法律声明

- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：*http://www.ibeifeng.com/*

人工智能之机器学习

决策树、随机森林和提升算法

主讲人：Gerry

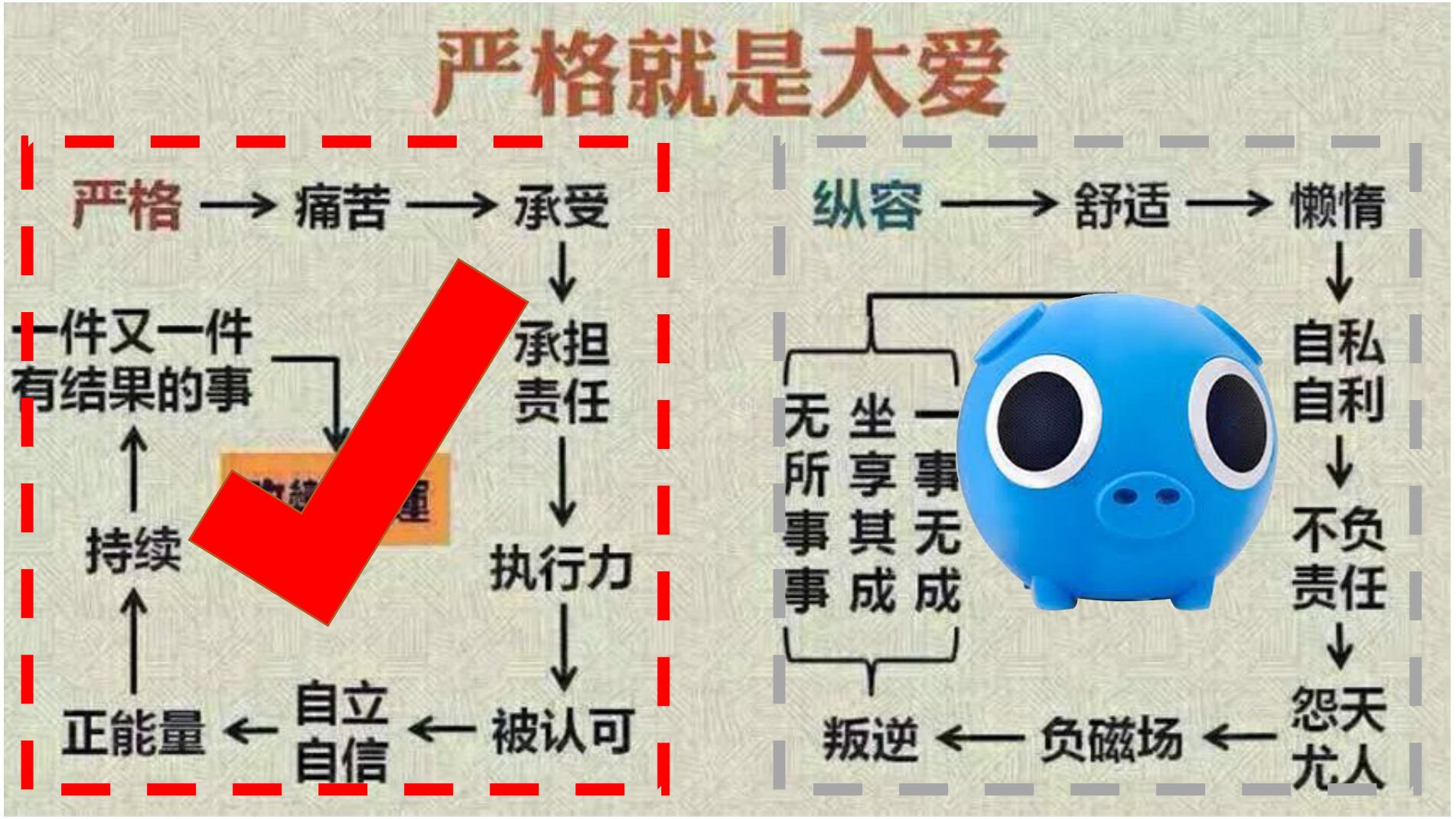
上海育创网络科技有限公司

北风网 IBEIFENG.COM

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

严格是大爱


严格就是大爱
严格
痛苦
承受
承担责任
执行力
被认可
自立自信
正能量
持续
一件又一件有结果的事
纵容
舒适
懒惰
自私自利
不负责任
怨天尤人
负磁场
叛逆
一事无成
坐享其成
无所事事

寄语

做别人不愿做的事，

做别人不敢做的事，

做别人做不到的事。

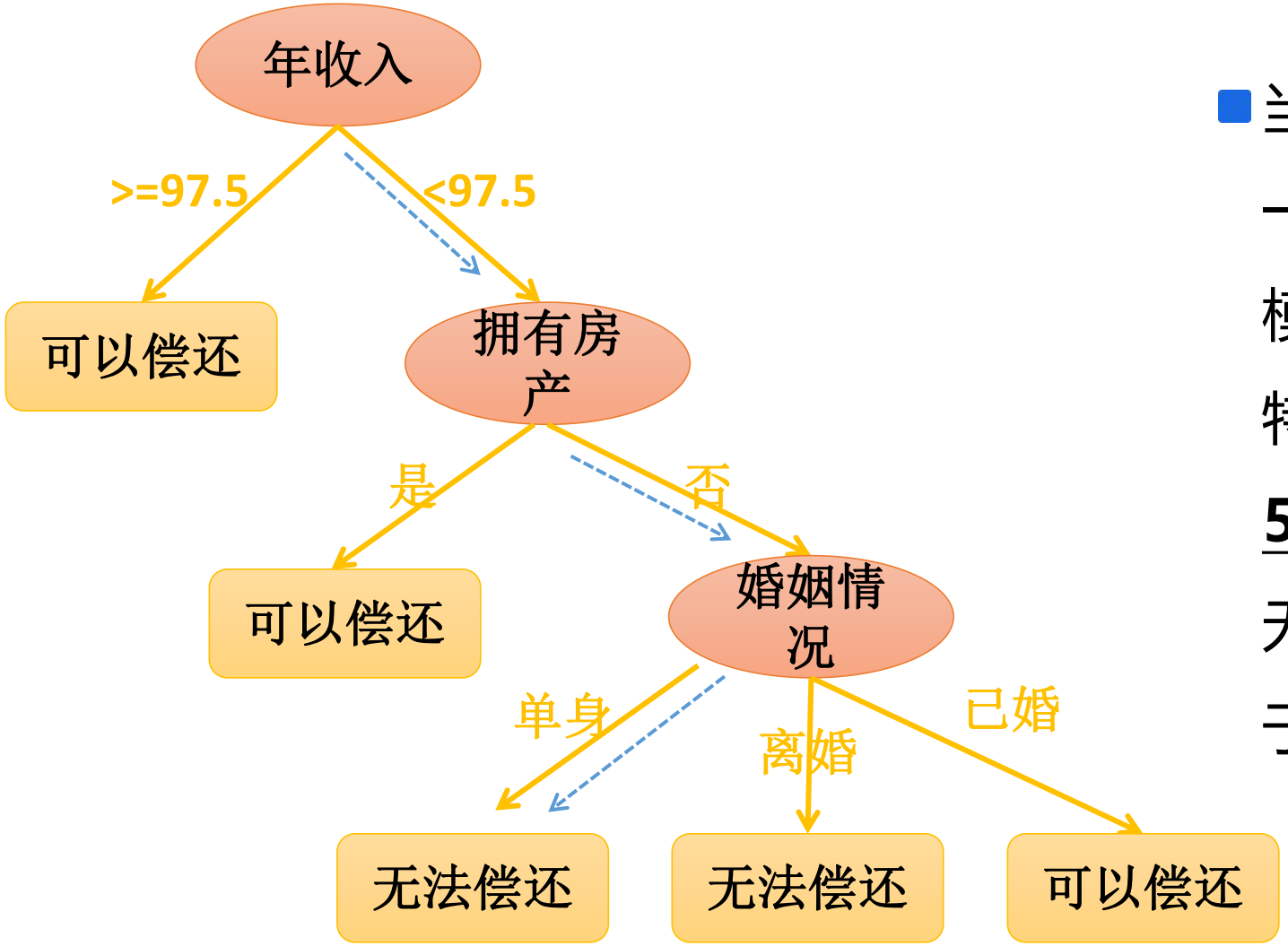
课程内容

- 信息熵
- 决策树
- 决策树优化
- 剪枝
- 随机森林
- 提升算法
- GBDT(迭代决策树)
- Adaboost

决策树直观理解

ID	拥有房产(是/否)	婚姻状态(单身\已婚\离婚)	年收入(单位:千元)	无法偿还债务(是/否)
1	是	单身	125	否
2	否	已婚	100	否
3	否	单身	100	否
4	是	已婚	110	否
5	是	离婚	60	否
6	否	离婚	95	是
7	否	单身	85	是
8	否	已婚	75	否
9	否	单身	90	是
10	是	离婚	220	否

决策树直观理解



- 当构建好一个判断模型后，新来一个用户后，可以根据构建好的模型直接进行判断，比如新用户特性为：**无房产**、**单身**、**年收入55K**，那么根据判断得出该用户无法进行债务偿还。这种决策对于借贷业务有比较好的指导意义。

比特化(Bits)

- 假设存在一组随机变量X，各个值出现的概率关系如图；
- 现在有一组由X变量组成的序列: BACADDCBAC.....；如果现在希望将这个序列转换为二进制来进行网络传输，那么我们得到一个得到一个这样的序列: 01001000111110010010.....
- 结论: 在这种情况下，我们可以使用两个比特位来表示一个随机变量。

P(X=A)=1/4	P(X=B)=1/4	P(X=C)=1/4	P(X=D)=1/4

A	B	C	D
00	01	10	11

比特化(Bits)

- 而当X变量出现的概率值不一样的时候，对于一组序列信息来讲，每个变量平均需要多少个比特位来描述呢??

P(X=A)=1/2	P(X=B)=1/4	P(X=C)=1/8	P(X=D)=1/8

A	B	C	D
0	10	110	111

$$E = 1*\frac{1}{2} + 2*\frac{1}{4} + 3*\frac{1}{8} + 3*\frac{1}{8} = 1.75$$

$$E = -\log_2\left(\frac{1}{2}\right)*\frac{1}{2} - \log_2\left(\frac{1}{4}\right)*\frac{1}{4} - \log_2\left(\frac{1}{8}\right)*\frac{1}{8} - \log_2\left(\frac{1}{8}\right)*\frac{1}{8} = 1.75$$

一般化的比特化(Bits)

- 假设现在随机变量X具有m个值，分别为: $V_1,V_2,....,V_m$；并且各个值出现的概率如下表所示；那么对于一组序列信息来讲，每个变量平均需要多少个比特位来描述呢??

$P(X=V_1)=p_1$	$P(X=V_2)=p_2$	$P(X=V_3)=p_3$	.....	$P(X=V_m)=p_m$

- 可以使用这些变量的期望来表示每个变量需要多少个比特位来描述信息:

$$E(X) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) - ... - p_m \log_2(p_m)$$

$$= -\sum_{i=1}^{m} p_i \log_2(p_i)$$

信息熵(Entropy)

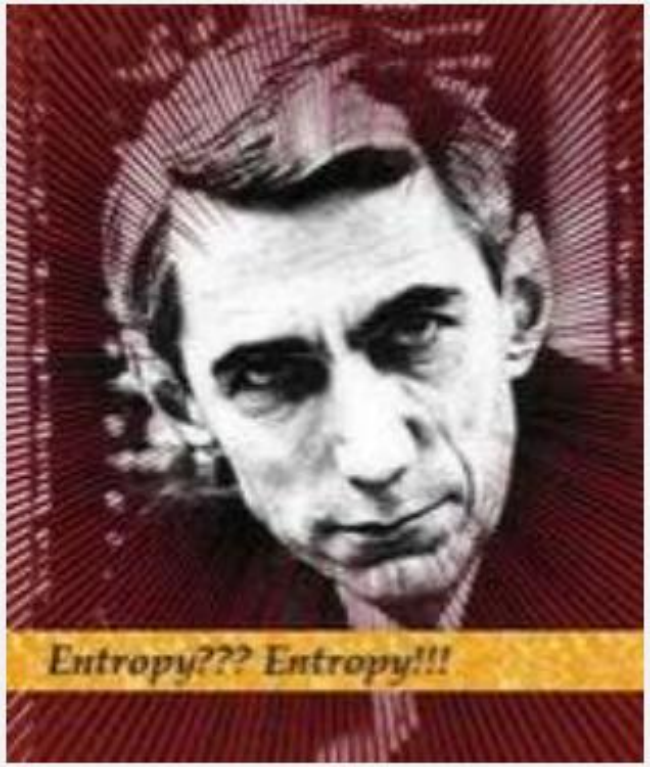
- H(X)就叫做随机变量X的信息熵；

$$H(X) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

信息熵(Entropy)

$$H(X) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

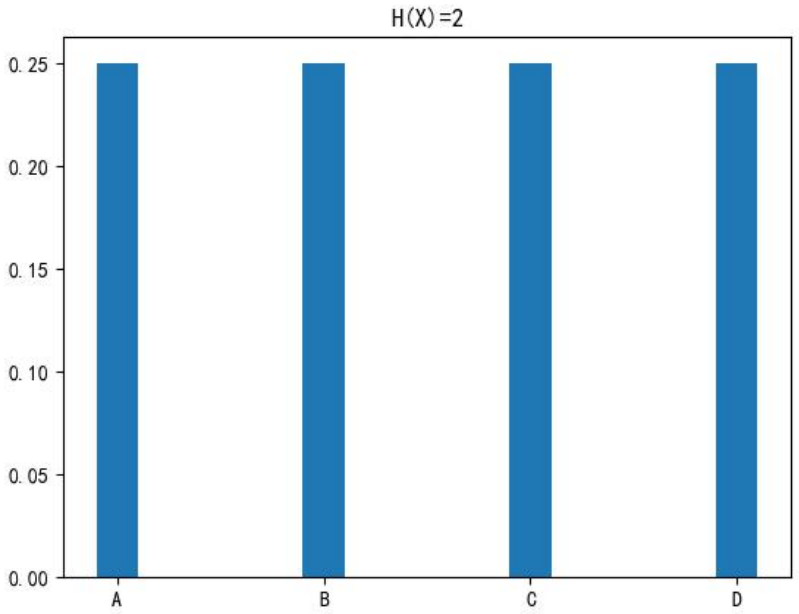
- 信息量：指的是一个样本/事件所蕴含的信息，如果一个事件的概率越大，那么就可以认为该事件所蕴含的信息越少。极端情况下，比如：“太阳从东方升起”，因为是确定事件，所以不携带任何信息量。
- 信息熵：1948年，香农引入信息熵；一个系统越是有序，信息熵就越低，一个系统越是混乱，信息熵就越高，所以信息熵被认为是一个系统有序程度的度量。
- **信息熵就是用来描述系统信息量的不确定度。**

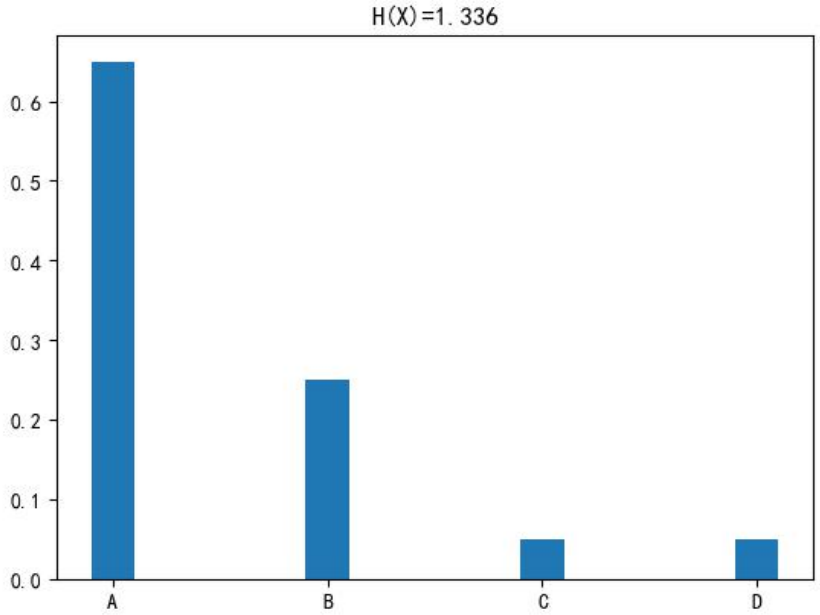


信息熵(Entropy)

$$H(X) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- High Entropy(高信息熵)：表示随机变量X是均匀分布的，各种取值情况是等概率出现的。
- Low Entropy(低信息熵)：表示随机变量X各种取值不是等概率出现。可能出现有的事件概率很大，有的事件概率很小。





信息熵(Entropy)案例 $H(X)=-\sum_{i=1}^{m} p_i \log_2(p_i)$

- 赌马比赛中，有两组赛马共八匹，获胜的概率如下：

	第一组	第二组
P(X=A)	1/4	13/20
P(X=B)	1/4	5/20
P(X=C)	1/4	1/20
P(X=D)	1/4	1/20

- 在比赛前，对于第一组而言，我们只知道A/B/C/D获胜的概率是一样的，我们是判断不出来任何偏向的；但是对于第二组而言，我们很清楚的就能够判断A会获胜。

条件熵H(Y|X)

- 给定条件X的情况下，随机变量Y的信息熵就叫做条件熵。

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

$$P(X=\text{数学})=0.5$$
$$P(Y=M)=0.5$$
$$P(X=\text{数学},Y=F)=0.25$$
$$P(Y=M\mid X=\text{英语})=0$$

$$H(X)=1.5$$
$$H(Y)=1$$

条件熵H(Y|X)

- 当专业(X)为数学的时候，Y的信息熵的值为:H(Y|X=数学)

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

专业(X)	性别(Y)
数学	M
数学	F
数学	M
数学	F

$$H(Y \mid X = 数学) = 1$$

条件熵H(Y|X)

- 给定条件X的情况下，所有不同x值情况下Y的信息熵的平均值叫做条件熵。

专业(X)	性别(Y)
数学	M
IT	M
英语	F
数学	F
数学	M
IT	M
英语	F
数学	F

$$H(Y|X)=\sum_{j=1} P(X=v_j)H(Y|X=v_j)$$

v_j	$P(X=v_j)$	$H(Y\|X=v_j)$
数学	0.5	1
IT	0.25	0
英语	0.25	0

$$H(Y|X)=0.5*1+0.25*0+0.25*0=0.5$$

条件熵H(Y|X)

- 给定条件X的情况下，所有不同x值情况下Y的信息熵的平均值叫做条件熵。另外一个公式如下所示：

$$H(Y \mid X) = H(X,Y) - H(X)$$

- 事件(X,Y)发生所包含的熵，减去事件X单独发生的熵，即为在事件X发生的前提下，Y发生“新”带来的熵，这个也就是条件熵本身的概念。

条件熵H(Y|X)

$$H(Y \mid X)=\sum_{j=1} P(\mathrm{X}=\mathrm{v}_{\mathrm{j}})H(Y \mid X=v_j)=\sum_{x} P(\mathrm{x})H(Y \mid x)$$

$$=\sum_{x} p(x)\left(-\sum_{y} p(y \mid x)\log(p(y \mid x))\right)=-\sum_{x}\sum_{y} p(x)p(y \mid x)\log(p(y \mid x))$$

$$=-\sum_{x}\sum_{y} p(x,y)\log\left(\frac{p(x,y)}{p(x)}\right)$$

$$=-\sum_{x}\sum_{y} p(x,y)\log(p(x,y))-\left[-\sum_{x}\left(\sum_{y} p(x,y)\right)\log(p(x))\right]$$

$$=H(X,Y)-\left[-\sum_{x} p(x)\log(p(x))\right]=H(X,Y)-H(X)$$

什么是决策树

- 决策树(Decision Tree)是在已知各种情况发生概率的基础上，通过构建决策树来进行分析的一种方式，是一种直观应用概率分析的一种图解法；决策树是一种预测模型，代表的是对象属性与对象值之间的映射关系；决策树是一种树形结构，其中每个内部节点表示一个属性的测试，每个分支表示一个测试输出，每个叶节点代表一种类别；决策树是一种非常常用的有监督的分类算法。
- 决策树的决策过程就是从根节点开始，测试待分类项中对应的特征属性，并按照其值选择输出分支，直到叶子节点，将叶子节点的存放的类别作为决策结果。
- 决策树分为两大类：分类树和回归树，前者用于分类标签值，后者用于预测连续值，常用算法有ID3、C4.5、CART等

决策树构建过程

- 决策树算法的重点就是决策树的构造；决策树的构造就是进行属性选择度量，确定各个特征属性之间的拓扑结构(树结构)；构建决策树的关键步骤就是分裂属性，分裂属性是指在某个节点按照某一类特征属性的不同划分构建不同的分支，其目标就是让各个分裂子集尽可能的'纯'(让一个分裂子类中待分类的项尽可能的属于同一个类别)。
- 构建步骤如下:
  - 1. 将所有的特征看成一个一个的节点；
  - 2. 遍历每个特征的每一种分割方式，找到最好的分割点；将数据划分为不同的子节点，eg： N_1、N_2....N_m；计算划分之后所有子节点的'纯度'信息；
  - 3. 对第二步产生的分割，选择出最优的特征以及最优的划分方式；得出最终的子节点: N_1、N_2....N_m
  - 4. 对子节点N_1、N_2....N_m分别继续执行2-3步，直到每个最终的子节点都足够'纯'。

决策树特征属性类型

- 根据特征属性的类型不同，在构建决策树的时候，采用不同的方式，具体如下：
  - 属性是离散值，而且不要求生成的是二叉决策树，此时一个属性就是一个分支
  - 属性是离散值，而且要求生成的是二叉决策树，此时使用属性划分的子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支
  - 属性是连续值，可以确定一个值作为分裂点split_point，按照>split_point和<=split_point生成两个分支

决策树分割属性选择

- 决策树算法是一种“贪心”算法策略，只考虑在当前数据特征情况下的最好分割方式，不能进行回溯操作。
- 对于整体的数据集而言，按照所有的特征属性进行划分操作，对所有划分操作的结果集的“纯度”进行比较，选择“纯度”越高的特征属性作为当前需要分割的数据集进行分割操作，持续迭代，直到得到最终结果。决策树是通过“纯度”来选择分割特征属性点的。

决策树量化纯度

- 决策树的构建是基于样本概率和纯度进行构建操作的，那么进行判断数据集是否“纯”可以通过三个公式进行判断，分别是Gini系数、熵(Entropy)、错误率，这三个公式值越大，表示数据越“不纯”；越小表示越“纯”；实践证明这三种公式效果差不多，一般情况使用熵公式

$$P(1)=7/10=0.7;\text{可以偿还概率}$$
$$P(2)=3/10=0.3;\text{无法偿还概率}$$

$$Gini=1-\sum_{i=1}^{n}P(i)^2 \qquad H(Entropy)=-\sum_{i=1}^{n}P(i)\log_2(P(i)) \qquad Error=1-\max_{i=1}^{n}\{P(i)\}$$

决策树量化纯度

- 当计算出各个特征属性的量化纯度值后使用**信息增益度**来选择出当前数据集的分割特征属性；如果信息增益度的值越大，表示在该特征属性上会损失的纯度越大，那么该属性就越应该在决策树的上层，计算公式为：

$$Gain = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D \mid \mathrm{A})$$

- Gain为A为特征对训练数据集D的信息增益，它为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差

决策树算法的停止条件

- 决策树构建的过程是一个递归的过程，所以必须给定停止条件，否则过程将不会进行停止，一般情况有两种停止条件：
  - 当每个子节点只有一种类型的时候停止构建
  - 当前节点中记录数小于某个阈值，同时迭代次数达到给定值时，停止构建过程，此时使用max(p(i))作为节点的对应类型
- 方式一可能会使树的节点过多，导致过拟合(Overfiting)等问题；比较常用的方式是使用方式二作为停止条件

决策树算法效果评估

- 决策树的效果评估和一般的分类算法一样，采用混淆矩阵来进行计算准确率、召回率、精确率等指标
- 也可以采用叶子节点的纯度值总和来评估算法的效果，值越小，效果越好

		predicted condition		
	total population	prediction positive	prediction negative	Prevalence = $\frac{\Sigma \text{condition positive}}{\Sigma \text{total population}}$	
true condition	condition positive	True Positive (TP)	False Negative (FN) (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection = $\frac{\Sigma TP}{\Sigma \text{condition positive}}$	False Negative Rate (FNR), Miss Rate = $\frac{\Sigma FN}{\Sigma \text{condition positive}}$
	condition negative	False Positive (FP) (Type I error)	True Negative (TN)	False Positive Rate (FPR), Fall-out, Probability of False Alarm = $\frac{\Sigma FP}{\Sigma \text{condition negative}}$	True Negative Rate (TNR), Specificity (SPC) = $\frac{\Sigma TN}{\Sigma \text{condition negative}}$
	Accuracy = $\frac{\Sigma TP + \Sigma TN}{\Sigma \text{total population}}$	Positive Predictive Value (PPV), Precision = $\frac{\Sigma TP}{\Sigma \text{prediction positive}}$	False Omission Rate (FOR) = $\frac{\Sigma FN}{\Sigma \text{prediction negative}}$	Positive Likelihood Ratio (LR+) = $\frac{TPR}{FPR}$	Diagnostic Odds Ratio (DOR) = $\frac{LR+}{LR-}$
		False Discovery Rate (FDR) = $\frac{\Sigma FP}{\Sigma \text{prediction positive}}$	Negative Predictive Value (NPV) = $\frac{\Sigma TN}{\Sigma \text{prediction negative}}$	Negative Likelihood Ratio (LR−) = $\frac{FNR}{TNR}$	

$$C(T) = \sum_{t=1}^{leaf} \frac{|D_t|}{|D|} \mathrm{H}(t)$$

决策树算法效果评估

- 决策树的损失函数(该值越小，算法效果越好)

$$loss = \sum_{t=1}^{leaf} \frac{|D_t|}{|D|} \mathrm{H}(t)$$

决策树直观理解结果计算

ID	拥有房产(是/否)	婚姻状态(单身\已婚\离婚)	年收入(单位:千元)	无法偿还债务(是/否)
1	是	单身	125	否
2	否	已婚	100	否
3	否	单身	100	否
4	是	已婚	110	否
5	是	离婚	60	否
6	否	离婚	95	是
7	否	单身	85	是
8	否	已婚	75	否
9	否	单身	90	是
10	是	离婚	220	否

$$Info(D) = -\sum_{i=1}^{2} P(i)\log_2(P(i)) = -0.7*\log_2(0.7) - 0.3\log_2(0.3) = 0.88$$

$$Info(D_{有房产}) = -4/4*\log_2(4/4) - 0*\log_2(0) = 0 \qquad Info(D_{无房产}) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$$

$$Gain(房产) = Info(D) - \sum_{j=1}^{2} \frac{N(D_j)}{N(D)} Info(D_j) = 0.88 - 0.4*0 - 0.6*1 = 0.28$$

$$Gain(婚姻) = 0.205 \qquad Gain(收入 = 97.5) = 0.395$$

第一个分割属性为收入

决策树直观理解结果计算

ID	拥有房产(是/否)	婚姻状态(单身\已婚\离婚)	年收入(单位:千元)	无法偿还债务(是/否)
1	是	单身	125	否
2	否	已婚	100	否
3	否	单身	100	否
4	是	已婚	110	否
5	是	离婚	60	否
6	否	离婚	95	是
7	否	单身	85	是
8	否	已婚	75	否
9	否	单身	90	是
10	是	离婚	220	否

$$p(x=是)=0.4 \qquad p(x=否)=0.6$$

$$p(y=是\mid x=是)=0 \qquad p(y=否\mid x=是)=1$$

$$p(y=是\mid x=否)=0.5 \qquad p(y=否\mid x=否)=0.5$$

$$Info(D_{有房产})=H(D_{有房产})=-1*\log_2(1)-0*\log_2(0)=0$$

$$Info(D_{无房产})=H(D_{无房产})=-0.5\log_2(0.5)-0.5\log_2(0.5)=1$$

X拥有房产(是/否)	Y无法偿还债务(是/否)
是	否
否	否
否	否
是	否
是	否
否	是
否	是
否	否
否	是
是	否

$$Gain(房产)=Info(D)-\sum_{j=1}^{2}\frac{N(D_j)}{N(D)}Info(D_j)=0.88-0.4*0-0.6*1=0.28$$

决策树直观理解结果计算

X(年收入)	60	75	85	90	95	100	100	110	125	220
Y(无法偿还债务)	否	否	是	是	是	否	否	否	否	否
分割点	~~67.5~~	**80**	~~87.5~~	~~92.5~~	**97.5**	~~100~~	~~105~~	~~117.5~~	~~172.5~~	

$$Gain(\text{收入} = 97.5) = 0.395$$

$$Gain(\text{收入} = 80) = 0.116$$

决策树生成算法

- 建立决策树的主要是以下三种算法
  - ID3
  - C4.5
  - CART（Classification And Regression Tree）

ID3算法

- ID3算法是决策树的一个经典的构造算法，内部使用**信息熵**以及**信息增益**来进行构建；每次迭代选择信息增益最大的特征属性作为分割属性

$$\mathrm{H}(\mathrm{D}) = -\sum_{i=1}^{n} P(i)\log_2(P(i))$$

$$Gain = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D \mid \mathrm{A})$$

ID3算法优缺点

- 优点:
  - 决策树构建速度快；实现简单；
- 缺点：
  - 计算依赖于特征数目较多的特征，而属性值最多的属性并不一定最优
  - ID3算法不是递增算法
  - ID3算法是单变量决策树，对于特征属性之间的关系不会考虑
  - 抗噪性差
  - 只适合小规模数据集，需要将数据放到内存中

C4.5算法

- 在ID3算法的基础上，进行算法优化提出的一种算法(C4.5)；现在C4.5已经是特别经典的一种决策树构造算法；使用**信息增益率**来取代ID3算法中的信息增益，在树的构造过程中会进行**剪枝**操作进行优化；能够自动完成对连续属性的离散化处理；C4.5算法在选中分割属性的时候选择信息增益率最大的属性，涉及到的公式为：

$$\mathrm{H}(\mathrm{D}) = -\sum_{i=1}^{n} P(i)\log_2(P(i))$$

$$Gain(A) = \Delta = \mathrm{H}(\mathrm{D}) - \mathrm{H}(D \mid \mathrm{A})$$

$$Gain\_ratio(A) = \frac{Gain(A)}{\mathrm{H}(A)}$$

C4.5算法优缺点

- 优点：
  - 产生的规则易于理解
  - 准确率较高
  - 实现简单
- 缺点：
  - 对数据集需要进行多次顺序扫描和排序，所以效率较低
  - 只适合小规模数据集，需要将数据放到内存中

CART算法

- 使用**基尼系数**作为数据纯度的量化指标来构建的决策树算法就叫做CART(Classification And Regression Tree，分类回归树)算法。CART算法使用**GINI增益**作为分割属性选择的标准，选择GINI增益最大的作为当前数据集的分割属性；可用于分类和回归两类问题。强调备注：**CART构建是二叉树**。

$$Gini = 1 - \sum_{i=1}^{n} P(i)^2$$

$$Gain = \Delta = \mathrm{Gini}(\mathrm{D}) - \mathrm{Gini}(D \mid \mathrm{A})$$

ID3、C4.5、CART分类树算法总结

- ID3和C4.5算法均只适合在小规模数据集上使用
- ID3和C4.5算法都是单变量决策树
- 当属性值取值比较多的时候，最好考虑C4.5算法，ID3得出的效果会比较差
- 决策树分类一般情况只适合小数据量的情况(数据可以放内存)
- CART算法是三种算法中最常用的一种决策树构建算法。
- 三种算法的区别仅仅只是对于当前树的评价标准不同而已，ID3使用**信息增益**、C4.5使用**信息增益率**、CART使用**基尼系数**。
- CART算法构建的一定是二叉树，ID3和C4.5构建的不一定是二叉树。

决策树案例一：鸢尾花数据分类

- 使用决策树算法API对鸢尾花数据进行分类操作，并理解及进行决策树API的相关参数优化
- 数据来源：鸢尾花数据

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Iris Data Set

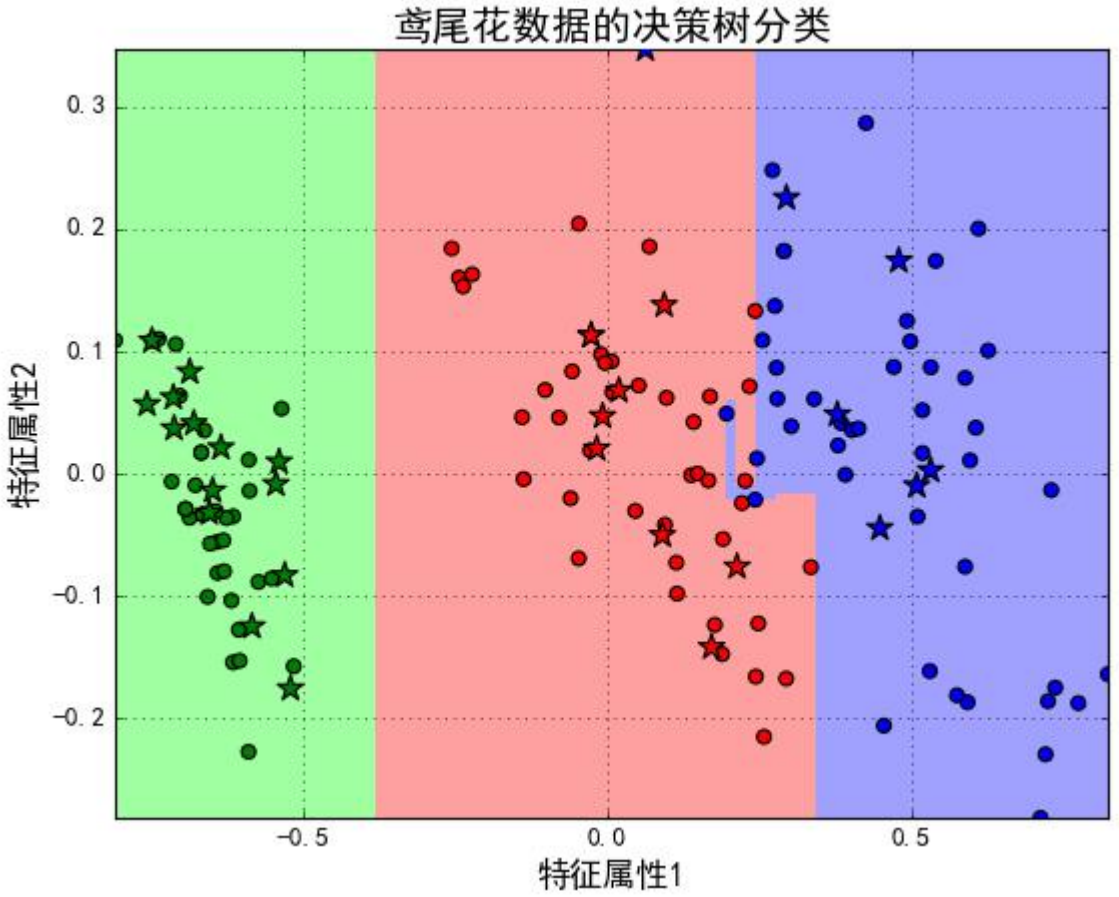
Download: Data Folder, Data Set Description

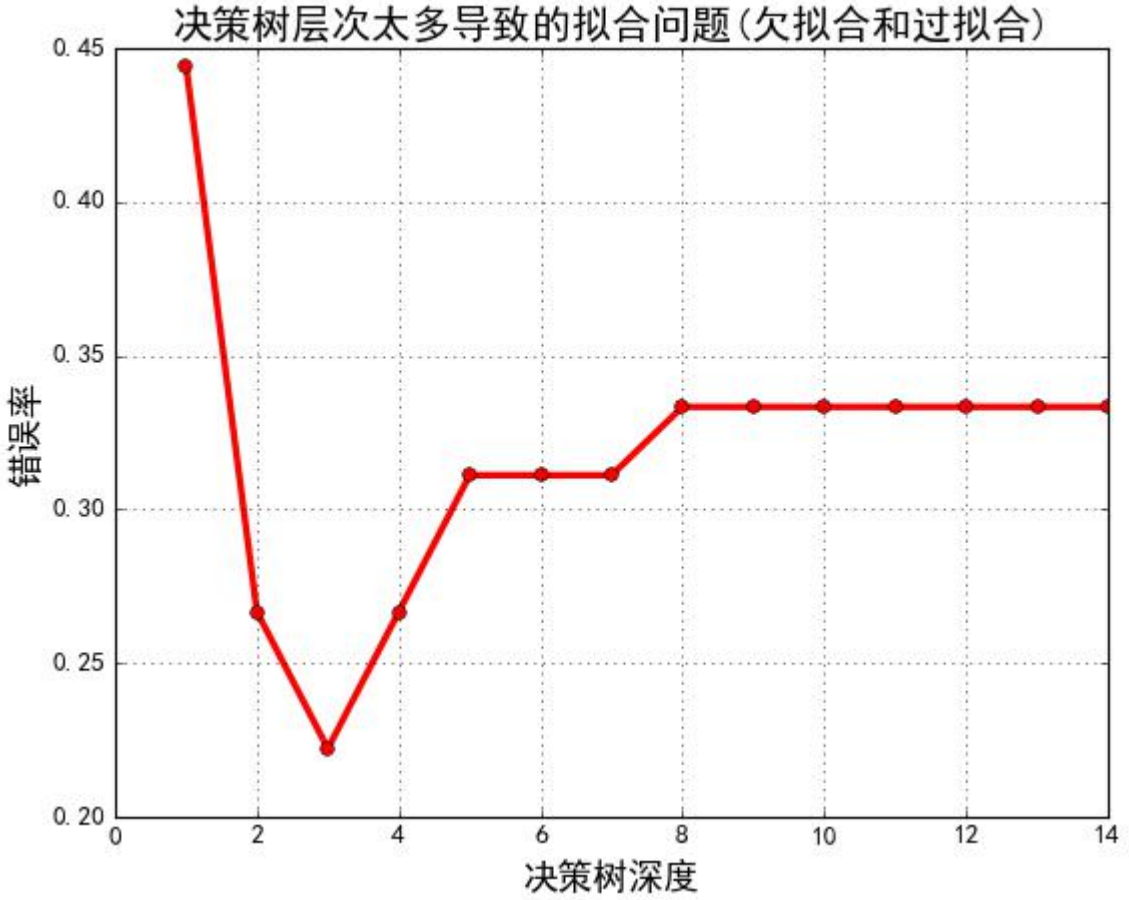
Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1323697

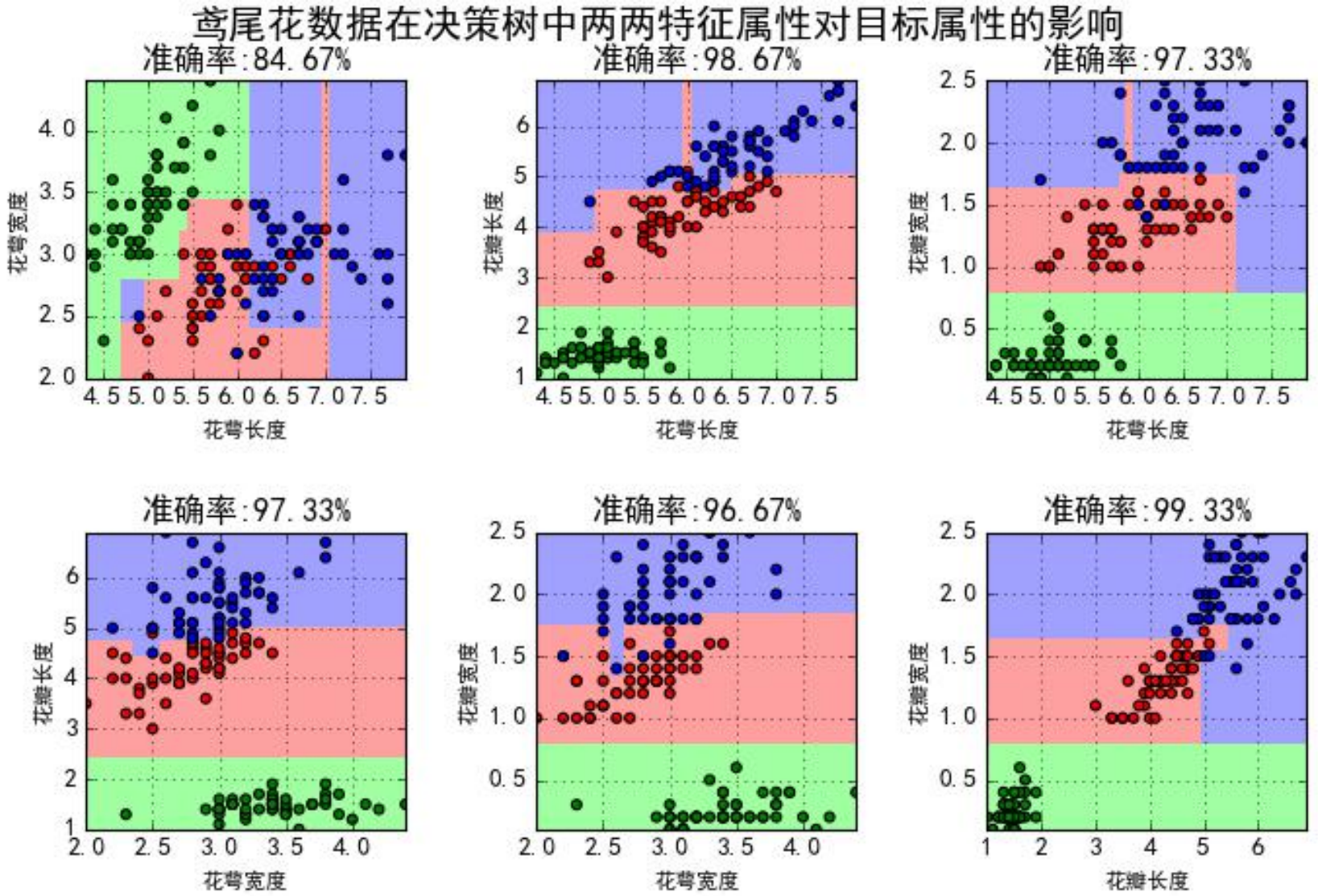
class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, class_weight=None*) ¶ [source]

决策树案例一：鸢尾花数据分类





决策树案例一：鸢尾花数据分类



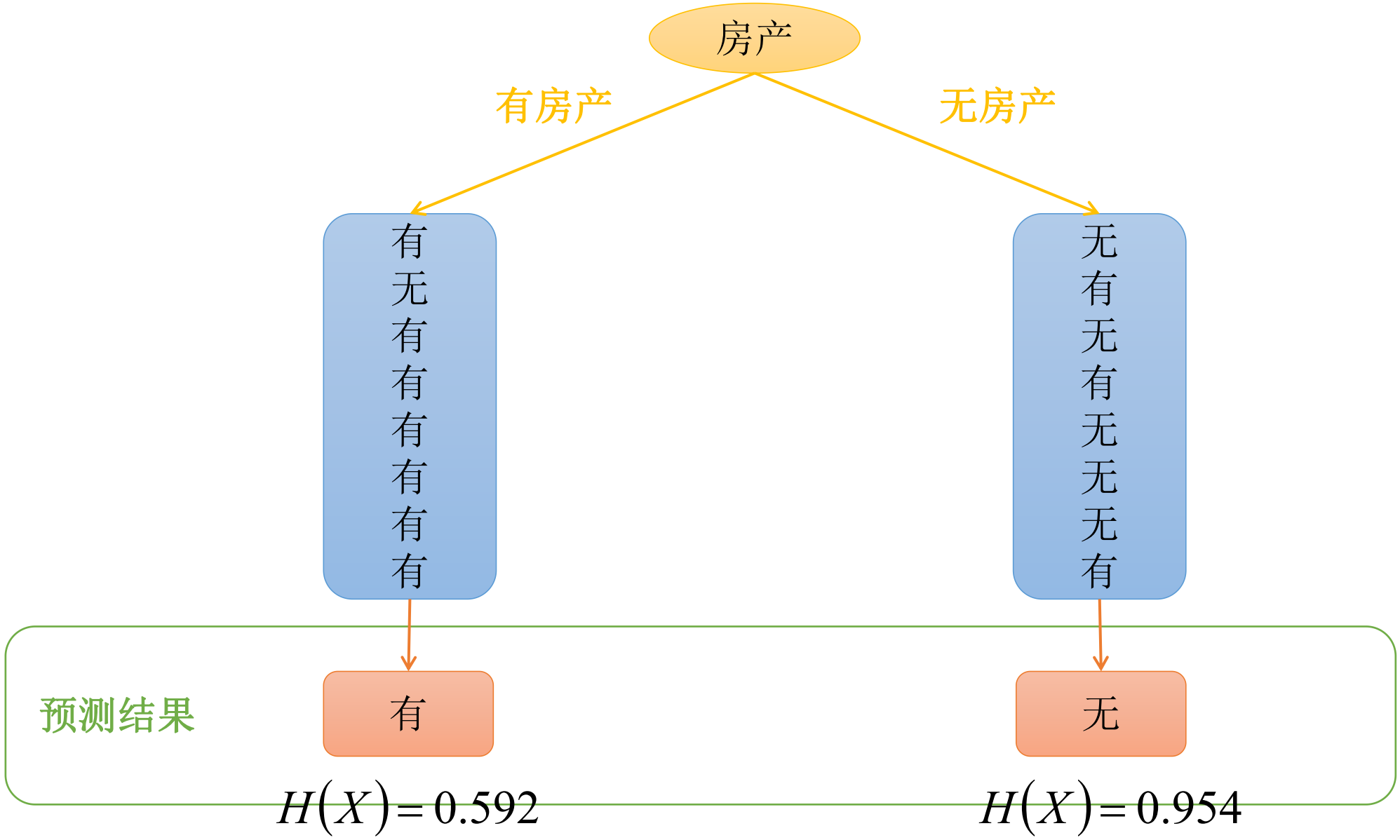
分类树和回归树的区别

- 分类树采用信息增益、信息增益率、基尼系数来评价树的效果，都是基于概率值进行判断的；而分类树的叶子节点的预测值一般为叶子节点中概率最大的类别作为当前叶子的预测值。
- 在回归树种，叶子节点的预测值一般为叶子节点中所有值的均值来作为当前叶子节点的预测值。所以在回归树中一般采用MSE作为树的评价指标，即均方差。

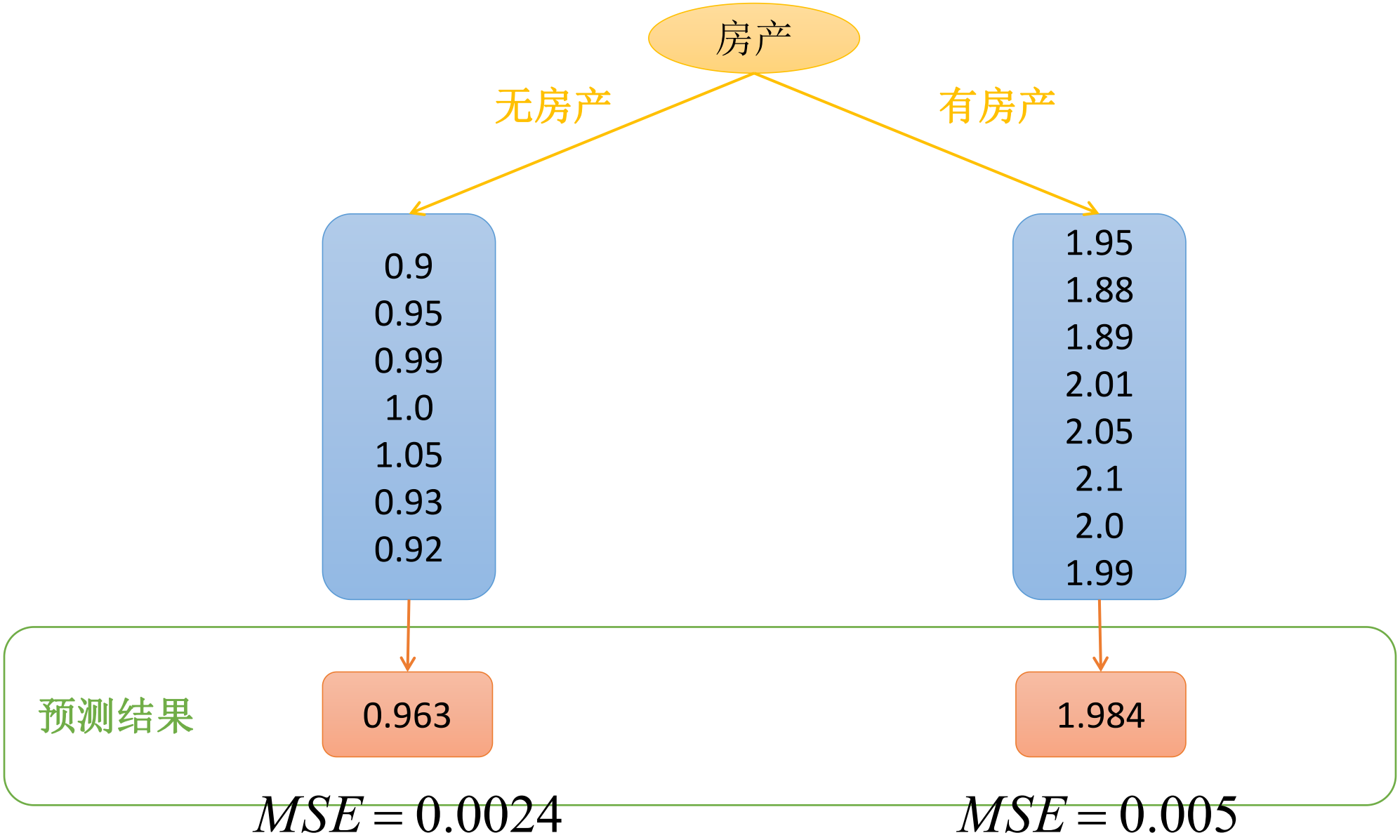
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- 一般情况下，只会使用CART算法构建回归树。

分类树和回归树的区别



分类树和回归树的区别


房产
无房产
有房产
0.9
0.95
0.99
1.0
1.05
0.93
0.92
1.95
1.88
1.89
2.01
2.05
2.1
2.0
1.99
预测结果
0.963
1.984
$MSE = 0.0024$
$MSE = 0.005$

决策树案例二：波士顿房屋租赁价格预测

- 使用决策树算法API对波士顿房屋租赁数据进行回归操作，预测房屋的价格信息，并理解及进行决策树API的相关参数优化
- 数据来源：波士顿房屋租赁数据

Housing Data Set

Download: Data Folder, Data Set Description

Abstract: Taken from StatLib library

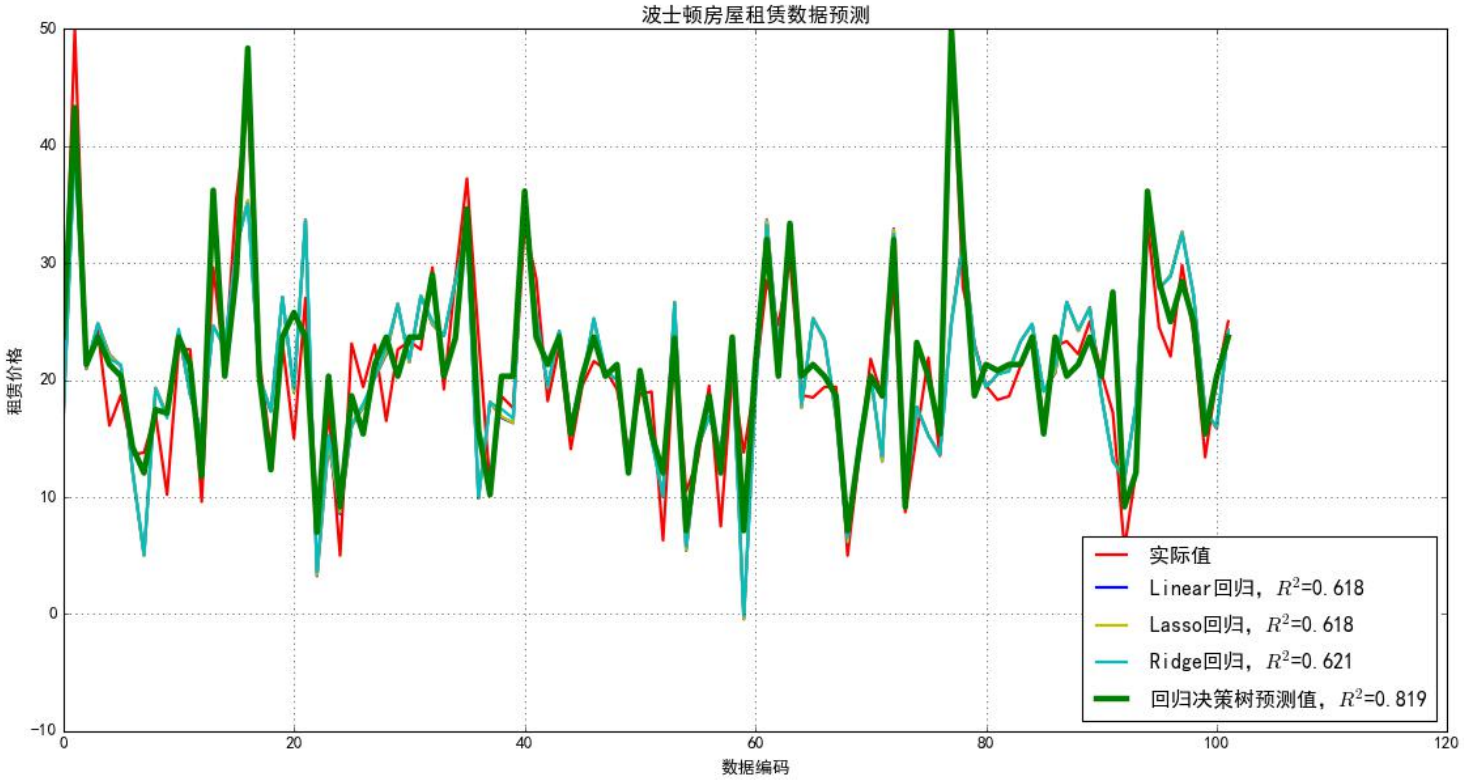
Data Set Characteristics:	Multivariate	**Number of Instances:**	506	**Area:**	N/A
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	14	**Date Donated**	1993-07-07
Associated Tasks:	Regression	**Missing Values?**	No	**Number of Web Hits:**	328263

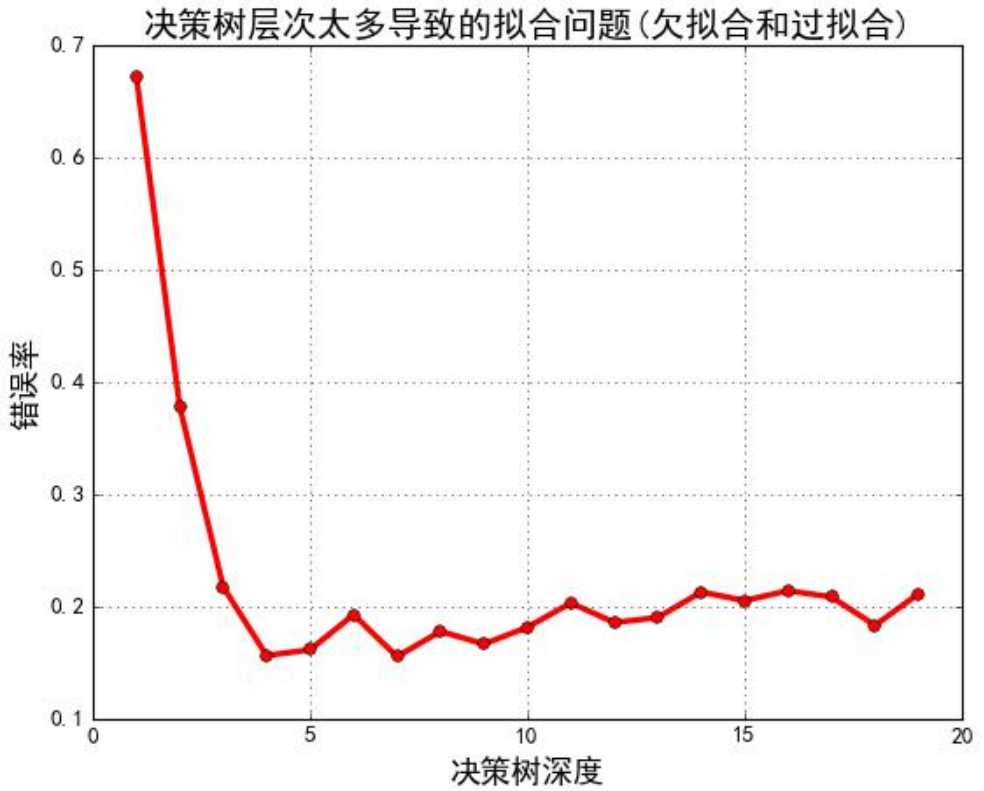
Attribute Information:

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in $1000's

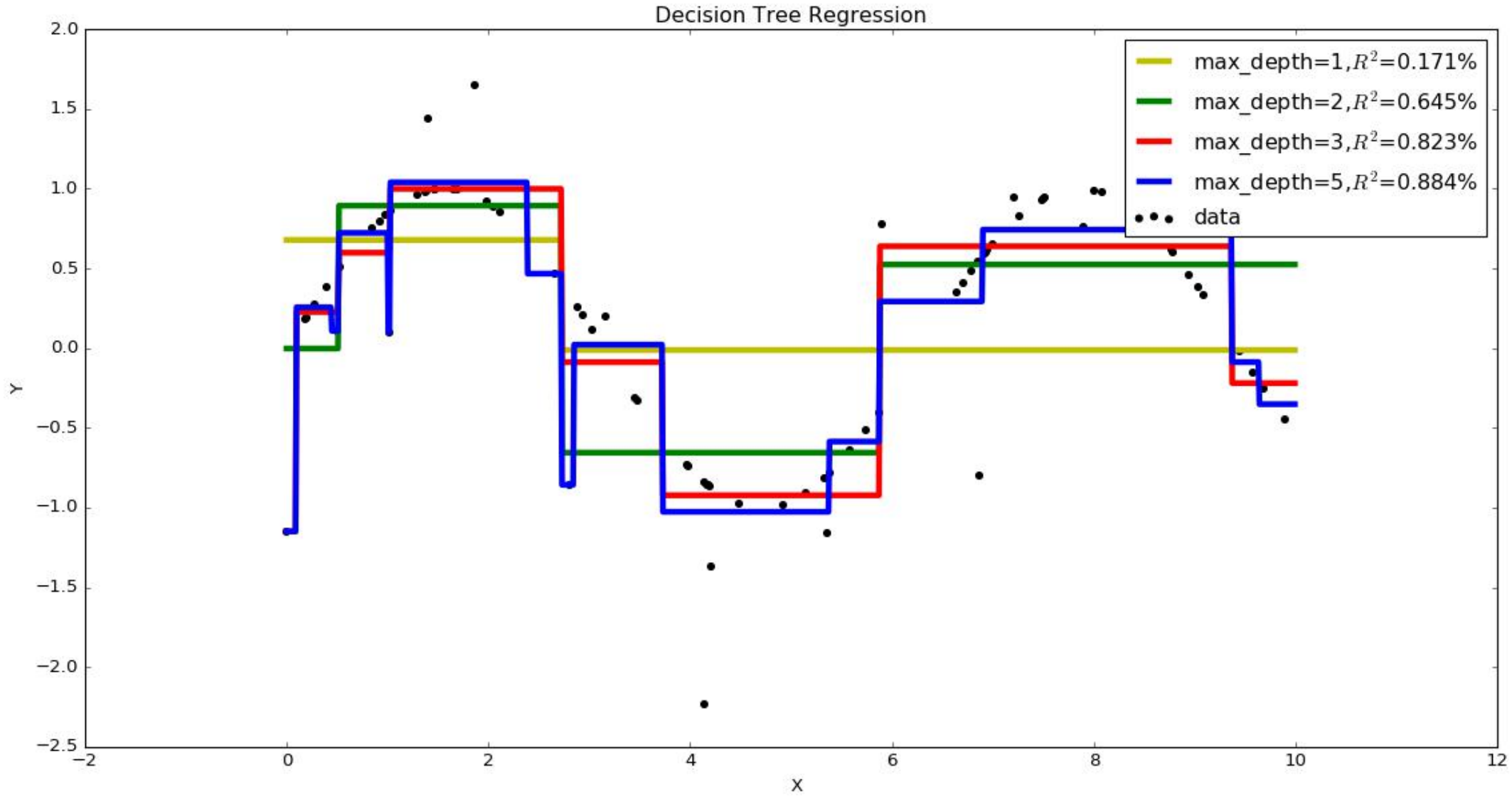
class sklearn.tree.**DecisionTreeRegressor**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None*) ¶ [source]

决策树案例二：波士顿房屋租赁价格预测





决策树过拟合和欠拟合


Decision Tree Regression
max_depth=1,R^2=0.171%
max_depth=2,R^2=0.645%
max_depth=3,R^2=0.823%
max_depth=5,R^2=0.884%
data
Y
X
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−2
0
2
4
6
8
10
12

决策树优化策略

- 剪枝优化
  - 决策树过渡拟合一般情况是由于节点太多导致的，剪枝优化对决策树的正确率影响是比较大的，也是最常用的一种优化方式。
- Random Forest
  - 利用训练数据随机产生多个决策树，形成一个森林。然后使用这个森林对数据进行预测，选取最多结果作为预测结果。

决策树的剪枝

- 决策树的剪枝是决策树算法中最基本、最有用的一种优化方案，主要分为两大类：
  - **前置剪枝**：在构建决策树的过程中，提前停止。结果是决策树一般比较小，实践证明这种策略无法得到比较好的结果。
  - **后置剪枝**：在决策树构建好后，然后再开始裁剪，一般使用两种方式：1)用单一叶子节点代替整个子树，叶节点的分类采用子树中最主要的分类；2)将一个子树完全替代另外一棵子树；后置剪枝的主要问题是计算效率问题，存在一定的浪费情况。
- 后剪枝总体思路(交叉验证)：
  - 由完全树T_0开始，剪枝部分节点得到T_1，在此剪枝得到T_2.....直到仅剩树根的树T_k
  - 在**验证数据集**上对这k+1个树进行评价，选择最优树T_a（损失函数最小的树）

决策树剪枝过程

- 对于给定的决策树T_0
  - 计算所有内部非叶子节点的**剪枝系数**
  - 查找**最小剪枝系数**的节点，将其子节点进行删除操作，进行剪枝得到决策树T_k；如果存在多个最小剪枝系数节点，选择包含**数据项最多**的节点进行剪枝操作
  - 重复上述操作，直到产生的剪枝决策树T_k只有1个节点
  - 得到决策树$T_0T_1T_2....T_k$
  - 使用**验证样本集**选择最优子树T_a
- 使用验证集选择最优子树的标准，可以使用原始损失函数来考虑：

$$loss = \sum_{t=1}^{leaf} \frac{|D_t|}{|D|} \mathrm{H}(t)$$

决策树剪枝损失函数及剪枝系数

- 原始损失函数 $$loss = \sum_{t=1}^{leaf} \frac{|D_t|}{|D|} \mathrm{H}(t)$$

- 叶节点越多，决策树越复杂，损失越大；修正添加剪枝系数，修改后的损失函数为：

$$loss_\alpha = loss + \alpha * leaf$$

- 考虑根节点为r的子树，剪枝前后的损失函数分别为loss(R)和loss(r)，当这两者相等的时候，可以求得剪枝系数

$$loss_\alpha(r) = loss(r) + \alpha \qquad loss_r(R) = loss(R) + \alpha * R_{leaf}$$

$$\alpha = \frac{loss(r) - loss(R)}{R_{leaf} - 1}$$

ID3、C4.5、CART分类树算法总结

算法	支持模型	树结构	特征选择	连续值处理	缺失值处理	剪枝
ID3	分类	多叉树	信息增益	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益率	支持	支持	支持
CART	分类、回归	二叉树	基尼系数、均方差	支持	支持	支持

集成学习(Ensemble Learning)

- 集成学习的思想是将若干个学习器(分类器&回归器)组合之后产生一个新学习器。弱分类器(weak learner)指那些分类准确率只稍微好于随机猜测的分类器(error rate < 0.5)；
- 集成算法的成功在于保证弱分类器的多样性(Diversity)。而且集成不稳定的算法也能够得到一个比较明显的性能提升。
- 常见的集成学习思想有：
  - Bagging
  - Boosting
  - Stacking

集成学习(Ensemble Learning)

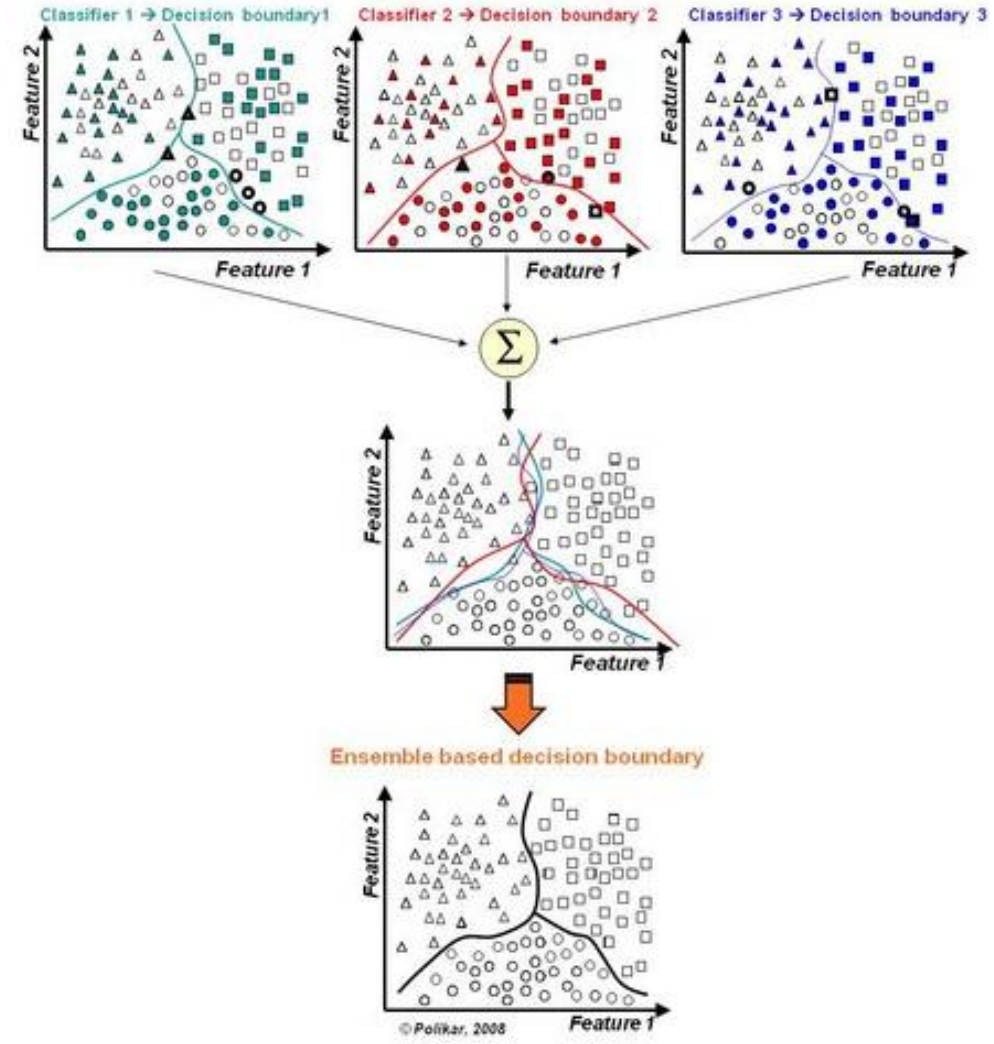


Figure 1: Combining an ensemble of classifiers for reducing classification error and/or model selection.

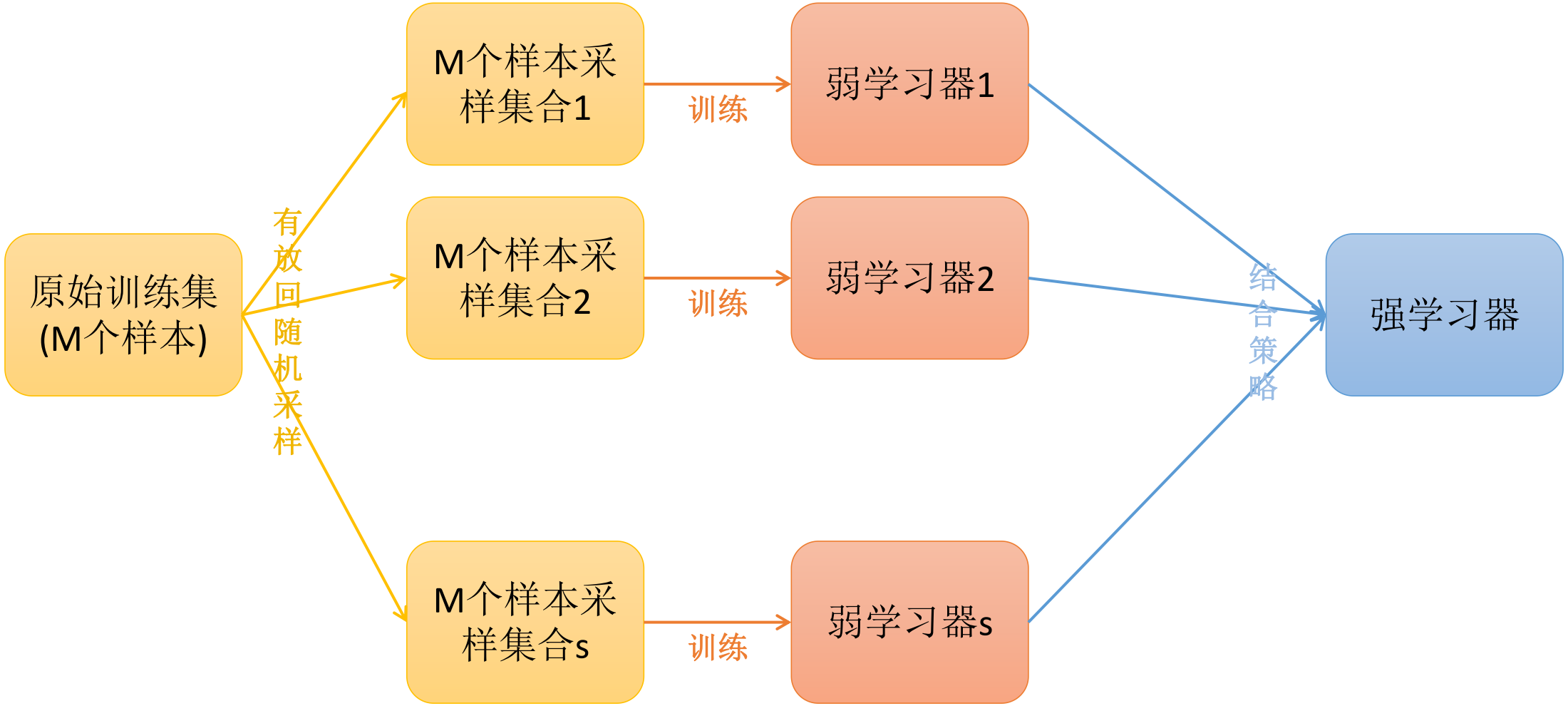
Why need Ensemble Learning?

- 1. 弱分类器间存在一定的差异性，这会导致分类的边界不同，也就是说可能存在错误。那么将多个弱分类器合并后，就可以得到更加合理的边界，减少整体的错误率，实现更好的效果；
- 2. 对于数据集过大或者过小，可以分别进行划分和有放回的操作产生不同的数据子集，然后使用数据子集训练不同的分类器，最终再合并成为一个大的分类器；
- 3. 如果数据的划分边界过于复杂，使用线性模型很难描述情况，那么可以训练多个模型，然后再进行模型的融合；
- 4. 对于多个异构的特征集的时候，很难进行融合，那么可以考虑每个数据集构建一个分类模型，然后将多个模型融合。

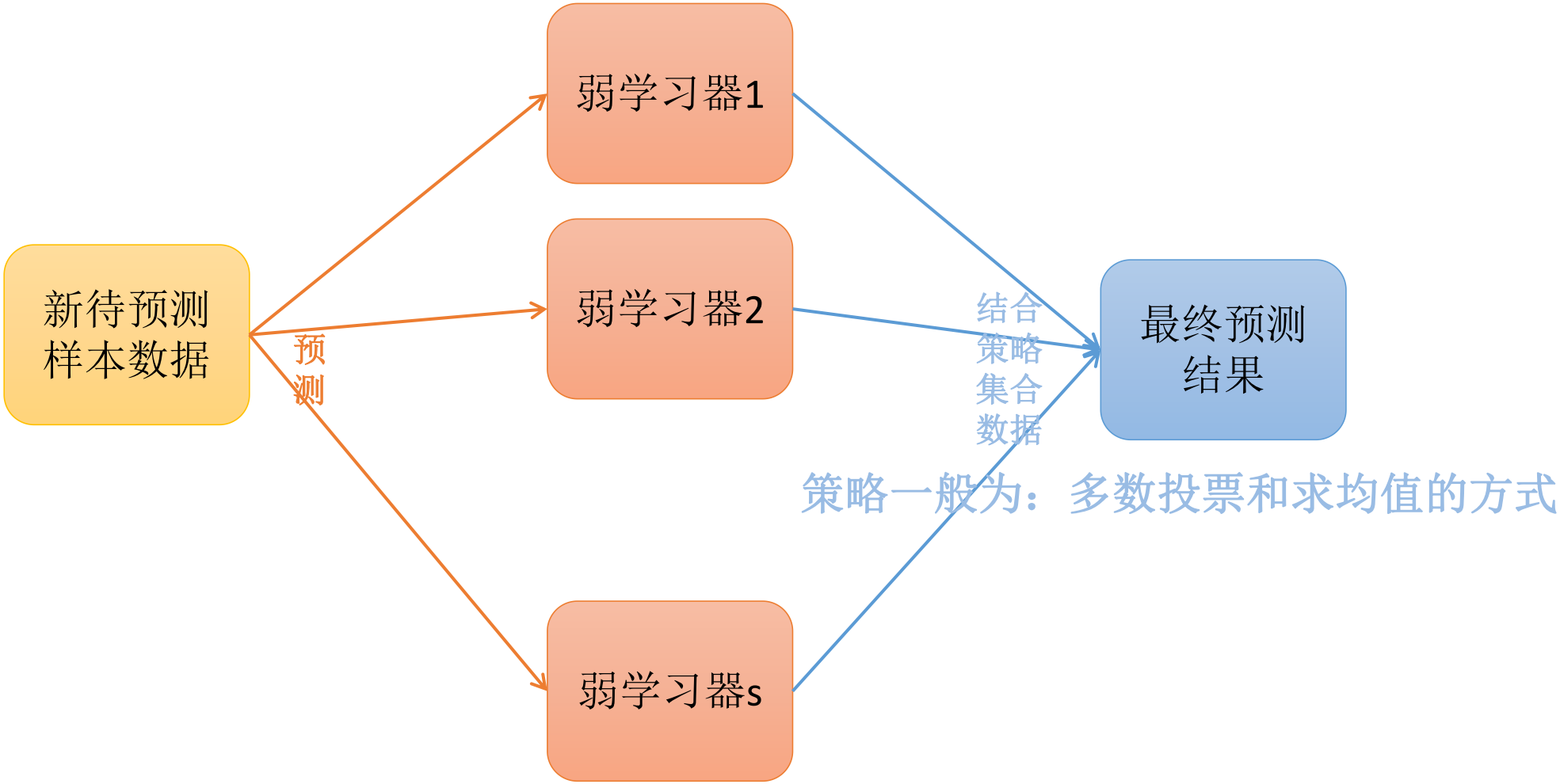
Bagging方法

- Bagging方法又叫做自举汇聚法(Bootstrap Aggregating)，思想是：在原始数据集上通过**有放回的抽样**的方式，重新选择出S个新数据集来分别训练S个分类器的集成技术。也就是说这些模型的训练数据中允许存在重复数据。
- Bagging方法训练出来的模型在预测新样本分类的时候，会使用**多数投票**或者**求均值**的方式来统计最终的分类结果。
- Bagging方法的弱学习器可以是基本的算法模型，eg: Linear、Ridge、Lasso、Logistic、Softmax、ID3、C4.5、CART、SVM、KNN等。
- 备注：Bagging方式是有放回的抽样，并且每个子集的样本数量必须和原始样本数量一致，但是子集中允许存在重复数据。

Bagging方法_训练过程



Bagging方法_预测过程

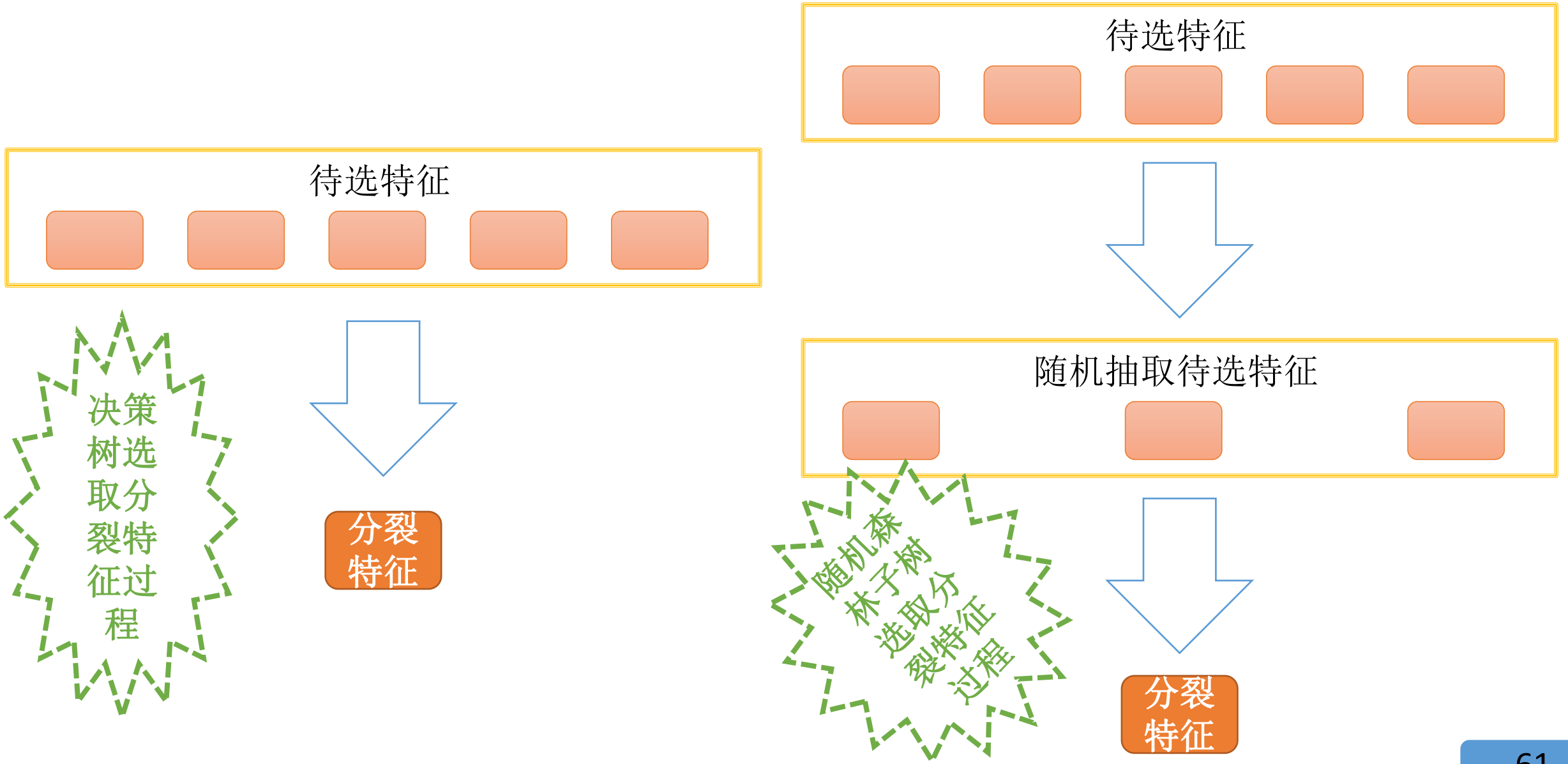


策略一般为：多数投票和求均值的方式

随机森林(Random Forest)

- 在Bagging策略的基础上进行修改后的一种算法
  - 从样本集中用Bootstrap采样选出n个样本；
  - 从所有属性中随机选择K个属性，选择出最佳分割属性作为节点创建决策树；
  - 重复以上两步m次，即建立m棵决策树；
  - 这m个决策树形成随机森林，通过投票表决结果决定数据属于那一类

随机森林(Random Forest)



RF的推广算法

- RF算法在实际应用中具有比较好的特性，应用也比较广泛，主要应用在：分类、回归、特征转换、异常点检测等。常见的RF变种算法如下：
  - Extra Tree
  - Totally Random Trees Embedding(TRTE)
  - Isolation Forest

Extra Tree

- Extra Tree是RF的一个变种，原理基本和RF一样，区别如下：
  - 1. RF会随机采样来作为子决策树的训练集，而Extra Tree每个子决策树采用原始数据集训练；
  - 2. RF在选择划分特征点的时候会和传统决策树一样，会基于信息增益、信息增益率、基尼系数、均方差等原则来选择最优特征值；而Extra Tree会随机的选择一个特征值来划分决策树。
- Extra Tree因为是随机选择特征值的划分点，这样会导致决策树的规模一般大于RF所生成的决策树。也就是说Extra Tree模型的方差相对于RF进一步减少。在某些情况下，Extra Tree的泛化能力比RF的强。

Totally Random Trees Embedding(TRTE)

- TRTE是一种非监督的数据转化方式。将低维的数据集映射到高维，从而让映射到高维的数据更好的应用于分类回归模型。
- TRTE算法的转换过程类似RF算法的方法，建立T个决策树来拟合数据。当决策树构建完成后，数据集里的每个数据在T个决策树中叶子节点的位置就定下来了，将位置信息转换为向量就完成了特征转换操作。
- 案例：有3棵决策树，每棵决策树有5个叶子节点，某个数据x划分到第一个决策树的第3个叶子节点，第二个决策树的第一个叶子节点，第三个决策树的第第五个叶子节点，那么最终的x映射特征编码为:(0,0,1,0,0, 1,0,0,0,0, 0,0,0,0,1)

Isolation Forest(IForest)

- IForest是一种异常点检测算法，使用类似RF的方式来检测异常点；IForest算法和RF算法的区别在于：
  - 1. 在随机采样的过程中，一般只需要少量数据即可；
  - 2. 在进行决策树构建过程中，IForest算法会随机选择一个划分特征，并对划分特征随机选择一个划分阈值；
  - 3. IForest算法构建的决策树一般深度max_depth是比较小的。
- 区别原因：目的是异常点检测，所以只要能够区分异常的即可，不需要大量数据；另外在异常点检测的过程中，一般不需要太大规模的决策树。

Isolation Forest(IForest)

- 对于异常点的判断，则是将测试样本x拟合到T棵决策树上。计算在每棵树上该样本的叶子节点的深度$h_t(x)$。从而计算出平均深度h(x)；然后就可以使用下列公式计算样本点x的异常概率值，p(s,m)的取值范围为[0,1]，越接近于1，则是异常点的概率越大。

$$p(x,m)=2^{-\frac{h(x)}{c(m)}}$$

$c(m)=2\ln(m-1)+\xi-2\frac{m-1}{m}$；$m$为样本个数，$\xi$为欧拉常数

RF随机森林总结

- RF的主要优点：
  - 1. 训练可以并行化，对于大规模样本的训练具有速度的优势；
  - 2. 由于进行随机选择决策树划分特征列表，这样在样本维度比较高的时候，仍然具有比较高的训练性能；
  - 3. 给以给出各个特征的重要性列表；
  - 4. 由于存在随机抽样，训练出来的模型方差小，泛化能力强；
  - 5. RF实现简单；
  - 6. 对于部分特征的缺失不敏感。
- RF的主要缺点：
  - 1. 在某些噪音比较大的特征上，RF模型容易陷入过拟合；
  - 2. 取值比较多的划分特征对RF的决策会产生更大的影响，从而有可能影响模型的效果。

随机森林算法案例

- 使用随机森林算法API对乳腺癌数据进行分类操作，根据特征属性预测是否会得乳腺癌的四个目标属性的值，并理解随机森林中决策树数量和决策树深度对模型的影响
- 数据来源：乳腺癌数据

(bool) Hinselmann: target variable
(bool) Schiller: target variable
(bool) Cytology: target variable
(bool) Biopsy: target variable

目标属性

Attribute Information:

(int) Age
(int) Number of sexual partners
(int) First sexual intercourse (age)
(int) Num of pregnancies
(bool) Smokes
(bool) Smokes (years)
(bool) Smokes (packs/year)
(bool) Hormonal Contraceptives
(int) Hormonal Contraceptives (years)
(bool) IUD
(int) IUD (years)
(bool) STDs
(int) STDs (number)
(bool) STDs:condylomatosis
(bool) STDs:cervical condylomatosis
(bool) STDs:vaginal condylomatosis
(bool) STDs:vulvo-perineal condylomatosis
(bool) STDs:syphilis
(bool) STDs:pelvic inflammatory disease
(bool) STDs:genital herpes
(bool) STDs:molluscum contagiosum
(bool) STDs:AIDS
(bool) STDs:HIV
(bool) STDs:Hepatitis B
(bool) STDs:HPV
(int) STDs: Number of diagnosis
(int) STDs: Time since first diagnosis
(int) STDs: Time since last diagnosis
(bool) Dx:Cancer
(bool) Dx:CIN
(bool) Dx:HPV
(bool) Dx

特征属性

Cervical cancer (Risk Factors) Data Set

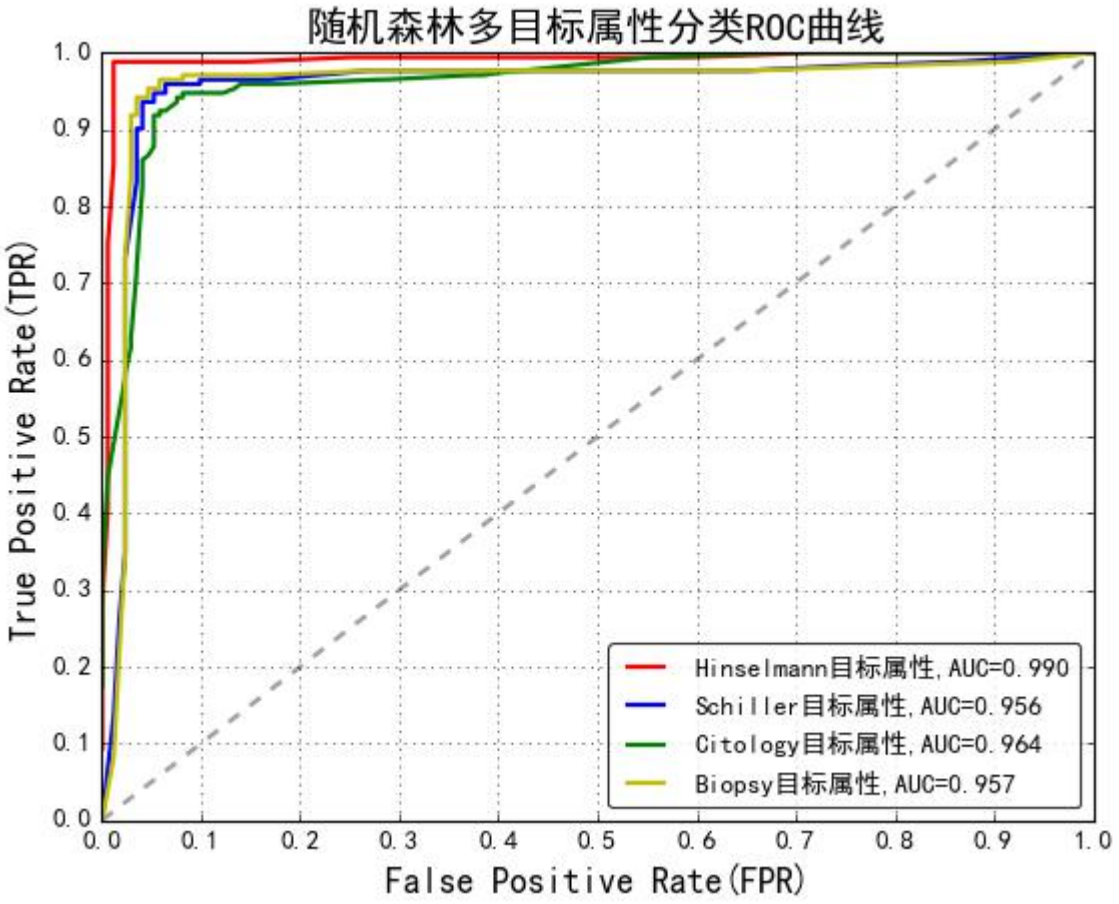
Download: Data Folder, Data Set Description

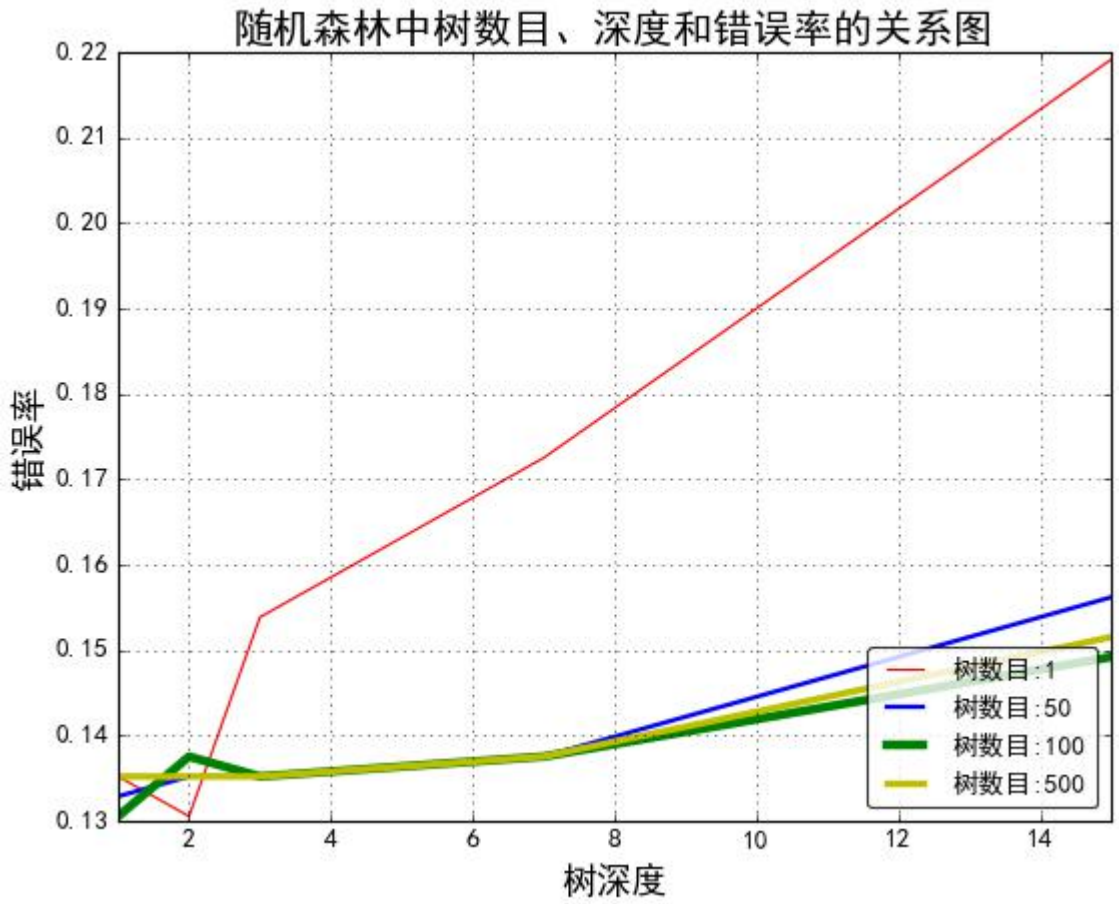
Abstract: This dataset focuses on the prediction of indicators/diagnosis of cervical cancer. The features cover demographic information, habits, and historic medical records.

Data Set Characteristics:	Multivariate	**Number of Instances:**	858	**Area:**	Life
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	36	**Date Donated**	2017-03-03
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	3687

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) [source]

随机森林算法案例


随机森林多目标属性分类ROC曲线
True Positive Rate(TPR)
False Positive Rate(FPR)
Hinselmann目标属性, AUC=0.990
Schiller目标属性, AUC=0.956
Citology目标属性, AUC=0.964
Biopsy目标属性, AUC=0.957


随机森林中树数目、深度和错误率的关系图
错误率
树深度
树数目:1
树数目:50
树数目:100
树数目:500

RF scikit-learn相关参数

参数	RandomForestClassifier	RandomForestRegressor
criterion	指定划分标准，默认为gini，不支持其它参数	指定划分标准，可选”mse”和”mae”; 默认mse
loss	不支持	指定误差的计算方式，可选参数”linear”, “square”, “exponential”, 默认为”linear”； 一般不用改动
n_estimators	最大迭代次数，也就是最多允许的决策树的数目，值过小可能会导致欠拟合，值过大可能会导致过拟合，一般50~100比较适合，默认10	
max_features	给定在进行最佳特征划分的时候，选择多少个特征进行考虑；默认为auto；max_features=sqrt(n_features)；一般不建议改动，具体参数见官网文档。	
max_depth	给定树的深度，默认为None，表示一致扩展到叶子节点足够纯或者样本数小于min_samples_split	
min_samples_split	给定树构建过程中，叶子节点中最少样本数量，默认为2	
min_samples_leaf	给定一棵树最少叶子数量，默认为1	
bootstrap	是否进行有放回的重采样，默认为True	

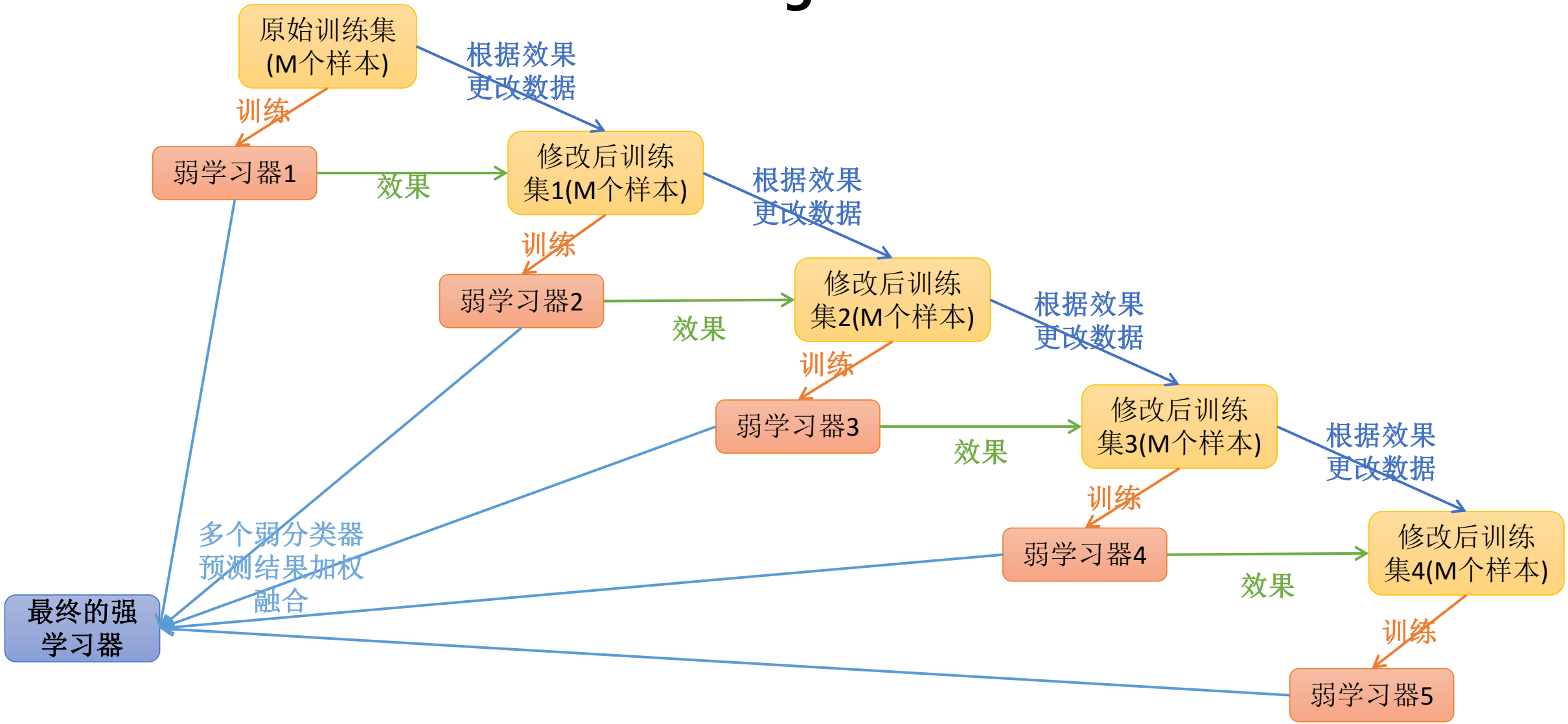
随机森林的思考

- 在随机森林的构建过程中，由于各棵树之间是没有关系的，相对独立的；在构建的过程中，构建第m棵子树的时候，不会考虑前面的m-1棵树。
- 思考：
  - 如果在构建第m棵子树的时候，考虑到前m-1棵子树的结果，会不会对最终结果产生有益的影响？
  - 各个决策树组成随机森林后，在形成最终结果的时候能不能给定一种既定的决策顺序呢？(也就是那颗子树先进行决策、那颗子树后进行决策)

Boosting

- 提升学习（Boosting）是一种机器学习技术，可以用于**回归**和**分类**的问题，它每一步产生**弱预测模型**(如决策树)，并**加权累加**到总模型中；如果每一步的弱预测模型的生成都是依据损失函数的梯度方式的，那么就称为梯度提升(Gradient boosting)；
- 提升技术的意义：如果一个问题存在**弱预测模型**，那么可以通过提升技术的办法得到一个**强预测模型；**
- 常见的模型有：
  - Adaboost
  - Gradient Boosting(GBT/GBDT/GBRT)

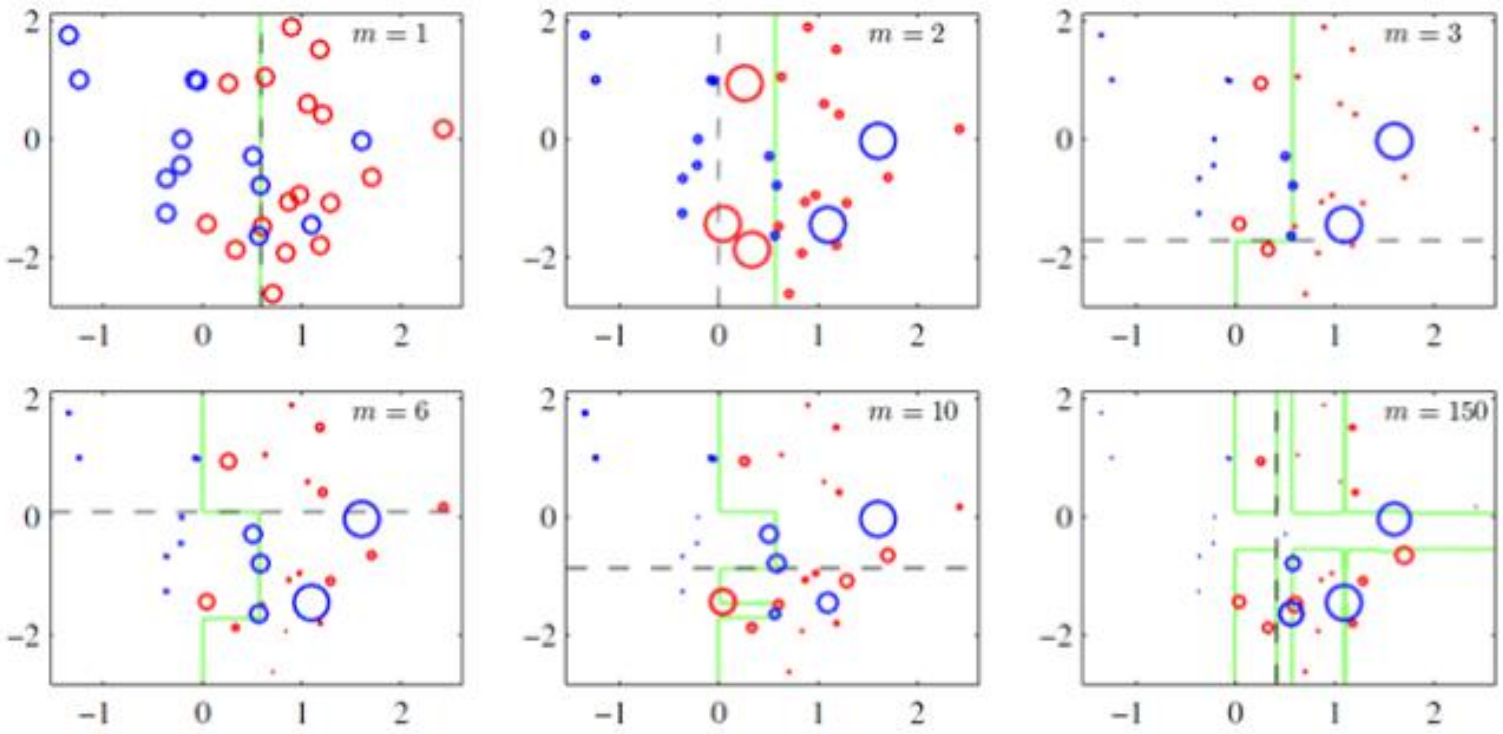
Boosting


原始训练集
(M个样本)
根据效果
更改数据
训练
弱学习器1
效果
修改后训练
集1(M个样本)
根据效果
更改数据
训练
弱学习器2
效果
修改后训练
集2(M个样本)
根据效果
更改数据
训练
弱学习器3
效果
修改后训练
集3(M个样本)
根据效果
更改数据
训练
弱学习器4
效果
修改后训练
集4(M个样本)
训练
弱学习器5
多个弱分类器
预测结果加权
融合
最终的强
学习器

AdaBoost算法原理

- Adaptive Boosting是一种迭代算法。每轮迭代中会在训练集上产生一个新的学习器，然后使用该学习器对所有样本进行预测，以评估每个样本的重要性(Informative)。换句话来讲就是，算法会为每个样本赋予一个权重，每次用训练好的学习器标注/预测各个样本，如果某个样本点被预测的越正确，则将其权重降低；否则提高样本的权重。权重越高的样本在下一个迭代训练中所占的比重就越大，也就是说越难区分的样本在训练过程中会变得越重要；
- 整个迭代过程直到错误率足够小或者达到一定的迭代次数为止。

样本加权



Adaboost算法

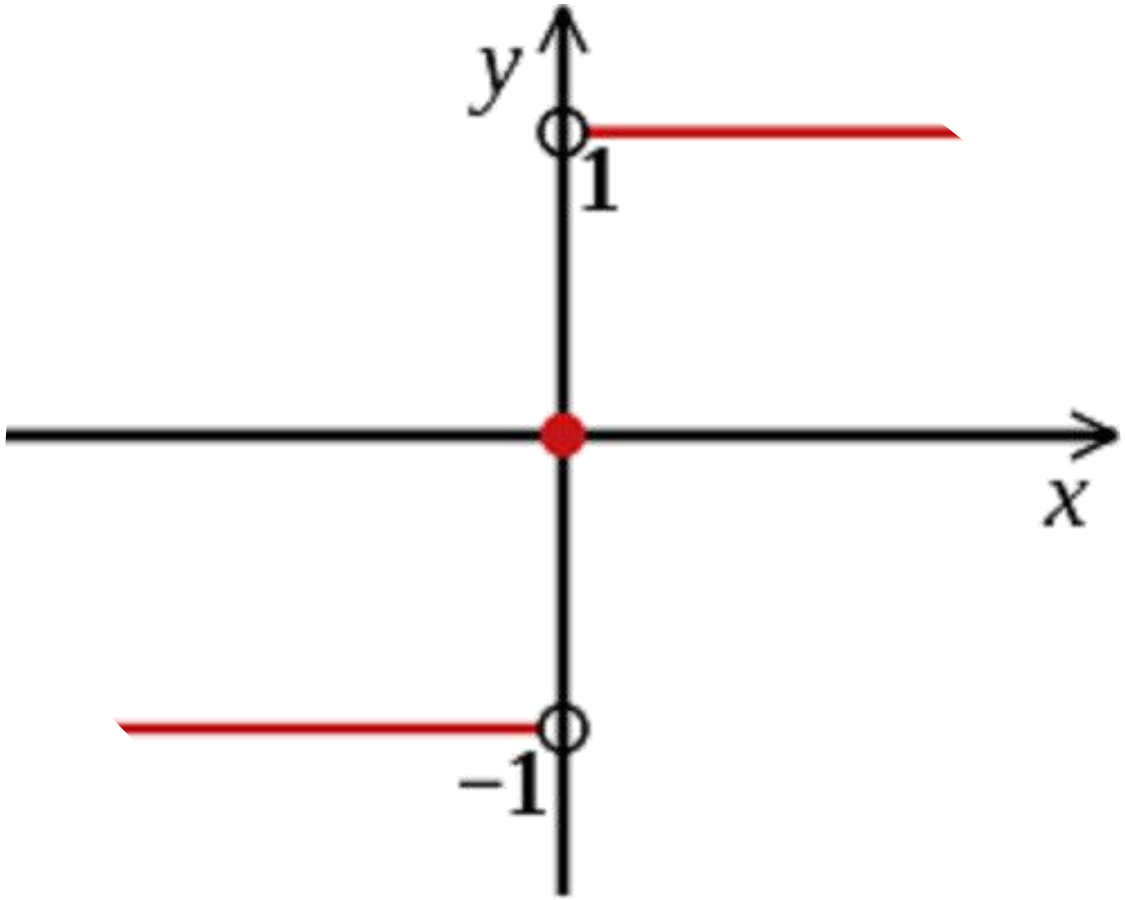
- Adaboost算法将基分类器的线性组合作为强分类器，同时给分类误差率较小的基本分类器以大的权值，给分类误差率较大的基分类器以小的权重值；构建的线性组合为：

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$$

- 最终分类器是在线性组合的基础上进行Sign函数转换：

$$G(x)=sign(f(x))=sign\left[\sum_{m=1}^{M}\alpha_m G_m(x)\right]$$

Sign函数



AdaBoost算法原理

- 最终的强学习器：$G(x)=sign(f(x))=sign\left[\sum_{m=1}^{M}\alpha_m G_m(x)\right]$
- 损失函数：$loss=\frac{1}{n}\sum_{i=1}^{n}I(G(x_i)\neq y_i)$
- 损失函数：$loss=\frac{1}{n}\sum_{i=1}^{n}I(G(x_i)\neq y_i)\leq\frac{1}{n}\sum_{i=1}^{n}e^{(-y_i f(x))}$

AdaBoost算法原理

$$loss = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i f(x_i))}$$

- 第k-1轮的强学习器：

$$f_{k-1}(x) = \sum_{j=1}^{k-1} \alpha_j G_j(x)$$

- 第k轮的强学习器：

$$f_k(x) = \sum_{j=1}^{k} \alpha_j G_j(x) \qquad f_k(x) = f_{k-1}(x) + \alpha_k G_k(x)$$

- 损失函数： $loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i (f_{m-1}(x) + \alpha_m G_m(x)))}$

AdaBoost算法原理

$$loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i(f_{m-1}(x)+\alpha_m G_m(x)))}$$

$$= \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f_{m-1}(x)} e^{(-y_i \alpha_m G_m(x))}$$

$$\xrightarrow{令\overline{w}_{mi} = e^{-y_i f_{m-1}(x)}}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \overline{w}_{mi} e^{(-y_i \alpha_m G_m(x))}$$

AdaBoost算法原理

- 使下列公式达到最小值的α_m和G_m就是AdaBoost算法的最终求解值

$$loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n}\overline{w}_{mi}e^{(-y_i\alpha_m G_m(x))}$$

- G这个分类器在训练的过程中，是为了让误差率最小，所以可以认为G越小其实就是误差率越小。

$$G_m^*(x) = \frac{1}{n}\sum_{i=1}^{n}\overline{w}_{mi}I(y_i \neq G_m(x_i)) \quad \varepsilon_m = P(G_m(x) \neq y) = \frac{1}{n}\sum_{i=1}^{n}\overline{w}_{mi}I(y_i \neq G_m(x_i))$$

- 对于α_m而言，通过求导然后令导数为零，可以得到公式（log对象可以以e为底也可以以2为底）：

$$\alpha_m^* = \frac{1}{2}\ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$$

Adaboost算法构建过程一

- 1. 假设训练数据集T={$(X_1,Y_1),(X_2,Y_2)....(X_n,Y_n)$}
- 2. 初始化训练数据权重分布

$$D_1 = (w_{11}, w_{12}, ...w_{1i}..., w_{1n}), \quad w_{1i} = \frac{1}{n}, \quad i = 1,2,...,n$$

- 3. 使用具有权值分布D_m的训练数据集学习，得到基本分类器

$$G_m(x): x \to \{-1,+1\}$$

- 4. 计算$G_m(x)$在训练集上的分类误差

$$\varepsilon_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{n} w_{mi} I(G_m(x_i) \neq y_i)$$

- 5. 计算$G_m(x)$模型的权重系数α_m：$\alpha_m = \frac{1}{2} * \log_2\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

Adaboost算法构建过程二

- 6. 权重训练数据集的权值分布

$$D_{m+1} = \left(w_{m+1,1}, w_{m+1,2}, \ldots w_{m+1,i} \ldots, w_{m+1,n}\right) \quad w_{m+1,i} = \frac{w_{m,i}}{Z_m} e^{-\alpha_m y_i \mathrm{G}_m(x_i)}$$

- 7. 这里Z_m是规范化因子(归一化) $Z_m = \sum_{i=1}^{n} w_{m,i} e^{-\alpha_m y_i \mathrm{G}_m(x_i)}$

- 8. 构建基本分类器的线性组合 $f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$

- 9. 得到最终分类器 $G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$

Adaboost算法的直观理解

- 权重训练用下列训练样本，试用AdaBoost算法学习一个强分类器

序号	1	2	3	4	5	6	7	8	9	10
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

- 初始化训练数据集的权值分布

$$D_1 = (w_{11}, w_{12}, \ldots w_{1i} \ldots, w_{1n}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\ldots, N$$

$$w_{1i} = 0.1$$

Adaboost算法的直观理解

- 对于m=1

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

- 在权值分布为D1的训练数据上，阈值v取2.5时误差率最低，故基本分类器为：

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

- G1(x)在训练数据集上的误差率 $\varepsilon_1 = P\left(G_1\left(x_i\right)\neq y_i\right)=0.3$

- 计算G1的系数 $\alpha_1=\frac{1}{2}\log_2\frac{1-\varepsilon_1}{\varepsilon_1}=0.6112$

Adaboost算法的直观理解

- 更新数据集的权值分布

$$D_{\mathrm{m+1}} = \left(w_{m+1,1}, w_{m+1,2}, \cdots w_{m+1,i} \cdots, w_{m+1,n}\right) \quad w_{m+1,i} = \frac{w_{m,i}}{Z_m} e^{-\alpha_m y_i \mathrm{G}_m(x_i)}$$

$$D_2 = (w_{21}, w_{22}, \ldots., w_{2n})$$

$$= (0.0582, 0.0582, 0.0582, 0.0582, 0.0582, 0.0582, 0.1976, 0.1976, 0.1976, 0.0582)$$

$$f_1(x) = 0.6112 G_1(x)$$

- 分类器sign(f1(x))在训练数据集上有3个误分类点

Adaboost算法的直观理解

- 对于m=2

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w2	0.0582	0.0582	0.0582	0.0582	0.0582	0.0582	0.1976	0.1976	0.1976	0.0582

- 在权值分布为D2的训练数据上，阈值v取8.5时误差率最低，故基本分类器为：

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

- G2(x)在训练数据集上的误差率

$$\varepsilon_2 = P(G_2(x_i)\neq y_i) = 0.0582 * 3 = 0.1746$$

Adaboost算法的直观理解

- 计算G_2的系数 $\alpha_2 = \frac{1}{2}\log_2\frac{1-\varepsilon_2}{\varepsilon_2} = 1.1205$

- 更新数据集的权值分布

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w3	0.0236	0.0236	0.0236	0.2218	0.2218	0.2218	0.0801	0.0801	0.0801	0.0236

$$f_2(x) = 0.6112G_1(x) + 1.1205G_2(x)$$

- 分类器sign(f2(x))在训练数据集上有3个误分类点

Adaboost算法的直观理解

- 对于m=3

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w3	0.0236	0.0236	0.0236	0.2218	0.2218	0.2218	0.0801	0.0801	0.0801	0.0236

- 在权值分布为D3的训练数据上，阈值v取5.5时误差率最低，故基本分类器为：

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

- G3(x)在训练数据集上的误差率

$$\varepsilon_3 = P(G_3(x_i)\neq y_i) = 0.0236 * 4 = 0.0944$$

Adaboost算法的直观理解

- 计算G3的系数 $\alpha_3 = \frac{1}{2}\log_2 \frac{1-\varepsilon_3}{\varepsilon_3} = 1.631$

$$f_3(x) = 0.6112G_1(x) + 1.1205G_2(x) + 1.631G_3(x)$$

- 分类器sign(f3(x))在训练数据集上有0个误分类点；结束循环。

案例列表

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
G1	1	1	1	-1	-1	-1	-1	-1	-1	-1
G_R1	1	1	1	-1	-1	-1	-1	-1	-1	-1
w2	0.0582	0.0582	0.0582	0.0582	0.0582	0.0582	0.1976	0.1976	0.1976	0.0582
G2	1	1	1	1	1	1	1	1	1	-1
G_R2	1	1	1	1	1	1	1	1	1	-1
w3	0.0236	0.0236	0.0236	0.2218	0.2218	0.2218	0.0801	0.0801	0.0801	0.0236
G3	-1	-1	-1	-1	-1	-1	1	1	1	1
G_R3	1	1	1	-1	-1	-1	1	1	1	-1

alpha	value
α_1	0.6112
α_2	1.1205
α_3	1.631

$$f_3(x) = 0.6112G_1(x) + 1.1205G_2(x) + 1.631G_3(x)$$

作业

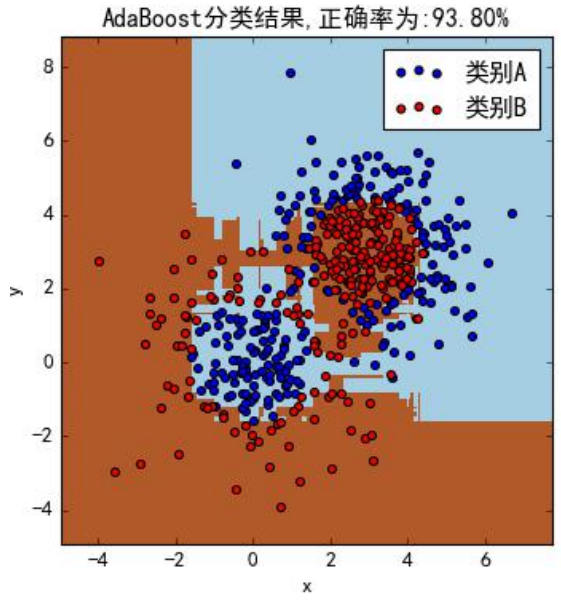
- 推导α求解过程中底数为e的情况下，该案例的最终参数情况：

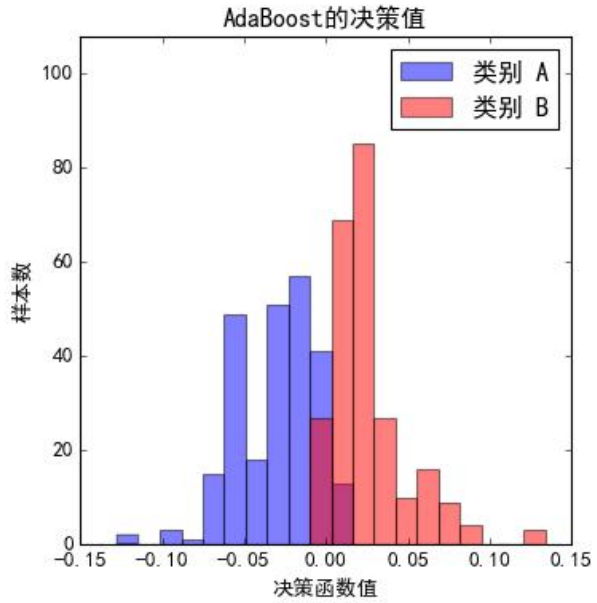
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$$\alpha_m = \frac{1}{2} * \ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$$

Adaboost案例一

- 基于python的sklearn模块中的API创建模拟数据，并进行Adaboost API进行数据分类开发测试



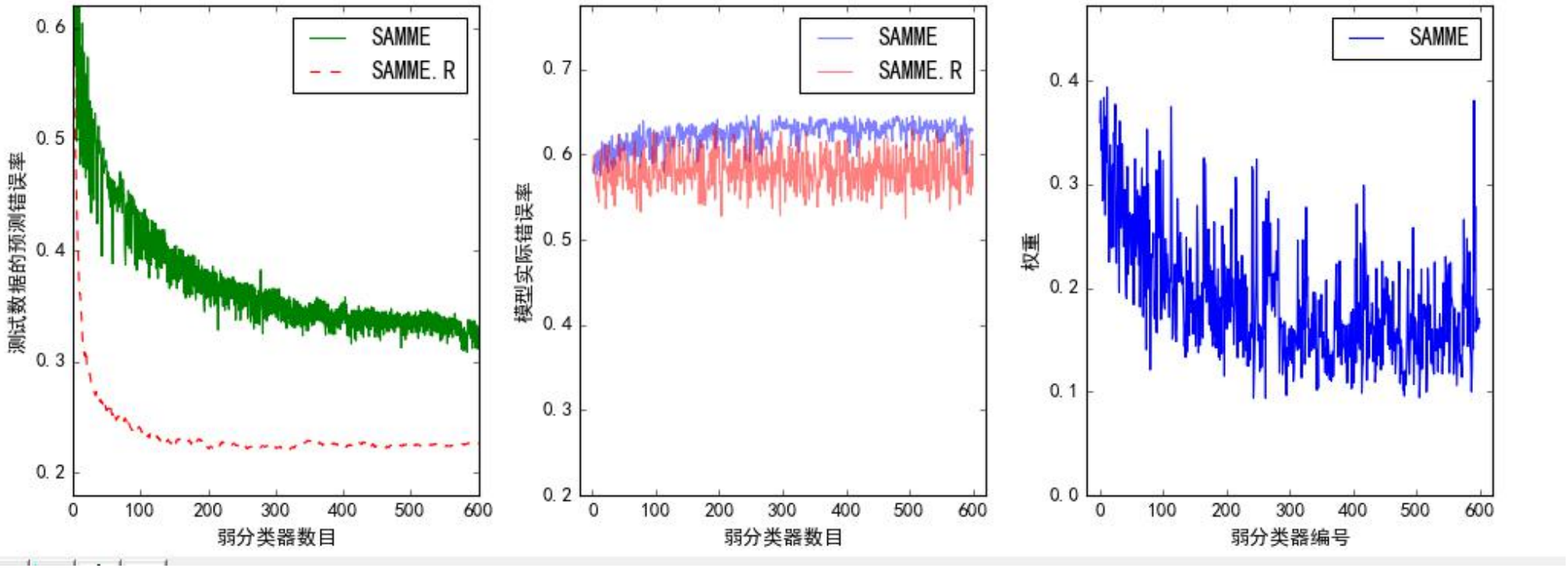


sklearn.datasets.**make_gaussian_quantiles**(*mean=None, cov=1.0, n_samples=100, n_features=2, n_classes=3, shuffle=True, random_state=None*) ¶ [source]

class sklearn.ensemble.**AdaBoostClassifier**(*base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

Adaboost案例二

- 基于python的sklearn模块中的API创建模拟数据，Adaboost API的参数algorithm的不同取值进行测试，并得到比较效果值



AdaBoost scikit-learn相关参数

参数	AdaBoostClassifier	AdaBoostRegressor
base_estimator	弱分类器对象，默认为CART分类树DecisionTreeClassifier；	弱回归器对象，默认为CART回归树DecisionTreeRegressor；
algorithm	SAMME和SAMME.R；SAMME表示构建过程中使用样本集分类效果作为弱分类器的权重；SAMME.R使用对样本集分类的预测概率大小作为弱分类器的权重。由于SAMME.R使用了连续的概率度量值，所以一般迭代比SAMME快，默认参数为SAMME.R；强调：使用SAMME.R必须要求base_estimator指定的弱分类器模型必须支持概率预测，即具有predict_proba方法。	不支持
loss	不支持	指定误差的计算方式，可选参数”linear”, “square”, “exponential”, 默认为”linear”；一般不用改动
n_estimators	最大迭代次数，值过小可能会导致欠拟合，值过大可能会导致过拟合，一般50~100比较适合，默认50	
learning_rate	指定每个弱分类器的权重缩减系数v，默认为1；一般从一个比较小的值开始进行调参；该值越小表示需要更多的弱分类器	

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x) \xrightarrow{\text{添加缩减系数}v} f(x)=\sum_{m=1}^{M} v\alpha_m G_m(x)$$

AdaBoost总结

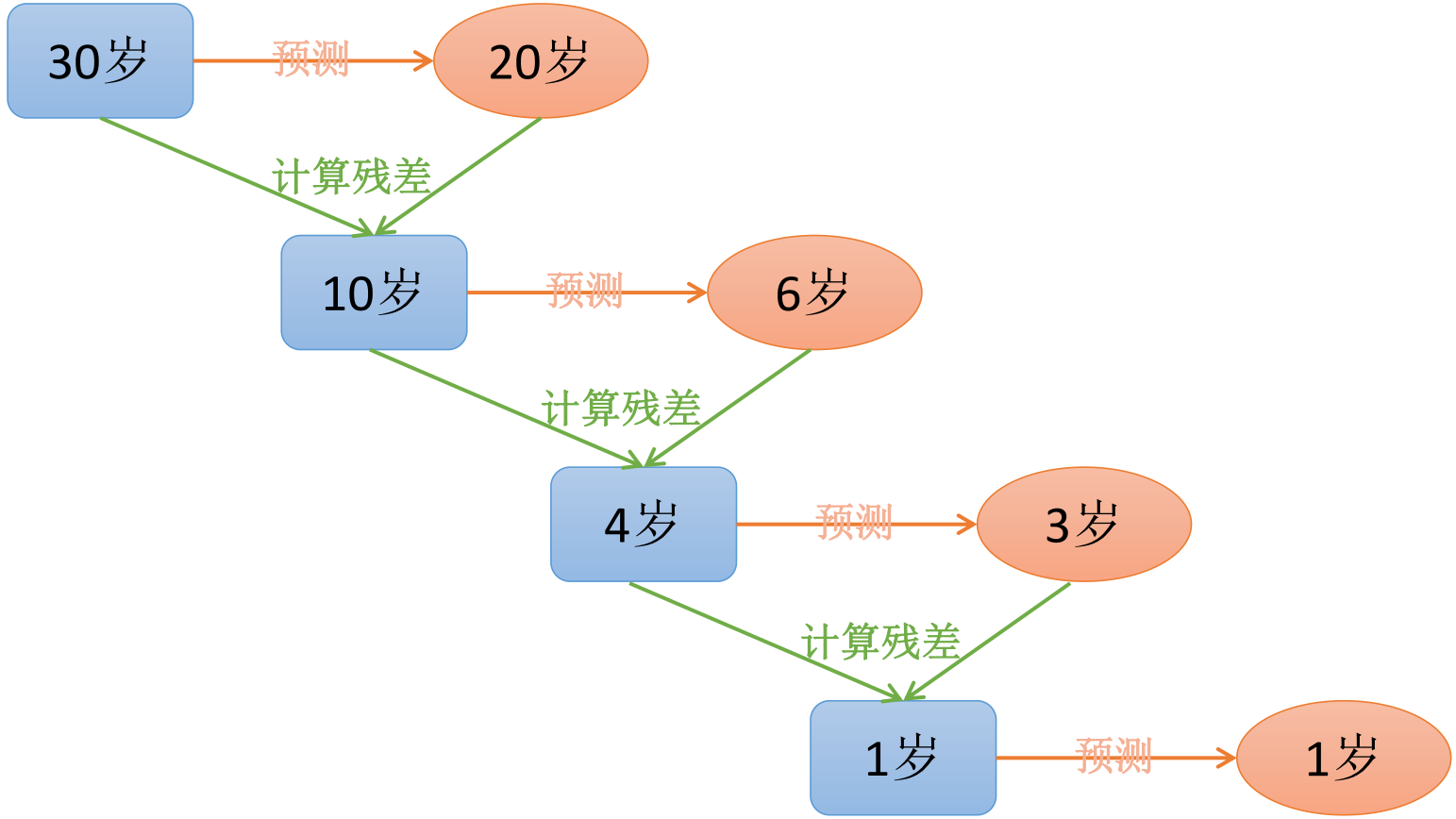
- AdaBoost的优点如下：
  - 可以处理连续值和离散值；
  - 模型的鲁棒性比较强；
  - 解释强，结构简单。
- AdaBoost的缺点如下：
  - 对异常样本敏感，异常样本可能会在迭代过程中获得较高的权重值，最终影响模型效果。

梯度提升迭代决策树GBDT

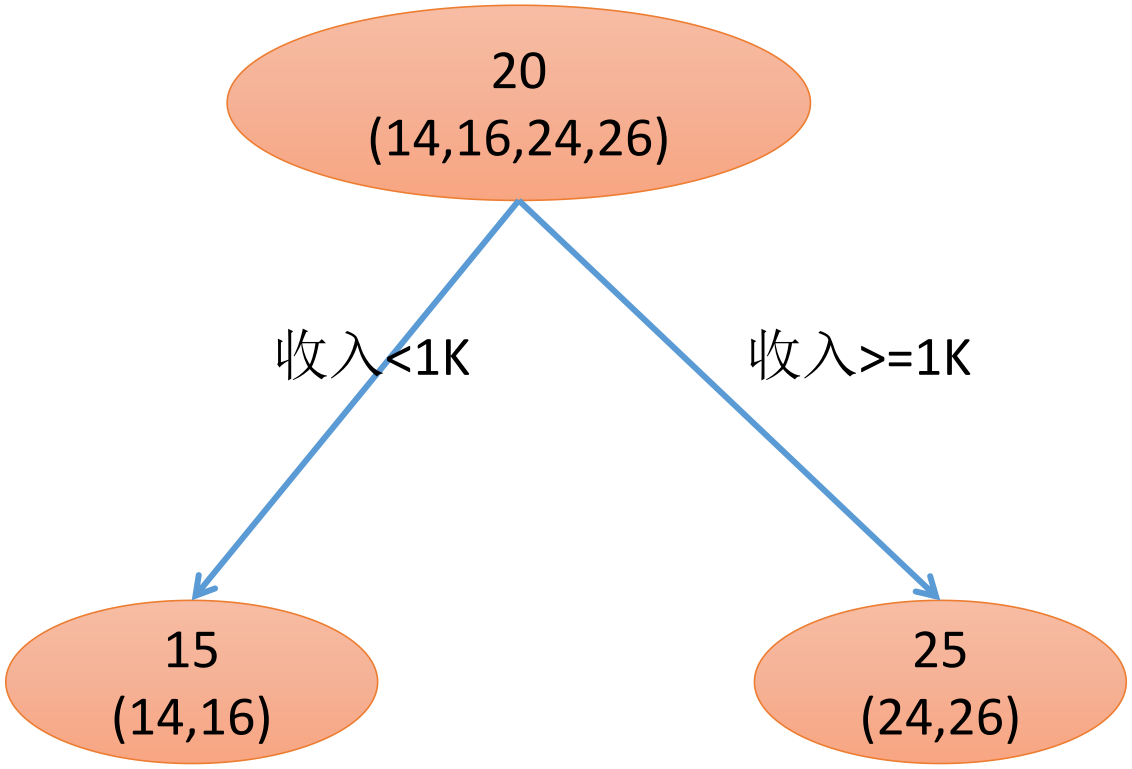
- GBDT也是Boosting算法的一种，但是和AdaBoost算法不同；区别如下：AdaBoost算法是利用前一轮的弱学习器的误差来更新样本权重值，然后一轮一轮的迭代；GBDT也是迭代，但是GBDT要求弱学习器必须是CART模型，而且GBDT在模型训练的时候，是要求模型预测的样本损失尽可能的小。
- 别名：GBT(Gradient Boosting Tree)、GTB(Gradient Tree Boosting)、GBRT(Gradient Boosting Regression Tree)、GBDT(Gradient Boosting Decison Tree)、MART(Multiple Additive Regression Tree)

$$f_{t-1}(x) \quad L(y, f_{t-1}(x)) \xrightarrow{\text{训练弱学习器:}h_t(x)} f_t(x) \quad L(y, f_{t-1}(x)+h_t(x))$$

GBDT直观理解

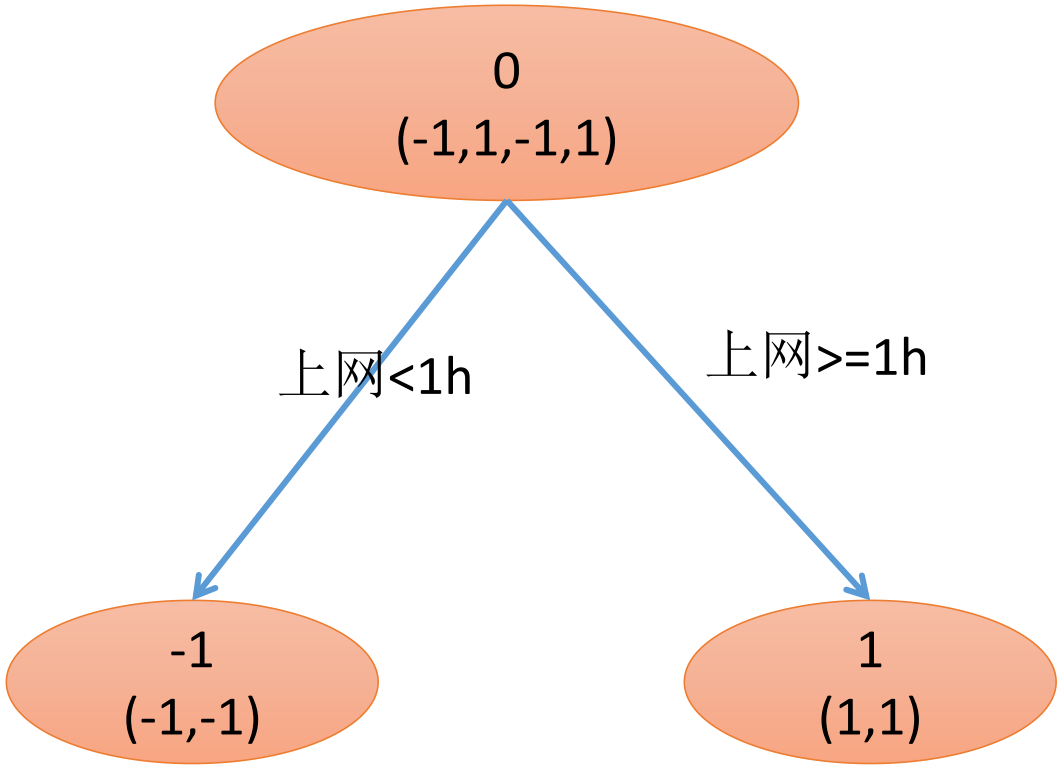


GBDT直观理解



树1

r



树2

- 当给定步长时候，给定一个步长step，在构建下一棵树的时候使用step*残差值作为输入值，这种方式可以减少过拟合的发生

梯度提升迭代决策树GBDT

- GBDT由三部分构成：DT(Regression Decistion Tree)、GB(Gradient Boosting)和Shrinkage(衰减)
- 由多棵决策树组成，所有树的结果累加起来就是最终结果
- 迭代决策树和随机森林的区别：
  - 随机森林使用抽取不同的样本构建不同的子树，也就是说第m棵树的构建和前m-1棵树的结果是没有关系的
  - 迭代决策树在构建子树的时候，使用之前子树构建结果后形成的残差作为输入数据构建下一个子树；然后最终预测的时候按照子树构建的顺序进行预测，并将预测结果相加

GBDT算法原理

- 给定输入向量X和输出变量Y组成的若干训练样本$(X_1,Y_1),(X_2,Y_2)\ldots\ldots(X_n,Y_n)$，目标是找到近似函数F(X)，使得损失函数L(Y,F(X))的损失值最小。
- L损失函数一般采用最小二乘损失函数或者绝对值损失函数

$$L(y, F(X)) = \frac{1}{2}(y - F(X))^2 \qquad L(y, F(X)) = |y - F(X)|$$

- 最优解为：$$F^*(X) = \arg\min_F L(y, F(X))$$
- 假定F(X)是一族最优基函数$f_i(X)$的加权和：

$$F(X) = \sum_{i=0}^{M} f_i(X)$$

防止每个学习器能力过强，可能导致过拟合；给定一个缩放系数v

$$F(X) = v\sum_{i=0}^{M} f_i(X)$$

GBDT算法原理

- 以贪心算法的思想扩展得到Fm(X)，求解最优f

$$\mathrm{F_m}(x) = F_{m-1}(x) + \arg\min_{\mathrm{f}} \sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(X_i\right) + f_m\left(X_i\right)\right)$$

- 以贪心法在每次选择最优基函数f时仍然困难，使用梯度下降的方法近似计算

- 给定常数函数$F_0(X)$

$$F_0\left(X\right) = \arg\min_{\mathrm{c}} \sum_{i=1}^{n} L\left(\mathrm{y}_i, c\right)$$

GBDT算法原理

- 根据梯度下降计算学习率

$$\alpha_{im} = [\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{\mathrm{F}(x)=F_{m-1}(x)}$$

- 使用数据 (x_i, α_{im})(i=1.....n)计算拟合残差找到一个CART回归树，得到第m棵树

$$c_{mj} = \arg\min_{c} \sum_{x_i \in leaf_j} L(y_i, f_{m-1}(x_i) + c) \qquad h_m(x) = \sum_{j=1}^{|leaf|_m} c_{mj} I(x \in leaf_{mj})$$

- 更新模型

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{|leaf|_m} c_{mj} I(x \in leaf_{mj}) \quad \Rightarrow \quad f(x) = f_0(x) + \sum_{m=1}^{M} \sum_{j=1}^{|leaf|_m} c_{mj} I(x \in leaf_{mj})$$

GBDT回归算法和分类算法的区别

- 两者唯一的区别就是选择不同的损失函数
- 回归算法选择的损失函数一般是均方差(最小二乘)或者绝对值误差；而在分类算法中一般的损失函数选择对数函数来表示。

$$L(\mathrm{y}, f(x)) = \frac{1}{2}(y - f(x))^2$$

$$L(\mathrm{y}, f(x)) = \ln\left(1 + e^{(-y^* f(x))}\right); y \in \{-1, +1\}$$

$$L(y, f(x)) = -\sum_{k=1}^{K} y_k \log(p_k(x)); K\text{分类中}$$

GBDT scikit-learn相关参数

参数	GradientBoostingClassifier	GradientBoostingRegressor
alpha	不支持	当使用huber或者quantile损失函数的时候，需要给定分位数的值，默认为0.9；如果噪音数据比较多，可以适当的降低该参数值
loss	给定损失函数，可选对数似然函数deviance和指数损失函数exponential；默认为deviance；不建议修改	给定损失函数，可选均方差ls、绝对损失lad、Huber损失huber、分位数损失quantile；默认ls；一般采用默认；如果噪音数据比较多，推荐huber；如果是分段预测，推荐quantile
n_estimators	最大迭代次数，值过小可能会导致欠拟合，值过大可能会导致过拟合，一般50~100比较适合，默认50	
learning_rate	指定每个弱分类器的权重缩减系数v，默认为1；一般从一个比较小的值开始进行调参；该值越小表示需要更多的弱分类器	
subsample	给定训练模型的时候，进行子采样的比例值，取值范围(0,1], 默认为1，表示不采用子采样；给值小于1表示采用部分数据进行模型训练，可以降低模型的过拟合情况；推荐[0.5,0.8]；采样采用的方式是不放回采样	
init	给定初始化的模型，可以不给定	

GBDT总结

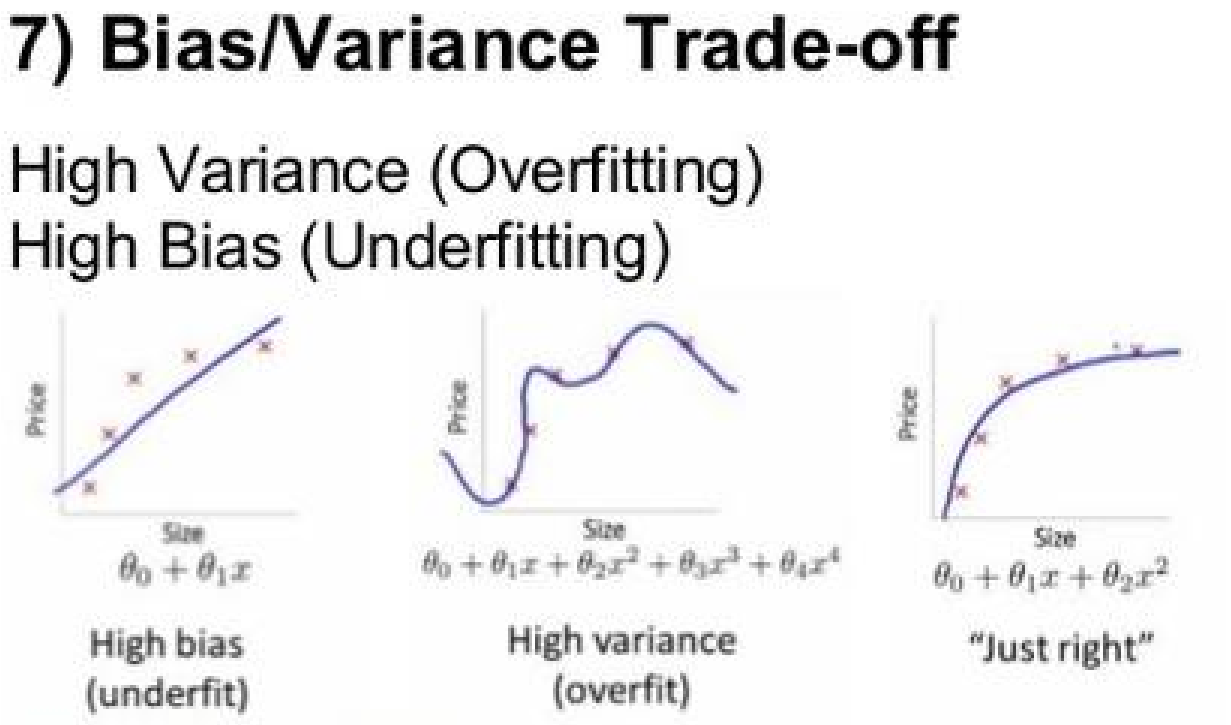
- GBDT的优点如下：
  - 可以处理连续值和离散值；
  - 在相对少的调参情况下，模型的预测效果也会不错；
  - 模型的鲁棒性比较强。
- GBDT的缺点如下：
  - 由于弱学习器之间存在关联关系，难以并行训练模型。

Bagging、Boosting的区别

- 1. 样本选择：Bagging算法是有放回的随机采样；Boosting算法是每一轮训练集不变，只是训练集中的每个样例在分类器中的权重发生变化，而权重根据上一轮的分类结果进行调整；
- 2. 样例权重：Bagging使用随机抽样，样例的权重；Boosting根据错误率不断的调整样例的权重值，错误率越大则权重越大；
- 3. 预测函数：Bagging所有预测模型的权重相等；Boosting算法对于误差小的分类器具有更大的权重。
- 4. 并行计算：Bagging算法可以并行生成各个基模型；Boosting理论上只能顺序生产，因为后一个模型需要前一个模型的结果；
- 5. Bagging是减少模型的variance(方差)；Boosting是减少模型的Bias(偏度)。
- 6. Bagging里每个分类模型都是强分类器，因为降低的是方差，方差过高需要降低是过拟合；Boosting里每个分类模型都是弱分类器，因为降低的是偏度，偏度过高是欠拟合。

Bagging、Boosting的区别

- error = Bias + Variance



Bagging、Boosting的区别

- Bagging对样本重采样，对每一轮的采样数据集都训练一个模型，最后取平均。由于样本集的相似性和使用的同种模型，因此各个模型的具有相似的bias和variance；

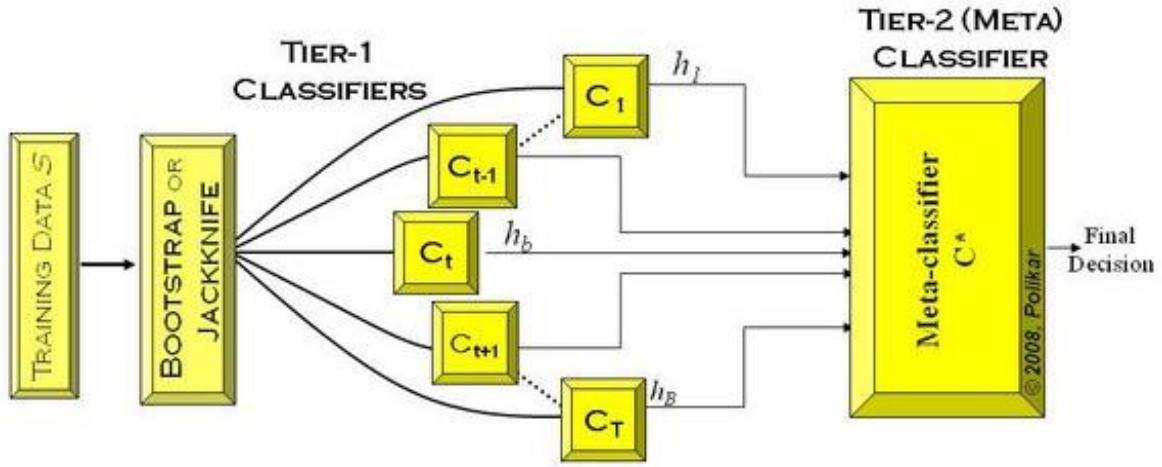
$$E\left(\frac{\sum_{i=1}^{n} X_i}{n}\right) = E(X_i)$$

$$Var\left(\frac{\sum_{n} X_i}{n}\right) = \frac{Var(X_i)}{n};$$ 模型完全独立

$$Var\left(\frac{\sum_{n} X_i}{n}\right) = Var(X_i);$$ 模型完全相同

Stacking

- Stacking是指训练一个模型用于组合(combine)其它模型(基模型/基学习器)的技术。即首先训练出多个不同的模型，然后再以之前训练的各个模型的输出作为输入来新训练一个新的模型，从而得到一个最终的模型。一般情况下使用单层的Logistic回归作为组合模型。



上海育创网络科技有限公司