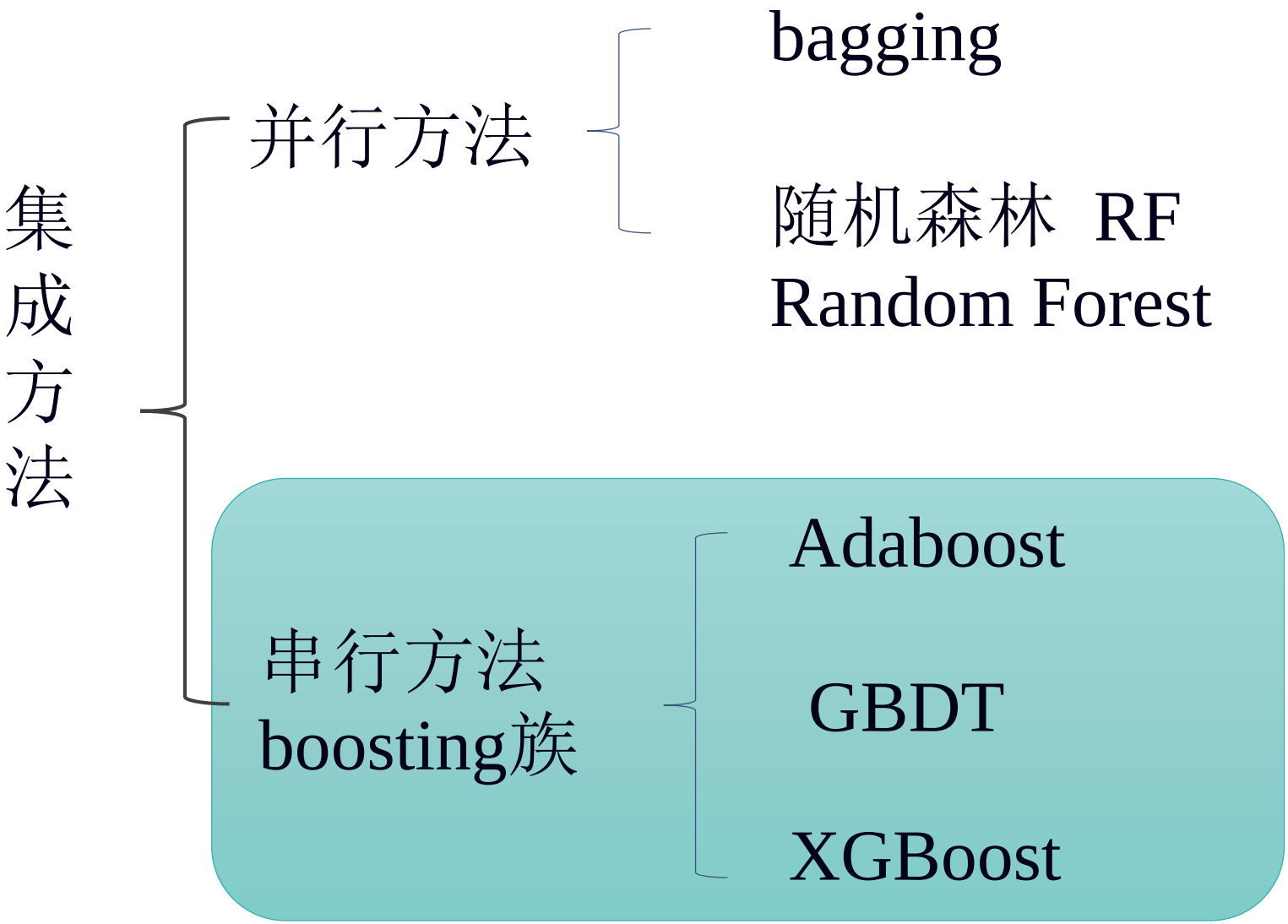
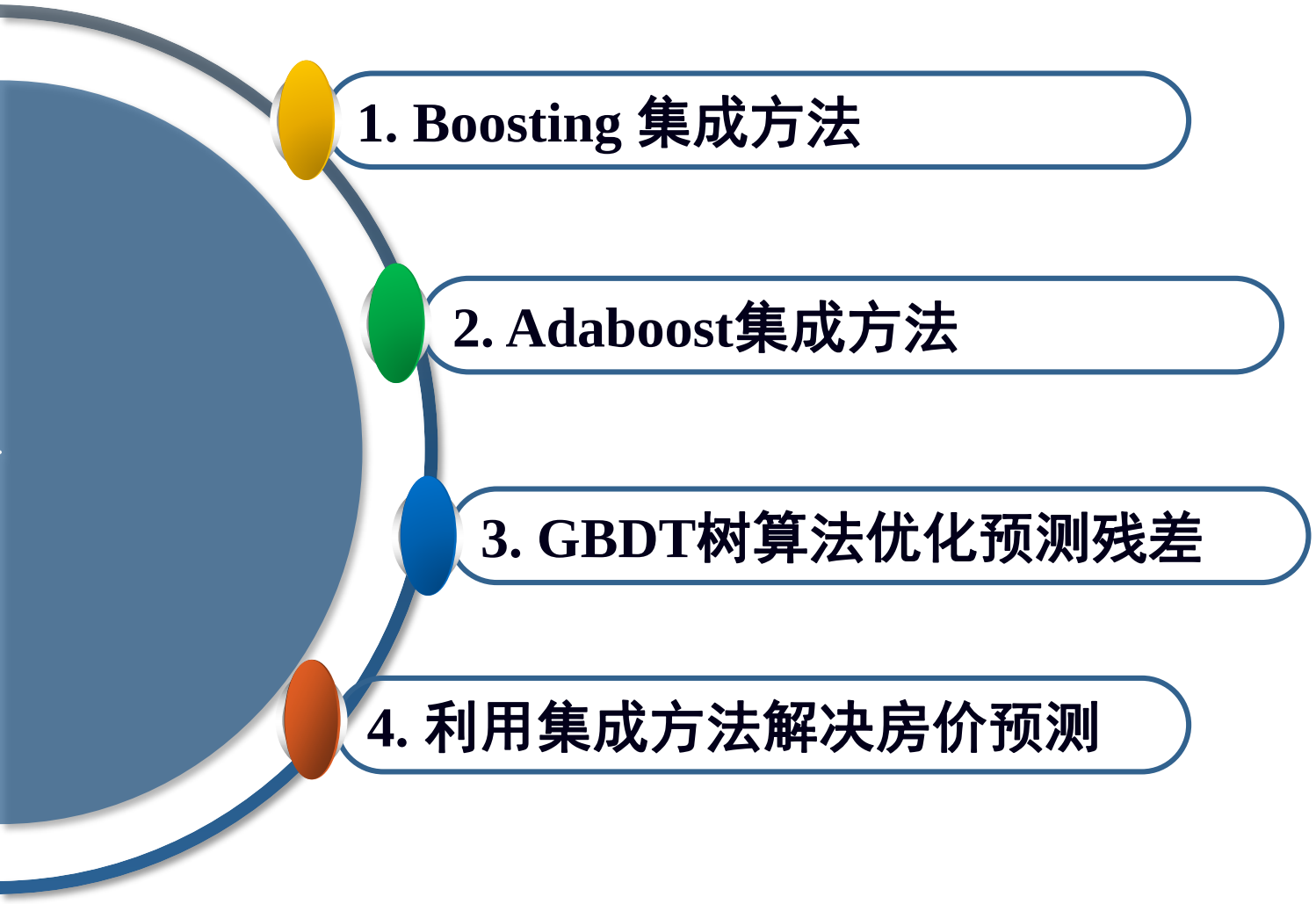
《集成算法之boosting》



本章授课内容


1. Boosting 集成方法
2. Adaboost集成方法
3. GBDT树算法优化预测残差
4. 利用集成方法解决房价预测

1 Boosting集成方法

boosting方法产生于计算学习理论(Computational Learning Theory)

Boosting是一族方法，该族方法具有一个类似的框架

1. 根据当前的数据训练出一个弱模型
2. 根据该弱模型的表现调整数据样本的权重，具体而言：
    1. 让该样本做错的样本在后续的训练中获得更多的关注
    2. 让该样本做对的样本在后续的训练中获得较少的关注
3. 最后再根据弱模型的表现决定该弱模型的“话语权”，亦即投票表决时的“可信度”。自然，表现越好的就越具有话语权。

2 Adaboost算法

由boosting方法的陈述可知，问题的关键在于两点：

1. 如何根据弱模型的表现更新训练集的权重

   对每个样本的作用

2. 如何根据弱模型的表现决定弱模型的话语权

   整体价值体现

Adaboost算法:

采取了加权错误率的方法更新样本的权重

用来解决二分类问题，标签是{-1,1}

弱分类器选择决策树桩，

决策树桩是单层二叉树，以加权错误率作为分割标准

2 Adaboost算法

假设现有的二分类训练数据集：

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$$

其中每个样本由特征x和类别y组成，且：

$$x_i \in X \subseteq \square^n; y_i \in Y=\{-1,+1\}$$

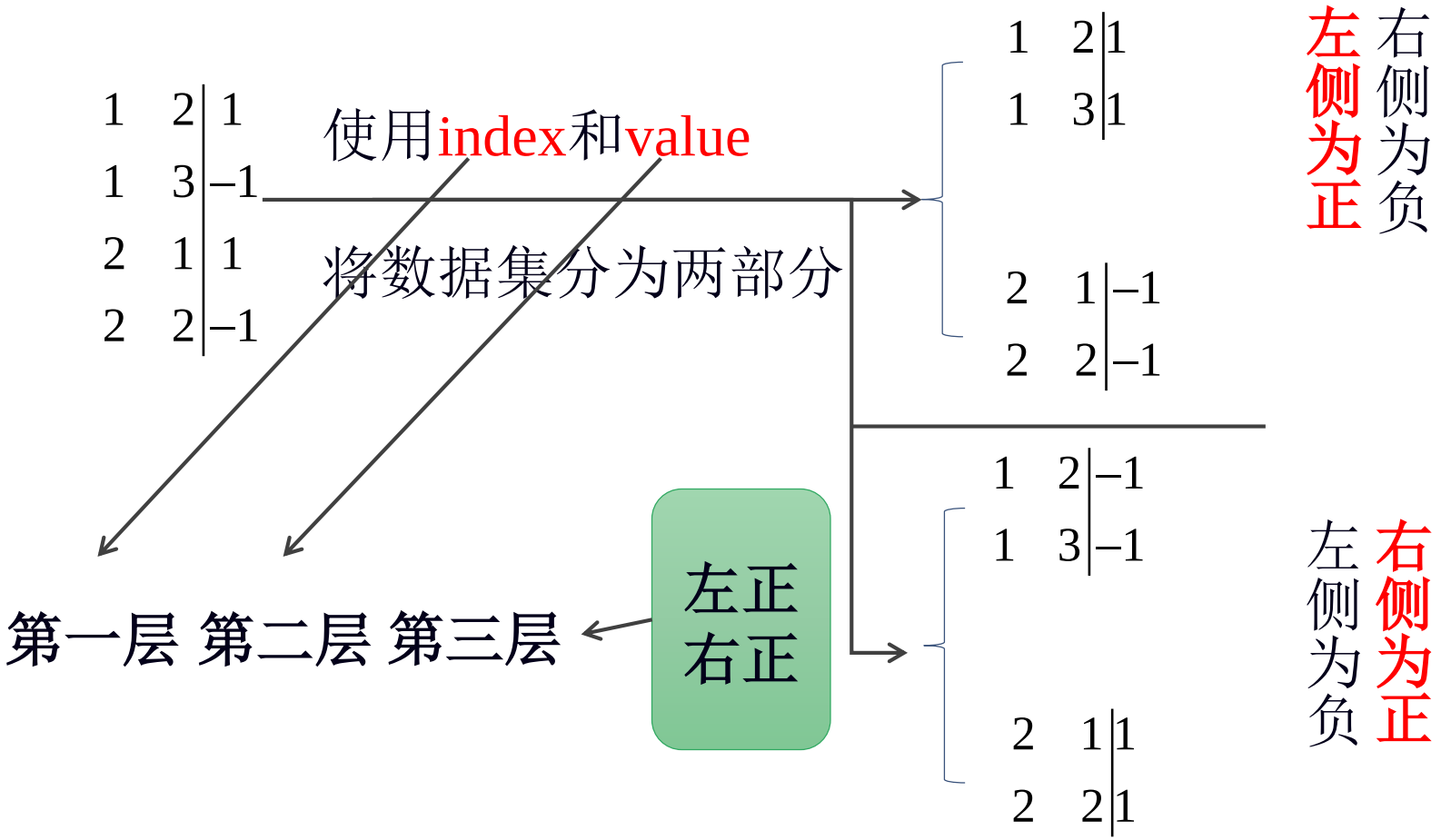
Adaboost算法步骤如下： Adaboost算法陈述.docx

第一个关键点：如何构建一个单层决策树作为弱分类器

以什么作为分类标准： **加权错误率**
二叉树还是多叉树： 二分类，二叉树
什么时候终止： 一层就停止

2.1 单层决策树

adaboost.py：
treeStump函数实现单层决策树



2.1 单层决策树

使用两个函数实现单层决策树：split函数实现对data的分割
tree_stump 实现决策树桩的建立

先来实现最内层：数据集进行二分
split(data , index , value , lr)#lr=‘l’或者’r’，data不包括标签列
要求data是mat，输出的预测pre是列向量

然后实现主循环：对data的特征索引，二分标准，左右进行循环选择出具有最小加权错误率的分配情况
tree_stump(data ,classLabel , w)　#w是样本的权重
输出：min_err:最小加权错误率
　　b_pre:最优预测结果
　　b_stump:{‘index’:　, ‘value’:　, ’lr’:　　}

2.1 单层决策树建树

split(data , index , value , lr)　#lr='l'或者'r'，要求data是mat
pre 的长度和data列的长度是一样的
首先创建pre为全一的列向量
如果lr为左: (分割值左侧为正样本)
　　pre[data[:,index]>value]=-1
如果lr为右：(分割值的右侧为正样本)
　　pre[data[:,index]≤value]=-1
返回pre

pre[data[:,index]>value]=-1
布尔值索引

2.1 单层决策树建树

tree_stump(data ,labelList, w) #
定义最小加权错误率 min_w_err
最优预测结果b_pre=np.mat(zeros((m,1)))
树桩stump={‘index’:,’value’:,’lr’:}
对data的每一列：
    获得该列的二分标准values
    对values中的每一个值:
        对lr属于[‘l’,’r’]:
            预测结果pre=split(data,index,value,lr)
            错误列向量err，shape=(m,1)
            (对应位置预测对为0，错误为1)
            加权错误率w_err=pre.T*w
            如果w_err<min_w_err:
                重新设定min_w_err,b_pre,stump
返回 min_w_err,b_pre,stump

2.2 树桩“话语权”与更新权重

计算树桩的“话语权”
e对应的是min_w_err
求得alpha

alpha=0.5*log((1-min_w_err)/(min_w_err+0.000001))

根据树桩的“话语权”alpha和
对每个样本的预测标签b_pre与真实标签labelList更新权重

先求Z:
连加符号可以看做是两个向量的内积：a.T*b a,b是列向量

$$w \quad \text{和} \quad \begin{matrix} \exp(-\alpha[y_1 g_{k+1}(x_1)]), \\ \exp(-\alpha[y_2 g_{k+1}(x_2)]), \\ \vdots \\ \exp(-\alpha[y_m g_{k+1}(x_m)]) \end{matrix} = \exp(-\alpha(y \otimes g_{k+1}(x)))$$

2.2 更新权重

向量解决问题：
连加：　　内积
对应项之间赋值：对应项相乘(np.multiply)

再更新w:
由两部分组成w/z 和exp内： 分别是两个向量
exp内也是由两部分组成y 真实标签和g 预测标签：两个向量

```
w_next = np.multiply ( w/(z+0.000001) ,
                    np.exp(-alpha*np.multiply(labelList ,b_pre)))
```

c=np.multiply(a,b)
a, b 都是行向量: c 是行向量，对应元素相乘
a, b 都是列向量: c是列向量，对应元素相乘
a, b其中一个是行向量，一个是列向量: c是矩阵，向量乘积

2.3 Adaboost集成

adaboost(data,labelList,n_stump)
创建adaboost集成器，其中含有n_stumps个决策树桩
需要返回的是每个决策树桩，及其“话语权”

boost_stumps={1:(alpha,stump),…,}

初始化w,长度为样本数m，权重为1/m
初始化boost_stumps={}
建立n_stumps个决策树桩：
加权误差w_err，预测结果pre，树桩stump通过调用tree_stump(data,labelList,w)得到
计算alpha
计算z
更新w
将alpha,stumps存入boost_stumps

对alpha*pre连加可以监测集成器目前的效果

返回 boost_stumps

2.4 Adaboost分类预测

predict(boost_stumps,sample)
boost_stumps是分类器生成的权重与决策树桩
sample是一个测试样本
初始化预测结果pre_bynow=0
对于boost_stumps中的每一个树桩：
　　pre_bynow+=alpha*树桩的预测结果
如果pre_bynow>0:返回1
如果pre_bynow<0:返回-1

树桩的预测结果：树桩{index，value, lr}，样本：sample
　　如果sample[index]<value:
　　　　如果lr='1'：bre=1 ； 否则：pre=-1
　　否则：
　　　　如果lr='1'：pre=-1 ； 否则：pre=1

2.5 例题

水雷-岩石数据