bagging和boosting考点

降低偏差/方差

在一个模型的表现中，我们可以将其误差分解为三个部分：偏差、方差和随机误差。偏差是由于模型假设与真实模型不同造成的误差，这意味着即使我们用相同的算法和数据重复多次训练，我们仍然会得到同样的错误的预测结果。方差则是由于模型对训练数据的变化而导致的误差，如果我们用相同的算法在不同的训练集上训练，我们会得到不同的预测结果。最后，随机误差是由于我们在采集数据的时候会有误差，所以我们的训练集是不完美的。

偏差（bias）是指模型的预测值与真实值之间的偏差，即模型本身的准确度。偏差较高的模型往往欠拟合，无法捕捉到数据中的复杂关系，会导致高误差。

方差（variance）是指在不同的训练集上，同一个模型所预测的结果的差异，即模型的泛化能力。方差较高的模型往往过拟合，对于训练数据的拟合程度很高，但对于新的数据却预测能力较差，也会导致高误差。

Bagging可以降低模型的方差，而不是偏差。

Bagging通过对训练数据进行有放回的随机抽样（bootstrap），得到多个不同的训练集，然后在每个训练集上分别训练一个模型，再通过平均或投票的方式得到最终的预测结果。这种方法可以降低模型的方差，因为每个模型都是在不同的训练集上训练的，而不是在同一个训练集上，使得模型更具泛化能力。理解: 多棵树的结果通过max或者平均值计算出最终一个结果, 所以方差减小.

当我们使用Bagging算法的时候，我们构建了多个基础模型，每个基础模型都是基于随机抽样的方式构建的。这种随机抽样可以在训练数据的方差上起到平滑的作用，因为每个模型都是在不同的训练集上训练的，所以模型的预测值之间的差异会减少，从而减少模型的方差。但是，由于每个基础模型都是使用相同的算法进行训练的，所以每个模型之间的偏差是相似的，而这种偏差无法通过平均的方式消除，所以Bagging算法并不能降低模型的偏差。

总的来说，Bagging算法通过平均化多个基础模型的预测结果来减少模型的方差，但是它不能减少模型的偏差。如果我们希望降低模型的偏差，我们可以使用其他的算法，比如Boosting算法。

Boosting 减少偏差, 但是通常不会减少模型的方差，反而可能会增加模型的方差。

Boosting 是一种集成学习方法，通过加权组合多个弱分类器来构建一个强分类器。在训练过程中，Boosting 会重点关注训练集中容易被错误分类的样本，通过逐步迭代，使得模型能够逐渐将这些样本正确分类。这样，由于模型更加关注难以分类的样本，可以减少偏差，提高模型的准确率。

过拟合问题

boosting会过拟合(Adaboost和GBDT都会过拟合)

Boosting 对训练集的过拟合风险很高，因为每个弱分类器都是在先前弱分类器的基础上进行训练的。这样，如果某个弱分类器出现了过拟合，那么后续的弱分类器就会学习到先前的错误，从而导致整个模型的方差增加。因此，Boosting 往往需要采用一些措施来减少过拟合的风险，比如限制每个弱分类器的复杂度、采用随机化技术等。

1. 过于关注训练集中的噪声：Boosting的核心思想是在前一轮的基础上加强被错误分类的样本的权重，因此训练集中的噪声样本可能会被过度关注，导致模型过拟合。当数据集中存在噪声或异常值时，Boosting会将它们视为重要样本，这可能导致模型过拟合这些异常样本。
2. 子模型复杂度过高：Boosting在每一轮训练中使用的是一个简单模型，如决策树桩或线性模型，但随着迭代次数的增加，模型会逐渐变得复杂，容易过拟合。

为了解决Boosting过拟合的问题，可以采取以下方法：

1. 限制基础模型的复杂度：可以采用正则化方法，如限制树的最大深度、最小叶子节点数等。
2. 采用交叉验证选择超参数：通过交叉验证方法选择最优的超参数，避免模型在特定数据集上过拟合。
3. 前期停止：在模型迭代训练中适当停止，防止模型过拟合。
4. 数据清洗：清除训练集中的异常值和噪声数据，减少模型对这些数据的依赖性。

Adaboost会过拟合

Adaboost是一种常用的集成学习方法，它通过迭代加权样本和训练基分类器的方式，提高了模型的泛化性能。相对于单个分类器，Adaboost的泛化误差往往能够得到显著的改善，但是在某些情况下，Adaboost仍然有可能出现过拟合的情况。

Adaboost在训练过程中，对于前一轮分类器分类错误的样本，会增加其在下一轮训练中的权重，这样有可能会导致模型对于噪声和异常值过于敏感，从而出现过拟合的情况。此外，如果训练集中存在大量噪声，Adaboost也有可能过拟合。

为了避免过拟合的风险，可以考虑调整Adaboost的参数，如迭代次数、基分类器的复杂度等。此外，使用一些正则化技术，如对基分类器使用正则化项、剪枝等方法，也可以提高模型的泛化性能，降低过拟合的风险。

随机森林一般不会过拟合

随机森林一般不容易过拟合，这是由于其随机性和决策树的天生抗噪声能力所决定的。

在随机森林中，每个决策树都是通过随机选择特征和样本来构建的，这种随机性能够降低模型的方差。同时，随机森林中的每个决策树都是基于不同的样本集和特征集进行训练的，因此它们互相独立，可以减少模型的相关性和降低过拟合的风险。

此外，决策树具有天生的抗噪声能力，因为它们能够处理不完整和有噪声的数据，并且不容易受到局部噪声的影响。

虽然随机森林一般不容易过拟合，但如果训练数据中存在严重的噪声或异常值，或者特征数量过多，模型仍然有可能过拟合。因此，在实际应用中，我们仍然需要对模型进行评估和调参，以避免过拟合。

增加决策树数量的影响

Boosting:

增加决策树的数量会让 Boosting 的效果变得更加明显，但是同时也可能导致过拟合的风险增加。在实践中，增加决策树的数量需要权衡模型的效果和过拟合的风险，需要通过交叉验证等方法来确定最优的决策树数量，以得到最佳的模型性能。同时，可以采用一些正则化技巧来控制过拟合的风险，如限制决策树的最大深度、采用早期停止等。

在 Boosting 中，**增加决策树的数量通常会减少偏差。**随着决策树数量的增加，模型的复杂度也随之增加，可以更好地拟合训练数据，因此偏差会逐渐减小。

然而，**随着决策树数量的增加，模型的方差也会随之增加。**因为每个决策树都可能会对数据进行过度拟合，而且不同的决策树之间往往存在较高的相关性(训练数据集和构造树的特征是完全相同的)，这可能会导致模型的方差逐渐增大。

因此，在实际应用中，需要通过交叉验证等方法来确定最优的决策树数量，以达到偏差与方差的平衡，以获得最佳的模型性能。

Bagging:

对于bagging算法，增加决策树的数量会降低方差，但偏差通常不会显著减小，甚至有可能增加。这是因为bagging算法本身就是用多个具有相同偏差的模型进行集成，因此增加模型数量并不会显著降低整体的偏差。

随着决策树数量的增加，模型的方差会逐渐减小。这是因为随着模型数量的增加，模型的集成会更加平滑，预测结果的波动性会减小。但是，如果模型数量增加过多，就有可能导致过拟合问题，此时模型的方差可能会反弹上升。

因此，**对于bagging算法，增加决策树的数量通常会减少方差，但偏差可能不会显著减小**，同时需要注意过拟合问题。

Adaboost易错点

权重更新趋势: 不一定一直减少

在Adaboost中，每一轮的错误率不一定会一直减少，因为Adaboost的目标是通过加权的方式去调整样本的分布，使得每一轮分类器更关注于错误分类的样本。如果在某一轮中，分类器对错误分类的样本分类效果不如前一轮，那么错误率就可能会上升。不过，一般情况下，随着Adaboost的迭代次数的增加，训练误差率会逐渐下降，同时在测试数据上的性能也会得到提升。

基学习器:

1. 二叉树桩(水平或者竖直的二分线)
2. 选择依据: 选使得错误率最小的

正则化考点

①正则化线性回归代价函数:

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}[((h\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2)]$$

②梯度下降算法:

$$\theta_j := \theta_j(1 - a\frac{\lambda}{m}) - a\frac{1}{m}\sum_{i=1}^{m}(h\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

①正则化逻辑回归代价函数:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

② 梯度下降算法

$$\theta_j := \theta_j - a[\frac{1}{m}\sum_{i=1}^{m}(h\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j]$$

正则化参数α的大小对模型参数的影响可以分为以下几个方面：

1. 影响模型的复杂度：正则化项惩罚模型参数的大小，α的值越大，对模型的约束力越强，模型的复杂度越小。因此，α越大，模型的拟合能力会降低，可能会导致欠拟合问题。
2. 影响参数的收缩方向：正则化项惩罚模型参数的绝对值，因此当α的值很大时，正则化项的梯度会使大部分参数朝着0的方向收缩，从而降低模型的复杂度和过拟合的风险。这种情况下，α值越大，收缩的效果越明显，可能导致一些参数被归零，从而减少了模型中的自由参数。
    - 参数值: 随着正则参数的增大而减小
        - 解释: 对于多次项, 如果**高次项的系数减小**了, 则曲线会变得更加平滑, 效果更好
    - 参数数量: 减小
        - 解释: 权重参数减为0, 即**去除了一些维度**, 减小过拟合
3. 影响优化过程：正则化项对损失函数的梯度有影响，因此不同的α值可能会影响随机梯度下降（SGD）优化算法的收敛速度和稳定性。当α很大时，正则化项的梯度会比损失函数的梯度大很多，可能导致SGD算法更新参数时震荡和不稳定。此外，当α很小时，正则化项的影响几乎可以忽略不计，模型的参数更新过程类似于没有正则化的情况。

因此，在实际应用中，我们需要根据模型的复杂度、数据集的大小、训练算法的稳定性等因素来选择合适的α值。如果模型存在过拟合问题，可以逐步增大α值来减少模型的复杂度；如果模型存在欠拟合问题，可以逐步降低α值来增加模型的复杂度。

正则化项对损失函数的优化起到了约束作用，它可以惩罚参数的绝对值，促使参数的值逐渐向零收缩。当正则化参数α较大时，正则化项的权重会逐渐增加，对损失函数的影响逐渐加强，导致大部分参数都朝着0的方向收缩。这种情况下，越靠近0的参数所对应的特征在模型中的贡献越小，因此某些参数可能会被压缩到接近0的程度，这些参数所对应的特征在模型中的重要性就会降低，甚至可以直接被忽略掉。

可以通过一个简单的例子来说明这一点。假设我们使用L2正则化来训练一个线性回归模型：

$\min_\theta \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\alpha}{2}\sum_{j=1}^{n}\theta_j^2$

其中，m是样本数量，n是特征数量，α是正则化参数。当α的值比较大时，正则化项的影响就比较明显，参数更新时就会被惩罚，从而被迫朝着0的方向收缩。如果某些参数在更新过程中一直保持很小的值，它们对应的特征在模型中的重要性就比较低，这些参数对应的特征可以被归零，从而减少模型中的自由参数，使得模型更加简洁和稳定。

因此，正则项导致收缩的效果越明显，可能导致一些参数被归零，从而降低模型的复杂度和过拟合的风险。这也是L1正则化在特征选择中比L2正则化更有效的原因之一。

过拟合和欠拟合考点

SVM

c=无穷大时, 变为线性可分

C增大, 更关注错误程度而不是最大间隔, 此时增加过拟合(训练错误率为0, 实际错误率很大)

当C取无穷大时，软间隔SVM的目标函数中的正则化项的影响变得非常重要，相当于对模型参数做了极强的约束，使得模型在训练集上的误差趋近于0。这种情况下，模型的目标是尽可能减小误差，而不是使得间隔最大化，即模型更加关注训练集上的细节而不是泛化性能。

具体来说，当C取无穷大时，模型的最优解是把所有的样本都分类正确，即每个样本的松弛变量都等于0。此时，模型的决策边界完全依赖于数据的分布和正则化项，而不是间隔。因此，当C取无穷大时，模型会变得非常严格，对于任何噪声或者异常点都会非常敏感，容易过拟合。