

硕 士 研 究 生 读 书 报 告



题目  异常检测算法之孤立森林

作者姓名 黄 闯

作者学号 21860451

指导教师  才振功

学科专业 计算机技术

所在学院 工程师学院

提交日期 2019年4月29日

Anomaly detection algorithm for isolated forest

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Cai zhen gong

By

Huang Chuang

Zhejiang University, P.R. China

2019

# xgboost之底层原理解析

## 摘要

xgboost作为一种提升算法，在机器学习领域有着举足轻重的地位，由于其相对较好的性能以及非常友好的数据处理输入与输出，受到众多机器学习工程师的青睐；但是如果想要深入的应用xgbost以及能够在大规模数据上较好的应用xboost，就需要做到对xgboost的熟烂于心，心中有数，这样在学习、工作和应用中才能够发挥xgboost的最大效用。本部分内容则是针对xboost的结构以及数学原理推导进行阐述，旨在对其原理作以解析。

关键词：集成学习，决策树，泰勒展开，机器学习。

**Abstract**

As a lifting algorithm, xgboost plays an important role in the field of machine learning. Due to its relatively good performance and very friendly data processing input and output, xgboost is favored by many machine learning engineers. However, if you want to apply xgbost in depth, The ability to use xboost on large-scale data requires a lot of familiarity with xgboost, so that you can use xgboost in learning, work, and applications. This part of the content is to explain the structure of xboost and the derivation of mathematical principles, aiming at the analysis of its principle.

**Keywords**: integrated learning, decision tree, Taylor expansion, machine learning.

## 前言

xgboost一直在竞赛江湖里被传为神器，比如时不时某个kaggle或者天池比赛中，某人用xgboost于千军万马中斩获冠军。

机器学习课程里也必讲xgboost，如寒所说：“RF和GBDT是工业界大爱的模型，Xgboost 是大杀器，Kaggle各种Top排行榜曾一度呈现Xgboost一统江湖的局面，另外某次滴滴比赛第一名的改进也少不了Xgboost的功劳”。

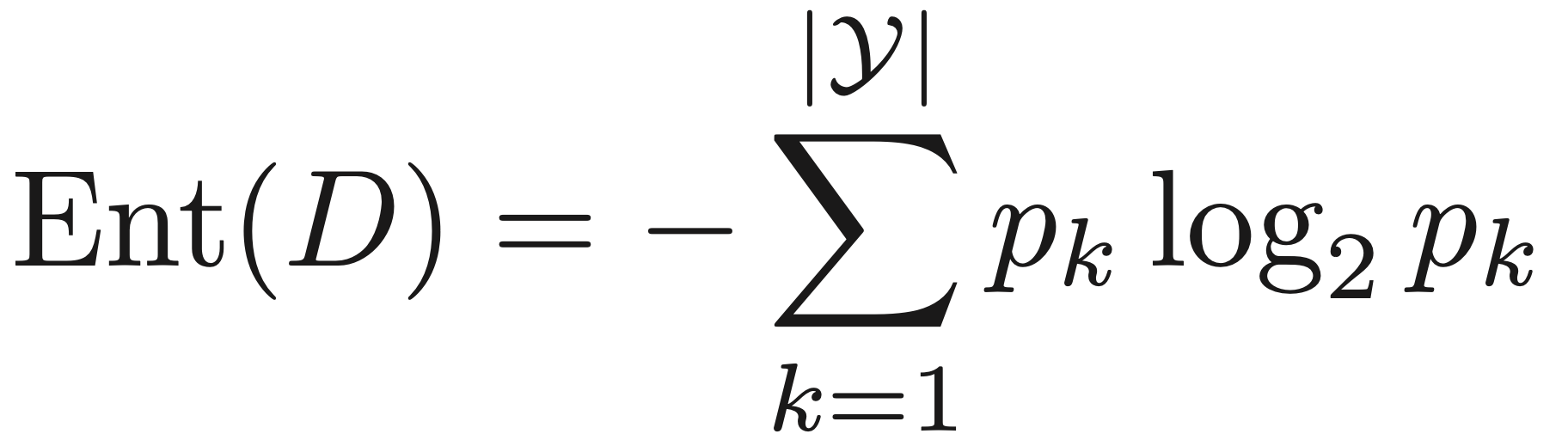
## 1 决策树

所谓决策树，就是当我们要针对某个问题需要得到结果的时候，我们应该根据那个特征以及什么标准进行分割数据集，

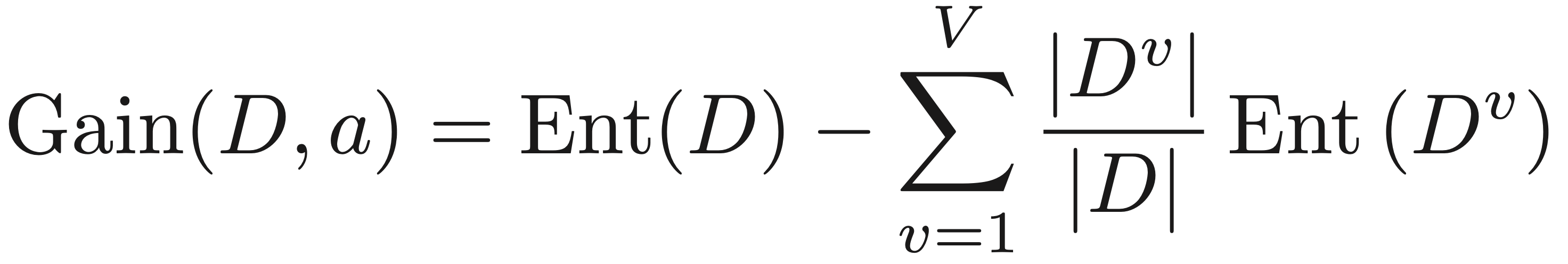
但究竟根据哪个特征划分更好呢？很直接的判断是哪个分类效果更好则优先用哪个。所以，这时就需要一个评价标准来量化分类效果了。

量化分类效果的方式有很多，比如信息增益（ID3）、信息增益率（C4.5）、基尼系数（CART）。

以下就是信息熵的计算公式



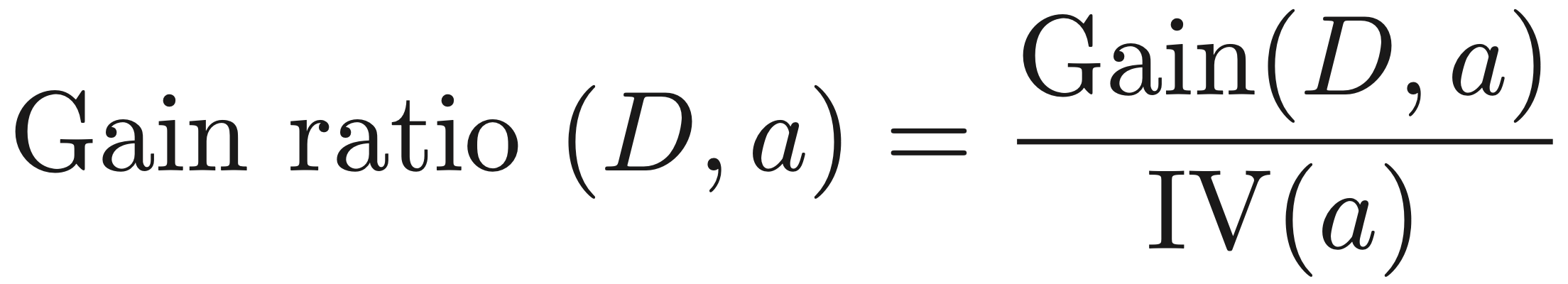
在信息熵的基础之上，如果利用某个特征来对样本进行划分，那么是势必会把样本本身进一步的细化，这样就会造成混乱程度增加，所以信息增益的计算公式如下：



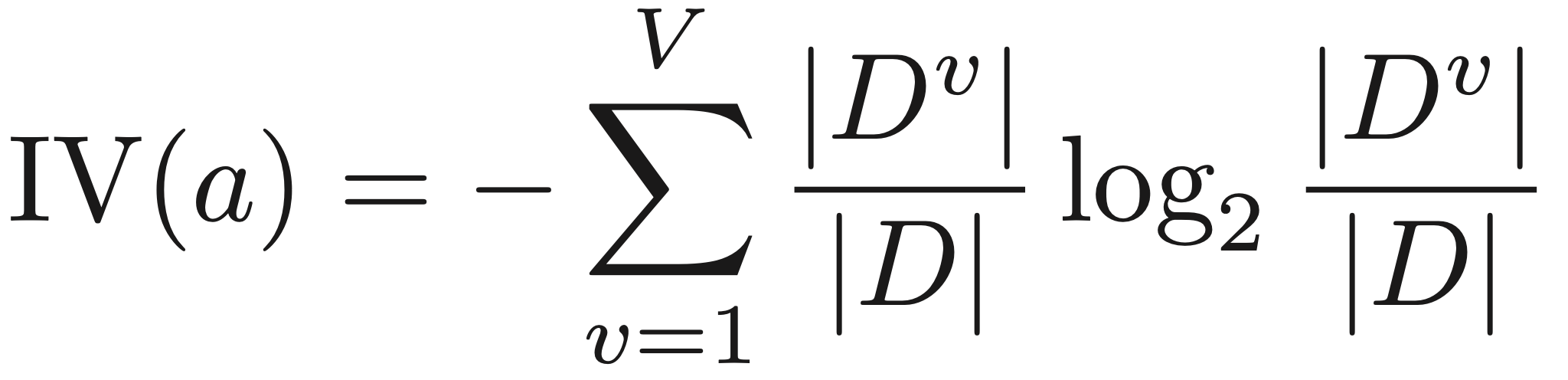
ID3算法的核心思想就是以信息增益度量属性选择，选择分裂后信息增益最大的属性进行分裂。

什么是信息增益呢？为了精确地定义信息增益，我们先定义信息论中广泛使用的一个度量标准，称为熵（entropy），它刻画了任意样例集的纯度（purity）。

信息增益率的公式如下：

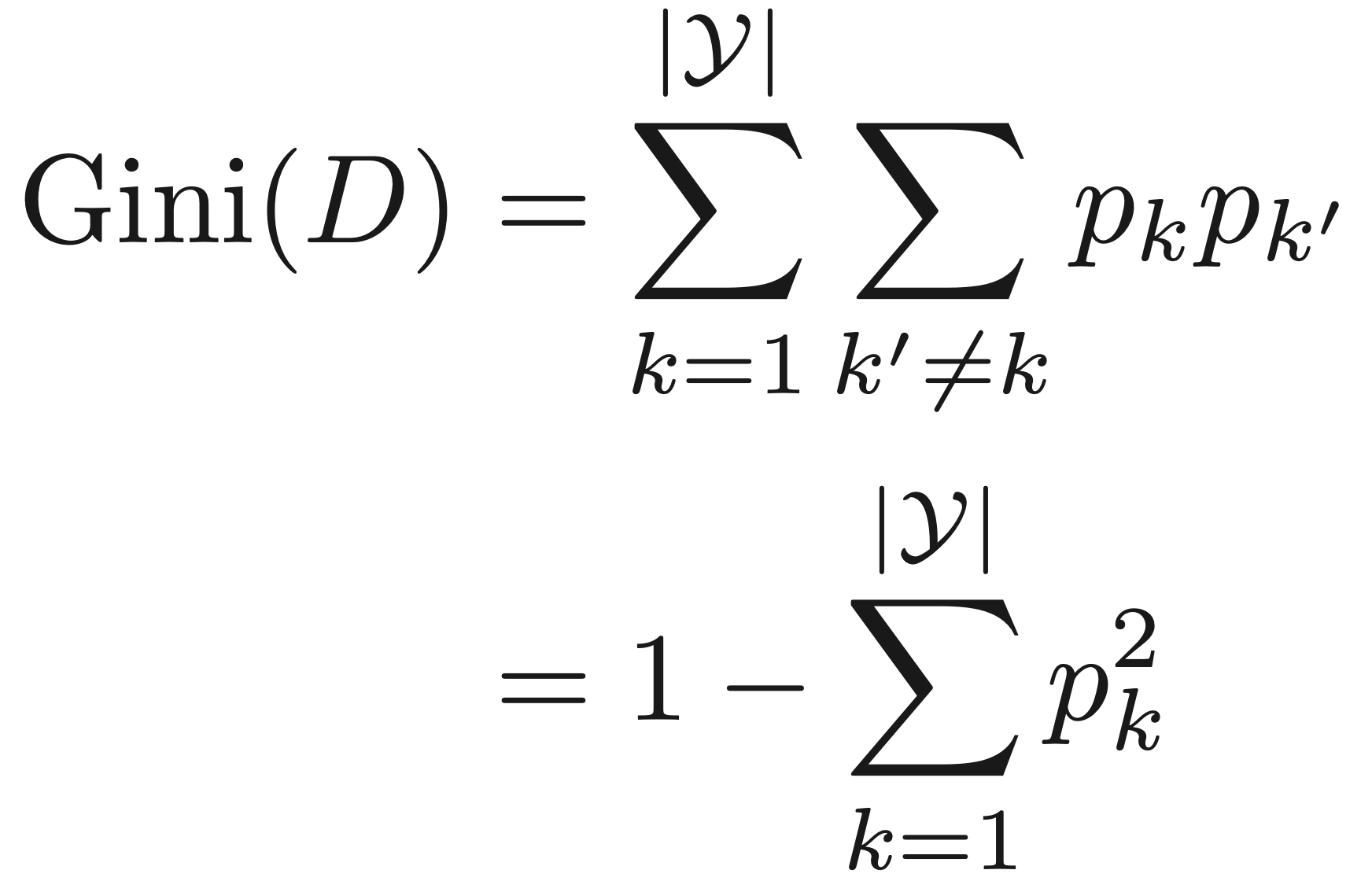


IV数值计算：



### C4.5就是用信息增益率来选取特征，来划分数据集的；

基尼系数公式：



### CART用的就是基尼系数这个标准，而GDBT的基础就是CART，这些都是一层一层搭建起来的机器学习方法体系。

## 2.回归树与集成学习

用一句话定义xgboost，很简单：Xgboost就是由很多分类和回归树集成。

数据挖掘或机器学习中使用的决策树有两种主要类型：

分类树分析是指预测结果是数据所属的类（比如某个电影去看还是不看）

回归树分析是指预测结果可以被认为是实数（例如房屋的价格，或患者在医院中的逗留时间）

而术语分类和回归树（CART，Classification And Regression Tree）分析是用于指代上述两种树的总称，由Breiman等人首先提出。

### 2.1 回归树

事实上，分类与回归是两个很接近的问题，分类的目标是根据已知样本的某些特征，判断一个新的样本属于哪种已知的样本类，它的结果是离散值。而回归的结果是连续的值。当然，本质是一样的，都是特征（feature）到结果/标签（label）之间的映射。

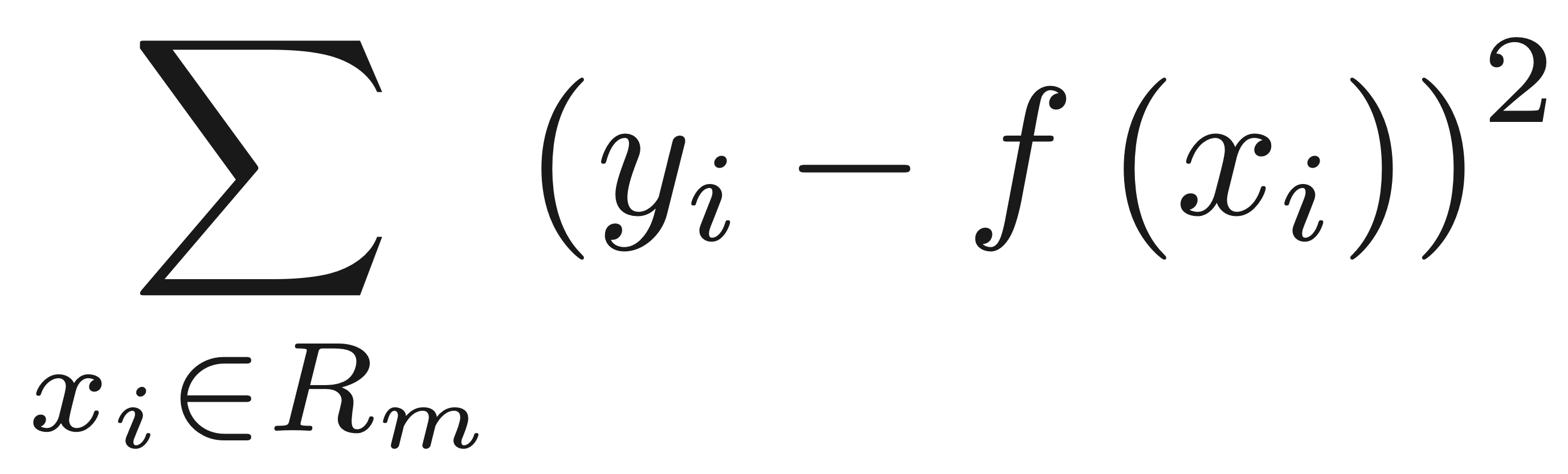
理清了什么是分类和回归之后，理解分类树和回归树就不难了。

所以，对于回归树，你没法再用分类树那套信息增益、信息增益率、基尼系数来判定树的节点分裂了，你需要采取新的方式评估效果，包括预测误差（常用的有均方误差、对数误差等）。而且节点不再是类别，是数值（预测值），那么怎么确定呢？有的是节点内样本均值，有的是最优化算出来的比如Xgboost。

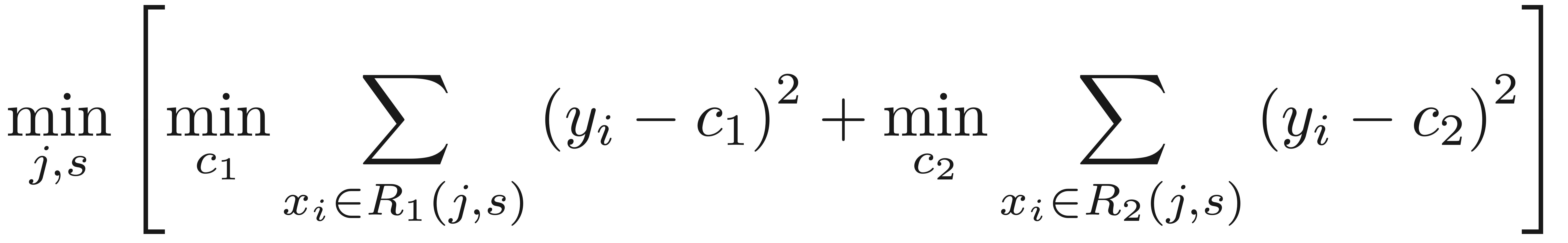
CART回归树是假设树为二叉树，通过不断将特征进行分裂。比如当前树结点是基于第j个特征值进行分裂的，设该特征值小于s的样本划分为左子树，大于s的样本划分为右子树。



而CART回归树实质上就是在该特征维度对样本空间进行划分，而这种空间划分的优化是一种NP难问题，因此，在决策树模型中是使用启发式方法解决。典型CART回归树产生的目标函数为：



因此，当我们为了求解最优的切分特征j和最优的切分点s，就转化为求解这么一个目标函数：



所以我们只要遍历所有特征的的所有切分点，就能找到最优的切分特征和切分点。最终得到一棵回归树。

### 2.2 boosting集成学习

所谓集成学习，是指构建多个分类器（弱分类器）对数据集进行预测，然后用某种策略将多个分类器预测的结果集成起来，作为最终预测结果。

集成学习根据各个弱分类器之间有无依赖关系，分为Boosting和Bagging两大流派：

Boosting流派，各分类器之间有依赖关系，必须串行，比如Adaboost、GBDT(Gradient Boosting Decision Tree)、Xgboost

Bagging流派，各分类器之间没有依赖关系，可各自并行，比如随机森林（Random Forest）

而著名的Adaboost作为boosting流派中最具代表性的一种方法，本博客曾详细介绍它。

AdaBoost，是英文"Adaptive Boosting"（自适应增强）的缩写，由Yoav Freund和Robert Schapire在1995年提出。它的自适应在于：前一个基本分类器分错的样本会得到加强，加权后的全体样本再次被用来训练下一个基本分类器。同时，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数。

具体说来，整个Adaboost 迭代算法就3步：

①初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予相同的权值：1/N。

②训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

③将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

而另一种boosting方法GBDT（Gradient Boost Decision Tree)，则与AdaBoost不同，GBDT每一次的计算是都为了减少上一次的残差，进而在残差减少（负梯度）的方向上建立一个新的模型。

很快你会意识到，Xgboost为何也是一个boosting的集成学习了。

而一个回归树形成的关键点在于：

• 分裂点依据什么来划分（如前面说的均方误差最小，loss）；

• 分类后的节点预测值是多少（如前面说，有一种是将叶子节点下各样本实际值得均值作为叶子节点预测误差，或者计算所得）

至于另一类集成学习方法，比如Random Forest（随机森林）算法，各个决策树是独立的、每个决策树在样本堆里随机选一批样本，随机选一批特征进行独立训练，各个决策树之间没有啥关系。本文暂不展开介绍。

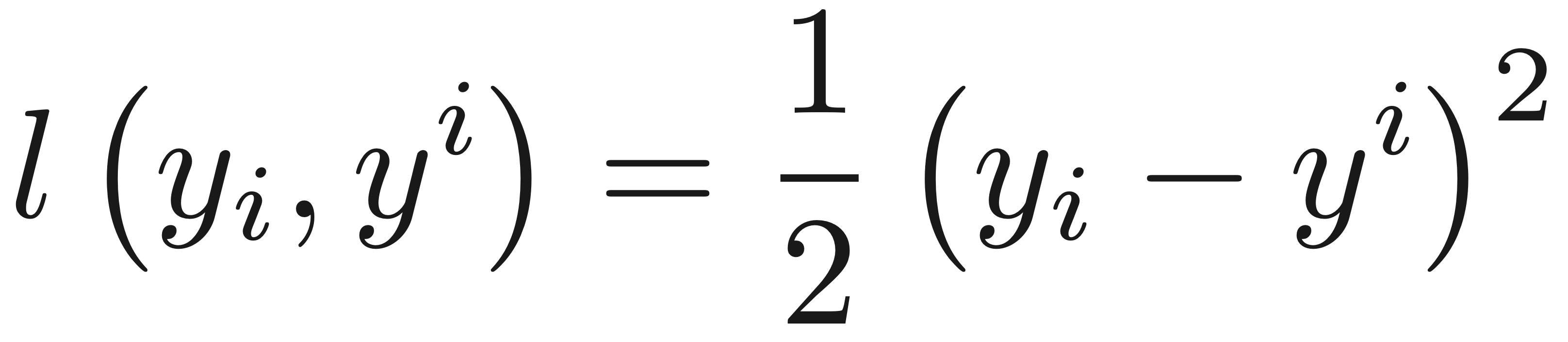
## 3.GBDT

Xgboost从GBDT(Gradient Boosting Decision Tree)演变而来。因为xgboost本质上还是一个GBDT，但是力争把速度和效率发挥到极致，所以叫X (Extreme) GBoosted。包括前面说过，两者都是boosting方法。

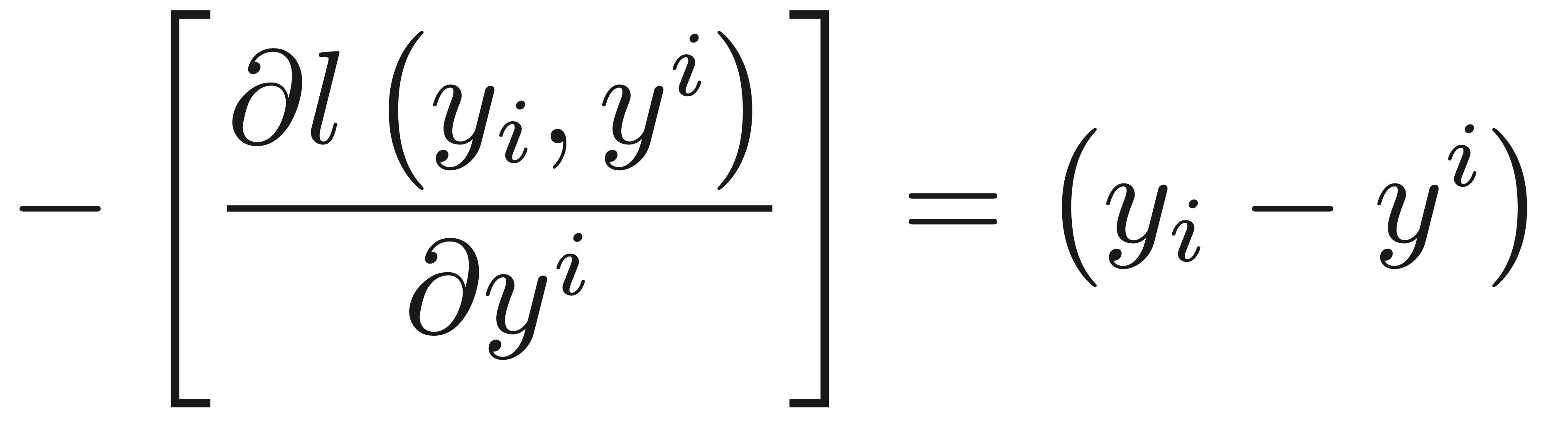
GBDT的原理很简单，就是所有弱分类器的结果相加等于预测值，然后下一个弱分类器去拟合误差函数对预测值的残差(这个残差就是预测值与真实值之间的误差)。当然了，它里面的弱分类器的表现形式就是各棵树。

注意，为何gbdt可以用用负梯度近似残差呢？

回归任务下，GBDT 在每一轮的迭代时对每个样本都会有一个预测值，此时的损失函数为均方差损失函数，



那此时的负梯度是这样计算的



所以，当损失函数选用均方损失函数是时，每一次拟合的值就是（真实值 - 当前模型预测的值），即残差。此时的变量是IMG_264，即“当前预测模型的值”，也就是对它求负梯度。

残差在数理统计中是指实际观察值与估计值（拟合值）之间的差。“残差”蕴含了有关模型基本假设的重要信息。如果回归模型正确的话， 我们可以将残差看作误差的观测值。

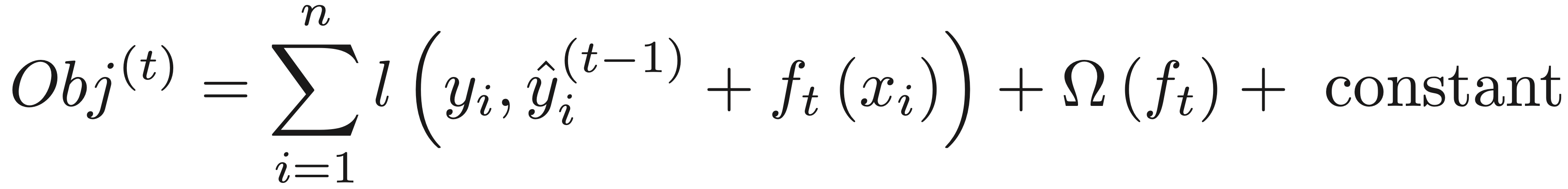
所以，GBDT需要将多棵树的得分累加得到最终的预测得分，且每一次迭代，都在现有树的基础上，增加一棵树去拟合前面树的预测结果与真实值之间的残差。

## 4.Xgboost

### 4.1 xgboost树的定义

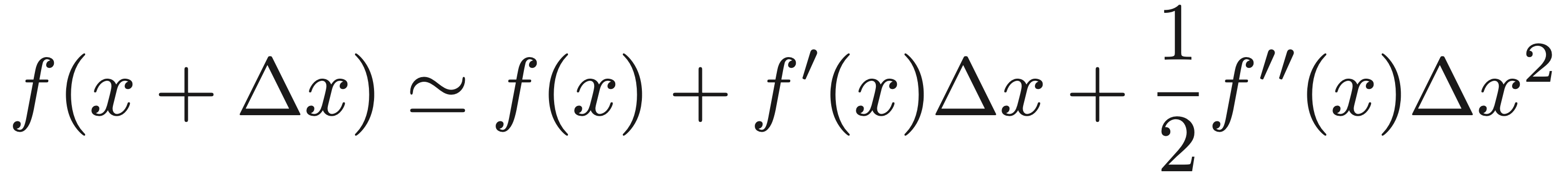
事实上，如果不考虑工程实现、解决问题上的一些差异，xgboost与gbdt比较大的不同就是目标函数的定义。xgboost的目标函数如下图所示：

目标函数

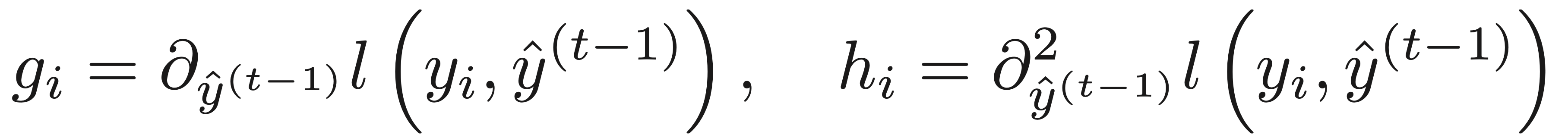


用泰勒展开来近似目标函数：

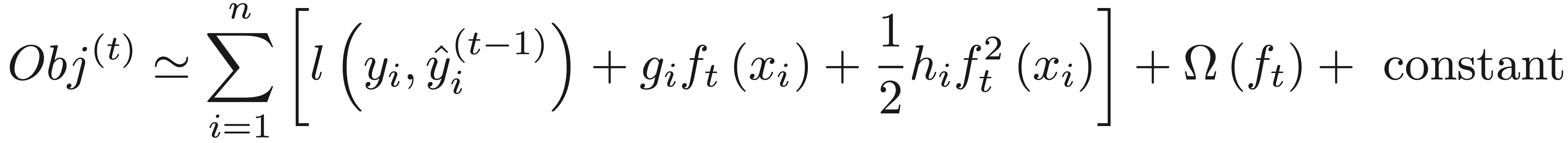
泰勒展开公式：



定义两个函数：



得到的近似目标函数就是：



目标函数分为三部分组成：

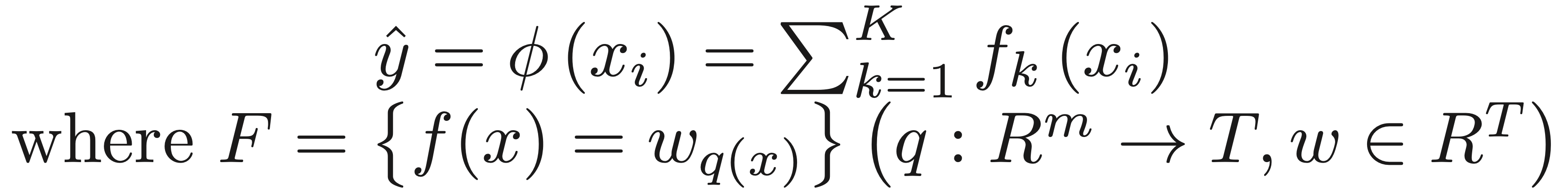
1. 损失函数，可以是平方损失函数，或者是logistic损失函数；
2. 正则项，可以是L1正则或者L2正则；
3. 最后一个就是常数项；

最终的目标函数只依赖于每个数据点的在误差函数上的一阶导数和二阶导数。

### 4.2 xgboost目标函数

xgboost的核心算法思想不难，基本就是

1. 不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。



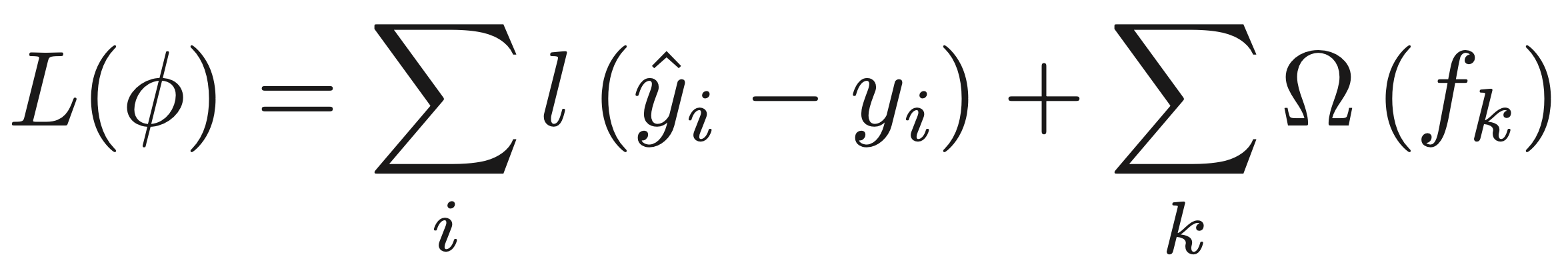
注：w\_q(x)为叶子节点q的分数，F 对应了所有K棵回归树（regression tree）的集合，而f(x)为其中一棵回归树。

2、当我们训练完成得到k棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数

3、最后只需要将每棵树对应的分数加起来就是该样本的预测值。

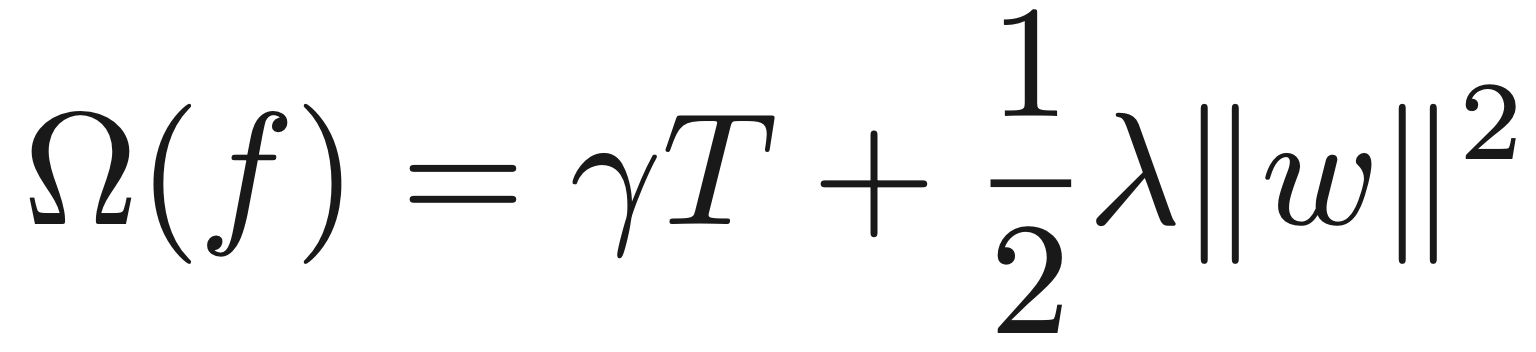
显然，我们的目标是要使得树群的预测值 IMG_273尽量接近真实值IMG_274，而且有尽量大的泛化能力。

所以，从数学角度看这是一个泛函最优化问题，故把目标函数简化如下：



如你所见，这个目标函数分为两部分：误差函数和正则化项。且误差/损失函数揭示训练误差（即预测分数和真实分数的差距），正则化定义复杂度。

对于上式而言，是整个累加模型的输出，正则化项IMG_277是则表示树的复杂度的函数，值越小复杂度越低，泛化能力越强，其表达式为

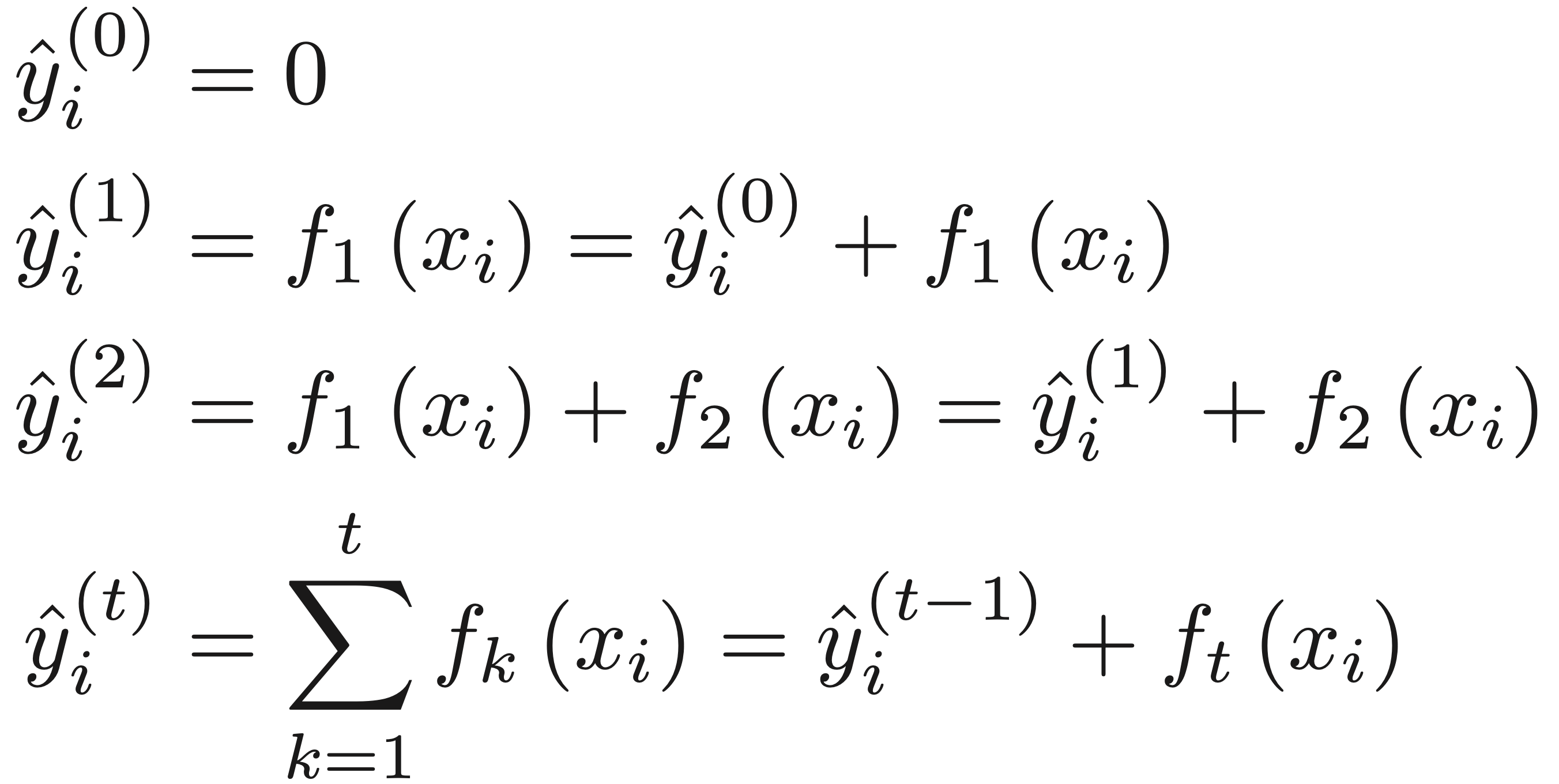


T表示叶子节点的个数，w表示叶子节点的分数。直观上看，目标要求预测误差尽量小，且叶子节点T尽量少（γ控制叶子结点的个数），节点数值w尽量不极端（λ控制叶子节点的分数不会过大），防止过拟合。

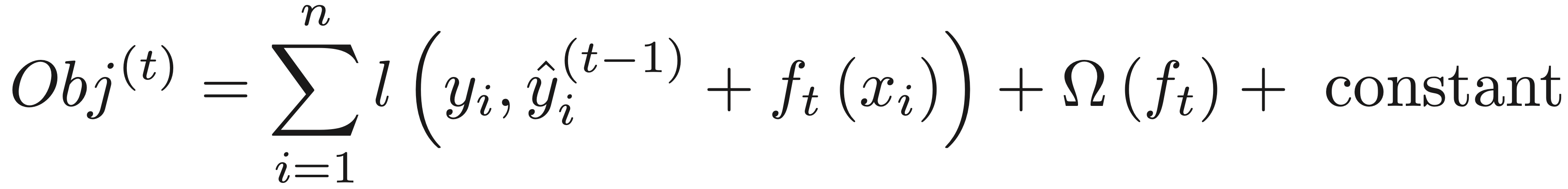
### 4.2.1 模型学习与训练误差

具体来说，目标函数第一部分中的IMG_280表示第IMG_281个样本，IMG_282(IMG_283−IMG_284) 表示第IMG_285个样本的预测误差，我们的目标当然是误差越小越好。

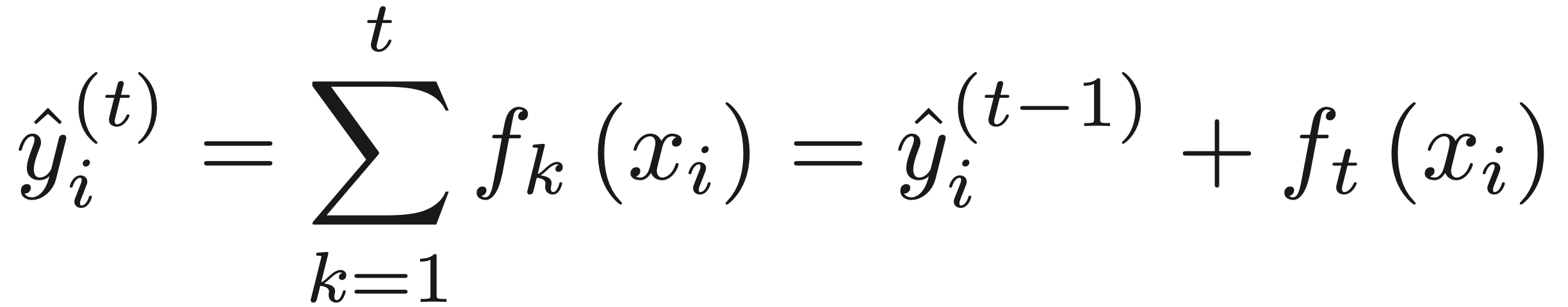
类似之前GBDT的套路，xgboost也是需要将多棵树的得分累加得到最终的预测得分（每一次迭代，都在现有树的基础上，增加一棵树去拟合前面树的预测结果与真实值之间的残差）。



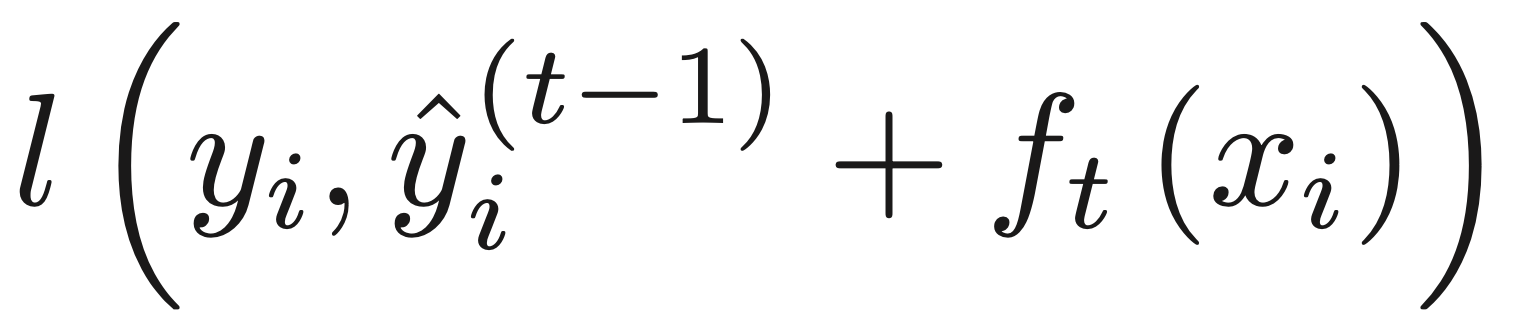
那接下来，我们如何选择每一轮加入什么 f 呢？答案是非常直接的，选取一个 f 来使得我们的目标函数尽量最大地降低。



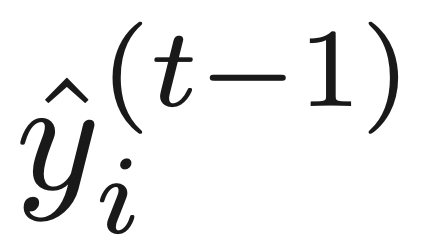
再强调一下，考虑到第t轮的模型预测值



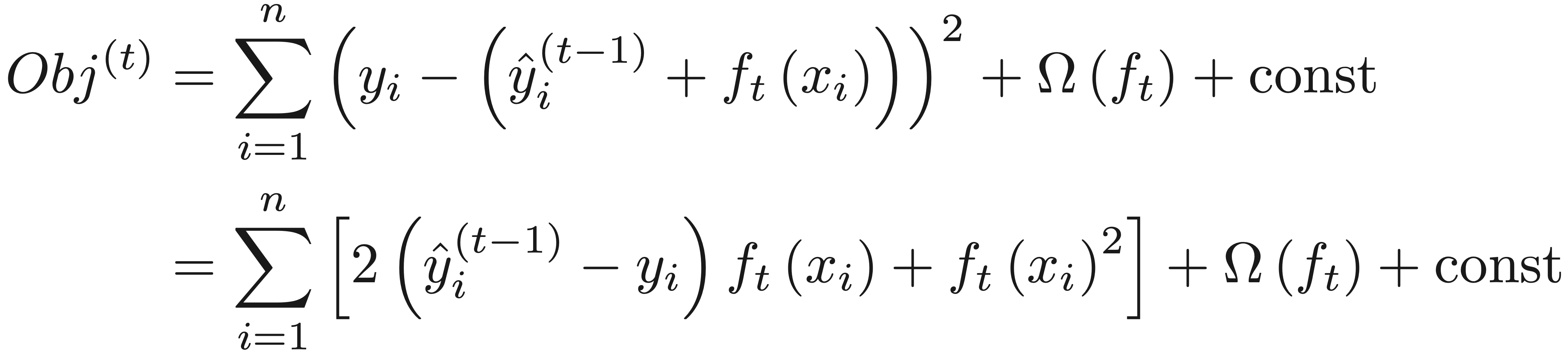
因此误差函数记为



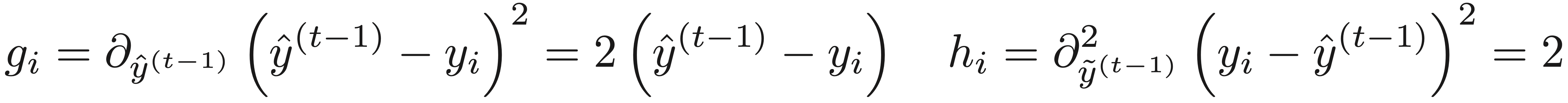
后面一项为正则化项。

对于这个误差函数的式子而言，在第t步，IMG_295是真实值，即已知，可由上一步第t-1步中的IMG_297加上IMG_298计算所得，某种意义上也算已知值，故模型学习的是IMG_299。

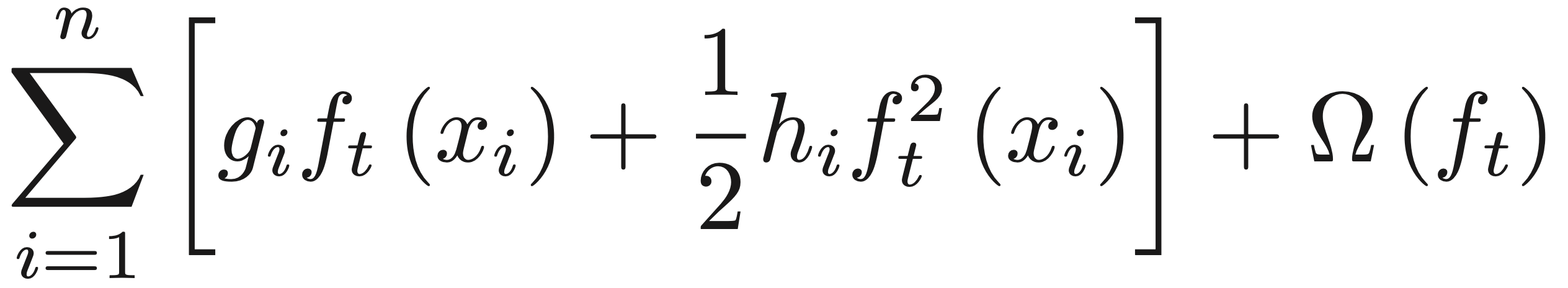
上面那个Obj公式可能有些过于抽象，我们可以考虑当 IMG_300是平方误差的情况（相当于IMG_301），这个时候我们的目标可以被写成下面这样的二次函数（图中画圈的部分表示的就是预测值和真实值之间的残差）：



更加一般的，损失函数不是二次函数咋办？泰勒展开，不是二次的想办法近似为二次（如你所见，定义了一阶导g和二阶导h）。



接下来，考虑到我们的第t 颗回归树是根据前面的t-1颗回归树的残差得来的，相当于t-1颗树的值IMG_320是已知的。换句话说，IMG_321对目标函数的优化不影响，可以直接去掉，且常数项也可以移除，从而得到如下一个比较统一的目标函数。



这时，目标函数只依赖于每个数据点在误差函数上的一阶导数g和二阶导数h总的指导原则如就职Google的读者crab6789所说：实质是把样本分配到叶子结点会对应一个obj，优化过程就是obj优化。也就是分裂节点到叶子不同的组合，不同的组合对应不同obj，所有的优化围绕这个思想展开

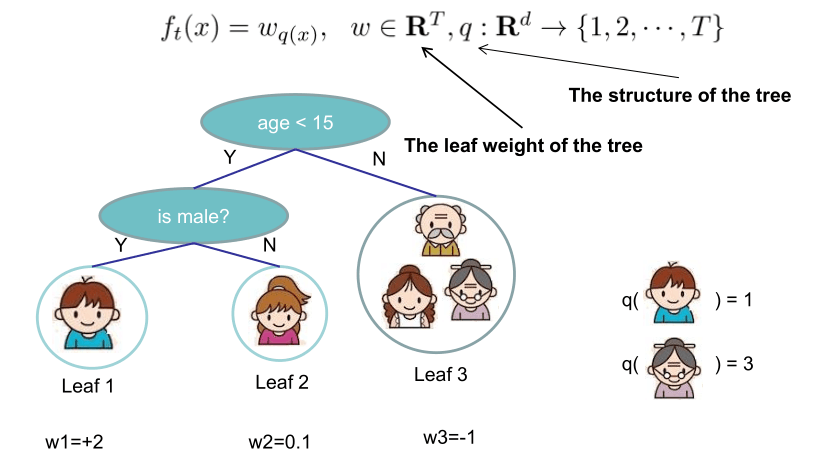
到目前为止我们讨论了目标函数中的第一个部分：训练误差。

接下来我们讨论目标函数的第二个部分：正则项，即如何定义树的复杂度。

### 4.2.2 正则项：树的复杂度

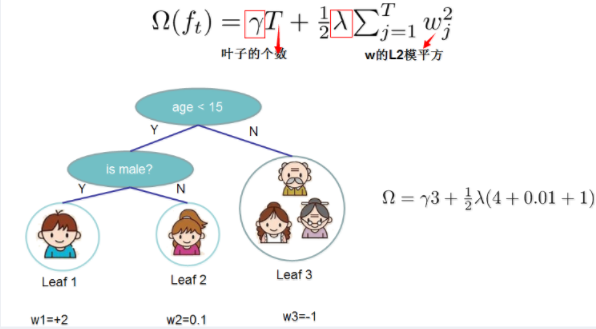
首先有以下概念：

1. 用叶子节点集合以及叶子节点得分表示
2. 每个样本都落在一个叶子节点上
3. q(x)表示样本x在某个叶子节点上，wq(x)是该节点的打分,即该样本的模型预测值

所以当我们把树成结构部分q和叶子权重部分w后，结构函数q把输入映射到叶子的索引号上面去，而w给定了每个索引号对应的叶子分数是什么。

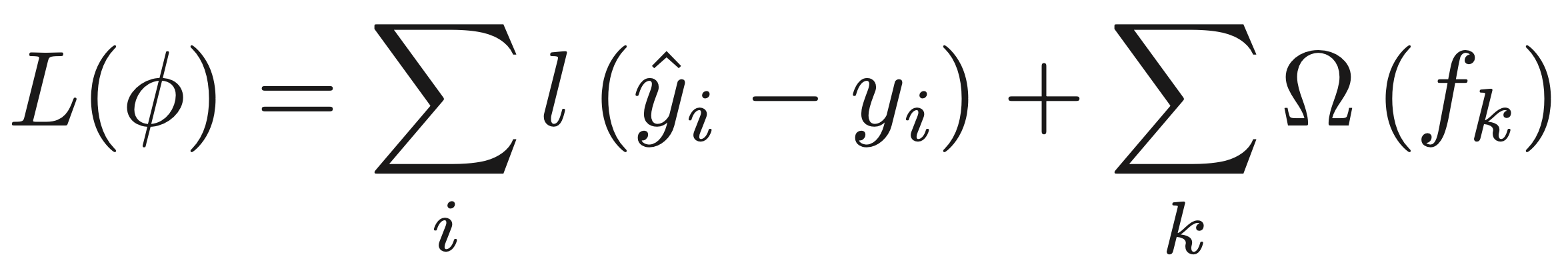
另外，如下图所示，xgboost对树的复杂度包含了两个部分：

1. 一个是树里面叶子节点的个数T
2. 一个是树上叶子节点的得分w的L2模平方（对w进行L2正则化，相当于针对每个叶结点的得分增加L2平滑，目的是为了避免过拟合）



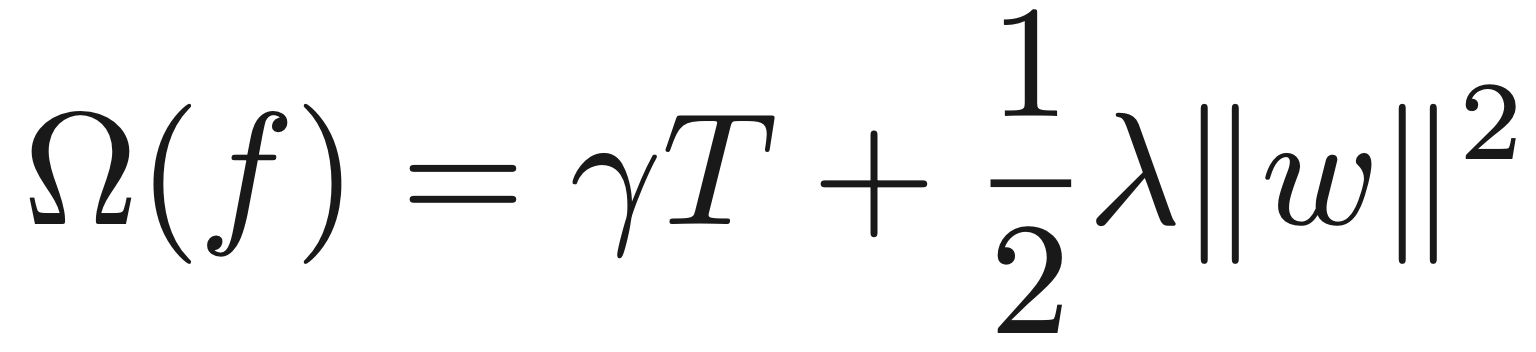
还记得4.2节开头对目标函数的说明吧（损失函数揭示训练误差 + 正则化定义复杂度）？

从数学角度看这是一个泛函最优化问题，故把目标函数简化如下：



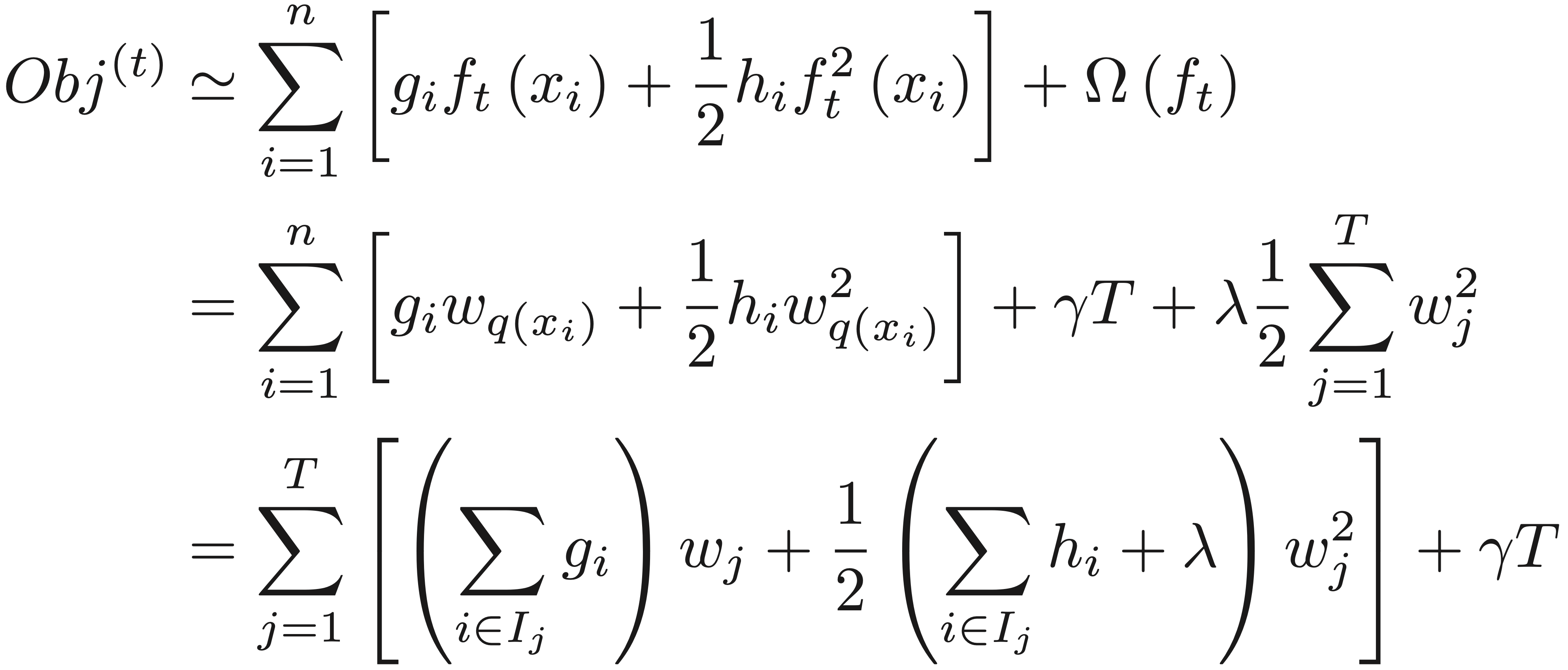
如你所见，这个目标函数分为两部分：误差函数和正则化项。且误差/损失函数揭示训练误差（即预测分数和真实分数的差距），正则化定义复杂度。

对于上式而言，IMG_326是整个累加模型的输出，正则化项∑kΩ(fk)是则表示树的复杂度的函数，值越小复杂度越低，泛化能力越强，其表达式为



T表示叶子节点的个数，w表示叶子节点的分数。直观上看，目标要求预测误差尽量小，且叶子节点T尽量少（γ控制叶子结点的个数），节点数值w尽量不极端（λ控制叶子节点的分数不会过大），防止过拟合。

在这种新的定义下，我们可以把之前的目标函数进行如下变形（另，别忘了：



其中IMG_329被定义为每个叶节点j上面样本下标的集合，g是一阶导数，h是二阶导数。这一步是由于xgboost目标函数第二部分加了两个正则项，一个是叶子节点个数(T),一个是叶子节点的分数(w)。

从而，加了正则项的目标函数里就出现了两种累加

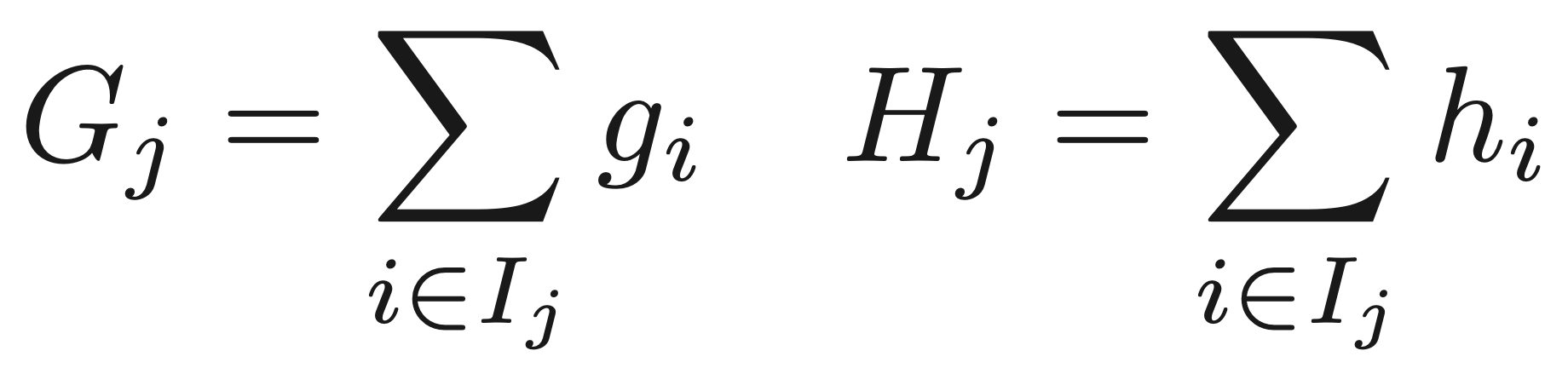
一种是IMG_331 - > n（样本数）

一种是IMG_332 -> T（叶子节点数）

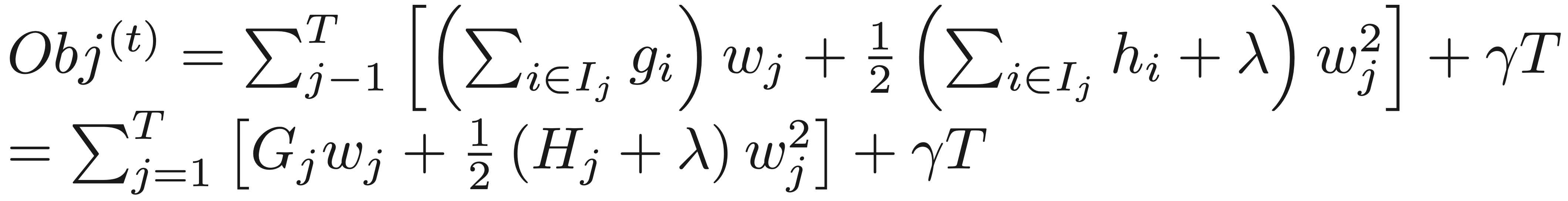
这一个目标包含了T个相互独立的单变量二次函数。

理解这个推导的关键在哪呢？在和AI lab陈博士讨论之后，其实就在于理解这个定义：IMG_333被定义为每个叶节点 j 上面样本下标的集合，这个定义里的q(xi)要表达的是：每个样本值xi 都能通过函数q(xi)映射到树上的某个叶子节点，从而通过这个定义把两种累加统一到了一起。

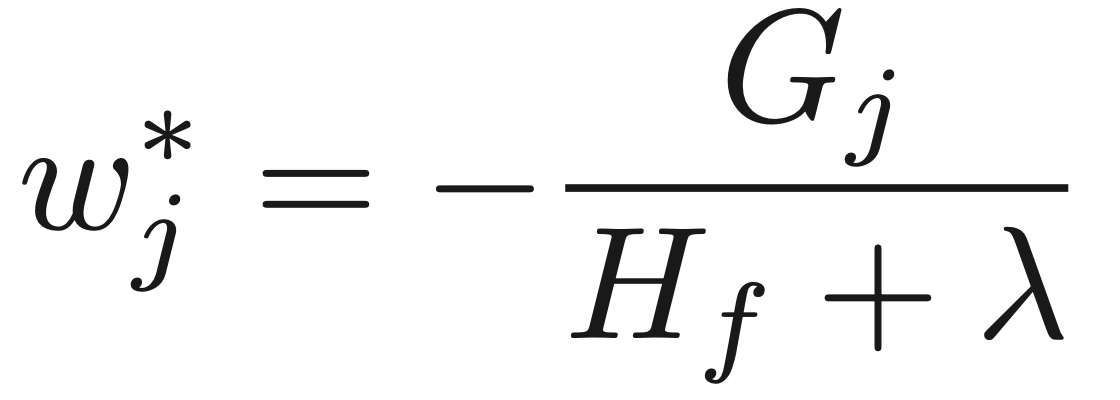
接着，我们可以定义



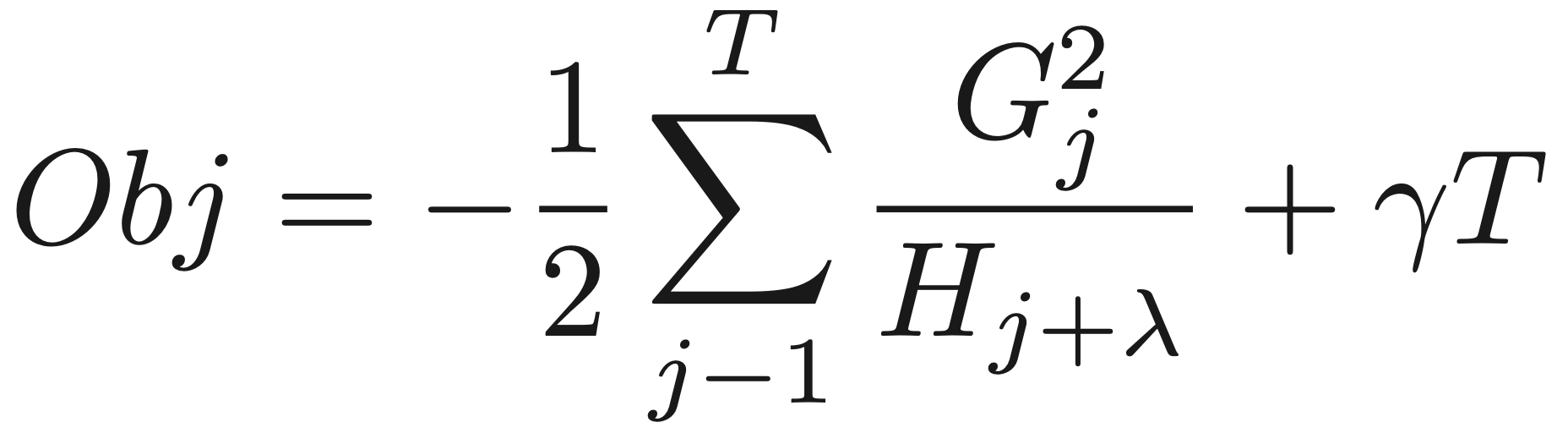
最终公式可以化简为



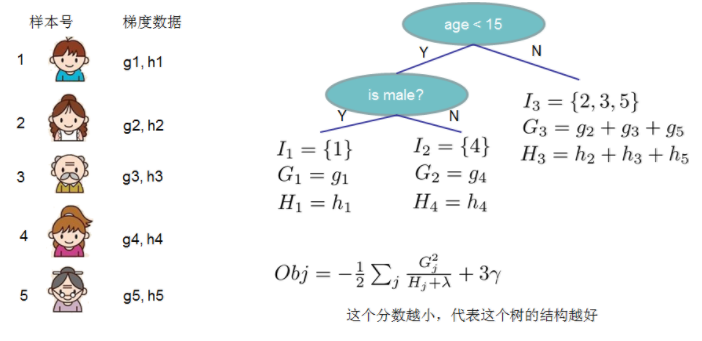
通过对wj求导等于0，可以得到



然后把wj最优解代入得到：



### 4.3 打分函数计算

Obj代表了当我们指定一个树的结构的时候，我们在目标上面最多少多少。我们可以把它叫做结构分数(structure score)

### 4.3.1 分裂节点

我们从头到尾了解了xgboost如何优化、如何计算，但树到底长啥样，我们却一直没看到。很显然，一棵树的生成是由一个节点一分为二，然后不断分裂最终形成为整棵树。那么树怎么分裂的就成为了接下来我们要探讨的关键。

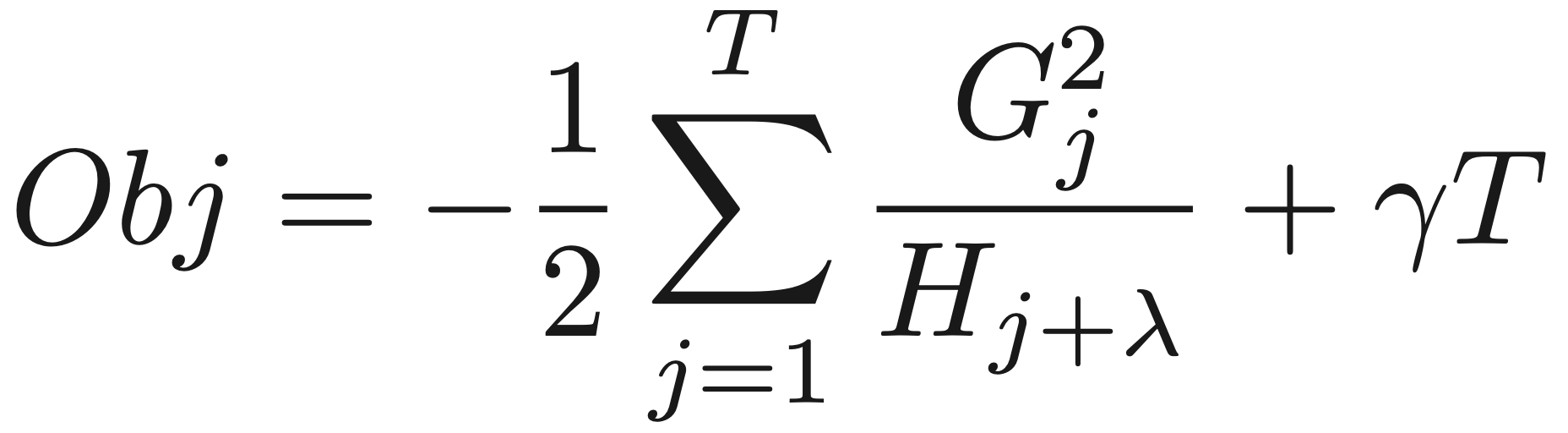
对于一个叶子节点如何进行分裂，xgboost作者在其原始论文中给出了两种分裂节点的方法

（1）枚举所有不同树结构的贪心法

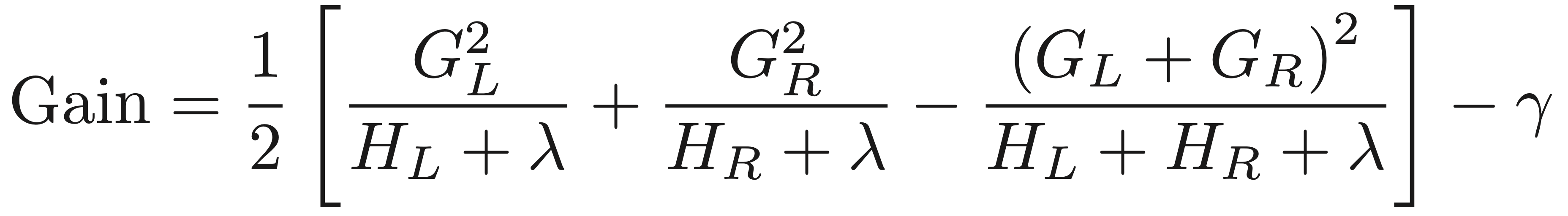
现在的情况是只要知道树的结构，就能得到一个该结构下的最好分数，那如何确定树的结构呢？

一个想当然的方法是：不断地枚举不同树的结构，然后利用打分函数来寻找出一个最优结构的树，接着加入到模型中，不断重复这样的操作。而再一想，你会意识到要枚举的状态太多了，基本属于无穷种，那咋办呢？

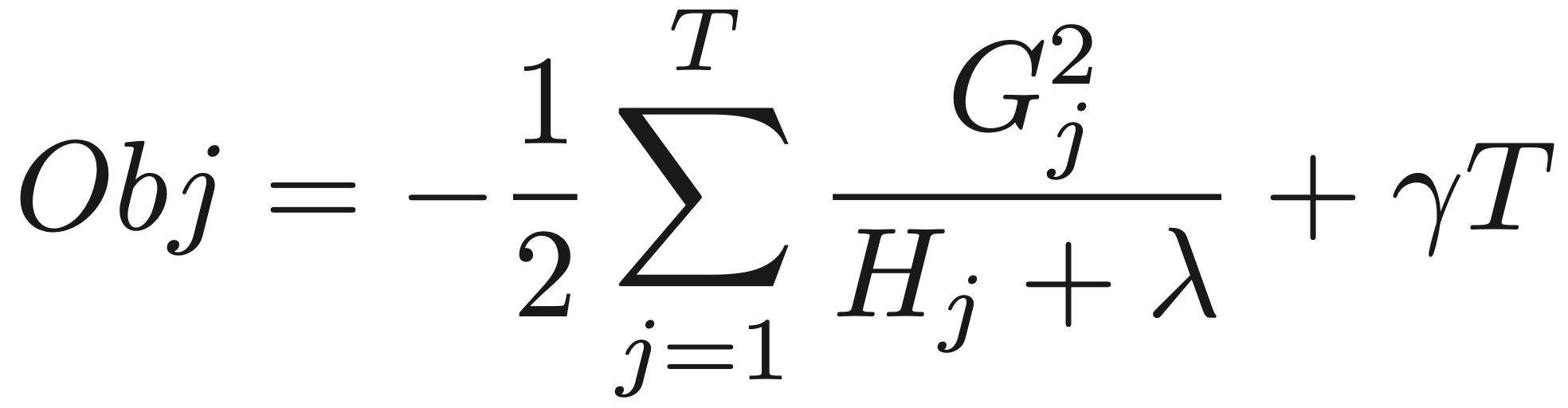
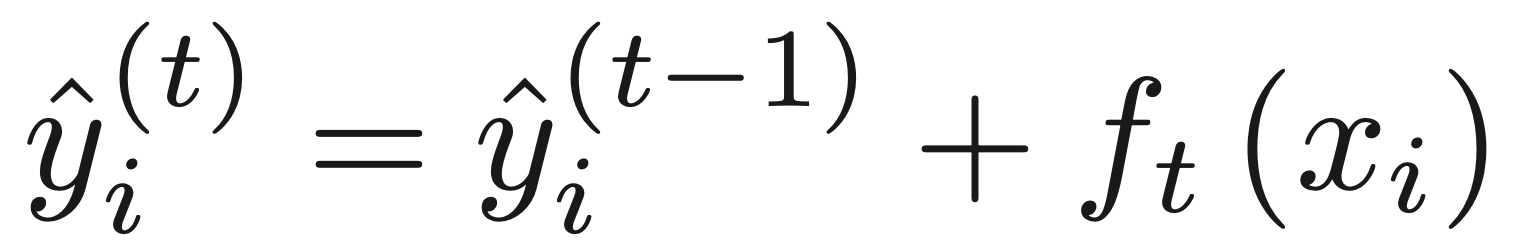
我们试下贪心法，从树深度0开始，每一节点都遍历所有的特征，比如年龄、性别等等，然后对于某个特征，先按照该特征里的值进行排序，然后线性扫描该特征进而确定最好的分割点，最后对所有特征进行分割后，我们选择所谓的增益Gain最高的那个特征，而Gain如何计算呢？



换句话说，目标函数中的G/(H+λ)部分，表示着每一个叶子节点对当前模型损失的贡献程度，融合一下，得到Gain的计算表达式，如下



### 4.4 总结

1. 在每一次迭代过程中增加一棵新树；
2. 在每一次迭代开始计算Gi和Hi
3. 利用这个统计数据贪心生成一棵树，目标函数如下
4. 

可以预测下一个y值，继续迭代。

上面这个目标函数跟实际的参数怎么联系起来，记得我们说过，回归树的参数：

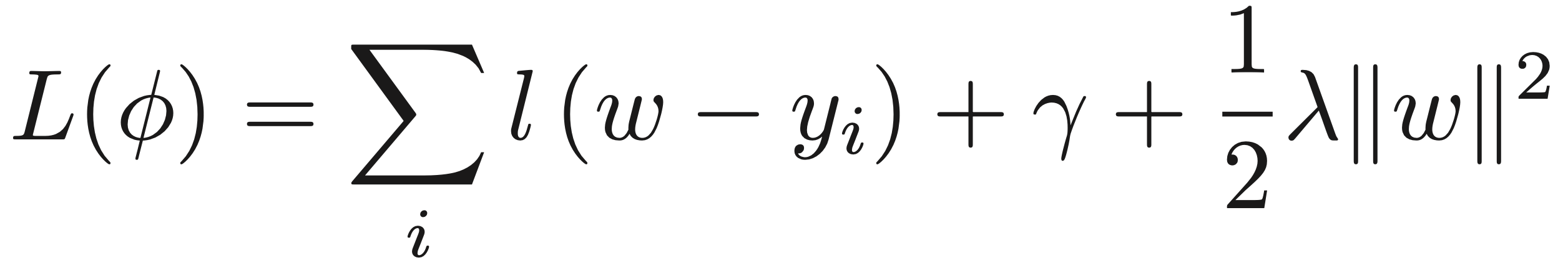
1、选取哪个feature分裂节点呢

2、节点的预测值（总不能靠取平均值这么粗暴不讲道理的方式吧，好歹高级一点）

最终的策略就是：贪心 + 最优化（对的，二次最优化） 。

通俗解释贪心策略：就是决策时刻按照当前目标最优化决定，说白了就是眼前利益最大化决定，“目光短浅”策略。

这里是怎么用贪心策略的呢，刚开始你有一群样本，放在第一个节点，这时候T=1，w多少呢，不知道，是求出来的，这时候所有样本的预测值都是w,带入样本的label数值，此时loss function变为



• 如果这里的l(w−yi)误差表示用的是平方误差，那么上述函数就是一个关于w的二次函数求最小值，取最小值的点就是这个节点的预测值，最小的函数值为最小损失函数。

• 本质上来讲，这就是一个二次函数最优化问题！但要是损失函数不是二次函数咋办？泰勒展开，不是二次的想办法近似为二次。

接着来，接下来要选个feature分裂成两个节点，变成一棵弱小的树苗，那么需要

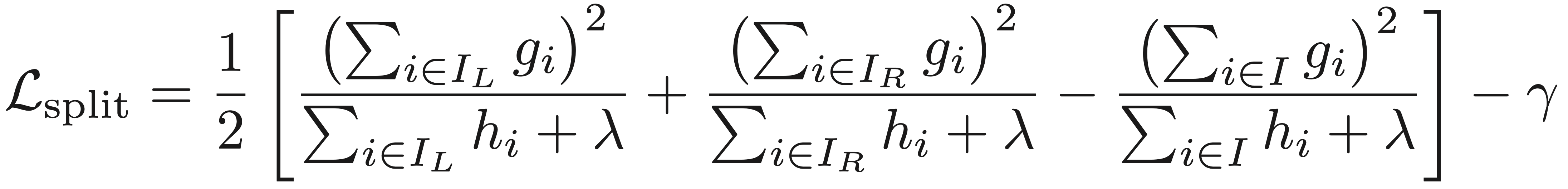
1、确定分裂用的feature，how？最简单的是粗暴的枚举/穷举（嗯，够粗暴），然后选择loss function效果最好的那个；

2、如何确立节点的w以及最小的loss function，大声告诉我怎么做？对，二次函数的求最值（计算二次的最值一般都有固定套路，即导数等于0的点） 。所以，选择一个feature分裂，计算loss function最小值，然后再选一个feature分裂，又得到一个loss function最小值，你枚举完，找一个效果最好的，把树给分裂，就得到了小树苗。

在分裂的时候，你可以注意到，每次节点分裂，loss function被影响的只有这个节点的样本，因而每次分裂，计算分裂的增益（loss function的降低量）只需要关注打算分裂的那个节点的样本。

总而言之，XGBoost使用了和CART回归树一样的想法，利用贪婪算法，遍历所有特征的所有特征划分点，不同的是使用的目标函数不一样。具体做法就是分裂后的目标函数值比单子叶子节点的目标函数的增益，同时为了限制树生长过深，还加了个阈值，只有当增益大于该阈值才进行分裂。

以下便为设定的阈值



从而继续分裂，形成一棵树，再形成一棵树，每次在上一次的预测基础上取最优进一步分裂/建树，是不是贪心策略？

凡是这种循环迭代的方式必定有停止条件，什么时候停止呢？简言之，设置树的最大深度、当样本权重和小于设定阈值时停止生长以防止过拟合。具体而言，则

1、当引入的分裂带来的增益小于设定阀值的时候，我们可以忽略掉这个分裂，所以并不是每一次分裂loss function整体都会增加的，有点预剪枝的意思，阈值参数为（即正则项里叶子节点数T的系数）；

2、当树达到最大深度时则停止建立决策树，设置一个超参数max\_depth，避免树太深导致学习局部样本，从而过拟合；

3、当样本权重和小于设定阈值时则停止建树。什么意思呢，即涉及到一个超参数-最小的样本权重和min\_child\_weight，和GBM的 min\_child\_leaf 参数类似，但不完全一样。大意就是一个叶子节点样本太少了，也终止同样是防止过拟合；

## 5 总结

通过对xgboost算法的探索以及数学原理分析，我们学习并且了解来xgboost的整个来龙去脉，从基本的决策树开始，到回归树与分类树的构建，再形成基本的提升树，最后再提升树的基础之上，加上正则化项，迭代生成一棵回归树，这就是xgboost。

## 参考文献：

[1]、Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen. (1984). Classification and Regression Trees. Chapman & Hall/CRC, Boca Raton, FL.

[2]、Brodley, C. E. and P. E. Utgoff. (1995). “Multivariate decision trees.” Machine Learning, 19(1):45-77.

[3]、Guo, H. and S. B. Gelfand. (1992). “Classification trees with neural network feature extraction.” IEEE Transactions on Neural Networks, 3(6):923-933.

[4]、Mingers, J. (1989a). “An empirical comparison of pruning methods for decision tree induction.” Machine Learning, 4(2):227-243.

[5]、Mingers, J. (1989b). “An empirical comparison of selection measures for

decision-tree induction.” Machine Learning, 3(4):319-342.

[6]、Murthy, S. K. (1998). “Automatic construction of decision trees from data

A multi-disciplinary survey.” Data Mining and Knowledge Discovery, 2(4)

345-389.

[7]、Murthy, S. K., S. Kasif, and S. Salzberg. (1994). “A system for induction ofoblique decision trees.” Journal of Artificial Intelligence Research, 2:1-32.

[8]、Quinlan, J. R. (1979). “Discovering rules by induction from large collections ofexamples.” In Expert Systems in the Micro-electronic Age (D. Michie, ed.),

168-201, Edinburgh University Press, Edinburgh, UK

[9]、Breiman, L. (1996a). “Bagging predictors.” Machine Learning, 24(2):123-140.

[10]、Breiman, L. (1996b). “Stacked regressions.” Machine Learning, 24(1):49-64

[11]、Breiman, L. (2000). “Randomizing outputs to increase prediction accuracy.”

Machine Learning, 40(3):113-120

[12]、Breiman, L. (2001a). “Random forests.” Machine Learning, 45(1):5 32.

[13]、Breiman, L. (2001b). “Using iterated bagging to debias regressions.” Machine Learning, 45(3):261-277

[14]、Clarke, B. (2003). “Comparing Bayes model averaging and stacking when model approximation error cannot be ignored.” Journal of Machine Learning

Research, 4:683-712.

[15]、Demiriz, A., K. P. Bennett, and J. Shawe-Taylor. (2008). “Linear programming Boosting via column generation.” Machine Learning, 46(1-3):225-254

Dietterich, T. G. (2000). “Ensemble methods in machine learning.” In Pro-

ceedings of the 1st International Workshop on Multiple Classifier Systems

(MCS), 1-15, Cagliari, Italy.

[16]、Dietterich, T. G. and G. Bakiri. (1995). “Solving multiclass learning problems via error-correcting output codes.” Journal of Artificial Intelligence Re

search, 2:263-286.

[17]、Freund, Y. and R. E. Schapire. (1997). “A decision-theoretic generalization of on-line learning and an application to boosting.” Journal of Computer and

System Sciences, 55(1):119-139.