决策树与随机森林

主讲老师：高彦杰

课程大纲

1）决策树算法 (★★★)

2）ID3, C4.5, CART(★★★)

3）随机森林(★★★)

4）Boost(★★)

5）GBDT(★★)

6）集成学习(★★)

7) XGBoost(★★)

8）案例实战(★★)

决策树算法

决策树


选择礼物
black
颜色
white
cheap
价格
expensive
small
大小
big
喜欢
不喜欢
不喜欢
喜欢

创建决策树

输入： 数据集

输出： 构造好的决策树（也即训练集）

```
def 创建决策树:
    if(数据集中所有样本分类一致):
        创建携带类标签的叶子节点
    else:
        寻找划分数据集最好特征
        根据最好特征划分数据集
        for 每个划分的数据集:
            创建决策子树(递归方式)
```

以特征为切分点，平行于轴切分，若数据沿轴方向分布则适合回归树，计算的是标签的均值，所以做预测时无法如线性模型那样均匀输出

特点：

不适合高维稀疏矩阵

像LR模型通过特征系数来决定某一特征的贡献程度，且有正则化进行限制，而

决策树直接选择对分类有效的特征

决策树算法衍生算法

参考文档
https://zhuanlan.zhihu.com/p/103235259

- 决策树的关键在于当前状态下选择哪个属性作为分类条件。
- 最佳分类属性这种“最佳性”可以用非纯度（impurity）进行衡量。
- 如果一个数据集合中只有一种分类结果，则该集合最纯，即一致性好
- 有许多分类，则不纯，即一致性不好

优缺点：
1.由于是对标签值统计计算，所以可不需要做onehot
2.只考虑到了当前的节点的最大增益，没有进行全局优化
可能出现某个分支特别长的问题

决策树算法衍生算法

有很多指标定义不纯度，根据不同判定不纯度的目标函数：

- ✓ ID3 （信息增益）：分类
- ✓ C4.5 （信息增益比）：分类
- ✓ CART （GINI系数）：分类与回归

ID3

- ✓ **分类算法**
- ✓ **熵作为衡量样本集合纯度的标准，熵越大，越不纯。**
- ✓ **希望在分类以后能够降低熵的大小，变纯一些。**
- ✓ **分类后熵变小可以用信息增益（Information Gain）来衡量。**

ID3缺陷
1．缺失值的情况没有做考虑
2．无法处理连续值
3．以下
　　取值比较多的特征比取值少的特征信息增益大。例，
一个变量有2个值，各为1/2，另一个变量为3个值，各为1/3，
其实他们都是完全不确定的变量，但是取3个值的比取2个值的信息增益大

ID3

https://blog.csdn.net/tsyccnh/article/details/79163834

1. 计算数据集D的经验熵$H(D)$ 具体算法参见ID3算法示例.pdf

信息量：-log(px)=log(1/px)
信息熵：-\sum_1^n p(xi)log(pxi)

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

2. 计算特征A对数据集D的经验条件熵$H(D|A)$

条件熵 => 联合熵推导
= \sum_{x}\sum_yp(x,y)logp(y|x)
= H(Y|X) = H(Y, X) - H(X)

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

3. 计算信息增益

使用当前没有使用过的所有特征计算，
选择增益最大的特征

$$g(D, A) = H(D) - H(D|A)$$

C4.5

主要采用了信息增益旅/连续值处理/和缺失值处理
主要采用了信息增益旅/连续值处理/和缺失值处理
对于连续值的处理方式，C4.5采用的是将这个离散变量进行离散量化。
优缺点：大量的耗时的对数运算

- ✓ **分类算法**
- ✓ **ID3的信息增益一个缺点，一般会优先选择有较多属性值的特征**
- ✓ **解决：增加惩罚项，C4.5使用信息增益比率(gain ratio)**

$$SplitInformation\,(D,A) = -\sum_{i=1}^{N}\frac{|D_i|}{|D|}\log\frac{|D_i|}{|D|}$$

$$GainRatio\,(D,A) = \frac{g\,(D,A)}{SplitInformation\,(D,A)}$$

CART

https://blog.csdn.net/snowdroptulip/article/details/102935227

CART算法（Classification and Regression Tree）

- ✓ 可以进行分类和回归
- ✓ 分类树使用最小GINI指数来选择划分属性
- ✓ 回归树使用Mean Squared Error:

基尼系数

$$Gini(D) = 1 - \sum_{i=0}^{n} \left(\frac{D_i}{D} \right)^2$$

$$Gini(D|A) = \sum_{j=0}^{k} \frac{D_j}{D} Gini(D_j)$$

回归

c1为区域R1的均值，同理c2

$$MSE$$

$$\min \left[\frac{1}{n_{R1}} \sum_{x_i \in R_1} (y_i - c_1)^2 + \frac{1}{n_{R2}} \sum_{x_i \in R_2} (y_i - c_2)^2 \right]$$

CART分类树算法

每个特征值转化为是或否，可以将多叉树转换为二叉树结构

每次仅仅对某个特征的值进行二分，而不是多分，

这样CART分类树算法建立起来的是二叉树，而不是多叉树

剪枝算法

- **预剪枝**
  - ✓ **在构造决策树的同时进行剪枝。**主要为树深度
  - ✓ **所有决策树的构建方法，都是在无法进一步降低熵的情况下才会停止创建分支的过程，为了避免过拟合，可以设定一个阈值。**
  - ✓ **例如：熵减小的数量小于这个阈值，即使还可以继续降低熵，也停止继续创建分支。**
- **后剪枝**
  - ✓ **决策树构造完成后进行剪枝。**
  - ✓ **剪枝的过程是对拥有同样父节点的一组节点进行检查，判断如果将其合并，熵的增加量是否小于某一阈值。**
  - ✓ **如果满足阈值要求，则这一组节点可以合并一个节点，其中包含了所有可能的结果。**

后剪枝-示例

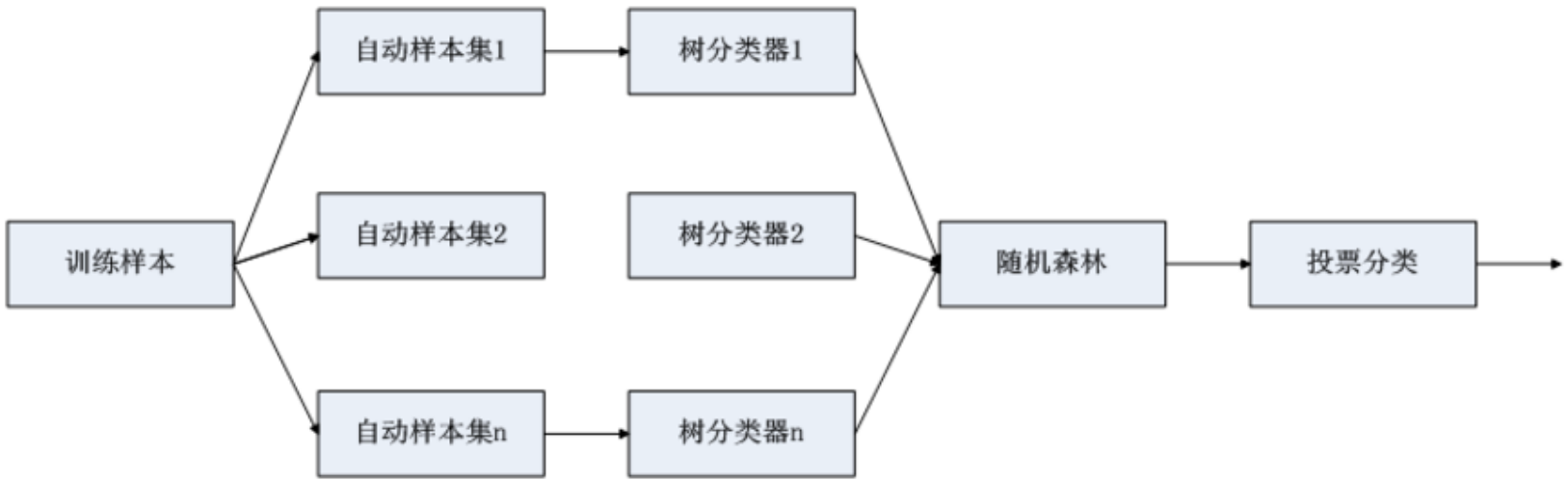
- $Reduced-Error\ Pruning\ (REP)$错误率降低剪枝
  - 思路：决策树过度拟合后，通过测试数据集来纠正。
  - 步骤： 递归清除

1. 对于完树中每一个非叶子节点的子树，尝试着把它替换成一个叶子节点
2. 该叶子节点的类别用子树覆盖训练样本中存在最多的那个类来代替，产生简化决策树
3. 然后比较这两个决策树在测试数据集中的表现
4. 简化决策树在测试数据集中的错误比较少，那么该子树就可以替换成叶子节点
5. 以bottom-up的方式遍历所有的子树，当没有任何子树可以替换提升，算法终止

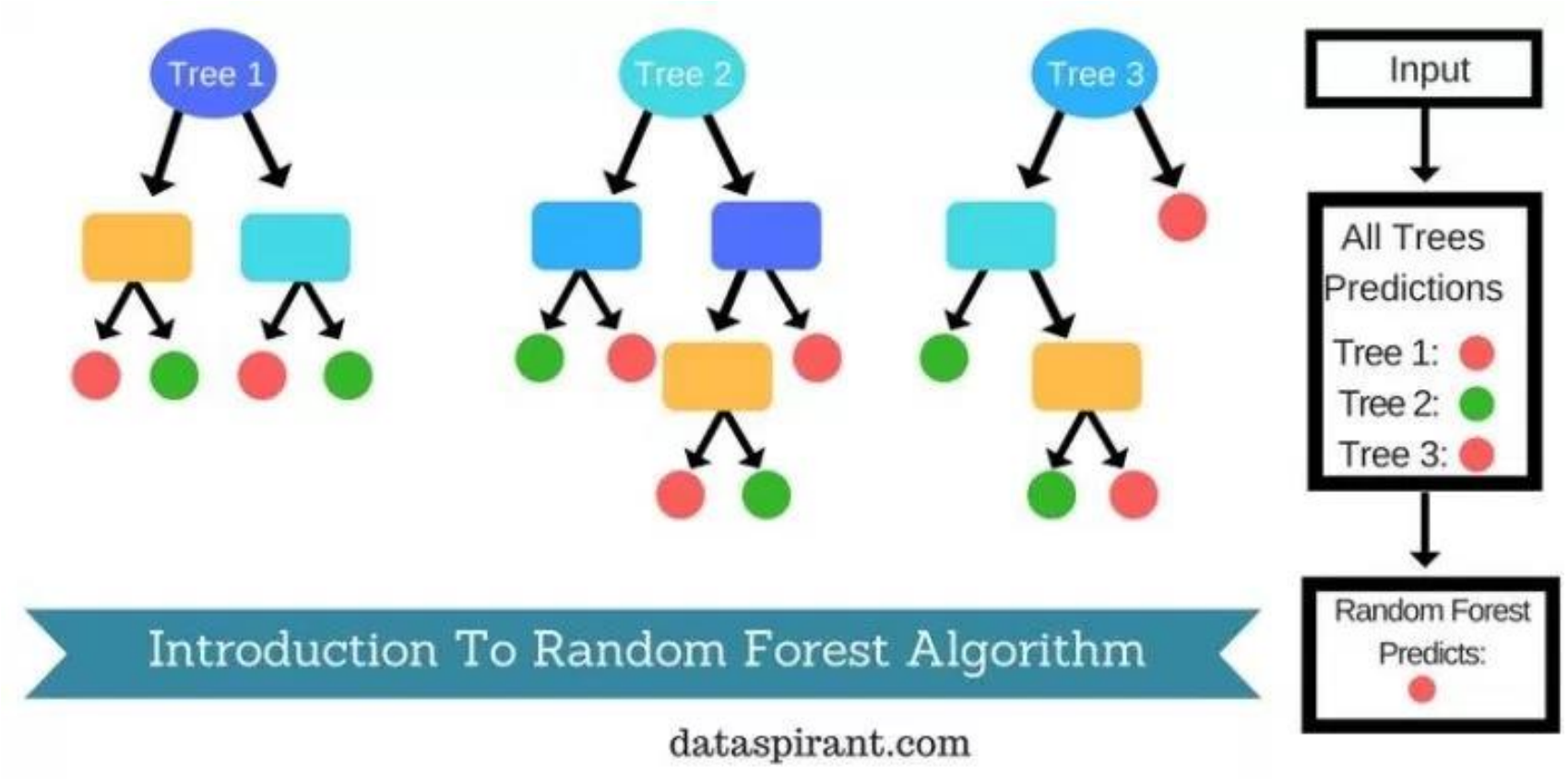
随机森林

随机森林

- 随机森林以随机的方式建立一个森林
- 森林里有很多决策树，且每棵树之间无关联
- 当有一个新样本进入后，让森林中每棵决策树分别各自独立判断，看这个样本应该属于哪一类（分类算法）
- 然后看哪一类被选择最多，就选择预测此样本为那一类



随机森林


Tree 1
Tree 2
Tree 3
Input
All Trees
Predictions
Tree 1:
Tree 2:
Tree 3:
Random Forest
Predicts:
Introduction To Random Forest Algorithm
dataspirant.com

随机森林构造过程

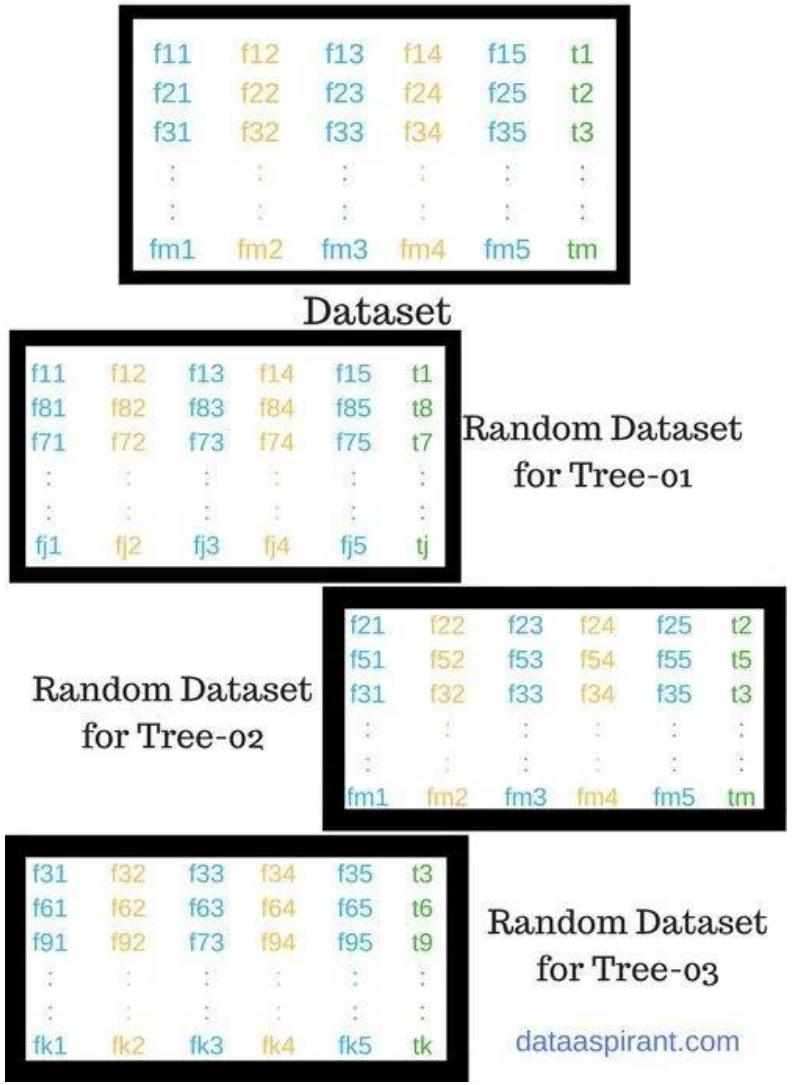
（1）原始训练集为N，应用$Bootstrap$法有放回地随机抽取K个新的自助样本集，并由此构建K棵分类树，每次未被抽到的样本组成了K个袋外数据

（2）设有m_{all}个变量，则在每一棵树的每个节点处随机抽取m_{try}个变量，然后在每个和每层层树训练过程中，m_{try}中选择一个最具有分类能力的变量，变量分类的阈值通过检查每一个分类点确定

（3）每棵树最大限度地生长，不做任何修剪

（4）将生成的多棵分类树组成随机森林，用随机森林分类器对新的数据进行判别与分类，分类结果按树分类器的投票多少而定

随机森林


f11 f12 f13 f14 f15 t1
f21 f22 f23 f24 f25 t2
f31 f32 f33 f34 f35 t3
fm1 fm2 fm3 fm4 fm5 tm
Dataset
f11 f12 f13 f14 f15 t1
f81 f82 f83 f84 f85 t8
f71 f72 f73 f74 f75 t7
fj1 fj2 fj3 fj4 fj5 tj
Random Dataset for Tree-01
Random Dataset for Tree-02
f21 f22 f23 f24 f25 t2
f51 f52 f53 f54 f55 t5
f31 f32 f33 f34 f35 t3
fm1 fm2 fm3 fm4 fm5 tm
f31 f32 f33 f34 f35 t3
f61 f62 f63 f64 f65 t6
f91 f92 f73 f94 f95 t9
fk1 fk2 fk3 fk4 fk5 tk
Random Dataset for Tree-03
dataaspirant.com

随机森林优缺点

优点：

1.适用数据集广

5．缓解决策树的过拟合问题

2.高维数据

3.Feature重要性排序 （可实现）

4.训练速度快，并行化 预测也可以并行化

缺点：

1.级别划分较多的属性影响大

当随机森林中的决策树个数很多时，训练时需要的空间和时间会比较大

在某些噪音比较大的样本集上容易陷入过拟合

Scikit-Learn RandomForest

RandomForestClassifier　　RandomForestRegressor

分类与回归模型

model = fit(x, y)

X : *array-like or sparse matrix of shape = [n_samples, n_features]*

The training input samples. Internally, its dtype will be converted to `dtype=np.float32`.

y : *array-like, shape = [n_samples] or [n_samples, n_outputs]*

The target values (class labels in classification, real numbers in regression).

Returns: **self : *object***

训练-输入与输出格式

results = predict(x)

X : *array-like or sparse matrix of shape = [n_samples, n_features]*

The input samples. Internally, its dtype will be converted to `dtype=np.float32`.

Returns: **y : *array of shape = [n_samples] or [n_samples, n_outputs]***

The predicted classes.

预测-输入与输出格式

企业案例分享

- ✓ 场景：入侵检测
- ✓ 挑战
- ✓ 问题建模：分类问题
- ✓ 算法选型：随机森林

Boost

Boost算法 拟合能力不足

- 决策树：单决策树时间复杂度较低，模型容易展示，但是容易$Over-Fitting$
  - ✓ 分类树
  - ✓ 回归树
- 决策树的$Boost$方法：迭代过程，新的训练为了改进上一次的结果
  - ✓ 传统$Boost$: 对正确、错误的样本进行加权，每一步结束后，增加分错点的权重，减少对分对点的权重 AdaBoost 加权
  - ✓ $GradientBoost$：梯度迭代，每一次建立模型是在之前建立的模型损失函数的梯度下降方向 拟合误差

Adaboost算法

- ✓ Adaboost的核心思想
- ✓ 关注被错分的样本，器重性能好的分类器
- ✓ 如何实现
  - ✓ 不同的训练集 -> 调整样本权重
  - ✓ 关注 -> 增加错分样本权重
  - ✓ 器重 -> 好的分类器权重大
  - ✓ 样本权重间接影响分类器权重

Adaboost算法步骤

For t=1, ……, T

1. Train learner h_t with min error $\varepsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$

若划分正确，则不计入误差，若所有元素都被正确划分，则误差为0

若划分错误，则计入误差

2. If $\varepsilon_t \geq 0.5$, then stop

3. Compute the hypothesis weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

The weight **Ada**pts. The bigger ε_t becomes the smaller α_t becomes.

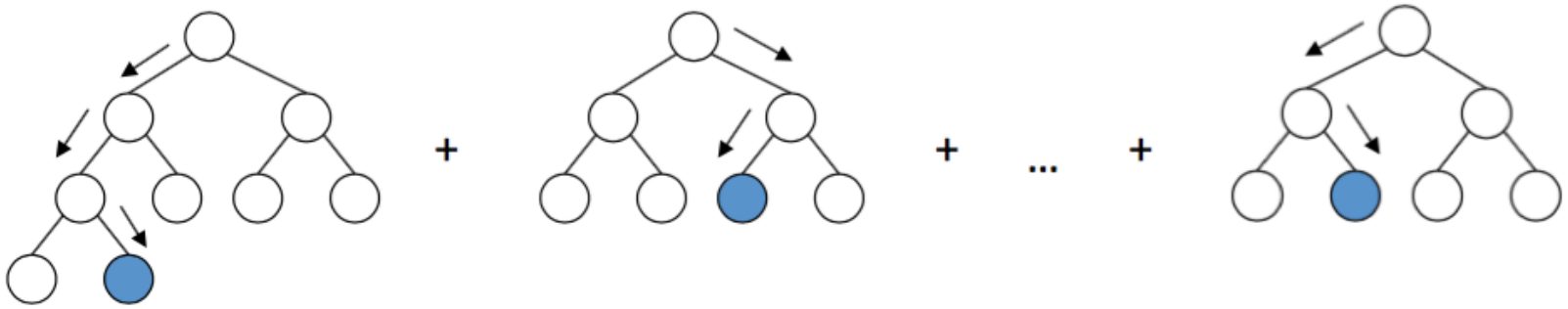
4. $$D_{t+1}(i) = D_t(i)\exp\left(\alpha_t * 1_{(h_t(i)=y_i)}\right) = \begin{cases} D_t(i), & 若 y_i = h_t(x_i) \\ D_t(i)\frac{1-\varepsilon_t}{\varepsilon_t}, & 若 y_i \neq h_t(x_i) \end{cases}$$ ？？？？

5.最后得到的强分类器: $H(x) = \text{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$

GBDT

GBDT(Gradient Boosting Decision Tree)算法

- ✓ 用一个初始值来学习一棵决策树，叶子处可以得到预测的值，以及预测之后的残差，然后后面的决策树就要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。
- ✓ 最后对于测试样本的预测值，就是前面许多棵决策树预测值的累加。



GBDT算法应用场景

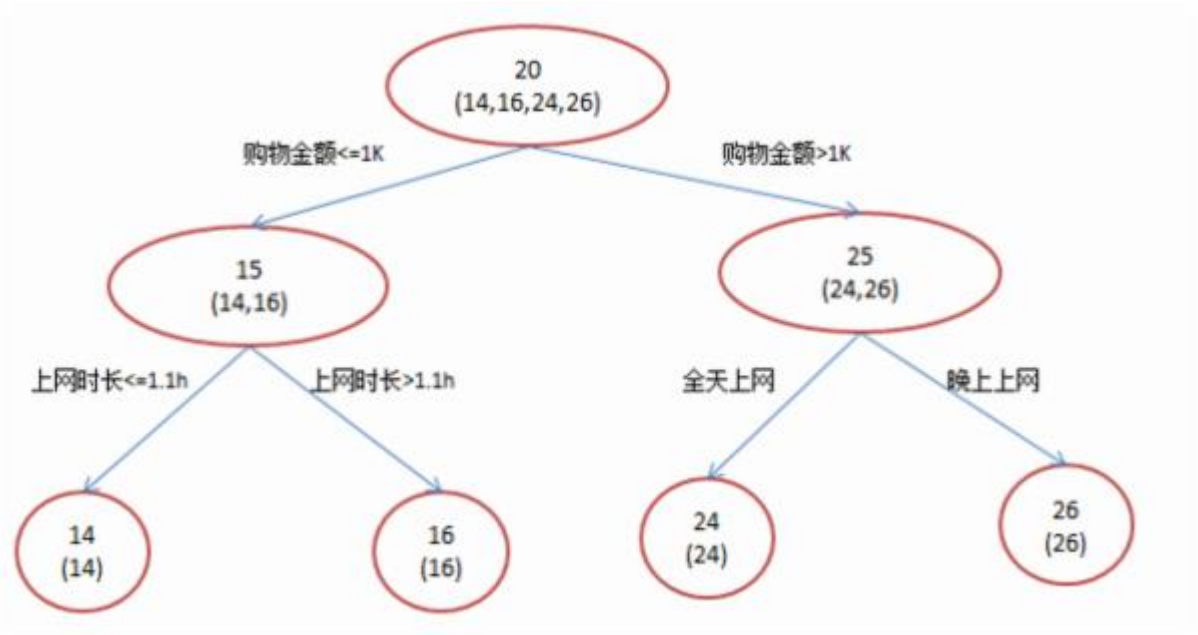
基本用于回归

- ✓ 适用问题广，可扩展性好
- ✓ GBDT几乎可以用于所有回归问题（线性、非线性）
- ✓ 分类问题
- ✓ 排序问题
- ✓ 常用于各大数据挖掘竞赛
- ✓ 广告推荐

GBDT实例

$$Training\ Set: (A, age = 14), (B, 16), (C, 24), (D, 26)$$

下面为决策树示例



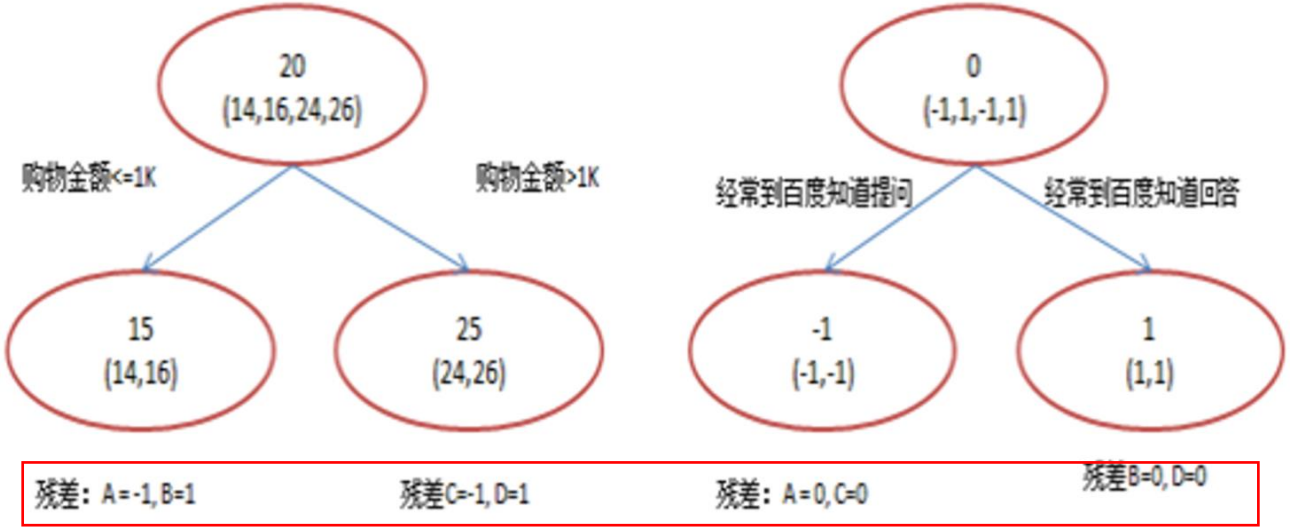
GBDT实例

A: 14岁高一学生，购物较少，经常问学长问题；预测年龄 $A = 15 - 1 = 14$

B: 16岁高三学生；购物较少，经常被学弟问问题；预测年龄 $B = 15 + 1 = 16$

C: 24岁应届毕业生；购物较多，经常问师兄问题；预测年龄 $C = 25 - 1 = 24$

D: 26岁工作两年员工；购物较多，经常被师弟问问题；预测年龄$D = 25 + 1 = 26$



GBDT算法

Algorithm 1: Gradient_Boost

1 $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$

2 For $m = 1$ to M do:

3 $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$

4 $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$

5 $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$

6 $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

7 endFor

end Algorithm

Random Forest VS GBDT

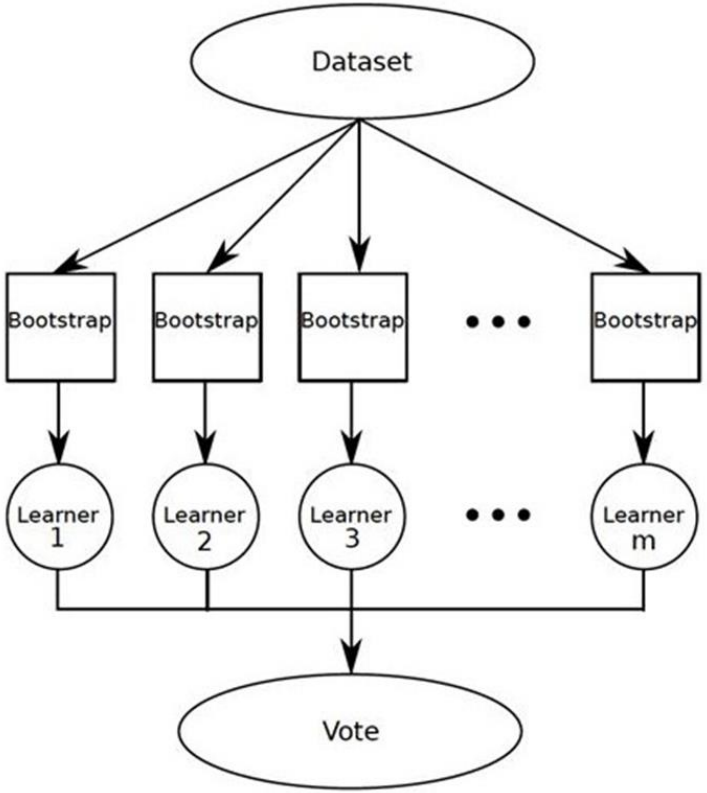
- 准确度与拟合： $GBDT > RF$ better accuracy with less trees. RF are harder to overfit than GBDT
- 建模能力： $GBDT > RF$ because boosted trees are derived by optimizing a objective function, basically it can be used to solve almost all objective you can write gradient out.
- 并行化：Random Forests can be easily deployed in a distributed fashion due to the fact that they can run in parallel, whereas Gradient Boosted Machines only run trial after trial.

集成学习

集成学习 Bagging

有放回抽样算法
代表，
随机森林

让该学习算法训练多轮，每轮的训练集由从初始的训练集中随机取出的n个训练样本组成，某个初始训练样本在某轮训练集中可以出现多次或根本不出现，训练之后可得到一个预测函数序列$h_1 \ \dots \ h_n$，最终的预测函数H对分类问题采用投票方式，对回归问题采用简单平均方法对新示例进行判别。



集成学习Boosting

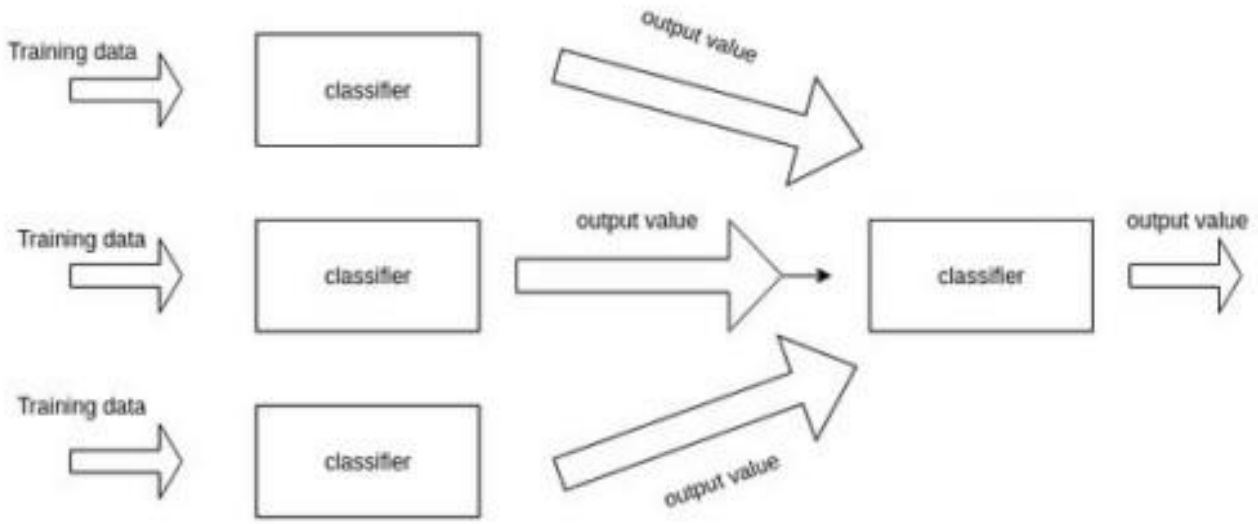
Boost目

- 初始化时对每一个训练例赋相等的权重$\frac{1}{n}$
- 然后用该学算法对训练集训练t轮
- 每次训练后，对训练失败的训练例赋以较大的权重，也就是让学习算法在后续的学习中集中对比较难的训练例进行学习，从而得到一个预测函数序列$h_1 \ \dots \ h_n$，其中h_i也有一定的权重，预测效果好的预测函数权重较大，反之较小。
- 最终的预测函数H对分类问题采用有权重的投票方式，对回归问题采用加权平均的方法对新示例进行判别。

集成学习 Stacking

将训练好的所有基模型对训练基进行预测，第j个基模型对第i个训练样本的预测值将作为新的训练集中第i个样本的第j个特征值，最后基于新的训练集进行训练。

同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测



XGBoost

XGBoost

- XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient, flexible and portable**.
- It implements machine learning algorithms under the Gradient Boosting framework.
- XGBoost provides a **parallel tree boosting** (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.
- The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

官网：https://github.com/dmlc/xgboost

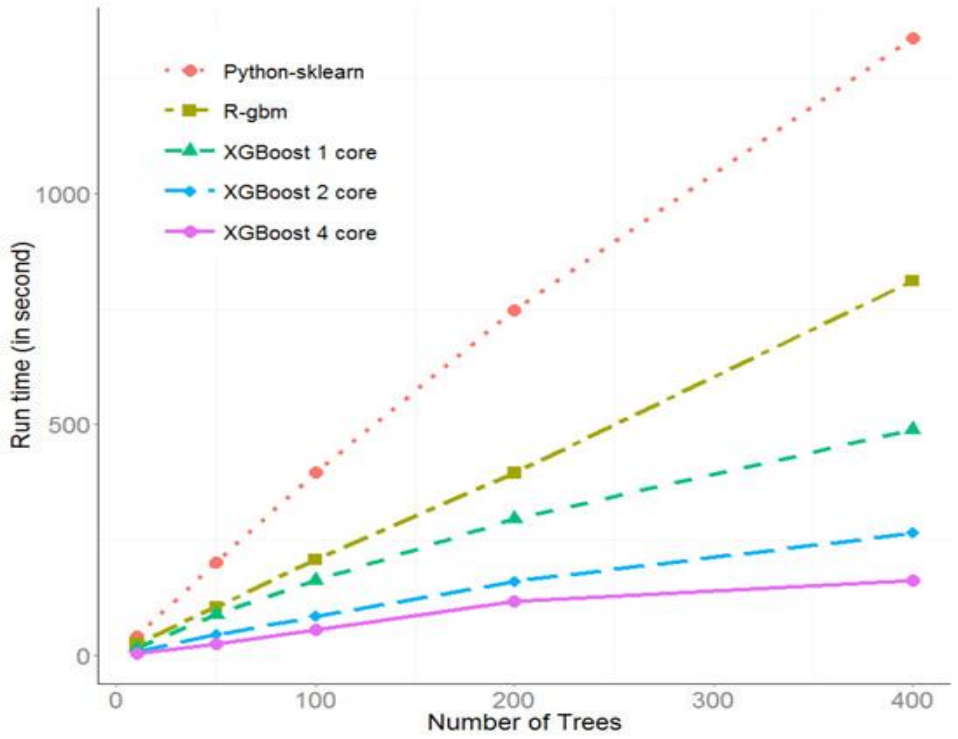
XGBoost

- ✓ Boosting分类器将成百上千个分类准确率较低的树模型组合起来，成为一个准确率很高的模型。
- ✓ 数据集较大较复杂的时候，我们可能需要几千次迭代运算，这将造成巨大的计算瓶颈。
- ✓ XGBoost正是为了解决这个瓶颈而提出。单机它采用多线程来加速树的构建，并可以进行分布式计算。
- ✓ XGBoost提供了 Python和R语言接口。

XGBoost

Kaggle的某竞赛数据上

- ✓ **单线程XGBoost比其他两个包均要快出50%**
- ✓ **在多线程上XGBoost更是有接近线性的性能提升**



案例实战

预习/复习内容重点与难点

- ✓ 重点：
- ✓ 1. 决策树
- ✓ 2. 随机森林
- ✓ 3. $Ensemble$方法
- ✓ 难点：

1. 树构建过程切分条件
2. $Boost$

参考资料

- Induction of Decision Trees
- An introduction to random forests
- Ensemble Learning
- 统计学习方法