑛个Epoch没有下降时，可以预测可能已经达到了最适合的 Epoch 附近，从而提前终止训练），正则化，dropout

## 正则化

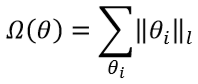
正则化一般通过在损失函数上添加额外的参数稀疏性惩罚项实现，在未加约束之前的优化目标是：



对模型的参数添加额外的约束后，优化的目标变为：



其中𝛺(𝜃)表示对网络参数𝜃的稀疏性约束函数。一般地，参数𝜃的稀疏性约束通过约束参数 𝜃的𝐿范数实现，即：



其中‖𝜃𝑖‖𝑙表示参数𝜃𝑖的𝑙范数。

新的优化目标除了要最小化原来的损失函数ℒ( ,𝑦)之外，还需要约束网络参数的稀疏 性𝛺(𝜃)，优化算法会在降低ℒ( ,𝑦)的同时，尽可能地迫使网络参数𝜃𝑖变得稀疏，它们之间 的权重关系通过超参数𝜆来平衡。较大的𝜆意味着网络的稀疏性更重要；较小的𝜆则意味着 网络的训练误差更重要。通过选择合适的𝜆超参数，可以获得较好的训练性能，同时保证网 络的稀疏性，从而获得不错的泛化能力。常用的正则化方式有 L0、L1、L2 正则化。

### L0正则化

L0正则化是指采用 L0 范数作为稀疏性惩罚项𝛺(𝜃)的正则化计算方式，即：

其中L0范数定义为𝜃𝑖中非零元素的个数。通过约束的大小可以迫使网络中的连接权值大部分为0，从而降低网络的实际参数量和网络容量。但是由于L0范数并不可导，不能利用梯度下降算法进行优化，在神经网络中使用的并不多。

### L1 正则化

采用 L1 范数作为稀疏性惩罚项𝛺(𝜃)的正则化计算方式叫作L1正则化，即

其中，L1范数定义为张量中所有元素的绝对值之和。L1 正则化也叫 Lasso Regularization，它是连续可导的，在神经网络中使用广泛。

采用TensorFlow实现L1正则化的代码如下：

|  |
| --- |
| # 创建网络参数 w1,w2  w1 = tf.random.normal([4,3])  w2 = tf.random.normal([4,2])  # 计算 L1 正则化项  loss\_reg = tf.reduce\_sum(tf.math.abs(w1)) + tf.reduce\_sum(tf.math.abs(w2)) |

### L2 正则化

采用L2范数作为稀疏性惩罚项𝛺(𝜃)的正则化计算方式叫做L2正则化，即：

其中，L2范数定义为张量中所有元素的平方和。L2正则化也叫Ridge Regularization，它和L1正则化一样，也是连续可导的，在神经网络中使用广泛。

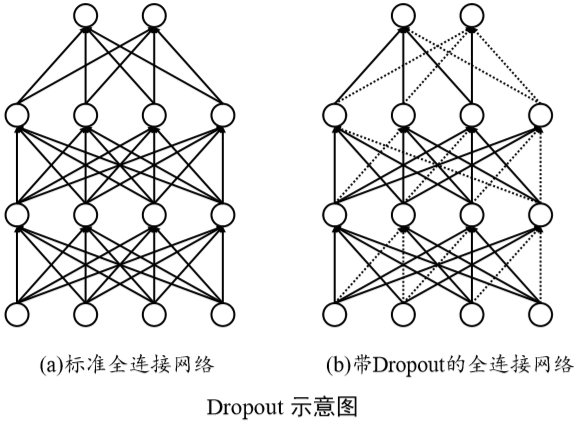
采用TensorFlow实现L2正则化的代码如下：

|  |
| --- |
| # 创建网络参数 w1,w2  w1 = tf.random.normal([4,3])  w2 = tf.random.normal([4,2])  # 计算 L2 正则化项  loss\_reg = tf.reduce\_sum(tf.square(w1)) + tf.reduce\_sum(tf.square(w2)) |

## Dropout

Dropout通过随机断开神经网络的连接，减少每次训练时实际参与计算的模型的参数量；但是在测试时，Dropout 会恢复所有的连接，保证模型测试时获得最好的性能。

下图是全连接层网络在某次前向计算时连接状况的示意图。图(a)是标准的全连接神经网络，当前节点与前一层的所有输入节点相连。在添加了Dropout功能的网络层中，如图(b)所示，每条连接是否断开符合某种预设的概率分布，如断开概率为𝑝的伯努利分布。图(b)中的显示了某次具体的采样结果，虚线代表了采样结果为断开的连接线，实线代表了采样结果不断开的连接线。



在TensorFlow 中，可以通过 tf.nn.dropout(x, rate)函数实现某条连接的Dropout功能， 其中rate参数设置断开的概率值𝑝。示例代码如下：

|  |
| --- |
| # 添加 dropout操作，断开概率为 0.5  x = tf.nn.dropout(x, rate=0.5)  # 也可以将 Dropout 作为一个网络层使用，在网络中间插入一个 Dropout 层。例如：  # 添加 Dropout 层，断开概率为 0.5  model.add(layers.Dropout(rate=0.5)) |

参考：

龙龙老师。TensorFlow深度学习