

1、什么是机器学习中的训练集与测试集？（1星）

训练集是用来训练机器学习模型的数据集，通常包含大量的样本数据和对应的标签。在训练过程中，模型会利用训练集中的数据来学习样本数据的特征和规律，并调整模型的参数以提高模型的预测能力。

测试集是用来测试机器学习模型预测能力的数据集，也包含大量的样本数据和对应的标签。测试集中的样本数据和训练集中的数据具有相同的特征分布和标签分布。在测试过程中，模型会根据测试集中的数据对模型进行评估，并计算模型的预测能力。

训练集和测试集的存在意义在于，它们可以帮助我们评估机器学习模型的泛化能力。泛化能力是指机器学习模型对新数据的适应能力，它是机器学习中非常重要的一个指标。通过将数据集分为训练集和测试集，我们可以在训练过程中评估模型的预测能力，并根据测试结果来调整模型的参数以提高模型的泛化能力。

需要注意的是，训练集和测试集需要保持独立和相互隔离，即测试集中的数据不能在训练过程中使用，对于同一个数据，它们要么在训练集里，要么在测试集里，以免模型出现过拟合问题。同时，在使用测试集进行模型评估时，需要注意选择合适的评估指标和评估方法，以保证评估结果的可靠性和准确性。

2、什么是模型的过拟合与欠拟合？它们产生的原因分别是什么？（3星）

模型的过拟合指的是模型在训练数据上表现很好，但在测试数据上表现较差的现象；而模型的欠拟合指的是模型在训练数据和测试数据上表现都较差的现象。这两种现象都是机器学习中经常遇到的问题。

产生过拟合的主要原因是**一、模型太过复杂**，以至于它学习了训练数据中的噪声和随机误差，而忽略了真实的模式和规律。**二、训练数据的数量不足**也可能导致过拟合。即使是简单的模型，如果训练样本的数量很少，对少数样本重复学习多次也容易发生过拟合。

产生欠拟合的主要原因是**模型过于简单**，难以捕捉数据中的复杂模式和规律，或者模型参数设置不当，导致模型无法拟合数据。同样，**训练数据的数量不足**，对少数样本学习次数不足也可能导致欠拟合。

为了解决过拟合和欠拟合问题，我们需要选择合适的模型和参数，增加训练数据的数量和多样性，以及使用正则化等技术来约束模型复杂度。

3、什么是正则化？它主要用来解决什么问题？主要有哪几种正则化方法？（3星）

正则化是一种用于降低模型复杂度、防止过拟合的技术。通过在模型损失函数中添加正则化项来限制模型的参数大小，从而防止模型过度适应训练数据和在未知数据上表现不佳的问题。

主要用途包括：

1. 减少模型对训练数据的过度拟合。
2. 控制模型的复杂度，防止模型过于复杂。
3. 提高模型的泛化能力，让模型能够准确地预测未知数据。

主要的正则化方法包括：

1. L1 正则化 (Lasso正则化)：通过向损失函数中添加 L1 范数的惩罚项来减少参数的数量，以此达到降低模型复杂度的目的。
2. L2 正则化 (Ridge正则化)：通过向损失函数中添加 L2 范数的惩罚项来限制参数的大小，以降低过拟合的风险。
3. Elastic-Net 正则化：是 L1 正则化和 L2 正则化的权衡，通过在损失函数中同时添加 L1 和 L2 范数的惩罚项来平衡两种正则化方法的优缺点。

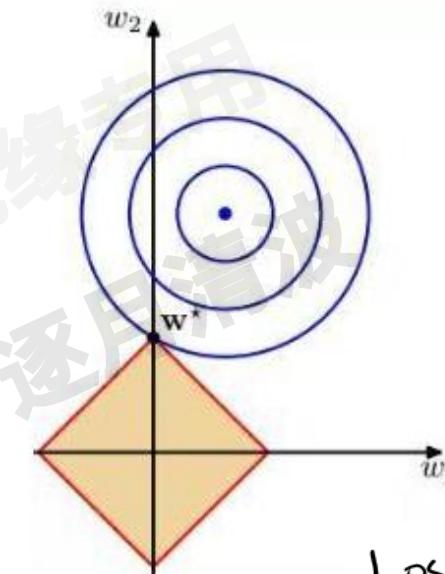
4、简要阐述一下L1正则化与L2正则化的原理与区别。（4星）

L1正则化和L2正则化的目的是为了防止模型过拟合。它们的主要区别在于它们的**惩罚项不同**。

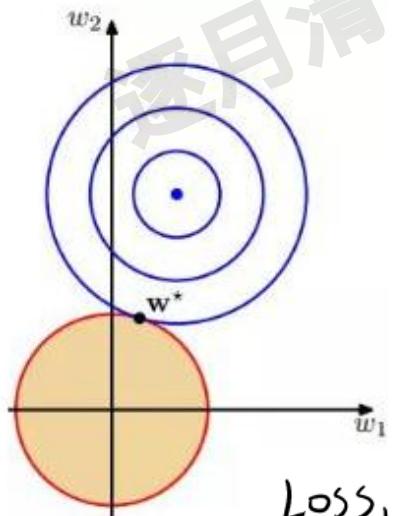
L1正则化指的是在损失函数的基础上，加上**所有参数的绝对值之和**（乘以一个常数），用于惩罚参数过大，促使模型更加稀疏化。L1正则化使得绝大部分的参数都为0，从而达到**特征选择(feature selection)**的效果，在一些需要特征选择的问题上表现得更加优秀。但L1正则化的导数在0时**不可导**，使得优化问题更加复杂。

L2正则化指的是在损失函数的基础上，加上**所有参数平方的和**（乘以一个常数），用于惩罚参数过大，使模型更加平滑。L2正则化保留了更多的参数，使其更加**连续和稳定**，此外，L2正则化的导数在所有位置都**可导**，优化问题比L1正则化要简单，更加常用。

需要注意的是，L1正则化和L2正则化之间不存在一定的优劣关系，选择哪个正则化方法需要根据具体的问题与实验结果来决定。



$$\text{Loss}_{L1} = \text{Loss} + \lambda \sum |w|$$



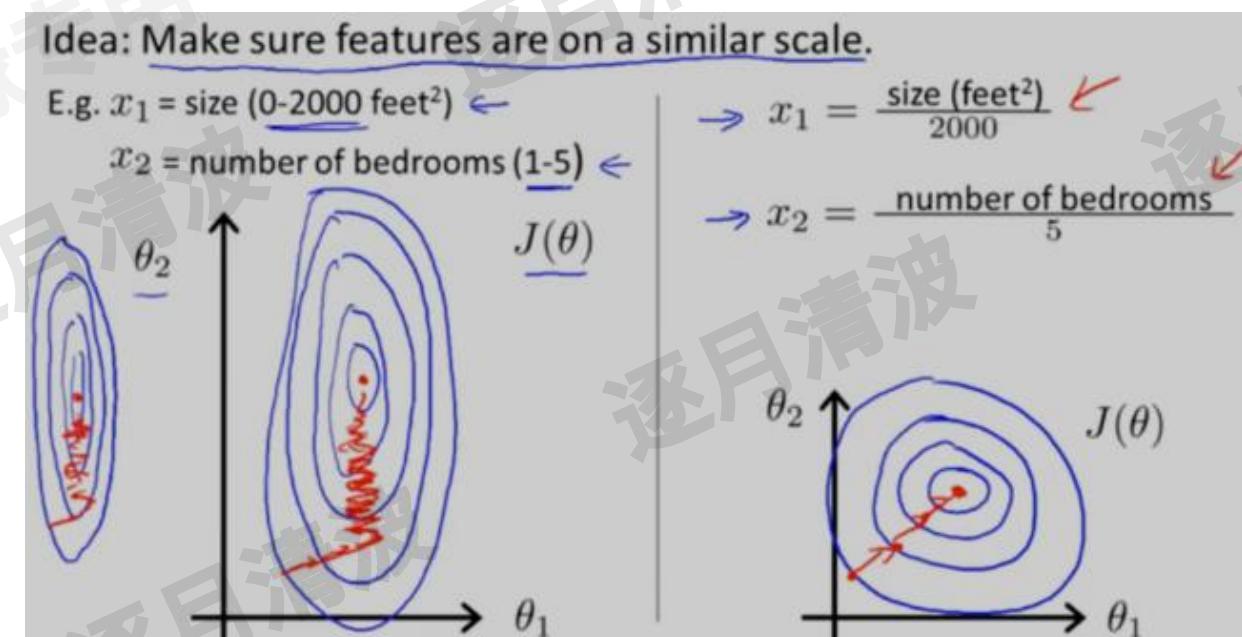
$$\text{Loss}_{L2} = \text{Loss} + \lambda \sum |w|^2$$

5、什么是特征归一化 (Normalization) ?为什么要如此操作? (2星)

特征归一化指将数据中每个特征的值按照一定的比例缩放，以使特征值落入一个**特定的范围内**，例如 $[0, 1]$ 或者 $[-1, 1]$ 。这样做有利于模型的训练和预测。具体来说，特征归一化的操作有以下几个好处：

1. 提高模型收敛速度。归一化可以使得每个特征值的取值范围接近，使梯度下降更容易找到损失函数最优解。
2. 提高模型精度。通过特征归一化，所有的特征值被转换成同样的尺度，避免因为某些特征值太大，导致其他特征对模型的贡献相对较小。
3. 帮助可视化。在二维或三维空间中更容易可视化和理解。

需要注意的是，特征归一化应当在**训练和测试数据上同时进行**，这样才能保证数据具有相同的标准化尺度，从而保证模型的泛化能力。



6、最常用的特征归一化方法有哪些？(3星)

Rescaling (min-max normalization、range scaling):

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

将每一维特征线性映射到目标范围 $[a, b]$ ，即将最小值映射为 a 。最大值映射为 b ，常用目标范围为 $[0, 1]$ 和 $[-1, 1]$ 。

特别地，映射到 $[0, 1]$ 计算方式为：

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Mean normalization:

$$x' = \frac{x - \bar{x}}{\max(x) - \min(x)}$$

将均值映射为 0，同时用最大值最小值的差对特征进行归一化，一种更常见的做法是用标准差进行归一化

Standardization (Z-score Normalization, 更多时候称作标准化):

$$x' = \frac{x - \bar{x}}{\sigma}$$

Scaling to unit length:

将每个样本的特征向量除以其长度，即对样本特征向量的长度进行归一化，长度的度量常使用的是 L2 norm (欧氏距离)，有时也会采用 L1 norm:

$$x' = \frac{x}{\|x\|}$$

(注：容易让人困惑的一点是指代混淆，Normalization 有时会指代 min-max normalization，有时会指代 Standardization，有时会指代 Scaling to unit length。)

7、什么是类别特征？如何处理类别特征？（3星）

类别特征（Categorical Feature）是指在数据集中以类别形式表示的特征。这类特征通常包含一组离散的、非连续的值，可以是有限的或无限的。分类特征的值通常表示某种类别或标签，如性别、血型、国家等。分类特征可能是有序的（例如评级：优秀、良好、一般）或无序的（例如颜色：红、蓝、绿）。

在对数据进行预处理时，对于类别型特征的处理通常有以下几种方法：

Label Encoding: 将每个类别映射为一个整数值。适用于类别间存在大小关系的情况，如年龄、学历等。

One-Hot Encoding: 将每个类别转化为一个二进制向量，向量的长度等于类别数。适用于类别间不存在大小关系的情况，例如颜色、品牌等。

Count Encoding: 将每个类别映射为该类别在训练集中出现的频数。适用于类别数较多的情况，可以减少维度。

Target Encoding: 将每个类别映射为该类别在训练集中的目标变量（标签）的平均值。适用于类别数较多的情况，可以在保持类别信息的同时，降低维度。

8、什么是多项式特征？如何处理多项式特征？（2星）

多项式特征是指通过原始特征的乘法和幂运算所组成的新的特征。它可以较好地捕获原始特征之间的相互作用。

例如，对于一个二维的数据点 (x_1, x_2) ，我们可以将它的特征表示为 $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ ，其中包含了原始特征以及它们的交叉项和二次项。

例如，在自然语言处理（NLP）中，经常使用多项式特征来表示文档中单词的出现次数。这种情况下，可以使用多项式朴素贝叶斯分类器（Multinomial Naive Bayes Classifier）对文档进行分类。多项式朴素贝叶斯分类器假设特征之间是条件独立的，并利用贝叶斯定理根据给定特征计算类别的概率。

多项式特征通常需要预处理，以便在机器学习模型中使用。例如，对于文本数据，可能需要将文本分词、去除停用词、词干提取或词形还原等。在处理多项式特征时，可能还需要进行特征选择、降维和归一化等操作，以提高模型性能。

9、什么是组合特征？如何处理组合特征？（2星）

组合特征（Combinatorial Feature）是指由多个特征组合而成的新特征。例如，在一个电商推荐系统中，可以将商品的类别、价格、品牌等多个特征组合起来，形成一个新组合特征，例如“高价位、大牌子、男性服装”等。

由于组合特征包含多个特征的信息，可以提高特征的表达能力，从而提高模型的预测准确性。但同时，组合特征也会带来高维度的问题，导致模型的训练和预测时间变慢，容易出现过拟合等问题。

针对高维组合特征，常用的处理方法包括：

1. 特征选择（Feature Selection）：从原始特征中选出最相关、最有代表性的特征。常用的特征选择方法包括Filter、Wrapper和Embedding等。
2. 特征降维（Feature Dimension Reduction）：通过降维技术将高维组合特征映射到低维空间，以减少特征数量和维度。常用的特征降维方法包括PCA、LDA、t-SNE等。
3. 特征交叉（Feature Cross）：将原始特征两两交叉，形成新的组合特征。该方法可以有效增加模型的表达能力，但也容易带来高维度问题。

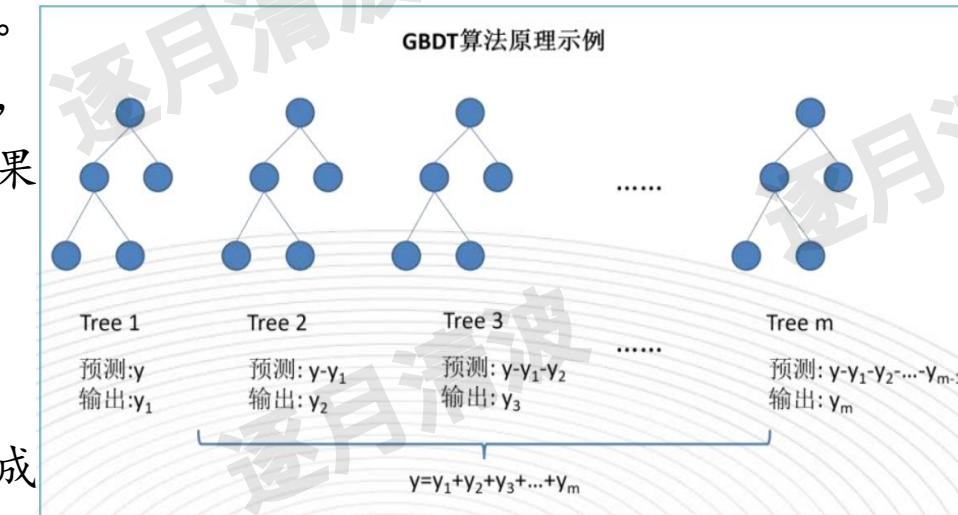
10、什么是GBDT？它有什么优点？（3星）

GBDT全称为**Gradient Boosting Decision Tree**, 是一种基于决策树模型的集成学习方法。它采用的是串行的加法模型, 即每一步训练的目的是减少上一步训练的残差, 从而逐步拟合数据的实际分布。

具体来说，GBDT通过一系列基于决策树的弱学习器组成一个强学习器。在每一次训练中，GBDT会根据上一轮的残差更新当前训练数据的权重，并使用新的数据集来训练下一棵决策树，最终将所有决策树的预测结果相加得到最终结果。

GBDT的优点主要有

高准确性：GBDT算法是集成算法，它将多个决策树的预测结果进行集成从而提高模型的准确性。可以处理非线性数据：可以很好地处理非线性数据，因为每棵决策树采用分段常数函数的方式拟合数据，从而不需要对数据进行线性变换。可以进行特征选择：可以考虑每个特征的重要性从而可以进行特征选择，从而简化模型并提高性能。对异常点和噪声的鲁棒性较高：通过多次迭代，逐渐减少每棵决策树的误差，从而减少模型对异常点和噪声数据的敏感性。



11、什么是XGBoost？它有什么优点？(3星)

XGBoost全称为eXtreme Gradient Boosting，它主要以决策树为基础模型，通过集成多个决策树来提高模型的准确度和鲁棒性。它的原理与传统的梯度提升决策树（Gradient Boosting Decision Tree, GBDT）类似，基于一种叫作Gradient Boosting的集成学习算法，它将多个弱学习器（决策树）组合成一个强学习器。

具体来说，它首先构建一个简单的决策树，对训练集进行拟合，然后计算预测误差（即真实值与预测值之间的差距），接着再根据这个误差构建一棵新的决策树，在此基础上将两棵决策树的预测结果相加，最终得到新的预测值。这个过程会不断重复，即每次用新的决策树去纠正前面所有决策树的预测误差，最终将它们组合成一个强学习器。XGBoost在原有GBDT的基础上，对模型的正则化项、损失函数和节点分裂方式等进行了优化，采用了一种二阶泰勒展开的方法来逼近损失函数的最优解。

它的主要优点有：高效的训练和预测速度：采用了多线程和缓存访问等技术，可以在大规模数据集上高效地进行训练和预测。支持多种损失函数和目标函数：可以灵活地应对不同的任务需求。内置的正则化方法：可以有效地防止过拟合和提高模型的泛化能力。可解释性强：可以输出特征重要性、树结构等信息，方便进行模型解释和可视化。可扩展性强：支持分布式训练和并行计算，可以应对海量数据的训练需求。

- Start from constant prediction, add a new function each time

$$\begin{aligned}\hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ &\dots \\ \hat{y}_i^{(t)} &= \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \end{aligned}$$

Model at training round t Keep functions added in previous round

New function

12、什么是LightGBM？它有什么优点？（2星）

LightGBM (Light Gradient Boosting Machine) 是一个微软开源的实现GBDT算法的框架，主要采用梯度提升算法 (Gradient Boosting) 和单边梯度采样 (Gradient-based One-Side Sampling, GOSS) 进行优化。其核心思想是将许多简单的决策树模型 (弱分类器) 集成起来，形成一个强分类器，通过对样本进行定向的学习，逐步提高模型的预测能力。

具体来说，LightGBM的优化过程主要包括两个阶段：构建决策树和更新叶子节点。在构建决策树阶段中，LightGBM通过使用GOSS算法对样本进行采样，保留具有较大梯度值的样本，忽略掉梯度值较小的样本，从而提高训练效率。在更新叶子节点阶段中，LightGBM采用基于梯度的直方图算法 (Gradient-based Histogram) 对特征进行离散化，利用直方图算法分组并计算特征值的分布情况，快速寻找最优分割点，从而加快了决策树的构建过程。

LightGBM的优点有：

- 更快的训练速度
- 更低的内存消耗
- 更好的准确率
- 分布式支持，可以快速处理海量数据

13、什么是Bagging思想？(4星)

Bagging的全称是bootstrap aggregating，思想就是从总体样本当中随机取一部分样本进行训练，通过多次这样的结果，进行投票获取平均值作为结果输出，这就极大可能的避免了不好的样本数据，从而提高准确度。因为有些是不好的样本，相当于噪声，模型学入噪声后会使准确度不高。

举个例子：

假设有1000个样本，如果按照以前的思维，是直接把这1000个样本拿来训练，但现在不一样，先抽取800个样本来进行训练，假如噪声点是这800个样本以外的样本点，就很有效的避开了。重复以上操作，提高模型输出的平均值。

Bagging的主要优势在于通过降低了模型方差，提高了模型的预测能力和泛化能力，适用于大部分机器学习算法，特别是对于容易过拟合的模型具有很好的效果。此外，Bagging可以使用并行算法优化，提高模型训练的效率。

需要注意的是，Bagging并不能减小模型的偏差，因为它只是通过使用多个模型的预测结果来减少模型的方差。通常情况下，我们需要同时使用Bagging和其他减小偏差的方法，例如Boosting等，才能达到更好的效果。

14、什么是随机森林？它有什么特点？(3星)

Random Forest(随机森林)是一种基于树模型的Bagging的优化版本，一棵树的生成肯定不如多棵树，因此就有了随机森林，解决决策树泛化能力弱的特点。而同一批数据，用同样的算法只能产生一棵树，这时Bagging策略可以帮助我们产生不同的数据集。

每棵树的按照如下规则生成：

如果训练集大小为N，对于每棵树而言，随机且有放回地从训练集中的抽取N个训练样本，作为该树的训练集；如果每个样本的特征维度为M，指定一个常数 $m < M$ ，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的；每棵树都尽最大程度的生长，并且没有剪枝过程。

一开始我们提到的随机森林中的“随机”就是指的这里的两个随机性。两个随机性的引入对随机森林的分类性能至关重要。它们的引入使得随机森林不容易陷入过拟合，并且具有很好得抗噪能力（如：对缺省值不敏感）。

总的来说，就是随机选择样本数，随机选取特征，随机选择分类器，建立多颗这样的决策树，然后通过这几课决策树来投票，决定数据属于哪一类(投票机制有一票否决制、少数服从多数、加权多数等)。

15、随机森林的优点和缺点分别是什么？（3星）

优点：

- 它能够处理很**高维度** (**feature**很多) 的数据，并且不用做特征选择(因为特征子集是随机选择的)。
- 在训练完后，它能够给出**哪些feature比较重要**。
- 训练速度快，容易做成**并行化**方法(训练时树与树之间是相互独立的)。
- 在训练过程中，能够检测到**feature**间的互相影响。
- 对于不平衡的数据集来说，它可以**平衡误差**。
- 如果有很大一部分的**特征遗失**，仍可以维持准确度。

缺点：

- 随机森林已经被证明在某些**噪音较大的**分类或回归问题上会过拟合。
- 对于有不同取值的属性的数据，**取值划分较多的属性会对随机森林产生更大的影响**，所以随机森林在这种数据上产出的属性权值是不可信的。

16、简述随机森林与GBDT之间有哪些联系与异同？（3星）

相同点：

- 都是由多棵树组成，最终的结果都是由多棵树一起决定。
- RF和GBDT在使用CART树时，可以是分类树或者回归树。

不同点：

- 组成随机森林的树可以并行生成，而GBDT是串行生成
- 随机森林的结果是多数表决表决的，而GBDT则是多棵树累加之和
- 随机森林对异常值不敏感，而GBDT对异常值比较敏感
- 随机森林是减少模型的方差，而GBDT是减少模型的偏差
- 随机森林不需要进行特征归一化。而GBDT则需要进行特征归一化

17、XGBoost与GBDT有什么联系和不同？（2星）

除了算法上与传统的GBDT有一些不同外，XGBoost还在工程实现上做了大量的优化。

总的来说，两者之间的区别可以总结成以下几个方面。

- GBDT是机器学习~~算法~~，XGBoost是该算法的工程~~实现~~。
- 在使用CART作为基分类器时，XGBoost显式地加入了~~正则项~~来控制模型的复杂度，有利于防止过拟合，从而提高模型的泛化能力。
- GBDT在模型训练时只使用了代价函数的一阶导数信息，XGBoost对损失函数进行~~二阶~~泰勒展开，可以同时使用一阶和二阶导数。
- 传统的GBDT采用CART作为基分类器，XGBoost支持~~多种类型的~~基分类器，比如线性分类器。
- 传统的GBDT在每轮迭代时使用全部的数据，XGBoost则采用了与随机森林相似的策略，支持对数据进行~~采样~~。
- 传统的GBDT没有设计对缺失值进行处理，XGBoost能够自动学习出~~缺失值~~的处理策略。

18、XGBoost和LightGBM二者对比各自有什么优劣? (2星)

XGBoost的优势：

相对成熟：XGBoost在机器学习领域已有较长时间的应用经验，且已经被广泛应用于各种竞赛和实际应用中。

稳定性高：XGBoost采用了基于梯度的决策树算法，可以有效地防止过拟合和提高模型的泛化能力，同时也比较稳定，不容易产生异常情况。

可扩展性好：XGBoost支持分布式训练和并行计算。

LightGBM的优势：

训练速度快：LightGBM采用基于直方图的决策树算法和稀疏特征算法，可以在大规模数据集上快速训练和预测。

高准确性：LightGBM在训练和预测效果上都有着较高的准确性和泛化能力。

内存占用小：LightGBM的算法设计较为精简，占用的内存较少，可以应对内存受限的情况。

总的来说：

LightGBM是针对大规模数据（样本量多，特征多）时，对XGBoost算法进行了一些优化，使得速度有大幅度提高，但由于优化方法得当，而精度没有减少很多或者变化不大，理论上还是一个以精度换速度的目的。

19、请举出一些使用机器学习算法解决的任务场景。 (1星)

以下是一些使用学习算法解决的任务的示例：

- **图像识别和分类**: 使用深度学习算法, 如卷积神经网络 (CNN) 来识别和分类图像。
- **语音识别和语音生成**: 使用循环神经网络 (RNN) 和卷积神经网络 (CNN) 等算法来识别和生成语音。
- **自然语言处理**: 使用深度学习算法, 如递归神经网络 (RNN) 、长短期记忆网络 (LSTM) 等来解决自然语言处理问题, 如情感分析、机器翻译、文本分类等。
- **推荐系统**: 使用协同过滤算法、矩阵分解算法等来进行用户商品推荐。
- **异常检测和欺诈检测**: 使用监督学习和无监督学习算法来检测异常和欺诈行为。
- **时间序列预测**: 使用循环神经网络 (RNN) 、长短期记忆网络 (LSTM) 等算法来进行时间序列预测, 如股价预测、气温预测等。
- **机器人控制**: 使用深度学习和强化学习等算法来控制机器人的行为, 如自动驾驶、机器人导航等。
- **游戏AI**: 使用深度强化学习算法来训练游戏AI, 如围棋、象棋等。
-

20、监督学习、无监督学习、半监督学习、强化学习分别是什么？（1星）

- 监督学习 (**Supervised Learning**)：指从标记数据（已知输出）中学习建立输入与输出之间的映射关系，然后根据这个映射关系对新的输入进行预测和分类。监督学习的主要目标是通过训练数据来构建一个准确的模型，使其能够对新的未知数据进行预测和分类。典型的监督学习算法包括：线性回归、逻辑回归、决策树、支持向量机 (SVM)、神经网络等。
- 无监督学习 (**Unsupervised Learning**)：指从无标记数据（未知输出）中学习数据本身的特征和结构，探索数据之间的关系和规律，从而发现隐藏在数据中的信息和知识。无监督学习的主要目标是对数据进行聚类、降维、异常检测、密度估计等操作，为进一步的数据分析和挖掘提供基础。典型的无监督学习算法包括：聚类、主成分分析 (PCA)、奇异值分解 (SVD)、概率图模型等。
- 半监督学习 (**Semi-supervised Learning**)：指同时使用有标记数据和无标记数据进行学习，通过利用有标记数据的信息来辅助无标记数据的学习，从而提高模型的准确性和泛化能力。半监督学习的主要目标是通过有限的有标记数据来训练模型，同时利用大量的无标记数据来提高模型的表现。典型的半监督学习算法包括：半监督聚类、半监督分类、自编码器等。
- 强化学习 (**Reinforcement Learning**)：指通过与环境交互的方式，从试错中学习一种策略，使智能体能够在环境中完成某种任务或达成某个目标。强化学习的主要目标是通过奖励信号来指导智能体的行为，从而使其能够逐步学习和改进策略，实现最优的长期累积奖励。典型的强化学习算法包括：Q-learning、SARSA、Deep Q-Networks (DQN) 等。

21、简述设计一个机器学习流程来解决问题的主要思路。 (2星)

设计一个机器学习流程解决问题通常需要以下步骤：

- **收集和准备数据**: 收集和准备相关的数据集，包括标记数据和未标记数据，并对数据进行预处理和清洗，以确保数据的准确性和可用性。
- **确定问题和目标**: 明确机器学习系统要解决的问题和目标，并根据问题类型选择合适的机器学习算法和模型。
- **特征工程**: 对数据进行特征提取和特征选择，以减少噪声和冗余信息，并提高模型的性能和泛化能力。
- **划分数据集**: 将数据集分为训练集、验证集和测试集，并根据需要进行交叉验证和集成学习等技术来提高模型的性能和泛化能力。
- **训练和优化模型**: 使用训练集和验证集来训练和优化机器学习模型，调整超参数和模型结构，以提高模型的性能和泛化能力。
- **评估和测试模型**: 使用测试集对训练好的模型进行评估和测试，检查模型的性能和泛化能力是否符合预期。
- **部署和监控模型**: 将训练好的模型部署到生产环境中，并进行监控和维护，确保模型的可靠性和稳定性。
- 在设计机器学习系统时，还需要考虑数据隐私和安全等方面的问题，并遵守相关法律法规和行业标准。同时，还需要持续优化和改进机器学习系统，以适应不断变化的市场和业务需求。

22、机器学习算法必要的三个组成部分是什么？(2星)

机器学习算法必要的三个组成部分是：

模型：模型是用来描述输入数据和输出数据之间的关系的数学函数。在机器学习中，我们通过训练模型来逼近真实的数据分布，从而实现对新数据的准确预测和分类。

目标函数：目标函数是用来衡量模型预测结果与真实数据之间的差异的数学函数。在机器学习中，我们通常使用目标函数来指导模型的训练和优化，以最小化预测误差并提高模型的性能和泛化能力。

优化算法：优化算法是用来寻找目标函数的最优解或局部最优解的数学算法。在机器学习中，我们通常使用梯度下降、牛顿法、遗传算法等优化算法来优化模型参数，以最大限度地提高模型的性能和泛化能力。

23、如何理解“Learning can be viewed as using direct or indirect experience to approximate a chosen target function”这句话？(2星)

这句话表达了机器学习的核心思想：利用直接或间接的经验来近似一个目标函数。

在机器学习中，我们通常将输入数据和对应的输出标签作为训练样本，然后使用训练样本来学习一个函数，使得该函数能够对新的输入数据进行预测并输出相应的输出结果。这个函数就是目标函数，也称为学习目标。

直接经验指使用已有的训练样本来直接学习目标函数，通过不断地迭代和优化，逐渐逼近目标函数的真实形式。

间接经验指使用已有的训练样本来学习一个中间函数或模型，然后再通过该中间函数或模型来近似目标函数，例如使用神经网络来学习复杂的非线性函数关系。

无论是直接经验还是间接经验，都是通过使用经验数据来学习目标函数或中间模型，以逼近目标函数的真实形式，从而实现对新数据的准确预测和分类。

因此，可以将机器学习视为利用直接或间接经验来近似目标函数的过程。

24、如何理解 “Function approximation can be viewed as a search through a space of hypotheses (representations of functions) for one that best fits a set of training data” 这句话？(2星)

这句话表达了机器学习中函数逼近的基本思想。在机器学习中，我们通常从**一组假设函数集合**（函数表示）中选择一个函数来逼近训练数据。这些假设函数可以是简单的线性函数，也可以是复杂的神经网络或决策树等。

我们通常会针对训练数据进行函数逼近，也就是找到一组函数表示，使得该函数表示能够最好地适应训练数据，从而对新数据进行预测和分类。这个过程可以看作是在一组假设函数集合中搜索最佳的函数表示，使得该函数表示能够**最小化预测结果与训练数据之间的误差**。

这个搜索过程可以采用不同的算法和技术来实现，如梯度下降、牛顿法、遗传算法等。在搜索过程中，我们需要根据训练数据和目标函数的特性来选择合适的搜索算法和优化策略，以最大限度地提高函数逼近的精度和泛化能力。

因此，可以将函数逼近视为在一组假设函数集合中搜索最佳适应一组训练数据的函数表示的过程。

25、如何理解“Different learning methods assume different hypothesis spaces (representation languages) and/or employ different search techniques”这句话？
(2星)

这句话表达了机器学习中不同学习方法的核心差异。在机器学习中，我们可以使用不同的学习方法来解决不同类型的问题。不同的学习方法通常采用不同的假设空间（即函数表示语言）和搜索技术来构建模型。

假设空间（函数表示语言）指的是模型可以表示的函数集合，不同的假设空间可以表示不同的函数类型，例如线性函数、非线性函数、深度神经网络等。

搜索技术指的是模型学习和参数优化的过程，不同的搜索技术可以使用不同的优化算法和搜索策略来寻找最佳的模型参数或函数表示。

举个例子，对于分类问题，我们可以使用支持向量机（SVM）或者决策树等不同的学习方法来构建分类器。SVM采用线性函数表示，而决策树则采用树形结构表示。在优化过程中，SVM使用凸优化算法，而决策树使用贪心算法。

因此，不同的学习方法在假设空间和搜索技术上存在差异，从而导致了不同的性能和适用范围。在实际应用中，我们需要根据问题类型和数据特点选择合适的学习方法和模型，并采用合适的优化算法和搜索策略来实现模型训练和参数优化。

26、为什么我们需要假设训练和测试样本是从相同分布的数据总体中独立抽取的？(3星)

因为这样可以保证机器学习模型的泛化能力和可靠性。

在机器学习中，我们通常使用训练样本来训练机器学习模型，然后使用测试样本来评估模型的性能和泛化能力。如果训练和测试样本是从不同的数据总体中抽取的，那么模型可能无法准确地泛化到新的数据集上，从而导致模型性能的下降。

因此，我们通常假设训练和测试样本是从相同的数据总体中独立抽取的，以保证机器学习模型的泛化能力和可靠性。这个假设通常被称为**独立同分布 (Independent and Identically Distributed, IID) 假设**。

通过假设训练和测试样本是独立同分布的，我们可以在训练阶段对模型进行良好的优化，并在测试阶段对模型进行准确的评估。同时，这个假设也有助于我们理解模型在实际应用中的适用性和限制性，并提出相应的改进策略和方法。

27、什么是神经网络 (NN)？它的组成与基本原理是什么？(2星)

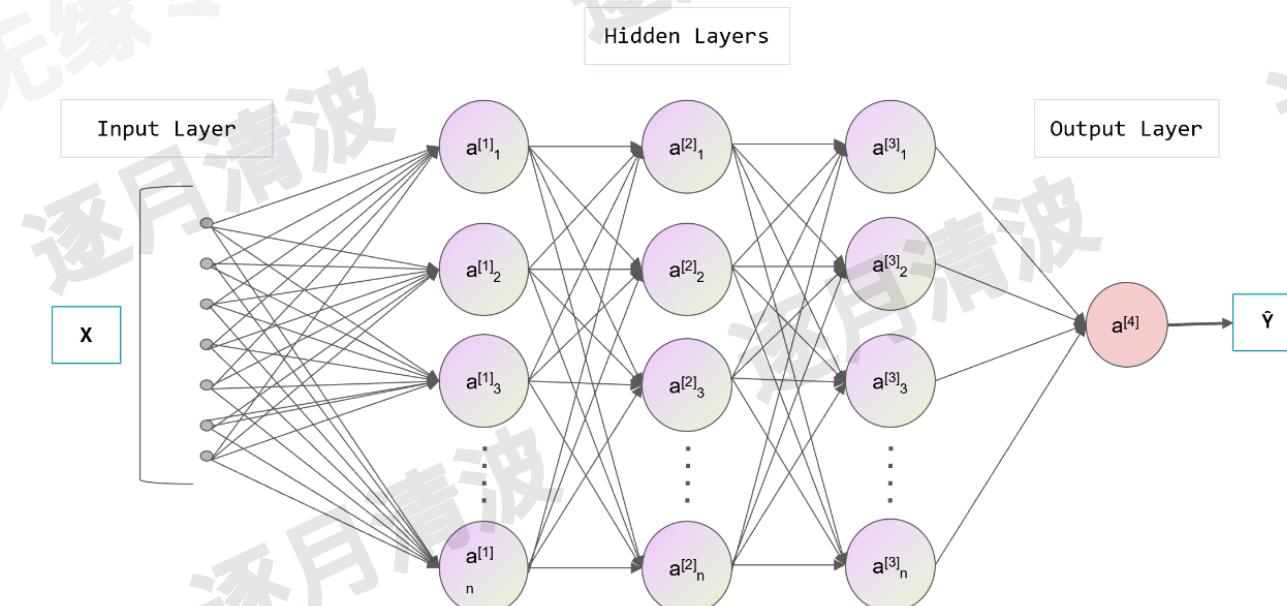
神经网络 (Neural Networks) 是一种模仿生物神经系统的计算模型，用于识别模式、对数据进行分类、预测等任务。一个神经网络由多个层组成，每层包含许多相互连接的神经元（或称节点）。这些神经元模拟生物神经元的工作原理，接收输入，对输入进行加权求和，然后通过激活函数（如ReLU、sigmoid等）生成输出。神经网络的层次结构通常包括输入层、隐藏层和输出层：

输入层：负责接收输入数据，将数据传递到下一层。输入层的神经元数量通常与数据的特征数量相同。

隐藏层：位于输入层和输出层之间的一层或多层神经元。隐藏层的神经元数量和层数可以根据问题复杂性和网络结构进行调整。

输出层：负责生成网络的输出结果，如分类标签或回归值。输出层的神经元数量取决于任务类型和类别数量。

神经网络通过一个称为**反向传播** (Backpropagation) 的训练算法进行学习。在训练过程中，网络通过**梯度下降** (Gradient Descent) 或其他优化算法不断调整连接权重，以最小化预测输出与实际目标之间的误差。



28、什么是支持向量机 (SVM) ? 它有什么优点? (2星)

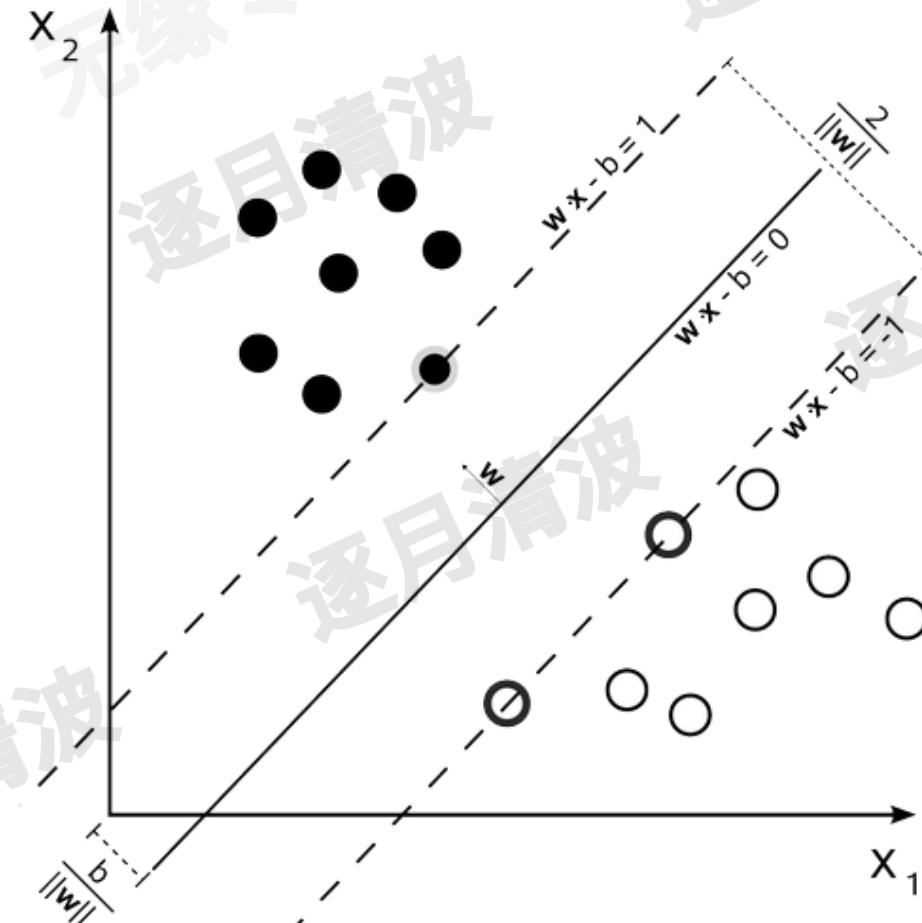
支持向量机 (Support Vector Machine, SVM) 是一种二分类模型，它的基本思想是通过在高维特征空间中寻找最优的超平面来实现对不同类别的数据样本进行分类。在支持向量机中，我们首先将数据样本映射到高维特征空间中，然后寻找一个最优的超平面，使得正负样本分别落在超平面的两侧，并且两侧的间隔最大化。这个最优的超平面被称为分离超平面 (Separating Hyperplane)，可以用来对新的数据样本进行分类。为了找到最优的分离超平面，支持向量机使用了一种称为“间隔最大化” (Maximum Margin) 的优化策略。它将样本点到分离超平面的距离定义为间隔，最大化间隔的同时，也可以实现对噪声数据的鲁棒性和对新样本的泛化能力。

支持向量机具有以下优点：

理论基础: 支持向量机基于统计学习理论和凸优化等数学理论，具有坚实的理论基础和严格的数学证明。

鲁棒性和泛化能力: 支持向量机可以实现对噪声数据的鲁棒性和对新样本的泛化能力，对于复杂的非线性问题也具有较好的分类性能。

可解释性和可视化: 支持向量机通过间隔和支持向量等概念，可以实现对分类结果的可解释性和可视化。



29、什么是决策树 (DT) ? 它有什么优缺点? (2星)

决策树是一种基于**树结构的分类和回归方法**，它通过对样本数据的分析，构建一个树形结构，通过依次对样本的属性进行选择并分裂，使得每个子节点中的样本属于同一类别或拥有相同的预测值。

决策树的**优点**包括：

1. 易于理解和解释，可以可视化展示。
2. 可以处理多类别问题。
3. 可以处理缺失数据。
4. 可以在很短的时间内对大型数据量进行预测。
5. 容易处理数值型数据和离散型数据。
6. 可以通过剪枝避免过拟合现象。
7. 决策树模型具有很好的鲁棒性。
8. 对于需要实时预测的应用非常有效。

需要注意的是，决策树的建模和训练也存在着一些**不足**和挑战，比如决策树有很强的倾向性，很容易陷入局部最优解而无法得到全局最优解；对噪声数据和过多的异常值非常敏感；对于连续型数据的分类效果不如其他方法。

30、简述一下一颗决策树的生成过程。（3星）

决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得各个子数据集有一个最好的分类的过程。这一过程对应着对特征空间的划分，也对应着决策树的构建。

- 1) 构建根节点，将所有训练数据都放在根节点，选择一个最优特征，按着这一特征将训练数据集分割成子集，使得各个子集有一个在当前条件下最好的分类。
- 2) 如果这些子集已经能够被基本正确分类，那么构建叶节点，并将这些子集分到所对应的叶节点去。
- 3) 如果还有子集不能够被正确的分类，那么就对这些子集选择新的最优特征，继续对其进行分割，构建相应的节点，如果递归进行，直至所有训练数据子集被基本正确的分类，或者没有合适的特征为止。
- 4) 每个子集都被分到叶节点上，即都有了明确的类，这样就生成了一颗决策树。
- 5) 剪枝：由于过度拟合的风险，需要对构建的决策树进行剪枝处理。常用的剪枝方法有预剪枝和后剪枝。
- 6) 在实际应用中，还会对决策树进行优化和改进，如增加随机性以提高精度和泛化能力，设置缺省节点等。

31、决策树有哪些常用的启发函数？(4星)

1、**ID3**: 对于样本集合D，类别数为K，则集合D的经验熵表示为 $H(D) = -\sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$

计算某个特征A对于集合D的经验条件熵为： $H(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = \sum_{i=1}^n \frac{|D_i|}{|D|} \left(-\sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \right)$

则信息增益 $g(D, A)$ 表示为二者之差： $g(D, A) = H(D) - H(D | A)$

选取信息增益最大的那个特征，进行分叉。

2、**C4.5**: 在ID3的基础上，特征A对于数据集D的信息增益比为 $g_R(D, A) = \frac{g(D, A)}{H_A(D)}$ $H_A(D) = -\sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$
按照最大的信息增益比来选择特征分叉

3、**CART**: 采取Gini系数，它描述的是数据的纯度： $Gini(D) = 1 - \sum_{k=1}^n \left(\frac{|C_k|}{|D|} \right)^2$

在每一次迭代中，选择Gini系数最小的特征及其对应的切分点进行分类。CART是一颗二叉树，每一次按照特征A的取值分为两份，分别进入左右子树。特征A的Gini系数定义为

$$Gini(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} Gini(D_i)$$

32、决策树中各启发函数之间有什么区别与优缺点？(3星)

- **划分标准**: ID3使用信息增益偏向特征值多的特征，C4.5使用信息增益率克服信息增益的缺点，偏向于特征值小的特征，CART使用基尼指数克服C4.5需要求 \log 的巨大计算量，偏向于特征值较多的特征。
- **使用场景**: ID3和C4.5都只能用于分类问题，CART可以用于分类和回归问题；ID3和C4.5是多叉树，速度较慢，CART 是二叉树，计算速度很快；
- **样本数据**: ID3只能处理离散数据且缺失值敏感，C4.5和CART可以处理连续性数据且有多种方式处理缺失值；从样本量考虑的话，小样本建议C4.5、大样本建议 CART。C4.5处理过程中需对数据集进行多次扫描排序，处理成本耗时较高，而CART本身是一种大样本的统计方法，小样本处理下泛化误差较大；
- **样本特征**: ID3和C4.5层级之间只使用一次特征，CART可多次重复使用特征；

33、如何对决策树进行预剪枝？(3星)

预剪枝，即在生成决策树的过程中提前停止树的增长。

预剪枝的核心思想是在树中结点进行扩展之前，先计算当前的划分是否能带来模型泛化能力的提升，如果不能，则不再继续生长子树。此时可能存在不同类别的样本同时存于结点中，按照多数投票的原则判断该结点所属类别。

预剪枝对于何时停止决策树的生长有以下几种方法：

1. 当树到达一定深度的时候，停止树的生长。
2. 当到达当前结点的样本数量小于某个阈值的时候，停止树的生长。
3. 计算每次分裂对测试集的准确度提升，当小于某个阈值的时候，不再继续扩展。

预剪枝具有思想直接、算法简单、效率高等特点，适合解决大规模问题。但如何准确地估计何时停止树的生长（即上述方法中的深度或阈值），针对不同问题会有很大差别，需要一定经验判断。

预剪枝存在一定局限性，有欠拟合的风险，虽然当前的划分会导致测试集准确率降低，但在之后的划分中，准确率可能会有显著上升。

34、如何对决策树进行后剪枝？(2星)

后剪枝的核心思想是让算法生成一棵完全生长的决策树，然后从最底层向上计算是否剪枝。

剪枝过程将子树删除，用一个叶子结点替代，该结点的类别同样按照多数投票的原则进行判断。同样地，后剪枝也可以通过在测试集上的准确率进行判断，如果剪枝过后准确率有所提升，则进行剪枝。相比于预剪枝，后剪枝方法通常可以得到泛化能力更强的决策树，但时间开销会更大。

常见的后剪枝方法包括：

- 错误率降低剪枝 (Reduced Error Pruning, REP)
- 悲观剪枝 (Pessimistic Error Pruning , PEP)
- 代价复杂度剪枝 (Cost Complexity Pruning, CCP)
- 最小误差剪枝 (Minimum Error Pruning, MEP)
- CVP(Critical Value Pruning)
- OPP (Optimal Pruning)
-

35、为什么CART使用基尼指数而不是交叉熵作为信息增益的衡量标准？(3星)

在决策树中，信息增益可以看作是一个属性对分类的贡献度，用于选择最佳属性作为划分依据。在CART（分类与回归树）中，使用基尼指数与信息增益来选择最佳属性。

其中，基尼指数可以理解为根据属性划分后，随机从数据集中选择两个样本，其类别不一致的概率，即：

$$Gini = \sum_{i=1}^J p_i(1 - p_i)$$

其中，J 表示类别数，Pi 表示第i个类别在样本中的占比。

与基尼指数不同，交叉熵需要计算样本的条件概率和真实标签的对数概率差，它的计算量较大，容易导致过拟合。而基尼指数是一个更加简单的计算方式，计算基尼不纯度的时间复杂度为 $O(n)$ ，是线性级别的，而且更加易于理解。因此，基于基尼指数作为衡量标准，能够在保证准确率的情况下，使得决策树构建的速度更快且更轻量化。

36、如果特征很多，决策树中最后没有用到的特征一定是无用的吗？（2星）

不是。决策树学习算法在每个节点选择划分特征时，考虑的是这个特征能否对数据进行更好的分类。但是当该特征在某个节点上并不能提高分类效果时，算法会将其舍弃。这并不意味着该特征在整个数据集中都是无用的。

可以从两个角度考虑：

- **特征替代性**。如果可以已经使用的特征A和特征B可以提取特征C，特征C可能就没有被使用，但是如果把特征C单独拿出来进行训练，依然有效。
- 决策树的每一条路径就是计算**条件概率**的条件。前面的条件如果包含了后面的条件，只是这个条件在这棵树中是无用的，如果把这个条件拿出来，也可以帮助分析数据。

37、如果异常值或数据分布不均匀，会对决策树有什么影响？（3星）

异常值或数据分布不均匀对决策树的影响可能有以下几个方面：

1. 异常值可能导致决策树过拟合，即过度学习训练数据中的噪声而忽略了一般规律。这样的决策树可能在训练集上表现很好，但在测试集或新数据上表现很差。
2. 异常值可能导致决策树的分裂点选择不合理，即选择了一个不能很好地区分数据集的特征或阈值。这样的决策树可能在某些分支上产生过多或过少的节点，降低了决策树的效率和准确性。
3. 数据分布不均匀可能导致决策树的偏倚，即倾向于学习占比较大的类别或区间的规律而忽略了占比较小的类别或区间的规律。这样的决策树可能在某些类别或区间上表现很好，但在其他类别或区间上表现很差。

为了减少异常值或数据分布不均匀对决策树的影响，可以采取以下一些措施：

1. 删除、替换、修正或标记异常值。
2. 标准化、归一化、离散化或编码数据。
3. 限制决策树的深度、叶子节点数、分裂节点数或纯度增益等。
4. 使用过采样、欠采样、随机采样或代价敏感学习等方法来平衡数据分布。

38、机器学习里分类问题和聚类问题有什么区别？(1星)

分类 (classification) 和聚类 (clustering) 是机器学习中两个常见的问题类型，它们的区别在于目标和方法。

分类是一种有监督学习，目标是将数据分为若干类别中的一类。在分类问题中，模型已经知道每个数据点属于哪个类别，并通过学习大量数据来预测新的未知数据的分类。典型的应用包括垃圾邮件过滤、图像识别、文本分类等。

聚类是一种无监督学习，目标是将数据分为若干组，每组数据之间的相似度高，组内的相似度低。在聚类问题中，模型不知道每个数据点属于哪个类别，而是通过分析数据本身的特征来将其分组。典型的应用包括分析市场、客户分类、基因组分析等。

在分类问题中，我们通常有标注好的数据进行学习；而在聚类问题中，我们通常没有任何标注，只能依靠数据本身的特征进行自然地分组。从方法上看，分类问题通常使用监督学习，需要训练数据和标签；而聚类问题通常使用无监督学习，可以使用各种不同的聚类算法进行分析。

39、聚类问题评判聚类效果的准则是什么？具体量化指标是？（2星）

SSE: 计算的是拟合数据和原始数据对应点的误差的平方和，当SSE越接近于0，说明模型选择和拟合更好。

$$\sum_{i=1}^n (y_i - y_i^*)^2$$

轮廓系数: 适用于实际类别信息未知的情况。对于单个样本，设a是与它同类别中其他样本的平均距离，b是与它距离最近不同类别中样本的平均距离，其轮廓系数为：对于一个样本集合，它的轮廓系数是所有样本轮廓系数的平均值。轮廓系数的取值范围是[-1, 1]，同类别样本距离越相近不同类别样本距离越远，分数越高

$$s = \frac{b-a}{\max(a,b)}$$

Calinski-Harabaz系数: 计算类中各点与类中心的距离平方和来度量类内的紧密度，通过计算各类中心点与数据集中心点距离平方和来度量数据集的分离度，CH指标由分离度与紧密度的比值得到。从而，CH越大代表着类自身越紧密，类与类之间越分散，即更优的聚类结果。（越大越好）。

$$s(k) = \frac{\text{tr}(B_k)m-k}{\text{tr}(W_k)k-1}$$

Davies-Bouldin系数: DB计算任意两类别的类内距离平均距离(CP)之和除以两聚类中心距离求最大值。DB越小意味着类内距离越小同时类间距离越大。（越小越好）。
$$DB = \frac{1}{n} \sum_{i=1}^n \max(j \neq i) \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

40、什么是K近邻算法(KNN) ? 简述其流程。 (2星)

KNN (K-Nearest Neighbor) 算法是机器学习算法中最基础、最简单的算法之一。它既能用于分类，也能用于回归。KNN通过测量不同特征值之间的距离来进行分类。KNN算法的思想非常简单：对于任意n维输入向量，分别对应于特征空间中的一个点，输出为该特征向量所对应的类别标签或预测值。

KNN算法没有一般意义上的学习过程。它的工作原理是利用训练数据对特征向量空间进行划分，并将划分结果作为最终算法模型。存在一个样本数据集合，也称作训练样本集，并且样本集中的每个数据都存在标签，即我们知道样本集中每一数据与所属分类的对应关系。输入没有标签的数据后，将这个没有标签的数据的每个特征与样本集中的数据对应的特征进行比较，然后提取样本中特征最相近的数据（最近邻）的分类标签。

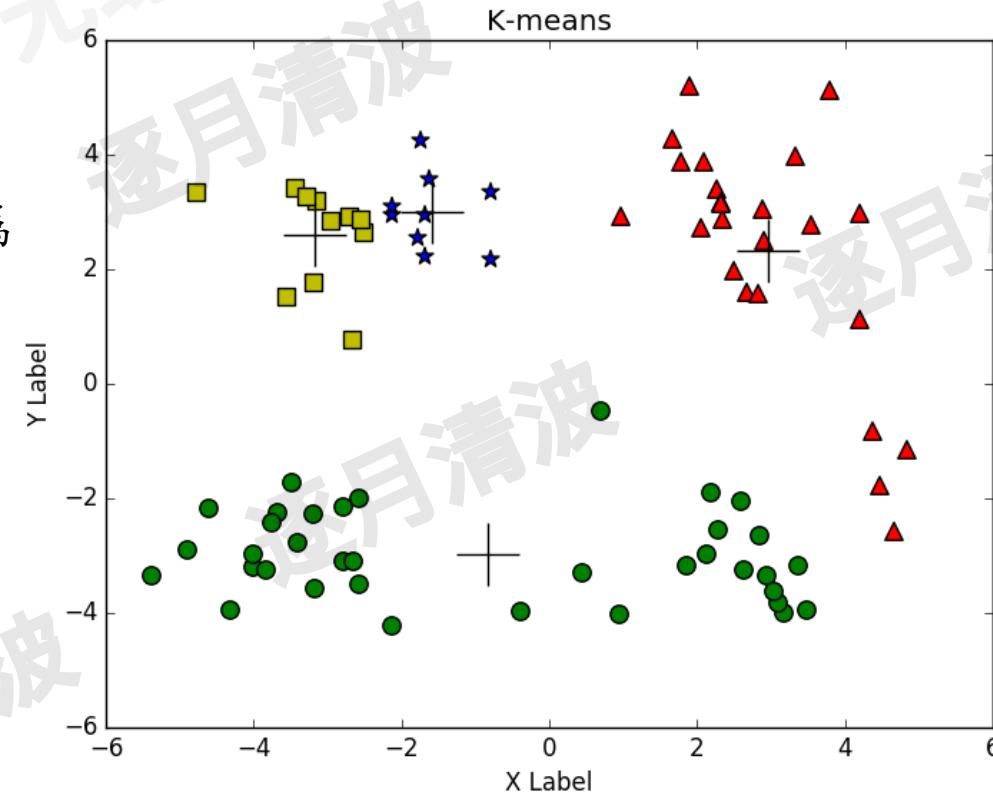
一般而言，我们只选择样本数据集中前k个最相似的数据，这就是KNN算法中K的由来，通常k是不大于20的整数。最后，选择k个最相似数据中出现次数最多的类别，作为新数据的分类。如果k值比较小，相当于我们在较小的领域内训练样本对实例进行预测。这时，算法的近似误差会比较小，因为只有与输入实例相近的训练样本才会对预测结果起作用。但是，它也有明显的缺点：算法的估计误差比较大，预测结果会对近邻点十分敏感，也就是说，如果近邻点是噪声点的话，预测就会出错。因此，k值过小容易导致KNN算法的过拟合。同理，如果k值选择较大的话，距离较远的训练样本也能够对实例预测结果产生影响。这时候，模型相对比较鲁棒，不会因为个别噪声点对最终预测结果产生影响。但是缺点也十分明显：算法的近邻误差会偏大，距离较远的点（与预测实例不相似）也会同样对预测结果产生影响，使得预测结果产生较大偏差，此时模型容易发生欠拟合。因此，在实际工程实践中，我们一般采用交叉验证的方式选取k值。通过以上分析可知，一般k值选得比较小，我们会在较小范围内选取k值，同时把测试集上准确率最高的那个确定为最终的算法超参数k。

41、什么是K均值聚类算法 (K-Means Clustering) ? 简述其流程。 (2星)

K均值聚类算法是一种用于聚类的无监督学习的方法，它可以将数据集划分为K个簇，使得簇内的数据相似度高，簇间的数据相似度低。其流程如下：

1. 确定聚类数量K，随机选取K个点作为初始的聚类中心。
2. 对于每一个数据点，计算其与这K个聚类中心的距离，将其归属到距离最近的聚类中心所在的类别。
3. 更新每个聚类中心的位置，即将归属到该聚类中心类别的所有数据点位置的平均值作为新的聚类中心位置。
4. 重复以上步骤2、3直到聚类中心的位置不再发生变化或达到设定的迭代次数。
5. 最终得到K个聚类。

K均值聚类算法对于初始点的选择会影响最终的聚类效果。因此通常会多次运行，选择最优的一次聚类结果作为最终结果。同时，不同的距离度量方法和距离计算方法也会对算法的效果产生影响。



42、什么是贝叶斯网络？它有什么特点？（1星）

贝叶斯网络（Bayesian networks）也被称为信念网络（belief networks）或概率有向无环图（probabilistic directed acyclic graph），是一种概率图模型，用于描述变量之间的依赖关系和条件概率分布。贝叶斯网络通常用于推理、诊断、决策和预测等领域。

贝叶斯网络由两部分组成：**结点和边**。贝叶斯网络的节点表示随机变量，可以是可观察到的变量或隐变量、未知参数等，边表示变量之间的依赖关系。在贝叶斯网络中，每个变量都有一个条件概率分布，该**分布描述了该变量在给定其所有父节点的取值的情况下**的概率分布。通过这些条件概率分布和网络的结构，我们可以对变量之间的依赖关系进行建模和推理。

贝叶斯网络具有以下特点：

- 可以处理不确定性：贝叶斯网络是一种概率图模型，可以处理变量之间的不确定性和随机性。
- 易于表达和解释：贝叶斯网络的结构和参数可以用图形化的方式表示，易于理解和解释。
- 适用性广泛：贝叶斯网络的应用包括风险评估、医学诊断、推荐系统、自然语言处理、图像识别等多个领域。

43、什么是朴素贝叶斯算法？它有什么特点？(2星)

朴素贝叶斯算法原理是基于概率论的分类算法，利用训练数据学习联合概率分布 $P(X, Y)$ 和条件概率分布 $P(X|Y)$ ，然后根据贝叶斯公式求得后验概率分布 $P(Y|X)$ ，将后验概率最大的类作为输入 X 的类输出。

优点：

朴素贝叶斯算法假设了数据集属性之间是相互独立的。因此算法的逻辑性十分简单，并且算法较为稳定，当数据呈现不同的特点时，朴素贝叶斯的分类性能不会有太大的差异。换句话说就是朴素贝叶斯算法的健壮性比较好，对于不同类型的数据集不会呈现出太大的差异性。当数据集属性之间的关系相对比较独立时，朴素贝叶斯分类算法会有较好的效果。

缺点：

属性独立性的条件同时也是朴素贝叶斯分类器的不足之处。数据集属性的独立性在很多情况下是很难满足的，因为数据集的属性之间往往都存在着相互关联，如果在分类过程中出现这种问题，会导致分类的效果大大降低。

设有样本数据集 $D = \{d_1, d_2, \dots, d_n\}$ ，对应样本数据的特征属性集为 $X = \{x_1, x_2, \dots, x_d\}$ 类变量为 $Y = \{y_1, y_2, \dots, y_m\}$ ，即 D 可以分为 y_m 类别。其中 x_1, x_2, \dots, x_d 相互独立且随机，则 Y 的先验概率 $P_{prior} = P(Y)$ ， Y 的后验概率 $P_{post} = P(Y|X)$ ，由朴素贝叶斯算法可得，后验概率可以由先验概率 $P_{prior} = P(Y)$ 、证据 $P(X)$ 、类条件概率 $P(X|Y)$ 计算出：

$$P(Y|X) = \frac{P(Y) P(X|Y)}{P(X)}$$

朴素贝叶斯基于各特征之间相互独立，在给定类别为 y 的情况下，上式可以进一步表示为下式：

$$P(X|Y = y) = \prod_{i=1}^d P(x_i|Y = y)$$

由以上两式可以计算出后验概率为：

$$P_{post} = P(Y|X) = \frac{P(Y) \prod_{i=1}^d P(x_i|Y)}{P(X)}$$

由于 $P(X)$ 的大小是固定不变的，因此在比较后验概率时，只比较上式的分子部分即可。因此可以得到一个样本数据属于类别 y_i 的朴素贝叶斯计算如下图所示：

$$P(y_i|x_1, x_2, \dots, x_d) = \frac{P(y_i) \prod_{j=1}^d P(x_j|y_i)}{\prod_{j=1}^d P(x_j)}$$

44、什么是实例空间 (Instance space) ? (1星)

实例空间 (instance space) 是指由所有可能的样本实例组成的空间。在机器学习中，实例通常由一组特征和一个标签组成，可以表示为 (x, y) 的形式，其中 x 是特征向量， y 是标签。

实例空间包含了所有可能的样本实例，是机器学习中非常重要的概念之一。在实际应用中，我们需要从实例空间中选择一些样本实例作为训练集，然后使用机器学习算法对训练集进行学习，最终得到一个模型，用于对新的样本进行预测或分类。

实例空间的大小取决于特征的数量和范围。对于离散特征和连续特征，实例空间的大小可能会有所不同。在实际应用中，为了避免过拟合和欠拟合等问题，通常需要对特征进行选择、降维或处理，从而减少实例空间的大小，提高模型的泛化能力。

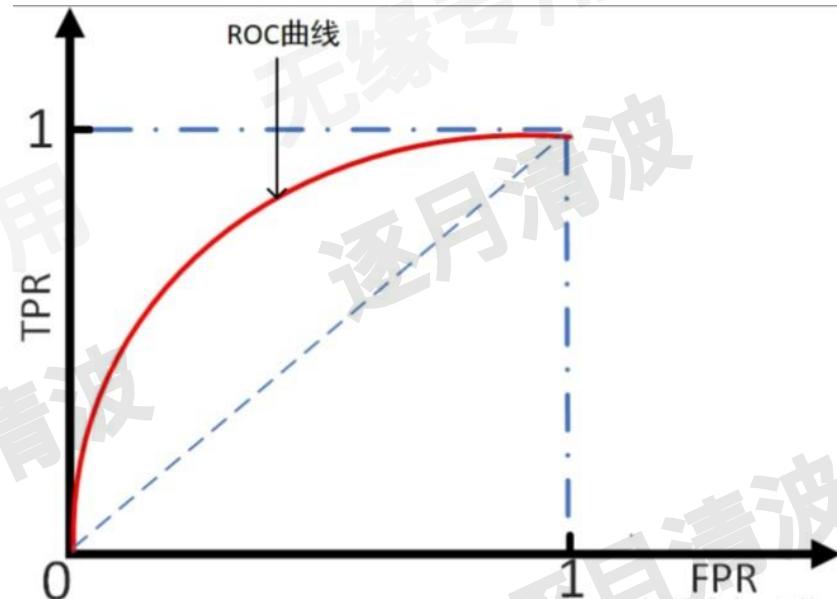
实例空间的概念也被用于一些基于实例的学习算法，如k近邻算法。在这些算法中，我们需要计算测试样本与训练集中每个样本之间的距离，从而找到最近的 k 个邻居，并根据邻居的标签进行分类或回归。因此，实例空间的大小和样本的分布对算法的性能和效率具有重要影响。

45、分类问题常用的评价指标有哪些？(4星)

在分类问题中，模型的性能往往通过预测结果的准确性、召回率、精确率、F1得分、AUC等指标来衡量。

常见的评价指标包括

- 准确率 (accuracy)：指模型预测正确的样本数量占总样本数的比例
- 召回率 (recall)：指模型正确预测为正例的样本数量占真实正例样本数的比例
- 精确率 (precision)：指模型预测为正例的样本中真正为正例的样本数量占预测为正例的样本数量的比例
- F1得分 (F1-score)：综合考虑了准确率和召回率，是它们的调和平均数
- AUC (Area Under the Curve)：是ROC曲线下的面积，用于衡量二分类模型的性能。



	预测值0	预测值1
真实值0	TN	FP
真实值1	FN	TP

$$TPR = \frac{\text{正样本预测正确量}}{\text{正样本总量}} = \frac{TP}{TP+FN}$$

$$FPR = 1 - TNR = \frac{\text{负样本预测错误量}}{\text{负样本总量}} = \frac{FP}{FP+TN}$$

46、阐述一下什么是模型的表达能力 (Expressive Power) ? (1星)

表达能力 (expressive power) 是指一个模型或算法能够表示不同种类的函数或数据分布的能力。在机器学习中，表达能力通常用来评估一个模型的适应能力和泛化能力。

一个具有较高表达能力的模型或算法能够表示多种复杂的函数或数据分布，因此可以更好地适应不同的数据。例如，深度神经网络拥有很高的表达能力，可以用来表示各种形式的函数，如线性函数、非线性函数、时序数据、图像数据等。相比之下，线性回归模型的表达能力较低，只能用来表示简单的线性关系。

然而，过高的表达能力也可能导致过拟合的问题，即模型过于复杂，过度拟合训练数据，而无法泛化到未知数据。因此，在选择模型时，需要在表达能力和泛化能力之间进行权衡。通常，可以通过正则化、集成学习等方法来平衡模型的表达能力和泛化能力。

总的来说，表达能力是衡量模型能力的一个重要指标，它对于选择合适的模型和算法，以及提高模型性能具有重要意义。

47、什么是机器学习中的局部最优与全局最优？(1星)

在机器学习中，模型的优化过程是通过不断调整模型参数来最小化一个损失函数（或最大化一个收益函数）。局部最优与全局最优是这个优化过程中的两个重要概念。

局部最优指的是在参数空间的某个小区域内，函数取得极小值，但这个值可能不是全局最小值。全局最优指的是在参数空间中，函数取得的最小值。通常，在实际问题中，函数空间可能非常复杂，局部最优点遍布其中，而全局最优点则可能非常难以找到。

在机器学习中，我们通常通过梯度下降法等优化算法来求解模型参数。这些算法通常能够找到函数的一个局部最优点，但无法保证找到全局最优解。针对这个问题，我们可以通过增加模型复杂度、随机化初始化或使用不同的优化算法等方式来提高算法的全局最优解的概率，但这仍然不可以完全解决全局最优问题。

需要注意的是，对于某些问题，局部最优解已经足够好，而且找到全局最优解的计算成本可能非常高，这时候可以放弃追求全局最优，而是采用一些启发式方法或近似算法来找到局部最优解。

48、什么是梯度下降算法？(2星)

梯度下降 (Gradient Descent) 算法是一种用于求解机器学习和深度学习中优化问题的迭代算法。其目标是通过不断更新参数来最小化目标函数（通常为损失函数或代价函数）的值。在许多问题中，如线性回归、逻辑回归和神经网络等，我们希望找到一组参数，使得损失函数的值尽可能小，从而使模型的预测更准确。

梯度下降的基本思想是利用目标函数的梯度（即偏导数）来确定参数更新的方向。在每次迭代中，参数沿着梯度的负方向更新，因为梯度的负方向是目标函数值下降最快的方向。梯度下降的一般步骤如下：

1. 初始化参数：为模型参数选择一个初始值。
2. 计算梯度：计算目标函数关于参数的梯度（偏导数）。
3. 更新参数：按照学习率和梯度的负方向更新参数。
4. 重复步骤2和3，直到满足停止条件（如梯度接近零、达到最大迭代次数或损失函数的变化小于某个阈值）。

梯度下降有多种变体，如批量梯度下降 (Batch Gradient Descent)、随机梯度下降 (Stochastic Gradient Descent) 和小批量梯度下降 (Mini-Batch Gradient Descent)。这些变体主要区别在于每次更新参数时所用的数据量不同。批量梯度下降使用整个训练集，随机梯度下降使用单个训练样本，而小批量梯度下降使用一小部分训练样本。根据问题和数据集的特点，可以选择合适的梯度下降变体以达到更好的优化效果。

49、梯度下降算法中的 α 又被称为什么？有什么要求？（2星）

梯度下降法中的阿尔法 (α)，也被称为学习率 (learning rate)，是一个非负实数，用于控制参数更新的幅度。在每次迭代过程中，我们按照学习率和梯度的负方向更新参数，更新的幅度是梯度乘以学习率。学习率对算法的收敛速度和最终结果有很大的影响。选择合适的学习率非常重要。

学习率必须是一个正数，因为我们需要沿着梯度的负方向更新参数。如果学习率是负数，参数更新将朝着梯度的正方向进行，导致损失函数值增加，无法收敛。如果学习率过大，参数更新可能会跳过最优值，导致损失函数在最小值附近震荡，甚至发散。**过大的学习率使得算法难以收敛到全局最小值或局部最小值。**如果学习率过小，参数更新的幅度非常小，**收敛速度会变得非常慢**，可能需要大量迭代才能达到最优解。这会导致算法在有限的迭代次数内无法收敛到一个较好的解。

在某些情况下，可以使用自适应学习率方法，如 Adam、AdaGrad 和 RMSProp 等。这些方法会根据梯度的变化动态调整学习率，以改善算法的收敛速度和稳定性。

50、什么是特征标准化 (Standardization) ?有什么用处? (2星)

特征标准化 (Feature Standardization) 是一种预处理技术，用于对数据集中的特征进行缩放，**使其具有统一的尺度**。特征标准化通常在机器学习和统计建模中使用，因为不同尺度的特征可能导致算法的性能受到影响，或者使得某些特征对模型的影响过大。

特征标准化的常见方法是将特征转换为均值为0，标准差为1的分布。这种方法也称为 Z-score 标准化：

$$z = (x - \mu) / \sigma$$

经过特征标准化处理后，每个特征的**均值为0，标准差为1**。

特征标准化的**优点**包括：

提高模型性能：某些机器学习算法，如支持向量机 (SVM) 、K-近邻 (KNN) 和梯度下降，对特征尺度敏感。特征标准化有助于提高这些算法的性能。

加速收敛：对于基于梯度的优化算法，特征标准化可以加速收敛过程，因为梯度下降的步长在各个特征维度上更加一致。

提高可解释性：特征标准化后，可以更容易地比较不同特征对模型的相对重要性。

请注意，特征标准化并不适用于所有情况。例如，在处理具有**明确物理意义和量纲的数据**时，特征标准化可能会导致信息丢失。在这种情况下，可以考虑使用其他缩放方法，如最小最大缩放 (Min-Max Scaling) 。

51、特征标准化与特征归一化的区别是什么？(2星)

特征标准化和特征归一化都是机器学习中用于数据预处理的方法，它们本质上是对数据进行缩放。区别主要在于缩放的方式和目的。

特征标准化，也称为Z-score标准化，是将数据缩放到均值为0，标准差为1的范围内。这种方法适用于数据分布近似为高斯分布的情况下，能够消除数据的缩放影响，使得数据更具有可比性。

特征归一化，也称为最小-最大规范化，是将数据缩放到指定的最小值与最大值之间。这种方法适用于数据的分布不是高斯分布的情况下，能够把数据映射到0和1之间，便于数据处理。

综上所述，特征标准化和特征归一化各有适用的场景，实际使用时需要根据数据的分布情况和特定问题来选择具体的方法。

但是！由于历史原因和工程问题，容易让人困惑的一点是指代混淆，Normalization有时会指代min-max normalization，有时会指代Standardization，有时会指代Scaling to unit length。需要具体分析时要注意到底是什么方法。

52、梯度下降法与闭式解 (closed-form solution) 相比，各有什么优劣？(2星)

梯度下降法是一种迭代优化算法，可以用于求解许多不可解析求解的问题，如逻辑回归、神经网络等。梯度下降法通过不断地调整模型参数来最小化损失函数，直到达到收敛条件为止。优点是可以处理大规模数据和高维特征，且在损失函数复杂或不可解析的情况下，仍能找到近似最优解。缺点是需要选择合适的学习率和迭代次数，且有可能陷入局部最优解。

闭式解 (Closed Form Solution)，又称解析解，通过求解数学公式来获得最优解，例如最小二乘线性回归的解析解。优点是能够直接得到最优解，不需要迭代优化，计算速度较快。缺点是只适用于部分问题，有些问题没有解析解或者解析解复杂，因此需要使用数值计算方法来近似求解。

因此，选择梯度下降法还是Closed Form Solution取决于具体的问题和算法。如果问题可以使用Closed Form Solution求解，并且计算复杂度不高，可以选择使用解析解法；如果问题无法使用解析解法求解，或者计算复杂度很高，可以选择使用迭代优化算法，如梯度下降法等。

53、推导最小二乘线性回归的解析解。（5星）

最小二乘法线性回归的损失函数如下：

$$\begin{aligned} J(\theta) &= \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})(h_\theta(x^{(i)}) - y^{(i)}) \\ &= \frac{1}{2} (X\theta - y)^T (X\theta - y) \\ &= \frac{1}{2} ((X\theta)^T - y^T)(X\theta - y) \\ &= \frac{1}{2} (\theta^T X^T - y^T)(X\theta - y) \\ &= \frac{1}{2} (\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y) \end{aligned}$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\begin{aligned} J'(\theta) &= \left[\frac{1}{2} (\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y) \right]' \\ &= \frac{1}{2} [(\theta^T X^T X\theta)' - (\theta^T X^T y)' - (y^T X\theta)' + (y^T y)'] \\ J'(\theta) &= \frac{1}{2} [(\theta^T X^T X\theta)' - (\theta^T X^T y)' - (y^T X\theta)' + (y^T y)'] \\ &= \frac{1}{2} [2X^T X\theta - X^T y - (y^T X)^T] \\ &= \frac{1}{2} [2X^T X\theta - 2X^T y] \\ &= X^T X\theta - X^T y = 0 \end{aligned}$$

$$\theta = (X^T X)^{-1} X^T y$$

54、什么是线性决策边界与非线性决策边界？(1星)

在机器学习中，决策边界（Decision Boundary）是将不同类别的数据样本区分开来的边界，可以看做是模型预测结果的分界线。

线性决策边界是指能够将数据点分成两个不同的类别的一条直线（在二维空间中）或超平面（在高维空间中）。通常用于线性分类问题，比如逻辑回归和支持向量机。

非线性决策边界则是指不能用一条直线或超平面将数据点分为两个类别，而需要更高阶的函数来进行分类。通常用于非线性分类问题，比如决策树、神经网络、KNN等算法。

例如，在一个二维平面上，将以红色和蓝色表示的数据点分成两类，如果使用一条直线能够将它们分开，则是线性决策边界；但如果需要使用一条曲线来将它们分开，则属于非线性决策边界。

55、如何使用最大似然估计 (Maximum Likelihood Estimation) 来推导损失函数？(5星)

以线性回归为例，我们假设随机变量Y与x之间是非确定性的（这里的x是可以控制或精确观察的变量，故我们不把x看成是随机变量，只把它看成是普通变量），即对于x的每一个取值，Y还会受到一个随机因素的影响，从而具有自己的分布。

用 $F(Y|x)$ 表示当x取到一个确定值时，所对应的Y的分布函数。Y的期望 $E(Y)$ 随x取值而定，是一个关于x的函数，我们将其记为 $\mu(x)$ ，称为Y关于x的回归函数。假设 $\mu(x)$ 是线性函数： $\mu(x) = ax + b$ ，并且误差服从正态分布：

要估计参数a, b，可用极大似然估计 $Y = ax + b + \epsilon, \epsilon \sim N(0, \sigma^2)$ yn)，写出似然函数有

$$L = \prod_{i=1}^n f(y_i|x_i; a, b) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - ax_i - b)^2}{2\sigma^2}}$$

要最大化L，其中 σ 又是无关变量 $L = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - ax_i - b)^2}$

$$\sum_{i=1}^n (y_i - ax_i - b)^2$$

这即是最小二乘法的解释，也是平方误差损失函数的推导得出的最小化式子。误差服从正态分布时利用极大似然估计推导得出的最小化式子。

56、向量化 (Vectorization) 的概念与目的是什么？(1星)

向量化 (Vectorization) 是一种将运算操作应用于整个数组或矩阵的技术，用于**加速数值计算和优化代码的执行效率**。它是一种使用向量或矩阵操作代替显式循环的技术，可以在减少代码量的同时提高代码的执行速度。

在计算机科学中，通常使用的是标量 (Scalar) 运算，即对单个数值进行操作，这样的运算通常需要使用循环来遍历整个数组或矩阵进行计算。而**向量化操作是对整个数组或矩阵进行运算**，这种运算操作是并行的，可以利用CPU和GPU的SIMD指令集 (Single Instruction Multiple Data) 来实现高效的并行计算，从而大大提高运算速度和效率。

例如，使用向量化操作可以将两个向量相加，而无需使用循环遍历每个元素进行相加，从而显著提高运算速度。在机器学习中，向量化操作可以大幅提高神经网络的训练速度，同时减少代码的实现难度。

常见的利用向量化处理数据的库包括NumPy、SciPy和PyTorch、TensorFlow等，它们提供了许多高效的向量化操作函数和工具，使得使用向量化操作可以更加方便和高效。

57、阐述一下机器学习中常见的距离度量指标。 (3星)

Distance Metrics（距离度量）是机器学习中常用的一种度量方法，用于衡量数据点之间的距离或相似性。在机器学习中，数据通常表示为一个向量或矩阵，而Distance Metrics可以根据这些向量或矩阵之间的距离或相似性来衡量它们之间的差异。常用的距离度量包括：

1. 欧几里得距离 (Euclidean Distance)：也称为欧式距离，是最常见的一种距离度量方法，表示为两个数据点之间的直线距离
2. 曼哈顿距离 (Manhattan Distance)：也称为城市街区距离，表示为两个数据点在坐标轴上的横纵坐标差的绝对值之和
3. 余弦相似度 (Cosine Similarity)：是一种用于衡量向量之间相似性的度量方法，它表示为两个向量之间的夹角余弦值
4. 闵可夫斯基距离 (Minkowski Distance)：是欧几里得距离和曼哈顿距离的推广，表示为 p 次方根号下两个数据点之间的各个维度的差的 p 次方之和
5. 切比雪夫距离 (Chebyshev Distance)：表示为两个数据点在各个维度上差值的最大值。

58、什么是特征工程？它包括哪些方面？（2星）

特征工程指的是在机器学习或数据挖掘任务中，通过对原始数据进行处理和转换，构建更具有表达能力和预测能力的特征集合的过程。特征工程是机器学习中非常重要的一步，因为它直接决定了模型的性能和泛化能力。

特征工程的目标是寻找那些能够更好地描述数据的特征，或者是从原始数据中提取那些对预测任务有意义的特征。特征工程包括以下几个方面：

- 1. 数据预处理：**数据预处理是特征工程的第一步。因为原始数据很难直接作为模型的输入，需要进行清洗、去重、填充缺失值等操作。
- 2. 特征抽取：**当原始数据为文本、图片、音频等非结构化数据时，需要借助特定的算法将其转换为结构化数据，例如文本抽取中的词频统计、TF-IDF等。
- 3. 特征变换：**在对数据进行建模过程中，可能需要对某些特征进行转换，以使得其更加符合模型的建模假设，例如对数变换、归一化处理等。
- 4. 特征选择：**当原始数据中存在大量的特征时，需要进行特征选择。即从所有可能的特征中选择最佳的特征子集，以避免维数灾难并提高模型的泛化能力。
- 5. 特征构建：**有时候我们可以从原始数据中构造新的特征，例如时间特征的提取、多项式特征的构建等。构建新特征可以帮助我们更好地描述数据。

59、什么是奥卡姆剃刀原理？（1星）

在机器学习中，Occam's Razor通常被称为奥卡姆剃刀（Ockham's Razor），是一种常见的偏好偏差（Preference Bias）。奥卡姆剃刀原则是指，如果有多种可能的解释或假设，那么应该选择最简单的解释或假设，即应该选择最少的假设来解释已知的现象。简单来说，就是选择最简单、最经济的解释。用一句话来概括，叫“**如无必要，勿增实体**”。

在机器学习中，奥卡姆剃刀原则可以用来指导特征选择和模型选择。通常来说，我们倾向于选择最简单的特征和模型，以避免过拟合和提高泛化能力。因此，在选择特征和模型时，我们应该考虑其复杂度和解释能力，并尽可能选择最简单、最可解释的特征和模型。

奥卡姆剃刀原则并不意味着我们应该简单地忽略那些复杂的解释或模型，而是在多种可能的解释或模型中，应该首先考虑最简单的假设。如果最简单的假设不能解释数据或者模型的预测效果不佳，我们可以考虑更复杂的假设或模型。

60、什么是信息熵 (Information Entropy) ? (3星)

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

其中， $H(X)$ 表示随机变量 X 的信息熵 (entropy)， $P(x)$ 表示 X 取某个值的概率， $\log_2(P(x))$ 表示该事件的信息量。求和遍历 X 的所有可能取值。它可以用于衡量某一随机变量的信息量，也可以用于衡量信息源的平均信息量。

信息熵越大，表示随机变量的不确定性越高，信息量也就越大。当随机变量完全确定时，其信息熵为0；当随机变量完全随机时，其信息熵达到最大值。

香农的信息度量被广泛应用于信息论、统计学、机器学习等领域。在机器学习中，信息熵常被用作衡量决策树节点的不确定性，以便选择最优的节点划分。

61、什么是条件熵 (Conditional Entropy) ? (2星)

条件熵是指在已知某一随机变量的取值的条件下，另一个随机变量的不确定性。

$$\begin{aligned} H(X|Y) &= \sum_y p(y)H(X|Y=y) \\ &= -\sum_y p(y)\sum_x p(x|y)\log(p(x|y)) \\ &= -\sum_y \sum_x p(x,y)\log(p(x|y)) \\ &= -\sum_{x,y} p(x,y)\log(p(x|y)) \end{aligned}$$

条件熵 $H(X|Y)$ 描述了在给定 Y 的取值的条件下， X 的不确定性，也就是在 Y 已知的情况下，对 X 进行预测的难度。在决策树算法中，选择分裂特征时，通常会优先选择能够最大限度地降低子集的条件熵的特征。

62、Reduced-Error Pruning是什么？(2星)

Reduced-Error Pruning是一种用于决策树的后剪枝算法。它是一种贪心算法，它尝试通过删除子树来降低决策树的误差率。

具体来说，Reduced-Error Pruning 的步骤如下：

1. 从训练集中随机选择一部分数据作为验证集。
2. 构建完整的决策树，对于每一个非叶子节点，计算剪枝后的验证集误差率。
3. 从底层开始逐个考虑节点进行剪枝，直到验证集误差率不再降低为止。
4. 在剪枝的过程中，如果某个节点的剪枝能够降低验证集误差率，那么这个节点及其子树就会被删除。删除后，该节点会变成叶节点，类别标记为该节点下样本数量最多的类别。
- 5.

Reduced-Error Pruning 通过随机选择验证集来避免过度拟合，并通过反复剪枝来选择最优决策树。

63、阐述一下集成学习 (Ensemble Learning) 的原理与常见方法。 (5星)

集成学习 (Ensemble Learning) 是一种机器学习策略，它结合了多个基础学习器 (Base Learners) 的预测结果，以获得比单个学习器更好、更稳定的性能。集成学习方法通常可以提高预测准确性和泛化能力，降低过拟合的风险。

集成学习的主要思想是：通过组合多个具有差异性和独立性的基础学习器，可以减少每个学习器的单独误差，从而提高整体性能。集成学习有许多实现方法，其中最常用的三种是：Bagging、Boosting 和 Stacking。

Bagging (Bootstrap Aggregating)：一种基于自助采样 (Bootstrap Sampling) 的集成方法。它通过多次有放回地从原始数据集中抽取样本，构建不同的子数据集，并用这些子数据集训练多个基础学习器。最后，对这些学习器的预测结果进行投票（分类任务）或求均值（回归任务）以得到最终结果。Bagging 可以降低模型的方差，提高稳定性。随机森林 (Random Forest) 就是基于 Bagging 方法的典型应用。

Boosting：一种迭代的集成方法，它通过对训练样本的权重进行调整，使得每轮训练关注上一轮分类错误的样本。Boosting 生成的基础学习器是有序的，每个学习器都试图修正前一个学习器的错误。最后，通过加权投票（分类任务）或加权求和（回归任务）得到最终结果。Boosting 主要降低偏差，提高模型的准确性。梯度提升树 (Gradient Boosted Trees, GBT) 和 AdaBoost (Adaptive Boosting) 都是基于 Boosting 方法的典型应用。

Stacking (Stacked Generalization)：一种多层集成方法，它首先训练多个基础学习器，然后使用这些学习器的预测结果作为输入，训练一个元学习器 (Meta Learner)，用于输出最终的预测结果。Stacking 可以组合不同类型的基础学习器，提高模型的泛化能力。

64、集成学习中多数投票方法有什么优势和劣势？(3星)

多数投票法是一种集成学习方法，它将多个模型的预测结果进行综合，从而得到一个更好的最终预测。

优点：

- 减少过拟合：多数投票法结合了多个模型的预测结果，有助于减少单个模型可能出现的过拟合现象。
- 提高准确性：多个模型的组合往往比单个模型具有更高的预测准确性，因为不同模型可能在不同方面有所长处，结合起来可以发挥互补作用。
- 健壮性：多数投票法具有较好的抗噪声性能，因为多个模型的预测结果可以抵消个别模型的误差。
- 泛化能力：多数投票法可以提高模型的泛化能力，使模型在面对新的数据时，具有更好的预测表现。

缺点：

- 计算复杂度：多数投票法需要训练和预测多个模型，因此计算复杂度较高，可能导致较长的训练和预测时间。
- 模型依赖：多数投票法的表现取决于所选模型的质量。如果所选模型性能较差，集成学习可能无法显著提高预测准确性。
- 难以解释：多数投票法涉及多个模型的组合，可能导致最终模型难以解释，对于需要解释性的场景不适用。
- 权重选择：在多数投票法中，给不同模型分配权重可能会影响最终结果。然而，权重的选择往往需要大量实验和调整，可能导致优化过程复杂。

65、什么是自助采样 (Bootstrap Sampling) ? 如何进行? (3星)

Bootstrap Sampling (自助采样) 是一种统计学中的抽样方法，主要用于估计统计量的分布、置信区间等。自助采样是**有放回地从原始数据集中随机抽取样本，构建一个新的数据集**。自助采样的步骤如下：

1. 给定一个包含 N 个样本的原始数据集 D 。
2. 有放回地随机抽取一个样本，将其加入新数据集 D' 。
3. 重复第二步 N 次，使得新数据集 D' 也包含 N 个样本。
4. 由于是有放回抽样，某些原始样本可能在新数据集 D' 中出现多次，而某些可能完全未被抽到。

在机器学习和统计学中，自助采样常用于以下场景：

集成学习：如 Bagging (自助聚合) 方法，它通过生成多个基于自助采样的子数据集来训练多个基础学习器，然后聚合这些学习器的预测结果以提高模型性能。

估计统计量的分布：通过对原始数据集进行多次自助采样，可以生成多个不同的子数据集。然后，根据这些子数据集计算所关心的统计量（如均值、中位数等），从而估计这些统计量的分布、置信区间等。

模型验证：例如自助法 (Bootstrap Method) 和交叉验证 (Cross Validation)，可以通过对原始数据集进行自助采样来评估模型的性能，包括准确性、稳定性等。

66、什么是AdaBoost？(2星)

AdaBoost (Adaptive Boosting) 是一种集成学习方法，主要用于分类任务，也可用于回归任务。它通过迭代地训练一系列弱学习器（通常为简单的分类器，如决策树桩），然后将这些弱学习器结合起来以形成一个强分类器。在每次迭代中，AdaBoost 会根据之前学习器的错误调整样本权重，使得后续学习器更关注于那些难以分类的样本。AdaBoost 的工作原理如下：

初始化训练数据集的权重，使得每个样本具有相等的权重。

在每次迭代中：

1. 使用当前的样本权重训练一个弱学习器。
2. 计算这个弱学习器在训练数据集上的错误率。
3. 更新弱学习器的权重，使得在后续迭代中，表现较差的弱学习器具有较小的权重。
4. 更新样本权重，增加被错误分类样本的权重，降低正确分类样本的权重。

将所有弱学习器按照它们的权重进行加权投票（分类任务）或加权平均（回归任务）得到最终预测结果。

67、Adaboost的优缺点分别是什么？(2星)

优点：

1. 高准确性：通过结合多个弱学习器的预测结果，AdaBoost 可以显著提高模型的准确性。
2. 自适应：AdaBoost 能自动调整样本权重，使得后续的弱学习器更关注于难以分类的样本，从而提高整体模型的性能。
3. 不易过拟合：尽管 AdaBoost 会关注难以分类的样本，但由于弱学习器的加权组合，它通常不容易出现过拟合现象。
4. 可以与多种弱学习器结合：尽管 AdaBoost 通常与决策树桩（一层决策树）结合使用，但它也可以与其他类型的弱学习器结合，如线性分类器或神经网络。
- 5.

缺点：

1. 对噪声敏感：AdaBoost 会关注那些难以分类的样本，这使得它对噪声和异常值较为敏感，可能导致性能下降。
2. 训练速度：由于 AdaBoost 是一种迭代方法，无法进行并行训练，因此训练过程可能相对较慢。

68、集成学习在什么情况下可以提高准确率？(3星)

集成学习是一种通过组合多个分类器来提高预测精度的技术。集成学习可以在以下情况下提高准确率：

1. **数据不足或存在噪声**: 集成学习通过将多个分类器的预测结果进行投票或加权平均，可以减少数据不足和噪声的影响，从而提高准确率。
2. **分类器的偏差和方差较大**: 单个分类器可能会出现偏差或方差过大的问题，如果使用多个分类器，则可以通过加权平均或投票来减小这些偏差和方差，从而获得更准确的预测结果。
3. **模型复杂度较高**: 复杂的模型可能会导致过拟合的问题，而集成学习可以通过降低单个模型的复杂度来避免过拟合。
4. **数据分布不均匀**: 某些分类器可能更适合特定类型的数据，而集成学习可以通过组合不同类型的分类器来解决数据分布不均匀的问题。

需要注意的是，在使用集成学习时，需要确保单个分类器之间的**差异性**，否则集成学习可能无法提高准确率，甚至会降低准确率。

69、什么是K折交叉验证 (K-Fold Cross-Validation) ? (2星)

K折交叉验证是一种模型评估方法。

它将原始数据集分成K个子集（通常是将数据集随机分成K个子集）。然后，使用其中的K-1个子集进行训练模型，使用剩余的1个子集来测试模型性能。

这个过程重复K次，每个子集都会参与一次测试，而剩下的 (K-1) 个子集都用来训练数据，并计算出每次测试的准确率（或其他性能指标）的平均值，作为该模型的最终性能指标。

使用K-fold交叉验证可以缓解数据集过小，样本分布不均等等问题，增加模型的稳定性，也可以有效地利用数据，避免过拟合等问题。

70、如何处理样本类别不均衡问题？(4星)

类别不均衡问题是指在分类任务中，不同类别的样本数量存在显著差异的情况。在这种情况下，分类模型可能会偏向于多数类，导致对少数类的识别性能较差。以下是一些处理类别不均衡问题的方法：

1. **重采样 (Resampling)**：通过添加或删除样本来平衡数据集中的类别。对于少数类别，可以通过增加随机生成或复制已有样本的方式来增加样本数量，或者删除多数类别样本来减少样本数量。
2. **过采样 (Oversampling)**：对少数类样本进行重复采样，以增加其在数据集中的比例。可以使用简单的随机过采样或更复杂的方法，如 SMOTE (Synthetic Minority Over-sampling Technique)。
3. **欠采样 (Undersampling)**：从多数类样本中随机删除一部分样本，以减小多数类与少数类之间的数量差距。注意，这种方法可能会丢失一些重要信息。
4. **权重调整 (Cost-sensitive learning)**：为不同类别的样本分配不同的权重，使得分类器在训练过程中对少数类样本给予更高的关注。对于一些机器学习算法，如支持向量机 (SVM) 和逻辑回归，可以直接在损失函数中添加类别权重。
5. **集成方法**：如 Bagging 和 Boosting，可以在一定程度上缓解类别不平衡问题。例如，可以对训练集进行多次采样生成多个子集，并在每个子集上训练一个基学习器，然后通过投票或平均的方式汇总基学习器的预测结果。
6. **调整分类阈值**：对于二分类问题，可以尝试调整分类阈值以提高少数类的识别性能。例如，在逻辑回归中，通常将预测概率大于 0.5 的样本划分为正类，可以将阈值降低以提高少数类的召回率。
7. **使用评估指标**：在类别不平衡问题中，准确率可能不是一个合适的评估指标。可以使用其他指标，如精确度、召回率、F1 分数和 AUC-ROC 曲线等，以更准确地衡量模型在少数类上的性能。
8. **数据合成**：使用生成模型（如生成对抗网络 GAN）合成新的少数类样本，以增加数据集中少数类的数量。

71、简要阐述一下数据预处理的简要流程和常见方法。（3星）

数据预处理常见的流程方法有：

1. **数据清洗**: 删除或修正缺失数据, 删除重复值, 修正错误值等。
2. **特征选择**: 从数据集中选择最相关的特征用于训练模型, 避免不相关或冗余特征对模型的干扰和过拟合。
3. **数据转换**: 采取对数据进行转换的方式使其更加适合于模型训练, 常见的包括特征缩放、特征转换、特征离散化、归一化等。
4. **数据集划分**: 将数据集分为训练集、验证集和测试集, 一般是对数据集进行随机划分或者时间序列划分, 以评估模型的性能与泛化能力。

除了上述步骤外, 还有一些常见的数据预处理技术, 如**数据平滑**、**数据聚合**、**特征构建**、**特征交叉**等。

1. 数据平滑是通过某种平均方法来平滑原始数据, 去除异常值的影响, 以减少误差。
2. 数据聚合是将原始数据中的某些特征进行分组, 然后将它们合并成一个新的变量, 从而降低数据复杂度, 提高模型训练效率。
3. 特征构建是利用原有的特征来构建新的特征, 以提高模型性能。比如从时间序列数据中提取季节性、趋势性、周期性等特征, 构建新的时间序列特征。
4. 特征交叉是指将两个或多个特征进行组合, 构造出新的特征, 以提高模型的拟合能力和泛化能力。

72、什么是随机梯度下降法？其中的随机 (Stochastic) 代表什么？(3星)

随机梯度下降法 (Stochastic Gradient Descent, SGD) 是一种常用的优化算法，广泛应用于深度学习等机器学习模型中。与传统的梯度下降法每次迭代使用全部训练数据进行模型参数的更新不同，SGD在每次迭代中仅使用部分训练数据进行更新，因此具有更快的计算速度和更低的内存消耗。

1. 使用整个训练集的优化算法被称为批量 (batch) 或确定性 (deterministic) 梯度算法，因为会在一个大批量中同时处理所有样本。
2. 每次只使用单个样本的优化算法有时被称为随机 (stochastic) 或者在线 (online) 算法。

注意，由于历史原因和工程问题，大多数用于深度学习的算法介于两者之间，使用一个以上而不是全部的训练样本。现在通常将他们简单地称为随机 (stochastic) 方法，其实更合适的称呼是“小批量随机梯度下降”。

在SGD算法中，每次更新的数据集称为一个mini-batch，与全局的梯度下降不同的是，SGD在每个mini-batch上计算损失函数的梯度，并沿该方向更新模型参数，由于每个mini-batch的样本和噪声不同，因此可以避免陷入局部最优解，并能够更快地逼近全局最优解。

73、随机梯度下降有什么优缺点？有什么改进方法？(2星)

使用随机梯度下降法的一些**优点**:

- **计算效率:** 因为每次迭代只使用一个训练样本来计算梯度，SGD 的计算效率较高，尤其在大数据集的情况下。这意味着 SGD 可以更快地进行模型参数更新，从而加速收敛过程。
- **噪声抵抗:** 由于 SGD 的随机性，它能更好地抵抗噪声样本对梯度的影响。在某些情况下，这种随机性甚至可以帮助算法跳出局部最优解，从而找到全局最优解。
- **在线学习:** SGD 可以适应在线学习 (Online Learning) 场景，即在训练过程中不断接收新的数据样本并更新模型。这对于那些需要实时更新模型的应用场景非常有用，如金融市场预测、推荐系统等。
- **易于实现:** SGD 相比其他优化算法更容易实现，因为它只需要在每次迭代时计算一个样本的梯度。

使用随机梯度下降法的一些**缺点**:

- **可能会收敛到局部最优解:** 由于每一次迭代只使用一小部分数据，因此可能会存在偏差，导致收敛到局部最优解而非全局最优解。
- **可能会出现震荡:** 由于每一次迭代只使用一小部分数据，因此每次计算的结果可能会有一些噪声，这些噪声可能会导致模型参数的震荡。
- **学习率难以设置:** 由于每次迭代的数据不同，因此学习率难以设置得到一个最优的值，会出现学习率过大或过小的问题。

为了克服这些缺点，提出了一些**SGD的变种和改进方法**，如带动量的SGD (Momentum)、Adagrad、RMSProp和Adam等。这些方法在某些方面优于原始的SGD，如收敛速度、稳定性和自适应学习率等。

74、推导逻辑回归梯度下降法的表达式。 (4星)

记 y_i 为样本 x_i 的真实标签, $\hat{y}_i = \frac{1}{1+e^{-(w^T x_i + b)}}$ 为预测其标签为1的概率, 逻辑回归损失函数是

交叉熵损失函数: $J = -\sum_{i=1}^N (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$ 。

由 $\hat{y}_i = P(y_i = 1|x_i)$, 显然: $1 - \hat{y}_i = P(y_i = 0|x_i)$ 。将两个式子结合起来:

$P(y_i|x_i) = \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}$, 这是一个伯努利分布。对于N个样本, 由独立同分布假设可

知: $P(Y|X) = \prod_{i=1}^N \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}$ 。由最大似然法, 要估计 \hat{y}_i 中的参数 w 和 b , 只需要求 $P(Y|X)$ 的最大值, 即:

$$\begin{aligned} w, b &= \arg \max_{w, b} P(Y|X) \\ &= \arg \max_{w, b} \log P(Y|X) \\ &= \arg \max_{w, b} \log \prod_{i=1}^N \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i} \\ &= \arg \max_{w, b} \sum_{i=1}^N \log \{\hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}\} \\ &= \arg \max_{w, b} \sum_{i=1}^N \{\log \hat{y}_i^{y_i} + \log (1 - \hat{y}_i)^{1-y_i}\} \\ &= \arg \max_{w, b} \sum_{i=1}^N \{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\} \\ &= \arg \min_{w, b} - \sum_{i=1}^N \{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\} \end{aligned}$$

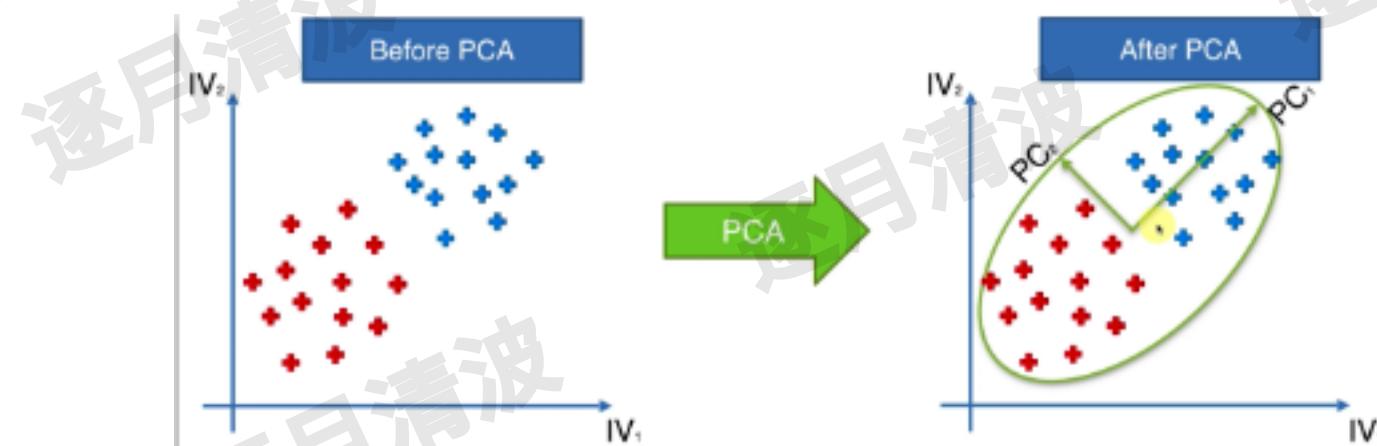
$$\begin{aligned} \nabla_w &= \sum_{i=1}^N \left\{ y_i \frac{\hat{y}_i(1 - \hat{y}_i)}{\hat{y}_i} x_i - (1 - y_i) \frac{\hat{y}_i(1 - \hat{y}_i)}{1 - \hat{y}_i} x_i \right\} \\ &= \sum_{i=1}^N \{y_i(1 - \hat{y}_i)x_i - (1 - y_i)\hat{y}_i x_i\} \\ &= \sum_{i=1}^N \{(y_i - \hat{y}_i)x_i\} \end{aligned}$$

$$w_t = w_{t-1} - \alpha \nabla_w$$

75、什么是主成分分析 (PCA) ? 有哪些应用场景? (2星)

主成分分析 (PCA) 是一种常见的数据降维技术，它寻找潜在的变量来解释数据中的方差，并选择最能表达原始数据的变量，从而实现数据简化和消除冗余的目的。具体来说，PCA将原始的高维数据投影到低维空间中，保留最多的数据方差，因此，PCA不仅能够减少数据的维度，还能够保留数据的主要特征。以下是PCA的一些应用场景：

1. 降维：PCA是降维技术中最常用的方法之一，可以用于压缩数据、可视化数据和减少噪声等。
2. 特征提取：PCA能够从原始数据中提取出最具有代表性的特征，这些特征可用于建立更加精确的模型。
3. 数据预处理：在一些机器学习的任务中，高维的数据可能导致过拟合的问题，因此需要将数据集进行降维处理，PCA可以在保留数据潜在变量的情况下，降低数据集的维度，减少过拟合的风险。
4. 数据压缩：大量的数据会占用大量的存储空间，影响存储和计算效率，PCA可以在不影响数据质量的前提下，将数据用更少的维度进行存储和传输。



76、什么是支持向量机中的核技巧 (Kernel Trick) ? (3星)

支持向量机 (Support Vector Machine, 简称SVM) 是一种常用的分类算法，其核心思想是将高维的数据通过某个超平面分隔开来。在实际应用中，我们往往需要处理非线性的数据，此时就需要引入核技巧 (Kernel Trick)。

核技巧实际上是一种在高维空间中对数据进行处理的方法，它可以将低维的非线性数据在高维空间中转换成线性可分的形式。这个过程使用一种叫做核函数 (Kernel Function) 的数学函数，它可以计算数据的相似度，从而在高维空间中找到相应的决策边界。

如果支持向量机的求解只用到内积运算，而在低维输入空间又存在某个函数 $K(x, x')$ ，它恰好等于在高维空间中这个内积，即 $K(x, x') = \langle \phi(x) \cdot \phi(x') \rangle$ 。那么支持向量机就不用计算复杂的非线性变换，而由这个函数 $K(x, x')$ 直接得到非线性变换的内积，使大大简化了计算。这样的函数 $K(x, x')$ 称为核函数。

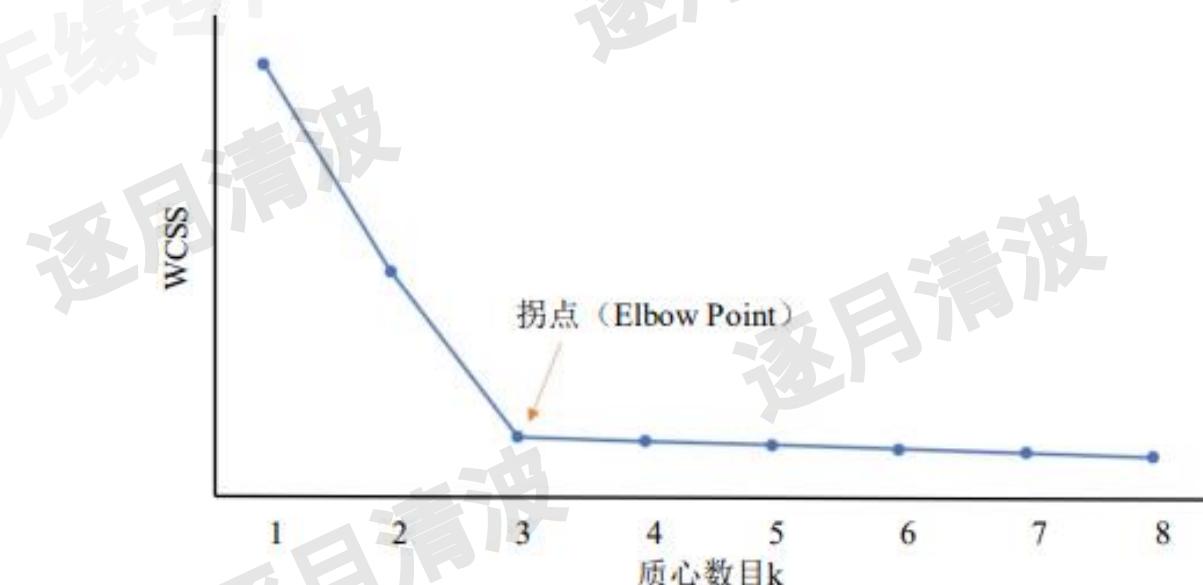
常用的核函数包括线性核函数、多项式核函数、径向基函数 (Radial Basis Function, 简称RBF) 核函数等。

77、在K-均值聚类算法中如何确定合适的K值？(3星)

确定K值的过程运用了手肘法则 (Elbow Rule)：

手肘法则的基本思路是通过不同的K值，对应不同的聚类结果，计算每个聚类结果中的数据点到对应聚类中心的距离之和（也称为误差平方和，Within-Cluster Sum of Square , WCSS）。然后将每个聚类结果的误差平方和相加，作为该K值的误差平方和。随着K值增加，**每个聚类结果的误差平方和会逐渐减小，但是减小的速度会变得越来越慢**。因此，我们可以画出K值和误差平方和之间的关系图，然后找到曲线中的“拐点”，即误差平方和下降趋势明显放缓的点，该点的K值就是我们需要选择的聚类数目。

因此，手肘法则的名称来源于曲线图中的“手肘”点，图像形状像一条手臂弯曲。它是一种简单但有效的方法，用于确定最佳的K值。



78、解释一下生成式与判别式学习的基本概念。 (2星)

生成式学习 (Generative Learning) 是指通过学习输入数据在特定条件下的联合分布概率来生成新的数据，也就是说，生成模型可以模拟样本在整个概率分布上的分布情况，可以用来从训练数据中学习样本 (特征) 之间的依赖关系，进而生成新的数据。典型的生成式模型包括朴素贝叶斯、隐马尔可夫模型 (HMM) 和深度信念网络 (DBN) 等。

判别式学习 (Discriminative Learning) 则是指直接建立输入到输出的映射模型，即模型的主要目标是建立输入数据和输出数据之间的对应关系。判别模型的目标是最大限度地减少分类误差或预测误差等，因为这些模型不需要考虑样本 (特征) 之间的概率分布，所以它们通常也具有更高的学习精度。典型的判别式模型包括逻辑回归、支持向量机 (SVM)、决策树、神经网络等。

生成式学习可以生成新的数据样本，对数据的概率分布有更强的假设；在小样本情况下表现较好，但假设可能过于强烈，导致模型无法很好地捕捉数据的复杂性；预测性能可能不如判别式模型。

判别式学习的预测性能通常优于生成式模型；对数据的概率分布假设较弱，能够捕捉数据的复杂性，但不能生成新的数据样本，需要较大的样本量才能达到较好的性能。

79、什么是感知机？它的基本原理是什么？(2星)

感知机（Perceptron）是一种简单的人工神经网络模型，它是监督学习中的一种线性分类器，用于对输入数据进行二元分类。感知机的基本结构是一个单层神经网络，包含一个输入层和一个输出节点。

感知机的工作原理如下：

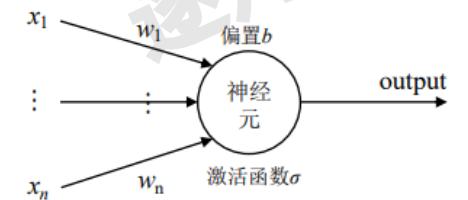
接收输入：感知机接收一个包含n个特征的输入向量 x ，每个特征都有一个权重 w 与之关联。

加权求和：感知机将输入向量 x 与权重向量 w 进行点积（内积），然后加上偏置项 b ，即线性组合： $z = w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n + b$ 。

激活函数：将线性组合 z 输入到一个激活函数（通常是阶跃函数或符号函数）中，生成输出。如果激活函数的输出大于0，感知机将输入分类为正类（通常表示为+1）；否则，将其分类为负类（通常表示为-1）。

感知机的训练过程包括调整权重 w 和偏置 b ，以使分类错误最小化。这通常通过随机梯度下降（Stochastic Gradient Descent）或类似的优化算法实现。当数据线性可分时，感知机可以找到一个超平面将两个类别完全分开。然而，当数据不是线性可分时，感知机将无法收敛到一个解决方案。

尽管感知机具有局限性（如只能处理线性可分问题），但它在历史上具有重要意义，因为它是神经网络和深度学习领域发展的基石。感知机的概念已经发展成更复杂的多层神经网络结构，如多层感知机（MLP）和深度学习模型，这些结构能够处理更复杂的非线性问题。



80、为什么神经网络中需要激活函数 (Activation Function) ? (2星)

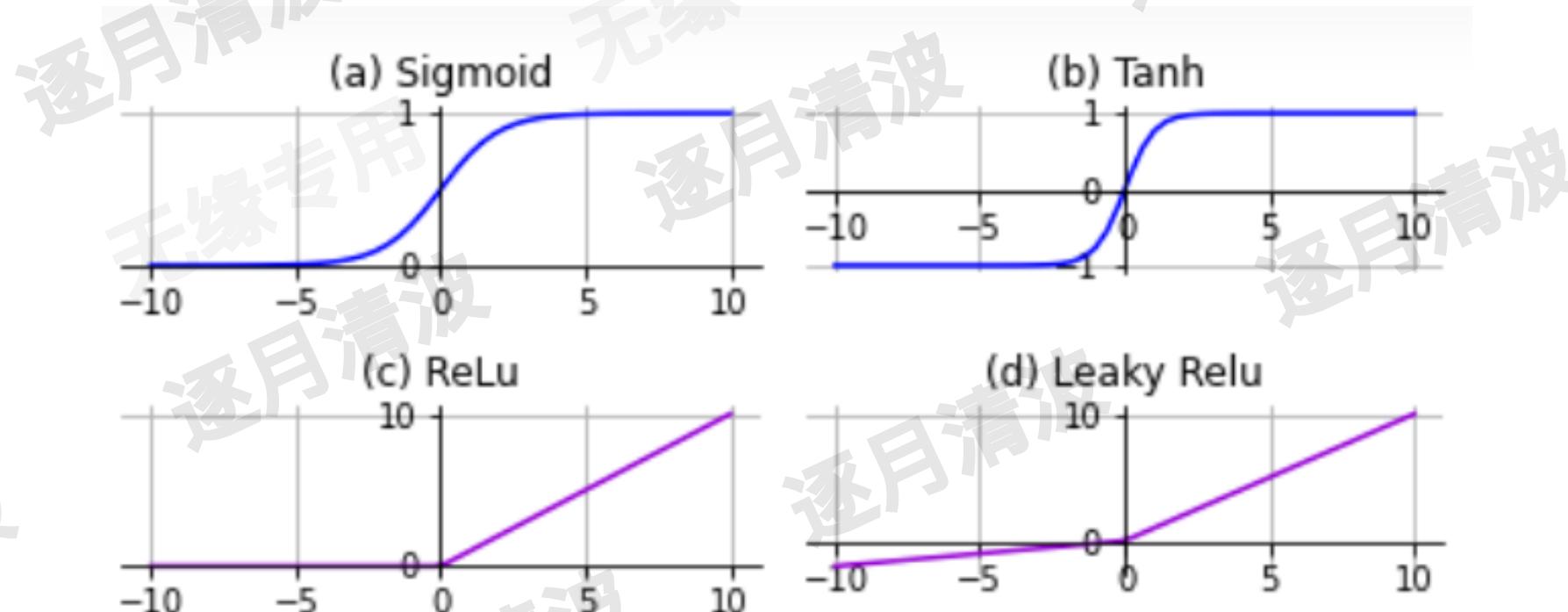
需要激活函数的原因主要有以下几点：

- 1. 引入非线性：**激活函数可以为神经网络引入非线性因素，使得网络能够拟合和解决复杂的非线性问题。如果不使用激活函数，神经网络将仅仅是一个线性模型，无法处理复杂的现实世界问题。
- 2. 模拟生物神经元：**激活函数的使用使得人工神经元在某种程度上能够模拟生物神经元的激发过程。当神经元输入信号超过某个阈值时，神经元才会激活并向下游传递信号。
- 3. 控制输出范围：**某些激活函数（如 Sigmoid 和 tanh）可以将神经元的输出限制在一个特定范围内。这有助于在训练过程中控制输出值，提高网络的稳定性。
- 4. 提高模型表达能力：**通过引入激活函数，神经网络可以表示更复杂数学函数，从而提高模型的表达能力。这使得神经网络可以在各种任务中表现出色，如图像识别、自然语言处理和强化学习等。

81、例举一些常见的激活函数与它们的图像特点。（3星）

常见的激活函数有：

- a. **Sigmoid**: 将输入的数值映射到0到1之间
- b. **Tanh**: 将输入的数值映射到-1到1之间
- c. **ReLU**: 在输入小于0时输出0，在输入大于0时输出等于输入的值
- d. **Leaky ReLU**: 在输入小于0时输出一个小小的非零值，在输入大于0时输出等于输入的值

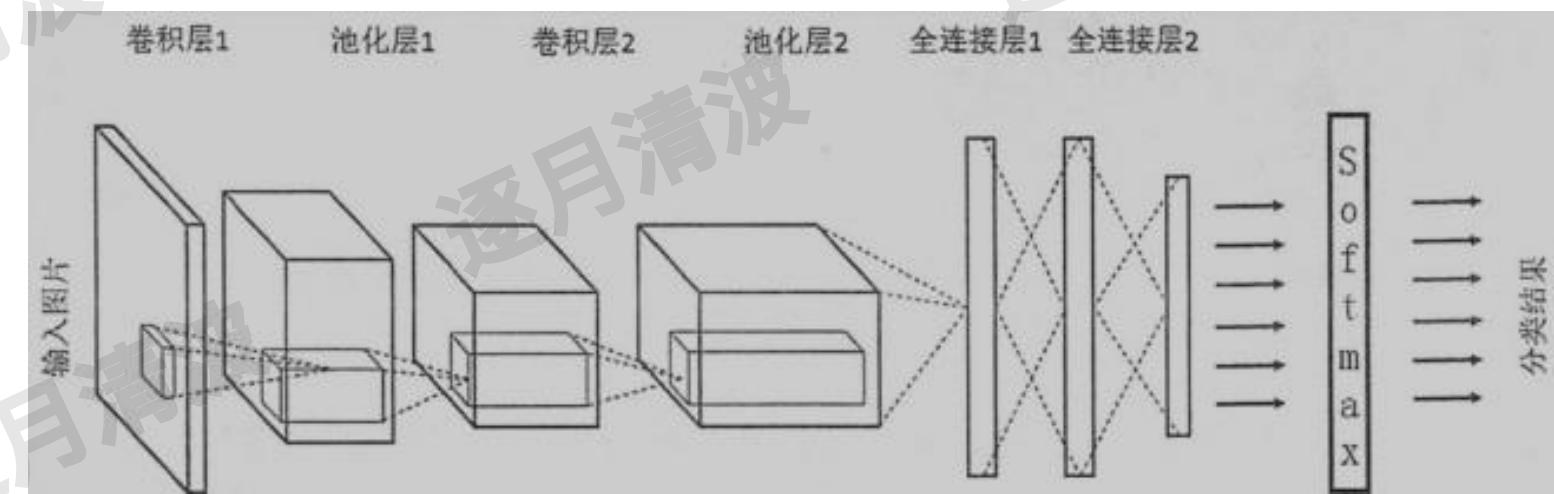


82、什么是卷积神经网络 (CNN) ? 解释一下它的组成与原理。 (3星)

卷积神经网络 (Convolutional Neural Networks, CNN) 是一种特殊类型的神经网络，主要用于处理具有网格结构的数据。CNN的主要特点是局部连接、权值共享和平移不变性。这些特点使得卷积神经网络能够高效地学习从局部特征到全局特征的层次结构，并且具有较强的泛化能力。卷积神经网络的基本结构包括：

- 卷积层 (Convolutional Layer)：通过卷积操作，在输入图像的局部区域上提取特征。卷积层使用一组可学习的过滤器（或卷积核），将这些过滤器在输入数据上滑动以计算卷积。
- 池化层 (Pooling Layer)：通过在空间上降采样输入特征图，实现对特征的空间不变性和减少计算量。常用的池化操作有最大池化 (Max Pooling) 和平均池化 (Average Pooling)。
- 全连接层 (Fully Connected Layer)：将卷积和池化层提取到的特征进行整合，用于分类或回归任务。在多分类问题中，全连接层通常与Softmax 激活函数结合使用，以输出每个类别的概率。

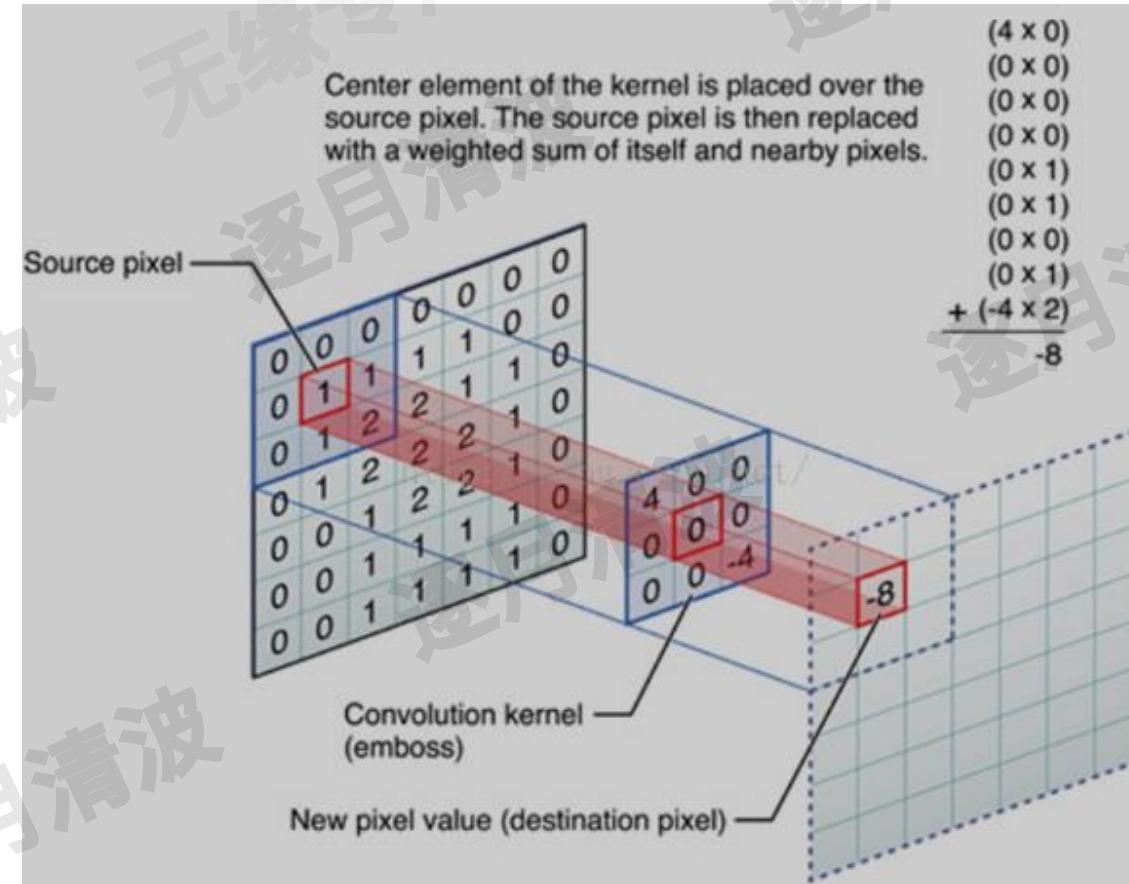
一个典型的卷积神经网络由多个卷积层、激活层、池化层堆叠而成，最后接一个或多个全连接层。卷积神经网络的深度和宽度可以根据任务的复杂性进行调整。



83、解释一下二维卷积操作的原理与计算过程。 (2星)

二维卷积 (2D Convolution) 是一种在二维数据 (如图像) 上进行的数学运算，它可以用于提取图像的局部特征、滤波、边缘检测等任务。在卷积神经网络 (Convolutional Neural Networks, CNN) 中，二维卷积是一个核心操作，通过将卷积核 (或过滤器) 在输入图像上滑动，提取出空间局部特征。

1. 将一个卷积核 (通常是一个较小的矩阵) 放在输入图像的左上角。
2. 对应元素相乘：将卷积核中的每个元素与输入图像中对应位置的元素相乘。
3. 对乘积求和：将上一步得到的乘积相加，得到一个标量值，就是输出特征图 (Feature Map) 中的一个元素。
4. 滑动卷积核：将卷积核向右 (或向下) 滑动一个步长 (stride)，然后重复步骤2和3。当卷积核滑动到输入图像的右边缘时，将其移回左边缘并向下滑动一个步长。持续进行这个过程，直到卷积核覆盖整个输入图像。



84、二维卷积中，输出的图片尺寸由哪些因素决定？(3星)

在二维卷积中，输出图像的尺寸由下列因素决定：

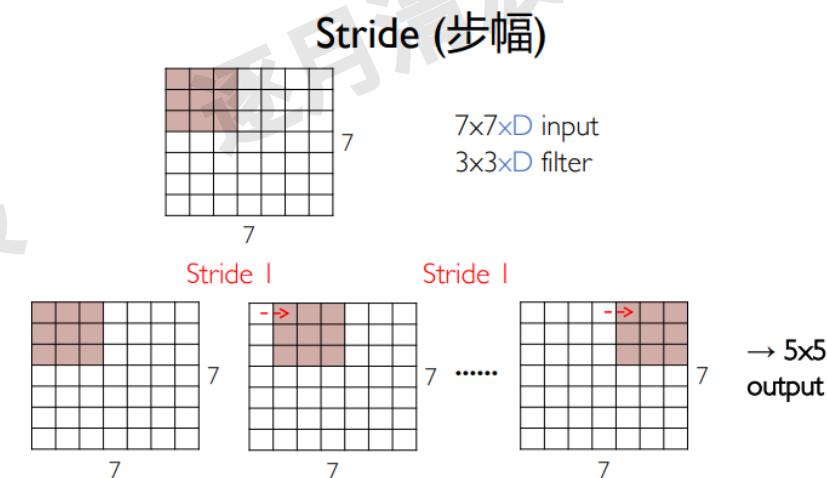
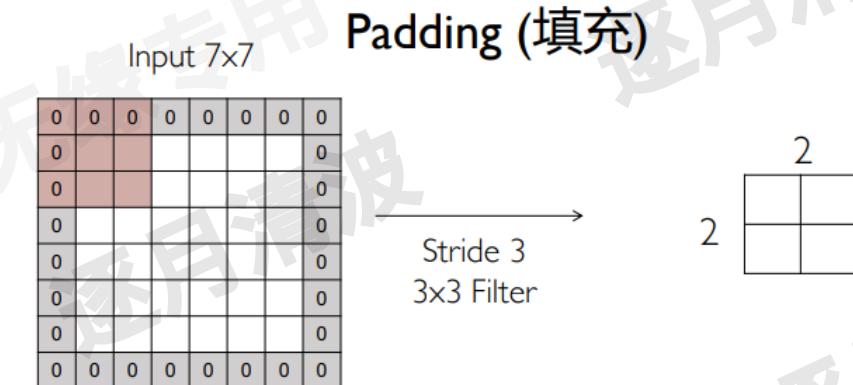
- **原始图像大小**：指输入到卷积层的二维图像的尺寸。通常用宽 (W) 和高 (H) 表示，原始图像大小为 $H \times W$ 。
- **滤波器大小**：也称为卷积核大小，指在卷积过程中滑动窗口的尺寸。通常用 F 表示，滤波器大小为 $F \times F$ 。
- **填充**：指在原始图像周围添加一定数量的零像素，以保持图像在经过卷积后的尺寸。填充数量用 P 表示。
- **步长**：指在卷积过程中滑动窗口每次移动的距离。步长大用 S 表示。

这些参数之间的关系决定了卷积层输出的尺寸。假设原始图像大小为 $H \times W$ ，滤波器大小为 $F \times F$ ，填充大小为 P，步长为 S，则卷积层输出的尺寸为：

$$\text{输出高度 (Output Height)} = (H - F + 2P) / S + 1$$

$$\text{输出宽度 (Output Width)} = (W - F + 2P) / S + 1$$

这里的输出高度和输出宽度应向下取整。



85、CNN中的卷积操作有哪些作用和优点？(3星)

1. **特征提取作用**: 卷积操作可以从输入图像中提取出特征信息，比如线条、轮廓、纹理等。这些特征对于识别、分类等任务十分重要。
2. **参数共享**: 卷积操作中，同一卷积核 (filter) 在不同位置上的操作采用的是同样的权值参数，这就大大减少了需要训练和存储的参数数量，同时也使得模型更加具有通用性。
3. **稀疏连接**: 卷积核的大小通常远小于输入层的大小，因此每个神经元只与输入层的一小部分神经元相连，这使得模型具有局部连接和权值共享的特点，从而使得模型更加有效和高效。
4. **平移不变性**: 卷积操作中的权值共享和稀疏连接使得模型具有平移不变性，即输入图像发生平移时，卷积核中的权重参数也会相应地进行平移，而输出结果不会发生改变。这为模型对于输入图像的位置变化具有一定的鲁棒性。
5. **降维作用**: 卷积操作中的池化 (pooling) 操作可以在不改变特征数量的情况下减小特征的空间尺度，从而降低模型的复杂度和计算量，同时也可以防止过拟合。

86、什么是CNN中的特征图堆叠 (Stacking Multiple Feature Maps) ? (2星)

在卷积神经网络中，堆叠多个特征图 (Stacking Multiple Feature Maps) 是指将 **多个特征图组合在一起，形成一个更丰富、更具表现力的特征表示**。在 CNN 中，特征图是卷积层输出的二维矩阵，其中每个元素表示某种局部特征在输入数据中的响应。

堆叠多个特征图的方法是在卷积层 **使用多个卷积核** (也称为过滤器)。每个卷积核负责从输入数据中提取一种特定的局部特征。例如，在图像处理任务中，不同的卷积核可能分别负责检测边缘、角点、纹理等。将这些卷积核得到的特征图堆叠在一起，可以形成一个更全面、更丰富的特征表示。

在实际应用中，堆叠多个特征图通常通过**增加卷积层的输出通道数 (output channels)** 来实现。例如，对于一个具有 64 个输出通道的卷积层，其实就是将 64 个特征图堆叠在一起。随着网络深度的增加，输出通道数通常会逐渐增大，以捕捉更多的特征组合和抽象层次。堆叠多个特征图的优点包括：

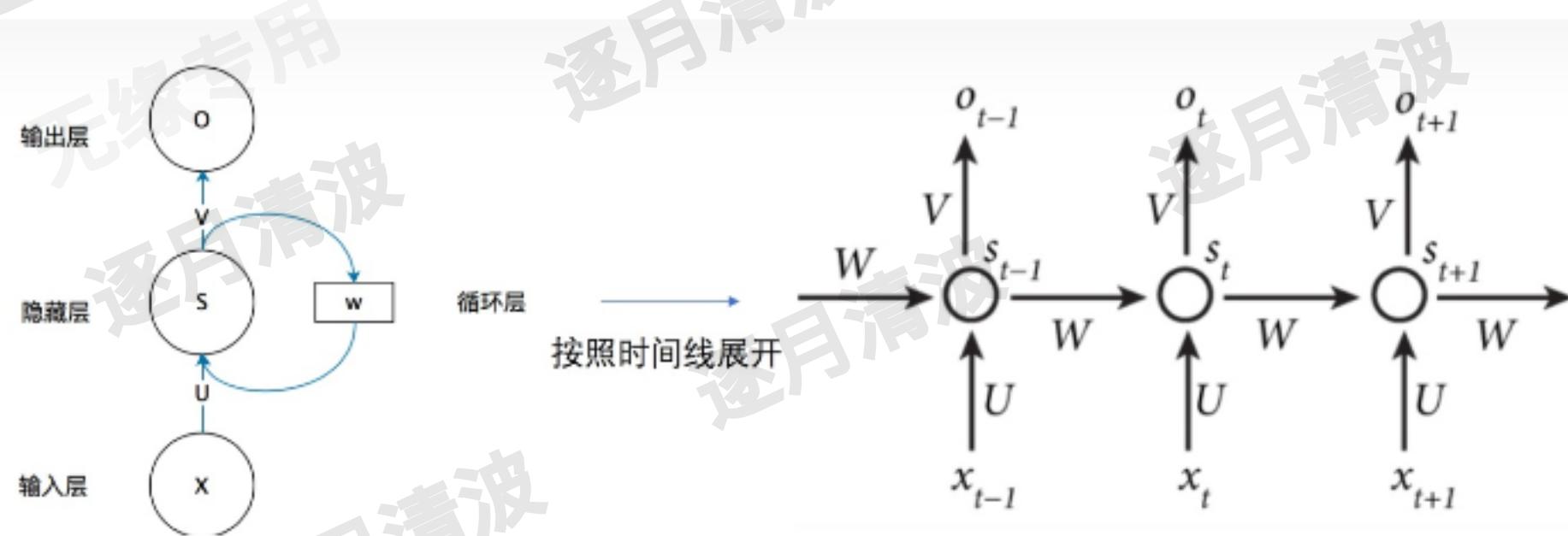
- **更丰富的特征表示**: 通过组合多种特征，可以表示更复杂的视觉模式。
- **更强的泛化能力**: 不同的特征图可以学习到不同的特征，使得网络对于不同类型的输入具有更好的适应性。
- **更高的模型容量**: 增加特征图的数量可以增加模型的容量，有助于学习更多的特征和提高模型性能。

然而，**过多的特征图也可能导致过拟合**，因此需要权衡模型复杂度与泛化能力。

87、什么是循环神经网络 (RNN) ?解释一下它的组成与原理。 (3星)

循环神经网络 (Recurrent Neural Networks, RNNs) 主要用于处理具有顺序性的数据，如时间序列数据、自然语言文本等。与传统的前馈神经网络 (Feedforward Neural Networks) 不同，RNNs具有循环连接，允许信息在网络中的不同时间步之间传递。这种循环结构使得RNN具有处理长度可变的序列数据和捕捉长期依赖关系的能力。

RNN的基本结构包括一个循环单元，可以将其视为网络的一个“记忆”。在每个时间步，**循环单元接收当前时间步的输入数据和上一个时间步的隐藏状态**。基于这两个输入，循环单元计算并输出新的隐藏状态，该状态可以传递到下一个时间步，如此反复。通过这种方式，RNN可以在不同时间步之间共享信息。



88、解释一下反向传播的原理与步骤。（4星）

反向传播（Backpropagation）的主要目标是最小化损失函数（Loss Function），即计算模型预测值与真实值之间的差距。它通过计算损失函数相对于模型参数的梯度来更新网络中的权重和偏置。反向传播算法是一种基于链式法则的高效梯度计算方法。反向传播算法的基本步骤如下：

- **前向传播**：将输入数据传入神经网络，经过每一层的加权求和、激活函数处理，直到最后一层输出产生预测值。
- **计算损失**：根据模型的预测值和实际标签计算损失函数的值。损失函数的目标是衡量模型的预测性能，常见的损失函数包括均方误差（Mean Squared Error, MSE）、交叉熵损失（Cross-Entropy Loss）等。
- **反向传播梯度**：从输出层开始，沿着网络结构逐层向前计算每层的梯度。根据链式法则，将损失函数相对于每个权重和偏置的梯度分解为各部分的导数。这一过程实际上是将梯度从输出层向输入层传播。
- **更新权重和偏置**：根据计算得到的梯度和学习率更新网络中的权重和偏置。权重和偏置的更新遵循梯度下降算法，即沿着梯度的负方向更新参数以最小化损失函数。
-
- 通过迭代这些步骤，神经网络的权重和偏置将逐渐调整，模型的预测性能将逐步提高。在训练过程中，可以使用不同的优化算法（如随机梯度下降、小批量随机梯度下降或其他优化器）来更新权重和偏置。

89、什么是梯度消失和梯度爆炸？为什么会出现这种情况？（4星）

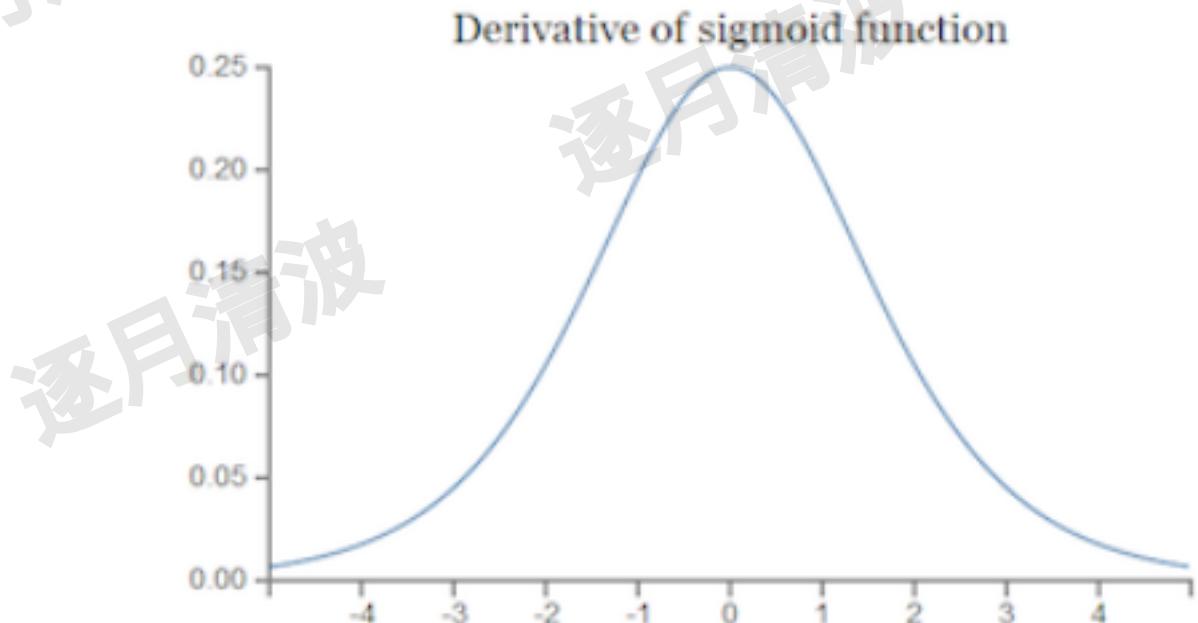
梯度消失和梯度爆炸是深度神经网络中常见的问题。这些问题源于在反向传播过程中，神经网络的每一层权重梯度都会被计算并传递给前一层，如果在传递过程中，权重梯度太小或太大，就会出现梯度消失或梯度爆炸。

梯度消失指的是，在反向传播中，由于激活函数的导数不易于计算，当神经网络的层数变得较深时，每一层的权重梯度可能会变得越来越小，梯度消失也就发生了。这会导致神经网络无法有效学习到深层次的特征，影响网络的训练效果。

梯度爆炸指的是，在反向传播中，由于某些原因，例如权重初始化过大或使用了大量带有较高权重的层，每一层的权重梯度可能会变得越来越大，梯度爆炸也就出现了。这会导致神经网络的权重变得异常大，模型不收敛，训练失败。



$$\begin{aligned}\frac{\partial C}{\partial b_1} &= \frac{\partial C}{\partial y_4} \frac{\partial y_4}{\partial z_4} \frac{\partial z_4}{\partial x_4} \frac{\partial x_4}{\partial z_3} \frac{\partial z_3}{\partial x_3} \frac{\partial x_3}{\partial z_2} \frac{\partial z_2}{\partial x_2} \frac{\partial x_2}{\partial z_1} \frac{\partial z_1}{\partial b_1} \\ &= \frac{\partial C}{\partial y_4} \sigma'(z_4) w_4 \sigma'(z_3) w_3 \sigma'(z_2) w_2 \sigma'(z_1)\end{aligned}$$

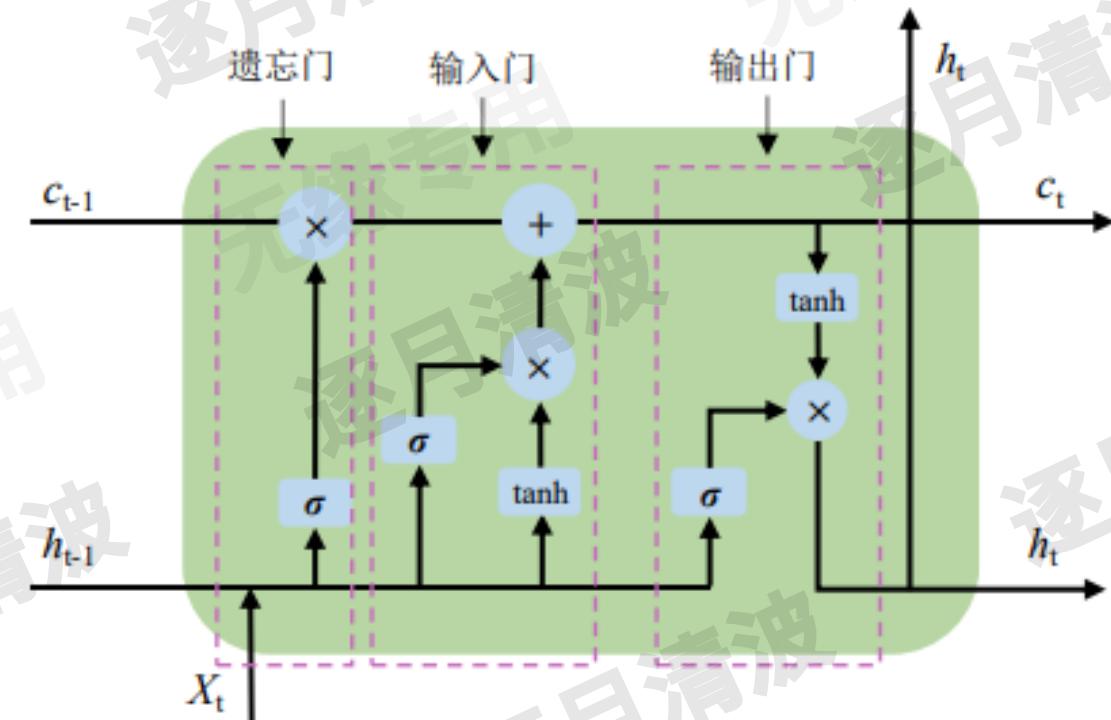


90、什么是长短期记忆网络 (LSTM) ? 解释一下它的组成和原理。 (4星)

长短时记忆网络 (Long Short-Term Memory, LSTM) 是一种特殊的循环神经网络 (RNN)，用于解决传统RNN在处理长距离依赖关系时遇到的梯度消失和梯度爆炸问题。LSTM通过引入所谓的“细胞状态” (Cell State) 来保持和传递长期信息。细胞状态是LSTM中的核心概念，它可以看作是LSTM的内部“记忆”。细胞状态在整个序列中沿水平方向 (时间步) 传递，允许信息在时间步之间传播。与普通RNN不同，LSTM使用一系列门控机制来有选择地添加或删除信息，以便有效地维护和更新细胞状态。LSTM的这些门控结构包括：

- **遗忘门 (Forget Gate)**：决定从细胞状态中丢弃哪些信息。遗忘门通过一个sigmoid激活函数计算一个0到1之间的值，表示保留或丢弃细胞状态中的某部分信息。
- **输入门 (Input Gate)**：确定哪些新信息将被添加到细胞状态中。输入门由两部分组成：一个sigmoid激活函数计算添加信息的比例，以及一个tanh激活函数计算新信息的候选值。
- **输出门 (Output Gate)**：确定当前时间步的隐藏状态 (也称为输出状态)。输出门使用一个sigmoid激活函数计算保留细胞状态信息的比例，然后将细胞状态通过一个tanh激活函数进行缩放，最后将两者相乘得到隐藏状态。

LSTM通过这些门控结构来有选择地保留和更新细胞状态，使其能够有效地处理长距离依赖关系。



$$O_t^f = \sigma(W_f \bullet [x_t, h_{t-1}] + b_f)$$

$$O_t^i = \sigma(W_{i1} \bullet [x_t, h_{t-1}] + b_{i1}) \odot \tanh(W_{i2} \bullet [x_t, h_{t-1}] + b_{i2})$$

$$c_t = O_t^f \odot c_{t-1} + O_t^i$$

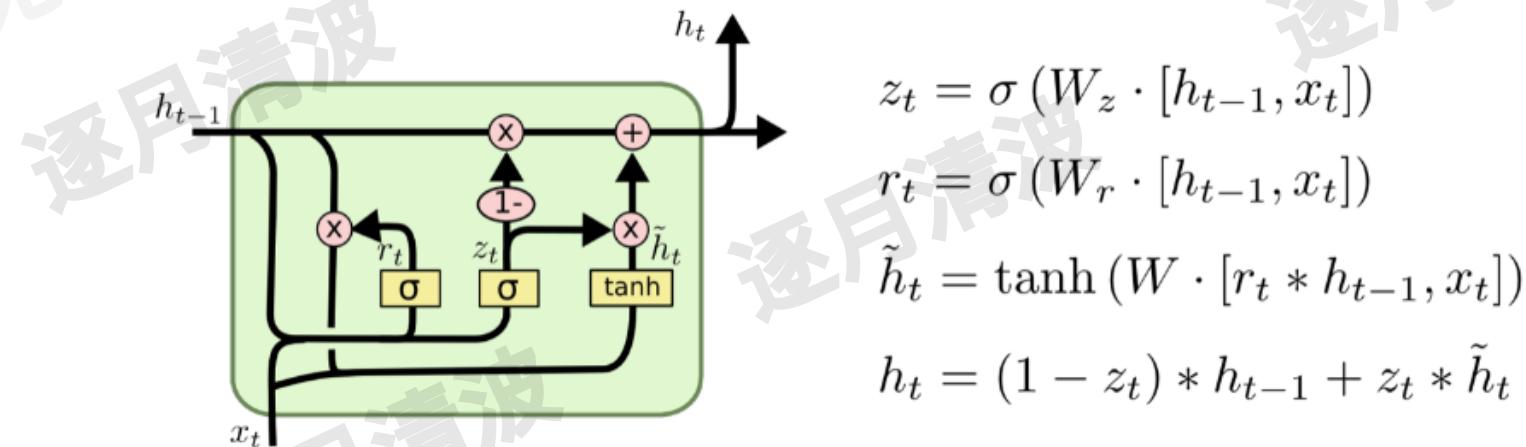
$$h_t = \sigma(W_o \bullet [x_t, h_{t-1}] + b_o) \odot \tanh(c_t)$$

91、什么是门控循环单元 (GRU)？解释一下它的组成和原理。 (2星)

Gated Recurrent Unit (GRU) 是一种循环神经网络 (RNN) 的变体，主要用于处理序列数据，例如时间序列分析、语音识别、自然语言处理等领域。GRU 是一种简化的长短时记忆网络 (LSTM)，在某些任务中表现相当。GRU 引入了门控机制 (gating mechanism)，这种机制允许网络根据当前输入和过去的隐藏状态来决定如何更新隐藏状态。GRU 有是更新门 (update gate) 和重置门 (reset gate)：

- **更新门**: 决定如何融合之前的隐藏状态与当前输入，以生成新的隐藏状态。它帮助确定过去的信息在多大程度上保留到当前时间步。
- **重置门**: 决定在计算当前隐藏状态时，如何将之前的隐藏状态与当前输入相结合。它帮助模型确定是否需要丢弃过去的隐藏状态。

这些门控机制使得 GRU 能够更好地捕捉长距离依赖关系，缓解传统 RNN 中常见的梯度消失和梯度爆炸问题。尽管 GRU 比 LSTM 更简单，参数更少，但在许多任务中，它们的性能相差无几。因此，GRU 在需要较少计算资源和内存的场景中，可能是一个更好的选择。

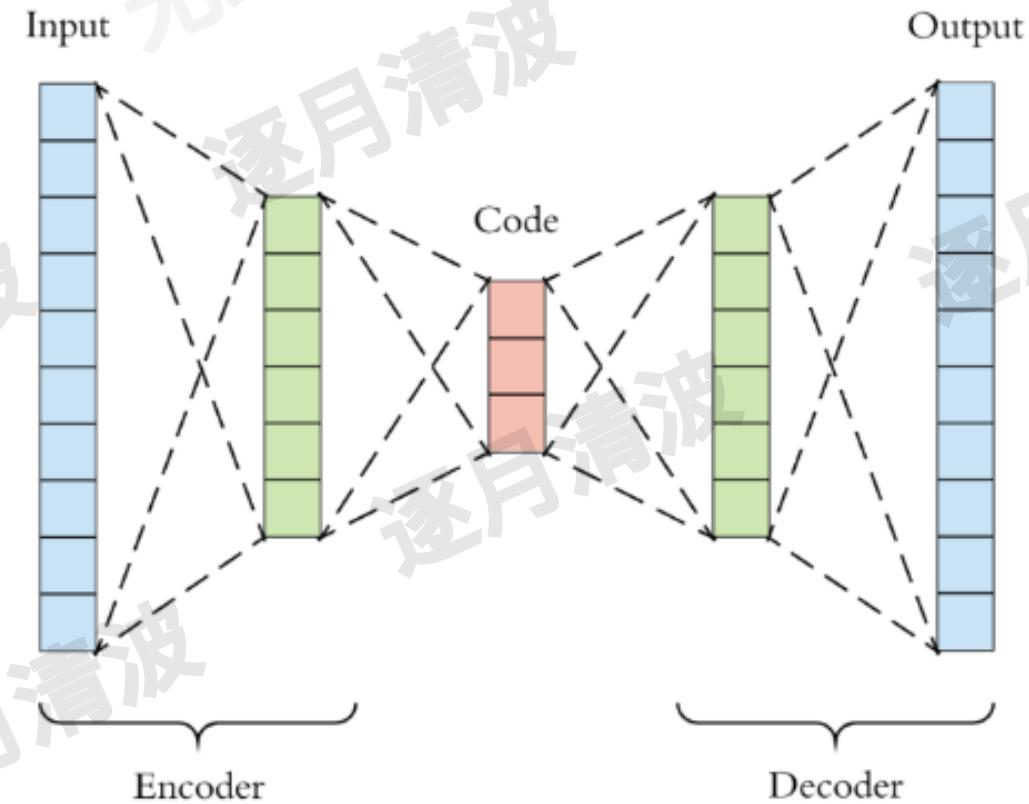


92、什么是自编码器（AE）？解释一下它的组成和原理。（3星）

自编码器（Auto-Encoder）是一种无监督学习模型，其目的是在不需要标记数据的情况下，将输入数据压缩成隐藏表示，并在需要时从该隐藏表示中重构原始输入数据。自编码器由两部分组成：编码器和解码器。编码器将输入数据转换为隐藏表示，解码器则将隐藏表示转换回原始输入数据。

自编码器的训练是通过最小化重构误差来完成的，也就是输入数据与其重构后的结果之间的差异。最常见的自编码器是基于多层感知器（MLP）的前馈神经网络，其中编码器和解码器都是由多个完全连接的层组成。编码器的层数通常小于解码器的层数，这使得自编码器能够对输入数据进行压缩。

自编码器的原理是基于神经网络和数据压缩的思想，其在训练过程中学习将数据压缩成低维表示，并在需要时将其解压缩回原始数据。自编码器被广泛应用于数据压缩、图像去噪、数据降维和特征提取等领域。



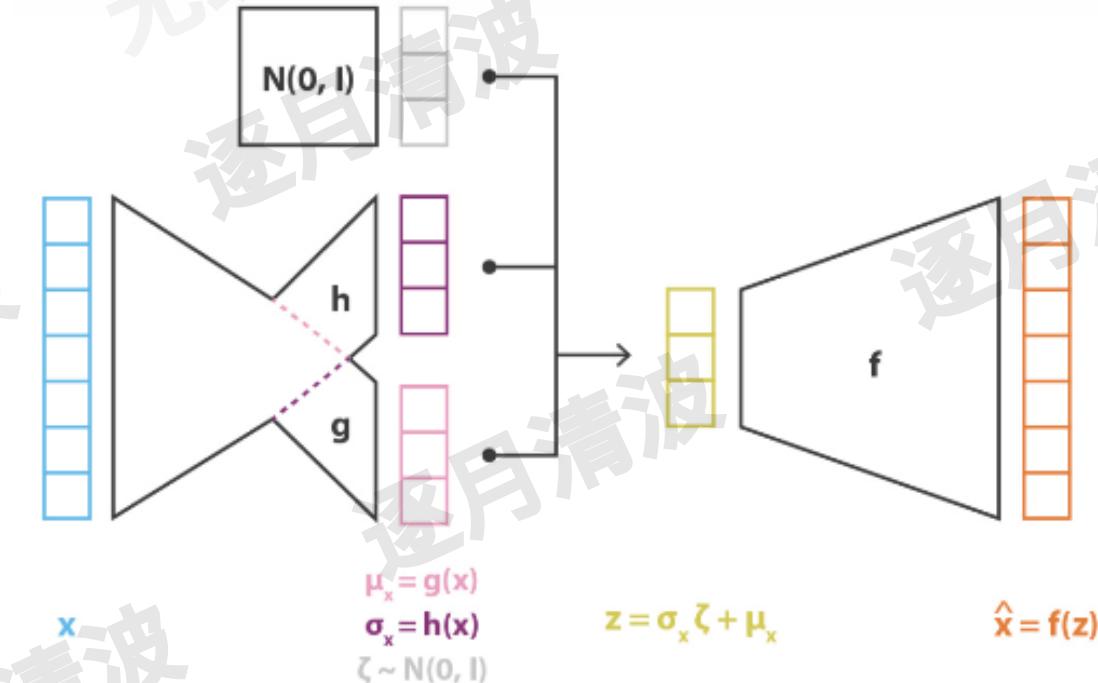
93、什么是变分自编码器 (VAE) ?解释一下它的组成和原理。 (3星)

变分自编码器 (Variational Autoencoder, VAE) 主要由两个部分组成：编码器 (Encoder) 和解码器 (Decoder)。它的主要原理是将输入数据通过编码器映射到一个潜在空间 (latent space) 中，并从潜在空间中的样本向量中解码重构输入数据。

VAE的核心思想是：在编码器中，将输入数据 x 转换成潜在空间 Z 中的均值向量 μ 与方差向量 σ ，使用 ξ 对这两个向量进行采样，生成一个潜在空间中的随机向量 $z = \mu + \sigma \cdot \xi$ ，其中 \cdot 为逐元素相乘。然后，将 z 送入解码器中，解码器会生成一组分布于数据空间内的输出 x 。与普通的自编码器相比，VAE在编码器中使用了概率分布，对于同一输入数据，它可以生成不同的随机向量和不同的输出，这是VAE生成多样性的保障。同时，VAE可以通过调整潜在空间中的随机向量 z 来生成各种新的输出图像或样本，这也使得它在数据生成方面表现出色。

在训练VAE时，我们的目标是最小化重构误差和潜在空间构成的KL散度。具体来说，我们希望通过训练得到的模型在解码器输出与输入之间保持最小的差距，同时使得训练得到的潜在向量在潜在空间上满足预先设定的先验分布，使得生成的潜在向量更加合理。

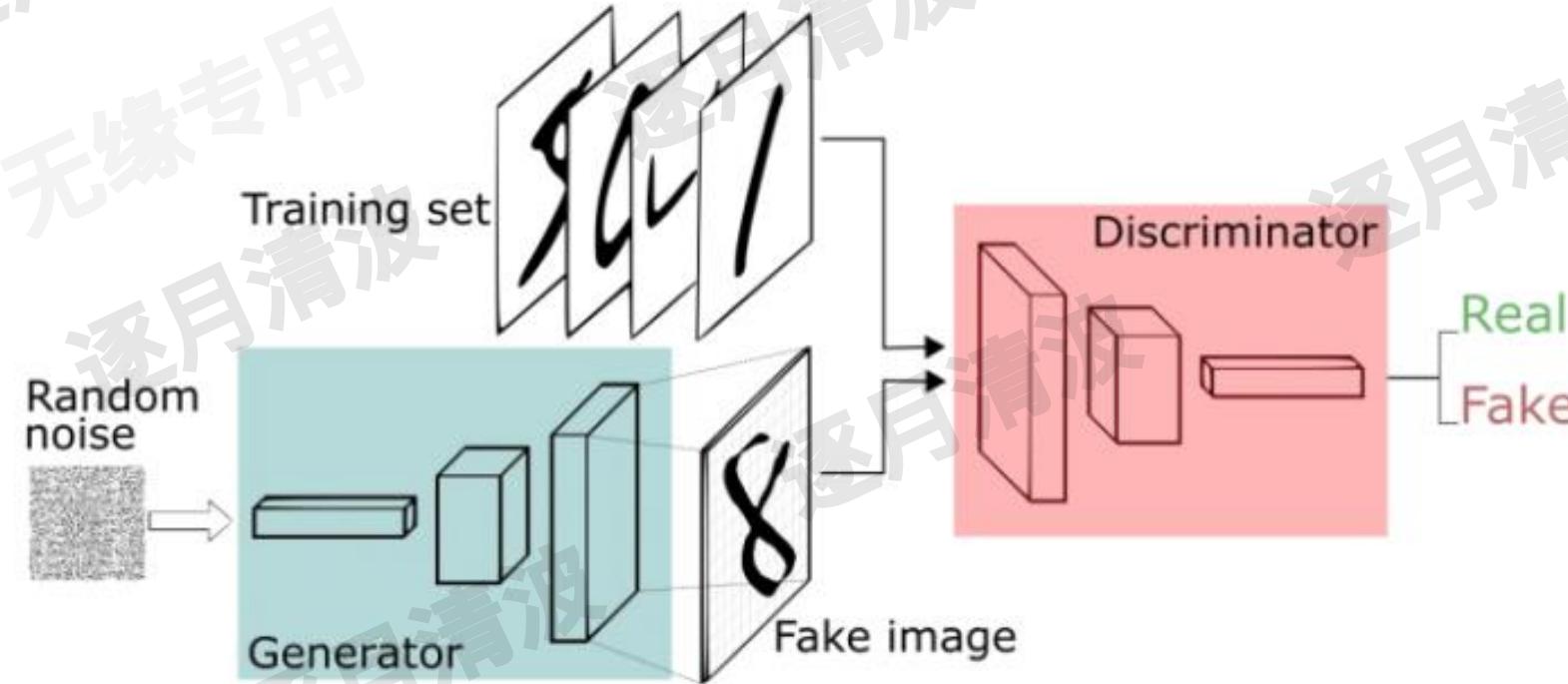
参考：<https://zhuanlan.zhihu.com/p/21741426>



94、什么是对抗生成网络 (GAN) ?解释一下它的组成和原理。 (3星)

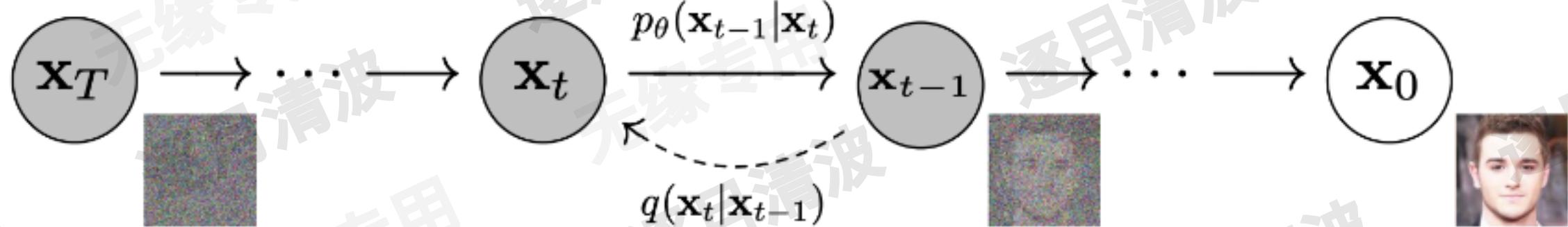
对抗生成网络 (GAN) 是一种用于生成模型的深度学习框架。它由两个神经网络组成：生成器和判别器。生成器尝试从噪声中生成与实际数据相似的样本，而判别器则尝试区分生成的样本与真实数据。GAN的核心原理是基于博弈论的最小最大 (min-max) 思想，即生成器和判别器各自形成自己的损失函数。生成器通过优化自身的损失函数，使得它生成的样本更接近真实数据，从而欺骗判别器；而判别器通过优化自身的损失函数，使得它能够更好地区分真实数据和生成数据，从而能够更快地发现生成器引入的错误。

GAN的总损失函数包含了生成器和判别器的损失函数，它们相互影响，通过迭代的训练过程不断提高对抗生成网络的性能。GAN在图像生成、文本生成等领域都有广泛的应用。



95、什么是扩散模型 (Diffusion Model) ? 解释一下它的组成和原理。 (2星)

扩散模型 (DM, Diffusion Model) 是一类生成模型，常见的生成模型还有GAN和VAE。扩散模型分为前向阶段和逆向阶段，在前向阶段中逐步向数据中添加噪声，直至数据变成完全的高斯噪声，然后在逆向阶段学习从高斯噪声中还原为原始数据。



前向阶段表示为图中从右往左的过程。从原始图像 x_0 开始，第 t 步在 x_{t-1} 的基础上添加噪声得到 x_t 。 x_t 只与 x_{t-1} 有关，直至 T 步后 x_T 完全变为高斯噪声。逆向过程表示为图中从左往右的过程。首先给定高斯噪声 x_T ，通过逐步去噪，直至将原始数据 x_0 恢复。

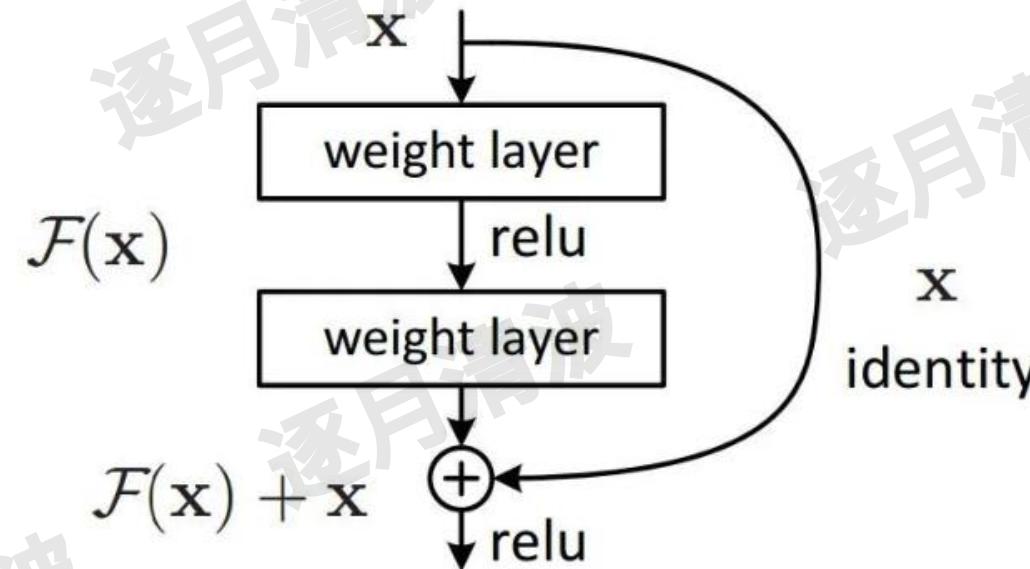
模型训练完后，只要给定高斯噪声就能够生成对应的图片，这样就达到了生成各种图片的目的。

96、什么是残差网络 (ResNet) ? 解释一下它的组成和原理。 (3星)

残差网络是一种用于深度神经网络中的特殊结构，它通过引入“残差块”来克服了梯度消失的问题。在传统的神经网络中，模型的表达能力会受到限制，因为网络深度增加时，梯度会变得越来越难以传递。残差块的引入解决了这个问题，使得网络可以轻松地达到数十层、数百层以及数千层。

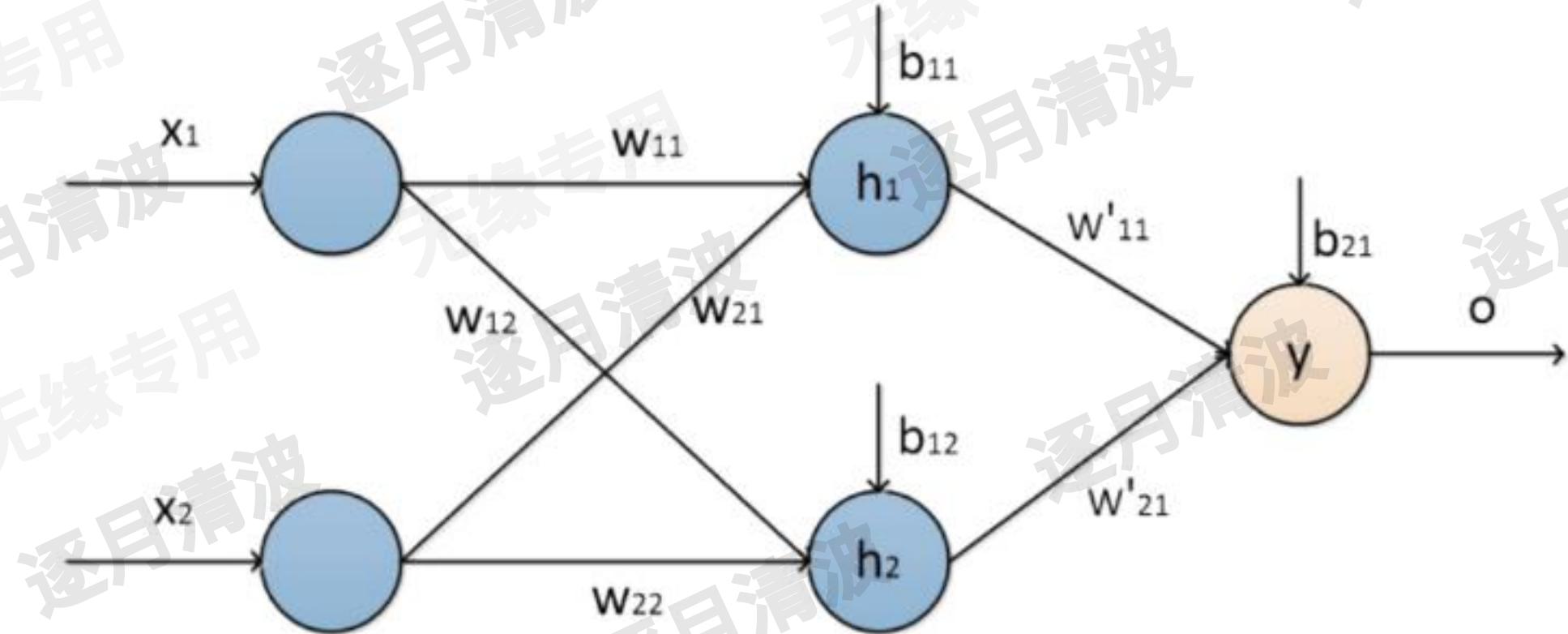
残差块是由多个卷积层和批归一层组成的，其中每个残差块的输入被用作下一层的输入，从而构建深度网络。残差块中的关键思想是使用跨层连接 (skip connections)，将输入直接加到输出上。这样做的原因是，网络可以通过这个跨层连接来学习差异性，即输出减去输入的差异，而不是从头开始学习整个输出。这种差异性可以让网络更容易地学习残差，从而提高模型的表现。

ResNet的原理可以被概括为：通过一种称为“跨层连接”的方法，允许网络重点关注于残留的特征学习，而不是从头开始学习每一层的特征。这个跨层连接释放了深度网络的潜力，使其可以进行更好的目标分类和识别。



97、如何通过神经网络来模拟异或 (XOR) 操作？(2星)

w₁₁=1
w₂₁=1
b₁₁=0.5
w₁₂=-1
w₂₂=-1
b₁₂=-1.5
w'11=1
w'21=1
b₁₁=1.5

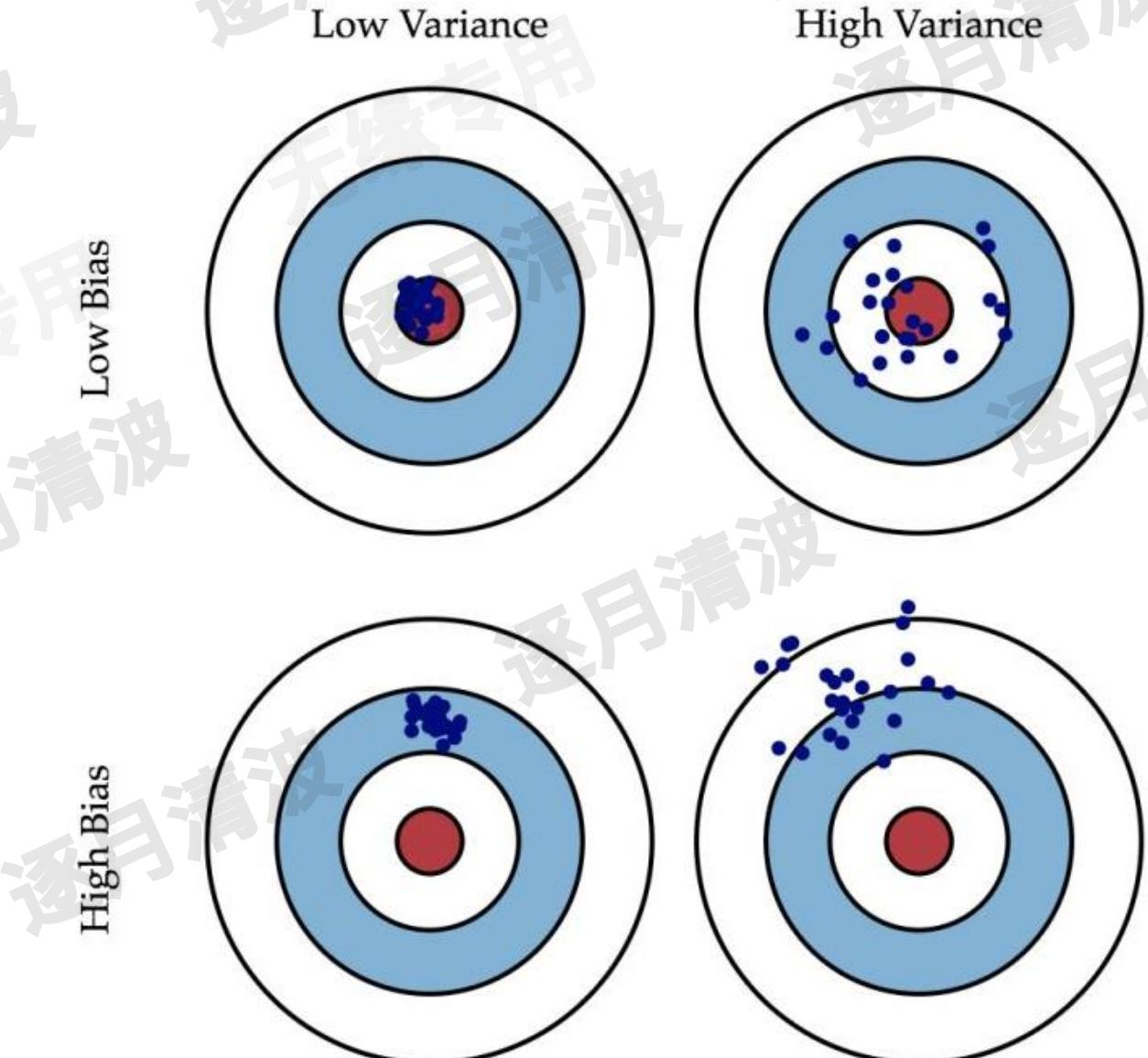


98、模型的泛化误差主要来源于哪两个方面？具体解释一下？（5星）

在有监督学习中，模型的泛化误差来源于两个方面—**偏差和方差**，具体来讲偏差和方差的定义如下：

偏差指的是由所有采样得到的大小为m的训练数据集训练出的所有模型的输出的平均值和真实模型输出之间的偏差。偏差通常是由于我们对学习算法做了错误的假设所导致的，比如真实模型是某个二次函数，但我们假设模型是一次函数。由偏差带来的误差通常在训练误差上就能体现出来。

方差指的是由所有采样得到的大小为m的训练数据集训练出的所有模型的输出的方差。方差通常是由模型的复杂度相对于训练样本数过高导致的，比如一共有 100 个训练样本，而我们假设模型是阶数不大于 200 的多项式函数。由方差带来的误差通常体现在测试误差相对于训练误差的增量上。



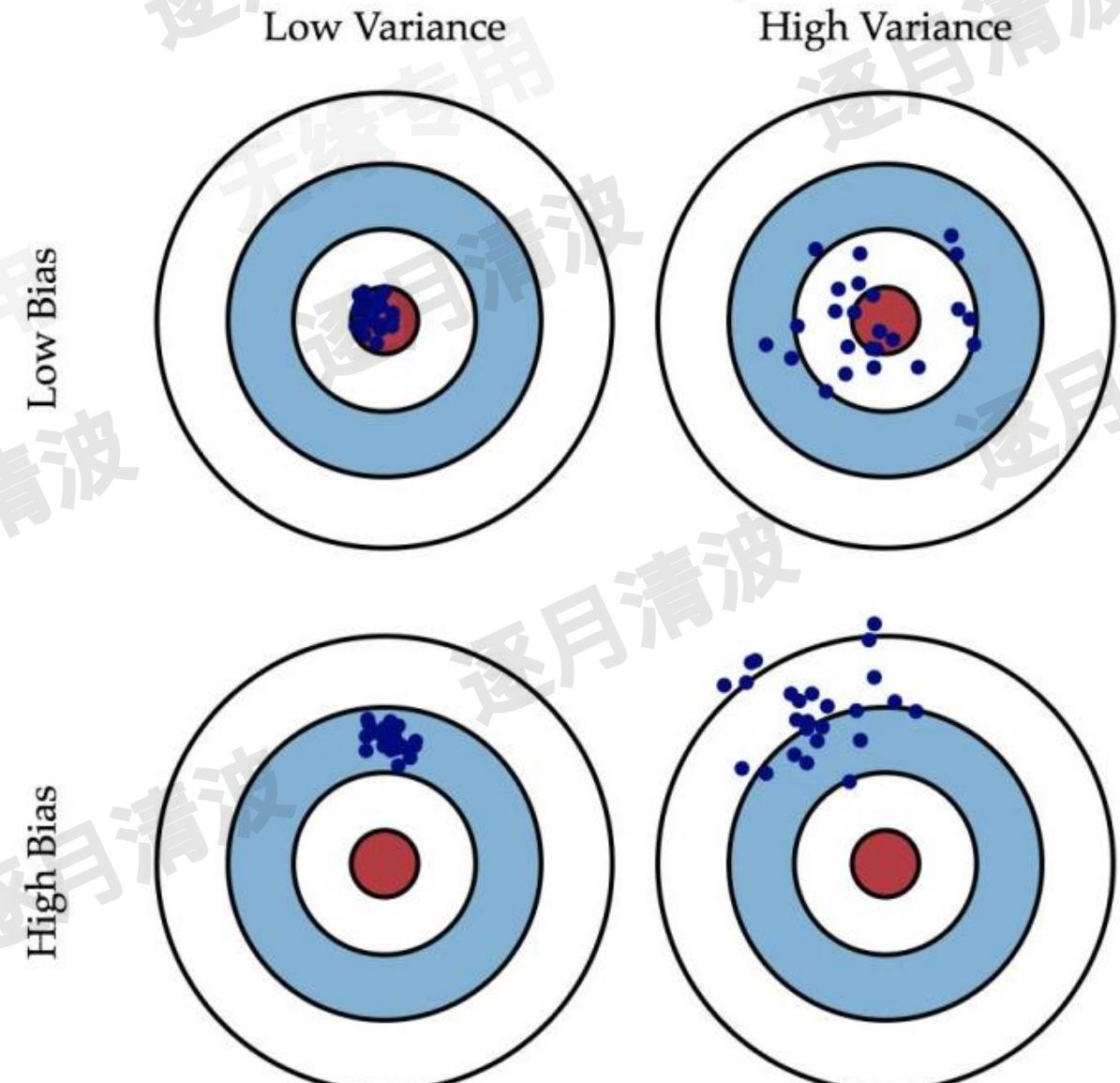
99、什么是偏差-方差权衡 (Bias-Variance trade-off) ? (4星)

偏差，是模型不考虑数据中的所有信息，而持续学习错误信息的倾向。

方差，是模型不考虑实际的数据情况，而持续学习随机信息的倾向。

如果模型对某一特定的数据集学习过多，并试图将该模型应用在其他未知数据上，则可能具有很高的误差。从给定的数据集中学习过多被称为过拟合。此种情况下，模型难以有效地推广应用于未知的数据。相反的，从给定的数据集中学习太少称为欠拟合。此种情况下，模型表现太差，甚至无法从给定的数据中学习。

机器学习解决问题的方式是不断努力寻找一个恰当的平衡点，构建一个**不过于复杂也不过于简单、能够泛化的、相对不准确但是有用的模型**。过拟合的模型会过于复杂，它在训练数据上表现非常好，但是在测试数据上表现欠佳；欠拟合的模型又过于简单，它在训练数据和测试数据上的表现都欠佳；一个良好的模型是在过拟合和欠拟合之间找到平衡，它表现良好，简单但不过于简单。这种平衡行为被称为偏差-方差权衡。



100、什么样的数据集不适用深度学习方法？(2星)

1. **数据集太小**，数据样本不足时，深度学习相对其它机器学习算法，没有明显优势。
2. **数据集的特征较为简单**：深度学习通常应用于具有高度抽象或非线性的特征的数据集，若数据集的特征较为简单，则深度学习的优势可能并不明显。
3. **数据集没有局部相关特性**，目前深度学习表现比较好的领域主要是图像 / 语音 / 自然语言处理等领域，这些领域的一个共性是局部相关性。图像中像素组成物体，语音信号中音位组合成单词，文本数据中单词组合成句子，这些特征元素的组合一旦被打乱，表示的含义同时也被改变。对于没有这样的局部相关性的数据集，不适于使用深度学习算法进行处理。
4. **需要可解释性的场景**。深度学习通常为黑箱模型，缺乏明确清晰的可解释性，如果对结果的可解释性要求很高，则不适用深度学习方法，或者需要经过特殊的处理。

101、描述一些KNN算法适用的场景。 (1星)

KNN (K-Nearest Neighbors) 算法通常适用于以下场景：

- 分类问题**: KNN算法是一种常用的分类算法，例如图像分类、文本分类和声音分类等。在分类时，我们将新的数据点与已有的数据点进行比较，并将其归到离它最近的数据点所属的类别中。
- 回归问题**: 例如房价预测或者贷款额度预估等。在回归问题中，KNN算法利用K个最近邻居的训练目标的平均值或加权平均值来对测试样本进行预测。
- 高维数据**: KNN算法是一种非参数算法，它不需要对数据进行任何的假设或者预测模型的构建。因此KNN算法对于高维数据拥有很好的性能，这是许多其他分类方法的局限所在。

102、描述一些神经网络适用的场景。（1星）

神经网络适用的场景为学习数据中的复杂关系和模式：

1. **计算机视觉**：神经网络可以用于图像的分类、目标检测、人脸识别和图像生成等方面。例如，卷积神经网络（CNN）在图像识别、物体检测等方面已经被广泛应用。
2. **自然语言处理**：神经网络可以用于语音识别、自然语言生成和情感分析等方面。例如，循环神经网络（RNN）和长短期记忆神经网络（LSTM）在自然语言处理领域中有很好的表现，以及基于编码器-解码器的机器翻译。
3. **安全问题**：神经网络可以用于垃圾邮件检测、信用风险评估和欺诈检测。
4. **时间序列预测**：神经网络可以用于时序性的数据处理与预测，例如天气预报等。
5. **智能控制**：神经网络可以用于实现自动驾驶汽车、飞行器等的智能控制。

103、为什么在拟合神经网络时随机优化方法比非随机方法更好？(2星)

在训练神经网络时，我们需要通过最小化损失函数来优化模型的权重和偏置，以使其能够更好地预测新数据。

深度神经网络通常拥有大量的参数，使得优化问题的搜索空间非常大，传统的非随机优化方法（如梯度下降）可能会1、卡在局部最优解中，导致泛化性能下降；2、计算量过于庞大以致优化速度很慢。

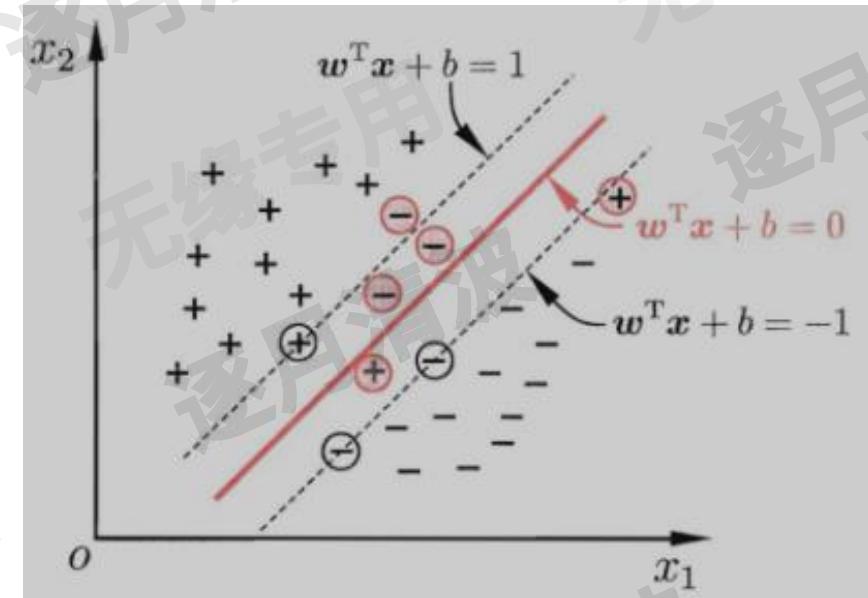
相比之下，随机优化方法（如随机梯度下降、Adam）可以通过随机化的方式避免陷入局部最优解，从而有更高的学习效率和优化性能。此外，随机优化方法易于扩展到大规模数据和高维参数空间，并可以并行化处理。

104、什么是软间隔支持向量机 (Soft-Margin SVM) ? (3星)

软间隔支持向量机是一种支持向量机算法的改进版，它允许在分类过程中某些数据点不满足线性可分条件，而是可以有一定的误差容忍度。软间隔支持向量机可以处理更为复杂的情形。它允许一些数据点处于边界线以内，从而增加了泛化能力，避免了硬间隔支持向量机可能出现的“过拟合”现象。

在软间隔支持向量机中，分类器会尽量最小化分类误差和间隔两个因素的综合代价函数，其中误差项表示了分类错误的样本点以及被分类器错误地分配到了间隔内部的样本点。同时，软间隔支持向量机仍然利用拉格朗日乘子法求解最优解，对应求解一个约束最优化问题。

在实践中，软间隔支持向量机几乎总是比硬间隔支持向量机鲁棒性更好。但是，与硬间隔支持向量机相比，软间隔支持向量机需要为目标函数引入一个正则化项，增加了计算负担和复杂性，需要更多的计算时间和资源。



$$\min_{[w,b]} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i$$

$$\text{s.t. } y_i (w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

$\xi_i = 0$ 时，样本满足约束 $y_i (w^T x_i + b) \geq 1$

$\xi_i > 0$ 时，样本满足 $y_i (w^T x_i + b) = 1 - \xi_i$

105、列举一些传统机器学习中防止过拟合的技术。 (3星)

1. **数据增强 (Data Augmentation)**: 通过增加训练数据集来扩充数据集，以使模型更容易泛化。
2. **正则化 (Regularization)**: 通过增加损失函数的惩罚项，限制模型的复杂度，从而使模型更具一般性，不容易过拟合。
3. **早停 (Early Stopping)**: 在训练过程中，当验证集的误差超过某个阈值时，停止训练，以避免过拟合。
4. **模型平均 (Model Averaging)**: 将多个训练出的模型进行集成，以避免单个模型的过拟合。如Bagging、Boosting等。
5. **模型简化 (Model Simplification)**: 对机器学习模型中的部分组成进行精简，防止学到过分细节的特征，例如决策树剪枝。

以上方法不仅适用于传统机器学习，也可以应用于深度学习领域。

106、列举一些深度学习中防止过拟合的技术。 (3星)

1. **Dropout**: 在训练过程中随机关闭一些神经元，以减少神经元之间的耦合，降低模型的复杂度，防止过拟合。
2. **Early Stopping**、数据增强与正则化（与机器学习中类似）
3. **Batch Normalization**: 通过对每一个批次的输入数据进行归一化，使其具有相同的均值和方差，减少了内部协变量的偏移，从而防止过拟合。**Layer Normalization**同理。
4. 降低学习率，减少迭代次数等防止过拟合。
5. 加注意力机制，**减少冗余**，让模型关注关键信息，而不是那些非核心的细节。

通常组合使用上述技术以获得更好的防止过拟合的效果。

107、考虑图片说明问题，给定一张图片作为输入，输出一个描述图片的句子。

(a) 描述一个能够做到这一点的神经网络结构，如何使用它们（输入是什么，输出是什么）。（2星）

这个问题通常涉及到图像描述 (image captioning) 任务，常用的神经网络模型是卷积神经网络和循环神经网络。

1. 卷积神经网络提取图像特征：将图片通过卷积神经网络（如 VGG、ResNet 等）的卷积层进行特征提取。这一步得到的是图像中不同区域的特征向量，通常使用一个全连接层将其转换为固定长度的特征向量。
2. 使用循环神经网络生成文本：使用一个循环神经网络（如 LSTM、GRU）在每个时间步生成一个单词，并将先前生成的单词作为输入。初始时，将前一步得到的图像特征作为网络的初始状态。在生成句子的过程中，可以使用一些技巧，如Beam Search或采样方法来增加句子的多样性。

输入为一张图片，输出是一句话描述图片中的内容。通过这种方式，可以训练模型去生成描述图像的自然语言句子，常常用在自动图像注释、自动图像搜索等相关领域。

108、(b) 描述训练过程（训练数据是什么？目标函数是什么？）(3星)

训练过程分为两个主要的部分：图像特征提取和文本生成。

1. 图像特征提取：首先，使用一个预训练的卷积神经网络（如 VGG、ResNet）对训练集中的图像进行特征提取。这些特征向量表示了图像中不同区域的语义信息。这个特征提取部分相当于一个特征提取器，由于其在检测图像特征方面表现优异，被广泛地应用于相关领域。
2. 文本生成：然后，在训练集中为每个图像提供相应的文本描述。一般来说，文本描述数目不止一个，因此一个图像会有多个描述或标注。对于每个图像和相应的文本描述，我们可以使用循环神经网络（如 LSTM、GRU）来学习图像特征和文本之间的对应关系。训练过程中的目标函数通常是，在所有图像和文本描述之间最小化损失函数。损失函数可以使用交叉熵损失、平均平方误差等。

109、(c) 描述测试过程（在测试阶段，给出一张图片作为输入，算法如何输出一个描述图片的句子）。（3星）

在测试阶段，我们可以使用之前训练好的模型来为一张新的图片生成一个描述句子。具体步骤如下：

1. 图像特征提取：使用预训练的卷积神经网络对输入的图片进行特征提取，并将其转化为一个固定大小的向量。
2. 文本生成：由于模型是使用循环神经网络进行训练的，因此在测试阶段，需要将该向量作为模型的初始状态。然后，在每个时间步，将先前的单词作为输入，并使用模型生成一个新的单词。生成的单词被组合在一起，形成一个完整的句子，直到遇到结束语（例如 STOP 或 END）。
3. 在生成句子的过程中，可以使用不同的技术，如 Beam Search 或采样方法来增加句子的多样性。Beam Search 是一种基于贪心算法的搜索方法，它在每个时间步维护一些可能的句子，并选择其中最有可能的 k 个句子作为候选，直到生成一个结束符（如 END）。采样方法则是在每个时间步根据当前单词的概率分布随机生成一个新的单词，然后再将其作为下一个时间步的输入。

110、(d) 假设我们想有一个参数来决定输出句子的多样性，可以通过在softmax函数中添加一个温度参数来实现。你会怎么做？(4星)

$$\text{softmax}(x, T)_l = \frac{e^{\frac{x_l}{T}}}{\sum_j e^{\frac{x_j}{T}}}$$

在计算softmax的时候，温度参数T可以调节不同值的“权重”。

当T<=1时，标签中最大值的权重会较大，也就是说让label更“尖锐”了；

反之，当T>1时，标签中最大值的权重会变小，除最大值以外的其它值权重会变大，让label更“平滑”了。

111、简单阐述boosting和bagging方法的异同点，并举几个相关的算法。（2星）

Baggging和Boosting都是模型融合，将弱分类器融合之后形成一个强分类器，融合之后的效果会比最好的弱分类器更好，它们的不同点在于：

1. 训练集

Bagging：每个训练集都是从原始训练集中有放回的选取出来的，每个训练集各不相同且相互独立。

Boosting：每一轮的训练集都是原始训练集，只是每次训练后会根据本轮的训练结果调整训练集中的各个样本的权重，调整完权重的训练集用于下一轮的训练。

2. 样本权重

Bagging：使用Bootstrap的方式均匀抽样

Boosting：根据每一轮的训练不断调整权值，分类错误的样本拥有更高的权值。

3. 弱分类器权重

Bagging：所有弱分类器权重相同，使用voting的方式（或均值）决定最终结果

Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

4. 并行能力

Bagging：各个预测函数可以并行生成，因为数据集相互独立，每个模型之间也独立，没有序列关系。

Boosting：各个预测函数只能顺序生成，因为下一个模型的产生依赖于之前模型的计算结果。

常见的基于Bagging的算法包括：随机森林、Bagging-SVC、Bagging-AdaBoost等

常见的基于Boosting的算法包括：AdaBoost算法、GBDT算法、XGBoost算法等

112、什么是GANs的非饱和损失函数？其相对于饱和损失的优势在何处？(2星)

1. 饱和Loss: 生成器希望最小化被判断为假的概率

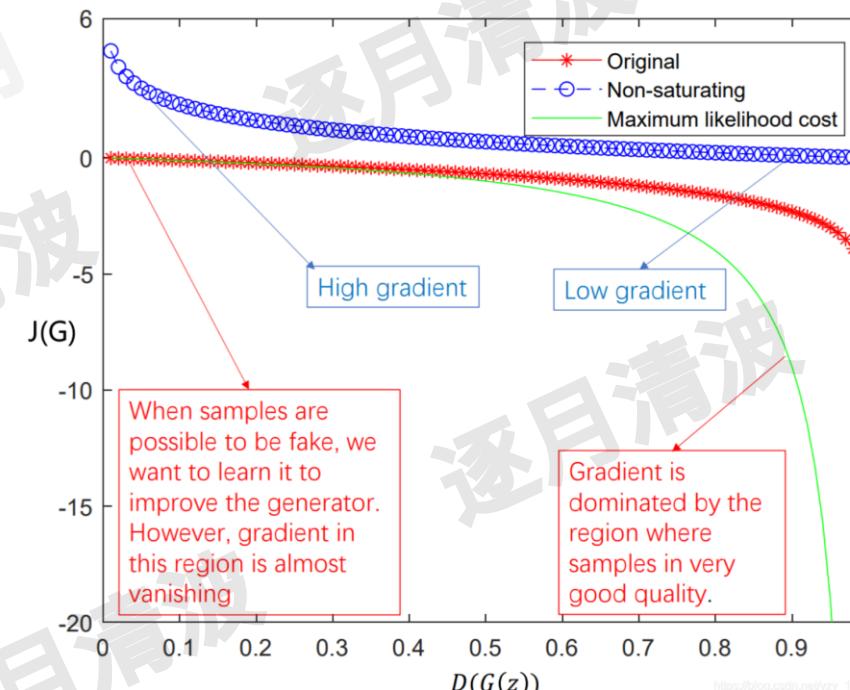
$$\min \log(1 - D(G(z)))$$

$$\max \log(D(G(z)))$$

$$\text{or } \min -\log(D(G(z)))$$

2. 非饱和Loss: 生成器希望最大化被判断为真的概率

在训练的初始阶段，生成器生成的样本很容易被判别器识别出来，也就是 $D(G(z))$ 趋近于0，此时饱和GAN的Loss_G的梯度会很小，所以饱和了。而非饱和GAN的Loss_G的梯度会大很多，能够为网络的权重更新提供好的梯度方向，帮助收敛。



113、在训练GANs的时候有哪些关键点或者需要注意的地方？（2星）

1. 需要尽力避免传给生成器的梯度消失；
2. 需要尽力避免梯度爆炸的问题；
3. 训练中后期需要让生成样本被分类为真的概率和真实样本被分类为真的概率相当，并且损失函数长时间保持稳定

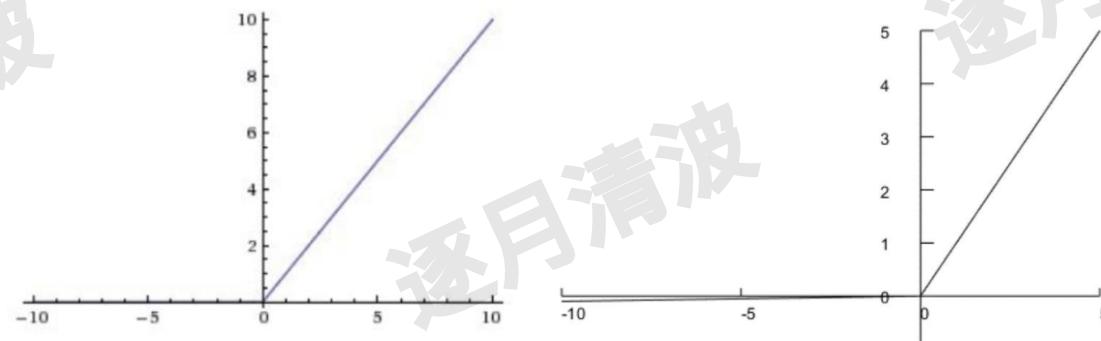
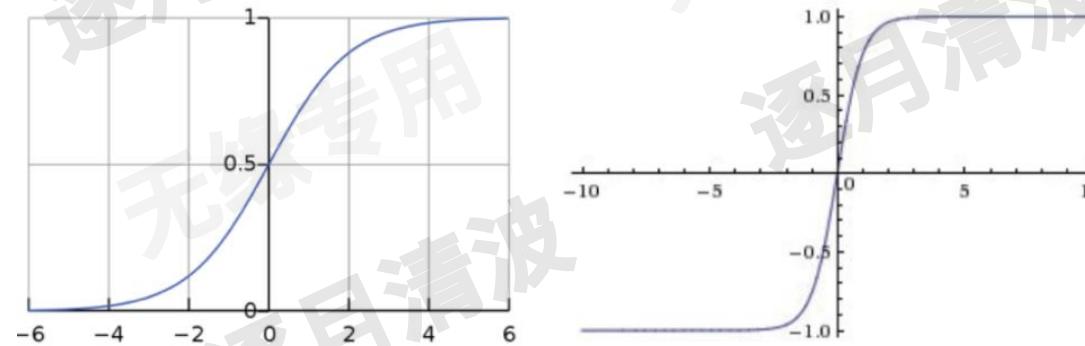
114、简述sigmoid/tanh/relu各自的优缺点，还有哪些可选择的激活函数可以有更优的改进？（4星）

sigmoid激活函数可将输入压缩到0到1之间的范围内，具有良好的数学性质和平滑的导数。但它的主要缺点是，在输入太大或太小时会**饱和**，导致梯度消失或梯度爆炸的问题。此外，**sigmoid**函数**中心不是零**，导致收敛速度缓慢；

tanh激活函数是**sigmoid**函数的扩展，它能够将输入数据映射到[-1, 1]的范围内，**是0中心的**。但是它也存在类似于**sigmoid**函数的问题，即当输入过大或过小时会饱和，导致梯度消失或梯度爆炸的问题。而且**tanh**函数的导数对于输入接近饱和区域的点的变化比较缓慢；

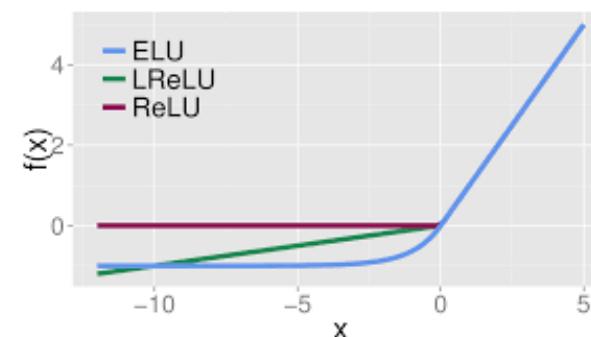
ReLU激活函数只有在输入为正时才返回一个非零值，可以加速模型的训练，特别是在使用反向传播算法时。另外，**ReLU**简单有效，**是线性的**，计算速度也非常快。但对于负数输入的梯度为零，因此训练过程中可能会出现“死亡神经元”问题。此外，当输入过大时，**ReLU**也会出现爆炸问题。

其他可选择的激活函数，如**LeakyReLU**, **ELU**等，它们针对**ReLU**的缺点进行了改进。例如**LeakyReLU**通过引入一个小的斜率来修正“死亡神经元”问题。而**ELU**在负数输入时具有非零导数，从而减少梯度消失的风险，同时对抗“死亡神经元”问题。



ELU函数表达式为：

$$f(x) = \begin{cases} \alpha(e^x - 1), & x \leq 0 \\ x, & x > 0 \end{cases}$$



115、决策树中，如果有重复特征，那么对树的分割与预测能力有什么影响？(3星)

1. 如果重复特征是离散的，并且是 ID3 或 C4.5 算法生成的决策树，那么重复特征只会被选择一次，因为这些算法不会复用已经选择过的特征。
2. 如果重复特征是连续的，并且是 CART 算法生成的决策树，那么重复特征可能会被多次选择，因为 CART 算法可以复用特征，并且每次选择一个最优的切分点。
3. 如果重复特征是离散的，并且是 CART 算法生成的决策树，那么重复特征也可能被多次选择，因为 CART 算法每次只做二分划分，而不是多分叉划分。
- 4.

决策树的预测能力不受重复特征（多重共线性）影响，但是数据的解释性会被影响。

随机森林可以返回特征的重要性（importance），当有多重共线性时，importance会被影响。一些具体多重共线性的特征的重要性会被相互抵消，从而影响我们解释和理解特征。

比如说，特征A和B完全一样，我们在用随机森林时，它们的重要性应该非常接近（考虑到随机性）。如果我们在训练前删除特征B，那么特征A的重要性就会翻番。这会影响了我们对特征、数据的理解。

116、决策树中，如果有单值特征，那么对信息熵以及树的分割有什么影响？(2星)

决策树中如果有单值特征，即某个特征的取值只有一个，那么这个特征对于划分数据集没有任何作用，因为它不能提高节点的纯度或者减少信息熵。

单值特征无法为不同的样本提供任何分类信息。在这种情况下，决策树算法将选择使用其他可用特征进行分裂，并简单地跳过该单值特征。

因此，一般在构建决策树之前，会先去掉单值特征，以减少计算量和提高效率，可以显著提高决策树性能，并且可以帮助消除决策树不稳定性和过度拟合的问题。

除此之外，还可以通过数据处理方法，在数据中使用一些聚类技术，将单一特征集成到一个合适的特征集合中，以避免单一特征对分类器的影响。

117、随机森林与XGBoost的区别是什么？(3星)

随机森林是基于Bagging思想的算法，XGBoost是基于Boosting思想的算法。

1. 构建过程

随机森林是通过随机的选择样本和特征来构建多个决策树的，每个决策树单独进行预测，通过投票或平均等方式来融合多个决策树的结果。而XGBoost是在已有的模型基础上，通过加入新的决策树来不断拟合模型的残差，进行模型迭代，最后将所有决策树的结果加权相加得到最终结果。

2. 损失函数

两者的损失函数有所不同。随机森林采用的是基尼不纯度或熵来决定每个节点的优化目标，而XGBoost则通过用目标函数的负梯度方向近似残差，来控制每个节点的划分。

3. 正则化

随机森林在构建每棵树时没有正则化，也不需要保证同一决策树的节点之间无相关性。而XGBoost则通过L1和L2正则化来防止模型过拟合，同时通过加上树的复杂度作为惩罚项来限制同一决策树节点之间的相关性。

4. 处理缺失值

随机森林能够自动处理缺失值，因为在每个节点划分时仅仅需要比较当前样本特征的某个维度是否缺失即可，不需要对缺失值进行填充；而XGBoost则需要指定一个分割方向作为缺失值的分支进行处理。

118、为什么不能在Boosting里使用强分类器 (Strong Classifier) ? (3星)

Boosting是一种将弱分类器提升为强分类器的方法，它通过不断调整样本的权重和分类器的权重来增强模型的表现。如果在Boosting里使用强分类器，可能会导致以下问题：

- 强分类器可能会过拟合训练数据，导致泛化能力差。
- 强分类器可能会降低Boosting的效果，因为它们之间的差异性小，难以互补。
- 强分类器可能会增加Boosting的计算复杂度，因为它们通常需要更多的时间和资源来训练。

在Boosting算法中，每个弱分类器被训练时都需要关注先前分类器未正确分类的数据点，以便尽可能减少其错误率。如果使用强分类器，它的分类能力已经很强了，并且很难进一步提高精度。这意味着在训练Boosting模型时，之后的弱分类器很难关注到强分类器错分的数据点，因此无法通过加权组合来提高整体分类准确度。同时，使用强分类器还会使得每个弱分类器的权重很小，从而让Boosting算法失去它的优势。

因此，在Boosting算法中，通常只使用弱分类器，以便可以通过加权组合来最大化模型的预测准确性。

119、如果训练过程中准确性很高，但是测试集上准确性很低该怎么办？（2星）

这个问题可能是由于以下原因造成的：

- 学习率过大或过小，导致梯度下降陷入局部最优或震荡
- 训练集和测试集数据分布不一致，导致模型泛化能力不足
- 训练集和测试集数据不平衡，导致模型偏向于某一类别
- 模型过于复杂，导致过拟合
- 输入特征与标签之间没有明确的关联，或特征太少
- 数据没有归一化或存在噪声

可以尝试以下方法：

- 调整学习率，使其适应数据的规模和复杂度
- 使用交叉验证或其他方法检查训练集和测试集的数据分布是否一致
- 增加数据量或使用数据增强技术，使训练集和测试集更加丰富和多样化
- 减少模型的参数量或使用正则化、dropout等技术防止过拟合
- 选择更合适的特征或使用特征提取、降维等技术增强特征表达能力
- 对数据进行归一化或去噪处理，使其更符合模型的假设

120、假设我们有一个自动驾驶系统和一个神经网络。这个神经网络可以预测1秒和10秒后的汽车位置。哪个结果需要更强的正则化？为什么？(2星)

在这个情景中，**10秒后的位置预测会需要更强的正则化**。主要原因有：

1. 预测10秒后的位置需要更多的信息来训练神经网络，如果我们不加正则化，那么网络可能会尝试过分拟合训练数据，这会导致过拟合现象。而过拟合会导致模型在面对新的数据时预测性能下降，这对于自动驾驶系统是非常危险的。
2. 预测10秒后的位置很可能会涉及到更多的特征，因而可能导致模型比较复杂。较为复杂的模型也要求使用更强的正则化来减少过拟合的影响。

121、什么是Dropout？它是如何防止过拟合的？(3星)

Dropout是一种防止神经网络过拟合的技术，它的基本思想是在训练过程中，以一定的概率随机地丢弃一些神经元，从而减少神经元之间的相互依赖，增强网络的泛化能力。

Dropout的原理是：在训练阶段，对于每一层神经元，按照一个给定的概率 p （通常为0.5）将其暂时从网络中移除，即将其输出值置为0。这样可以防止某些特征只在特定的神经元组合下有效，迫使网络学习更加鲁棒的特征。

Dropout可以有效地缓解过拟合的原因有以下几点：

- Dropout相当于对多个不同的子网络进行集成学习，取平均的效果可以抵消一些过拟合的影响。
- Dropout减少了神经元之间复杂的共适应关系，使得每个神经元都要对输入数据有较强的适应性，从而提高了网络的泛化能力。
- Dropout可以看作是一种数据增强的方法，通过随机地屏蔽掉一些神经元，相当于增加了训练样本的多样性，降低了模型对噪声和异常值的敏感度。

122、Dropout在训练阶段和测试/推理阶段的计算方式有何区别？（3星）

Dropout是一种常用的防止过拟合的方法，在训练深度神经网络时广泛使用。在训练阶段，Dropout会随机选择一部分神经元并将其输出置为0，**这部分神经元在该batch的训练中不会被使用。**在测试/推理阶段，**Dropout不会使用**，因为这会导致输出结果随机不稳定。

实际上，在测试/推理阶段，为了保持网络的期望输出不变，需要对训练过程中被置为0的神经元的权重进行一定的修正。这种修正方法称为“**inverted dropout**”（反向Dropout），具体方法是将训练中**某个神经元的输出乘以其保留概率**（即不被置为0的概率）。例如，在训练时Dropout的保留概率为0.5，在测试/推理时，需要将某个神经元的输出乘以0.5，以保证网络的期望输出不变。

需要注意的是，之所以要进行dropout是因为激活函数的神经元在训练阶段容易出现过拟合，而在测试阶段不存在这个问题，所以要加以区分。

123、什么是Huber损失函数？它有什么优点？(3星)

Huber损失函数是一种用于回归问题的损失函数，它是均方误差损失函数和绝对值误差损失函数的折中方案。

Huber损失函数的数学表达式为：

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & \text{if } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$

其中 y 是目标值， $f(x)$ 是模型预测的值， δ 是一个常数，通常取值在1到2之间。当预测值和目标值之差小于 δ 时，采用均方误差损失函数，否则采用绝对值误差损失函数，这样可以平衡均方误差损失和绝对值误差损失在抵消噪声和保留离群值方面的效果。

Huber损失函数在回归问题中的优点主要包括两个方面：一是对于离群点的敏感程度相对于均方误差损失函数较小；二是在训练过程中可以对损失进行削减，这对于在处理极端离群点时非常有用。相对于只使用绝对值误差损失函数的做法，Huber损失函数能更好地权衡各种数据情况下损失函数的平滑性和鲁棒性。

124、监督学习常见的损失函数有哪些？简述它们的特点。（4星）

监督学习中常见的损失函数有以下几种：

1. **平方损失函数** (Mean Square Error, MSE)：是回归问题中最常见的损失函数，其特点是误差求平方，因此对误差大的样本惩罚更严厉，易受异常值影响。
2. **绝对损失函数** (Mean Absolute Error, MAE)：也是回归问题中常用的损失函数，其特点是对误差求绝对值，对异常值的影响不是很大，但是不容易求导。
3. **对数损失函数** (Logarithmic Loss, Log Loss)：是分类问题中常用的损失函数，它惩罚分类错误的情况，对正负样本的错误惩罚是不同的，对于二分类问题，其值域范围为 $[0, 1]$ ，值越小表示模型的分类结果越好。
4. **Hinge损失函数**：也是分类问题中常用的损失函数，用于支持向量机 (SVM) 算法。其特点是当样本分类正确时，损失为0，当样本分类错误时，损失为 $|y-w*x+b|$ ，其中 y 为真实值， w 和 b 为模型参数， x 为样本特征向量。
5. **交叉熵损失函数** (Cross-Entropy Loss)：也是常见的分类问题中的损失函数，在神经网络的训练中被广泛应用。特别地，对于二分类问题，交叉熵损失函数可以简化为Log Loss。和Log Loss一样，交叉熵损失函数惩罚分类错误的情况，对正负样本的错误惩罚是不同的，值越小表示模型的分类结果越好。

125、简要说明使用信息增益来做出筛选离散特征的过程。 (3星)

在决策树算法中，通过判断每个特征对于分类结果的影响来进行特征选择。

信息增益是一种经典的特征选择方法，它是用来衡量一个特征对数据集分类能力的影响程度。

具体来说，信息增益是从信息论中引入的概念，其定义为在已知一个事件发生时所获得的信息量。在分类问题中，信息增益的计算方式是通过比较没有特征约束的熵和特征约束下的熵之差来度量特征的分类能力。具体过程如下：

1. 计算未分类时的熵 $H(D)$ 。假设有 K 个类别， k 指第 k 个类别

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

其中， $|D|$ 表示样本总数， $|C_k|$ 表示属于第 k 个类别的样本数量。

2. 对于每个离散特征 A ，计算该特征条件下数据集的熵 $H(D|A)$ 。

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i)$$

其中， n 表示特征 A 的取值个数， D_i 表示 A 特征为第 i 个取值时的样本集合，即 $D_i = \{x | x \in D, A(x) = a_i\}$ ， $H(D_i)$ 表示样本集合 D_i 的熵。

3. 计算信息增益 $g(D, A)$ 。

$$g(D, A) = H(D) - H(D|A)$$

4. 选择具有最大信息增益的特征进行分类。

126、对于连续特征，我们通常设定一个阈值，将数据划分为2个部分。简要说明如何决定最优的阈值。（2星）

1. 假设在样本集合D上特征a有n个取值，先把n个值从小到大排个序。确定一个阈值，把样本集合分成两部分。
2. 阈值怎么取？取排序后两个相邻的值的均值（那么n个值就有 $n-1$ 个阈值），相当于n个小朋友从矮到高站成一排，你选个位置分成两堆，位置就等同于阈值。
3. 分别比较这 $n-1$ 个阈值的信息增益，选使得信息增益最大的那个值作为阈值来划分。

127、为什么朴素贝叶斯在有重复特征时常常会表现很差？（2星）

朴素贝叶斯在有重复特征时常常会表现很差的原因是：

朴素贝叶斯的基本假设是特征之间是条件独立的，也就是说每个特征对分类结果的影响都是独立的，不受其他特征的影响。但是如果有重复特征，那么这个假设就不成立了，因为重复特征之间是完全相关的，它们会对分类结果产生冗余或者干扰的影响。

例如，如果一个分类器有两个相同的特征 x_1 和 x_2 ，那么在计算后验概率 $P(y|x_1, x_2)$ 时，就相当于把 x_1 的影响重复了两次，这样就会导致后验概率的估计不准确，可能会偏向于某个类别，而忽略了其他特征的作用。

因此，在使用朴素贝叶斯时，应该尽量避免有重复或者高度相关的特征，或者对特征进行降维或者选择，以提高分类器的性能和准确度。

128、你在使用各种模型来预测股票盈利能力。下面哪种模型是你应该避免使用的？随机森林、神经网络、k近邻、朴素贝叶斯。（2星）

朴素贝叶斯。

它是一种用于分类和预测的简单且有效的模型，但它不适合用于预测股票盈利能力。

相比之下，随机森林、神经网络和k近邻等模型可能都比朴素贝叶斯更适合用于此类问题。这是因为预测股票盈利能力需要考虑大量的因素，如财务报告、宏观经济因素、市场趋势等等。这些因素之间可能存在复杂的相互影响和非线性关系。

而朴素贝叶斯模型通常假设各个特征之间是独立的，这种假设可能无法刻画复杂的因素之间的关系。因此，朴素贝叶斯不适合用于预测股票盈利能力。

129、为什么在k-均值聚类中要使用不同的初始聚类中心运行多次？（2星）

这是因为在k-均值聚类中，聚类的结果会受到初始聚类中心的选取影响。

如果初始聚类中心选取比较差，可能会导致聚类结果不理想，出现聚类中心陷入局部最优解，或者收敛速度变慢等问题。因此，多次运行k-均值聚类并在每次运行时使用不同的初始聚类中心，可以增加聚类结果的稳定性和准确性，减少陷入局部最优解的可能性。

常见的做法是运行k-均值聚类n次，然后综合多次运行的结果，例如通过计算平均值或者选取最好的一次聚类结果作为最终的聚类结果。此外，多次运行k-均值聚类可以帮助选择最优的k值，即在多次运行后比较不同k值下的聚类效果，来确定最佳的k值。需要注意的是，不同的初始聚类中心应该具有一定的随机性，以避免出现重复结果和过度拟合。

130、使用随机梯度下降时的基本假设是什么？(2星)

使用随机梯度下降时的基本假设是

1. 每个训练样本都是相互独立且具有相同分布的。这个假设常常被称为**独立同分布假设** (**Independent and Identically Distributed**, 简称*i. i. d.*)
2. 目标函数的梯度是可计算、连续的 (有时候提到**梯度Lipschitz连续**) $\|\nabla F(w) - \nabla F(w')\|_2 \leq L\|w - w'\|_2$
3. 能够被分解为许多小批次数据的梯度之和。

换句话说，假设每个训练样本的损失函数都是可导的，因此对于给定训练数据集，应该存在一个可优化的损失函数，其梯度可以在数据集中的小批量数据上计算，同时这些小批量数据的梯度之和等于整个数据集的梯度。这个假设是随机梯度下降算法的基础，因为算法的核心是在数据集的小批量数据上计算梯度，并使用该信息更新模型参数。

131、什么是超参数？举一个调参的例子。（1星）

超参数（Hyperparameters）是在训练模型之前需要手动设置的参数。这些参数不能自动从数据中学习，但它们可以影响模型的性能和学习效果。

一个经典的调参例子是在深度学习模型中调整学习率。学习率是控制模型在每次迭代中更新权重的速度的参数。如果学习率设置得太高，模型可能会发散而无法收敛；如果学习率设置得太低，模型会收敛缓慢或在局部最优解处卡住。因此，调整学习率是一个重要的调参任务。可以使用网格搜索或随机搜索等技术来寻找最佳超参数。例如，我们可以通过尝试不同的学习率，比如0.001、0.01、0.1等来找到最佳的值。

另一个例子是，如果我们使用K近邻算法进行分类，K值也是一个超参数。我们可以通过网格搜索、随机搜索或贝叶斯优化等方法，尝试不同的K值（最相近的点的个数），并用交叉验证或其他评估指标来比较模型的准确率，从而找到最优的K值。

132、如何选择学习率？如果学习率太低或太高会发生什么？(2星)

学习率是一个重要的超参数，它决定了梯度下降法中参数更新的速度和方向。

如果学习率太低，那么训练过程会很慢，而且可能陷入局部最优解或者鞍点；

如果学习率太高，那么训练过程会很不稳定，而且可能错过全局最优解或者导致损失函数发散。

如何选择学习率，有以下几个建议：

- 一般来说，可以从一个较大的学习率开始，比如0.01或0.001，然后根据训练效果逐渐减小学习率，比如每隔一定的轮数或者当损失函数下降缓慢时，将学习率乘以一个衰减因子。
- 可以使用一些自适应的优化算法，比如Adam、RMSProp等，它们可以根据梯度的大小和方差动态调整学习率。
- 可以使用一些启发式的方法，比如“三角方法”或“余弦方法”，它们可以让学习率在一个范围内周期性地变化，从而避免陷入局部最优解或者鞍点。

133、Batch Norm与Layer Norm分别是什么？(3星)

Batch Normalization的基本思想是对于每个 mini-batch 中的样本，对其输入的每个特征在 mini-batch 的维度上进行归一化。具体来说，对于输入特征 x ，BN的计算过程可以表示为：

- 1、在 mini-batch 的维度上计算均值和方差
- 2、对于每个特征 i ，对输入特征 x 进行归一化
- 3、对归一化后的特征进行缩放和平移。 g 和 b 是可学习的参数，它们的维度与输入特征的维度相同

BN可以有效减少梯度消失和梯度爆炸问题。

$$\mu_i = \frac{1}{M} \sum x_i, \quad \sigma_i = \sqrt{\frac{1}{M} \sum (x_i - \mu_i)^2 + \epsilon}$$

$$\hat{x} = \frac{x - \mu}{\sigma}$$

$$y = g \cdot \hat{x} + b$$

而Layer Normalization方式是综合考虑一层所有维度的输入，计算该层的平均输入值和输入方差，然后用同一个规范化操作来转换各个维度的输入，是在输入特征的维度上进行归一化

$$\sigma^l = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i^l - \mu^l)^2}$$

$$\mu^l = \frac{1}{H} \sum_{i=1}^H x_i^l$$

$$x^l = w_i^l h^l$$

$$\hat{x} = \frac{x - \mu}{\sigma}$$

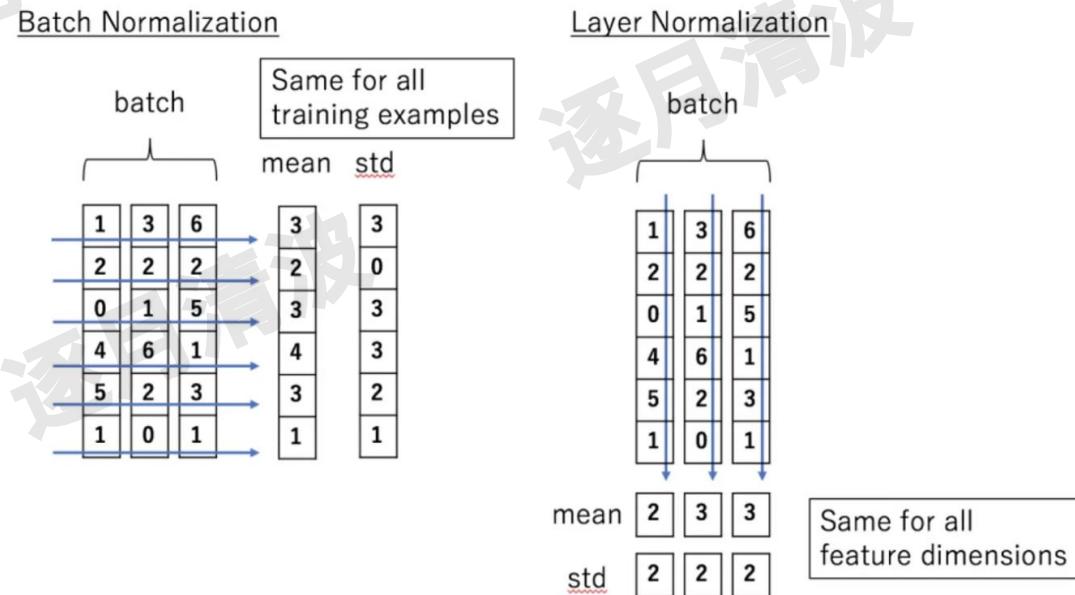
$$y = g \cdot \hat{x} + b$$

其中 H 枚举了该层所有的输入神经元的个数。对应到标准公式中，四大参数 μ, δ, g, b 均为标量(BN中是向量)，所有输入共享一个规范化变换。

134、Batch Norm与Layer Norm分别适用的场景是什么？(2星)

Batch Norm有两个优点，第一是可以解决训练过程中，各层分布不同，增大了学习难度的问题。它可以使损失平面更加的平滑，从而加快收敛速度；第二个优点是缓解了梯度饱和问题（如果使用sigmoid这种含有饱和区间的激活函数的话），加快收敛。因此Batch Norm在MLP和CNN上使用的效果都比较好，在进行训练之前，要做好充分的shuffle，否则效果会差很多。

而Layer Norm针对单个训练样本进行，不依赖于其他数据，因此可以避免BN中受mini-batch数据分布影响的问题，可以用于小mini-batch场景、动态网络场景和RNN，特别是自然语言处理领域。此外，LN不需要保存mini-batch的均值和方差，节省了额外的存储空间。



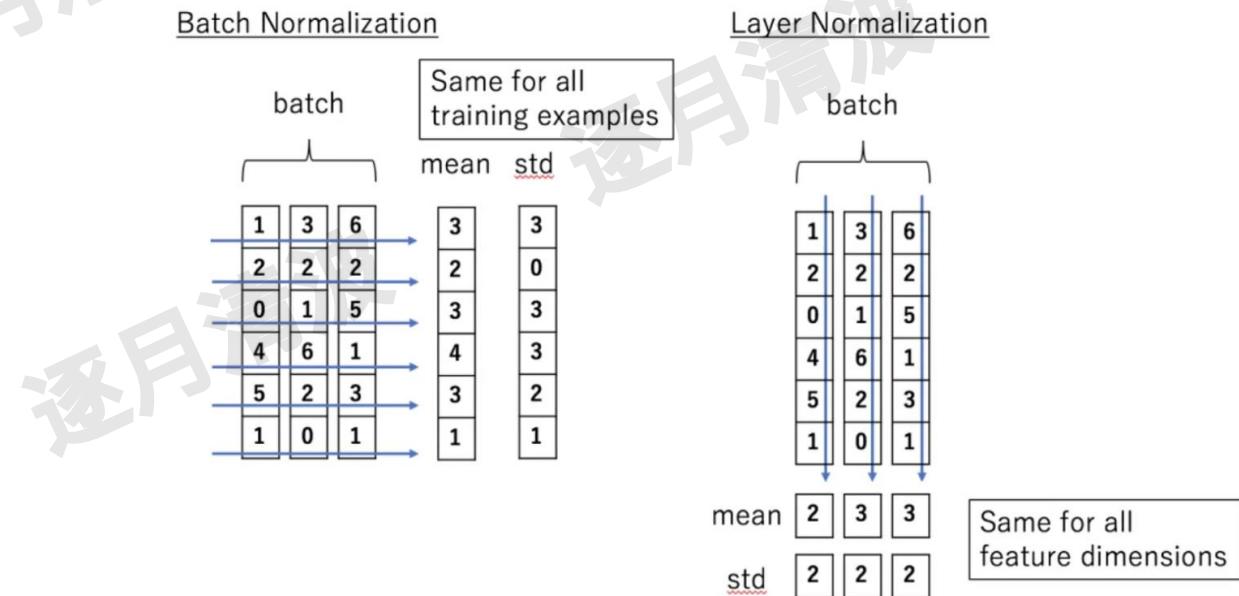
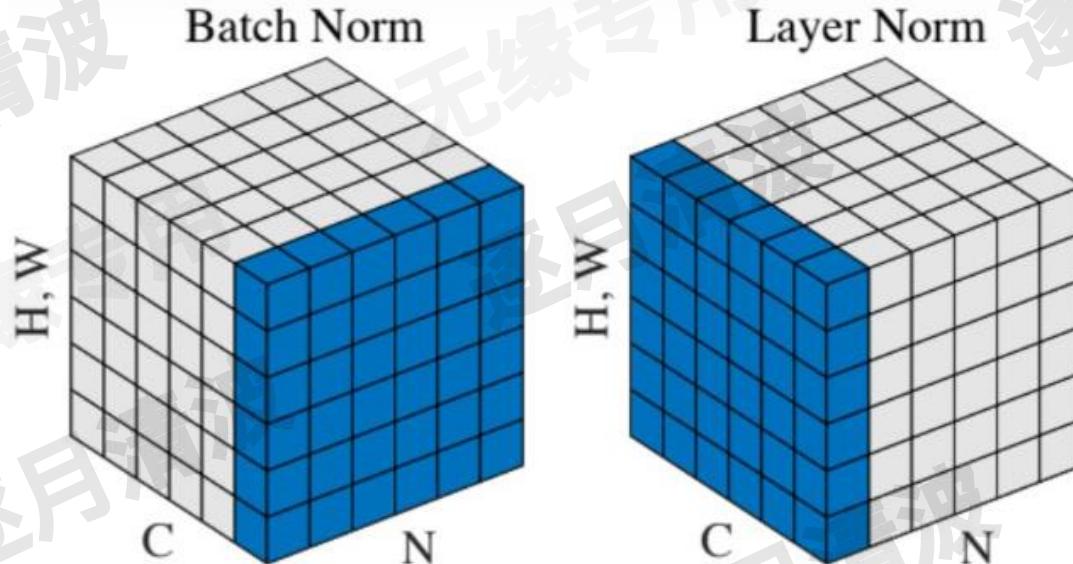
135、对比一下Batch Norm与Layer Norm的异同与优缺点。 (2星)

从操作过程上来讲，BN针对的是同一个batch内的所有数据，而LN则是针对单个样本。

从特征维度来说，BN对同一batch内的数据的同一纬度做归一化，因此有多少维度就有多少个均值和方差；而LN则是对单个样本的所有维度来做归一化，因此一个batch中就有batch_size个均值和方差。

BN的转换是针对单个神经元可训练的——不同神经元的输入经过再平移和再缩放后分布在不同的区间
而LN对于一整层的神经元训练得到同一个转换——所有的输入都在同一个区间范围内。

如果不同输入特征不属于相似的类别（比如颜色和大小），那么LN 的处理可能会降低模型的表达能力。



136、什么是SGD with Momentum? (2星)

为了抑制SGD的震荡，SGDM认为梯度下降过程可以加入惯性。下坡的时候，如果发现是陡坡，那就利用惯性跑的快一些。SGDM全称是SGD with momentum，在SGD基础上引入了一阶动量。而所谓的一阶动量就是该时刻梯度的指数加权移动平均值： $\eta \cdot m_t := \beta \cdot m_{t-1} + \eta \cdot g_t$

(其中当前时刻的梯度 g_t 并不严格按照指数加权移动平均值的定义采用权重 $1-\beta$ ，而是使用我们自定义的学习率 η)，那么为什么要用移动平均而不用历史所有梯度的平均？因为移动平均存储量小，且能近似表示历史所有梯度的平均。SGDM的参数更新公式为

$$\Delta\theta_t = -\eta \cdot m_t = -(\beta m_{t-1} + \eta g_t)$$

$$\theta_{t+1} = \theta_t - (\beta m_{t-1} + \eta g_t)$$

在SGD中增加动量的概念，使得前几轮的梯度也会加入到当前的计算中（会有一定衰减），通过对前面一部分梯度的指数加权平均使得梯度下降过程更加平滑，减少动荡，收敛也比普通的SGD快。当前梯度方向与累计梯度方向一致时，梯度会被加强，从而这一步下降幅度增大，若方向不一致，则会减弱当前下降的梯度幅度。

137、什么是SGD with Nesterov Acceleration? (2星)

在SGD的改善中，除了利用动量法，使用惯性跳出局部沟壑以外，我们还可以尝试往前看一步。

想象一下你走到一个盆地，四周都是略高的小山，你觉得没有下坡的方向，那就只能待在这里了。可是如果你爬上高地，就会发现外面的世界还很广阔。因此，我们不能停留在当前位置去观察未来的方向，而要向前一步、多看一步、看远一些。

SGD with Nesterov Acceleration 是在SGD、SGD-M的基础上的进一步改进，改进点在于当前时刻梯度的计算，我们知道在时刻t的主要下降方向是由累积动量决定的，自己的梯度方向说了也不算，那与其看当前梯度方向，不如先看看如果跟着累积动量走了一步，那个时候再怎么走。也即在Momentum的基础上将当前时刻的梯度 g_t 换成下一时刻的梯度 $\nabla J(\theta_t - \beta m_{t-1})$ ，参数更新公式为

$$\Delta\theta_t = -\eta \cdot m_t = -(\beta m_{t-1} + \eta \nabla J(\theta_t - \beta m_{t-1}))$$

$$\theta_{t+1} = \theta_t - (\beta m_{t-1} + \eta \nabla J(\theta_t - \beta m_{t-1}))$$

它在Momentum的基础上进行了改进，比Momentum更具有前瞻性，除了利用历史梯度作为惯性来跳出局部最优的沟壑以外，还提前走一步看看能否直接跨过沟壑。

138、什么是Adagrad？有什么特点？（2星）

Adagrad即adaptive gradient，是一种自适应学习率的梯度法。它通过记录并调整每次迭代过程中的前进方向和距离，使得针对不同问题都有一套自适应学习率的方法。Adagrad最大的优势是不需要手动来调整学习率，但与此同时会降低学习率。

SGD及其变种以同样的学习率更新每个维度的参数（因为 θ 通常是向量），但深度神经网络往往包含大量的参数，这些参数并不是总会用得到。对于经常更新的参数，我们已经积累了大量关于它的知识，不希望被单个样本影响太大，希望学习速率慢一些；对于偶尔更新的参数，我们了解的信息太少，希望能从每个偶然出现的样本身上多学一些，即学习速率大一些。

因此，AdaGrad则考虑对于不同维度的参数采用不同的学习率，具体的，对于那些更新幅度很大的参数，通常历史累计梯度的平方和会很大，相反的，对于那些更新幅度很小的参数，通常其累计历史梯度的平方和会很小。所以在一个固定学习率的基础上除以历史累计梯度的平方和就能使得那些更新幅度很大的参数的学习率变小，同样也能使得那些更新幅度很小的参数学习率变大，所以AdaGrad的参数更新公式为

$$V_t = \text{diag}(v_{t,1}, v_{t,2}, \dots, v_{t,d}) \in R^{d \times d}$$

$$v_{t,i} = \sum_{t=1}^t g_{t,i}^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{V_t + \epsilon}} \odot g_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \odot g_t$$

139、什么是RMSProp？(1星)

由于AdaGrad单调递减的学习率变化过于激进，我们考虑一个改变二阶动量计算方法的策略：不累积全部历史梯度，而只关注过去一段时间窗口的下降梯度，采用Momentum中的指数加权移动平均值的思路。

RMSProp就是在AdaGrad的基础上将普通的历史累计梯度平方和换成历史累计梯度平方和的指数加权移动平均值，所以只需将AdaGrad中的 $v_{t,i}$ 的公式改成指数加权移动平均值的形式即可，也即

$$v_{t,i} = \beta v_{t-1,i} + (1 - \beta) g_{t,i}^2$$

$$V_t = \text{diag}(v_{t,1}, v_{t,2}, \dots, v_{t,d}) \in R^{d \times d}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{V_t + \epsilon}} \odot g_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \odot g_t$$

140、什么是Adam ? (2星)

Adam即Adaptive Moment Estimation，是能够自适应时刻的估计方法，能够针对每个参数，计算自适应学习率。这是一种综合性的优化方法，在机器学习实际训练中，往往能够取得不错的效果。

Adam=adagrad(用于处理稀疏的梯度) + RMSProp(处理非常态数据)

首先计算一阶动量

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

类似RMSProp计算二阶动量

$$v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$$

$$V_t = \text{diag}(v_{t,1}, v_{t,2}, \dots, v_{t,d}) \in R^{d \times d}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_{t,i} = \frac{v_{t,i}}{1 - \beta_2^t}$$

$$\hat{V}_t = \text{diag}(\hat{v}_{t,1}, \hat{v}_{t,2}, \dots, \hat{v}_{t,d}) \in R^{d \times d}$$

分别加上指数加权移动平均值的修正因子

$$\Delta\theta_t = -\frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \hat{m}_t$$

Adam的参数更新公式为：

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \hat{m}_t$$

141、什么是Nadam？(2星)

Adam只是将Momentum和Adaptive集成了，但是没有将Nesterov集成进来，而Nadam则是在Adam的基础上将Nesterov集成了进来，也即 $\text{Nadam} = \text{Nesterov} + \text{Adam}$ 。

具体思想如下：由于Nesterov的核心在于，计算当前时刻的梯度时使用了未来梯度 $\nabla J(\theta_t - \beta m_{t-1})$ 。NAdam 提出了一种公式变形的思路，大意可以这样理解：只要能在梯度计算中考虑到未来因素，就算是达到了Nesterov的效果。既然如此，我们就不一定非要在计算 g_t 时使用未来因素，可以考虑在其他地方使用未来因素。

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \hat{m}_t \\ &= \theta_t - \frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right)\end{aligned}$$

此时，如果我们将 $t-1$ 时刻的动量用第 t 时刻的动量近似代替的话，那么我们就引入了未来因素，所以将 m_{t-1} 替换成 m_t 即可得到Nadam的表达式

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \left(\frac{\beta_1 m_t}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \\ &= \theta_t - \frac{\eta}{\sqrt{\hat{V}_t + \epsilon}} \odot \left(\beta_1 \hat{m}_t + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right)\end{aligned}$$

142、哪些机器学习算法需要做归一化？（3星）

需要进行归一化处理的算法是

- 一、对特征之间距离度量敏感的算法。例如，基于距离计算的模型：如KNN等
- 二、需要通过梯度下降法求解的模型：线性回归、逻辑回归、支持向量机、神经网络等

143、哪些机器学习算法不需要做归一化？(3星)

不需要进行归一化处理的算法是树类模型：例如决策树与随机森林。

因为它们不关心变量的值，而是关心变量的分布和变量之间的条件概率；

因为数值缩放不影响分裂点位置，对树模型的结构不造成影响。按照特征值进行排序的，排序的顺序不变，那么所属的分支以及分裂点就不会有不同。

而且，树模型是不能进行梯度下降的，因为构建树模型（回归树）寻找最优点时是通过寻找最优分裂点完成的，因此树模型是阶跃的，阶跃点是不可导的，并且求导没意义，也就不需要归一化。

144、简述判别式模型的原理、思路与优缺点。 (2星)

判别式模型 (Discriminative Models) 则是指直接建立输入到输出的映射模型，即模型的主要目标是建立输入数据和输出数据之间的对应关系。判别模型的目标是最大限度地减少分类误差或预测误差等，因为这些模型不需要考虑样本 (特征) 之间的概率分布，所以它们通常也具有更高的学习精度。典型的判别式模型包括逻辑回归、支持向量机 (SVM) 、决策树、神经网络等。

总结：

- 判别模型是直接对 $P(Y|X)$ 建模，就是说，直接根据 X 特征来对 Y 建模训练。
- 判别式模型是思路是对 $P(Y|X)$ 建模；对所有的样本只构建一个模型，确认总体判别边界；观测到输入什么特征，就预测最可能的label；
- 判别式模型的优点是什么对数据量要求没生成式模型严格，速度也会快，小数据量下准确率也会好些。

145、简述生成式模型的原理、思路与优缺点。 (2星)

生成式模型 (Generative Models) 是指通过学习输入数据在特定条件下的联合分布概率来生成新的数据，也就是说，生成模型可以模拟样本在整个概率分布上的分布情况，可以用来从训练数据中学习样本 (特征) 之间的依赖关系，进而生成新的数据。典型的生成式模型包括朴素贝叶斯、隐马尔可夫模型 (HMM) 和深度信念网络 (DBN) 等。

总结：

- 生成式模型是对 $P(X, Y)$ 建模，确定维护这个联合概率分布的所有信息参数
- 对于分类问题，是要对每个 label 都需要建模，最终选择最优概率的 label 为结果，所以没有判别边界；
- 生成式模型的优点是所包含的信息非常齐全，不仅可以用来输入 label，还可以干其他的事情。生成式模型关注结果是如何产生的。
- 但是生成式模型需要非常充足的数据量以保证采样到了数据本来的面目，所以速度相比之下较慢。

146、简述Stacking的原理、特点和基本思路。 (3星)

Stacking的原理是基于串行策略的方式进行学习。它是一种模型集成的方法，将多个基模型的预测结果作为元特征 (meta feature)，再训练一个高层模型，使用元特征和真实标签进行训练，从而得到最终的预测结果。

Stacking的特点是初级学习器与次级学习器之间存在依赖关系，初学习器的输出作为次级学习器的输入

Stacking的基本思路是：

1. 先从初始训练集训练 T 个不同的初级学习器；
2. 利用每个初级学习器的输出构建一个次级数据集，该数据集依然使用初始数据集的标签；
3. 根据新的数据集训练次级学习器；
4. 多级学习器的构建过程类似；

147、在Bagging与Boosting中，哪些模型更适合作为基学习器？能使用线性分类器作为基学习器吗？（4星）

在Bagging与Boosting中**不稳定的学习器**更适合作为基学习器。因为不稳定的学习器容易受到样本分布的影响（方差大），很好的引入了随机性；这有助于在集成学习（特别是采用 Bagging策略）中提升模型的泛化能力。为了更好的引入随机性，有时会随机选择一个属性子集中的最优分裂属性，而不是全局最优（随机森林）。

所以，**决策树和神经网络**适合作为基学习器。神经网络也属于不稳定的学习器；通过调整神经元的数量、网络层数，连接方式初始权重也能很好的引入随机性和改变模型的表达能力和泛化能力。

Bagging中**不推荐**使用线性分类器作为基学习器，线性分类器都属于稳定的学习器（方差小），对数据不敏感；甚至可能因为 Bagging 的采样，导致在训练中难以收敛，增大集成分类器的偏差；

Boosting中**可以使用**，因为Boosting方法主要通过降低偏差的方式来提升模型的性能，而线性分类器本身具有方差小的特点，所以两者有一定共性。例如，XGBoost 中就支持以线性分类器作为基学习器

148、从对特征的处理角度对比一下逻辑回归与GBDT。（2星）

从特征组合角度：

逻辑回归不具有特征组合的能力，只是一次性地寻求最大化熵的过程，对每一维的特征都假设独立，因此只具备对已有特征空间进行分割的能力，更不会对特征空间进行升维（特征组合）；

GBDT采用最小化均方损失来寻找分裂特征及对应分裂点，所以自动会在当前根据特征 A 分裂的子树下寻求其他能使负梯度最小的其他特征 B，这样就自动具备寻求好的特征组合的性能，因此也能给出哪些特征比较重要（根据该特征被选作分裂特征的次数）。

从特征的稀疏性：

逻辑回归假设特征各个维度独立，因此只具有线性分界面，实际应用中，多数特征之间有相关性，只有维度特别大的稀疏数据中特征才会近似独立，所以适合应用在特征稀疏的数据上；

GBDT更适合处理稠密特征，对于连续型特征导入GBDT做特征组合来代替一部分手工特征工程，而对于ID类特征，目前的主流做法是直接 one-hot + embedding 来将高维稀疏特征压缩为低纬稠密特征，也进一步引入了语意信息，有利于特征的表达。

149、CNN中，池化层针对的区域是什么？(2星)

池化层针对区域是非重叠区域。

池化层可以对该输出进行下采样 (Down-Sampling)，通常使用max pooling或average pooling的方式来进行。

在进行池化操作时，我们将滤波器按照步幅 (stride) 的间隔在输出中滑动，将滤波器覆盖到指定区域上，然后通过max或average的方式得到该区域的池化值，最终将池化值输出给后续层进行处理。通过池化操作，我们可以减小模型的计算量和参数量，同时还能帮助模型抵抗一定程度的过拟合风险。

池化层是一个特征选择，信息过滤的过程。也就是说损失了一部分信息，这是一个和计算性能的一个妥协。随着运算速度的不断提高，这个妥协会越来越小。现在部分网络都开始少用或者不用池化层了。但是池化层的防止过拟合作用依然存在。

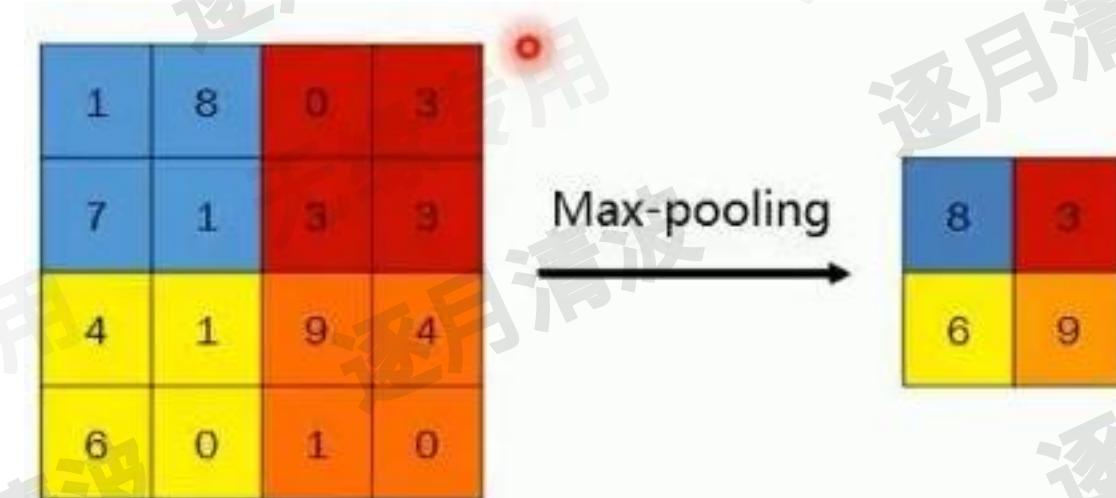
150、CNN中，池化层的种类有哪些，作用是什么？(3星)

一、最大值池化

思路：取领域内最大值

优点：抑制网络参数误差造成估计均值偏移问题

特点：更好提取纹理信息

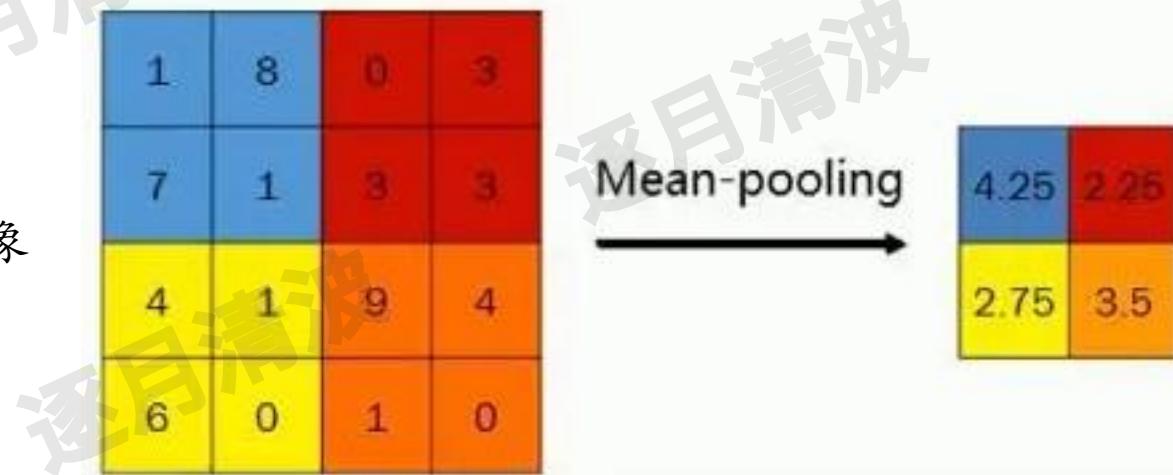


二、均值池化

思路：对领域内特征值取平均

优点：抑制由领域大小受限造成的估计值方差增大现象

特点：对背景的保留效果好



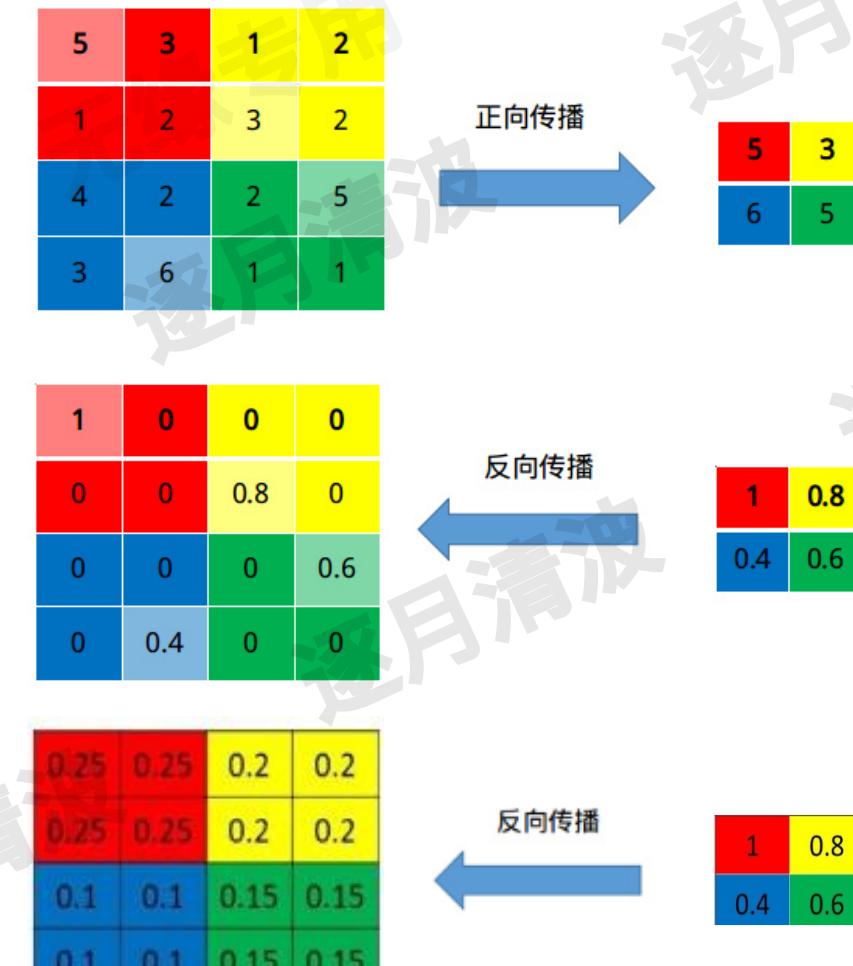
151、CNN中，池化层的反向传播是什么样的？(2星)

由于 Pooling 操作容易改变 feature map 的尺寸大小，使得 Pooling 层不可导

方法：将一个像素的 loss (梯度) 传递给4个像素，且需要保证传递的 loss (梯度) 总和不变。

Mean pooling: 将值的梯度均等分为 $n*n$ 份 分配给上一层，以保证 池化 前后 梯度之和保存不变；

Max pooling: 把梯度直接传给前一层值最大的一个像素，而其他像素不接受梯度，也就是为0



152、CNN为何具有平移不变性和平移等变性？(3星)

平移不变性是指目标在空间内发生平移，但是结果（标签）不变。

平移等变性是指如果我们移动输入图像中的物体，它的表示也会在输出中移动同样的量。

卷积神经网络的平移不变性，是池化层赋予的。卷积神经网络的平移等变性，是参数共享赋予的。卷积层对平移是敏感的，输入的平移能等价地影响输出。直观地，如果把一张输入图像先平移后卷积，其结果与先卷积后平移效果是一样的。在传统的神经网络中，当计算一层的输出时，权重矩阵的每个元素只使用一次。当它乘以输入的一个元素后，就再也不会用到了。但是在卷积神经网络中，卷积核的每一个元素都作用在输入的每一个位置上。卷积运算中的参数共享保证了我们只需要学习一个参数集合，而不是对于每个位置都需要学习一个单独的参数集合。

不管采用什么样的池化函数，当输入做出少量平移时，池化能够帮助输入的表示近似不变。例如我们使用最大池化，只要变换不影响到最大值，我们的池化结果不会收到影响。对于一个卷积核来说，只有一个值的变动会影响到输出，其他的变换都不会造成扰动。平均池化的近似不变性就稍弱些。

153、CNN有什么局限性？(2星)

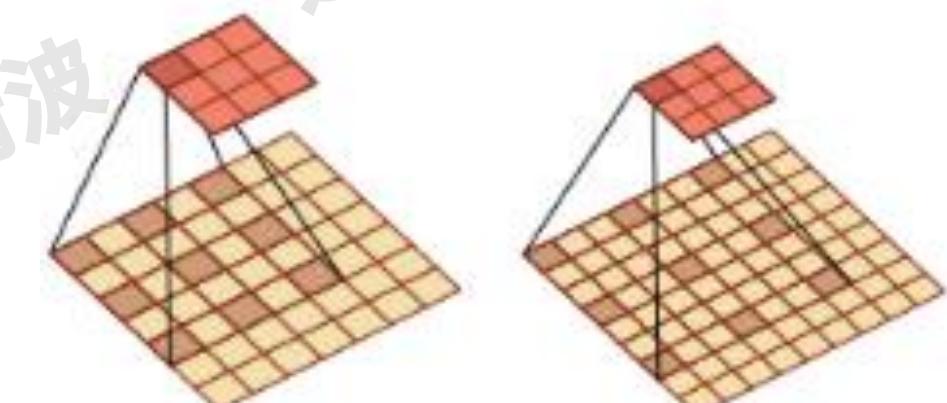
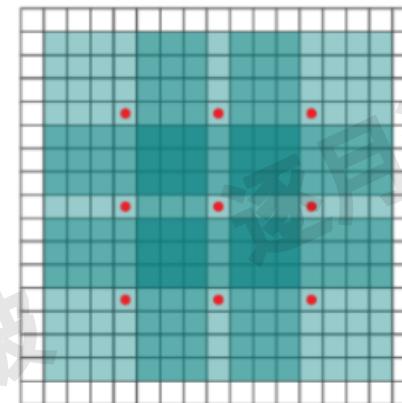
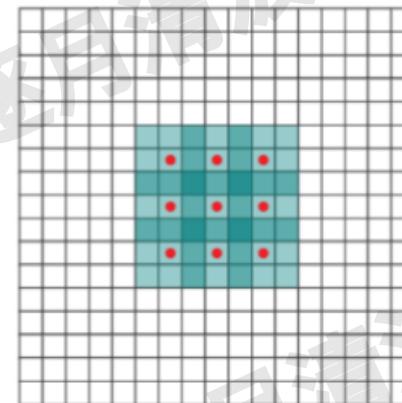
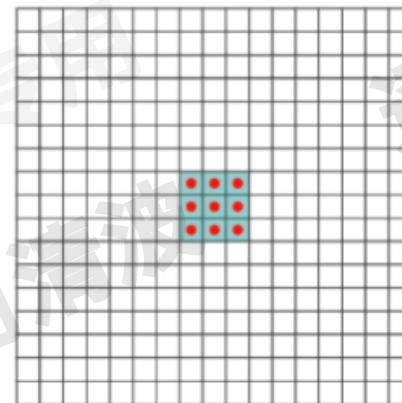
- 以CNN为代表的神经网络使用了条件独立假设，其特征提取范围受到限制；而循环神经网络或者Attention机制的模型能够看到序列整体的信息，因此对于序列建模任务，单纯的CNN存在先天不足。
- 卷积神经网络的核心思想是捕捉局部特征。对于序列任务，比如文本来说，局部特征就是由若干单词组成的滑动窗口。卷积神经网络的优势在于能够自动地对特征进行组合和筛选，获得不同抽象层次的语义信息。例如加入更多卷积层 -> 网络深度增加 -> 参数量增大 -> 容易过拟合 -> 引入 Dropout 正则化 -> 引入超参数 -> 网络庞大冗余
- 使用CNN做序列任务的优点在于其共享权重的机制，使得训练速度相对较快；但是其最大的缺点在于无法构建长距离依存关系。
- 有许多试图改进CNN模型的尝试，使其能够应用于序列任务。诸如用于句子语义建模的动态卷积神经网络DCNN、可以在整个序列的所有窗口上进行卷积的时延神经网络TDNN等。

154、什么是空洞卷积 (Dilated CNN) ? (2星)

Dilated Convolutions, 翻译为空洞卷积, 与普通的卷积相比, 除了卷积核的大小以外, 还有一个扩张率参数, 主要用来表示扩张的大小, 以及去掉 pooling。

空洞卷积与普通卷积核的大小是一样的, 在神经网络中即参数数量不变, 区别在于扩张卷积具有更大的感受野。感受野指的是特征图上某个元素的计算受输入图像影响区域的范围, 即特征图上神经元能够“看到”的图像范围, 例如 3×3 卷积核的感受野大小为9。

- (a) 普通卷积, 1-dilated convolution, 卷积核的感受野为 $3 \times 3 = 9$ 。
- (b) 空洞卷积, 2-dilated convolution, 卷积核的感受野为 $7 \times 7 = 49$ 。
- (c) 空洞卷积, 4-dilated convolution, 卷积核的感受野为 $15 \times 15 = 225$ 。



155、什么是注意力机制 (Attention) ?为什么要引入它? (3星)

注意力机制 (Attention Mechanism) 模仿了生物观察行为的内部过程，即一种将内部经验和外部感觉对齐从而增加部分区域的观察精细度的机制。例如人的视觉在处理一张图片时，会通过快速扫描全局图像，获得需要重点关注的目标区域，也就是注意力焦点。然后对这一区域投入更多的注意力资源，以获得更多所需要关注的目标的细节信息，并抑制其它无用信息。Attention帮助模型对输入的每部分赋予不同的权重，抽取更重要的信息，使模型做出准确判断。同时，不会给模型计算与存储带来更大开销；

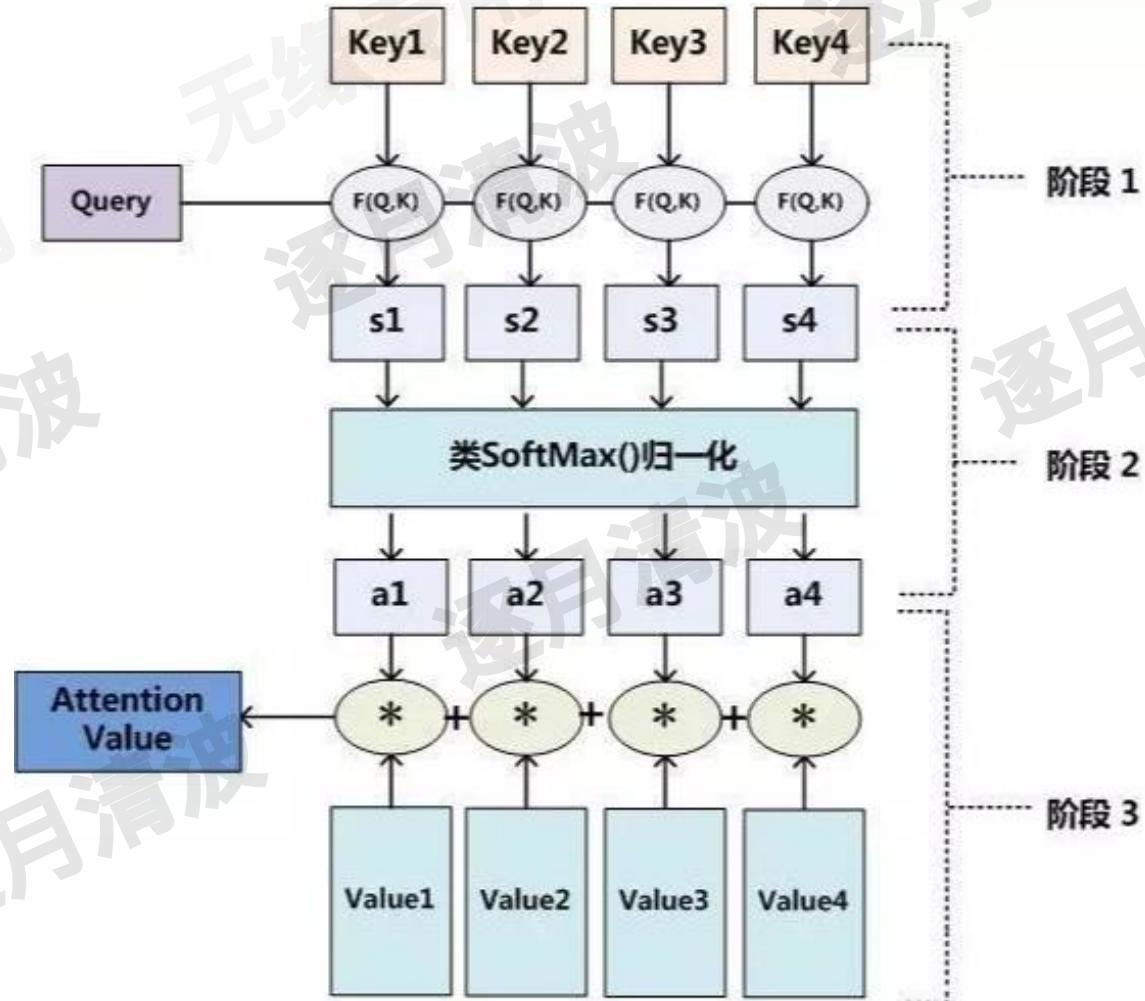
为什么要引入注意力机制呢？

1. 计算能力的限制：当要记住很多“信息”，模型就要变得更复杂，然而目前计算能力依然是限制神经网络发展的瓶颈。
2. 优化算法的限制：虽然局部连接、权重共享以及pooling等优化操作可以让神经网络变得简单一些，有效缓解模型复杂度和表达能力之间的矛盾；但是，如循环神经网络中的长距离以来问题，信息“记忆”能力并不高。

156、注意力 (Attention) 的执行流程是什么样的？(2星)

注意力机制可以分为不同的类型，如空间域、通道域、自注意力、非局部注意力等，它们的执行流程也有所不同，但一般都包括以下几个步骤：

1. 根据输入数据（如图像、文本、语音等）生成一个或多个查询 (Query)，键 (Key) 和值 (Value) 的向量表示；
2. 根据查询和键之间的相似度函数（如点积、拼接、感知机等）计算出一个注意力分数或权重矩阵；
3. 对注意力分数或权重矩阵进行归一化处理，如使用 Softmax 函数，得到一个注意力概率分布；
4. 根据注意力概率分布和值向量进行加权求和，得到一个输出向量，表示输入数据中的重要信息。



157、Transformer的提出原因及其作用是什么？(2星)

在Transformer之前，神经网络以CNN和RNN为主，这些技术所存在的问题有：

RNN：能够捕获长距离依赖信息，但是无法并行；

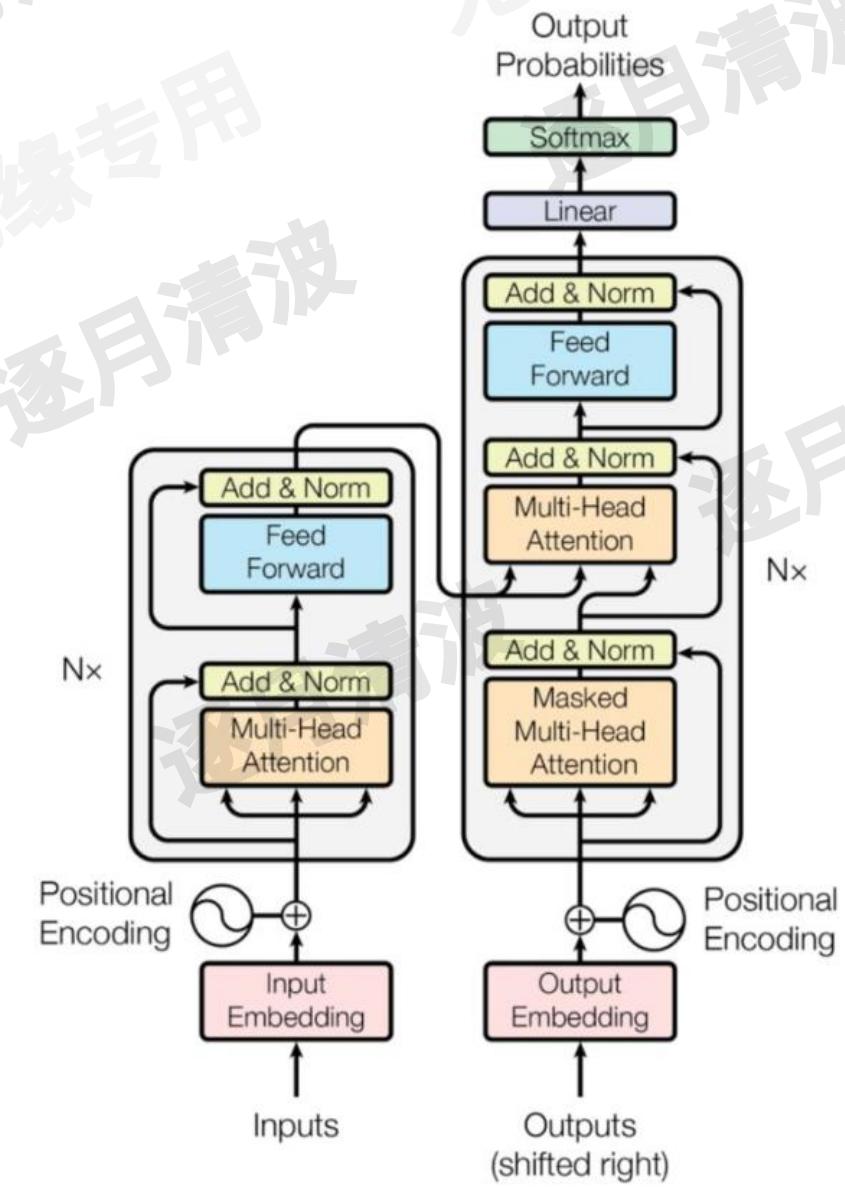
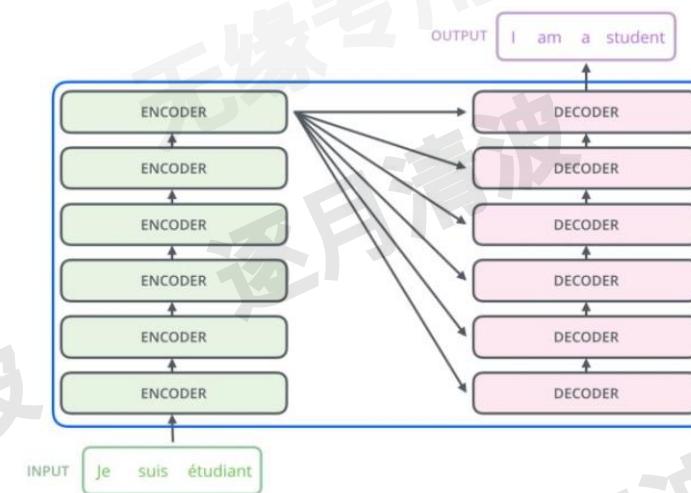
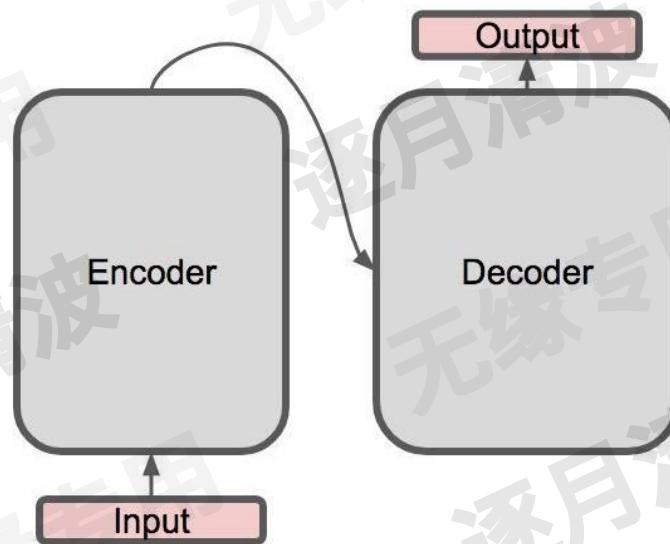
CNN：能够并行，无法捕获长距离依赖信息（需要通过层叠or扩张卷积核来增大感受野）；

而最初的Attention方法是基于源端和目标端的隐向量计算Attention，计算源端每个词与目标端每个词间的依赖关系【源端->目标端】，而忽略了远端或目标端词与词间的依赖关系

Transformer作用是什么？

基于Transformer的架构主要用于建模语言理解任务，它避免了在神经网络中使用递归，而是完全依赖于self-attention机制来绘制输入和输出之间的全局依赖关系。

158、简述Transformer的encoder和decoder结构。（4星）



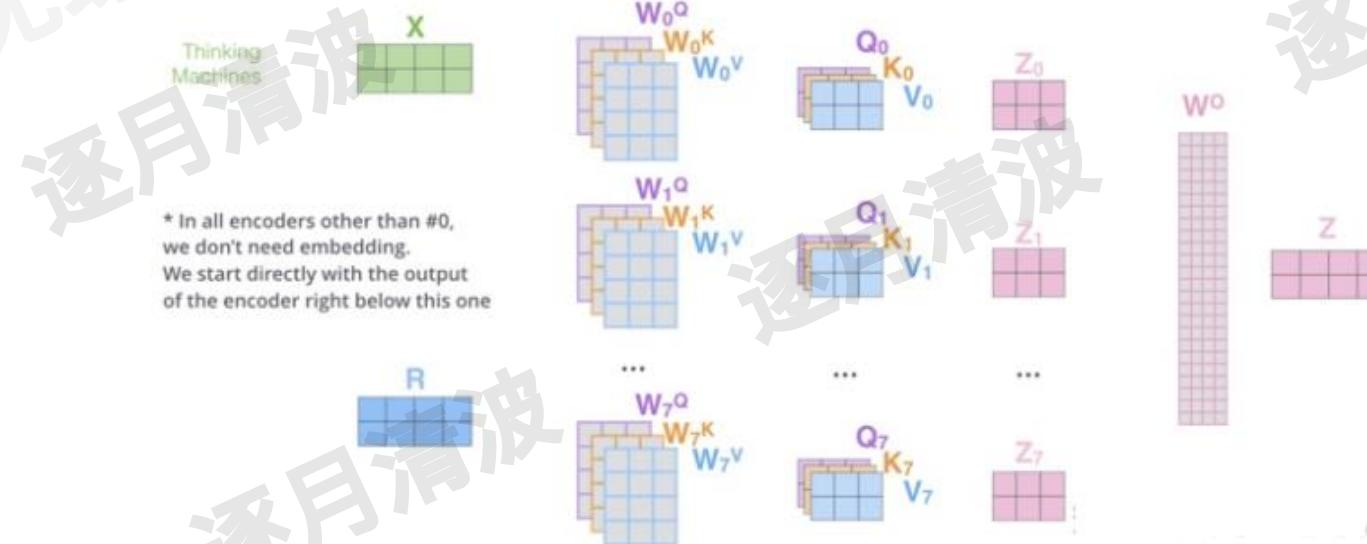
159、自注意力 (Self-attention) 的核心思想是什么？为什么会提出它？(2星)

self-attention 的核心思想在计算每个token时，总是会考虑整个序列其他token的表达；

举例：“我爱中国”这个序列，在计算“我”这个词的时候，不但会考虑词本身的embedding，也同时会考虑其他词对这个词的影响

self-attention 的目的是：学习句子内部的词依赖关系，捕获句子的内部结构。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



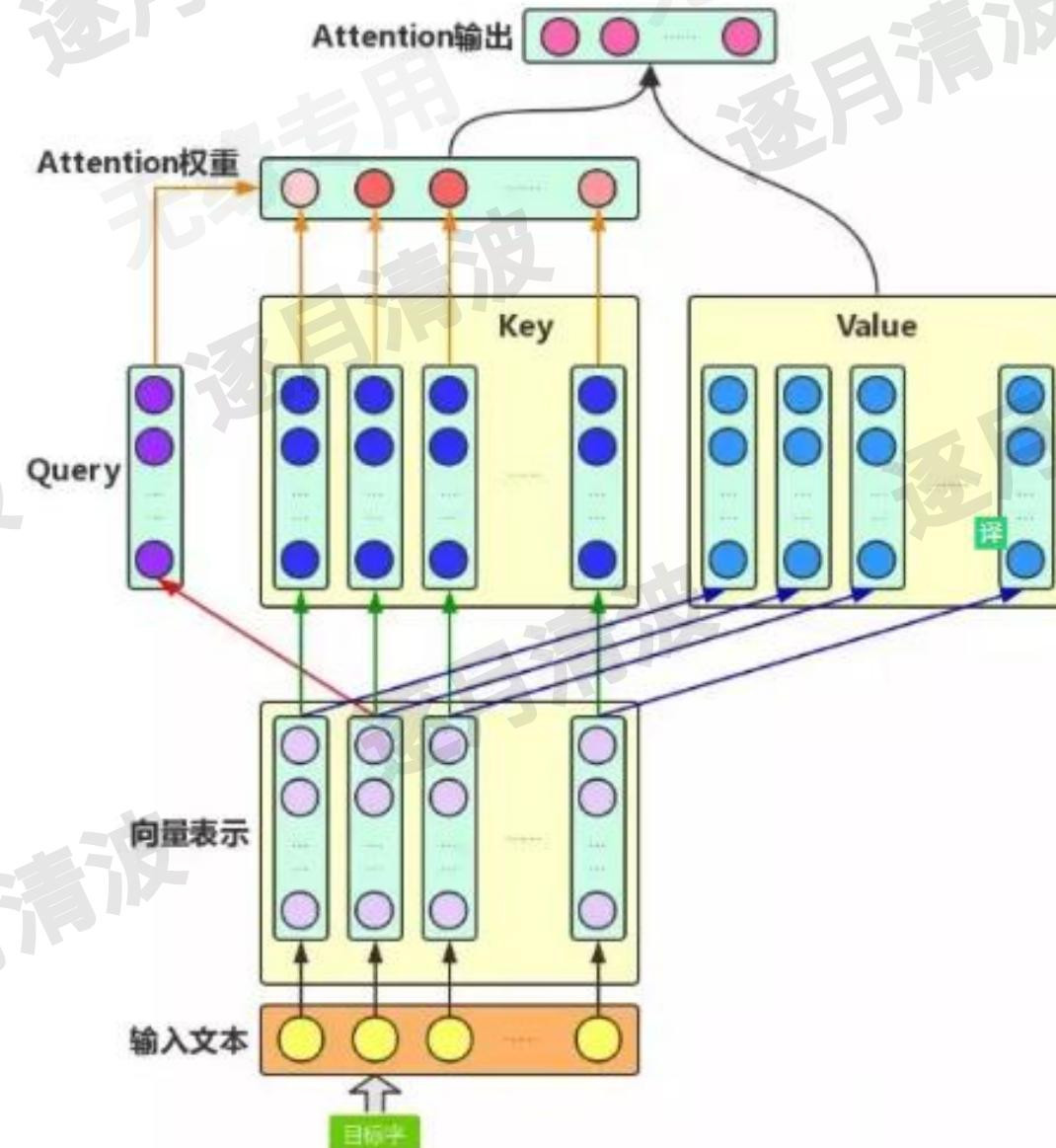
160、详述自注意力的结构。为什么Q和K使用不同的权重矩阵生成？(4星)

一句话概述：每个位置的embedding对应 Q, K, V 三个向量，这三个向量分别是embedding点乘 WQ, WK, WV 矩阵得来的（矩阵是可以学习的参数）。每个位置的Q向量乘上所有位置的K向量，其结果经过softmax变成attention score，以此作为权重对所有V向量做加权求和即可。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Q、K、V的不同，可以保证在不同空间进行投影，增强了表达能力，提高了泛化能力。

<https://zhuanlan.zhihu.com/p/410776234>



161、为什么Self-attention采用点积模型，而不采用加性模型？（2星）

主要原因是：在理论上，加性模型和点积模型的复杂度差不多，但是点积模型在实现上可以更好地利用矩阵乘法，而矩阵乘法有很多加速策略，因此能加速训练。

但是有论文的实验表明，当维度越来越大时，加性模型的效果会略优于点积模型，原因应该是加性模型整体上还是比点积模型更复杂（有非线性因素）。

162、Self-attention 如何解决长距离依赖问题？(2星)

对于序列问题，第 t 时刻的输出 依赖于 t 之前的输入，当间隔 k 逐渐增大时， $x_{\{t-k\}}$ 的信息将难以被 y_t 所学习到，也就是说，很难建立这种长距离依赖关系。在传统的网络架构中，如果两个元素之间的距离非常遥远，信息很难传递，整个模型的泛化能力就会降低。比如在一个很长的句子中，如果要理解一个单词，只看它周围的几个单词是远远不够的，需要考虑整个句子的语义信息。

Self-attention 利用注意力机制来“动态”地生成不同连接的权重，从而处理变长的信息序列，是解决长距离依赖问题的一种非常有效的方式。Self-attention 机制通过在同一序列中不同位置之间计算注意力权重，使得每个位置可以根据与之相关的所有其他位置来获取信息。它可以被看作一种“soft”排序，根据序列中元素之间的相似度对这些元素进行加权求和。

这种方式可以捕捉到所有其他位置对于某个位置的重要性，因此能够支持模型在处理输入的时候，不仅考虑到周围的少数几个单词，而是能够同时感知整个句子中的信息，从而支持模型学习更长期和更复杂的依赖关系，提高模型的泛化性能。

163、Self-attention 如何并行化？(1星)

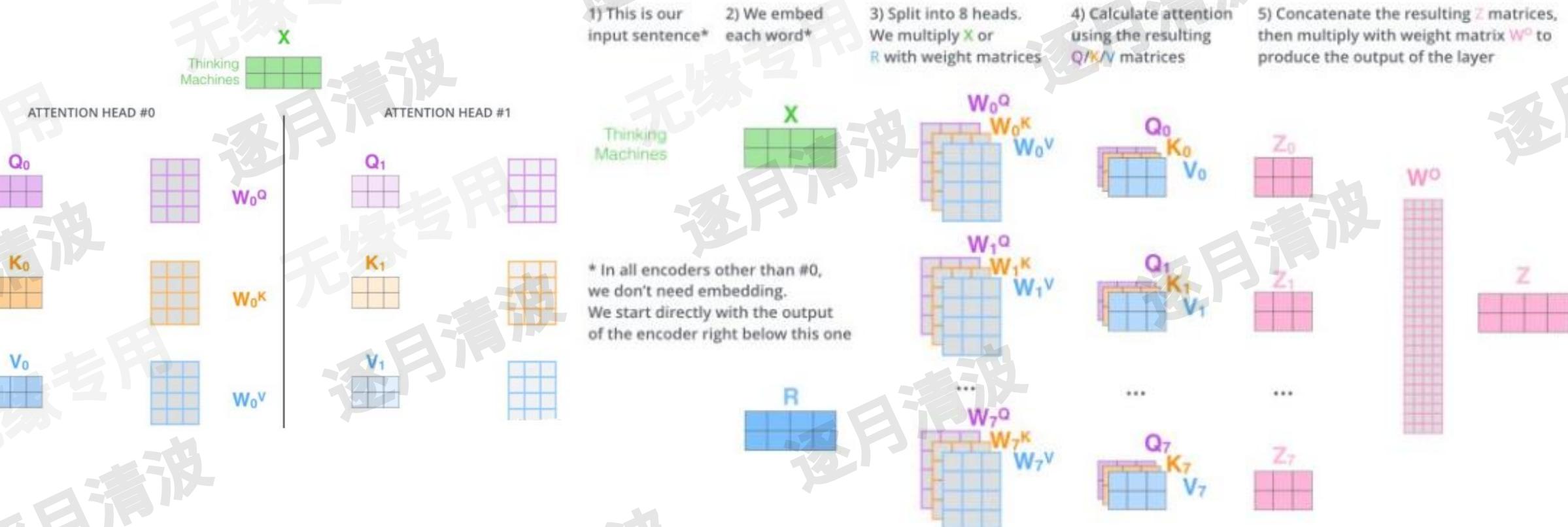
Self-Attention并行地计算 句子中不同的 query，因为每个 query 之间并不存在先后依赖关系。

Self-Attention的计算其实都是矩阵运算，因此可以使用GPU加速，使得它能够并行化。

1. 输入为 I
2. $Q = W^q I, K = W^k I, V = W^v I$
3. $A = K^T Q$
4. $\hat{A} = \text{softmax}(A)$
5. 输出为 $O = V \hat{A}$

164、多头注意力机制 (Multi-head attention) 的思路与步骤是什么？(3星)

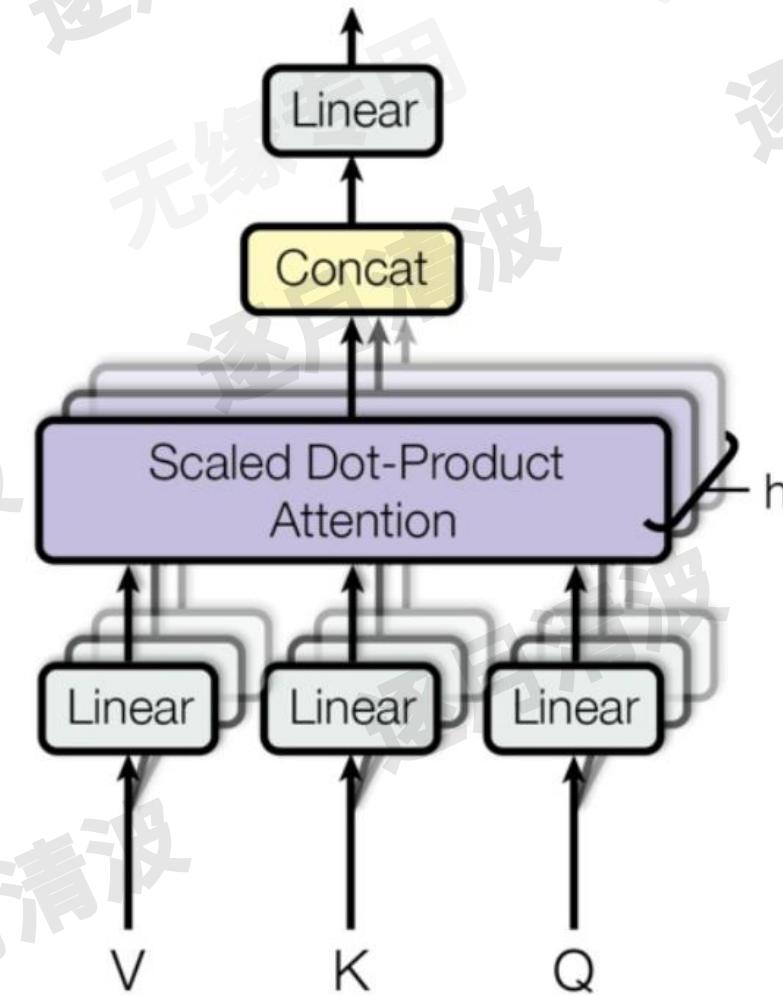
多头注意力机制 (Multi-head attention) 的思路相当于n个不同的 self-attention 的集成，就是把self-attention做n次，取决于 head 的个数。



165、为什么Transformer要用多头？(2星)

Transformer使用多头注意力机制的目的是为了
让Transformer能够注意到不同子空间的信息，
从而捕获到跟多的特征信息。其本质也是实验
定律，即在工程中有效是最终的目的。

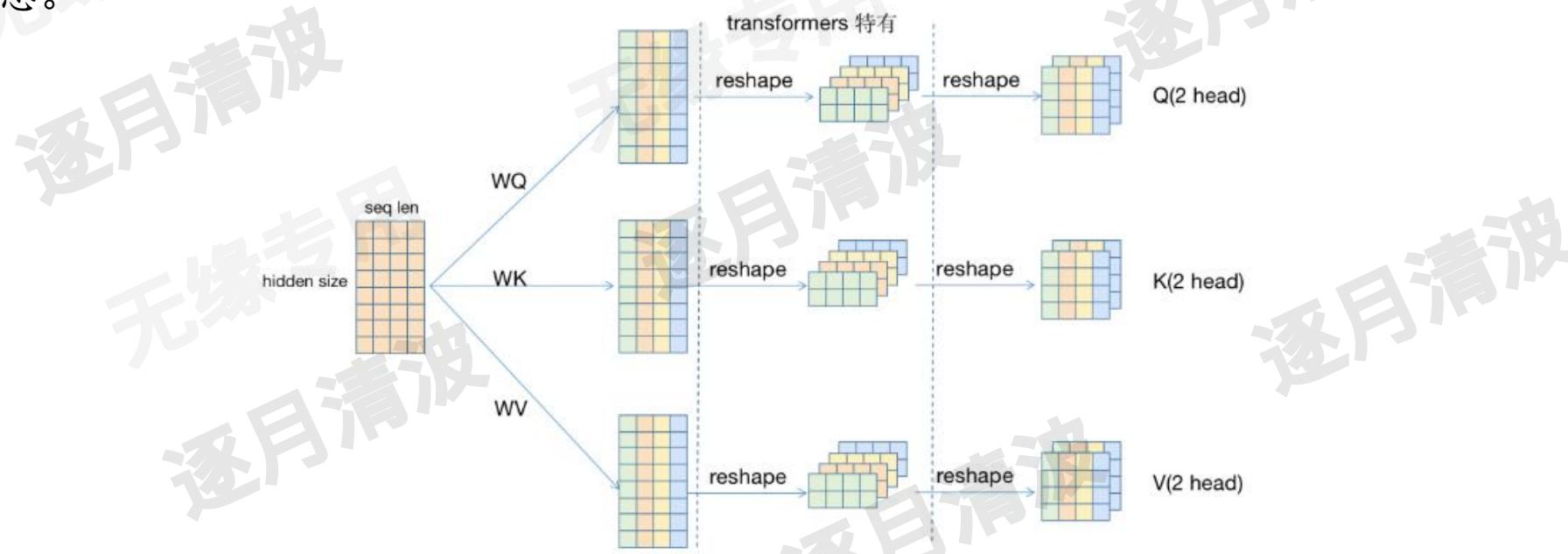
多头机制类似于CNN中通过多通道机制进行特征
选择，Transformer中使用切头(split)的方法，
是为了在不增加复杂度($O(n^2 d)$)的前提下
享受类似CNN中“不同卷积核”的优势。



166、Transformer多头注意力需要注意什么？(2星)

Transformer进行多头注意力的时候需要对每个head进行降维（切割）。

Transformer的多头注意力看上去是借鉴了CNN中同一卷积层内使用多个卷积核的思想，原文中使用了8个“scaled dot-product attention”，在同一“multi-head attention”层中，输入均为“KQV”，同时进行注意力的计算，彼此之前参数不共享，最终将结果拼接起来，这样可以允许模型在不同的表示子空间里学习到相关的信息。



在此之前的 A Structured Self-attentive Sentence Embedding 也有着类似的思想。简而言之，就是希望每个注意力头，只关注最终输出序列中一个子空间，互相独立。其核心思想在于，抽取到更加丰富的特征信息。

167、介绍一下位置编码 (Positional Encoding) 的思路、作用与步骤。 (3星)

为什么要加入位置编码 (Position encoding) ?

因为attention缺乏一种表示输入序列中单词顺序的方法。

模型不包括Recurrence/Convolution, 因此是无法捕捉到序列顺序信息的, 例如将K、V按行进行打乱, 那么Attention之后的结果是一样的。但是序列信息非常重要, 代表着全局的结构, 因此必须将序列的分词相对或者绝对position信息利用起来, 所以需要加入词序信息, 使Attention能够分辨出不同位置的词。

思路:

在encoder层和decoder层的输入添加了一个额外的向量

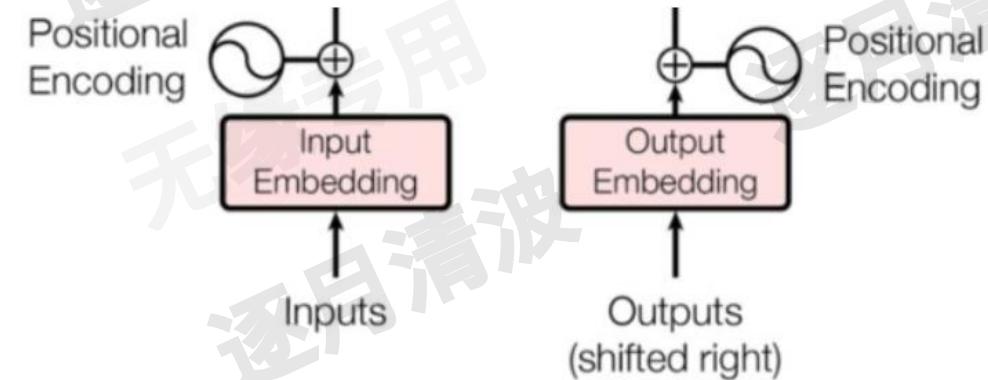
Positional Encoding, 维度和embedding的维度一样, 让模型学习到这个值。

作用:

决定当前词的位置; 计算在一个句子中不同的词之间的距离

步骤:

将每个位置编号, 然后每个编号对应一个向量, 通过将位置向量和词向量相加, 就给每个词都引入了一定的位置信息。



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

168、Position encoding和Position embedding有什么区别？(3星)

Position embedding大意是 (x_0, x_1, x_2, \dots) token序列经过embedding矩阵变成 (w_0, w_1, w_2, \dots) 的词向量序列，同时 $(0, 1, 2, \dots)$ 的绝对位置序列也经过另一个embedding矩阵变为 (p_0, p_1, p_2, \dots) 。最终的embedding序列就是 $e = (x_0+p_0, x_1+p_1, x_2+p_2, \dots)$ 。**Position embedding**矩阵靠训练学习获得，而**Position Encoding**用三角函数计算直接获得。

Position encoding构造简单直接无需额外的学习参数；能兼容预训练阶段的最大文本长度和训练阶段的最大文本长度不一致；

Position embedding需要额外的学习参数；训练阶段的最大文本长度不能超过预训练阶段的最大文本长度（因为没学过这么长的，不知道如何表示）；但是Position embedding的潜力在直觉上会比Position encoding大，因为毕竟是自己学出来的，只有自己才知道自己想要什么（前提是数据量得足够大）。

169、为什么提出Transformer时采用的是 Position Encoder，而Bert却采用的是 Position Embedding ? (2星)

事实上，Transformer 的作者在论文中对比了 Position Encoding 和 Position Embedding，在模型精度上没有明显区别。出于对序列长度限制和参数量规模的考虑，最终选择了 Encoding 的形式。

那么为什么Bert不这么干呢？主要原因如下：

1. 模型的结构需要服务于模型的目标：Transformer最早提出是针对机器翻译任务的，而机器翻译任务对词序特征要求不高，因此在效果差不多的情况下选择Position Encoder 足矣。但是Bert是作为通用的预训练模型，下游任务有很多对词序特征要求很高，因此选择潜力比较大的Position Embedding会更好；
2. 数据量的角度：Transformer用的数据量没有Bert的数据量大，所以使用潜力无限的 Position Embedding 会有大力出奇迹的效果；

170、transformer模型在时间序列上表现如何？(2星)

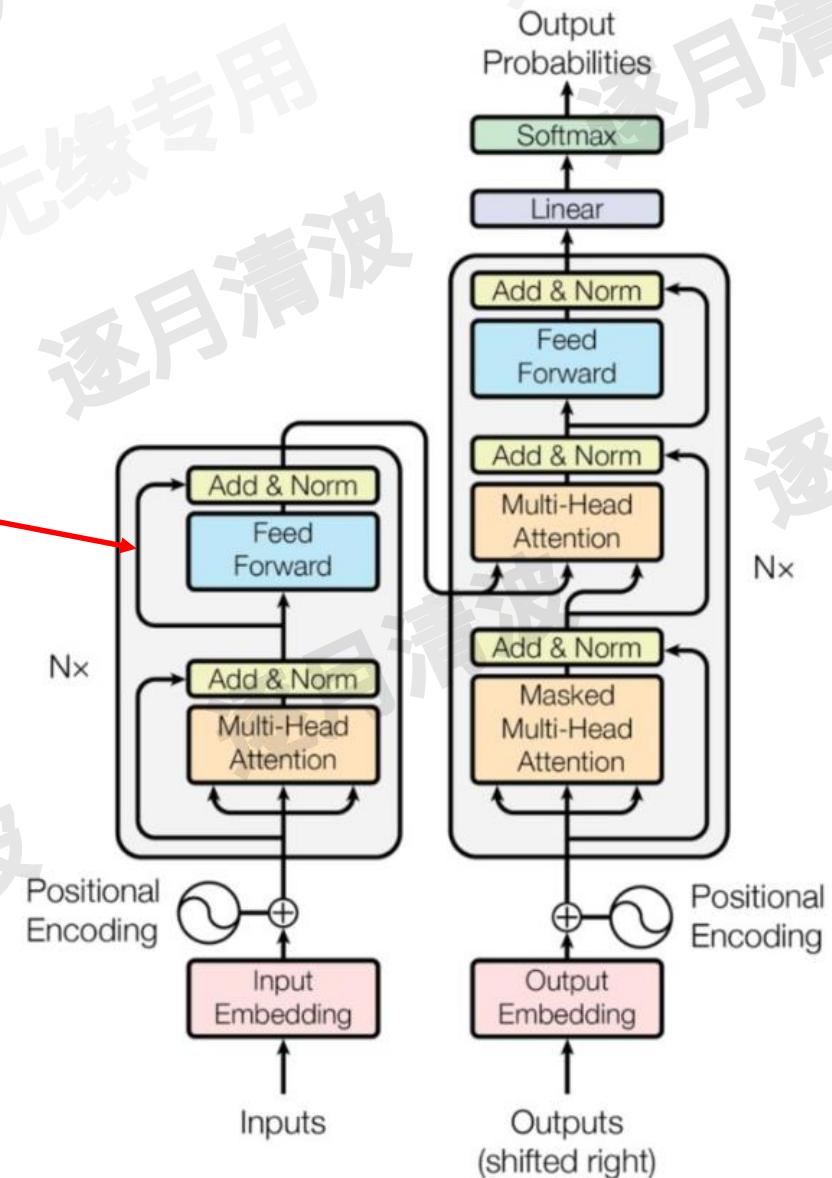
Transformer模型通过引入自注意力机制（self-attention mechanism）和位置编码（Positional Encoding），将序列中任意两个时间步长之间的关系捕捉到特征中，因此在处理时间序列上有着先天优势。此外，由于它们是并行计算的，从而能够加快训练和推理时间，使其在大规模时间序列数据中表现非常出色。因此，Transformer模型在时间序列预测等领域中已经得到了广泛的应用。

- Transformer模型可以捕捉长期依赖和彼此交互的突出能力，对于时间序列建模特别有吸引力，能在各种时间序列应用程序中取得令人兴奋的进展。
- Transformer模型需要针对时间序列的特点进行一些改造，主要包括增加位置编码，注意力模块修改和架构层面创新。
- Transformer模型可以应用于预测，异常检测和分类等多种时间序列任务，有许多最新的研究工作展示了其优异的效果和创新的设计。
- 举例(相关论文)：Autoformer、Pyraformer、Informer等

171、什么是残差模块（Residual），为什么要加入残差模块？（3星）

因为transformer堆叠了很多层，容易梯度消失或者梯度爆炸。残差模块（Residual）是一种卷积神经网络的结构，它可以通过在网络的某些层添加一个跳跃连接（skip connection）来实现对输入特征的直接反馈，从而提高网络的性能和深度。

残差模块的基本思想是，如果一个深层网络可以拟合出一个复杂的函数关系，那么它也应该可以拟合出一个恒等映射（identity function），即输出等于输入。这样，网络就可以通过学习残差（residual），即目标函数与恒等映射之间的差异，来优化网络的参数。残差部分不会经过梯度的连乘，不容易引起梯度消失或爆炸。



172、解释一下Transformer Decoder中用到的Mask。（3星）

Mask用于decoder的self-attention 中，其目的是为了让decoder更好地进行预测工作，避免作弊。

sequence mask是为了使得decoder不能看见未来的信息。也就是对于一个序列，time_step为t的时刻，我们的解码输出应该只能依赖于t时刻之前的输出，而不能依赖t之后的输出。因此我们需要想一个办法，把t之后的信息给隐藏起来。

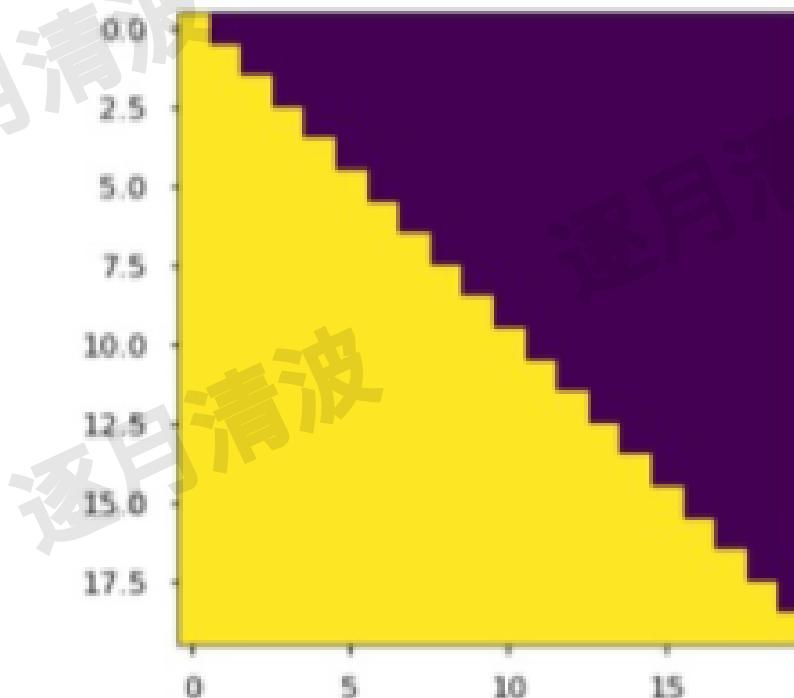
具体做法是，产生一个下三角矩阵，上三角的值全为0，下三角全是1。把这个矩阵作用在每一个序列上，就可以达到我们的目的。

$$XW_i^Q = Q_i, \quad XW_i^K = K_i, \quad XW_i^V = V_i, \quad i = 1, \dots, 8$$

$$Z_i = \text{Attention}(Q_i, K_i, V_i)_i = \text{softmax}(\text{Mask}(\frac{Q_i K_i^T}{\sqrt{d_k}}))V_i, \quad i = 1, \dots, 8$$

$$Z = \text{MultiHead}(Q, K, V) = \text{Concat}(Z_1, \dots, Z_8)W^O$$

$$Z = \text{LayerNorm}(X + Z)$$

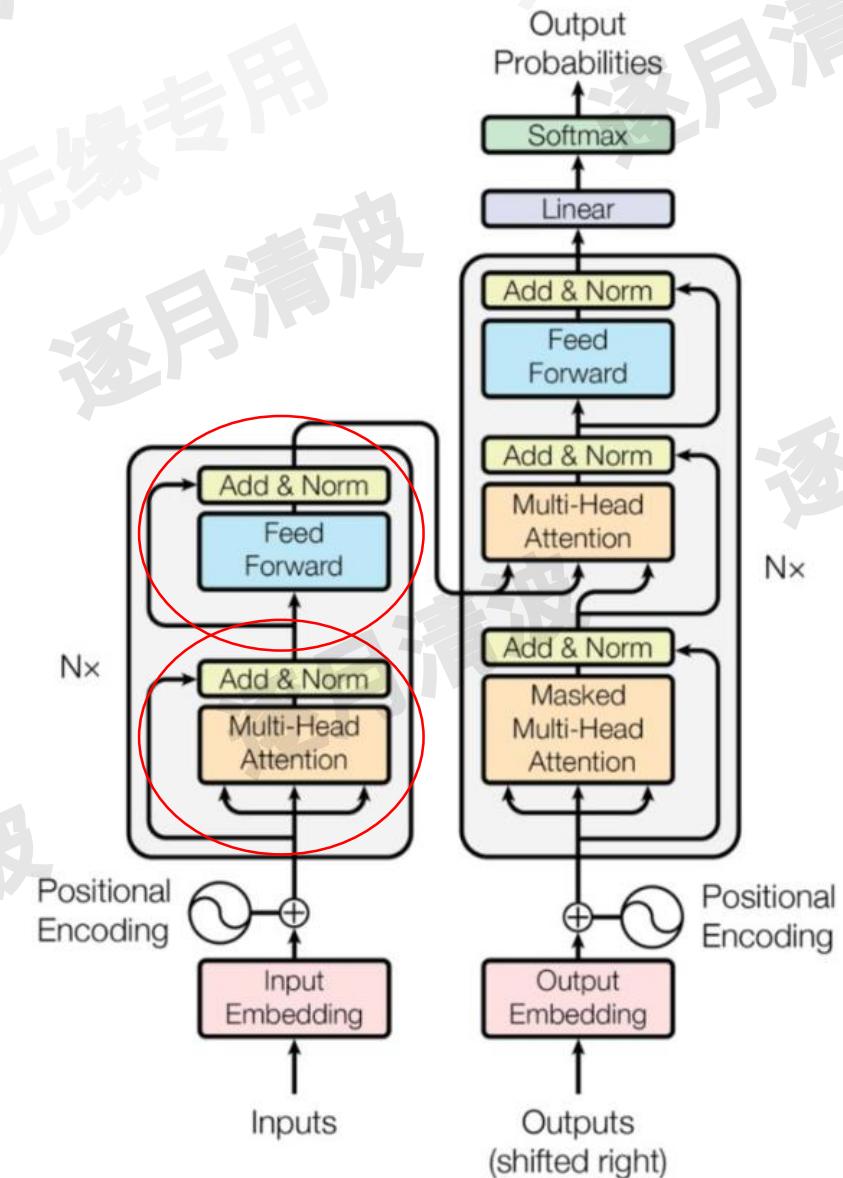


173、Transformer的非线性来自于哪里？(3星)

一、FFN中的激活函数

二、self-attention，注意self-attention是非线性的（softmax）。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



174、简要介绍一下Transformer存在的问题？（2星）

- Transformer不能很好的处理超长输入问题，例如在Bert 里面，输入句子的默认长度为512
- Transformer方向信息以及相对位置的缺失问题，Position Encoding和embedding相较于纯位置编码来说互有优劣。
- Transformer缺少conditional computation；transformer在encoder的过程中，所有输入元素都有相同的计算量，比如对于“*I arrived at the bank after crossing the river*”，和"river"相比，需要更多的背景知识来推断单词"bank"的含义，然而transformer在编码这个句子的时候，无条件对于每个单词应用相同的计算量，这样的过程显然是低效的。
- Transformer时间复杂度和空间复杂度过大问题；Transformer中用到的自注意力与长度n呈现出 $O(n^2)$ 的时间和空间复杂度

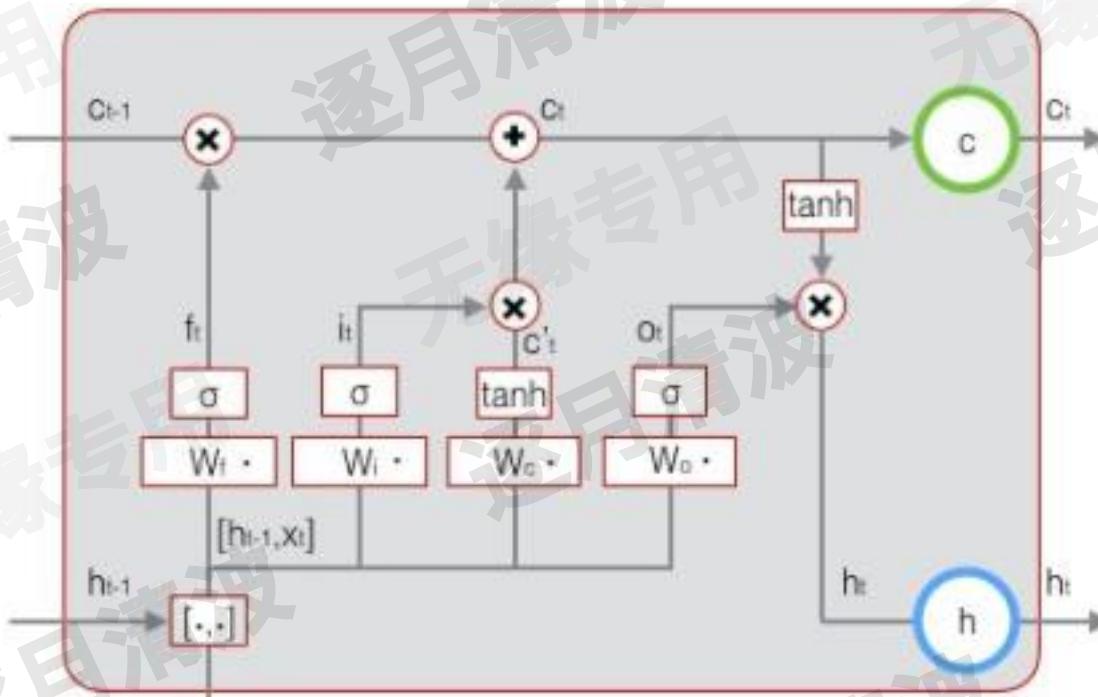
175、LSTM如何缓解RNN中的梯度消失和梯度爆炸问题？(3星)

引入门控机制。门实际上就是一层全连接层，它的输入是一个向量，输出是一个0到1之间的实数向量。

遗忘门：控制继续保存长期状态c；

输入门：控制把即时状态输入到长期状态c；

输出门：控制是否把长期状态c作为当前的LSTM的输出；



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \circ \tanh(c_t)$$

176、描述一下LSTM的激活函数与复杂度。 (3星)

LSTM中，门控的激活函数为sigmoid，输出的激活函数为tanh函数。这两个激活函数都是饱和的。也就是说在输入达到一定值的情况下，输出就不会发生明显变化了。如果是用非饱和的激活函数，例如ReLU，那么将难以实现门控的效果。

Sigmoid的输出在0-1之间，符合门控的物理定义，且当输入较大或较小时，其输出会非常接近1或0，从而保证该门开或关，在生成候选记忆（输出）时，使用tanh函数，是因为其输出在-1到1之间，这与大多数场景下特征分布是0中心的吻合。此外，tanh函数在输入为0附近相比Sigmoid函数有更大的梯度，通常使模型收敛更快。

所以现代的LSTM采用 Sigmoid和tanh作为激活函数。事实上在门控中，使用Sigmoid函数是几乎所有现代神经网络模块的共同选择。例如在门控循环单元和注意力机制中，也广泛使用Sigmoid函数作为门控的激活函数。

假设序列长度为T，隐藏层维度为H，则LSTM的复杂度为 $O(T * H^2)$ 。

177、one-hot编码是什么？有什么特点？（2星）

为什么有one-hot？

由于计算机无法识别文本语言，所以需要将文本数字化，**one-hot方法最早的一种将文本数字化的方法。**

one-hot是什么？

用一个很长的向量来表示一个词，向量长度为词典的大小N，**每个向量只有一个维度为1，其余维度全部为0，**为1的位置表示该词语在词典的位置。

one-hot有什么特点？

维度长：向量的维度为词典大小；

一一其零：每个向量只有一个维度为1，其余维度全部为0，为1的位置表示该词语在词典的位置；

178、one-hot编码存在哪些问题？（3星）

One-Hot编码存在的问题有：

1. **维度灾难**: 容易受维数灾难的困扰，每个词语的维度就是语料库字典的长度；
2. **离散、稀疏问题**: 因为 one-Hot 中，句子向量，如果词出现则为1，没出现则为0，但是由于维度远大于句子长度，所以句子中的1远小于0的个数；
3. **维度鸿沟问题**: 词语的编码往往是随机的，导致不能很好地刻画词与词之间的相似性。例如，两个近义词的含义非常相似，但在词典中出现的位置可能相差很远，不利于采集语料信息。

179、什么是ROC曲线？如何计算AUC？（3星）

ROC曲线是*Receiver Operating Characteristic Curve*的简称，中文名为“受试者工作特征曲线”。ROC曲线源于军事领域，而后在医学领域应用甚广，“受试者工作特征曲线”这一名称也正是来自于医学领域。

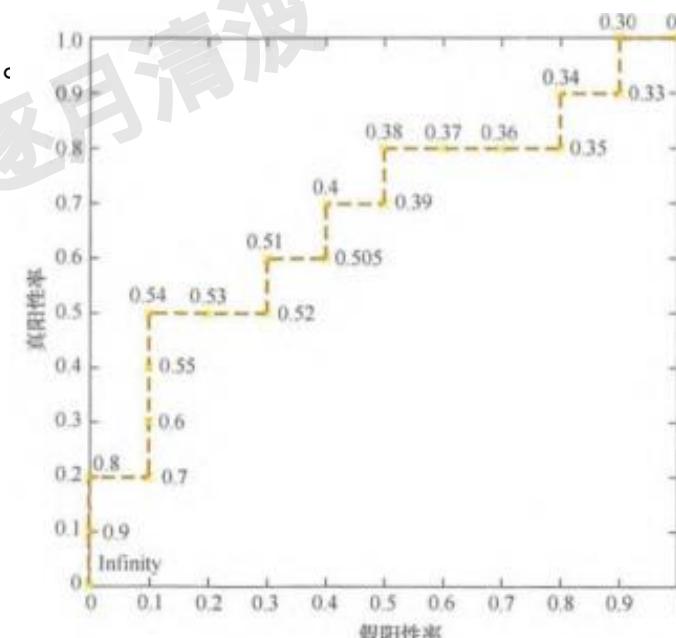
ROC曲线的横坐标为假阳性率 (False Positive Rate, FPR)，错误预测为正例的个数/所有负例个数；纵坐标为真阳性率 (True Positive Rate, TPR)，正确预测为正例的个数/所有正例个数。

AUC指的是ROC曲线下的面积大小，该值能够量化地反映基于1100 曲线衡量出的模型性能。计算AUC值只需要沿着ROC横轴做积分就可以了。由于ROC曲线一般都处于 $y=x$ 这条直线的上方（如果不是的话，只要把模型预测的概率反转成 $1-p$ 就可以得到一个更好的分类器），所以AUC的取值一般在0.5-1之间。

- $AUC = 1$, 完美分类器，采用这个预测模型时，存在至少一个阈值能完美预测。
- $0.5 < AUC < 1$, 优于随机猜测。这个分类器妥善设定阈值的话有预测价值。
- $AUC = 0.5$, 跟随机猜测一样（例：丢铜板），模型没有预测价值。
- $AUC < 0.5$, 比随机猜测还差；但只要总是反预测而行，就优于随机猜测。

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$



180、逻辑回归与线性回归相比有何异同？(2星)

逻辑回归和线性回归的不同之处：逻辑回归处理的是分类问题，线性回归处理的是回归问题，这是两者最本质的区别。逻辑回归中，因变量取值是一个二元分布，模型学习得出的是 $E[y|x, \theta]$ ，即给定自变量和超参数后，得到因变量的期望，并基于此期望来处理预测分类问题。而线性回归中实际上求解的是 $y' = \theta x$ ，是对我们假设的真实关系 $y = \theta x + \varepsilon$ 的一个近似，其中 ε 代表误差项，我们使用这个近似项来处理回归问题。

逻辑回归和线性回归的相同之处：首先我们可以认为二者都使用了极大似然估计来对训练样本进行建模。线性回归使用最小二乘法，实际上就是在自变量 x 与超参数 θ 确定，因变量 y 服从正态分布的假设下，使用极大似然估计的一个化简；而逻辑回归中通过对似然函数的学习，得到最佳参数 θ 。

另外，二者在求解超参数的过程中，都可以使用梯度下降的方法，这也是监督学习中一个常见的相似之处。

181、K-均值聚类的缺点是什么？针对它的缺点，如何改进模型？(3星)

K均值聚类算法的主要缺点如下：

1. 需要人工预先确定初始K值，且该值和真实的数据分布未必吻合。
2. K均值只能收敛到局部最优，效果受到初始值很大。
3. 易受到噪点的影响。
4. 样本点只能被划分到单一的类中。

K均值的改进算法中，对初始值选择的改进是很重要的一部分。例如K-means++ 算法。原始K均值算法最开始随机选取数据集中K个点作为聚类中心，而K-means++按照如下的思想选取聚类中心：

假设已经选取了n个初始聚类中心($0 < n < K$)，则在选取第n+1个聚类中心时，距离当前n个聚类中心越远的点会有更高的概率被选为第n+1个聚类中心。在选取第一个聚类中心时同样通过随机的方法。可以说这也符合我们的直觉，聚类中心当然是互相离得越远越好。当选择完初始点后，后续的执行和经典K均值算法相同，这也是对初始值选择进行改进的方法的共同点。

182、什么是ISODATA算法？它有哪些参数？（2星）

ISODATA的全称是迭代自组织数据分析法。在K均值算法中，聚类个数K的值需要预先人为地确定，并且在整个算法过程中无法更改。而当遇到高维度、海量的数据集时，人们往往很难准确地估计出K的大小。

ISODATA算法就是针对这个问题进行了改进，它的思想也很直观。当属于某个类别的样本数过少时，把该类别去除；当属于某个类别的样本数过多、分散程度较大时，把该类别分为两个子类别。ISODATA算法在K均值算法的基础之上增加了两个操作，一是分裂操作，对应着增加聚类中心数；二是合并操作，对应着减少聚类中心数。

ISODATA需要制定4个参数：

1. 预期的聚类中心数目 K_0 。在ISODATA运行过程中聚类中心数可以变化， K_0 是一个用户指定的参考值，该算法的聚类中心数目变动范围也由其决定。最终输出的聚类中心数常见范围是从 K_0 的一半到两倍。
2. 每个类所要求的最少样本数目 N_{min} 。如果分裂后会导致某个子类别所包含样本数目小于该阈值，就不会对该类别进行分裂操作。
3. 最大方差 σ ，用于控制某个类别中样本的分散程度。当样本的分散程度超过这个阈值时，且分裂后满足条件1，进行分裂操作。
4. 两个聚类中心之间所允许最小距离 D_{min} 。如果两个类靠得非常近（即这两个类别对应聚类中心之间的距离非常小），小于该阈值时，则对这两个类进行合并操作。

183、机器学习中，哪些是凸优化问题，哪些是非凸优化问题？（2星）

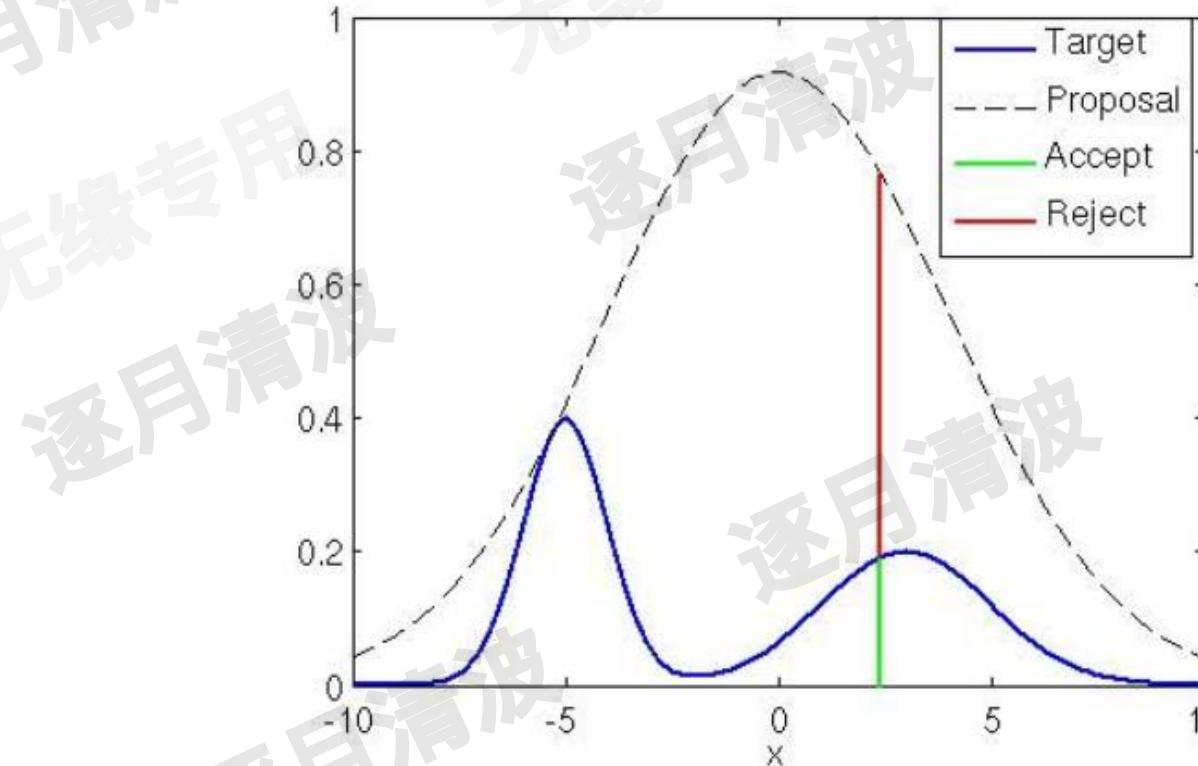
凸优化问题是指目标函数是凸的，二阶Hessian矩阵是半正定的。在机器学习中，常见的凸优化问题包括线性回归、逻辑回归、支持向量机等。其中，线性回归和逻辑回归是求解带有凸损失函数的凸优化问题，而支持向量机是求解带有凸二次约束条件的凸二次规划问题。这些凸优化问题的共同特点是目标函数和约束条件都是凸函数，所有的局部极小值都是全局极小值，因此这类问题一般认为是比较容易求解的问题。

而非凸优化问题则更加复杂，例如低秩矩阵分解、神经网络中的训练问题、深度学习中的目标函数优化等。这些问题的目标函数和约束条件都是非凸函数，通常需要用到迭代算法来求解，例如梯度下降、牛顿法、拟牛顿法等。在非凸优化问题中，常常会遇到局部最优和鞍点等问题，需要使用一些技巧来绕开这些困难。近年来，深度学习领域发展出了一些新的非凸优化算法，例如Adam、Adagrad、RMSprop等，它们对于求解深度学习中的优化问题具有很好的效果。

184、什么是拒绝采样 (Rejection Sampling) ? 简述它的具体操作。 (3星)

拒绝采样，也被称为接受采样 (Accept Sampling)，对于目标分布 $p(x)$ ，选取一个容易采样的参考分布 $q(x)$ ，使得对于任意的 x 都有： $p(x) \leq M \cdot q(x)$ 。

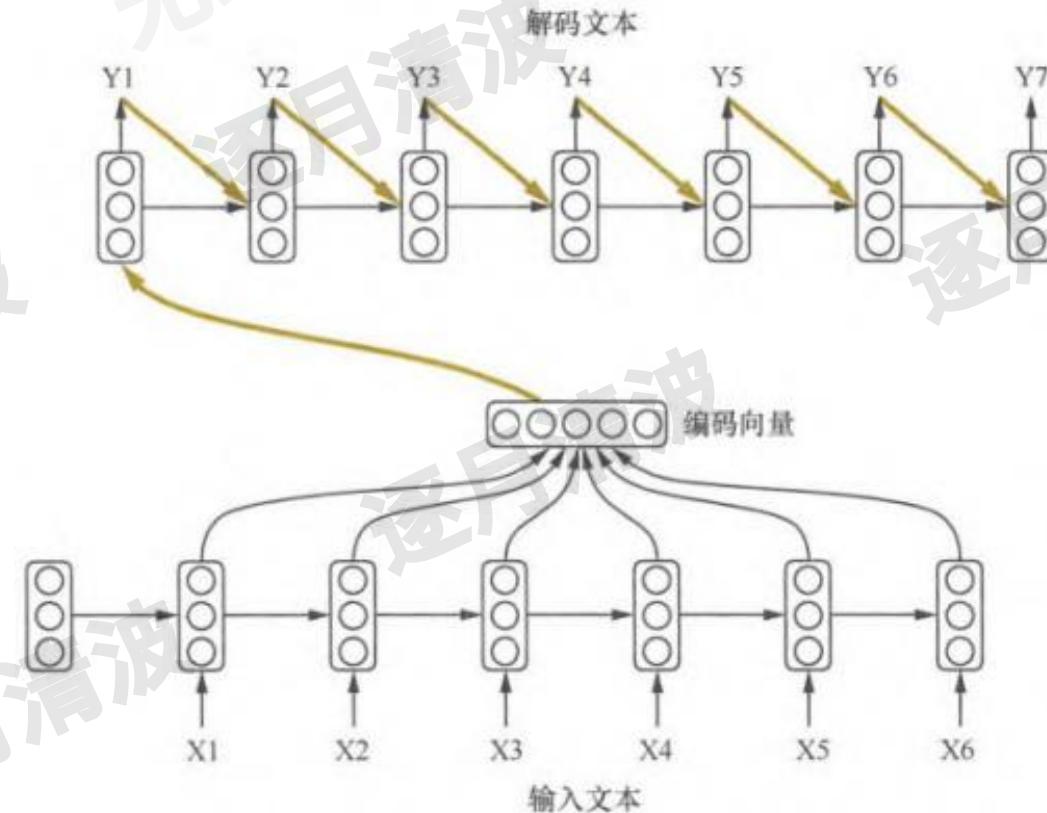
- 1) 从参考分布 $q(x)$ 中随机抽取一个样本 x_i
- 2) 从均匀分布 $U(0, 1)$ 产生一个随机 u_i
- 3) 如果 $u_i < p(x_i)/M \cdot q(x_i)$ ，则接受样本 x_i ，否则拒绝，一直重复1-3步骤，直到新产生的样本量满足要求。



185、什么是Seq2Seq模型？它有哪些优点？（3星）

Seq2Seq模型的核心思想是，通过深度神经网络将一个作为输入的序列映射为一个作为输出的序列，这一过程由编码输入与解码输出两个环节构成。在经典的实现中，编码器和解码器各由一个循环神经网络构成，既可以选择传统循环神经网络结构，也可以使用长短期记忆模型、门控循环单元等，两个循环神经网络是共同训练的。它的优点有：

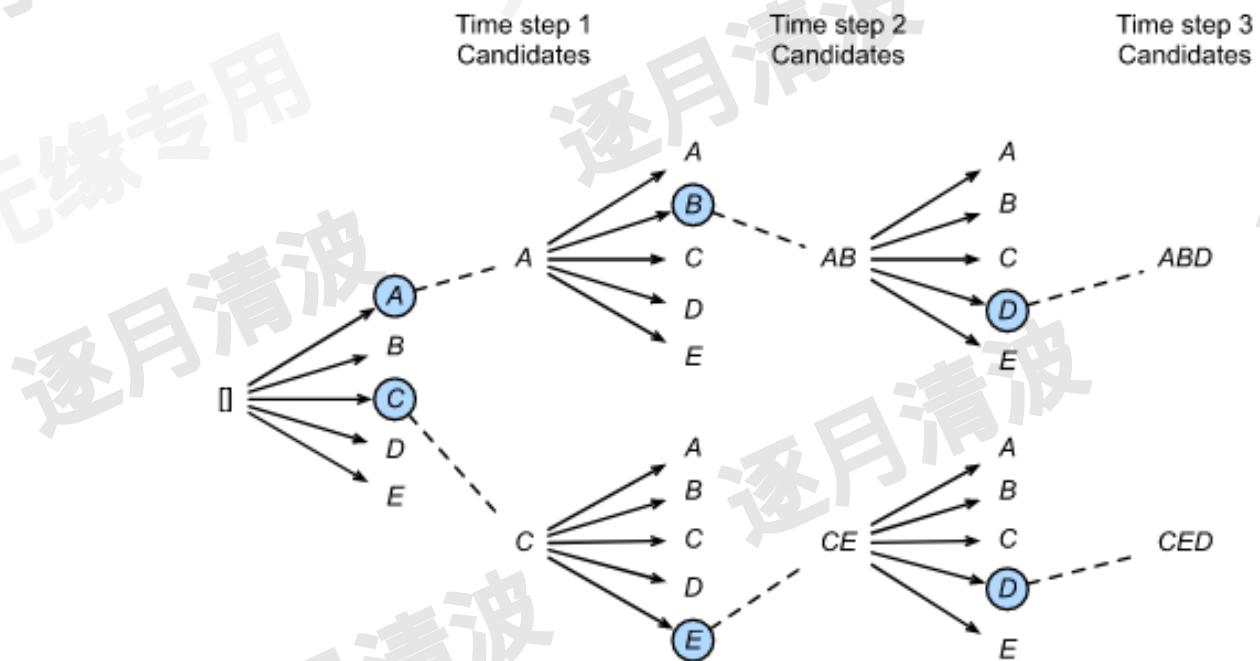
- 1、Seq2Seq的架构相对于常规的cnn或者rnn处理序列问题的效果在实际的应用中效果更好；
- 2、Seq2Seq的框架灵活可以支持不定长的输入和输出；
- 3、Seq2Seq中的很多技巧可以进一步提升模型的效果，例如attention机制、teacher forcing等；
- 4、Seq2Seq考虑了输出的标签之间的序列依赖性，这是常规的RNN或CNN无法做到的，常规的RNN和CNN只能以输出向量的形式解决多步预测问题，这意味着它们无法考虑到输出标签之间的序列依赖性。



186、什么是Seq2Seq解码时的束搜索 (Beam search) ? (3星)

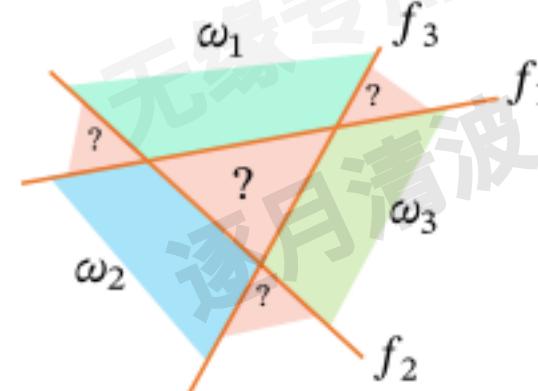
在Seq2Seq解码时，如何根据输出的概率分布来得到一个序列？如果只有准确性最重要？则显然是穷举搜索。如果计算成本最重要？则显然是贪心搜索。实际应用则介于这两个极端之间。

束搜索 (beam search) 是贪心搜索的改进版本。它有一个超参数，名为束宽 (beam size) k 。在时间步1，**我们选择具有最高条件概率的k个标记**。这 k 个标记将分别是 k 个候选输出序列的第一个标记。在随后的每个时间步，基于上一时间步的 k 个候选输出序列，我们将继续从 $k | n |$ 个可能的选择中挑出具有最高条件概率的 k 个候选输出序列， n 是词典大小。束搜索的时间复杂度是 $O(knT)$ 。

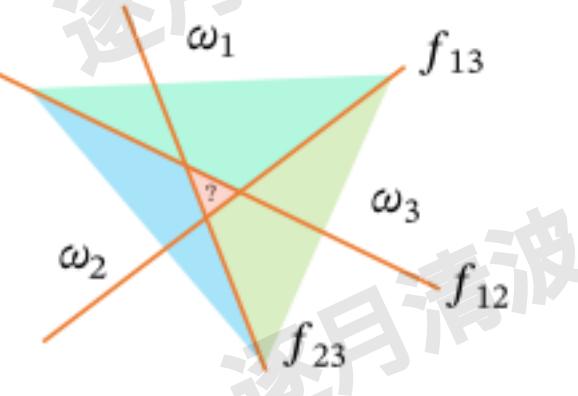


187、逻辑回归在处理多标签分类问题时，有哪些做法？(3星)

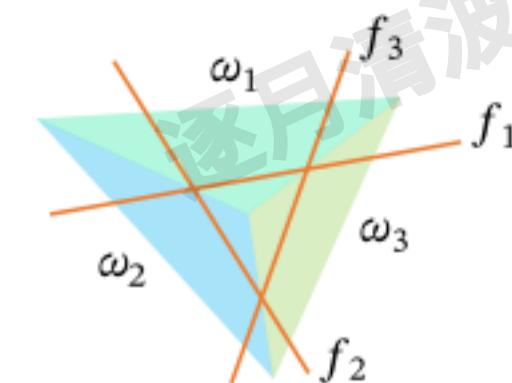
- (1) **一对多方式**: 把多分类问题转换为 C 个“一对其余”的二分类问题。这种方式共需要 C 个判别函数，其中第 c 个判别函数 f_c 是将类别 c 的样本和不属于类别 c 的样本分开。
- (2) **一对一方式**: 把多分类问题转换为 $C(C - 1)/2$ 个“一对一”的二分类问题。这种方式共需要 $C(C - 1)/2$ 个判别函数，其中第 (i, j) 个判别函数是把类别 i 和类别 j 的样本分开。
- (3) **argmax 方式**: 一种改进的“一对多”方式，共需要 C 个判别函数 $f_c(x; \mathbf{w}) = \mathbf{w}x + b$ ，对于样本 x ，如果存在一个类别 c ，相对于所有的其他类别 \sim 有 $f_c(x; \mathbf{w}_c) > f^\sim(x, \mathbf{w}^\sim)$ ，那么 x 属于类别 c 。



(a) “一对其余”方式



(b) “一对一”方式



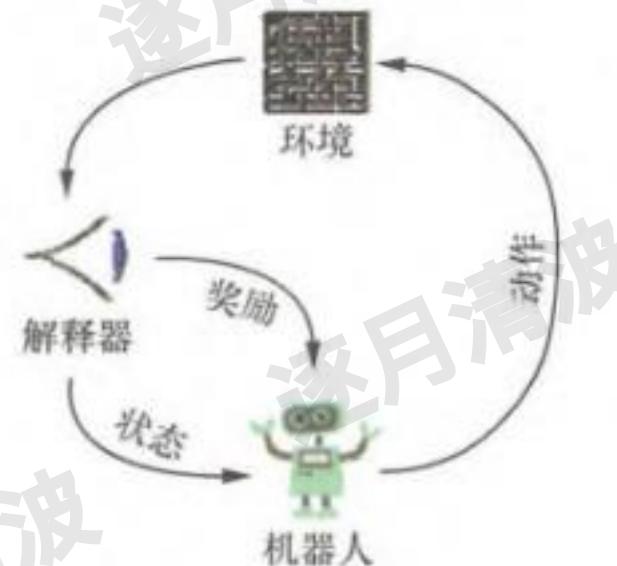
(c) “argmax”方式

188、强化学习 (Reinforcement Learning) 有哪些基本概念？(3星)

强化学习的基本场景可以用下图来描述，主要由环境 (Environment)、机器人 (Agent) 、状态 (State) 、动作 (Action) 、奖励 (Reward) 等基本概念构成。

一个机器人在环境中会做各种动作，环境会接收动作，并引起自身状态的变迁，同时给机器人以奖励。机器人的目标就是使用一些策略，做合适的动作，最大化自身的收益。

强化学习的核心任务是，学习一个从状态空间S到动作空间A的映射，最大化累积受益。常用的强化学习算法有Q-Learning、策略梯度等。



189、协方差和相关性有什么区别？(2星)

协方差和相关性都是用来衡量两个随机变量之间的关系的量。

协方差衡量的是两个变量的变化趋势是否相似，它的取值范围是正负无穷，具体的范围则由变量的单位决定。如果协方差为正，表示两个变量的变化趋势是一致的；如果协方差为负，表示两个变量的变化趋势是相反的；如果协方差为零，表示两个变量之间没有线性关系。

相关系数则是标准化的协方差，用来衡量两个变量之间的线性关系的强弱，取值范围是-1到1之间。如果相关系数为1，表示两个变量完全正相关；如果相关系数为-1，表示两个变量完全负相关；如果相关系数接近于0，则表示两个变量之间不存在线性关系。

因此，协方差和相关系数都可以用来衡量两个变量之间的关系，但是它们在量化和解释关系的方式上略有不同。协方差本身很难做比较。例如，如果我们计算工资（\$）和年龄（岁）的协方差，因为这两个变量有不同的度量，所以我们会得到不能做比较的不同的协方差。为了解决这个问题，计算相关性来得到一个介于-1和1之间的值，就可以忽略它们各自不同的度量。

$$\Sigma_{ij} = \text{cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = E[X_i X_j] - \mu_i \mu_j$$

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

190、从减少方差和偏差的角度解释Boosting和Bagging的原理。（4星）

Bagging能够提高弱分类器性能的原因是降低了方差，Boosting能够提升弱分类器性能的原因是降低了偏差。

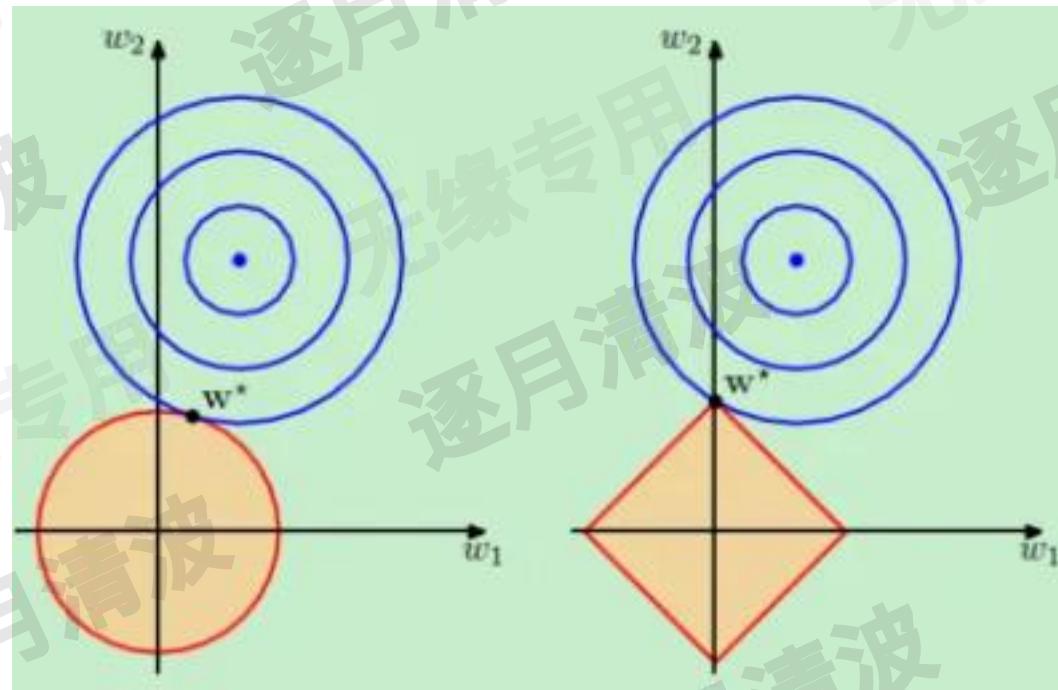
Bagging是Bootstrap Aggregating的简称，意思就是再抽样，然后在每个样本上训练出来的模型取平均。假设有n个随机变量，方差记为 σ^2 ，两两变量之间的相关性为 ρ ，则n个随机变量的均值 $\sum X_i/n$ 的方差为 $\rho \sigma^2 + (1-\rho) \sigma^2/n$ 。在随机变量完全独立的情况下，n个随机变量的方差减小为原来的 $1/n$ 。从模型的角度理解这个问题，对n个独立不相关的模型的预测结果取平均，方差是原来单个模型的 $1/n$ 。模型之间不可能完全独立。为了追求模型的独立性，诸多Bagging的方法做了不同的改进。比如在随机森林算法中，每次选取节点分裂属性时，会随机抽取一个属性子集，而不是从所有属性中选取最优属性，这就是为了避免弱分类器之间过强的相关性。通过训练集的重采样也能够带来弱分类器之间的一定独立性，从而降低 Bagging 后模型的方差。

再看Boosting的训练过程，在训练好一个弱分类器后，我们需要计算弱分类器的错误或者残差，作为下一个分类器的输入。这个过程本身就是在不断减小损失函数，来使模型不断逼近“靶心”，使得模型偏差不断降低。但 Boosting 的过程并不会显著降低方差。这是因为Boosting的训练过程使得各弱分类器之间是强相关的，缺乏独立性，所以并不会对降低方差有作用。

191、L1正则化使模型具有稀疏性的原理是什么？(3星)

在二维的情况下， 橙色的部分是L2和L1正则项约束后的解空间，蓝色的等高线是凸优化问题中目标函数的等高线，如图所示。

由图可知，L2正则项约束后的解空间是圆形，而L1正则项约束的解空间是多边形。显然，多边形的解空间更容易在尖角处与等高线碰撞出稀疏解。



$$\min_w \sum_{n=1}^N (h_w(x_i) - y_i)^2$$

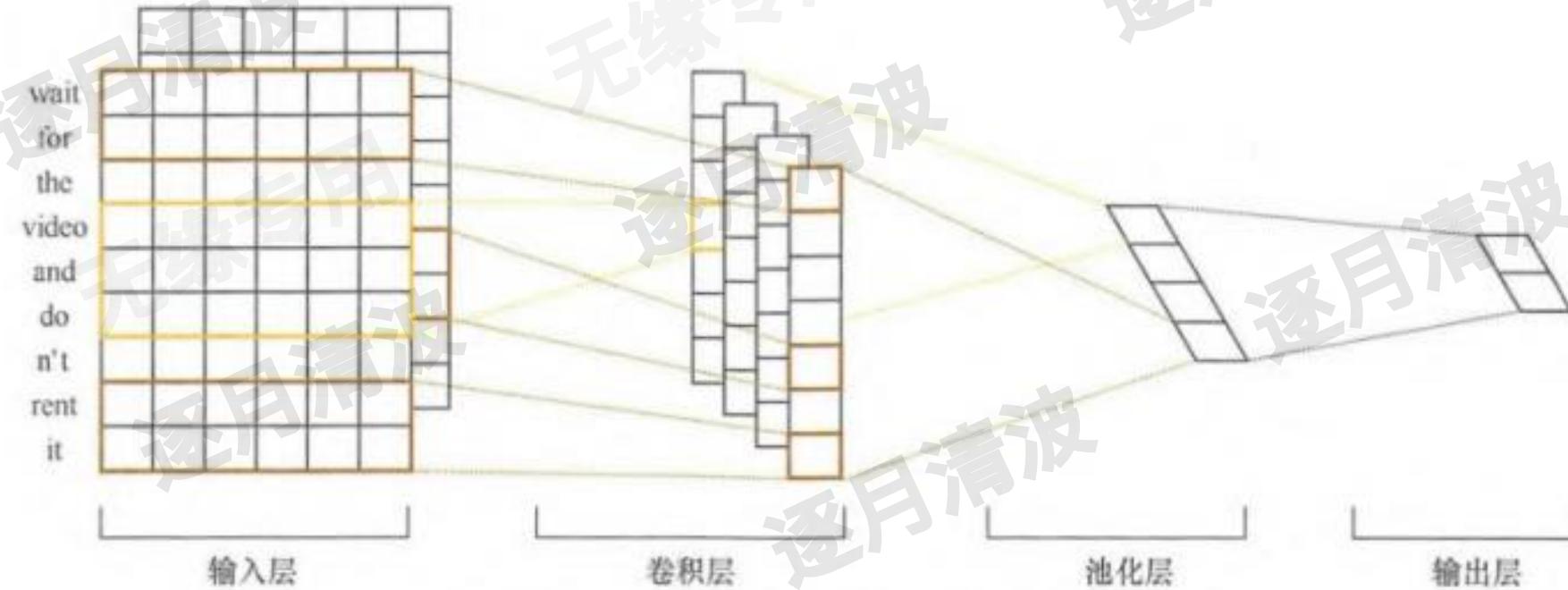
$$s.t. \quad w_1^2 + w_2^2 \leq \eta$$

$$\min_w \sum_{n=1}^N (h_w(x_i) - y_i)^2$$

$$s.t. \quad |w_1| + |w_2| \leq \eta$$

192、卷积神经网络可否用于文本分类任务？(3星)

可以。卷积神经网络的核心思想是捕捉局部特征，起初在图像领域取得了巨大的成功，后来在文本领域也得到了广泛的应用。对于文本来说，**局部特征就是由若干单词组成的滑动窗口**，类似于卷积神经网络的优势在于能够自动地对特征进行组合和筛选，**获得不同抽象层次的语义信息**。由于在每次卷积中采用了共享权重的机制，因此它的训练速度相对较快，在实际的文本分类任务中取得了非常不错的效果。



193、循环神经网络中能否使用ReLU作为激活函数？(2星)

可以，但是需要对矩阵的初值做一定限制，否则十分容易引发数值问题。

如果采取ReLU作为神经网络中的激活函数，假设ReLU一直处于激活区域（即输入大于零），则最终的表达式中会包含无数个权重连乘，最终的结果会趋向于0或无穷。

为什么在卷积神经网络中不会出现这样的现象呢？因为在卷积神经网络中每一层的权重矩阵W是不同的，并且在初始化时它们是独立同分布的，因此可以相互抵消，在多层之后一般不会出现严重的数值问题。

当采用ReLU作为循环神经网络中隐含层的激活函数时，只有当权重矩阵W的取值在单位矩阵附近时才能取得比较好的效果，因此需要将W初始化为单位矩阵。

194、神经网络训练时是否可以将全部参数初始化为0？(3星)

不可以。

考虑全连接的深度神经网络，同一层中的任意神经元都是同构的，它们拥有相同的输入和输出，如果再将参数全部初始化为同样的值，那么无论前向传播还是反向传播的取值都是完全相同的。学习过程将永远无法打破这种对称性，最终同一网络层中的各个参数仍然是相同的。

因此，我们需要随机地或采取一些分布的方法来初始化神经网络参数的值，以打破这种对称性。

195、写出L2正则化下多层感知机的平方误差和交叉熵损失函数。（3星）

假设样本集的大小为m: $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, 权重矩阵为W, 偏置为b

则 平方误差损失函数 为 $J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|y^{(i)} - \mathcal{L}_{w,b}(x^{(i)})\|^2 \right] + \frac{\lambda}{2} \sum_{l=1}^{N-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ij}^{(l)})^2$

交叉熵损失函数（二分类场景）为 $J(W, b) = - \left[\frac{1}{m} \sum_{i=1}^m \{y^{(i)} \ln o^{(i)} + (1 - y^{(i)}) \ln(1 - o^{(i)})\} \right] + \frac{\lambda}{2} \sum_{l=1}^{N-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ij}^{(l)})^2$

196、平方损失函数和交叉熵损失函数分别适合什么场景？(3星)

平方损失函数更适合输出为连续，并且最后一层不含Sigmoid或Softmax激活函数的神经网络；
交叉熵损失函数则更适合二分类与多分类的场景。

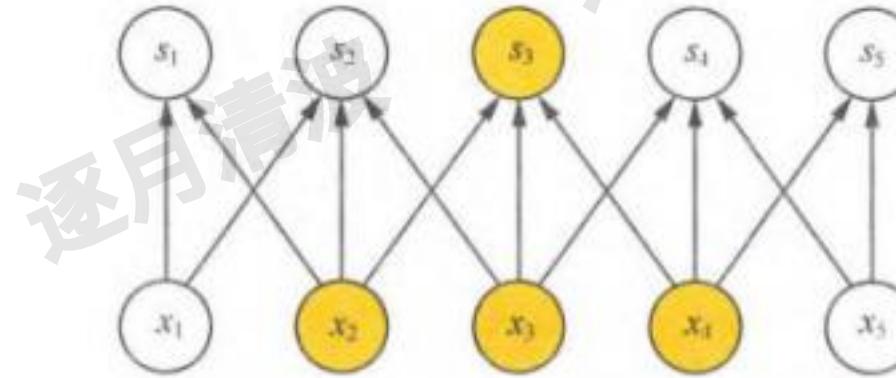
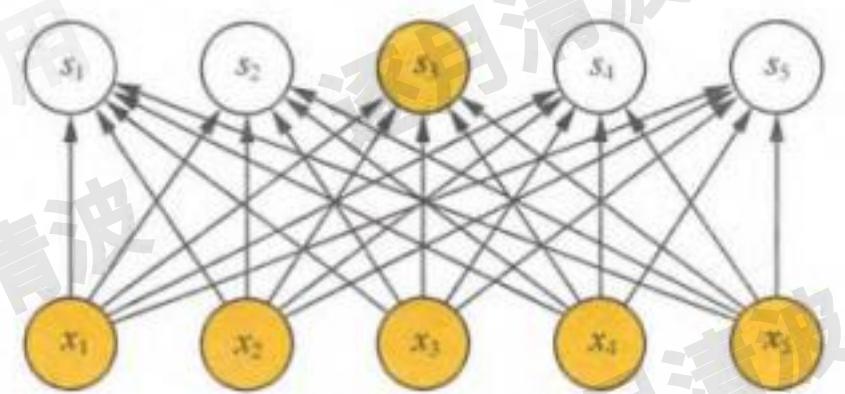
为何平方损失函数不适合最后一层含有Sigmoid或Softmax的神经网络呢？因为平方误差损失函数相对于输出层的导数会对激活函数求导，当激活函数为Sigmoid时，如果输出的绝对值较大，函数的梯度会趋于饱和，即梯度的绝对值非常小，使得基于梯度的学习速度非常缓慢。

当使用交叉熵损失函数时，相对于输出层的导数（也可以被认为是残差）是线性的，不会存在学习速度过慢的问题。

197、解释CNN中的稀疏交互和参数共享，分析它们的作用。（4星）

在传统神经网络中，网络层之间输入与输出的连接关系可以由一个权值参数矩阵来表示，其中每个单独的参数值都表示了前后层某两个神经元节点之间的交互。对于全连接网络，任意一对输入与输出神经元之间都产生交互，形成稠密的连接结构。而在卷积神经网络中，**卷积核尺度远小于输入的维度**，这样每个输出神经元仅与前一层特定局部区域内的神经元存在连接权重，这种特性称为稀疏交互。优化过程的时间复杂度将会减小几个数量级，过拟合的情况也能得到较好的改善。

参数共享是指在同一个模型的不同模块中使用相同的参数，它是卷积运算的固有属性。全连接网络中，计算每层的输出时，权值参数矩阵中的每个元素只作用于某个输入元素一次；而在卷积神经网络中，**卷积核中的每一个元素将作用于每一次局部输入的特定位置上**。根据参数共享的思想，我们**只需要学习一组参数集合**，而不需要针对每个位置的每个参数都进行优化，从而大大降低了模型的存储需求。参数共享的物理意义是使得卷积层具有平移等变性。



198、批量归一化在卷积神经网络中使用时需要注意什么？(3星)

批量归一化可以看作在每一层输入和上一层输出之间加入了一个新的计算层，对数据的分布进行额外的约束，从而增强模型的泛化能力。

批量归一化在卷积神经网络中应用时，需要注意卷积神经网络的参数共享机制。每一个卷积核的参数在不同位置的神经元当中是共享的，因此也应该被一起归一化。

具体实现中，假设网络训练中每一批包含 b 个样本，由一个卷积核生成的特征图的宽高分别为 w 和 h ，则每个特征图所对应的全部神经元个数为 bwh 。利用这些神经元对应的所有输入数据，我们根据一组待学习的参数对每个输入数据进行批量归一化操作。如果有 k 个卷积核，就对应 k 个特征图和 k 组不同的归一化参数。

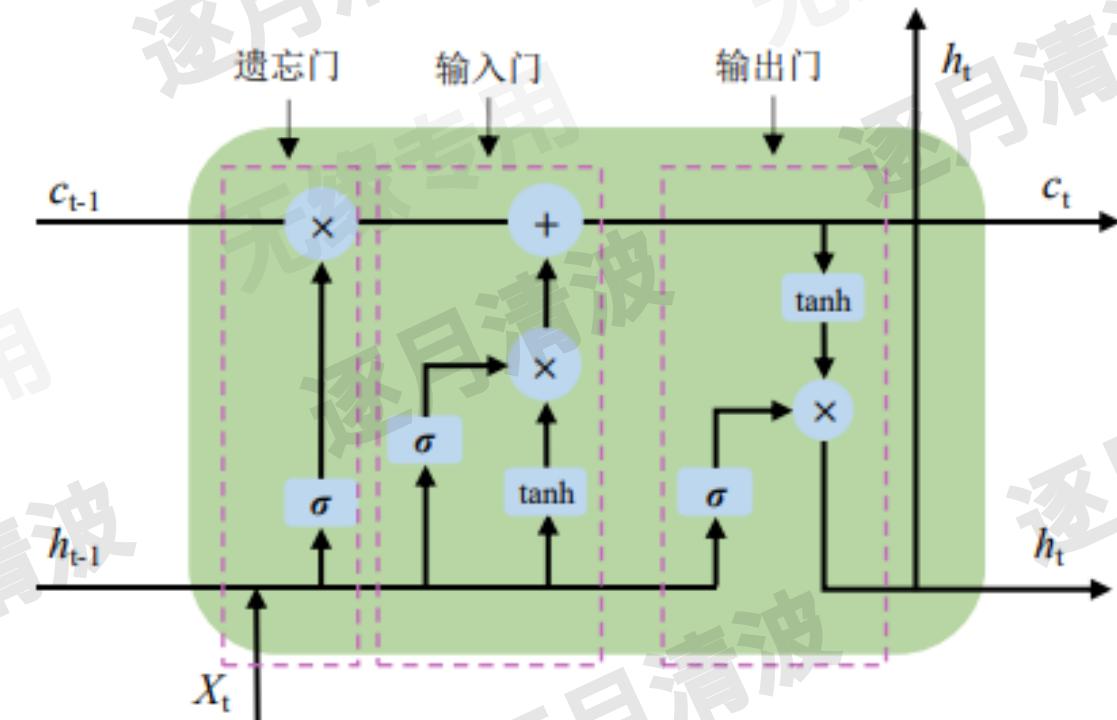
199、LSTM如何实现长短期记忆功能？(3星)

输入门控制当前计算的新状态以多大程度更新到记忆单元中；遗忘门控制前一步记忆单元中的信息有多大程度被遗忘掉；输出门控制当前的输出有多大程度上取决于当前的记忆单元。

在一个训练好的网络中，当输入的序列中没有重要信息时，LSTM的遗忘门的值接近于1，输入门的值接近于0，此时过去的记忆会被保存，从而实现了长期记忆功能；

当输入的序列中出现了重要的信息时，当将其存入记忆中，此时其输入门的值会接近于1；当输入的序列中出现了重要信息，且该信息意味着之前的记忆不再重要时，输入门的值接近1，而遗忘门的值接近于0，这样旧的记忆被遗忘，新的重要信息被记忆。

经过这样的设计，整个网络更容易学习到序列之间的长期依赖。



$$O_t^f = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f)$$

$$O_t^i = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \odot \tanh(W_{i2} \cdot [x_t, h_{t-1}] + b_{i2})$$

$$c_t = O_t^f \odot c_{t-1} + O_t^i$$

$$h_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \odot \tanh(c_t)$$

200、什么是异常值？异常值在哪些情况下有益或有害？（2星）

在机器学习中，异常检测和处理是一个比较小的分支，或者说是机器学习的一个副产物。

在一般的预测问题中，模型通常是对整体样本数据结构的一种表达方式，这种表达方式通常抓住的是整体样本一般性的性质，而那些在这些性质上表现完全与整体样本不一致的点，我们就称其为异常点，**通常异常点在预测问题中是不受欢迎的**，因为预测问题通常关注的是**整体样本的性质**，而异常点的生成机制与整体样本完全不一致，如果算法对异常点敏感，那么生成的模型并不能对整体样本有一个较好的表达，从而预测也会不准确。

基于观测降雨量的深度值预测中离群值具有重要意义。房价预测中的异常值则没有任何意义。从另一方面来说，异常点在**某些预测特定类别场景下**反而令分析者感到极大兴趣，如疾病预测，通常健康人的身体指标在某些维度上是相似，如果一个人的身体指标出现了异常，那么他的身体情况在某些方面肯定发生了改变，当然这种改变并不一定是由疾病引起（这类异常值通常被称为噪声），但异常的发生和检测是疾病预测一个重要起始点。相似的场景也可以应用到信用欺诈，网络攻击等等场景中。

201、什么是混淆矩阵?它的作用是什么? (4星)

混淆矩阵是分类模型中常用的一种模型评估指标。它展示了分类模型对样本的分类结果与实际分类结果之间的比对。混淆矩阵通常是一个 $N \times N$ 的矩阵，其中 N 代表分类模型的类别数。对于一个二分类模型，混淆矩阵通常如下所示：

	预测值0	预测值1
真实值0	TN	FP
真实值1	FN	TP

其中，TP (True Positive) 表示模型将正类预测为正类的数量，FN (False Negative) 表示模型将正类预测为负类的数量，FP (False Positive) 表示模型将负类预测为正类的数量，TN (True Negative) 表示模型将负类预测为负类的数量。

混淆矩阵的作用是衡量分类模型的预测能力，可以通过混淆矩阵计算出各种分类评价指标，例如精确率、召回率、F1值等。通过分析混淆矩阵，我们可以了解模型的预测表现，进一步优化模型或者调整阈值，使得模型性能达到最佳。

202、如何提升机器学习模型的稳定性？(3星)

1. **数据增强**: 通过一系列规则和变换，增加原始数据集的样本数，如旋转、平移、缩放、镜像等方法。这样可以避免过拟合，从而提高模型的稳定性。
2. **特征选择**: 选择有代表性的特征并去掉不相关的特征，这样可以降低模型的复杂度，提高泛化能力。
3. **模型集成**: 将多个模型的预测结果进行加权平均或投票，可以降低模型方差，提高模型稳定性和准确率。
4. **正则化方法**: 通过添加惩罚项或限制模型参数的大小来控制模型的复杂度，从而避免过拟合，提高模型的稳定性。
5. **交叉验证**: 通过将数据集分成若干份，在每次训练中使用其中一份作为验证集来评估模型的泛化能力，可以避免过拟合和欠拟合，提高模型的稳定性。
6. **早停**: 提前停止是指模型在验证集上取得不错的性能时停止训练。这种方式本质和正则化是一个道理，能减少方差的同时增加的偏差。目的为了平衡训练集和未知数据之间在模型的表现差异。

203、有哪些改善机器学习模型的思路？（3星）

- 1. 数据角度：**增强数据集。无论是有监督还是无监督学习，数据永远是最重要的驱动力。更多的类型数据对良好的模型能带来更好的稳定性和对未知数据的可预见性。对模型来说，“看到过的总比没看到的更具有判别的信心”。
- 2. 模型角度：**模型的容限能力决定着模型可优化的空间。在数据量充足的前提下，对同类型的模型，增大模型规模来提升容限无疑是最直接和有效的手段。
- 3. 调参优化角度：**如果你知道模型的性能为什么不再提高了，那已经向提升性能跨出了一大步。超参数调整本身是一个比较大的问题。一般可以包含模型初始化的配置，优化算法的选取、学习率的策略以及如何配置正则和损失函数等等。
- 4. 训练角度：**在越大规模的数据集或者模型上，诚然一个好的优化算法总能加速收敛。但你在未探索到模型的上限之前，永远不知道训练多久算训练完成。所以在改善模型上充分训练永远是最必要的过程。充分训练的含义不仅仅只是增大训练轮数。有效的学习率衰减和正则同样是充分训练中非常必要的手段。

204、阐述一下强化学习和监督学习、无监督学习的区别。 (3星)

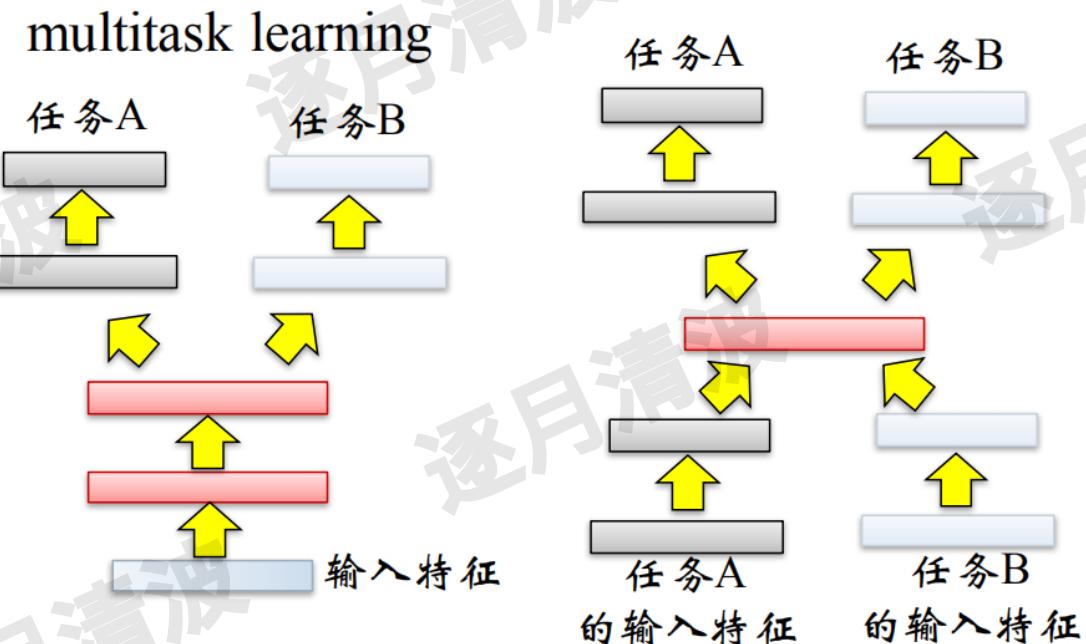
1. 监督式学习好比在学习的时候，有一个导师在旁边指点，他知道怎么是对的怎么是错的。强化学习会在**没有任何标签的情况下**，通过先尝试做出一些行为得到一个结果，通过这个结果是对还是错的反馈，调整之前的行为，就这样不断的调整，算法能够学习到在什么样的情况下选择什么样的行为可以得到最好的结果。
2. 监督式学习出的是输入输出之间的关系，可以告诉算法什么样的输入对应着什么样的输出。监督学习做了比较坏的选择会立刻反馈给算法。强化学习给出的是**机器的反馈**，即用来判断这个行为是好是坏。另外强化学习的结果反馈有延时，有时候可能需要走了很多步以后才知道以前的某一步的选择是好还是坏。
3. 监督学习的输入是**独立同分布的**。强化学习面对的输入总是在变化，每当算法做出一个行为，它影响下一次决策的输入。
4. 非监督式不是学习输入到输出的映射，而是模式(自动映射)。对强化学习来说，它通过对没有概念标记、但与一个延迟奖赏或效用相关联的训练例进行学习，以获得**某种从状态到行动的映射**。

	监督学习	非监督学习	强化学习
标签	正确且严格的标签	没有标签	没有标签，通过结果反馈调整
输入	独立同分布	独立同分布	输入总是在变化，每当算法做出一个行为，它影响下一次决策的输入。
输出	输入对应输出	自学习映射关系	reward function，即结果用来判断这个行为是好是坏

205、什么是多任务学习？（3星）

在机器学习中，我们通常关心优化某一特定指标，不管这个指标是一个标准值，还是企业KPI。为了达到这个目标，我们训练单一模型或多个模型集合来完成指定得任务。然后，我们通过精细调参，来改进模型直至性能不再提升。尽管这样做可以针对一个任务得到一个可接受得性能，但是我们可能忽略了一些信息，这些信息有助于在我们关心的指标上做得更好。具体来说，这些信息就是相关任务的监督数据。通过在相关任务间共享表示信息，我们的模型在原始任务上泛化性能更好。这种方法称为多任务学习（Multi-Task Learning）。

多任务学习旨在通过在一个学习器上同时处理多个相关任务来提高学习性能。与传统的单任务学习相比，MTL 常常可以更好地利用数据之间的关联性，从而达到更高的泛化能力和效率。在 MTL 中，多个任务通过共享特征处理器或部分网络结构来实现联合学习。这种共享机制可以帮助多个任务共同学习特征，促进任务之间的知识迁移和正则化，降低过拟合的风险，提高学习效率和准确率。



206、什么是迁移学习？为什么需要迁移学习？（3星）

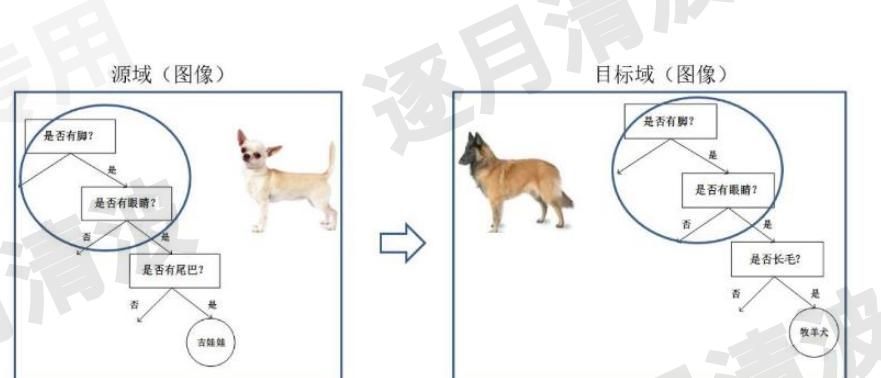
迁移学习(Transfer Learning)是一种机器学习方法，就是把为任务A开发的模型作为初始点，重新使用在为任务B开发模型的过程中。**迁移学习是通过从已学习的相关任务中转移知识来改进学习的新任务**，虽然大多数机器学习算法都是为了解决单个任务而设计的，但是促进迁移学习的算法的开发是机器学习社区持续关注的话题。迁移学习对人类来说很常见，例如，我们可能会发现学习识别苹果可能有助于识别梨，或者学习弹奏电子琴可能有助于学习钢琴。如果能找到目标问题的相似性，就可以进行模型之间的迁移。迁移学习任务就是从相似性出发，将旧领域(domain)学习过的模型应用在新领域上。

为什么需要迁移学习呢？原因有：

- 大数据与少标注的矛盾**：虽然有大量的数据，但往往都是没有标注的，无法训练机器学习模型。人工进行数据标注太耗时。
- 大数据与弱计算的矛盾**：部分场景缺乏庞大的数据量与计算资源。因此需要借助于模型的迁移。
- 普适化模型与个性化需求的矛盾**：即使是在同一个任务上，一个模型也往往难以满足每个人的个性化需求，比如特定的隐私设置。这就需要在不同人之间做模型的适配。

207、迁移学习有哪些常见的方法？（2星）

1. 基于样本的迁移 (Instance based TL): 通过权重重用源域和目标域的样例进行迁移
2. 基于特征的迁移 (Feature based TL): 将源域和目标域的特征变换到相同空间
3. 基于模型的迁移 (Parameter based TL): 利用源域和目标域的参数共享模型
4. 基于关系的迁移 (Relation based TL): 利用源域中的逻辑网络关系进行迁移



208、迁移学习与传统机器学习有什么区别？(2星)

1. 从**数据分布的角度**来说：迁移学习的训练和测试数据不需要同分布，而传统机器学习的训练和测试数据是要求同分布的
2. 从**数据标签的角度**来说：迁移学习不需要足够的数据标注，而传统机器学习要求足够的数据标注
3. 从**建模的角度**来说：迁移学习可以重用之前的模型，而传统机器学习的每个任务分别建模

传统的机器学习方法通常是在一个独立的数据集上训练模型，然后使用该模型进行预测或分类。而迁移学习则是利用预先在其他相关任务上训练的模型参数或特征来加速或改善新任务的学习。这种方法可以减少在新数据集上的大规模数据收集和标注开销，因此迁移学习在面对数据量相对较少的问题上表现出了非常好的效果。

209、迁移学习和多任务学习有什么区别？(2星)

1. 从**任务阶段**来说：多任务学习是同时学习多个不同任务，而归纳迁移学习通常分为两个阶段，即源任务上的学习阶段和目标任务上的迁移学习阶段；
2. 从**任务目的**来说：迁移学习是单向的知识迁移，希望提高模型在目标任务上的性能，而多任务学习是希望提高所有任务的性能。

总的来说，迁移学习是利用一个领域的知识去改善另一个领域的模型，而多任务学习则是通过共享参数实现多个任务之间的联合学习来提高整体模型的性能。两种方法可以结合使用，比如在迁移学习的过程中采用多任务学习的方式来共同学习源领域和目标领域的任务。

210、什么是微调 (Fine-tune) ?为什么要使用微调？(4星)

在实际的应用中，我们通常不会针对一个新任务，就去从头开始训练一个神经网络。这样的操作显然是非常耗时的。尤其是当我们的训练数据不大，不足以训练出泛化能力足够强的深度神经网络时。即便有如此之多的训练数据，从头开始训练的代价也通常不可承受的。而微调指的是将一个在大型数据集上预先训练过的深度神经网络模型应用于一个新的但相似的任务（例如，图像分类器在一个新的但相似的图像数据集上做出预测）。在微调时，我们保留已经训练好的神经网络中的大部分权重，并在新数据集上训练一小部分权重。

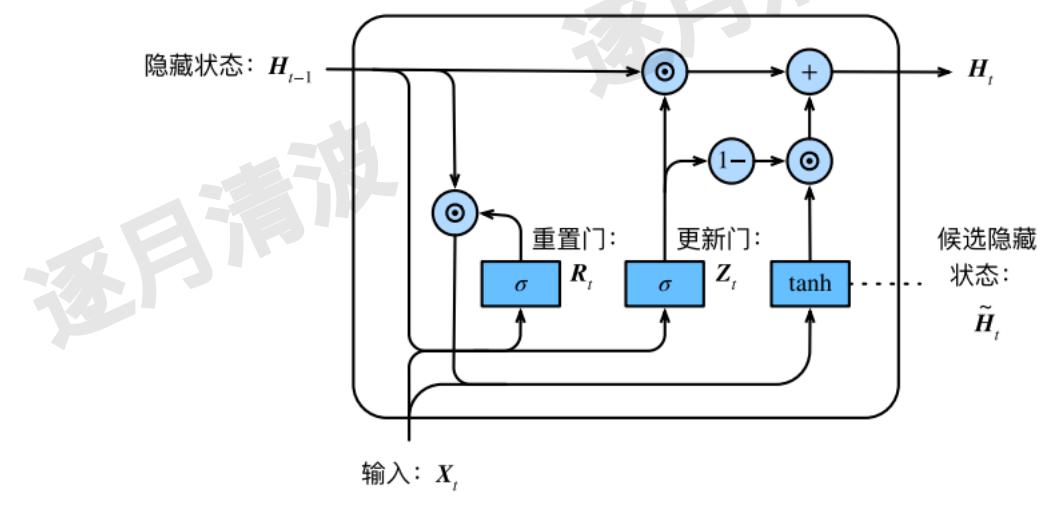
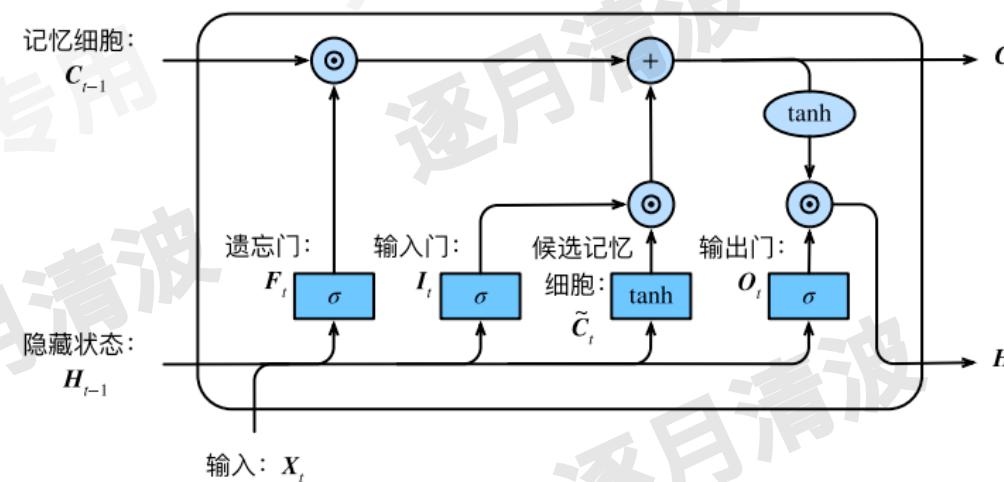
微调是有意识地使用预训练模型来解决新问题的策略。从头开始训练深度神经网络可能需要很长时间，并需要更多的数据和计算资源。

Fine-tune也许是**最简单的深度网络迁移方法**。因为别人训练好的模型，可能并不是完全适用于我们自己的任务。可能别人的训练数据和我们的数据之间不服从同一个分布；可能别人的网络能做比我们的任务更多的事情；可能别人的网络比较复杂，我们的任务比较简单。通过微调，我们可以利用已经在某个大型数据集上学到的特征，加速学习过程，并获得更好的性能。使用微调，我们可以采用新任务的更少的数据，并获得与从头开始训练相比更快的训练速度。

211、简述LSTM与GRU的区别。 (3星)

LSTM与GRU二者结构十分相似，它们的区别在于：

1. 新的记忆都是根据之前状态及输入进行计算，但是GRU中有一个**重置门**控制之前状态的进入量，而在LSTM里没有类似门；
2. 产生新的状态方式不同，LSTM有两个不同的门，分别是**遗忘门(forget gate)**和**输入门(input gate)**，而GRU只有一种**更新门(update gate)**；
3. LSTM对新产生的状态可以通过**输出门(output gate)**进行调节，而GRU对输出**无任何调节**。
4. GRU的优点是这是个**更加简单的模型**，所以更容易创建一个更大的网络，而且它只有两个门，在计算性上也运行得更快，然后它可以扩大模型的规模。
5. LSTM更加强大和灵活，因为它有三个门而不是**两个**。



212、简述余弦距离与欧式距离，什么时候会使用余弦距离？(4星)

欧式距离：是欧几里得空间中两点间“普通”（即直线）距离。

余弦距离：是指两个“向量”之间的余弦相似度

$$\cos(A, B) = \frac{A * B}{\|A\|_2 \|B\|_2}$$

对于两个向量A和B，余弦距离关注的是向量之间的角度关系，并不关心它们的绝对大小，其取值范围是[-1, 1]。

余弦距离常常用在文本、图像、视频等领域。当一对文本相似度的长度差距很大、但内容相近时，如果使用词频或词向量作为特征，它们在特征空间中的的欧式距离通常很大；而如果使用余弦相似度的话，它们之间的夹角可能很小，因而相似度高。

此外，在文本、图像、视频等领域，研究的对象的特征维度往往很高，余弦相似度在高维情况下依然保持“相同时为1，正交时为0，相反时为-1”的性质，而欧式距离的数值则受维度的影响，范围不固定，并且含义也比较模糊。

213、简要叙述几种模型的评估方法。（3星）

1. **Holdout检验**: Holdout 检验是最简单也是最直接的验证方法，它将原始的样本集合随机划分成训练集和验证集两部分。比方说，对于一个点击率预测模型，我们把样本按照 70%~30% 的比例分成两部分，70% 的样本用于模型训练；30% 的样本用于模型验证，包括绘制ROC曲线、计算精确率和召回率等指标来评估模型性能。Holdout 检验的缺点很明显，即在验证集上计算出来的最后评估指标与原始分组有很大关系。为了消除随机性，研究者们引入了“交叉检验”的思想。
2. **交叉检验**: k-fold 交叉验证：首先将全部样本划分成k个大小相等的样本子集；依次遍历这k个子集，每次把当前子集作为验证集，其余所有子集作为训练集，进行模型训练和评估；最后把k次评估指标的平均值作为最终的评估指标。在实际实验中，k经常取10。
3. **自助法**: 不管是Holdout检验还是交叉检验，都是基于划分训练集和测试集的方法进行模型评估的。然而，当样本规模比较小时，将样本集进行划分会让训练集进一步减小，这可能会影响模型训练效果。有没有能维持训练集样本规模的验证方法呢？自助法可以比较好地解决这个问题。自助法是基于自助采样法的检验方法。对于总数为n的样本集合，进行n次有放回的随机抽样，得到大小为n的训练集。n次采样过程中，有的样本会被重复采样，有的样本没有被抽出过，将这些没有被抽出的样本作为验证集，进行模型验证，这就是自助法的验证过程。

214、简要叙述几种超参数调优的方法。（2星）

超参数调优一般有网格搜索、随机搜索、贝叶斯优化等算法。在具体介绍算法之前，需要明确超参数搜索算法一般包括哪几个要素。一是目标函数，即算法需要最大化/最小化的目标；二是搜索范围，一般通过上限和下限来确定；三是算法的其他参数，如搜索步长。

1. **网格搜索**，可能是最简单、应用最广泛的超参数搜索算法，它通过查找搜索范围内的所有的点来确定最优值。如果采用较大的搜索范围以及较小的步长，网格搜索有很大概率找到全局最优值。然而，这种搜索方案十分**消耗计算资源和时间**，特别是需要调优的超参数比较多的时候。因此，在实际应用中，网格搜索法一般会先使用较广的搜索范围和较大的步长，来寻找全局最优值可能的位置；然后会逐渐缩小搜索范围和步长，来寻找更精确的最优值。这种操作方案可以降低所需的时间和计算量，但由于目标函数一般是非凸的，所以很可能会错过全局最优值。

2. **随机搜索**，随机搜索的思想与网格搜索比较相似，只是不再测试上界和下界之间的所有值，而是在搜索范围内随机选取样本点。它的理论依据是，如果样本点集足够大，那么通过随机采样也能大概率地找到全局最优值，或其近似值。随机搜索一般会比网格搜索要快一些，但是和网格搜索的快速版一样，它的结果也是没法保证的。

3. **贝叶斯优化算法**，网格搜索和随机搜索在测试一个新点时，会忽略前一个点的信息；而贝叶斯优化算法则充分利用了之前的信息。贝叶斯优化算法通过对目标函数形状进行学习，找到使目标函数向全局最优值提升的参数。

215、为什么说神经网络是端到端的模型？(3星)

神经网络是端到端的模型，是因为它可以**直接从输入数据中学习到输出数据之间的映射关系，无需手动设计特征和模型**，从而实现了全自动化的机器学习。也就是说，神经网络将数据输入模型，通过多层的非线性变换，自动地提取出数据中的高层次特征，再将这些特征转换为最终的输出结果。这个过程是端到端的，因为输入数据和输出数据连成了一条整个模型的“端到端”路径，没有中间环节需要手动干预或者设计。

端到端学习(end-to-end)是一种解决问题的思路，**与之对应的是多步骤解决问题**，也就是将一个问题拆分为多个步骤分步解决，而端到端是由输入端的数据直接得到输出端的结果，不要预处理和特征提取。特征提取包含在神经网络内部，所以说神经网络是端到端的网络。

这种端到端的方法有助于**减少了机器学习的复杂度，也方便了实际场景中的部署和使用**。通过神经网络，我们可以直接学习到数据的高层次特征，更好地解决机器学习中存在的各种问题。同时，也降低了DIY需要经验的难度和负担，进一步加速了机器学习的落地。

216、什么是softmax？(4星)

Softmax是一种常用的数学函数，常用于多分类问题中，用于将输出层的原始分数或概率转换为分类概率。Softmax将原始分数转换为概率的方式是将每个类别的分数指数化并除以所有类别得到的总和，这使得每个类别的概率为零或正且总和为1。Softmax的公式如下：

$$P(y = j|x) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

其中， z_j 代表第j个类别的原始分数， K 代表总共的类别数， $P(y=j|x)$ 表示给定输入x，输出为第j个类别的概率。

Softmax函数的优点是，它可以将分数转换为概率，并将它们归一化，从而方便我们进行分类。此外，由于softmax函数将输出转换为概率分布，因此我们可以对概率值进行解释和分析，比如找出最可能的类别、计算每个类别的置信度等等。

217、Batch Size 对神经网络的训练效果有什么影响？(3星)

神经网络中的batch size是指在每一次迭代（epoch）中，输入数据被分成的小批量的样本数。换句话说，batch size指的是每次模型在训练时一次性处理的样本数。通常情况下，一个batch size的大小是32、64、128、256等等。batch size的大小对于模型的训练速度、学习率和收敛性都有很大的影响。

1. Batch Size太小，模型表现效果极其糟糕(error飙升)。
2. 随着Batch Size增大，处理相同数据量的速度越快。
3. 随着Batch Size增大，达到相同精度所需要的epoch数量越来越多。
4. 由于上述两种因素的矛盾，Batch Size增大到某个时候，达到时间上的最优。
5. 由于最终收敛精度会陷入不同的局部极值，因此Batch Size增大到某些时候，达到最终收敛精度上的最优。

总而言之，较小的batch size可以增加噪声和随机性，并使训练更快，但代价是训练过程更不稳定，而较大的batch size可以提高训练稳定性，但是会增加训练时间和内存开销。

218、神经网络中的激活函数需要具备哪些性质？(4星)

1. **非线性**: 当激活函数是非线性的，一个两层的神经网络就可以基本上逼近所有的函数；
2. **可微性**: 当优化方法是基于梯度下降的时候，可微的激活函数是必须的；
3. **单调性**: 当激活函数是单调的时候，单层网络能够保证是凸函数；
4. **$f(x) \approx x$** : 当激活函数满足这个性质的时候，如果参数的初始化是随机的较小值，那么神经网络的训练将会很高效；如果不满足这个性质，那么就需要详细地去设置初始值；
5. **输出值的范围**: 当激活函数输出值是**有限**的时候，基于梯度的优化方法会**更加稳定**，因为特征的表示受有限权值的影响更显著；当激活函数的输出是**无限**的时候，模型的训练会**更加高效**，不过在这种情况下，一般需要更小的学习率。

219、在k-means或kNN中，我们常用欧氏距离来计算最近的邻居之间的距离，有时也用曼哈顿距离，对比一下这两种距离的差别。（3星）

欧氏距离虽然很有用，但也有明显的缺点。它将样品的不同属性（即各指标或各变量量纲）之间的差别等同看待，这一点有时不能满足实际要求。例如，在教育研究中，经常遇到对人的分析和判别，个体的不同属性对于区分个体有着不同的重要性。因此，**欧氏距离适用于向量各分量的度量标准统一的情况。**

曼哈顿距离，我们可以定义曼哈顿距离的正式意义为L1-距离或城市区块距离，也就是在欧几里得空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和。例如在平面上，坐标 (x_1, y_1) 的点P1与坐标 (x_2, y_2) 的点P2的曼哈顿距离为： $|x_1-x_2| + |y_1-y_2|$ ，要注意的是，曼哈顿距离依赖座标系统的转度，而非系统在座标轴上的平移或映射。当坐标轴变动时，点间的距离就会不同。

通俗来讲，想象你在曼哈顿要从一个十字路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。而实际驾驶距离就是这个“曼哈顿距离”，这也是曼哈顿距离名称的来源，同时，曼哈顿距离也称为城市街区距离(City Block distance)。

曼哈顿距离和欧式距离一般用途不同，无相互替代性。**相对于欧氏距离，曼哈顿距离更适用于维度较低、稀疏数据的情况。在计算机视觉中，曼哈顿距离也被用来衡量图片的相似度。**

220、CNN最成功的应用是在CV，为什么NLP的很多问题也可以用CNN，AlphaGo里也用了CNN？CNN通过什么手段抓住了这个共性？（3星）

以上几个不相关问题的相关性在于，都存在**局部与整体的关系**，由低层次的特征经过组合，组成高层次的特征，并且得到不同特征之间的空间相关性。因为**CNN**拥有对于局部模式的很好的特征提取能力，而文本等类似数据也同样拥有这种局部模式。

CNN抓住此共性的手段主要有四个：**局部连接 / 权值共享 / 池化操作 / 多层次结构**。

- 局部连接使网络可以提取数据的局部特征；
- 权值共享大大降低了网络的训练难度，一个Filter只提取一个特征，在整个图片（或者语音 / 文本）中进行卷积；
- 池化操作与多层次结构一起，实现了数据的降维，将低层次的局部特征组合成为较高层次的特征，从而对整个图片进行表示。

221、为什么朴素贝叶斯被描述为“朴素”？(2星)

朴素贝叶斯算法被描述为“朴素”是因为它做了一个非常理想化的假设：假设所有的特征之间相互独立，互不影响，而且同样重要。这个假设虽然在现实生活中往往不成立，而且很难成立。

但是对于有限的数据集而言，这个假设的表现还是相当优秀的。这个朴素的假设使得朴素贝叶斯算法计算简单，速度快，并且不需要大量的训练数据，因此成为了文本分类、垃圾邮件过滤、情感分析等自然语言处理任务中最常用的算法之一。

222、分析K-Means算法的时间与空间复杂度。 (4星)

分析一下算法原理：

1. 随机选取k个中心点
2. 遍历所有数据，将每个数据划分到最近的中心点中
3. 计算每个聚类的平均值，并作为新的中心点；
4. 重复2-3，直到这k个中心点不再变化（收敛了）

时间复杂度： $O(tKn)$ ，其中， t 为迭代次数， K 为簇的数目， m 为训练样本数， n 为维数

空间复杂度： $O((m+K)n)$ ，其中， K 为簇的数目， m 为训练样本数， n 为维数。

223、特征比数据量还大时，应选择什么样的分类器？（3星）

线性分类器。

因为维度高的时候，数据一般在维度空间里面会比较稀疏，很有可能线性可分。

进阶的解决方案是使用带有L1正则化的线性模型，如Lasso Regression。L1正则化可以将无用特征的系数缩小甚至置为零，从而减少了特征对分类结果的影响。

另外，使用**基于树的方法**，如随机森林和Boosting方法，通常也表现良好，因为它们能够选择重要特征并将其用于分类任务。除此之外，降维方法如主成分分析（PCA）和线性判别分析（LDA）也是可选的方法。它们可以减小特征空间，并保留最能区分不同类别的信息。

需要注意的是，特征选择仍然是更好的做法，因为它可以从原始特征中选择最相关或最重要的那些特征，而不是直接舍弃，从而在保留更有效信息的同时，减小数据集的复杂度。

224、AUC的缺点是什么？(3星)

AUC (Area Under the Curve) 是用于评估分类器性能的一种指标，它测量分类器在不同阈值下的假阳性率 (False Positive Rate) 和真阳性率 (True Positive Rate) 之间的面积。AUC的缺点主要有以下：

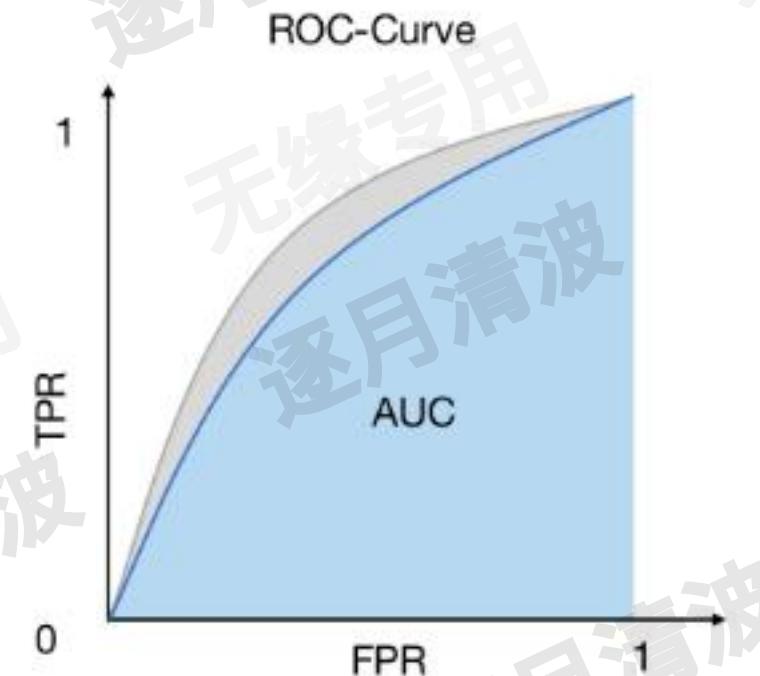
1. AUC不能解释分类器的预测结果。它只给出了分类器的总体分类性能，无法解释对每个类别的预测结果。
2. AUC对类别不平衡敏感。对于严重不平衡的数据集，AUC会高估分类器性能，因为即使分类器的真实阳性预测很少，假阳性也很少，导致AUC的值很高。
3. AUC无法处理多分类问题，因为AUC要求分类器的输出是二元的。
4. AUC无法处理连续输出，需要将输出转化为二元分类结果。
5. AUC只关注分类器的排序能力，而不考虑分类器的绝对预测值。将样本预测为 $(0.51, 0.49)$ 与预测为 $(1, 0)$ 得到的auc是相同的。
6. 即使两个模型的能力不同，但可能auc是相等的（即两条曲线完全不同，但曲线下的总面积相同），这种模型能力的差别无法由auc反应。

225、AUC是否受正负样本比例影响？(3星)

不受正负样本比例影响。

第一种角度：从AUC的定义来看，实质是随机取一对正负样本，正样本得分大于负样本得分的概率，因此AUC不能衡量正样本内部的排序。

第二种角度：横轴FPR只关注负样本，与正样本无关；纵轴TPR只关注正样本，与负样本无关。所以横纵轴都不受正负样本比例影响，积分当然也不受其影响。



$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

226、参数和非参数机器学习模型有什么区别？(4星)

在统计学中，参数模型 (parametric model) 通常假设总体服从某个分布，这个分布可以由一些参数确定，如正态分布由均值和标准差确定，在此基础上构建的模型称为参数模型；非参数模型 (non-parametric model) 对于总体的分布不做任何假设或者说假设自由，只知道其分布是存在的，所以就无法得到其分布的相关参数，只能通过非参数统计的方法进行推断。

参数模型和非参数模型中的“参数”并不是模型中的参数，而是数据分布的参数。问题中有没有参数，并不是参数模型和非参数模型的区别。其区别主要在于总体的分布形式是否已知。而为何强调“参数”与“非参数”，主要原因在于参数模型的分布可以有参数直接确定。

一、非参数模型并不是说模型中没有参数，而是参数很多或者说参数不确定。这里的non-parametric类似单词priceless，并不是没有价值，而是价值非常高，也就是参数是非常非常非常多的！（注意：所谓“多”的标准，就是参数数目大体和样本规模差不多）反过来，可以通过有限个参数来确定一个模型，这样的方式就是“有参数模型”，也就是这里说的参数模型，如线性回归、Logistic回归（假定样本维度为N，则假定N个参数 $\theta_1, \theta_2 \dots \theta_N$ ）。

二、参数模型对学到的函数方程有特定的形式，也就是明确指定了目标函数的形式。比如线性回归模型，就是一次方程的形式，然后通过训练数据学习到具体的参数。该假设可以极大地简化学习过程，但是同样可以限制学习的内容。简化目标函数为已知形式的算法就称为参数机器学习算法。

常见的参数机器学习模型有：线性回归、逻辑回归、感知机等

常见的非参数机器学习模型有：决策树、朴素贝叶斯、支持向量机、神经网络等

227、在什么情况下，更适合用决策树而不是随机森林？（3星）

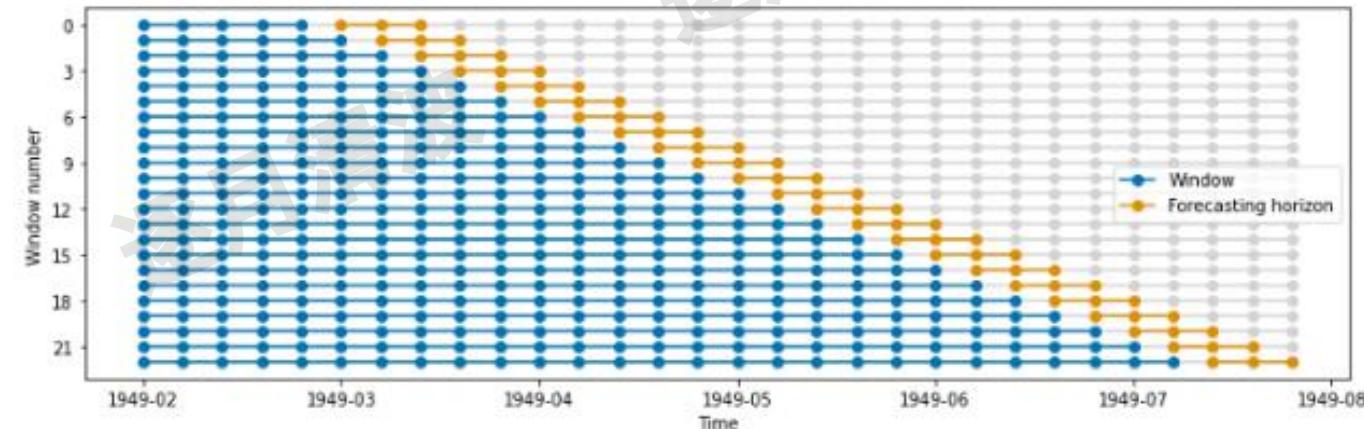
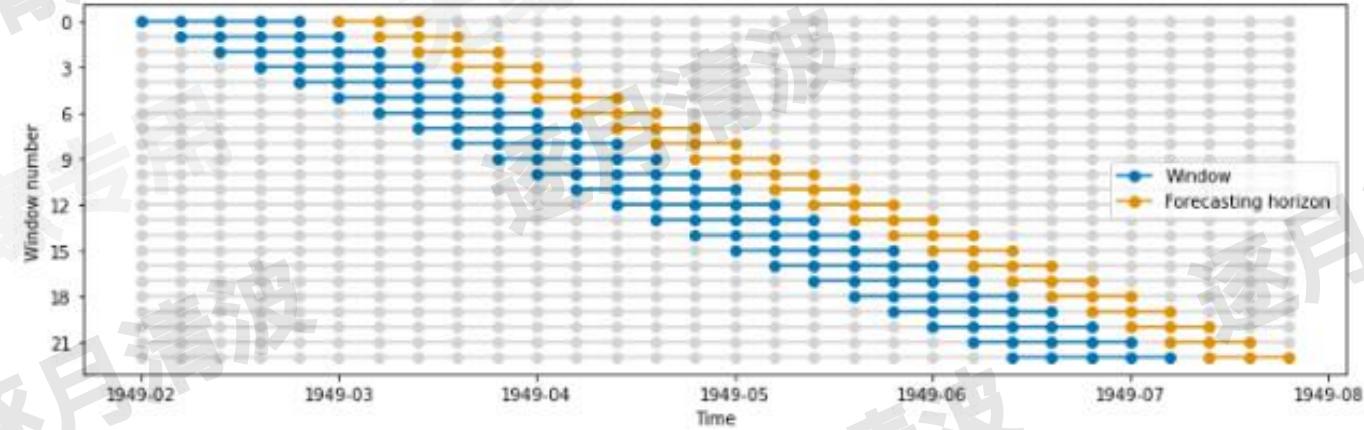
决策树是一种简单且易于解释的算法，它通过将数据集分成不同的子集来建立一棵树型结构的模型，每一个节点代表着一个特征，根据该特征将数据分到不同的子节点中。在训练完决策树之后，我们可以根据树的结构来解释决策的过程。与此不同，随机森林是一种通过集成多棵决策树来提高性能的算法。随机森林由多个决策树组成，每棵决策树都用一部分训练数据进行训练，并且采用了随机的特征选择方式来降低模型的方差和过拟合风险，最终通过投票的方式来产生预测结果。

1. **需要可解释性时**：当要求不需要性能，但还需要解释工作和解决方案。当你用决策树解决任何问题时，你会得到一个适当的树状结构来解释完整的树构建过程，可以轻易解释模型的工作。
2. **缺乏计算能力时**：如果设备的计算能力较低，可以使用决策树。
3. **需要筛选有用的特征时**：这是一个实际用例，当有一些想优先使用的方便的特征时，决策树很有帮助，因为在随机森林中，特征是随机选择的。

综上所述，一般来说，当数据集较小、特征较少、对模型解释性要求较高，或者需要理解决策流程时，决策树更为适合。而随机森林则适用于大规模的、高维度的数据集，需要提高模型精度和泛化性能的情况下。然而，需要注意的是，随机森林相对于决策树而言，模型较为复杂，训练过程也较长，因此如果时间和计算资源受限，可能不利于使用随机森林建模。

228、交叉验证如何用在时间序列数据上？(3星)

与标准的k-fold交叉检验不同，时序数据里的数据不是随机分布的，而是具有时序性的。如果模式出现在后期，模型仍然需要选择先前时间的数据，可以如下这么做：



229、什么是box-cox变换？(3星)

Box-Cox变换是一种常用的数据变换方法，用于将数据转换为服从正态分布的数据。它的基本思想是通过对目标变量进行幂函数转换，将具有任意分布的随机变量转换为近似正态分布的随机变量。

具体来说，对于Box-Cox变换，当 $\lambda \neq 0$ 时，它的公式为：

$$y^{(\lambda)} = \begin{cases} \frac{(y^\lambda - 1)}{\lambda}, & \lambda \neq 0 \\ \ln(y), & \lambda = 0 \end{cases}$$

其中， y 是原始数据， λ 是需要确定的参数。当确定了最佳的 λ 值后，将原始数据进行Box-Cox变换，可以使数据更好地满足数据分析和建模的假设条件。经常在回归分析、ANOVA分析和时间序列分析中使用Box-Cox变换来预处理数据。

230、深度学习中网络参数初始化有哪些方法？(4星)

深度学习中网络参数的初始化可以影响模型的收敛速度和效果。常用的参数初始化方法有：

1. **随机初始化**: 在神经网络中，一般会随机初始化网络参数，这样不同的神经元就不会有相同的初始权重，从而避免了对称性，有利于网络的学习。
2. **Xavier 初始化**: Xavier 初始化也称为“Glorot 初始化”，是一种常用的初始化方法，它使得每一层的激活值的方差和上一层的输入的方差相等，这样可以使得信号在正向传播和反向传播中保持相同的数据量级，有利于网络的训练。

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

3. **He (Kaiming) 初始化**: 在ReLU等激活函数中使用比较合适，它将权重初始化为均值为0，方差为 $2/n$ 的正态分布，其中n为前一层神经元的个数。

$$w \sim G\left[0, \sqrt{\frac{2}{n}}\right]$$

4. **正交初始化**: 正交矩阵是一个特殊的矩阵，它的行和列都是互相正交的，正交初始化将神经网络的权重矩阵初始化为一个正交矩阵，可以有效地避免梯度消失和梯度爆炸问题。

5. **预训练初始化**: 在某些情况下，可以使用预训练的初始化方法，即使用已经在大量数据上训练好的模型的参数来初始化当前模型的参数，从而加速模型的收敛速度。

231、深度学习中有哪些网络训练技巧？（3星）

1. 参数初始化：Xavier、Kaiming、Uniform、Normal等，结果基本都差不多。但是一定要做。否则可能会减慢收敛速度，影响收敛结果，甚至造成Nan等一系列问题
2. 数据预处理：z-score等
3. 梯度裁剪：限制最大梯度，如果梯度超过了阈值，就直接截断，或根据梯度的范数来裁剪
4. 除了gate之类的地方，需要把输出限制成0-1之外，尽量不要用sigmoid
5. 尽量对数据做shuffle，避免一些意料之外的信息泄露
6. 在数据集很大的情况下，先用 1/100、1/10 的数据跑一跑，对模型性能和训练时间有个底，外推一下全量数据到底需要跑多久。在没有足够的信心前不做大规模实验
7. 一轮加正则，一轮不加正则，反复进行等

232、什么是数据增强？对于各种类型的数据应该怎样操作？（3星）

数据增强（Data Augmentation）是指通过对已有数据进行一系列变换来生成新的样本数据，以此增加模型训练数据，提升模型的鲁棒性和泛化能力。这也与增加噪声是类似的，或称之为增加扰动。它起到了与正则化方法类似的作用，即抑制训练数据的过拟合。对于不同类型的数据，可采用不同的数据增强方法。例如：

1. **图像数据**：常用的数据增强方法包括翻转、旋转、缩放、裁剪、加噪声等。这些操作可用于生成更多的样本，同时使得模型对于物体出现的角度、大小、位置等因素更加具有鲁棒性。
2. **自然语言数据**：可对文本进行删除、替换、插入、打乱顺序等方式进行数据增强。这些方法可以提升模型对于语义、语法、语境等方面的理解能力。
3. **语音数据**：常用的数据增强方法包括声音剪切、语速变换、语调变换等。这些操作可用于增加音频样本的数量，同时增强模型对于不同说话人、不同语速、不同音调的语音识别能力。

增强数据时要保证变换后生成的新数据仍具有原有数据的特征和分布，以免影响模型的训练效果。同时，合理选择数据增强方式也需要依赖于具体的应用场景。

233、为什么一般情况下ReLU好于tanh和sigmoid ? ReLU有没有可以改进的地方 ? (4星)

ReLU的优势在于：

1. 采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法，计算量相对大，而采用Relu激活函数，整个过程的计算量节省很多。
2. 对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失的情况（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失，从而无法完成深层网络的训练。
3. 第三，Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。

ReLU也有可以改进的地方：

1. ReLU的输出不是zero-centered
2. Dead ReLU Problem，指的是某些神经元可能永远不会被激活，导致相应的参数永远不能被更新。有两个主要原因可能导致这种情况产生：(1) 非常差的参数初始化，这种情况比较少见；(2) 学习率太高导致在训练过程中参数更新太大，不幸使网络进入这种状态。解决方法是可以采用Xavier初始化方法，以及避免将学习率设置太大或使用adagrad等自动调节学习率的算法。

234、哪些模型对缺失值更敏感？哪些更不敏感？(3星)

对缺失值敏感的：

1. 涉及到距离度量 (distance measurement) 时，如计算两个点之间的距离，缺失数据就变得比较重要。因为涉及到“距离”这个概念，那么缺失值处理不当就会导致效果很差，如K近邻算法 (KNN)、支持向量机 (SVM)。
2. 线性模型的代价函数 (loss function) 往往涉及到距离 (distance) 的计算，计算预测值和真实值之间的差别，这容易导致对缺失值敏感。

对缺失值不敏感的：

1. 树模型对缺失值的敏感度低，大部分时候可以在数据缺失时使用。
2. 神经网络的鲁棒强，对于缺失数据不是非常敏感，但一般没有那么多数据可供使用。
3. 贝叶斯模型对于缺失数据也比较稳定，数据量很小的时候选贝叶斯模型。

总体来看，对于有缺失值的数据在经过缺失处理后：

- 数据量很小，朴素贝叶斯
- 数据量适中或者较大，用树模型，优先xgboost
- 数据量较大，也可以用神经网络
- 避免使用距离度量相关的模型，如KNN和SVM

235、在决策树选择分裂属性的时候，训练样本存在缺失值，如何处理？（2星）

在选择分裂属性的时候，训练样本存在缺失值，可以如下处理：**计算分裂损失减少值时忽略特征缺失的样本，最终计算的值乘以比例**（实际参与计算的样本数除以总的样本数。）

例如，假如使用ID3算法，那么选择分类属性时，就要计算所有属性的熵增(信息增益，Gain)。

假设10个样本，属性是a, b, c。在计算a属性熵时发现，第10个样本的a属性缺失，那么就把第10个样本去掉，前9个样本组成新的样本集，在新样本集上按正常方法计算a属性的熵增。然后结果乘0.9（新样本占raw样本的比例），就是a属性最终的熵。

236、决策树分类属性选择完成，对训练样本分类，发现样本属性缺失如何处理？(2星)

在属性选择完成的时候，对训练样本分类，存在缺失值，可以如下处理：

- 将该样本分配到所有子节点中，权重由1变为具有属性a的样本被划分成的子集样本个数的相对比率。
- 计算错误率的时候，需要考虑到样本权重。

比如该节点是根据a属性划分，但是待分类样本a属性缺失，怎么办呢？假设a属性离散，有1, 2两种取值，那么就把该样本分配到两个子节点中去，但是权重由1变为相应离散值个数占样本的比例。然后计算错误率。

注意，不是每个样本都是权重为1，存在分数。

237、训练完成，给测试集样本分类，有缺失值如何处理？（2星）

训练完成，给测试集分类，存在缺失值，可以如下处理：

分类时，如果待分类样本有缺失变量，而决策树决策过程中没有用到这些变量，则决策过程和没有缺失的数据一样；否则，如果决策要用到缺失变量，决策树也可以在当前节点做多数投票来决定（选择样本数最多的特征值方向）。如果有单独的缺失分支，使用此分支。

有以下几种不同的处理方法：

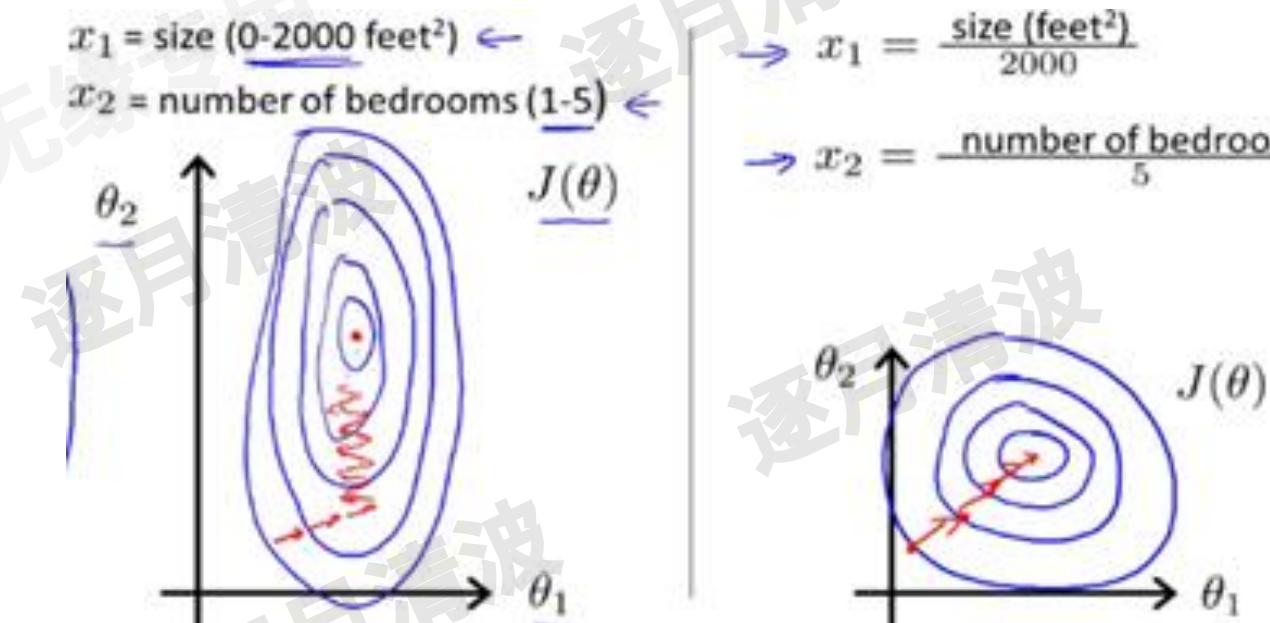
- 把待分类的样本的属性 a 值分配一个最常出现的 a 的属性值，然后进行分支预测。
- 根据其他属性为该待分类样本填充一个属性 a 值，然后进行分支处理。
- 在决策树中属性 a 节点的分支上，遍历属性 a 节点的所有分支，探索可能所有的分类结果，然后把这些分类结果结合起来一起考虑，按照概率决定一个分类。
- 待分类样本在到达属性 a 节点时就终止分类，然后根据此时 a 节点所覆盖的叶子节点类别状况为其分配一个发生概率最高的类。

238、归一化为什么能提高梯度下降法求解最优解的速度？(3星)

如图所示，蓝色的圆圈代表的是两个优化目标函数的等高线。

其中左图两个特征 X_1 和 X_2 的区间相差非常大， X_1 区间是 $[0, 2000]$ ， X_2 区间是 $[1, 5]$ ，其所形成的等高线非常尖。当使用梯度下降法寻求最优解时，很有可能走“Z字型”路线（垂直等高线走），从而导致需要迭代很多次才能收敛；

而右图对两个原始特征进行了归一化，其对应的等高线显得很圆，在梯度下降进行求解时能较快的收敛。



239、逻辑回归为什么要对特征进行离散化？(4星)

1. 非线性逻辑回归属于广义线性模型，表达能力有限，单变量离散化为N个后，每个变量有单独的权重，相当于为模型引入了非线性，能够提高模型表达力；此外，离散特征的增加和减少都很容易，易于模型的快速迭代；
2. 速度快。稀疏向量内积乘法运算速度快，计算结果方便存储，容易扩展；
3. 鲁棒性！离散化后的特征对异常数据有很强的鲁棒性：比如一个特征是年龄 >30 是1，否则0。如果特征没有离散化，一个异常数据“年龄300岁”会给模型造成很大的干扰；
4. 方便交叉与特征组合：离散化后可以进行特征交叉，由 $M+N$ 个变量变为 $M*N$ 个变量，进一步引入非线性，提升表达能力；
5. 稳定性：特征离散化后，模型会更稳定，比如如果对用户年龄离散化，20-30作为一个区间，不会因为一个用户年龄长了一岁就变成一个完全不同的人。当然处于区间相邻处的样本会刚好相反，所以怎么划分区间也是需要仔细推敲的。

模型是使用离散特征还是连续特征，其实是一个“海量离散特征+简单模型”和“少量连续特征+复杂模型”的权衡。既可以离散化用线性模型，也可以用连续特征加深度学习。就看是喜欢折腾特征还是折腾模型。

240、为什么有时候RNN训练的时候Loss波动很大？怎么解决？(3星)

这是由于RNN特有的记忆功能会影响后期其他节点的特点，导致优化时梯度时大时小，学习率没法个性化的调整，导致RNN梯度更新过程不稳定，从而影响损失函数的波动。

为了解决这个问题，可以尝试以下方法：

1. 使用**梯度剪切**：即限制梯度的范围，避免梯度爆炸。例如，限制梯度的范围在一个较小的值范围内，如 $[-5.0, 5.0]$ 。
2. 使用**变化的学习率**：尝试改变学习率的大小和衰减速度，以达到更好的更新。
3. 使用**优化器**：著名的优化器如Adam, Adadelta等。

241、对比一下逻辑回归 (LR) 与支持向量机 (SVM) 的联系与区别。 (3星)

联系：

- 1、LR和SVM都可以处理分类问题，且一般都用于处理线性二分类问题（在改进的情况下可以处理多分类问题）
- 2、两个方法都可以增加不同的正则化项，如L1、L2等。所以在很多实验中，两种算法的结果是很接近的。

区别：

- 1、LR是参数模型，SVM是非参数模型。
- 2、从损失函数来看，区别在于逻辑回归采用的是Logistical Loss，SVM采用的是hinge loss. 这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。
- 3、SVM的处理方法是只考虑Support Vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重。
- 4、逻辑回归相对来说模型更简单，好理解，特别是大规模线性分类时比较方便。而SVM的理解和优化相对来说复杂一些，SVM转化为对偶问题后，分类只需要计算与少数几个支持向量的距离，这个在进行复杂核函数计算时优势很明显，能够大大简化模型和计算。
- 5、SVM的适用范围更广：Logic能做的SVM能做，但可能在准确率上有问题，SVM能做的Logic有的做不了。

242、简要叙述一下数据清洗的基本思路。 (3星)

具体而言，数据清洗的基本思路包括以下几个步骤：

1. **去重**：对重复的数据记录进行删除或合并，使数据集中不再存在重复数据。
2. **处理缺失值**：对于缺失数据值进行处理，常用的方法有填充缺失值、删除含有缺失值的行或列和插值等方法。
3. **处理异常值**：通过数据可视化和统计学分析方法检测和识别异常值，例如基于统计、基于距离、基于密度的异常点检测算法等，对于异常值进行处理，常用的方法有删除异常值、用均值、中位数或众数代替等方法。
4. **类型转换**：将数据进行类型转换，以便于后续的分析和建模，比如将类别属性进行编码、将时间序列数据转化为时间戳等。
5. **归一化**：对不同属性间的数据量纲进行统一化，以减少不同属性对分析结果的影响，常用的方法有最大最小值归一化、Z-score标准化等。
6. **重采样**：对于数据特别大或不平衡的情况，需要进行数据采样，以确保模型训练和预测结果的可靠性和准确性。

243、简要叙述一下特征选择的基本思路。 (3星)

特征选择的基本思路是：

1. 生成子集：搜索特征子集，为评价函数提供特征子集
2. 评价函数：评价特征子集的好坏
3. 停止准则：与评价函数相关，一般是阈值，评价函数达到一定标准后就可停止搜索
4. 验证过程：在验证数据集上验证选出来的特征子集的有效性

通常来说，从两个方面考虑来选择特征：

1. 特征是否发散：如果一个特征不发散，例如方差接近于0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
2. 特征与目标的相关性：这点比较显见，与目标相关性高的特征，应当优先选择。

根据特征选择的形式，可分为三大类：

1. **Filter** (过滤法)：按照发散性（方差法）或 相关性（其他方法）对各个特征进行评分，设定阈值或者待选择特征的个数进行筛选
2. **Wrapper** (包装法)：根据目标函数（往往是预测效果评分），每次选择若干特征，或者排除若干特征
3. **Embedded** (嵌入法)：先使用某些机器学习的模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征（类似于Filter，只不过系数是通过训练得来的）

244、特征选择中Filter法主要有哪些？(2星)

1. **方差选择法**。使用方差选择法，先要计算各个特征的方差，然后根据阈值，选择方差大于阈值的特征，因为方差较小的特征往往表现不佳。
2. **Pearson相关系数法**。皮尔森相关系数是一种最简单的，能帮助理解特征和响应变量之间关系的方法，衡量的是变量之间的线性相关性，结果的取值区间为 $[-1, 1]$ ， -1 表示完全的负相关(这个变量下降，那个就会上升)， $+1$ 表示完全的正相关， 0 表示没有线性相关性。
3. **卡方检验法**，选择与目标变量显著性差异较大的特征，例如可以选择p值较小的特征。
4. **互信息法**。衡量两个变量关联性，两个变量独立则为0，关联性越高，互信息值越大。
5. **距离相关系数法**，通常距离相关系数的值越高，越能预测输入和输出之间的相关性。

245、特征选择中Wrapper法主要有哪些？（2星）

特征选择中的Wrapper方法主要包括以下几种：

1. **正向搜索**：从一个空特征集合开始，依次加入一个特征，直到达到预设的特征数或得到最佳模型为止。
2. **反向搜索**：从原始特征集合开始，依次删除一个特征，直到达到预设的特征数或得到最佳模型为止。
3. **递归式特征消除法**：先用原始特征训练出一个模型，然后根据特征的权重或重要性指标，选择其中权重或重要性最低的一个特征，将其从特征集合中删除，得到新的特征集合，继续用新的特征集合训练模型，重复上述过程，直到满足预设的特征数或得到最佳模型为止。

Wrapper法相对于其他的特征选择方法，比如Filter法和Embedded法，更加准确，但是计算量也更大，需要进行多轮模型训练和评估。因此，Wrapper法一般建议在数据集较小、特征数较少的情况下使用，或者结合特征筛选过程进行优化。

246、特征选择中Embedded法主要有哪些？(2星)

Embedded嵌入式法，故名思议，与前面两者最大的不同就是将特征选择，降维和模型训练同时完成。将特征选择嵌入到模型训练当中。常见的Embedded法主要有以下几种：

1. **Lasso回归**：在线性回归模型中，Lasso回归是一种针对过拟合的方法，它使用L1正则化来惩罚某些特征系数的绝对值，使得部分系数为0，从而达到特征选择的目的。
2. **Ridge回归**：与Lasso回归不同，Ridge回归使用L2正则化来惩罚特征系数的平方值，使得所有系数都缩小但不至于降到0。通过调节正则化参数，我们可以选择更强调特征选择或更强调模型效果。
3. **决策树**：决策树在每个节点上选择最佳特征进行切分，同时通过计算信息增益或基尼不纯度来衡量特征的重要性，进而对特征进行排序和选择。
4. **带L1正则化的逻辑回归 (L1-Logistic Regression)**：类似于Lasso回归，L1-Logistic Regression可以使用L1正则化来惩罚某些特征系数的绝对值，进而实现特征的选择。
5. **Elastic-Net**：Elastic-Net结合了L1和L2正则化，既有Lasso回归的特征选择功能，也有Ridge回归的模型优化功能，可以更加精确地选择特征以及拟合模型。

这些Embedded法都利用了模型训练过程中的特征重要性，进行特征选择和模型优化，可以在特征选择和建模两方面达到更好的效果。

247、在自助 (Bootstrap) 采样中，当采样次数趋于无穷大时，样本集的采样情况如何？(4星)

设样本集的大小为 m ，则样本在 m 次采样中始终不被采到的概率是 $(1 - 1/m)^m$ ，

当 m 趋于正无穷时，上式取极限为 $1/e \approx 0.368$ 。即通过自助采样，初始数据集中仍约有 36.8% 的样本未出现在采样数据集 D' 中。

于是我们可将 D' 用作训练集，剩下的用作测试集。这样实际评估的模型与期望评估的模型都使用 m 个训练样本，而我们仍有数据总量约 36.8% 的、没在训练集中出现的样本用于测试。这样的测试结果，亦称 "**包外估计**" (*out-of-bag estimate*)。

248、什么是P-R曲线？(4星)

P-R曲线是将分类器的精度 (Precision) 和召回率 (Recall) 之间的关系可视化的曲线。

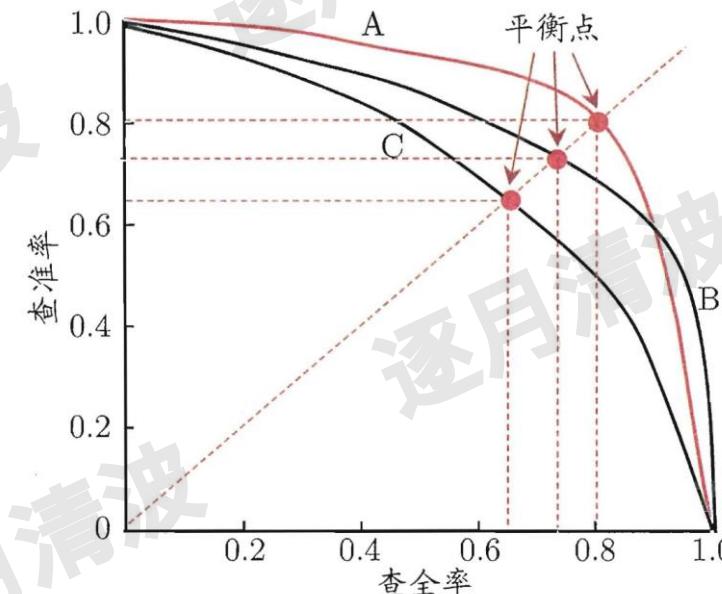
Precision 表示分类器分类为正类的样本中，真正为正类的样本所占的比例。

Recall 表示在所有真实正样本中，分类器正确识别为正类的概率。

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

精确率和召回率是既矛盾又统一的两个指标，为了提高精确率，分类器需要尽量在“更有把握”时才把样本预测为正样本，但此时往往会因为过于保守而漏掉很多“没有把握”的正样本，导致召回率降低。为了综合评估一个排序模型的好坏，不仅要看模型在不同Top N下的精确率和召回率，而且最好绘制出模型的P-R曲线。通过P-R曲线可以直观地了解分类器在不同阈值下的分类效果。同时，我们可以选择适合我们任务的阈值点作为模型的最终结果。



249、P-R曲线与ROC曲线相比有什么区别？谁的适用场景更多？(3星)

ROC曲线和P-R曲线都是用来评估二分类模型的性能指标，它们的主要区别在于以下几点：

1. **X轴和Y轴不同**: ROC曲线以“False Positive Rate (FPR)”作为X轴，以“True Positive Rate (TPR)”作为Y轴；而P-R曲线以“Recall (召回率)”作为X轴，以“Precision (精确率)”作为Y轴。
2. **样本分布不均时的处理方式不同**: ROC曲线适用于样本分布不均的情况，而且对于不同的分类阈值(threshold)情况下，FPR和TPR的变化不敏感；而P-R曲线在面临样本分布不均时，容易出现误导性的结果，因此更适合于相对均衡的二分类问题。
3. **注重范围不同**: ROC曲线注重分类器的整体性能，而P-R曲线注重在正类判断上的精度和召回率。具体来说，ROC曲线主要反映模型与随机模型之间的表现差异（因此可以用于评估模型的鲁棒性、稳定性等），而P-R曲线主要反映模型在正类判断上的表现优劣。

相比P-R曲线，ROC曲线有一个特点，当正负样本的分布发生变化时，ROC曲线的形状能够基本保持不变，而P-R曲线的形状一般会发生较剧烈的变化。所以，**ROC曲线的适用场景更多**，被广泛用于排序、推荐、广告等领域。

250、正则化可以减轻过拟合的原理是什么？(4星)

- 从先验概率角度：**无正则化的损失函数可以认为是满足关于 w 的均匀分布先验，而L1和L2正则化可以看作是满足关于 w 的拉普拉斯先验和高斯先验，所以L1和L2实际上是对 w 的取值范围做了约束，以此减少模型的方差（偏置-方差困境），从而减轻了过拟合（过拟合可以看作是模型方差过大）。
- 从病态矩阵的角度：**对于原始损失函数，其解析解为： $w = (X^T X)^{-1} X^T y$ 。这里存在两个问题，一个是 $X^T X$ 可能不可逆，说明 $X^T X$ 中有特征值为0，即 X 的维度比样本数还大。另一个问题是即使可逆，但是这个矩阵是病态的，也就是说如果 y 存在很小的波动，被逆矩阵乘了以后，结果 w 会发生很大的变化。那么这个 w 就非常的不稳定，并不是一个好的模型，即 X 的维度比样本数差不多大小。判断一个矩阵是不是病态矩阵，可以通过计算矩阵的条件数。条件数等于矩阵的最大奇异值和最小奇异值之比。如果矩阵 $X^T X$ 存在很小的奇异值，那么它的逆就存在很大的奇异值，这样对 y 中的微小变化会放大很多。加了L2正则项之后，我们的解析解变为： $w = (X^T X + \lambda I)^{-1} X^T y$ 。也就是给所有奇异值加上一个 λ ，可以确保奇异值不会太小，这样有效的解决了病态矩阵的问题。过拟合的实质可以看作由于病态矩阵的存在，如果 y 有一点波动，整个模型需要大幅度调整。解决了病态矩阵问题，就解决了过拟合。

251、比较一下决策树与逻辑回归作为分类器的区别。（3星）

1. 从特征工程角度看，对于决策树可以应对有缺失值的数据，而逻辑回归需要预先对缺失数据进行处理；逻辑回归需要做特征的归一化，决策树不需要。
2. 逻辑回归对数据整体结构的分析优于决策树，而决策树对局部结构的分析优于逻辑回归。决策树由于采用分割的方法，所以能够深入数据内部，但同时失去了对全局的把握。一个分层一旦形成，它和别的层面或节点的关系就被切断了，以后的挖掘只能在局部中进行。同时由于切分，样本数量不断萎缩，所以无法支持对多变量的同时检验。而逻辑回归始终着眼整个数据的拟合，所以对全局把握较好。但无法兼顾局部数据，或者说缺乏探查局部结构的内在机制。
3. 逻辑回归擅长分析线性关系，而决策树对线性关系的把握较差。线性关系在实践中有很多优点：简洁，易理解，可以在一定程度上防止对数据的过度拟合。
4. 逻辑回归适合高维稀疏场景，决策树数据不能太稀疏。
5. 逻辑回归对极值比较敏感，容易受极端值的影响，而决策树在这方面表现较好。
6. 执行速度：当数据量很大的时候，逻辑回归的执行速度非常慢，而决策树的运行速度明显快于逻辑回归。

252、KNN算法中K值选择大小有什么影响？如何选择K值？(3星)

1. 如果选择较小的K值，就相当于用较小的领域中的训练实例进行预测，“学习”近似误差会减小，只有与输入实例较近或相似的训练实例才会对预测结果起作用，与此同时带来的问题是“学习”的估计误差会增大，换句话说，K值的减小就意味着整体模型变得复杂，容易发生过拟合；
2. 如果选择较大的K值，就相当于用较大领域中的训练实例进行预测，其优点是可以减少学习的估计误差，但缺点是学习的近似误差会增大。这时候，与输入实例较远（不相似的）训练实例也会对预测器作用，使预测发生错误，且K值的增大就意味着整体的模型变得简单。
3. K=N，则完全不足取，因为此时无论输入实例是什么，都只是简单的预测它属于在训练实例中最多的类，模型过于简单，忽略了训练实例中大量有用信息。

在实际应用中，K值一般取一个比较小的数值，采用交叉验证法（简单来说，就是一部分样本做训练集，一部分做测试集）来选择最优的K值。

253、什么是高斯混合模型 (GMM) ? (3星)

高斯混合模型 (Gaussian Mixture Model, GMM) 是一种用于聚类和密度估计问题的生成模型。它假设每个聚类是由若干个高斯分布组合而成。具体来说，在高斯混合模型中，每个数据点被视为由一个或多个高斯分布生成的结果，因此高斯混合模型可以用于聚类问题，也可以用于密度估计问题。

高斯混合模型的核心思想是，数据可以看作从多个高斯分布中生成出来的。在该假设下，每个单独的分模型都是标准高斯模型，其均值和方差是待估计的参数。此外，每个分模型都还有一个参数，可以理解为权重或生成数据的概率。高斯混合模型的公式为：

$$P(x) = \sum_{i=1}^K \pi_i N(x|u_i, \Sigma_i)$$

GMM的训练过程和K-means类似。首先初始化所有的参数（类似于K-means初始化所有的聚类中心）。所以每次循环时，先固定当前的高斯分布不变，获得每个数据点由各个高斯分布生成的概率（类似于K-means计算每个数据点关联到最近的聚类中心，只是这里是软分类，以不同的概率分给不同的高斯分布）。然后固定该生成概率不变，根据数据点和生成概率，获得一个组更佳的高斯分布（类似于K-means调整K个聚类中心，这里是调整高斯分布的参数）。循环往复，直到参数的不再变化，或者变化非常小时，便得到了比较合理的一组高斯分布。

254、什么是表示学习？(2星)

表示学习是指通过数据中学习到数据的特征或表示的机器学习任务。它的目标是学习到一个良好的特征表示，在这个表示中数据的区别更加明显且更易于分类或聚类。这种学习方式通常在无监督或弱监督场景中使用，能够自动地从数据中学习到特征，避免了手工设计特征的繁琐以及不准确性。表示学习可以被看做是一种能够发掘数据中潜在规律的方法，在计算机视觉、自然语言处理、语音识别等领域都有广泛应用。

表示学习也可以在监督学习任务中使用。在监督学习中，我们需要将原始数据进行手工特征提取，然后再将提取到的特征传递给分类器。这种传统的方法会遇到一些问题，例如无法捕捉所有有用的特征，手工选取的特征可能不是最有用的特征，而且提取出的特征难以泛化到新的、未见过的数据样本中。为了解决这些问题，使用表示学习方法可以自动地从原始数据中提取最有用的特征，然后再将这些特征传递给分类器。

为什么有了特征工程也需要表示学习呢？因为数据量小时我们可以根据人工经验和先验知识判断用什么特征以及什么模型好，但数据一多，我们对数据的理解就相对比较浅薄，做出的假设也越来越倾向于随机，先验知识所占的分量急剧下降，那么此时人工特征工程往往是无效的，因此就需要依靠极其强大的计算力去弥补我们自身知识的不足，比如深度学习或者集成模型。

255、什么是深度学习中的网络退化问题？(3星)

1. 深度学习中的网络退化问题通常指的是，在使用非常深的神经网络训练数据时，当网络的深度继续增加时，网络的训练误差将逐渐增加，导致网络的性能反而下降的现象。这种现象是由于深层网络之间损失的梯度信息难以传递，从而导致训练过程失效所致。
2. 在使用较深的神经网络时，通过反向传播算法计算和传递梯度可能会导致梯度消失或梯度爆炸，这是深度学习中常见的问题。当梯度消失或爆炸时，神经网络的训练误差将停滞或震荡，并且神经网络性能的提高速度将变慢。这反过来会带来过度学习或模型过度拟合等问题，从而导致网络退化。
3. 梯度消失/爆炸问题导致模型训练难以收敛，这个问题很大程度上可以被标准初始化和中间层正规化方法有效控制了，这些方法使得深度神经网络可以收敛。在神经网络可以收敛的前提下，随着网络深度增加，网络表现先是逐渐增加至饱和，然后迅速下降。这一点并不符合常理：如果存在某个n层的网络是当前最优的网络，那么可以构造一个更深的网络，其最后几层仅是该网络n层输出的恒等映射(Identity Mapping)，就可以取得与浅网络一致的结果；如果n还不是所谓“最佳层数”，那么更深的网络就可以取得更好的结果。总而言之，与浅层网络相比，更深的网络的表现不应该更差。因此，一个合理的猜测就是，对神经网络来说，恒等映射 $f(x)=x$ 不容易拟合。一个直观的解释就是由于非线性激活函数ReLU的存在，每次输入到输出的过程都几乎是不可逆的，这也造成了许多不可逆的信息损失。一个特征的一些有用的信息损失了，那么恒等映射显然不容易拟合。

256、Batch Norm应该放在激活函数之前还是之后？(3星)

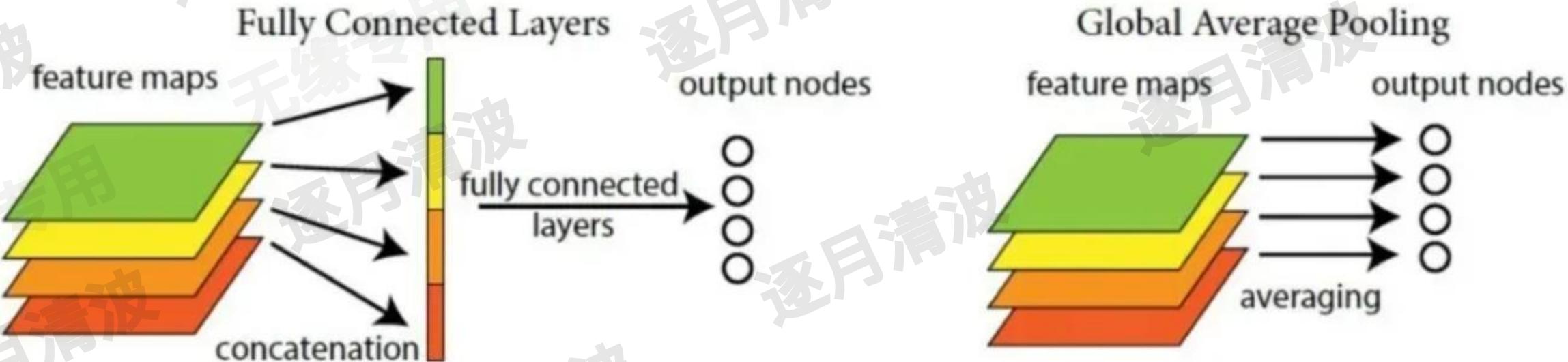
这个问题没有明确的答案。

- 把Batch Normalization放在激活层之前，可以有效避免Batch Norm破坏非线性特征的分布。另外，Batch Norm还可以使数据点尽量不落入激活函数的饱和区域，缓解梯度消失问题。
- 但是由于现在常用的激活函数是ReLU，它没有 Sigmoid、Tanh 函数的饱和问题，因此也可以把Batch Norm 放在激活层之后，避免数据在激活层之前被转化成相似的模式从而使得非线性特征分布趋于同化。
- Batch Norm的原始论文是将它放在激活层之前的，但学术界和工业界也有不少人曾表示倾向于将批量归一化放在激活层之后，从近两年的论文来看，有不少的部分是将批量归一化放在激活层之后的，Batch Norm 究竟应该放在什么位置，仍是一个存争议的问题，可以根据具体的效果来选择。

257、什么是全局平均池化 (GAP) ? (2星)

一般情况下，卷积层用于提取二维数据如图片、视频等的特征，针对于具体任务（分类、回归、图像分割）等，卷积层后续会用到不同类型的网络，拿分类问题举例，最简单的方式就是将卷积网络提取出的特征（feature map）输入到softmax全连接层对应不同的类别。

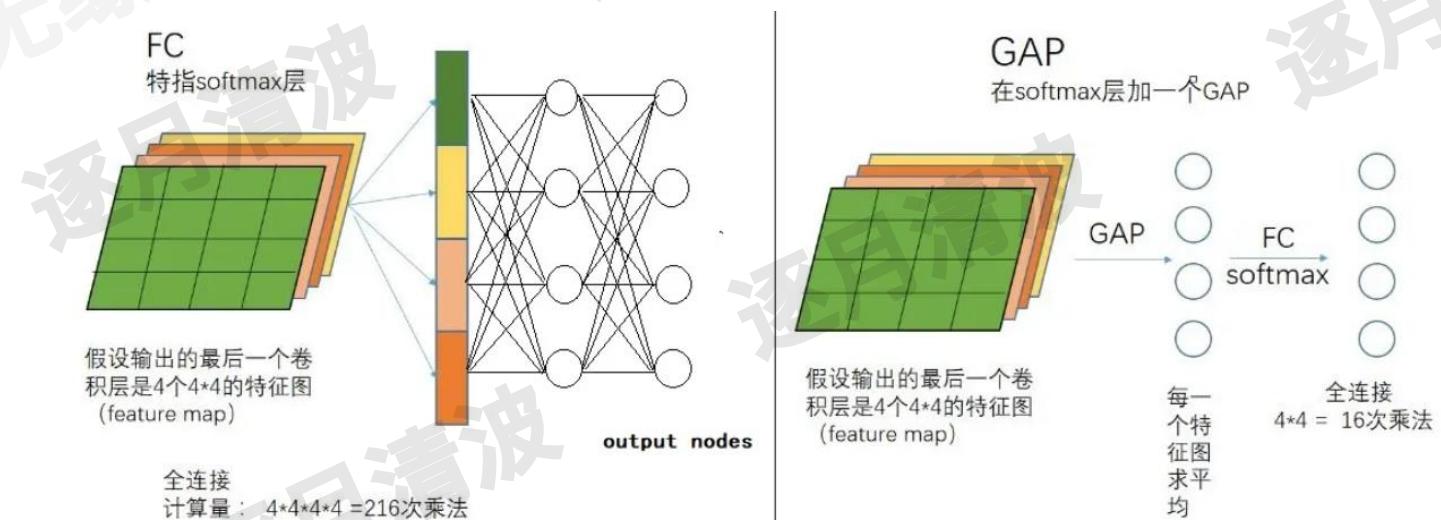
首先，这里的feature map是二维多通道的数据结构，类似于三个通道（红黄绿）的彩色图片，也就是这里的feature map具有空间上的信息；其次，在GAP被提出之前，常用的方式是将feature map直接拉平成一维向量（图左），但是GAP不同，是将每个通道的二维图像做平均，最后也就是每个通道对应一个均值（图右）。



258、全局平均池化替代全连接层有什么好处？(3星)

全局平均池化与全连接层有着相似的效果(可以提取全局信息)，且具有如下优点：

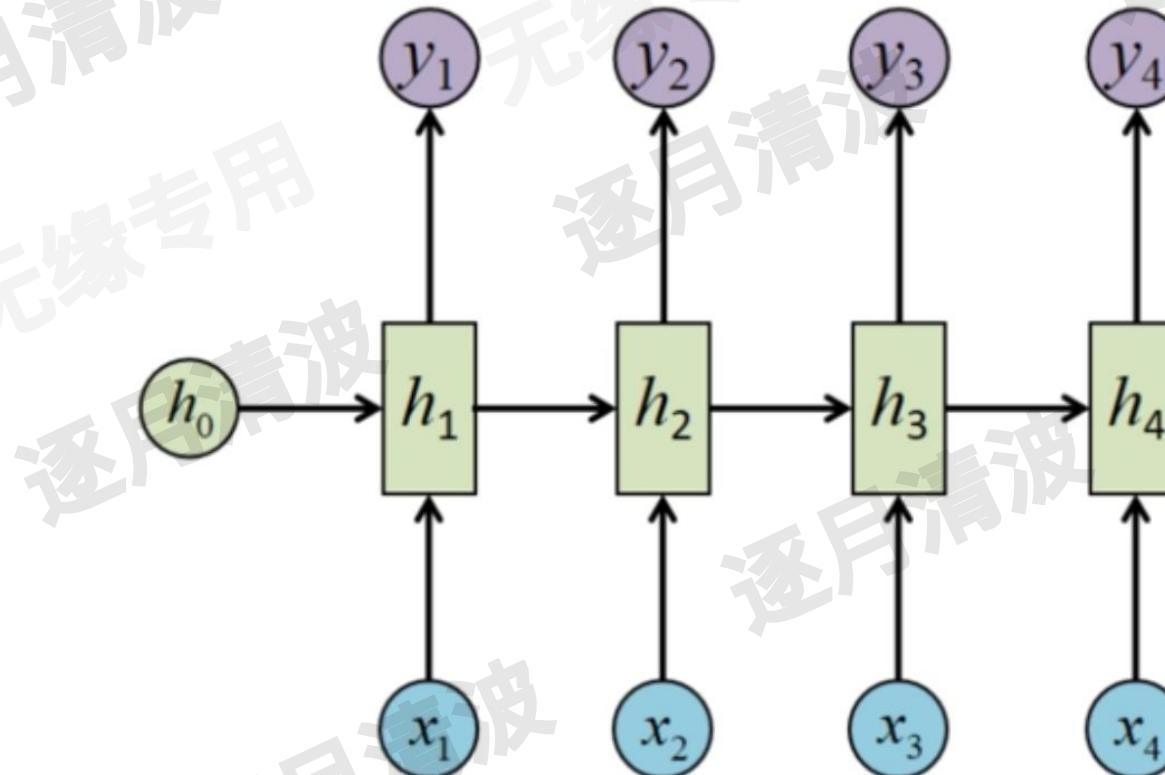
1. **抑制过拟合**。直接拉平做全连接层的方式依然保留了大量的空间信息，假设feature map是32个通道的 10×10 图像，那么拉平就得到了 $32 \times 10 \times 10$ 的向量，如果是最后一层是对应两类标签，那么这一层就需要 3200×2 的权重矩阵，而GAP不同，将空间上的信息直接用均值代替，32个通道GAP之后得到的向量都是32的向量，那么最后一层只需要 32×2 的权重矩阵。相比之下GAP网络参数会更少，而全连接更容易在大量保留下来的空间信息上面过拟合。
2. **输入尺寸更加灵活**。在第1点的举例里面可以看到feature map经过GAP后的神经网络参数不再与输入图像尺寸的大小有关，也就是输入图像的长宽可以不固定。
3. **具有较好的可解释性**。比如，我们可以知道特征图上哪些点对最后的分类贡献最大。



259、为什么说循环神经网络（RNN）是一个又浅又深的网络？（3星）

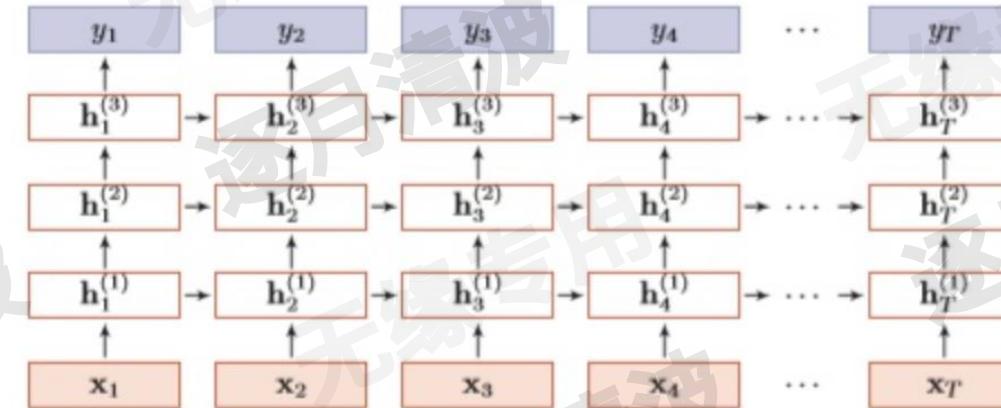
循环神经网络可以看作是既“深”又“浅”的网络。

- 一方面来说，如果我们把循环网络按时间展开，长时间间隔的状态之间的路径很长，循环网络可以看作是一个非常深的网络。
- 从另一方面来说，如果我们考察同一时刻网络输入到输出之间的路径 $x_t \rightarrow y_t$ ，这个网络是非常浅的。

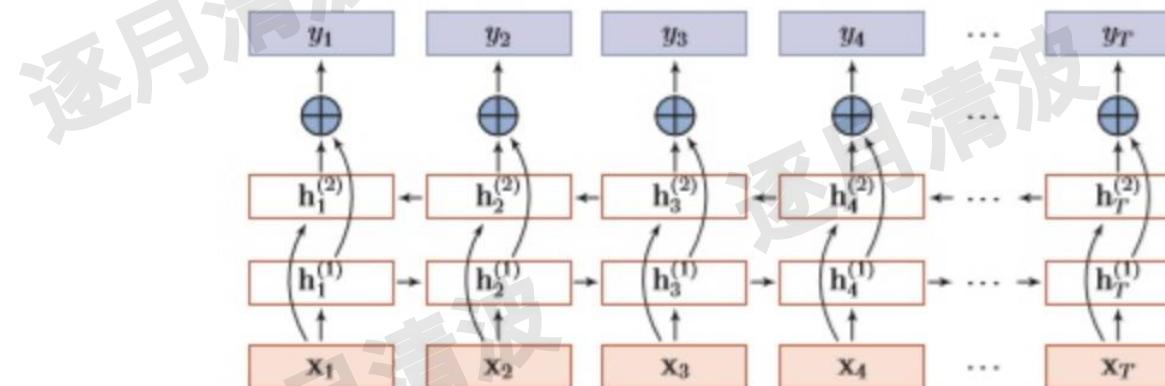


260、要想扩展RNN的深度，有哪些办法？（3星）

1. 堆叠循环神经网络。就是像传统神经网络一样，直接将多个循环网络堆叠起来。



2. 双向循环神经网络。在有些任务中，一个时刻的输出不但和过去时刻的信息有关，也和后续时刻 的信息有关。比如给定一个句子，其中一个词的词性由它的上下文决定，即包含左右两边的信息。因此在这些任务中，我们可以增加一个按照时间的逆序来传递信息的网络层，来增强网络的能力。



261、在循环神经网络（RNN）中如何使用Dropout？（4星）

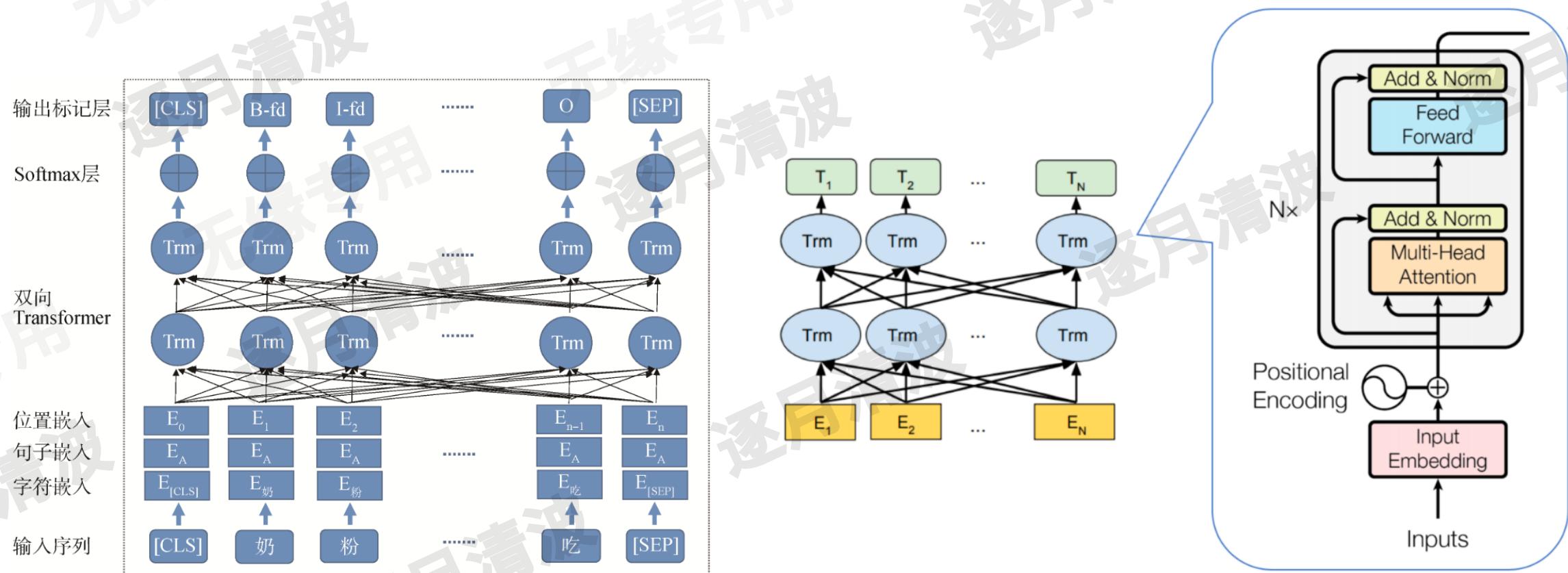
循环神经网络（RNN）具有记忆功能，其神经元的状态包含了之前时刻的状态信息，**如果直接用Dropout删除一些神经元，会导致循环神经网络的记忆能力减退。**另外有实验表明，如果在循环神经网络不同时刻间的连接层中加入噪声，则噪声会随着序列长度的增加而不断放大，并最终淹没重要的信号信息。

在循环神经网络中，连接层可以分为两种类型：一种是从 t 时刻的输入一直到 t 时刻的输出之间的连接，称为前馈连接。另一种是从 t 时刻到 $t+1$ 时刻之间的连接，称为循环连接。

如果要将Dropout用在循环神经网络上，一个较为直观的思路就是，只将Dropout用在前馈连接上，而不用在循环连接上。然而，只在前馈连接中应用Dropout对于过拟合问题的缓解效果并不太理想，这是因为循环神经网络中的大量参数其实是在循环连接中的。一种改进方法是，**对于同一个序列，在其所有时刻的循环连接上采用相同的丢弃方法**，也就是说不同时刻丢弃的连接是相同的。实验结果表明，这种Dropout在语言模型和情感分析中会获得较好的效果。

262、什么是BERT？它的结构是什么？(3星)

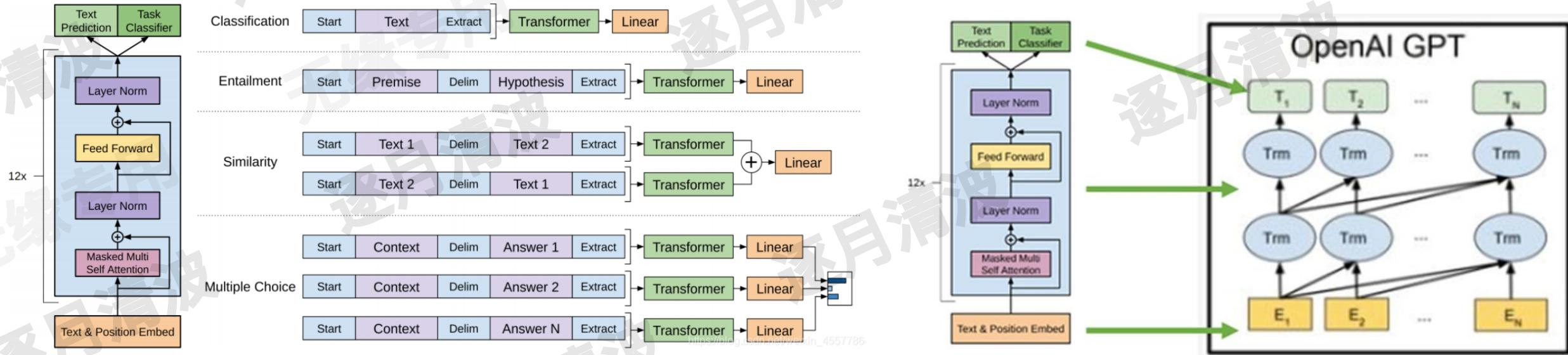
BERT的全称为Bidirectional Encoder Representation from Transformers，是一个预训练的语言表征模型，能生成深度的双向语言表征。BERT可以认为一个黑盒子，这个黑盒子的作用就是产生新的一种词向量。这种词向量实际上是对一个句子里面的各种分析各种特征提取之后的新词向量，可以用作于各种任务，例如分类、情感分析、自然语言推理等。它的结构如下。



263、什么是GPT？它的结构是什么？（3星）

GPT (Generative Pre-trained Transformer) 是一种基于Transformer模型的自然语言处理模型。GPT模型使用了大量的无监督训练数据，通过自我监督学习的方法来预训练模型，结合有监督的下游任务精调，进而可以用于各种NLP任务，例如机器翻译、语言生成、文本分类等。

GPT模型的基本结构是多层Transformer编码器。每一层都由一个多头自注意力机制和前馈神经网络组成，以实现对输入的理解和语言表示的生成。不同层之间的输出是相互叠加的，最终的输出结果是通过一个全连接输出层来生成的。



264、用过哪些深度学习框架？它们有什么优缺点？（4星）

框架	优点	缺点
TensorFlow	1. 功能很齐全，能够搭建的网络更丰富。2. 支持多种编程语言。 3. 拥有强大的计算集群。4. 谷歌支持 5. 社区活跃度高。6. 支持多GPU。 7. TensorBoard支持图形可视化。	1. 编程入门难度较大。 2. 计算图是纯 Python 的，因此速度较慢 3. 图构造是静态的，意味着图必须先被「编译」再运行
Keras	1. Keras是TensorFlow高级集成API 2. 代码更加可读和简洁。 3. Keras属于高度集成框架。4. 社区活跃。	1. Keras框架环境配置比其他底层框架要复杂一些。 2. 虽然更容易创建模型，但是面对复杂的网络结构时可能不如TensorFlow。 3. 性能方面比较欠缺。
Pytorch	1. 它可以在流程中更改体系结构。 2. 训练神经网络的过程简单明了。 3. 可以使用标准 Python 语法编写 for 循环语句。 4. 大量预训练模型	1. 不够TensorFlow全面，不过未来会弥补。 2. PyTorch部署移动端不是很好。
MXNet	1. 支持多语言。 2. 文档齐全。 3. 支持多个GPU。 4. 清晰且易于维护的代码。 5. 命令式和符号式编程风格之间进行选择。	1. 不被广泛使用。 2. 社区不够活跃。 3. 学习难度大一些。

265、解释一下卷积与互相关的区别？(3星)

卷积操作是一种线性运算，它是将卷积核和输入信号逐个相乘，并将结果相加来获得卷积特征的过程。在计算过程中，卷积核会在每个元素上翻转180度，再与输入信号的相应部分相乘，最后将所有结果相加得到输出。

卷积核对输入信号从左到右的计算顺序和对称性是卷积运算的关键特点。

而互相关操作也是将卷积核与输入信号逐个相乘，并将结果相加来获得卷积特征的过程。与卷积操作不同的是，互相关运算中卷积核在每个元素上不是翻转180度，而是按照原来的顺序进行计算。

因此，与卷积操作相比，互相关操作更加灵活，不受对称性的限制。而卷积操作则更加常见，因为卷积核的训练和设计更加容易，而且卷积操作具有平移不变性，在图像处理领域有着广泛的应用。

a	b	c
d	e	f
g	h	i

20	20	10	20	10	20	10	10	13
30	0	0	0	0	0	0	0	30
20	0	A	B	C	90	90	0	20
20	0	D	E	F	90	90	0	20
10	0	G	H	I	90	90	0	10
10	0	90	90	90	90	90	0	10
10	0	90	90	90	90	90	0	10
20	0	0	0	0	0	0	0	20
20	20	10	20	10	20	10	10	13

互相关

$$G[3,3] = a * A + b * B + c * C + d * D + e * E + f * F + g * G + h * H + i * I$$

卷积

$$G[3,3] = a * I + b * H + c * G + d * F + e * E + f * D + g * C + h * B + i * A$$

266、什么是变分推断？它有什么应用？（2星）

变分推断（Variational Inference）是一种机器学习中的概率推断方法。它的主要目标是通过最小化推断所得到的近似后验分布与真实后验分布之间的差异，来求解模型中的隐变量和参数。相比于传统的推断方法，如马尔可夫链蒙特卡洛（MCMC），变分推断通常更快、更易扩展，但是在一些情况下可能会产生较大的近似误差。变分推断在许多应用中都发挥重要的作用，比如：

1. 主题模型：变分推断被广泛应用于主题模型中，例如潜在狄利克雷分配（LDA），它可以用来学习文档集合中的主题。
2. 深度生成模型：许多深度生成模型，如变分自编码器（VAE）和生成对抗网络（GAN）都使用了变分推断。
3. 时间序列模型：变分推断也可以用于时间序列模型中，例如状态空间模型，卡尔曼滤波器和卡尔曼平滑器等。

扩展：<https://zhuanlan.zhihu.com/p/49401976>

267、CNN中除了普通的卷积层还有什么别的卷积层？(3星)

除了普通的卷积层，CNN中还有池化层、反卷积层、可分离卷积层和空洞卷积层等卷积层类型。

1. **池化层**：主要有最大池化和平均池化两种形式，用于减少特征图的尺寸以及特征的数量。最大池化层是在不改变特征数量的情况下减小特征图的空间大小，使得特征更加鲁棒、不变性更好；平均池化层也是为了减小特征图的空间大小，但会减弱特征信号之间的差异。
2. **反卷积层**：也被称为转置卷积层或上采样层，主要用于对低分辨率的特征图进行上采样，增加特征图的空间分辨率和特征数量，使得模型更具有表达能力，但会增加参数量和计算量。
3. **可分离卷积层**：将输入的特征图分为通道维和空间维，先对通道维进行逐通道卷积，再对空间维进行逐像素卷积。可分离卷积层既能减少计算量、参数量，又能保证准确率。
4. **空洞卷积层**：也被称为膨胀卷积层，是在卷积核内部插入分组的0，使得卷积核的感受野变大，能够合理处理输入图像中的长距离信息，进而增加特征图和模型表达能力。但是，空洞卷积层也会增加计算量和参数量。

268、针对Channel维度做concat操作的话，是NCHW的数据格式效率高还是NWHC的效率高？（3星）

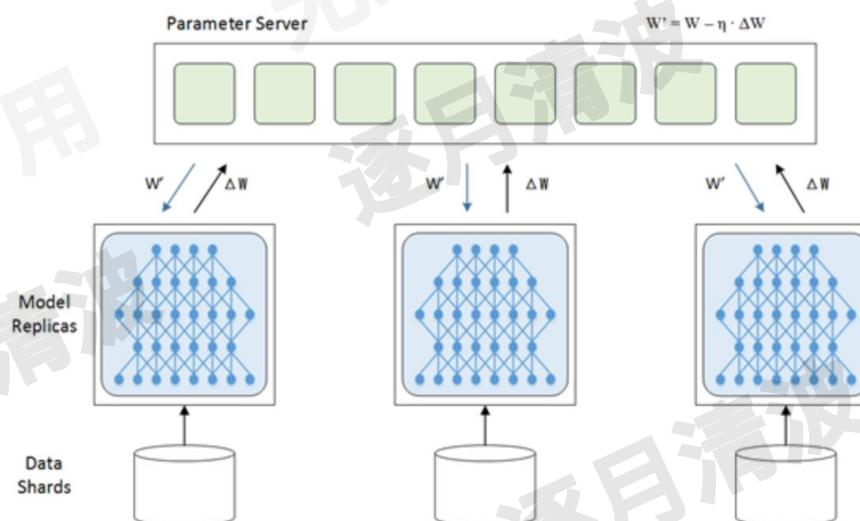
(N: batch维度, C: Channel维度, H: height, W: width)

NCHW效率高。

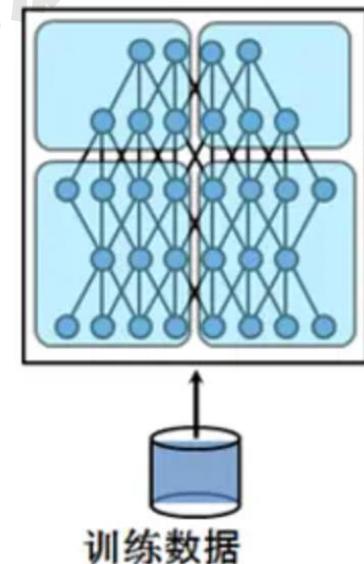
因为数据在计算机中的实际存储方式是一维线性的。要concat C维度的话，C维度在前，需要改变位置的数据比较少，

269、pytorch的并行计算中数据并行和模型并行的区别是什么？(3星)

每一个节点（或者叫进程）都有一份模型，然后各个节点取不同的数据，通常是一个batch_size，然后各自完成前向和后向的计算得到梯度，这些进行训练的进程我们成为worker，除了worker，还有参数服务器，简称ps server，这些worker会把各自计算得到的梯度送到ps server，然后由ps server来进行update操作，然后把update后的模型再传回各个节点。因为在这种并行模式中，被划分的是数据，这种方式叫数据并行。



深度学习的计算其实主要是矩阵运算，而在计算时这些矩阵都是保存在内存里的，如果是用GPU卡计算的话就是放在显存里，可是有的时候矩阵会非常大，比如在CNN中如果类别达到千万级别，那一个FC层用到的矩阵就可能会大到显存塞不下。这个时候就不得不把这样的超大矩阵给拆了分别放到不同的卡上去做计算，从网络的角度来说就是把网络结构拆了，其实从计算的过程来说就是把矩阵做了分块处理。这种方式叫模型并行：



270、深度学习训练中Batch的大小和模型泛化性的关系是什么？(4星)

- Batchsize越小，则梯度下降时的随机性越大，模型不易落入局部最优解，不易过拟合，可泛化能力增强；
- Batchsize越大，则梯度下降过程越稳定，收敛速度快，但是容易过拟合，可泛化能力减弱。

271、请解释概率和概率密度的关系。 (3星)

概率和概率密度是概率论中两个常见的概念，它们通常用于描述随机事件发生的可能性。

概率是一个事件发生的可能性，**用于离散随机变量的度量**，通常表示为一个介于0和1之间的数。例如，假设我们有一个硬币，正反面的概率相等，那么掷硬币正面朝上的概率将为0.5。

概率密度则是用于**连续随机变量**概率的度量。概率密度是一个函数，描述某个随机变量的取值在一个给定的区间内出现的频率。与离散随机变量不同，连续随机变量的概率不是通过简单的计数来计算的，而是需要使用积分。

概率密度函数可以被看作是对概率分布的描述，因为概率密度函数在连续分布中起到与概率在离散分布中相同的作用。通过概率密度函数，我们可以计算一个随机变量取某个值的概率，或者在某个区间内取值的概率。同时，概率密度函数也可以用于计算随机变量的期望值、方差等统计值。概率密度函数并不是概率，因为它可以大于1。但是，通过计算概率密度函数求得的面积总和等于1，因为它代表了所有可能取值的概率的总和。因此，如果要计算连续随机变量的概率，需要对概率密度函数进行积分。

272、请解释KL散度和交叉熵的关系，并推导公式。 (4星)

假设有两个离散概率分布P和Q，KL散度（Kullback–Leibler divergence）可以用来衡量P相对于Q的“距离”，即 $q(x)$ 能在多大程度上表达 $p(x)$ 所包含的信息，KL散度越大，表达效果越差。

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

KL散度来源于信息论，信息论的目的是以信息含量来度量数据。信息论的核心概念是信息熵(Entropy)，使用H来表示。概率论中概率分布所含的信息量同样可以使用信息熵来度量。

$$H = - \sum_{i=1}^N p(x_i) \log p(x_i)$$

当我们使用一个较简单、常见的分布(如均匀分布、二项分布等)来拟合我们观察到的一个较为复杂的分布时，由于拟合出的分布与观察到的分布并不一致，会有信息损失的情况出现。KL散度就是为了度量这种损失而被提出的。若我们使用分布 q 来表示分布 p ，那么信息熵的损失如下：

$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i)(\log p(x_i) - \log q(x_i)) = \sum_{i=1}^N p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

记忆口诀：KL散度等于交叉熵减去信息熵

273、解释极大似然估计的原理，并推导二项分布极大似然的公式。 (5星)

极大似然估计 (Maximum Likelihood Estimation, MLE) 是概率统计中的一种常用的参数估计方法。其核心思想是基于已知的观测结果，在所有可能的参数取值中，选取能够产生这些观测结果的参数值作为估计的参数值，即选择使似然函数最大的参数值。假设有样本集合 X ，其中每个样本的分布是由未知参数 θ 所确定的分布函数 $f(x; \theta)$ 生成的，其中 x 是样本观测值。那么极大似然估计就是寻找一个参数，使得观测到的样本在该参数下的概率是最大的：

$$\hat{\theta} = \arg \max_{\theta} \prod_{i=1}^n f(x_i; \theta)$$

在实际应用时，通常取似然函数的对数来求解极大值，此时就称为对数似然估计。这样做的好处是：一方面对数函数是单峰函数，在求导时更方便，另一方面也可以避免乘积中出现下溢或上溢导致计算精度降低。

以二项分布为例，参数为 p ，估计样本的概率为： $P(x_1, x_2, \dots, x_n | p) = \prod_{i=1}^n P(x_i | p) = \prod_{i=1}^n p^{x_i} (1-p)^{1-x_i}$

那么对数似然函数为：
$$l(p) = \sum_{i=1}^n x_i \log p + (1 - x_i) \log(1 - p)$$

求导得到：
$$\frac{\partial}{\partial p} l(p) = \frac{\sum_{i=1}^n x_i}{p} - \frac{\sum_{i=1}^n (1 - x_i)}{1 - p}$$

令其为0，解得：
$$\hat{p} = \frac{\sum_{i=1}^n x_i}{n}$$

274、请解释对抗生成网络中的两个阶段。 (2星)

对抗生成网络 (GAN, Generative Adversarial Networks) 是一种深度学习模型，可以生成具有真实感的数据样本。GAN主要由两个部分组成：生成器 (Generator) 和判别器 (Discriminator) 。GAN的两个阶段如下：

1. **生成器阶段**: 在这个阶段，生成器生成和真实数据相似的数据样本。生成器接收一个随机噪声作为输入，并使用神经网络从该噪声中生成假的数据样本。生成器经过多次训练后，应该能够生成越来越接近真实数据分布的数据样本。**这一阶段判别器的参数锁定。**
2. **判别器阶段**: 在这个阶段，判别器使用神经网络来区分生成器生成的假数据和真实数据。判别器的目标是准确识别出哪些数据是真实的，哪些是生成器生成的。判别器经过多次训练后，应该能够准确地区分真实数据和生成器生成的数据。**这一阶段生成器的参数锁定。**

在这两个阶段中，生成器和判别器都在互相竞争中提高自己的性能。通过不断的迭代训练，生成器不断改进生成数据的能力，判别器也不断提高自己的准确率。这样，GAN可以逐步学习到真实数据的分布特征，生成具有真实感的数据。

275、BERT的预训练任务有几个？分别是什么？（3星）

BERT是一种基于Transformer的预训练模型，它的预训练任务主要有两个：

1. **Masked Language Model (MLM)**：随机从输入序列中选取一部分作为被生成词的填充，然后模型需要预测这些被遮盖的词。
2. **Next Sentence Prediction (NSP)**：检测两个句子是否是连续的。具体地，模型需要判断两个句子是否是相邻的，并标记它们之间的关系为“`is_next`”或“`not_next`”。

这两个预训练任务可以让BERT学习到语言的表示能力，尤其是上下文相关的语境信息，从而在下游各种自然语言处理任务中取得较好的效果。

276、为什么分类任务不用MSE loss优化而用交叉熵loss优化？(4星)

参考问题55。

正如均方误差损失是正态分布的极大似然估计一样，交叉熵损失是多项分布的极大似然估计。

作为练习，可以尝试推导该极大似然估计的过程。

277、什么是JS散度？它和KL散度的区别是什么？(3星)

JS散度（Jensen-Shannon divergence）是衡量两个概率分布相似度的度量工具，它是由Jensen不等式推导而来的，由于具有非负性、对称性和三角不等式等良好的性质，因此常被用于聚类、分类、信息检索等领域。一般而言，假设有连续的概率分布 $P(x)$ 和 $Q(x)$ ，JS散度的计算如下：

$$JS(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$
$$M = \frac{1}{2}(P + Q)$$

KL散度（Kullback-Leibler divergence）也被称为相对熵，它也是一种衡量两个概率分布距离的度量工具。与JS散度不同的是，KL散度是不对称的，即 $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ 。它的计算方式如下：

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

因此，JS散度相对于KL散度具有平衡性。

278、写出平衡交叉熵函数 (balanced cross entropy) 的公式。 (3星)

普通交叉熵损失函数： $Cross_Entropy = L = -y \log(\hat{p}) - (1 - y) \log(1 - \hat{p})$

对于二分类问题，损失函数可以写为： $L = \frac{1}{N} \left(\sum_{y_i=1}^m -\log(\hat{p}) + \sum_{y_i=0}^n -\log(1 - \hat{p}) \right)$

其中 m 为正样本个数， n 为负样本个数。

基于样本非平衡造成的损失函数倾斜，一个直观的做法就是在损失函数中添加权重因子，提高少数类别在损失函数中的权重，平衡损失函数的分布。如在上述二分类问题中，添加权重参数 α (取值 0 至 1 之间) :

$$Balanced_Loss = \frac{1}{N} \left(\sum_{y_i=1}^m -\alpha \log(\hat{p}) + \sum_{y_i=0}^n -(1 - \alpha) \log(1 - \hat{p}) \right)$$

其中 $\frac{\alpha}{1 - \alpha} = \frac{n}{m}$

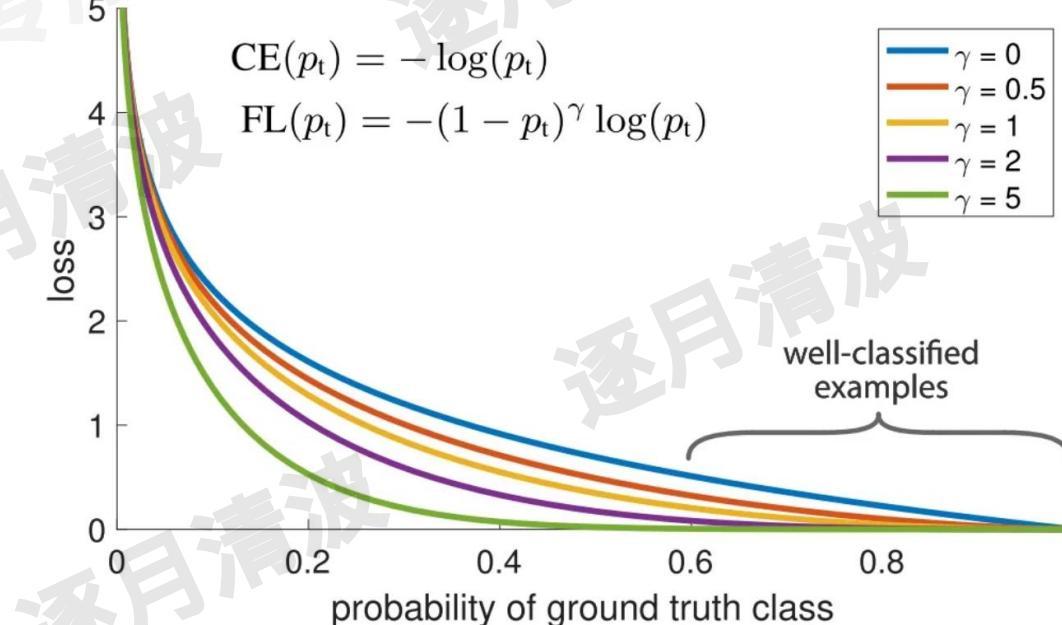
279、什么是focal loss？写出它的公式。（3星）

Focal Loss是一种用于解决类别不平衡问题的损失函数，它给予网络显著错误样本更大的惩罚，以提高模型的分类性能。Focal Loss的公式如下：

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

其中，当 $y=1$ 时， $p_t = \hat{p}$ 。当 $y=0$ 时， $p_t = 1 - \hat{p}$ 。

当 $\gamma=0$ 时，Focal Loss退化为交叉熵损失。 γ 的值越大，越会减小易分类样本的权重，增加难分类样本的权重，从而更加关注难以分类的样本，提高模型分类能力。Focal Loss已被广泛应用于目标检测和图像分割等计算机视觉任务中，取得了较好的效果。



280、什么是对比学习？如何进行对比学习？(2星)

对比学习 (Contrastive Learning) 是指通过对两个不同样本之间的差异性进行比较，来学习数据的表征。简而言之，对比学习就是让模型学习如何区分不同的数据样本或数据集。例如，在非监督学习中，我们往往需要在没有标签的数据上进行训练，而这部分数据往往比较难以直接进行建模。而对比学习则可以通过比较两个样本的相似性或差异性，来学习数据的表征，从而解决无标签数据难以建模的问题。

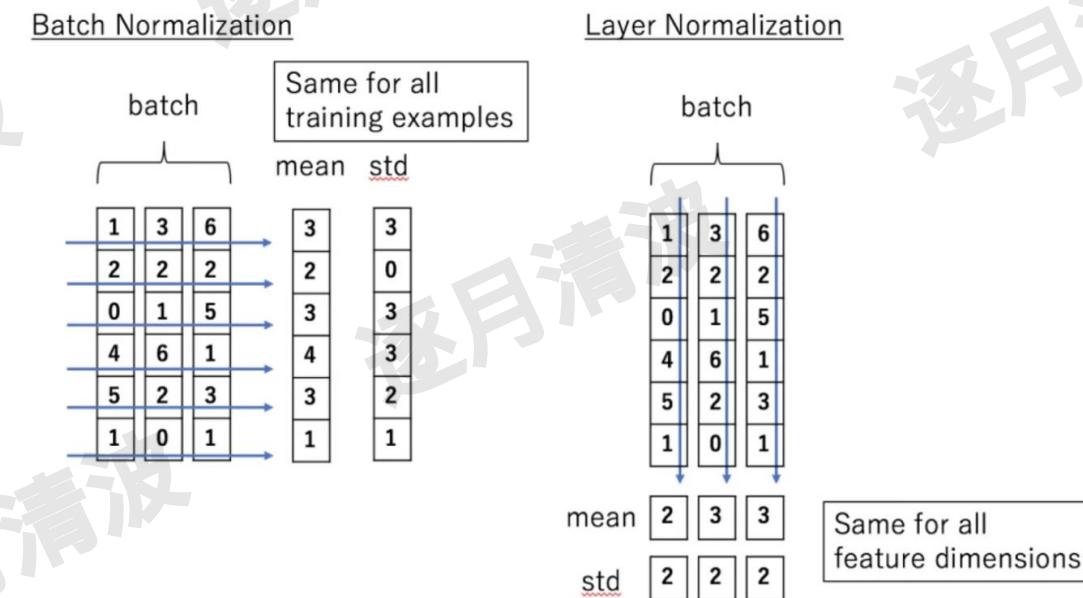
对比学习的目的是让模型学习哪些样本是相似的，哪些样本是不相似的。模型不需要知道样本所属的类别，只需要知道样本之间是否相似就可以了。也就是说，假如现在有三张图 x_1, x_2, x_3 ，其中 x_1, x_2 相似， x_3 与 x_1, x_2 不相似，三张图片在经过编码器编码后得到三个特征 f_1, f_2, f_3 ，此时模型就需要让 f_1, f_2 尽量接近，而让 f_3 与 f_1, f_2 尽量远离。但没有标签信息，又如何知道样本之间是否相似进而提供监督信号去训练模型呢？这一般是通过使用代理任务 (pretext task)，人为定义一些规则，用于定义哪些图片是相似的，哪些图片是不相似的来完成的。代理任务多种多样，例如在个体判别 (instance discrimination) 中，我们从数据集中随机选取一张图片，对该图片做随机裁剪并进行数据增强得到两张新图，由于它们都是从同一张图片得到的，语义信息应该相近，因此可以将它们视为相似的正样本，数据集中的所有其他样本都是不相似的负样本，这就相当于将数据集中的每一个样本都看作一个单独的类别。在得到正样本和负样本之后，利用对比损失 (contrastive loss) 函数使得相似样本的特征尽量接近，不相似样本的特征尽量远离。

281、Transformer中使用什么Normalization？为什么？（3星）

Transformer里面实际使用Layer Normalization。

- 首先，Batch Norm在mini-batch较小的情况下不太适用。BN是对整个mini-batch的样本统计均值和方差，当训练样本数很少时，样本的均值和方差不能反映全局的统计分布信息，从而导致效果下降。
 - 循环神经网络实际是共享的多层感知机，在时间维度上展开。由于不同句子的同一位置的分布大概率是不同的，所以应用BN来约束没有意义。BN可以用在CNN的原因是同一个channel的特征图由同一个卷积核产生的。
 - NLP模型在训练时，对BN来说需要保存每个step的统计信息（均值和方差）。在测试时，由于变长句子的特性，测试集可能出现比训练集更长的句子，所以对于后面位置的step，是没有训练的统计量使用的。（实践中的操作主要都是固定了最大长度，然后padding。）
 - 不同句子的长度不一样，对所有的样本统计均值无意义，因为某些样本在后面的时刻其实是padding。

实践过程中发现LN的效果还很不错，比BN好，所以就变成自然语言处理的默认方式了。

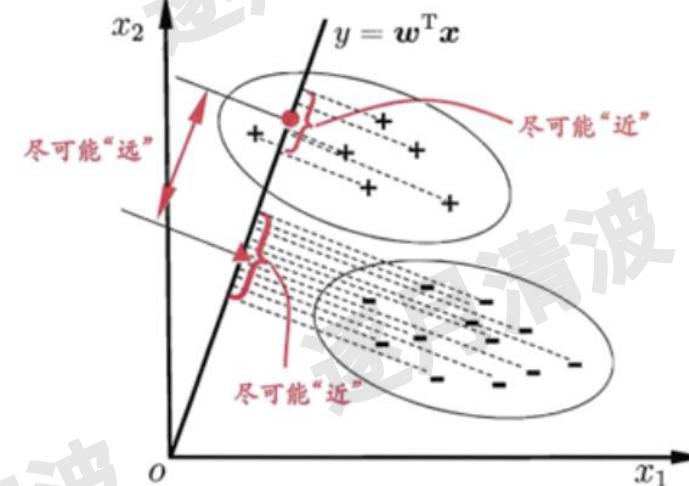


282、如何对高维数据进行降维？如果高维数据有标签的话呢？（3星）

常用的降维方法有主成分分析（PCA）、线性判别分析（LDA）、t-SNE等。

如果数据无标签，则可采用PCA（参考75题），它是一种无监督的降维方法，用于减少数据的维度，同时保留数据的最大方差。

如果高维数据有标签，那我们可以使用有监督的降维方法，如LDA。LDA的基本思想是：给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近、异类样例的投影点尽可能远离，在对新样本进行分类时，将其投影到同样的这条直线上，再根据投影点的位置来确定新样本的类别。

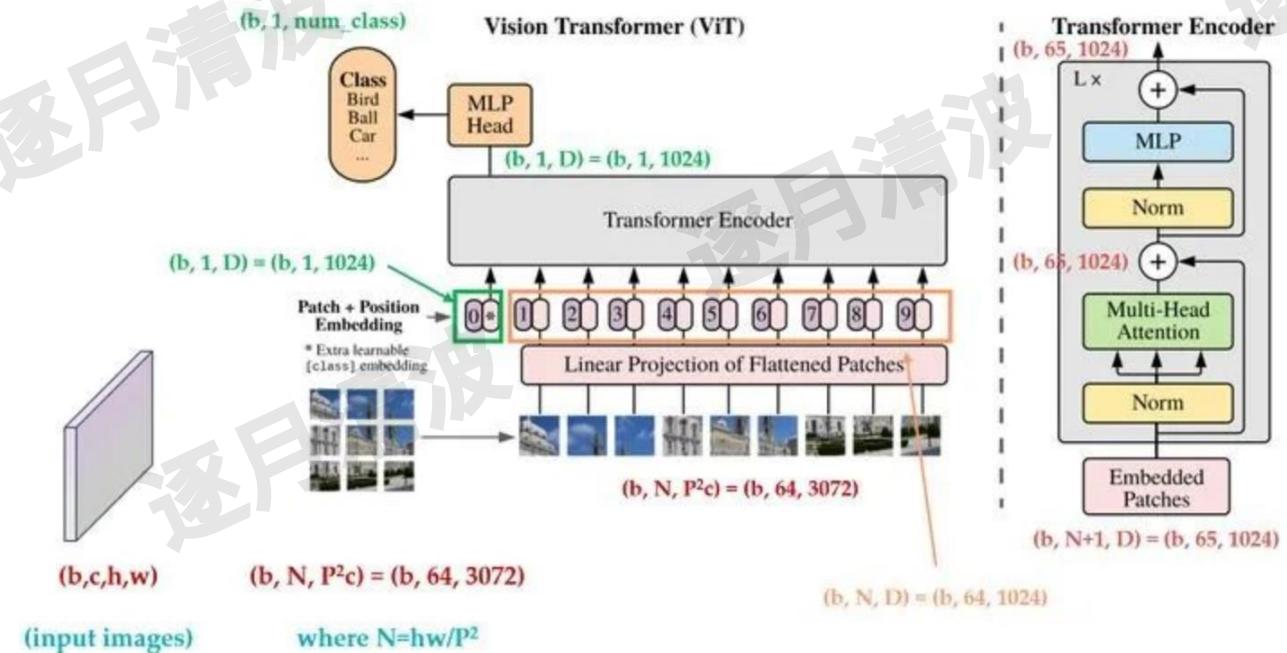


283、什么是ViT？它的结构是什么样的？（3星）

ViT代表Vision Transformer，是一种基于Transformer结构的图像分类模型，它是将NLP中常用的Transformer网络应用于图像分类的先例，通过将输入图像的像素视为序列来处理。

ViT的网络结构是由多个Transformer Block串联而成的。每个Transformer Block中包含两个子层：注意力子层和全连接子层。注意力子层主要负责捕捉输入特征之间的关系，而全连接子层则用于对特征进行线性变换和非线性变换，从而提取出高层次的特征表示。同时，ViT还引入了一个预处理阶段，即将输入图像分成若干个固定大小的小块(patch)并进行嵌入操作，从而得到固定长度的向量序列，这些序列被送入Transformer模型中进行处理。

总体来说，ViT的结构可以被表示为一个以Transformer Block为基本单元的编码器-解码器框架，其中编码器部分用于学习特征表示，而解码器部分则用于进行图像分类。



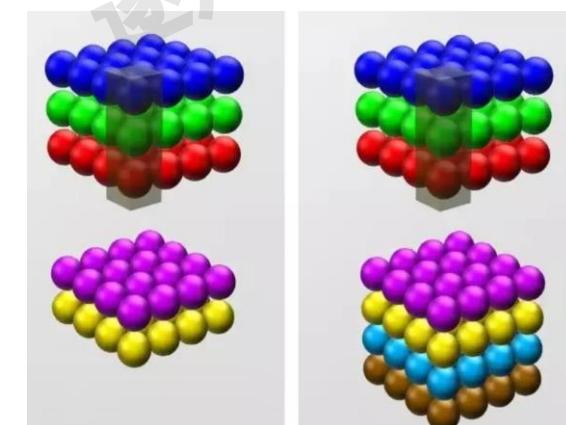
284、CNN中 1×1 卷积核是什么？有什么作用？(3星)

1. **增加网络深度**（增加非线性映射次数）：首先直接从网络深度来理解， 1×1 的卷积核虽小，但也是卷积核，加一层卷积，网络深度自然会增加。卷积核越大，它生成的 featuremap 上单个节点的感受野就越大，随着网络深度的增加，越靠后的 featuremap 上的节点感受野也越大。因此特征也越来越抽象。有的时候，我们想在不增加感受野的情况下，让网络加深，为的就是引入更多的非线性。 1×1 卷积核可以在保持 feature map 尺度不变的（即不损失分辨率）的前提下大幅增加非线性特性（利用后接的非线性激活函数），把网络做得很深。
2. **升维/降维**：这里的升维、降维具体指的是通道数的变化，当我们确定了卷积核尺寸后，我们的 height、width 都不变，那么这里的维度具体指的就是 channels。我们通过改变卷积核的数量来改变卷积后特征图的通道 channels 来实现升维、降维的效果。
3. **通道间信息共享**： 1×1 卷积核只有一个参数，当它作用在多通道的 feature map 上时，相当于不同通道上的一个线性组合，实际上就是加起来再乘以一个系数，但是这样输出的 feature map 就是多个通道的整合信息了，能够使网络提取的特征更加丰富。使用 1×1 卷积核，实现降维和升维的操作其实就是 channel 间信息的线性组合变化。
- 4.

$$\begin{matrix} 1 & 2 & 3 & 6 & 5 & 8 \\ 3 & 5 & 5 & 1 & 3 & 4 \\ 2 & 1 & 3 & 4 & 9 & 3 \\ 4 & 7 & 8 & 5 & 7 & 9 \\ 1 & 5 & 3 & 7 & 4 & 8 \\ 5 & 4 & 9 & 8 & 3 & 5 \end{matrix} \quad 6 \times 6 \times 1$$

$$* \quad \boxed{2} \quad =$$

$$\begin{matrix} 2 & 4 & 6 & \dots \\ \hline & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \end{matrix}$$

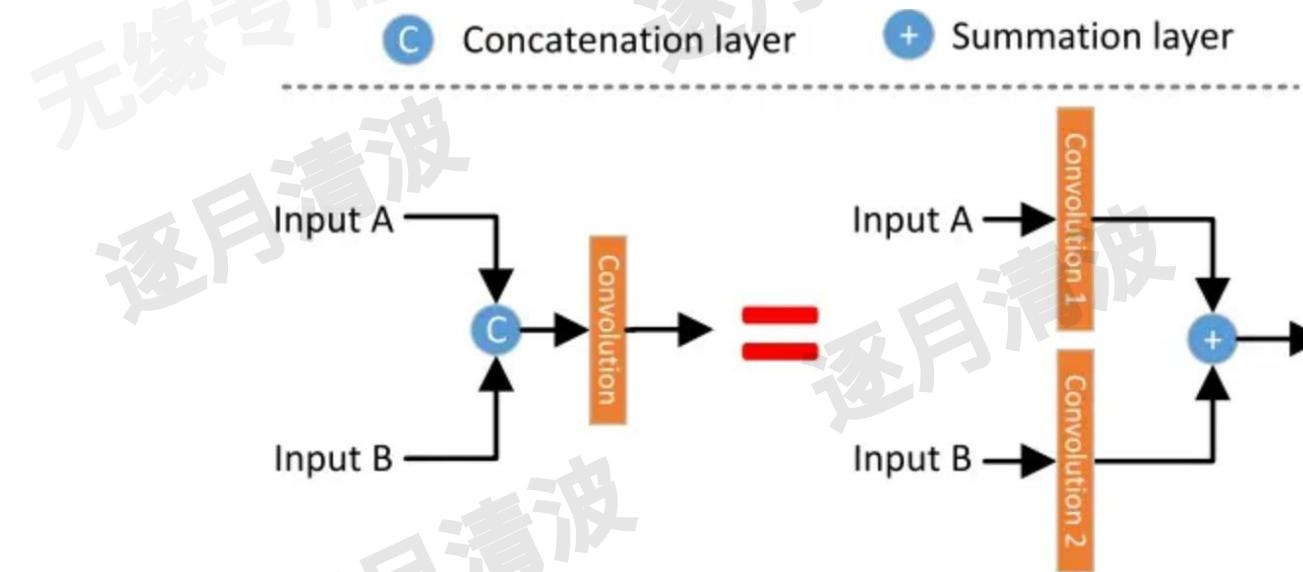


285、神经网络中不同特征之间concat和add有什么联系？(4星)

对于两路输入来说，如果是通道数相同且后面带卷积的话，**add等价于concat之后对应通道共享同一个卷积核**。

因此add相当于加了一种“先验”，当两路输入可以具有“对应通道的特征图语义类似”的性质的时候，可以用**add来替代concat**，这样更**节省参数和计算量**（concat是add的2倍）。

从另一方面来说，concat会消耗更多计算量和参数，但concat避免了**直接add对信息造成负面影响**。而且逐元素加和的方式要求不同层的feature map具有完全一致的channel数量，而concat不受channel数量的限制。



286、什么是消融实验 (Ablation experiment) ?有什么作用? (4星)

消融实验是指在机器学习系统中，人为地削减或者修改某些模块或变量以观察系统的性能变化。消融实验常用于解释某些部分对于模型的表现影响程度的分析中，以验证某些因素对于模型性能的重要性。消融实验就是控制变量法的别称。

消融实验对于分析模型的复杂性和识别关键因素很有用。通过消融实验可以验证某些因素对模型性能的影响，比如当去掉一个特征时，模型性能是否会下降；或者当修改模型架构时，模型性能是否会有提高等。这些实验可以帮助我们进一步理解模型的工作原理，并更好地解释模型的表现。

消融实验并不能提供完整的模型解释，仅能为我们提供一些关于模型行为的初步了解。在进行消融实验时，需要避免过度消融和过分依赖结果，尽可能采用多种实验方式以增强范围和可解释性。

287、神经网络不同层的权重是否以不同的速度收敛？(3星)

神经网络训练过程中，不同层的权重确实可能会以不同的速度进行收敛。

这是因为在基于梯度下降算法的优化器中，每个权重参数都会根据其对损失函数的贡献程度来进行更新，而不同层的权重对应着不同的特征提取和抽象程度，因此对损失函数的影响程度也不同，从而导致不同层的权重可能会以不同的速度更新。

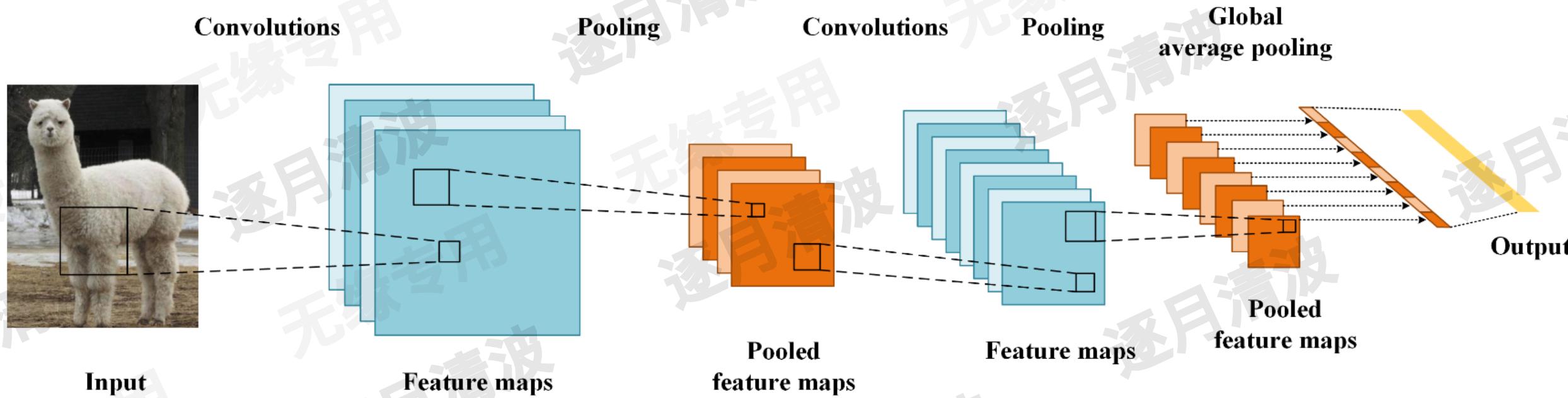
例如，靠近输出层的权重对输出误差的影响更大，因此可能会更快地进行更新。而靠近输入层的权重则需要更多的训练数据和迭代次数才能进行较好的训练，从而收敛到最优解。

一些优化算法，比如动量随机梯度下降 (SGD with Momentum) 等也可以通过积累历史梯度信息来加速收敛速度，从而缓解不同层之间的收敛速度差异。

288、当图像尺寸变为 2 倍，CNN 的参数数量变为几倍？(4星)

这是专门设置误导性的问题，因为大多数人会把注意力放在 CNN 参数的数量会增加多少倍的问题上。

然而，根据 CNN 的架构：



CNN 模型的参数数量取决于滤波器的数量和大小，而不是输入图像。因此，将图像的大小加倍并不会改变模型的参数数量。只需要调整输入的size符合模型参数个数就可以了。

289、宽而浅的模型与窄而深的模型哪个更好？(2星)

我们可以把神经网络看作是不断提取特征并抽象概括的过程。更深的网络，有更好的非线性表达能力，可以学习更复杂的变换，从而可以拟合更加复杂的特征。因此，提升同样效果需要增加的宽度远远超过需要增加的深度，从这一方面来说，**网络的深度更加重要**；

从另一方面来说，宽而浅的网络(Wide)可能比较擅长记忆，窄而深的网络(Deep)擅长概括即泛化能力。因此，宽网络适合与简单特征的数据，深网络适合更加复杂的应用情形。

深度和宽度并不是完全对立的关系，增加深度和宽度都是在增加可学习参数的个数，从而增加神经网络的拟合能力，在网络设计时，两者都会考虑，追求深度与广度的平衡。增加网络深度可以获得更大的感受野来帮助捕获更多像素点的类似特征，增加网络宽度可以获得更细粒度、更丰富的特征。

290、PyTorch中的clone与detach分别是什么？(2星)

在PyTorch中，`clone()`和`detach()`都是返回一个新的tensor，但是它们的功能有所不同。

- `clone()`方法开辟新内存空间，其值和源张量一样，参与计算图中梯度计算并可将梯度信息传递到源张量进行累积，但通过`clone`方法得到的张量其只是一个中间值，梯度为None。
- `detach()`方法得到一个和源张量共享内存，但没有梯度信息的张量，不参与计算图的梯度计算。

`clone`提供了非数据共享的梯度追溯功能，而`detach`又“舍弃”了梯度功能，因此`clone加上detach`意味着只做简单的数据复制，既不数据共享，也不对梯度共享，从此两个张量无关联。

先`clone`还是先`detach`，其返回值一样，一般采用`tensor.clone().detach()`。

291、为何batchsize通常设置成2的幂次（例如32、64、128）？（2星）

batch_size表示单次传递给程序用以训练的数据（样本）个数。

如果我们的数据集钟含有的样本总数为12800个样本，batch_size=128，那么就需要10个batch才能够训练完一个epoch。

batch_size一般取值为2的N次幂的形式，是因为CPU或者GPU的内存架构是2的N次幂。CPU在读取内存时是一块一块进行读取的，块的大小可以是2, 4, 8, 16（总之是2的倍数）。因此，选取2的n次幂作为batch大小，主要是为了将一个或多个批次整齐地安装在一个页面上，以帮助GPU并行处理。

但事实上，在工程实践中，这更多地是约定俗成的一个规矩。经过大量实验表明，事实上，不是2次幂的batch_size对速度和结果影响并不大（可能会慢一点点，但是几乎可以忽略不记）。

292、在PyTorch训练神经网络的过程中如何冻结某些层的参数不变？(2星)

- 首先使用`net.named_parameters()`方法获取网络中每一层的名字
- 对于要冻结的层，将该层的`requires_grad`参数设置为`False`:

```
NET = NET.CTPN() # 获取网络结构
FOR NAME, VALUE IN NET.NAMED_PARAMETERS():
    IF NAME IN NO_GRAD:
        VALUE.REQUIRES_GRAD = FALSE
    ELSE:
        VALUE.REQUIRES_GRAD = TRUE
```

- 在定义优化器时，设置只对`requires_grad`为`True`的层进行更新:

```
optimizer = optim.Adam(filter(lambda p: p.requires_grad, net.parameters()))
```

293、在PyTorch中torch.no_grad()有什么作用？(2星)

在PyTorch中，当我们执行一些不需要梯度计算的操作时，我们可以通过将代码包裹在torch.no_grad()上下文管理器中来减少内存的消耗并提高代码执行效率。

```
model.eval()
with torch.no_grad():
    for inputs, targets in validation_loader:
        outputs = model(inputs)
        # 计算指标，如准确率、损失等
```

具体来说，`torch.no_grad()`是一个上下文管理器，它会在执行被包裹的代码块时关闭PyTorch张量的自动求导功能。这意味着，被包裹的代码块中的所有计算操作都不会被记录在PyTorch的计算图中，从而减少了计算图的大小，也避免了不必要的内存消耗。

在eval验证模型、通过优化器更新参数、以及其他确定不需要计算梯度的情况下，使用`torch.no_grad()`可以节省计算资源并提高运行效率。

294、Pytorch中带有下划线（_）的方法有什么含义？（2星）

在方法名后带下划线的函数的作用：对Tensor进行in-place=True的修改。

【in-place=True：在一个tensor上操作后，直接修改这个tensor本身】

【in-place=False：在一个tensor上操作后，不修改这个tensor本身，生成一个修改后新的Tensor】

例如：

```
a = torch.tensor([2, 4, 6])
b = a.add(10)
a is b
```

False, 会创造一个新的张量，分配新的内存

```
a = torch.tensor([2, 4, 6])
b = a.add_(10)
a is b
```

True, 不会创造一个新的张量，不分配新的内存

295、介绍一下torch.nn.Conv2d层中各参数的含义。 (2星)

torch. nn. Conv2d

(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')

- **in_channels**: 输入通道数目
- **out_channels**: 输出通道数目
- **kernel_size**: 卷积核大小，如果输入是一个值，比如3，那么卷积核大小就是 3×3 ，如果不想要卷积核宽和高相等，还可以输入tuple类型数据，比如：(3, 5)
- **stride**: 步长大小，跟上面卷积核参数一样，如果输入是一个值，比如2，步长就是 2×2 ，还可以输入(2, 1)，表示卷积核每次向右移动1个步长，向下移动2个步长。
- **padding**: 填充，参数表示在周围补0的情况。补0的方向为上、下、左、右四个方向。如果是输入是单个值，比如1，就是在上下左右四个方向补一圈0。如果输入是元组比如(2, 1)，表示在上下方向各补两行0，在左右方向各补一列0。
- **dilation**: 进行扩展卷积需要的参数。
- **groups**: 进行分组卷积需要的参数。
- **bias**: 偏置，布尔类型，默认为True，即增加一个学习的偏置项。
- **padding_mode**: 填充的模式，默认是zero，还可以选择reflect、replicate、circular。

296、介绍一下torch.optim.Adam优化器参数的含义。 (2星)

```
torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0)
```

- **params (iterable)**: 待优化参数的 iterable 或者是定义了参数组的 dict
- **lr (float, 可选)**: 学习率 (默认: $1e-3$)
- **betas (Tuple[float, float], 可选)**: 用于计算梯度以及梯度平方的运行平均值的系数 (默认: 0.9, 0.999), beta1是一阶矩阵的指数衰减率, beta2是二阶矩阵的指数衰减率, 该超参数在稀疏梯度 (如在NLP 或计算机视觉任务中) 应该设置为接近1的数
- **eps (float, 可选)**: 为了增加数值计算的稳定性而加到分母里的项, 防止出现除以0 (默认: $1e-8$)
- **weight_decay (float, 可选)**: 权重衰减 (L2惩罚) (默认: 0)

297、PyTorch中torch.Tensor和torch.tensor有什么不同？（2星）

Tensor和tensor都可以用于生成新的张量，它们之间的区别在于：

- **Tensor**是一个类，是FloatTensor的别名，会调用构造函数，生成float类型的张量；
- **tensor**仅仅是一个纯函数，根据指定的数据类型分别构造生成不同的LongTensor、FloatTensor、DoubleTensor等。

因此，`torch.tensor(1)`把1当成一个整型的value传入，返回一个值为[1]的张量；

而`torch.Tensor(1)`把1当成张量的size传入，生成一个1维长度为1的张量，值是随机生成的或为0。

不能通过`tensor()`来创建空张量，可以用`Tensor()`来创建空张量。

298、解释一下PyTorch中train和eval两种模式的区别。（2星）

在使用 pytorch 构建神经网络的时候，训练过程中会在程序上方添加一句`model.train()`，作用是启用batch normalization和dropout。

`model.train()`保证BN层能够用到每一批数据的均值和方差。对于Dropout，`model.train()`随机取一部分网络连接来训练更新参数。

`model.eval()`的作用是不启用Batch Normalization和Dropout。

`model.eval()`保证BN层能够用全部训练数据的均值和方差，即测试过程中要保证BN层的均值和方差不变。对于Dropout，`model.eval()`利用到了所有网络连接，即不进行随机舍弃神经元。

299、PyTorch里nn.functional中的函数和nn.Module主要有什么区别？(2星)

这二者能够实现一些网络层的定义，对于nn中大多数的layer，在nn.functional中都有对应的函数算子。它们之间的区别是：

- nn.Module实现的layers是一个特殊的类，都是由class layer(nn.Module)定义，会自动提取可学习的参数
- nn.functional中的函数更像是纯函数，由def function(input)定义。

也就是说如果模型有可学习的参数，最好用nn.Module否则使用哪个都可以，二者在性能上没多大差异；

对于卷积，全连接等具有可学习参数的网络建议使用nn.Module；

激活函数（ReLU, sigmoid, tanh），池化等可以使用functional替代。对于不具有可学习参数的层，将他们用函数代替，这样可以不用放在构造函数__init__中。

300、介绍一下PyTorch中的DataLoader函数与参数的意义。（2星）

PyTorch中的DataLoader函数可以对数据集进行批量加载，数据的预处理，shuffling以及并行加速等操作。

DataLoader主要有以下几个参数：

1. `dataset`: 数据集。
2. `batch_size`: 每个batch的数据量。
3. `shuffle`: 是否对数据进行随机重排列。
4. `sampler`: 样本采样，主要用于数据集不平衡或者需要重载采样的情况。
5. `num_workers`: 用于数据导入的进程数量，0代表不使用多进程。
6. `drop_last`: 如果某个batch的数据量小于batch_size，是否舍弃该batch。
7. `collate_fn`: 可选，用来将一组样本拼接成一个batch，如果不指定则默认使用列表拼接成一个tensor。

其中，`sampler`可以使用的有随机采样(`RandomSampler`)、顺序采样(`SequentialSampler`)、权重采样(`WeightedRandomSampler`)、分布式采样等，根据不同的需求选择不同的采样策略。