**XGBoost相比传统GBDT有何不同？XGBoost为什么快？XGBoost如何支持并行？**

* **基分类器的选择：**传统GBDT以CART作为基分类器，XGBoost还支持线性分类器，这个时候XGBoost相当于带L1和L2正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）。
* **二阶泰勒展开：**传统GBDT在优化时只用到一阶导数信息，XGBoost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。顺便提一下，XGBoost工具支持自定义损失函数，只要函数可一阶和二阶求导。**（类似于梯度下降法加上了牛顿法）**
* **XGBoost工具支持并行：**Boosting不是一种串行结构吗?怎么并行的？注意XGBoost的并行不是tree粒度的并行，XGBoost也是一次迭代完才能进行下一次迭代的（第次迭代的损失函数里包含了前面次迭代的预测值）。XGBoost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），XGBoost在训练之前，预先对数据进行了排序，然后保存为block(块)结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。
* **方差-方差权衡：**XGBoost在目标函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出分数的L2模的平方和。从Bias-variance tradeoff角度来讲，正则项降低了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是XGBoost优于传统GBDT的一个特性。
* **Shrinkage（缩减）：**相当于学习速率（xgboost中的）。XGBoost在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。实际应用中，一般把eta设置得小一点，然后迭代次数设置得大一点。（补充：传统GBDT的实现也有学习速率）
* **列抽样（column subsampling）：**XGBoost借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是XGBoost异于传统GBDT的一个特性。
* **缺失值处理：**XGBoost考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率，paper提到50倍。即对于特征的值有缺失的样本，XGBoost可以自动学习出它的分裂方向。
* **线程缓冲区存储：**按照特征列方式存储能优化寻找最佳的分割点，但是当以行计算梯度数据时会导致内存的不连续访问，严重时会导致cache miss，降低算法效率。paper中提到，可先将数据收集到线程内部的buffer（缓冲区），主要是结合多线程、数据压缩、分片的方法，然后再计算，提高算法的效率。
* **可并行的近似直方图算法：**树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以xgboost还提出了一种可并行的近似直方图算法，用于高效地生成候选的分割点。大致的思想是根据百分位法列举几个可能成为分割点的候选者，然后从候选者中根据上面求分割点的公式计算找出最佳的分割点。

XGBoost最初源于CART回归树集成。回归树的结点不再是类别，是数值（预测值）。

**AdaBoost 和 GBDT**

和AdaBoost一样，Gradient Boosting也是重复学习弱学习器，且每次基于先前模型的表现进行调整。不同的是，AdaBoost是通过提升错分数据点的权重来定位模型的不足，而Gradient Boosting是通过算梯度（gradient）来定位模型的不足，并使用梯度下降法逐步调整。

相比AdaBoost，Gradient Boosting可以使用更多种类的目标函数。二分类问题时，直接把AdaBoost基学习器换成CART分类树就是提升树。回归问题时，GBDT先初始学习一棵决策树，叶子处得到预测的值以及预测之后的残差。之后的决策树基于前面决策树的残差来学习，直到预测值和真实值的残差为零。最后对测试样本的预测值，就是前面多棵决策树预测值的累加。

GBDT两种解释：

（1）梯度下降法

（2）残差最小化

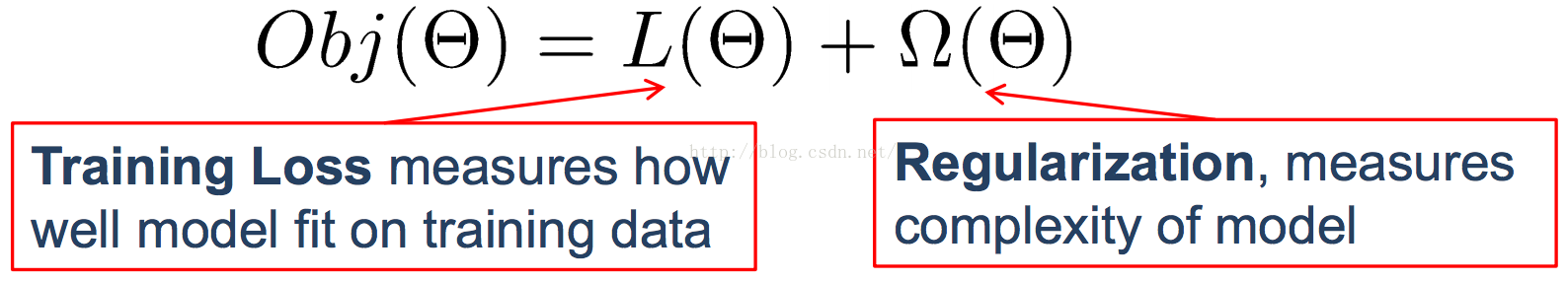
两者原理一样。

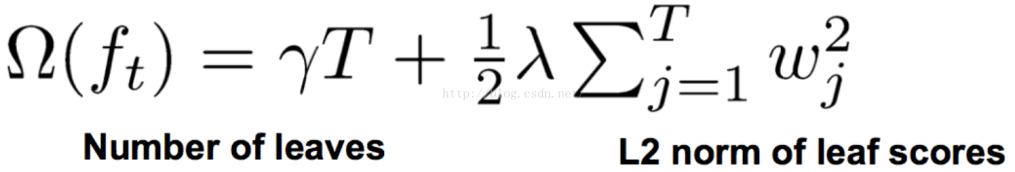
**残差：***r*mi = yi － fm-1(xi) 实际训练数据y减去上一轮拟合结果

**GBDT 和LR**

从决策边界来说，线性回归的决策边界是一条直线，逻辑回归的决策边界是一条曲线，而GBDT的决策边界可能是很多条线。

GBDT并不一定总是好于线性回归或逻辑回归。根据没有免费的午餐原则，没有一个算法是在所有问题上都能好于另一个算法的。根据奥卡姆剃刀原则，如果GBDT和线性回归或逻辑回归在某个问题上表现接近，那么我们应该选择相对比较简单的线性回归或逻辑回归。具体选择哪一个算法还是要根据实际问题来决定。

**XGBoost最小化的目标函数（经验风险 + 结构风险 => 原损失函数 + 正则化项）**

**XGBoost正则化项（L1项 + L2项）**

T：叶结点个数 ω：叶子权重

**XGBoost损失函数有二阶泰勒展开，即在一阶导的同时使用二阶导去优化损失函数。**