数据建模：从新手到老司机

朱杰、李奇倚@大数据中心

通过本课程能学到什么

- CART（分类回归树）基本概念
- XGBoost算法基本原理
- XGBoost如何调节参数
- LightGBM与XGBoost的区别

逻辑回归建模常见场景

userid	芝麻分	户籍所在地	是否有房产	…	30+是否逾期
0001	265	山西	是	…	是
0002	351	河北	是	…	否
0003	422	湖南	否	…	否
0004	377	广西	否	…	是
…	…	…	…	…	…

建立模型，来预测新用户逾期的可能性

逻辑回归建模流程

逻辑回归的建模步骤

- 处理缺失值
- 数值型变量：分Bin，计算WOE
- 字符型变量：Onehot编码/WOE
- 去除相关性过高的变量
- 筛选变量
- 建立模型

优点

- 变量可解释性强

缺点

- 过程繁琐
- 只能捕捉线性关系，结果并非最优

今天要介绍的建模方法

优点很多：

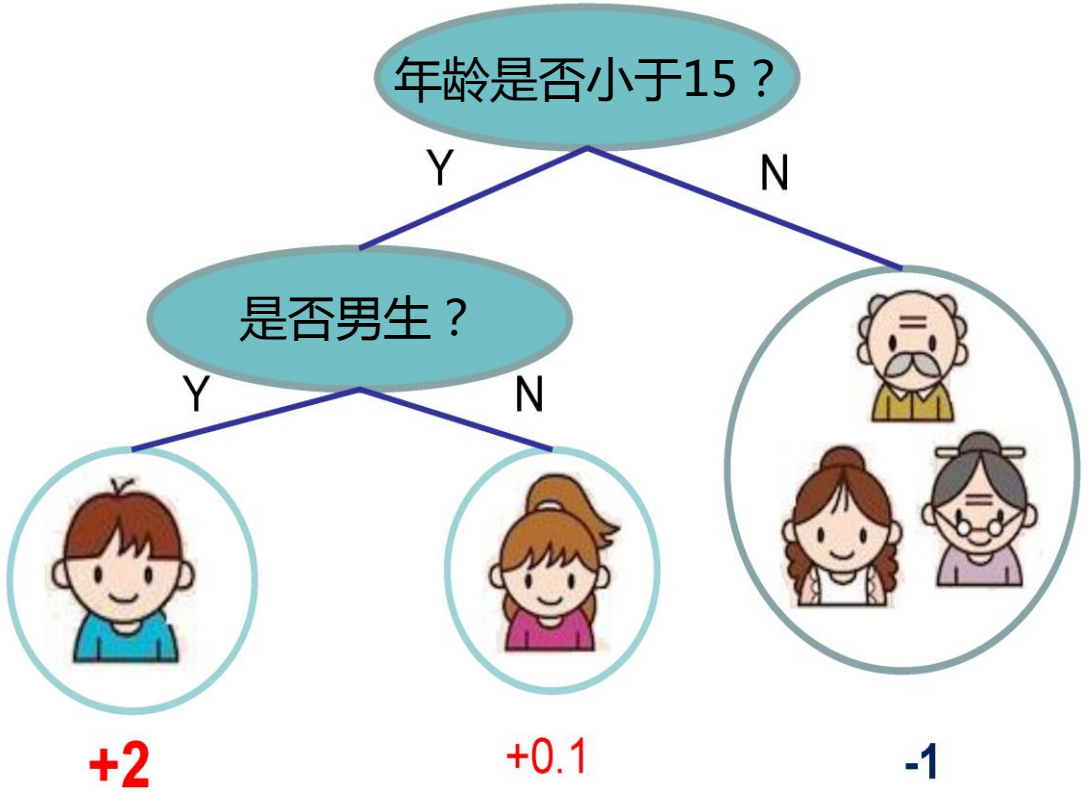
- 自动筛选特征
- 自动处理缺失值
- 减少人工处理特征的步骤
- 预测效果更好
- 可以拟合非线性关系

CART（分类回归树）

- 全称是“Classification And Regression Tree”
- 这里我们举一个回归的例子：

- 知道各家庭成员的性别，年龄，职业等信息
- 预测他们对电脑游戏的喜爱程度

CART（分类回归树）

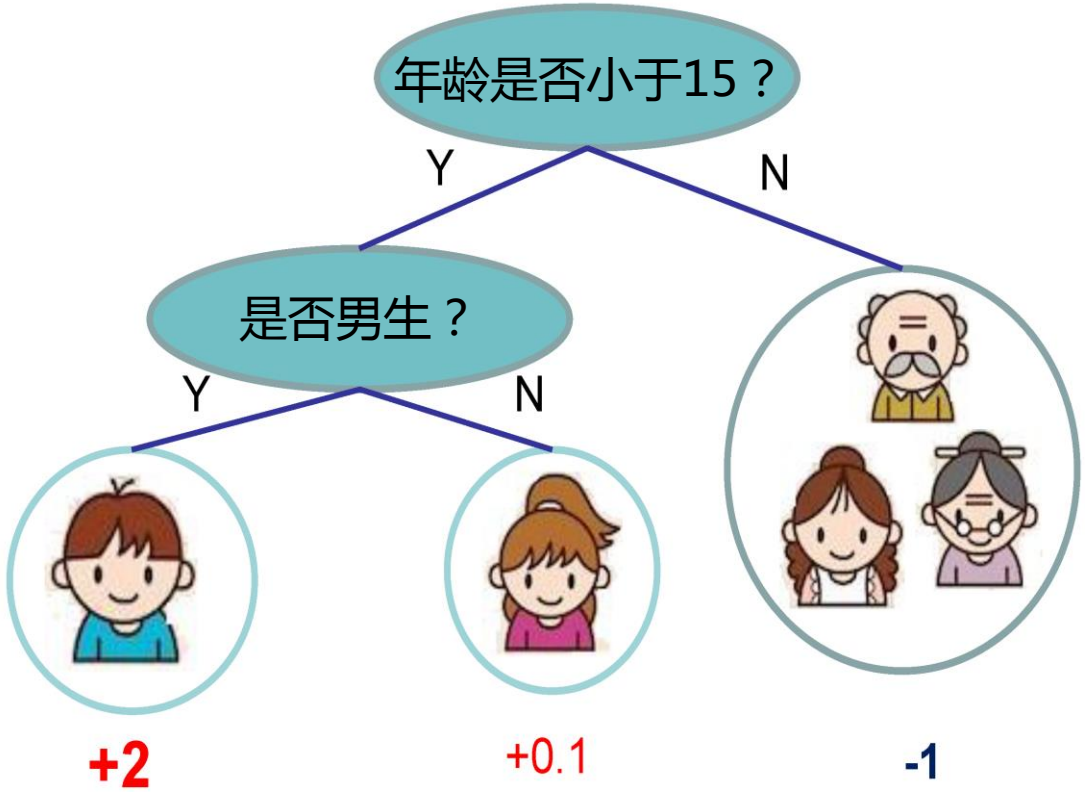


每个叶子节点中都会有一个权重

$$\arg\min_{j,s}\left[\min_{c_1}\sum_{x\in R_1(j,s)}(y-c_1)^2+\min_{c_2}\sum_{x\in R_2(j,s)}(y-c_2)^2\right]$$

- 建一棵二叉树
- 不断寻找最佳分割点，将所有样本分配到叶子节点中
- 根据损失函数（通常是平方误差最小）来计算各叶子节点权重

CART（分类回归树）

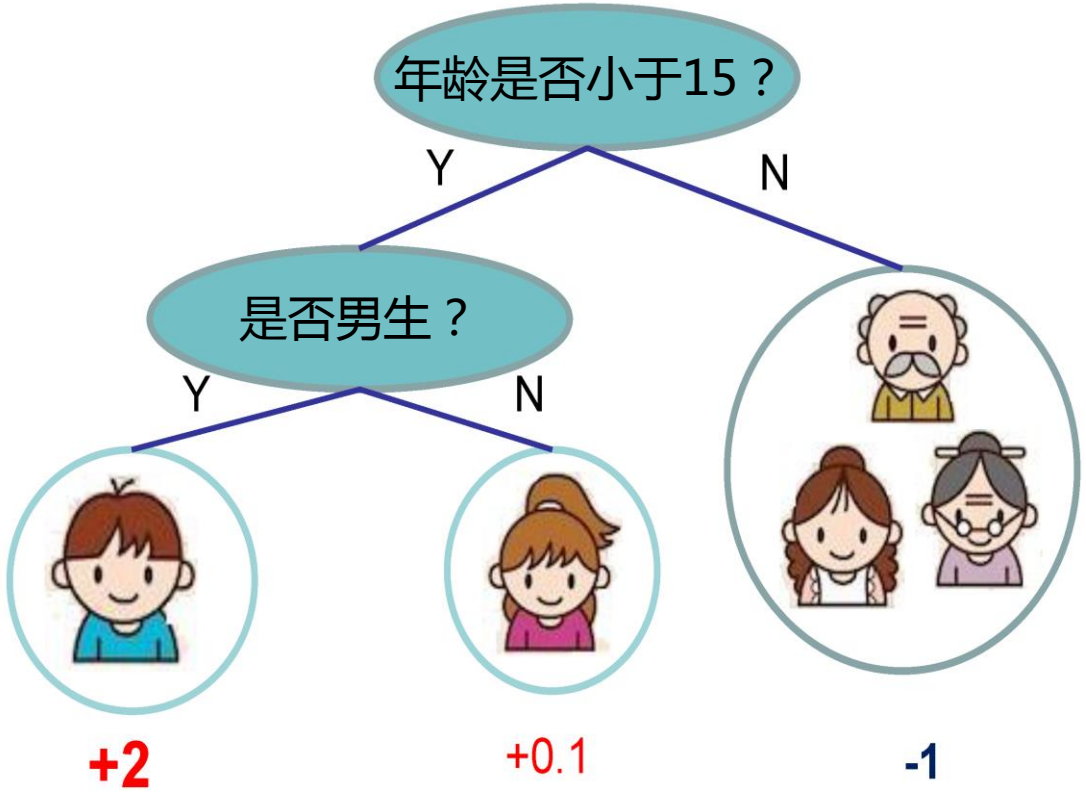


叶子节点的权重怎么算？

$$\arg\min_{y} \sum_{i=1}^{n} (y - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} y_i$$

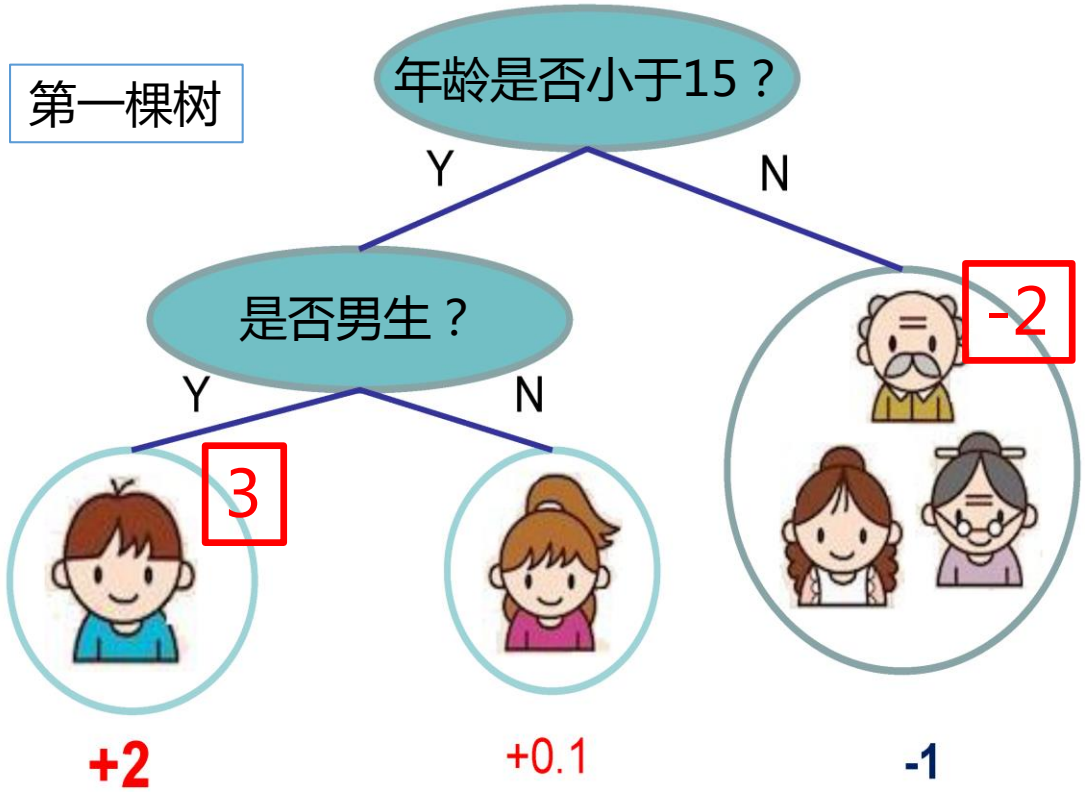
CART+BOOSTING（提升算法）

- 单颗树只能提供有限拟合准确度
- Boosting：将多个弱分类器进行组合
- 建很多棵树，每棵树针对此前拟合残差进行拟合

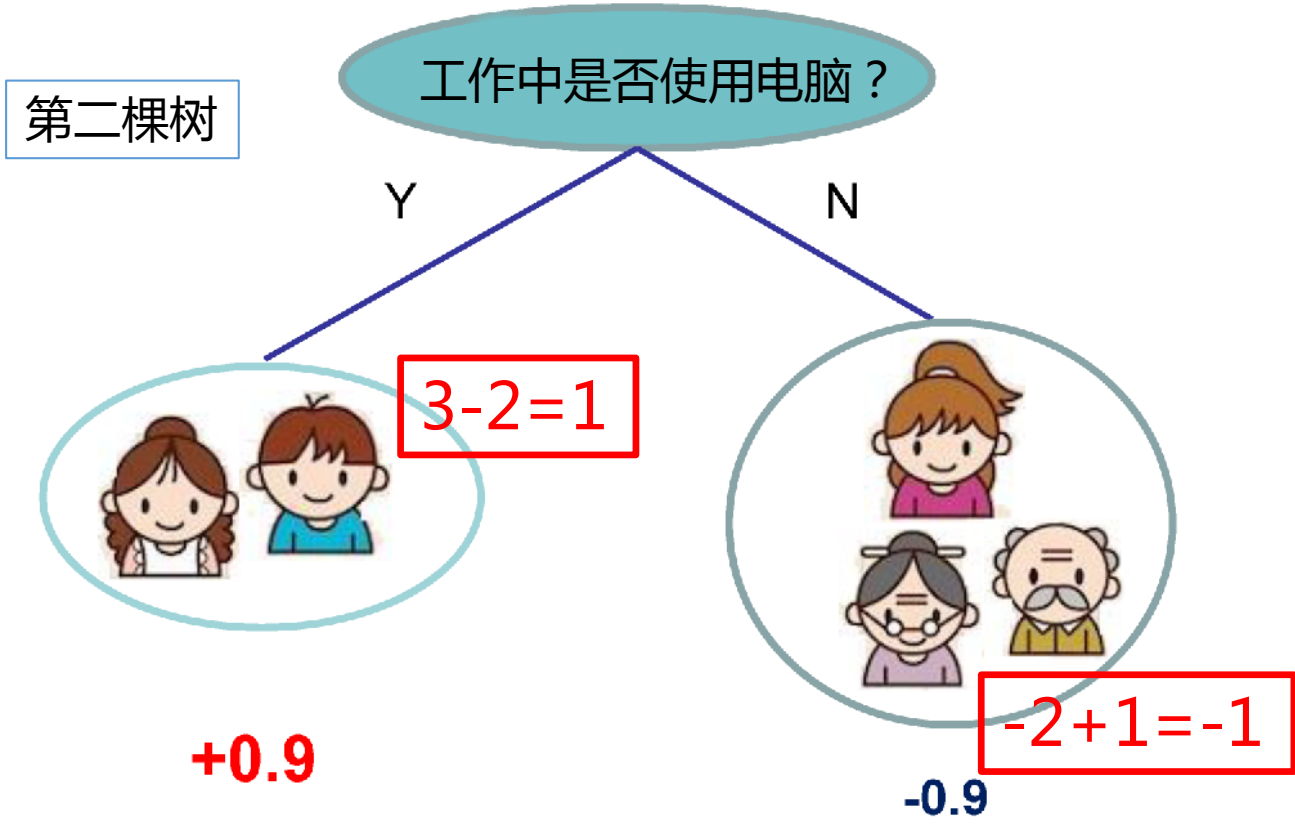


CART+BOOSTING（提升算法）

第一棵树



第二棵树



f() = 2 + 0.9= 2.9　　f()= -1 - 0.9= -1.9

XGBOOST（进阶版CART+BOOSTING）

要思考的几个问题：

- 优化目标是什么？
- 怎样建一棵树？
- 如何防止过拟合？

模型优化目标

假定我们已经有了一个包含K棵树的模型

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in \mathcal{F}$$

包含了所有可能树结构的函数空间

一棵回归树就是一个由feature到score的映射

目标函数设定为

$$Obj = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k)$$

表征数据拟合的质量

表征树的复杂度

这个是常量

如何构建整个模型

每一轮针对当前残差进行拟合：

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

每轮增加一棵新的树

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

$$\cdots$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

第t棵树的优化目标

只考虑第t棵树（此时，前面$(t-1)$棵树的信息可视为已知）

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + \sum_{i=1}^{t-1} \Omega(f_i)$$

这两个已知

这个也已知

目标函数的二阶近似

考虑二阶泰勒展开

$$f(x+\Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

目标函数可以改写为（对Obj的第一部分进行近似）

$$Obj^{(t)} = \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t) + constant$$

其中 $g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$ $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$

大家来动手算一下

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) \quad h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$

对于 $l(\hat{y}, y) = (\hat{y} - y)^2$

$$g_i = \partial_{\hat{y}_i^{(t-1)}} \left(\hat{y}_i^{(t-1)} - y_i\right)^2 = 2\left(\hat{y}_i^{(t-1)} - y_i\right)$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} \left(\hat{y}_i^{(t-1)} - y_i\right)^2 = 2$$

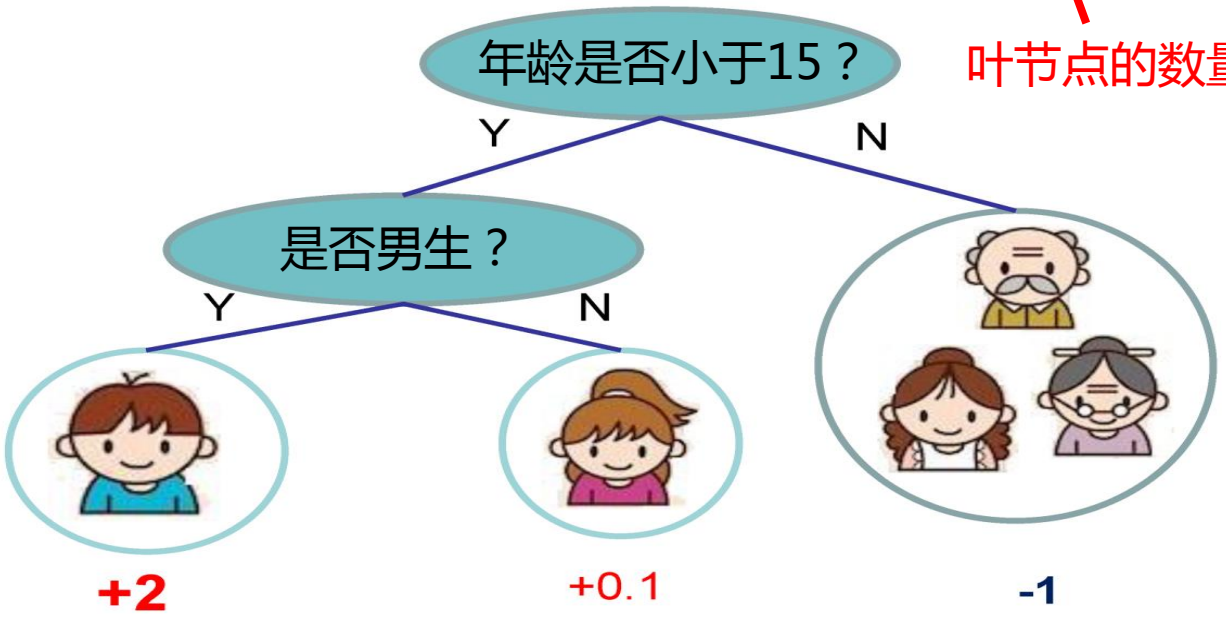
模型复杂度的定义

针对模型复杂度的penalty定义为

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

叶节点权重的L2 norm

叶节点的数量



$$\Omega = \gamma * 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

目标函数的最终形式

重新考虑第t棵树的目标函数

x_i到w_i的映射

$$Obj^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t)$$

做替换

$$= \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right)w_j^2\right] + \gamma T$$

I_j表示第j个叶子节点中的所有样例的集合

改写为叶节点的形式

目标函数的最优解

对目标函数求极值，可得最优解

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

叶子节点的最优权重

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

整棵树对应的最优分数

其中 $G_i = \sum_{i \in I_j} g_j$

第j个叶子节点中， g_j的加和

$H_i = \sum_{i \in I_j} h_j$

第j个叶子节点中， h_j的加和

现在我们可以愉快地建树了^_^

树的生成

每一次节点分裂的收益

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

右节点的分数

增加的树结构复杂度

左节点的分数

父节点的分数

针对每个节点，遍历所有feature，寻找最佳分割点

如何防止过拟合

- 进行列采样
- 控制学习率
- 调整L2正则
- Early stopping
- 控制树的最大深度
- …

接口调用


```
import xgboost as xgb

# 读入数据
dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
dtest = xgb.DMatrix('demo/data/agaricus.txt.test’)

# 设置参数
param = {'max_depth':2, 'eta':1, 'objective':'binary:logistic’ }

# 训练模型
bst = xgb.train(param, dtrain, num_round=2)

# 做预测
preds = bst.predict(dtest)
```

思考题

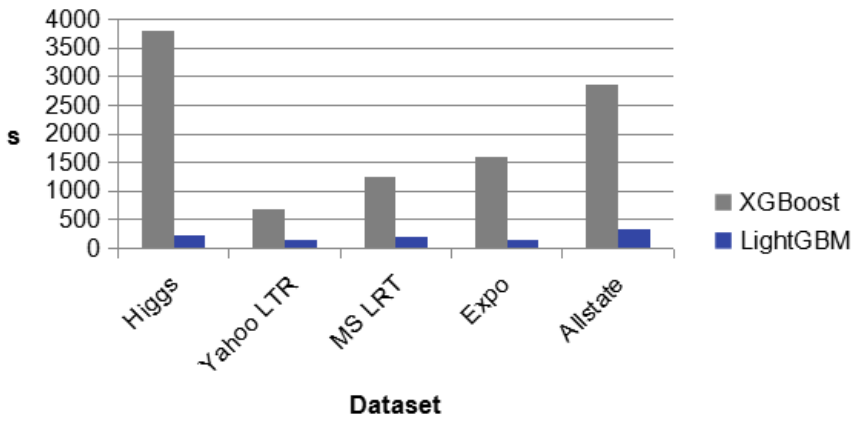
- XGBoost如何处理缺失值？
- 并行计算是怎么回事？

GBDT/XGBOOST在实际中面临的问题

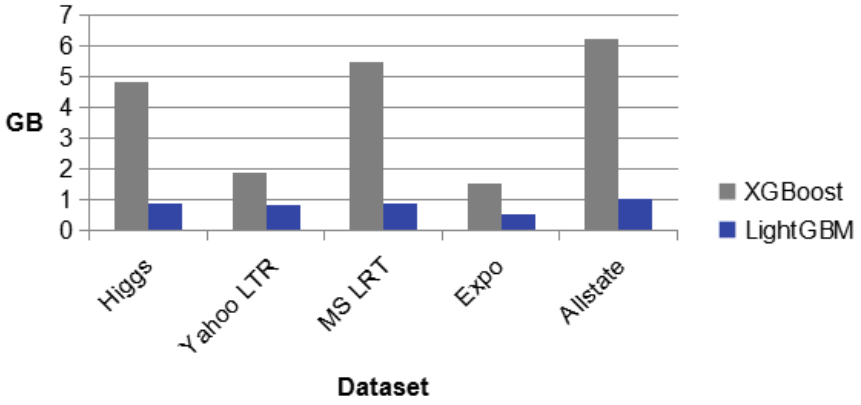
- 千万级的数据量，上千或上万的高维稀疏数据。
- 选定一组训练参数，收敛需要一周甚至更长？
- 服务器几百G的内存 out of memory error？

LightGBM

训练时间：

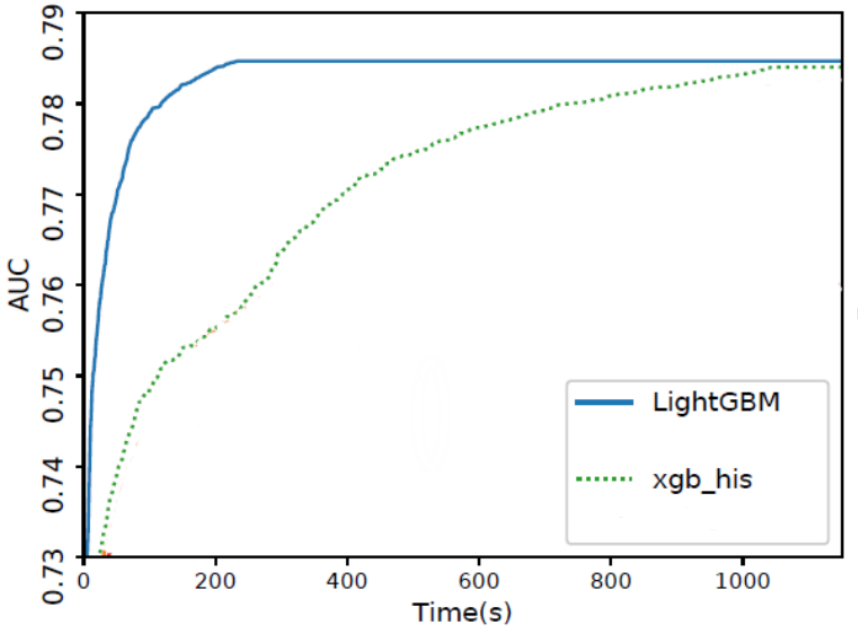


占用内存：



航班延误数据集 500W

	XGBoost	Light BGM	
Parameters Used	max_depth: 50 learning_rate: 0.16 min_child_weight: 1 n_estimators: 200	max_depth: 50 learning_rate: 0.1 num_leaves: 900 n_estimators: 300	
Training AUC Score	0.999	Without passing indices of categorical features	Passing indices of categorical features
		0.992	0.999
Test AUC Score	0.789	0.785	0.772
Training Time	970 secs	153 secs	326 secs
Prediction Time	184 secs	40 secs	156 secs
Parameter Tuning Time (for 81 fits, 200 iteration)	500 minutes	200 minutes	



LightGBM

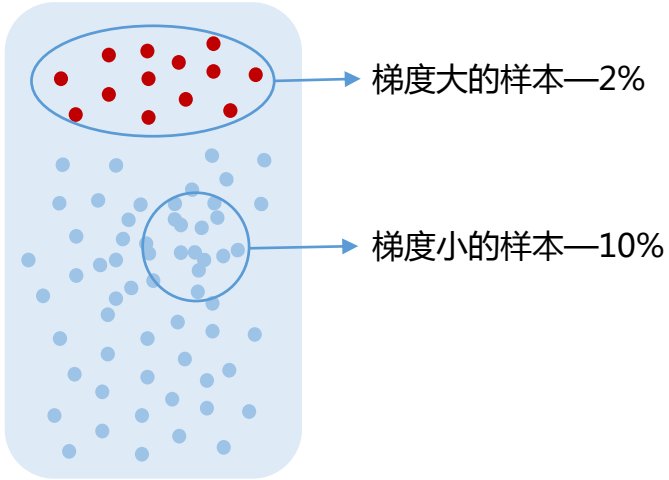
两个出发点：

减少数据量：
GOSS：根据信息增益的定义，拥有大的梯度的实例贡献更多的信息增益，所以为了维持结果的准确性，尽量保留梯度大的实例，而舍弃梯度值小的实例。

减少特征数：
EFB：在真实场景中，高维度特征具有稀疏性，这样可以设计一个减少有效特征数量的无损的方法，特别是在稀疏特征中，许多特征是互斥的，出现大量0，例如one-hot。所以，将互斥特征捆绑

减少数据量-GOSS

- GOSS(Gradient-based One Side Sampling)
  - 在每一次迭代前，利用了GBDT中的样本梯度和误差的关系，对训练样本进行采样：
    - 对误差大（梯度绝对值大）的数据保留；
    - 对误差小的数据采样一个子集，但给这个子集的数据一个权重，让这个子集可以近似到误差小的数据的全集。

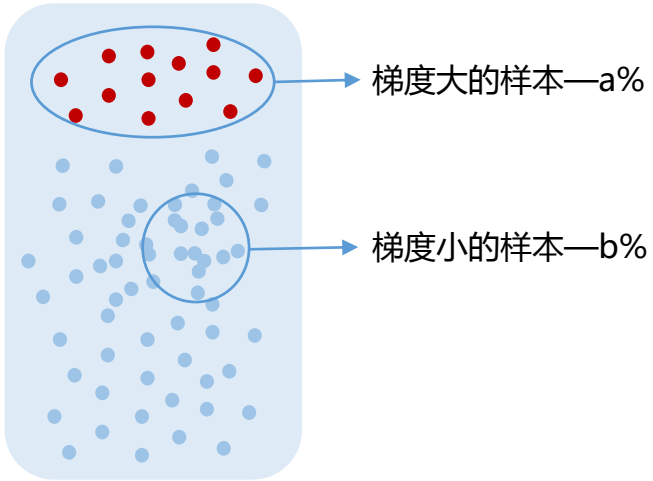


假如有 50 万行数据，其中 1 万行数据的梯度较大。

样本选择方法（这 1 万行梯度很大的数据+x% 从剩余 49 万行中随机抽取的结果）。
若 x %取 10%，那么 1+0.1*49=5.9万

减少数据量-GOSS

- GOSS(Gradient-based One Side Sampling)
  - 采样是否引入新的问题，改变数据的分布？



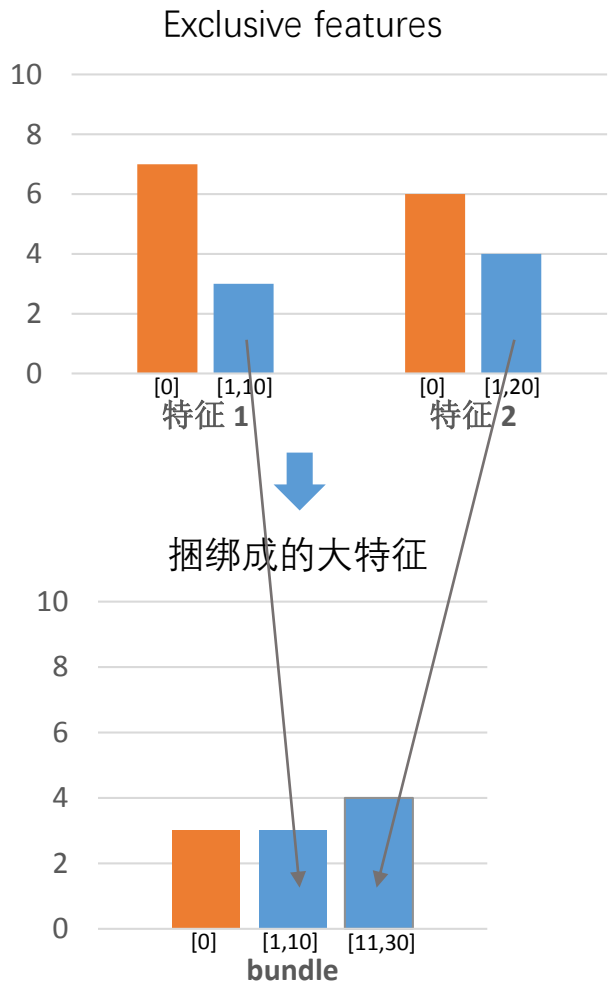
策略：
计算信息增益时，将小梯度样本的信息增益乘以常数（1-a）/b；

证明了这种计算方法不会损失太多精度，并且效果优于随机采样。

减少特征数-EFB

- EFB (Exclusive Feature Bundling)
  - 高特征维度的数据一般比较稀疏，这种稀疏性提供了无损失降低特征维度的可能性。
  - Exclusive features(互斥特征)：对于每一个样本，最多有一个特征不为0。
  - 每一组互斥特征都可以无损地合并成一个“大特征”。

原理举例：



接口调用

- LightGBM API
  - lightgbm.LGBMModel(boosting_type='gbdt', , **kwargs)

- 重要参数：
  - num_leaves 常用策略为num_leaves < 2^(max_depth)
  - min_data_in_leaf

作业

使用XGBoost和LightGBM对数据进行建模，要求

- 结合模型结果，表述模型重要参数对模型结果的影响
- 通过数据，比较XGBoost和LightGBM的异同

作业提交形式：

- Jupyter notebook
- 附带说明（也可直接写在Jupyter notebook中）

作业 – 数据描述

Variable	Definition	Key
survival	Survival	0=No, 1=Yes
pclass	Ticket class	1=1st, 2=2nd, 3=3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings/spouses aboard the Titanic	
parch	# of parents/children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C=Cherbourg, Q=Queenstown, S=Southampton

Q&A

拍拍贷 ppdai.com 触手可及的金融

扫描二维码，填写课程反馈