从梯度提升树到XGBoost

机器学习算法已经成为研究和应用的热点。目前，最流行的两类算法莫过于**神经网络算法（卷积神经网络、循环神经网络、生成式对抗网络和图神经网络）**与**树形算法（随机森林、GBDT、XGBoost和LightGBM）**。树形算法的基础就是**决策树**，由于其易理解、易构建、速度快等特点，被广泛的应用在数据挖掘、机器学习等领域。因此，决策树是经典的机器学习算法，很多复杂的机器学习算法都是由决策树演变而来。对于决策树的学习，是我们机器学习课程中非常重要的一个环节。

根据处理数据类型的不同，决策树又分为两类：**分类决策树与回归决策树**(我们之前的)。分类决策树可用于处理离散型数据，回归决策树可用于处理连续型数据。

之后我们又讨论决策树的局限性，尤其是它们容易过拟合的问题之后，我们引入了随机森林算法。随机森林的核心思想是基于集成学习中的Bagging（Bootstrap Aggregating）方法。这种方法通过构建多棵决策树，每棵树都在数据的一个随机子集上训练，从而减少模型的方差，提高泛化能力。在随机森林中，每棵树的构建是相互独立的，这意味着它们不会相互影响，这样的设计有助于捕捉数据中的不同特征和模式，从而降低整体模型过拟合的风险。通过这种方式，随机森林能够综合多棵树的预测结果，以提高整体的预测准确性和鲁棒性。

然而，尽管随机森林通过集成多棵独立的决策树来提高模型的稳定性和准确性，但这种方法并没有充分利用树与树之间的信息。在随机森林中，每棵树的构建和预测都是相互独立的，它们不会考虑其他树的预测结果，这在一定程度上限制了模型性能的进一步提升。这就好比在现实社会中，尽管我们每个人在某种程度上是独立的个体，但我们的学习、工作和生活往往不是完全孤立的。我们会借鉴前人的经验，从他人的成败中吸取教训，以此来优化自己的行为和决策。

正是基于这样的观察，我们引入了另一种集成学习思想——Boosting。与Bagging不同，Boosting方法中的每棵树都会考虑前序树的预测结果，通过逐步修正前序树的错误来提升模型性能。这种方法强调了树与树之间的依赖关系，每棵树都在尝试减少前序树的错误，从而实现模型性能的逐步提升.

而后XGBoost（Extreme Gradient Boosting）在梯度提升决策树（GBDT）的基础上引入了正则化,并且支持了并行处理使得模型可以更加快读的训练,以及防止模型过拟合.

LightGBM在XGBoost的基础上进一步做了以下改进：

1. 基于直方图的算法：LightGBM采用了直方图算法将特征值离散化为有限的直方图桶，加快了计算速度并降低内存占用 。
2. 叶子节点增长策略：LightGBM采用基于叶子节点的增长策略（Leaf-wise Growth），与XGBoost的按层生长（Level-wise）策略相比，可以更好地利用数据特性，构造出更加精准的决策树 。

3. 特征单边梯度采样：LightGBM基于梯度采样，保留较大梯度的样本，同时随机采样较小梯度的样本，以达到加速模型训练的目的 。
4. 特征互斥机制：LightGBM将稀疏特征进行捆绑处理，进一步减少特征维度 。类
5. 别特征支持：LightGBM支持类别特征，不需要进行独热编码处理，这使得LightGBM在处理具有类别特征的数据集时更加高效和方便 。
6. 缓存优化：LightGBM的直方图算法对Cache友好，提高了缓存命中率，同时降低了存储消耗 。

这些改进使得LightGBM在处理大规模数据集时具有更快的训练速度和更低的内存消耗，特别适合于大数据场景 .

在探讨了XGBoost和LightGBM这两种基于梯度提升的集成学习方法之后，我们可以看到它们在处理大规模数据集和提高模型性能方面的优势。然而，尽管这些算法在许多场景下表现出色，但在处理类别型特征时，它们通常需要额外的预处理步骤，如独热编码，这不仅增加了计算成本，还可能导致维度灾难。

CatBoost的引入：

在这样的背景下，CatBoost应运而生，它特别针对类别型特征的处理进行了优化。CatBoost不仅能够直接处理类别型特征，无需任何预处理，而且还在模型的准确性和训练速度上进行了显著的改进。

CatBoost的关键特性：

1. **类别型特征的原生支持**：CatBoost不需要对类别型特征进行独热编码或其他转换，它可以直接从这些特征中学习，这大大简化了数据预处理的步骤。
2. **对称决策树**：CatBoost使用对称决策树，这意味着在树的构建过程中，所有树都使用相同的分裂条件，这有助于减少模型的复杂性，并提高预测的速度。
3. **有序提升**：CatBoost采用了一种称为有序提升的方法，这种方法通过在不同的数据子集上交替训练模型和计算残差，有效地减少了过拟合的风险。
4. **鲁棒性和泛化能力**：CatBoost的设计使其对异常值和噪声具有很好的鲁棒性，同时在未见数据上的泛化能力也得到了增强。
5. **GPU加速**：CatBoost支持GPU加速，这使得在处理大规模数据集时，训练速度得到了显著提升。
6. **自动特征选择**：CatBoost还能够自动选择最重要的特征，减少了特征工程的工作量。

通过这些特性，CatBoost在许多机器学习任务中，尤其是在那些包含大量类别型特征的数据集上，展现出了卓越的性能。它与XGBoost和LightGBM一起，构成了梯度提升决策树算法的三大支柱，各自在不同的应用场景中发挥着重要作用。

> **NOTE**

CatBoost是俄罗斯的搜索巨头Yandex在2017年开源的机器学习库，是Boosting族算法的一种。CatBoost和XGBoost、LightGBM并称为GBDT的三大主流神器，都是在GBDT算法框架下的一种改进实现。XGBoost被广泛的应用于工业界，LightGBM有效的提升了GBDT的计算效率，而Yandex的CatBoost号称是比XGBoost和LightGBM在算法准确率等方面表现更为优秀的算法。

CatBoost是一种基于对称决策树（oblivious trees）为基学习器实现的参数较少、支持类别型变量和高准确性的GBDT框架，主要解决的痛点是高效合理地处理类别型特征，这一点从它的名字中可以看出来，CatBoost是由Categorical和Boosting组成。此外，CatBoost还解决了梯度偏差（Gradient Bias）以及预测偏移（Prediction shift）的问题，从而减少过拟合的发生，进而提高算法的准确性和泛化能力

CatBoost和XGBoost、LightGBM并称为GBDT的三大主流神器，所以他们都是基于GBDT改进而来，而说到，梯度提升决策树（Gradient Boosting Decision Tree，GBDT）它的主要思想又是来自于Boosting思想.因此在开始之前，我们非常有必要了解一下Boosting思想

Boosting基本概念

提升（Boosting）方法是一种常用的统计学习方法，应用广泛且有效。在分类问题中，它通过改变训练样本的权重，学习多个分类器，并将这些分类器进行线性组合，提高分类的性能。

提升方法基于这样一种思想：对于一个复杂任务来说，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断好。实际上，就是“三个臭皮匠顶个诸葛亮”的道理。

历史上，Kearns和Valiant首先提出了“**强可学习**（Strongly Learnable）”和“**弱可学习**（Weakly Learnable）”的概念。指出：在概率近似正确（Probably Approximately Correct，PAC）学习的框架中，一个概念（一个类），如果存在一个多项式的学习算法能够学习它，并且正确率很高，那么就称这个概念是强可学习的；一个概念，如果存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜测略好，那么就称这个概念是弱可学习的。非常有趣的是Schapire后来证明强可学习与弱可学习是等价的，也就是说，在PAC学习的框架下，一个概念是强可学习的充分必要条件是这个概念是弱可学习的。

这样一来，问题便成为，在学习中，如果已经发现了“弱学习算法”，那么能否将它提升（Boost）为“强学习算法”。大家知道，发现弱学习算法通常要比发现强学习算法容易得多。那么如何具体实施提升，便成为开发提升方法时所要解决的问题。关于提升方法的研究很多，有很多算法被提出。最具代表性的**是AdaBoost算法（AdaBoost algorithm）**以及 **梯度提升树（Gradient Boosting Trees，GBDT）**。

Boosting算法的两个核心问题：

（1）在每一轮如何改变训练数据的权值或概率分布？

AdaBoost的做法是，提高那些被前一轮弱分类器错误分类样本的权值，而降低那些被正确分类样本的权值。这样一来，那些没有得到正确分类的数据，由于其权值的加大而受到后一轮的弱分类器的更大关注。于是，分类问题被一系列的弱分类器“分而治之”。

问题1：如何调整训练数据的权重？

想象一下，你是一位老师，正在教一群学生。在一次数学测试后，你发现有些学生做得很好，而有些学生则犯了错误。作为老师，你可能会更关注那些犯了错误的学生，希望他们能在下次测试中做得更好。在Boosting算法中，我们也是这么做的：

- **提高错误样本的权重**：在第一轮学习中，如果一个弱分类器（可以想象成一个不太聪明的学生）在某些样本上犯了错误（比如，没能正确分类），那么这些样本在下一轮学习中的权重就会增加。这意味着，这些“困难”的样本会在后续的学习中被更加重视。
- **降低正确样本的权重**：相反，那些被正确分类的样本的权重会降低。这样，算法就会更加关注那些难以分类的样本，而不是那些已经能轻松分类的样本。

通过这种方式，Boosting算法能够逐步改进，重点关注那些难以分类的数据点。

> **NOTE**
>
> 在每一轮如何改变训练数据的权值或概率分布？提高上一个弱分类器分错的样本的权值，减小前一轮中分类正确的样本的权值，让误分类的样本在后续受到更多的关注。

（2）如何将弱分类器组合成一个强分类器？ 弱分类器的组合，AdaBoost采取加权多数表决的方法。具体地，加大分类误差率小的弱分类器的权值，使其在表决中起较大的作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。

问题2：如何将多个弱分类器组合成一个强分类器？

现在，想象一下你有一个团队，团队中的每个成员都有自己的专长和弱点。你的目标是让整个团队的表现尽可能好。在Boosting算法中，我们也是这样做的：

- **加权多数表决**：每个弱分类器（团队成员）都会对最终结果进行投票。但是，那些表现好的分类器（即分类误差率小的成员）的投票权重会更大，这意味着他们在最终决策中的影响力更大。
- **调整权重**：相反，那些表现不佳的分类器（即分类误差率大的成员）的投票权重会较小，他们在最终决策中的影响力就会降低。

通过这种方式，我们可以将多个弱分类器的预测结果结合起来，形成一个更强大的分类器。这个强分类器能够综合各个弱分类器的优点，提高整体的预测准确性。

> **NOTE**

如何将弱分类器组合成为一个强分类器？通过加法模型将弱分类器进行线性组合。比如AdaBoost通过加权多数表决的方式，即增大错误率较小的分类器的权值，同时减小错误率较大的分类器的权值；提升树通过拟合残差的方式逐步减小残差，将每一步生成的模型叠加得到最终模型。

提升树是以分类树或回归树为基本分类器的提升方法。提升树被认为是统计学习中性能最好的方法之一。提升方法实际采用加法模型（即基函数的线性组合）与前向分步算法。以决策树为基函数的提升方法称为提升树（Boosting Tree）。对分类问题决策树是二叉分类树，对回归问题决策树是二叉回归树。下面让我们深入理解提升树的具体算法吧！这里我们先来看最第一个成员Adaboost

Boosting基本成员Adaboost-提升法代表

Boosting家族中另一个非常重要的成员是AdaBoost（Adaptive Boosting）。AdaBoost算法的核心在于通过迭代地训练一系列弱分类器，并根据每个弱分类器的表现调整样本权重，最终将这些弱分类器组合成一个强分类器。AdaBoost的核心思想在于，每个新的弱分类器都会更加关注前一个分类器错误分类的样本，从而逐步提高模型的整体性能。

AdaBoost是Adaptive Boosting的简称，是由Yoav Freund和Robert Schapire提出的一种机器学习元算法，他们的工作获得了2003年哥德尔奖。AdaBoost算法可以和许多其他类型的学习算法一起使用，以提高性能。其他学习算法（“弱学习器”）的输出被组合成一个加权和，代表提升分类器的最终输出。AdaBoost是自适应的，即后续的弱学习器会被调整，以支持那些被之前的分类器错误分类的实例。各个学习器可以很弱，但只要每个学习器的性能比随机猜测稍好，就可以证明最终的模型收敛到一个强学习器。

以决策树为弱学习器的AdaBoost通常被称为最好的开箱即用的分类器。AdaBoost算法的每个阶段收集的关于每个训练样本分类的相对“难度”的信息被反馈给树生长算法，这样以后的树就会倾向于关注更难分类的样本。

机器学习中的问题常常受到维度灾难问题的影响，评估每个特征不仅会降低分类器的训练和执行速度，事实上还会降低预测能力。与神经网络和SVM不同，AdaBoost训练过程只选择那些已知能提高模型预测能力的特征，减少了维度，并可能提高执行速度，因为不相关的特征不需要计算.

AdaBoost算法的工作流程大致如下：

1. **初始化样本权重**：给定训练数据集，初始化每个样本的权重，通常每个样本的初始权重相等。
2. **迭代训练弱分类器**：对于每一轮迭代，使用当前的样本权重分布来训练一个弱分类器，并计算该分类器的分类误差率。

3. **更新样本权重**：根据当前弱分类器的表现更新样本权重，错误分类的样本权重会增加，而被正确分类的样本权重会减少。
4. **组合弱分类器**：将所有训练得到的弱分类器按照它们的权重组合起来，形成一个强分类器。
5. **终止条件**：达到预定的迭代次数或者性能不再提升时停止迭代。

AdaBoost算法的特点在于其自适应性，它能够根据样本被错误分类的难易程度动态调整样本权重，使得模型更加关注那些难以分类的样本。此外，AdaBoost算法在理论上和实践中都已被证明是一种有效降低模型偏差的方法。

AdaBoost中的前向分步加法模型

考虑加法模型（Additive Model）如下：

$$f(x) = \sum_{m=1}^{M} \beta_m b\left(x; \gamma_m\right)$$

其中，$b(x;\gamma_m)_{\text{为基函数}}$，$\gamma_m$ 为基函数的参数，β_m 为基函数的系数。显然上式是一个加法模型。

在给定训练数据及损失函数 $L(Y, f(x))_{\text{的条件下，学习加法模型}}$ $f(x)$ 成为经验风险极小化，即损失函数极小化的问题：

$$\min_{(\beta_m, \gamma_m)} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b\left(x_i; \gamma_m\right)\right)$$

通常这是一个复杂的优化问题。前向分布算法（Forward Stagewise Algorithm）求解这一优化问题的想法是：因为学习的是加法模型，如果能够从前向后，每一步只学习一个基函数及其系数，逐步逼近上面要优化的目标函数，那么就可以简化优化的复杂度。

当然我们这里其实并不用太过关心具体的数学计算.我们直接代码看看他是怎么玩的.