# 金融科技学

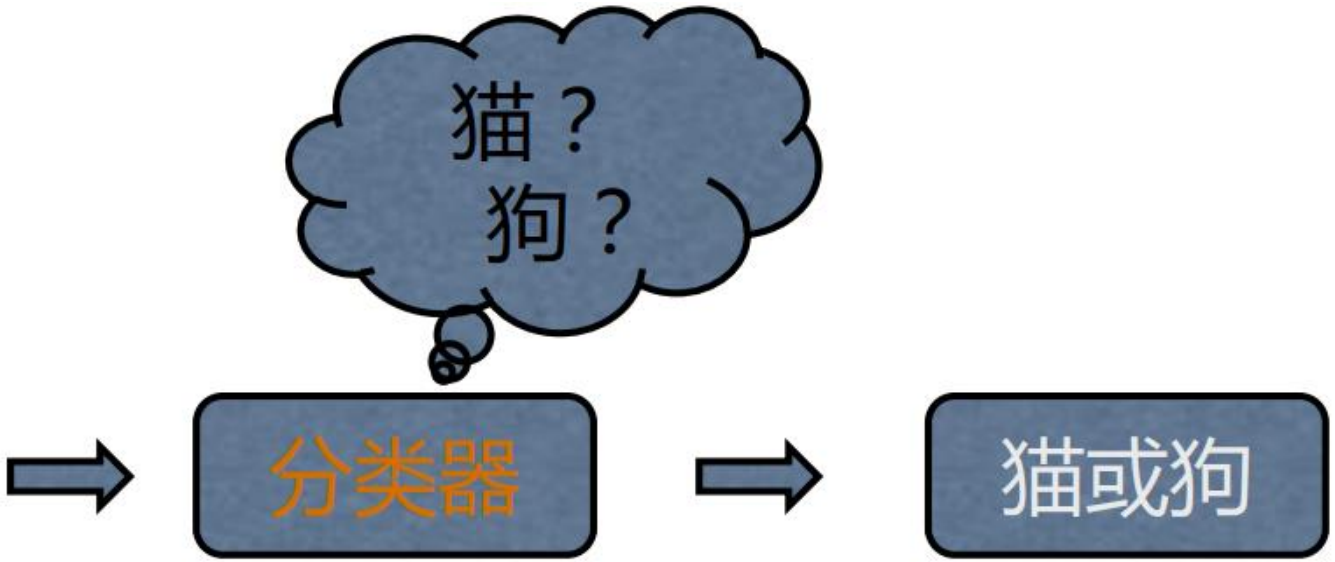
李彦

liyan_zjgsu.163.com

# Preview

- 机器学习及其发展历程
- 机器学习的类别（有监督、无监督）及典型的使用场景（回归、分类、聚类、降维）
- 机器学习中的常见问题（过拟合）及评估方法（留出、交叉验证、自助）
- 机器学习算法的性能度量（分类：混淆矩阵；回归：RMSE；聚类：MI）
- sklearn库简介
- 机器学习的经典算法

# 分类任务



# 分类学习

- **输入：**一组有标签的训练数据(也称观察和评估)，标签表明了这些数据（观察）的所属类别。
- **输出：**分类模型根据这些训练数据，训练自己的模型参数，学习出一个适合这组数据的分类器，当有新数据（非训练数据）需要进行类别判断，就可以将这组新数据作为输入送给学好的分类器进行判断。

# 分类学习-评价

- **训练集(training set)**：顾名思义用来训练模型的已标注数据，用来建立模型，发现规律。
- **测试集(testing set)**：也是已标注数据，通常做法是将标注隐藏，输送给训练好的模型，通过结果与真实标注进行对比，评估模型的学习能力。
- 训练集/测试集的划分方法：根据已有标注数据，随机选出一部分数据（70%）数据作为训练数据，余下的作为测试数据，此外还有交叉验证法，自助法用来评估分类模型。

# 分类

- 分类学习是最常见的监督学习问题
  - 二分类问题：如手机垃圾短信识别
  - 多分类问题：手写数字识别、图像分类等
- 分类学习采用不同的算法得到不同的分类模型
  - 决策树（Decision Tree）
  - 贝叶斯分类
  - KNN（K 近邻）
  - 支持向量机（SVM， Support Vector Machine）
  - 神经网络（Neural Network）
  - 集成学习（Ensemble learning）
  - ……

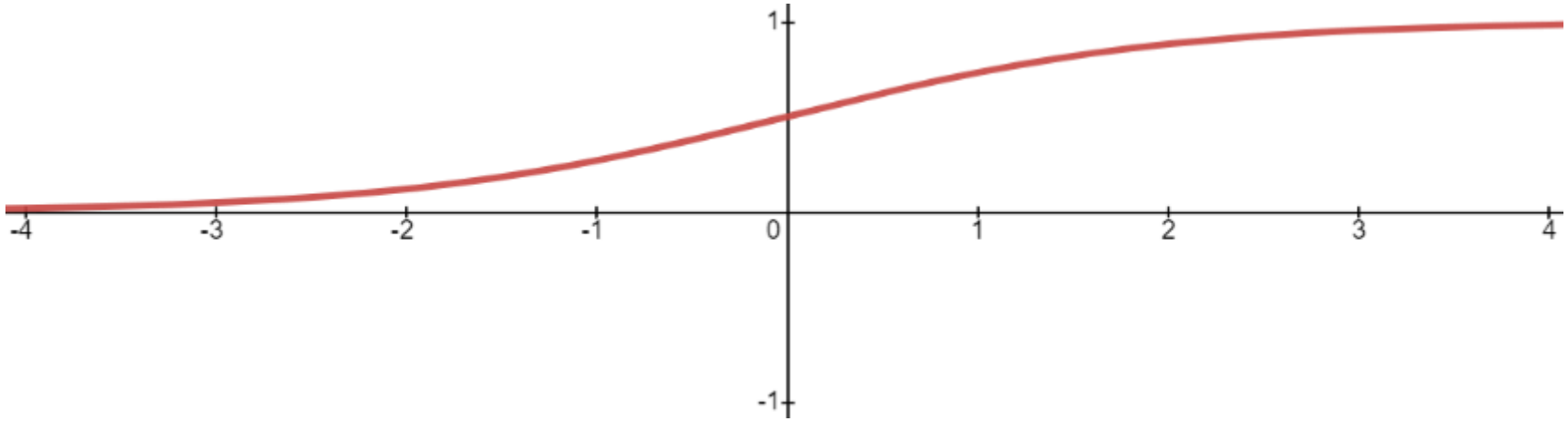
# Logistic回归算法（分类问题）

- 分类问题是当前机器学习的研究热点，它被广泛应用到各个领域，比如图像识别、垃圾邮件处理、预测天气、疾病诊断等等。
- “分类问题”的预测结果是**离散**的，它比线性回归要更加复杂

- **Logistic 回归算法**，又叫做逻辑回归算法（Logistic Regression）
- 在机器学习中，逻辑回归算法通常用来解决二元分类问题，也就是涉及两个预设类别的问题

# Logistic回归算法（分类问题）

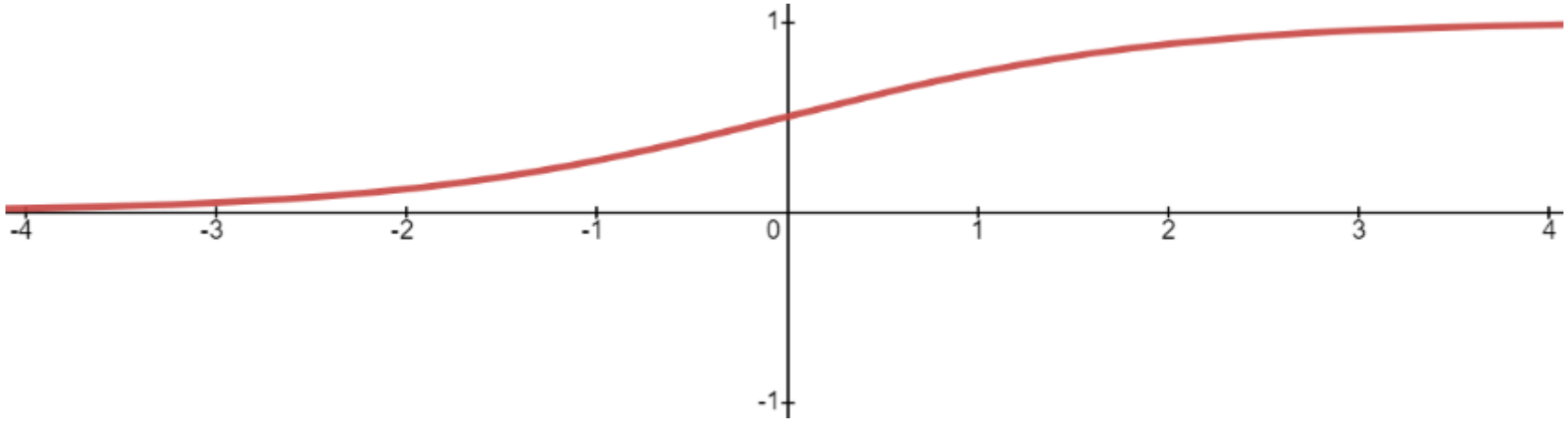
- 分类问题对输出数据的类别做判断，比如将类别预设条件分为“0”类和“1”类（“是”或“否”）
- Logistic 函数可以将一个实数映射到 (0,1) 的区间，非常适合做二元分类。




- 数学表达式 $Logistic(z) = \frac{1}{1+e^{-z}}$

# Logistic回归算法（分类问题）

- 对于 Logistic 函数而言，任何大于 0.5 的数据都会被划分到“1”类中；而小于 0.5 会被归入到“0”类。




- 如果想要 Logistic 分类器预测准确，那么 x 的取值距离 0 越远越好，这样结果值才能无限逼近于 0 或者 1。

# 代码表示

```
#以 0.5 为间隔将其分开
if (logistic函数输出的是连续值>0.5):
      return 1
else:
      return 0
```

# Logistic函数数学解析——假设函数

- 已知 Logistic 函数能够很好的拟合“离散数据”，因此可以把它看做“假设函数”，但是还需要稍稍的改变一下形式，如下所示：

$$H(x) = \frac{1}{1 + e^{-(w^T x_i + b)}}$$

- 上述公式将幂指数换成了“线性函数”表达式

# Logistic函数数学解析——损失函数

- 逻辑回归算法的损失函数：

$$L(x) = -ylogH(x) - (1-y)log(1-H(x)) \quad (1)$$

- 假设函数的值域是 (0,1) 之间的数值。如果我们把预测结果看做概率，则可以得到损失函数：

$$L(x) = -H(x_i)^{y_i}(1-H(x_i))^{1-y_i} \quad (2)$$

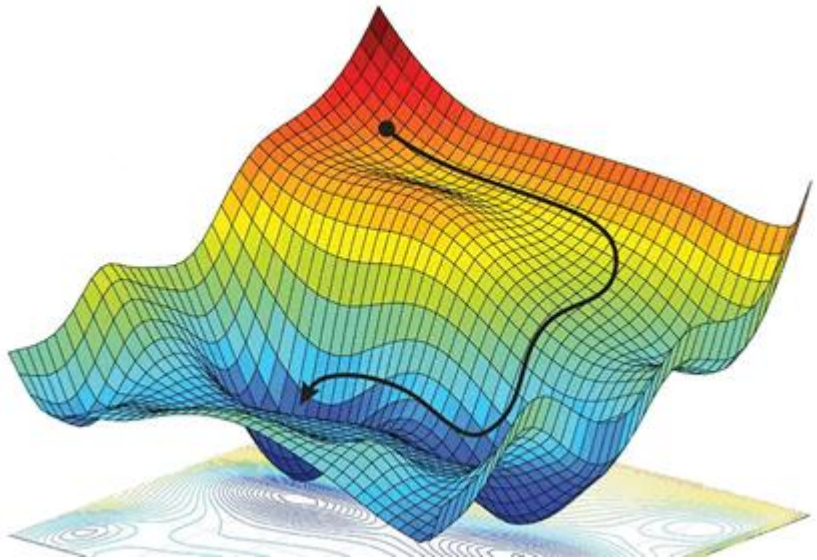
  - 上述函数由 $H(x_i)^{yi}$ 和 $(1-H(x_i))^{1-yi}$ 两部分组成，由于 y 的取值只会是 0 或 1，因此每次只有一部分输出值。
  - 当 y = 1 时，公式（2）的第二部分始终为1。如果预测正确，预测值$H(x_i)$无限接近 1，损失值则为 -1；如果预测错误，损失值为 0。
  - 预测错误的损失值比预测正确的损失值大（0 > -1），满足要求。
  - 公式（2）不能使用梯度下降（上升）等优化方法，因此取对数获得公式（1）

# 梯度下降

- 梯度是微积分学的术语，它本质上是一个向量，表示函数在某一点处的偏导数沿着特定的方向取得最大值，即函数在该点处沿着该方向**变化最快，变化率最大**。
- 梯度下降法的计算过程就是沿梯度方向求解极小值，当然也可以沿梯度上升的方向求解极大值。

  - “沿着最陡峭的坡度下山最快”

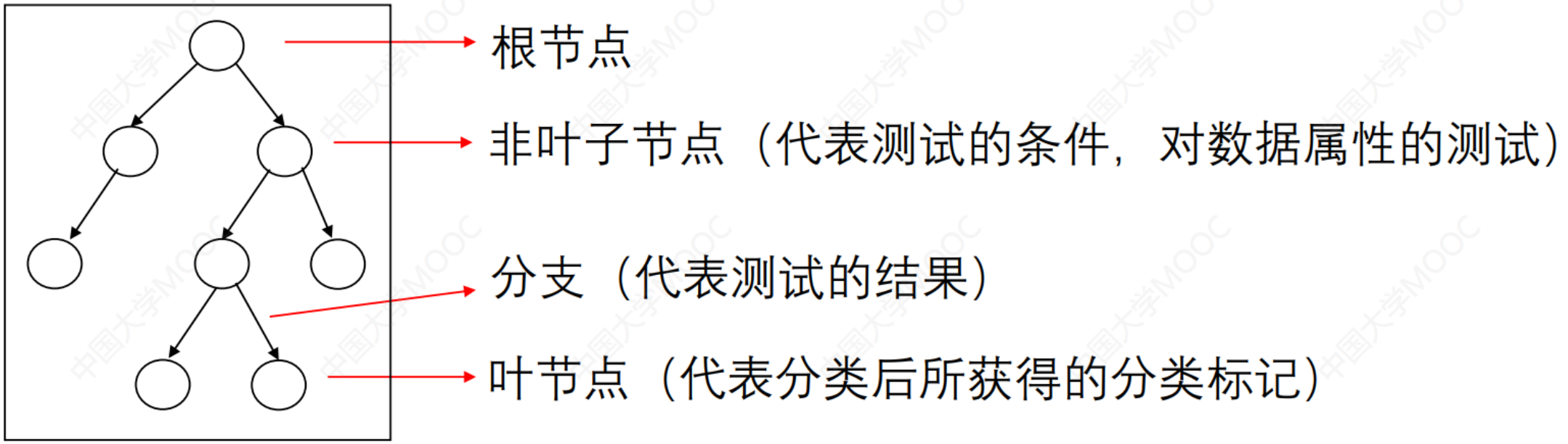
  - $w_{new} = w + \alpha \nabla_w f(w)$
  - α为步长



- 实例:
- 使用逻辑回归算法进行分类

# 什么是决策树

- 决策树，又称为判定树，是数据挖掘技术中的一种重要分类方法，它是一种以树结构来表达的预测分析模型：



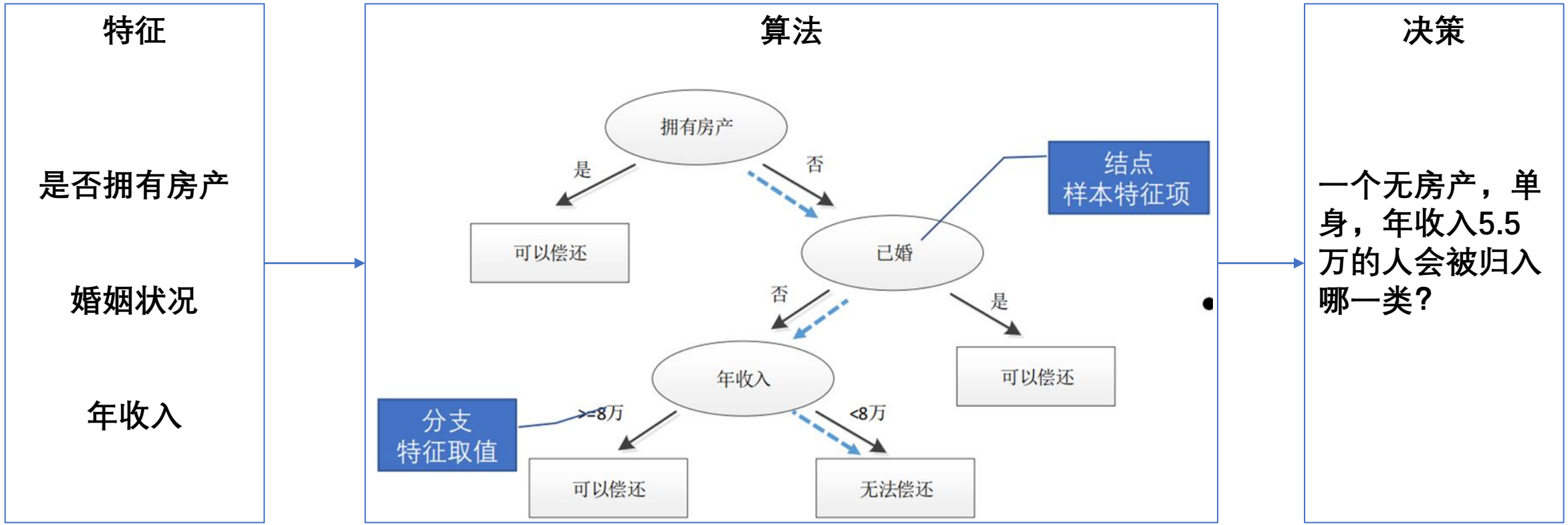
# 分类算法的应用：银行客户

- 某银行拥有客户的基本信息、是否能够偿还债务的历史记录
- 建立模型，预测新客户是否具有偿还债务的能力

| 序号 | 拥有房产（是/否） | 婚姻状况（单身、已婚、离婚） | 年收入（单位：万元） | 无法偿还债务（是/否） |
|---|---|---|---|---|
| 1 | 是 | 单身 | 12.5 | 否 |
| 2 | 否 | 已婚 | 10 | 否 |
| 3 | 否 | 单身 | 7 | 否 |
| 4 | 是 | 已婚 | 12 | 否 |
| 5 | 否 | 离婚 | 9.5 | 是 |
| … | … | … | … | … |

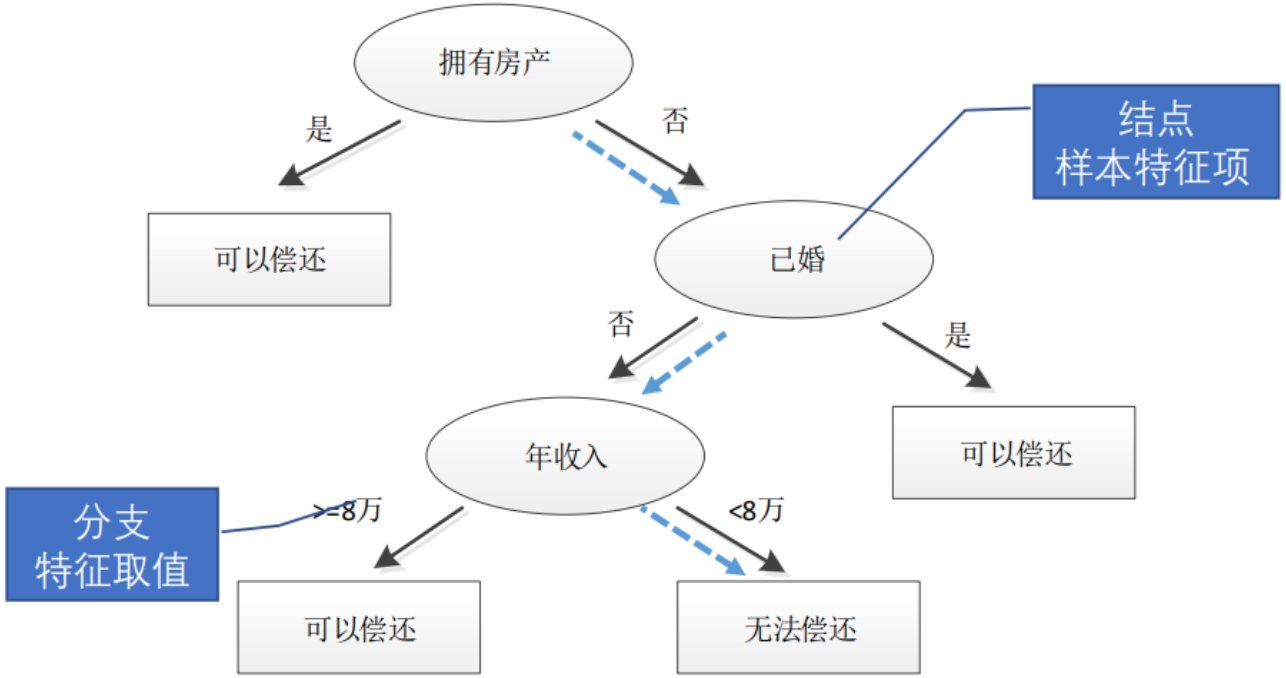
- 数据集中每条数据包括多个**特征项**（房产、婚姻和年收入）以及一个**分类标签**（是否无法偿还债务）
- 分类算法通过数据集自动学习获得**分类模型**（也称为**分类器**）
- 新客户贷款，依据客户各项特征的值，分类模型预测此客户未来是否具有偿还能力

# 智能体的决策过程




对于一个智能决策系统，它有三个重要组成部分：特征+算法+决策

# 决策树的构建




- 构建决策树的要点：
  - 结点属性测试
    - 为何选择“是否拥有房产”作为根结点？可以选择其他特征吗？
  - 特征变量
    - 有些属性值是离散的（是否拥有房产）
    - 有些属性值是连续的（年收入）
  - ……

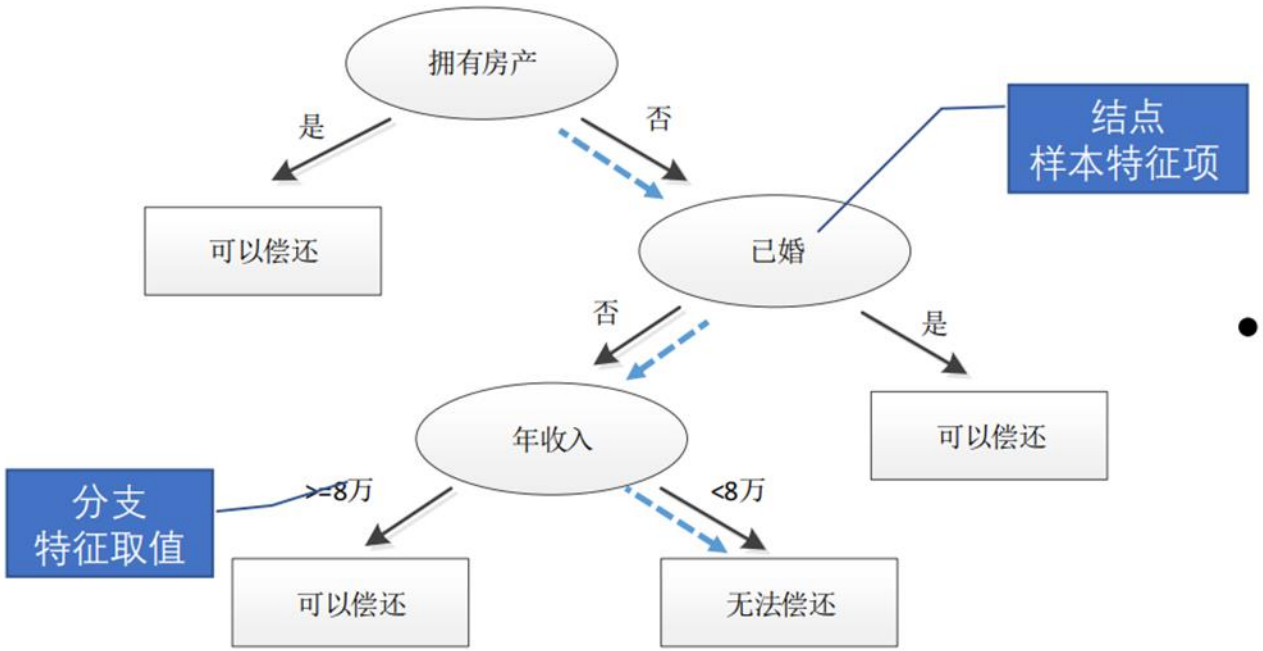
# Q：

- How to construct the decision tree?

- Feature selection
- Construction termination

# 决策树的构建

- 像人类一层一层做决策一样，决策树也需要从根结点到叶结点一层一层构建：




- 1、在根结点上，遍历所有可能的特征以及某一特征取值下所有可能的子结点，取划分后子结点**纯度之和最高**的特征，进行结点分裂
- 2、对子结点**递归**调用步骤1，直至分裂完成
- 3、生成决策树

# 决策树的构建

- 随着决策树构建过程的不断进行，我们希望决策树的分支结点所包含的样本越来越归属于同一类别，即结点的“不纯度”(impurity) 越来越低。
- 因此，为了确定按某个特征划分的效果，我们需要**比较划分前（父结点）和划分后（所有子结点） 不纯度的降低程度**，降低越多，划分的效果就越好。

- *如何度量“纯度”?*
- （*考虑只有“正类与负类”的情况*）
  - 某个分支结点下所有样本都属于同一个类别，纯度达到最高值。
  - 某个分支结点下样本所属的类别一半是正类一半是负类，此时，纯度取得最低值。
  - 纯度和类在子集中的占比有关，它并不在乎该类究竟是正类还是负类。比如，某个分支下不管是正类占比 60% 还是负类占比 60%，其纯度的度量值都是一样的。

# 如何定义纯度的变化？——信息增益

- 划分数据集的大原则是：将无序数据变得更加有序，但是各种方法都有各自的优缺点。
- 信息论是量化处理信息的分支科学，在划分数据集前后信息发生的变化称为**信息增益**，获得**信息增益最高的特征就是最好的选择**。
- 如何计算信息增益？
- ——集合信息的度量方式称为**香农熵**，或者简称熵。

# 信息熵

- 信息熵这一概念由**克劳德·香农**于1948 年提出。香农是美国著名的数学家、信息论创始人，他提出的“信息熵”的概念，为信息论和数字通信奠定了基础。

- 什么是“信息”?
- 一篇 10 万字的论文包含多少信息量?

- 信息熵用来解决对信息的量化问题。

# 信息熵

- ***信息熵是用于衡量不确定性的指标，也就是离散随机事件出现的概率，简单地说“情况越混乱，信息熵就越大，反之则越小”。***

  设$X$是一个取有限个值的离散随机变量，其概率分布为:

  $$P(X = x_i) = p_i, i = 1, 2, \cdots, n$$

  则随机变量$X$的熵定义为:

  $$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

  通常，上式中的对数以2为底或以$e$为底（自然对数）

- 在决策树分类算法中，我们可以按各个类别的占比（占比越高，该类别纯度越高）来理解，其中 n 表示类别数目，而 $P_i$ 表示类别 i 在子集中的占比。

# 信息熵

- 由定义可知，熵只依赖于X的分布，而与X的取值无关，所以也可将X的熵记作H(p)，即:

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i$$

- 熵越大，随机变量的不确定性就越大。

- 从熵的定义可以验证：$0 \leq H(p) \leq \log n$
  - 考虑如下两种情况:
  - p(X=xi)=1，i=1
  - p(X=xi)=1/n，i=1, 2, …, n

# 信息熵

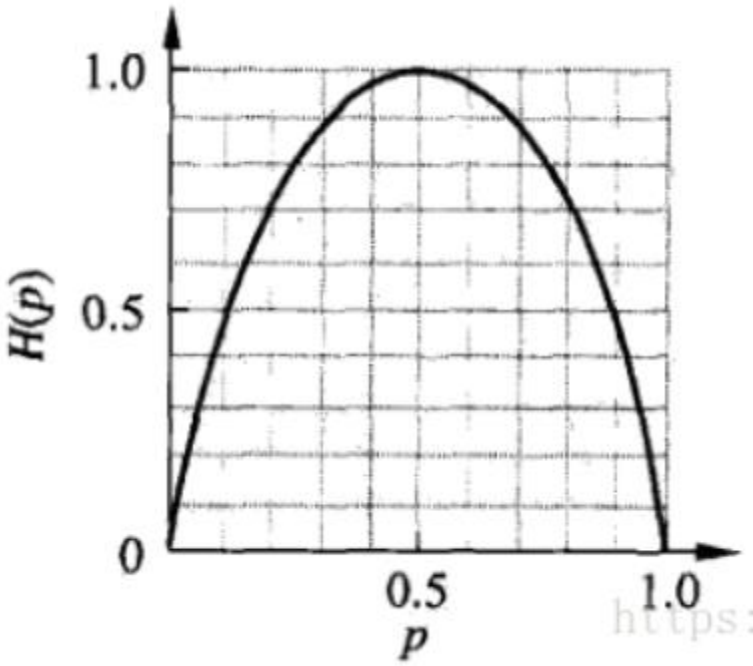
- 当随机变量仅有两个取值（如0和1时），其概率分布为

$$P(X=1)=p, \quad P(X=0)=1-p, \quad 0 \le p \le 1$$

- 其熵为：

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

- 此时，熵H(p)随概率p变化的曲线如图所示（单位为比特）：
- 当p = 0 或 p = 1 时，H(p)=0，随机变量完全没有不确定性；
- 当p = 0.5 时，H(p)=1，熵取值最大，随机变量的不确定性最大。



# 条件熵

- 条件熵H(Y|X)表示在已知随机变量X的情况下随机变量Y的不确定性。
- H(Y|X)定义为X给定条件下Y的条件概率分布的熵对X的数学期望：

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

其中，$p_i = P(X = x_i), i = 1, 2, \cdots, n$。

# 信息增益

- 信息增益表示**得知特征X的信息而使得类Y的信息不确定性减少的程度**。
- 特征A对训练数据集D的信息增益g(D, A)，定义为集合D的信息熵H(D)与特征A给定条件下D的条件熵H(D|A)之差，即

$$g(D, A) = H(D) – H(D|A)$$

- 一般地，熵H(D)与条件熵H(D|A)之差称为互信息(mutual information)。
- 决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

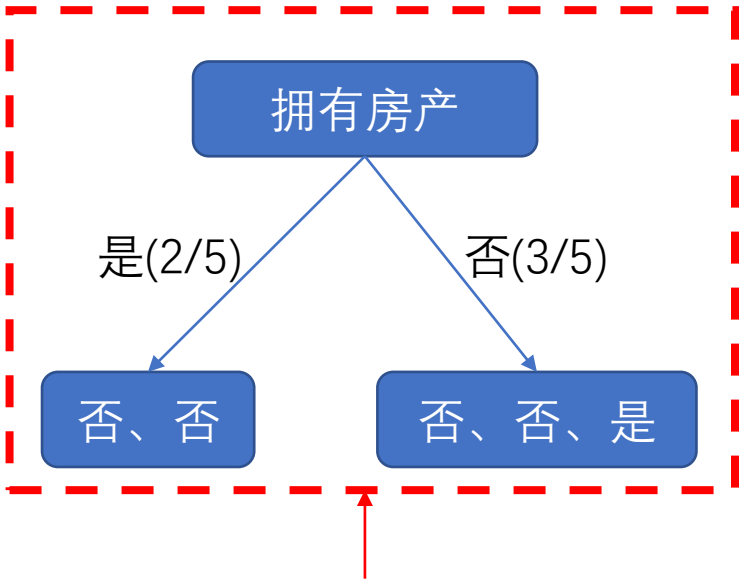
# 例1：计算信息增益

| 序号 | 拥有房产（是/否） | 婚姻状况（单身、已婚、离婚） | 年收入（单位：万元） | 无法偿还债务（是/否） |
|---|---|---|---|---|
| 1 | 是 | 单身 | 12.5 | 否 |
| 2 | 否 | 已婚 | 10 | 否 |
| 3 | 否 | 单身 | 7 | 否 |
| 4 | 是 | 已婚 | 12 | 否 |
| 5 | 否 | 离婚 | 9.5 | 是 |
| … | … | … | … | … |

- 最终分类结果只有两类，即可偿债和不可偿债。根据表中的数据统计可知，在5个数据中，4个数据的结果为不可偿债，1个数据的结果为可偿债。所以数据集D的熵H(D)为：
- **H(D)=-(1/5)*ln(1/5)-(4/5)*ln(4/5)=0.5**

# 例1：计算信息增益

| 序号 | 拥有房产（是/否） | 婚姻状况（单身、已婚、离婚） | 年收入（单位：万元） | 无法偿还债务（是/否） |
|---|---|---|---|---|
| 1 | 是 | 单身 | 12.5 | 否 |
| 2 | 否 | 已婚 | 10 | 否 |
| 3 | 否 | 单身 | 7 | 否 |
| 4 | 是 | 已婚 | 12 | 否 |
| 5 | 否 | 离婚 | 9.5 | 是 |
| … | … | … | … | … |




- 选择*是否拥有房产*作为根结点，则条件熵**H(D|A)=-(2/5)*1*ln(1)-(3/5)*[(2/3)*ln(2/3)+(1/3)*ln(1/3)]=0.382**
- 同理，选择*婚姻状况*作为根结点，则条件熵H(D|A)=0
- 由于信息增益g(D, A)=H(D)-H(D|A)越大越好，因此选择*婚姻状况*作为根结点？
  - 样本不均衡，只有5个数据

# 信息增益的局限性

- 信息增益的局限：信息增益偏向**取值较多的特征**。
- 原因：当特征的取值较多时，根据此特征划分更容易得到纯度更高的子集。
- 比如：有一个特征可以将训练集的每一个样本都当成一个分支，若有n个样本，该特征就将树分成n叉树，此时划分后的熵为0，因此信息增益最大。

# 信息增益率

- 使用**信息增益率**可以对这个问题进行校正，这是特征选择的另一个标准。

- 信息增益率=信息增益/训练数据集的经验熵

$$g_R(D, A) = g(D, A)/H_A(D)$$

$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}$$

注意H(D)与$H_A(D)$的差别!

# 例2：计算信息增益率

| 序号 | 拥有房产（是/否） | 婚姻状况（单身、已婚、离婚） | 年收入（单位：万元） | 无法偿还债务（是/否） |
|---|---|---|---|---|
| 1 | 是 | 单身 | 12.5 | 否 |
| 2 | 否 | 已婚 | 10 | 否 |
| 3 | 否 | 单身 | 7 | 否 |
| 4 | 是 | 已婚 | 12 | 否 |
| 5 | 否 | 离婚 | 9.5 | 是 |
| … | … | … | … | … |

- 最终分类结果只有两类，即可偿债和不可偿债。根据表中的数据统计可知，在5个数据中，4个数据的结果为不可偿债，1个数据的结果为可偿债。所以数据集D的熵H(D) = 0.5
- 选择*是否拥有房产*作为根结点，则条件熵 H(D|A) = 0.382
- 对应的信息增益率 g(D, A) = [H(D)-H(D|A)]/$H_A(D)$ = (0.5-0.382)/[-(2/5)ln(2/5)-(3/5)ln(3/5)] = 0.175
- H(D)以类值作为随机变量，$H_A(D)$以特征取值作为随机变量

# 信息增益小结

- 决策树学习应用**信息增益准则**选择特征。给定训练数据集D和特征A
  - 经验熵H(D) 表示对数据集D 进行分类的不确定性。
  - 而经验条件熵H(D|A) 表示在特征A 给定的条件下对数据集D 进行分类的不确定性。
  - 那么它们的差，即信息增益，就表示由于特征A 而使得对数据集D 的分类的不确定性减少的程度。
  - 显然，对于数据集D而言，**信息增益依赖于特征**，不同的特征往往具有不同的信息增益，**信息增益大的特征具有更强的分类能力**。

# 决策树算法

- 大多数决策树学习算法采用**自上而下的贪婪搜索遍历**可能的决策树空间

- **ID3算法**是用来构造决策树的常用算法，该方法使用**信息增益**选择特征。
- **C4.5算法**由J.Ross Quinlan 在ID3的基础上提出的。用**信息增益率**来选择特征，克服了信息增益偏向取值较多的特征的不足。

- 信息增益：$g(D, A) = H(D) - H(D|A)$

- 信息增益率：$g_R(D, A) = g(D, A)/H_A(D)$

# C4.5算法

- 信息增益率**对可取值数目较少的特征有所偏好**：当测试属性在当前结点中几乎都取相同的属性值时，会导致增益率无定义或者非常大。（即$H_A(D)=0$）

- 为了避免此种情况，C4.5算法并不是直接选择信息增益率最大的特征，而是使用了一个**启发式方法**：先计算所有特征的信息增益及平均值，然后仅对信息增益高于平均值的特征应用信息增益率度量。

# 平均增益

- 为了克服信息增益和增益率度量的问题，平均增益(AverGain)度量[Wang & Jiang, 2007]被提出。**平均增益度量**用特征取值的个数替换结点分裂信息，不仅惩罚了取值较多的特征，还避免了增益率度量的实际问题。
- 具体的度量公式如下：

$$AverGain = \frac{\Delta_{Info}}{k}$$

- 其中，k是特征取值的个数。

# 基尼系数

- 数据集的纯度还可以用基尼系数来衡量，**CART 算法**采用基尼系数选取最优划分特征：

$$\begin{aligned}\mathrm{Gini}(X) &= \sum_{x \in X} p(x)\,(1 - p(x)) \\ &= 1 - \sum_{x \in X} p(x)^2\end{aligned}$$

- 基尼系数反映了从数据集中随机抽取两个样本，其类别标记不一致的概率。
- 结点分裂后的基尼系数为各个子结点的基尼系数以分支概率为权重进行加和：

$$\begin{aligned}\mathrm{Gini}(X, A) &= \sum_{a \in A} p(a)\,\mathrm{Gini}(X|A = a) \\ &= \sum_{a \in A} p(a)\,(1 - \sum_{x \in X} p(x|a)^2)\end{aligned}$$

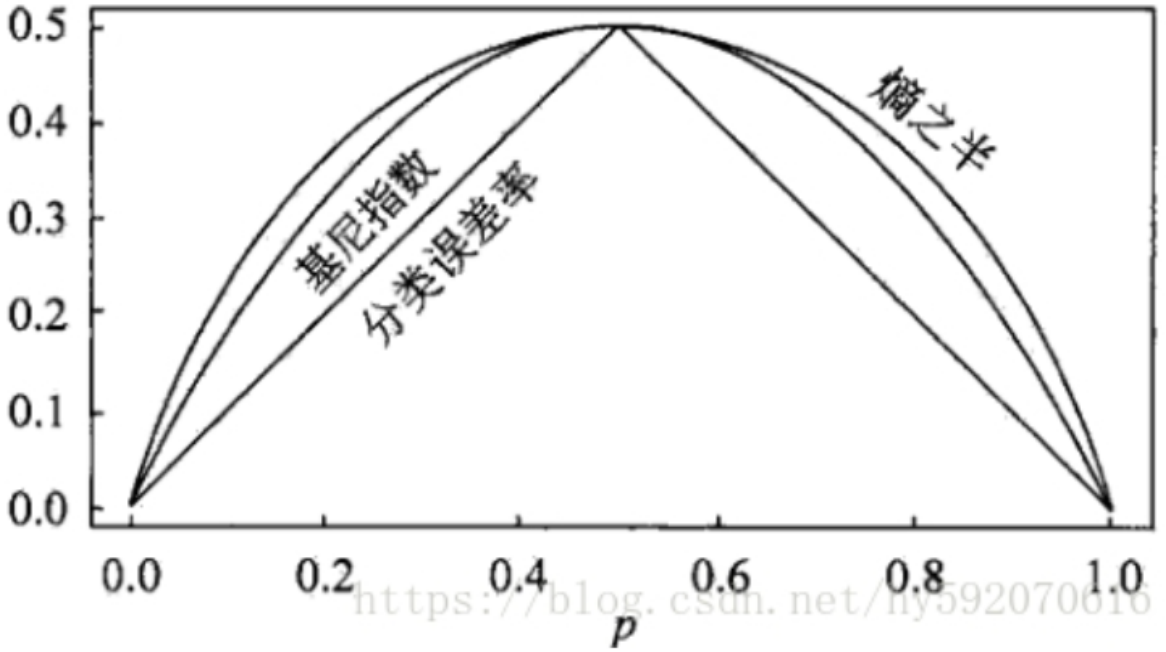
# 基尼系数

- CART算法在候选特征集合A中，选择那个使得**结点分裂后基尼系数最小**的特征作为最优划分特征。

- 下图展示了二分类问题中基尼系数、熵之半和分类误差率的关系

- 熵（Entropy）之半：-[pln(p)+(1-p)ln(1-p)]/2
- 基尼指数（Gini）：2p(1-p)
- 分类误差率:

$$error(p) = \begin{cases} p, & p < 0.5 \\ 1-p, & p \geq 0.5 \end{cases}$$



# CART算法

- CART决策树又称**分类回归树**，当数据集的因变量为连续性数值时，该树算法就是一个回归树，可以用叶结点的均值作为预测值；当数据集的因变量为离散型数值时，该树算法就是一个分类树，可以很好的解决分类问题。
- 当CART是**分类树**时，采用**GINI Index**作为结点分裂的依据；当CART是**回归树**时，采用**MSE(均方误差)**作为结点分裂的依据。
- **CART算法**采用的是一种二分循环分割的方法，每次都把当前样本集划分为两个子样本集，使生成的决策树的结点均有两个分支，显然，这样就构造了一个**二叉树**。如果特征有多于两个取值，在分裂时会对特征取值进行组合，选择最佳的两个组合分支。

# 小结

- **Gini 指数 vs 熵**
- 这两个度量都可以表示数据的不确定性、不纯度。有什么区别?
  - Gini 指数的计算不需要对数运算，更加高效;
  - Gini 指数更偏向于连续属性，熵更偏向于离散属性。

| 算法 | ID3 | C4.5 | CART |
| --- | --- | --- | --- |
| 结点分裂度量 | 信息增益 | 信息增益率 | Gini指数 |

# 决策树的实现

- ~~最优划分特征的选择（结点分裂度量）~~
- 处理特征连续值
- 处理特征缺失值
- 叶子结点的判定（构建终止）
- 怎样解决过拟合

# 处理特征连续值

- 实际的任务中常会遇到连续特征，对于全部为连续特征的样本来说，我们一般使用回归决策树（CART）来处理。
- 由于连续特征的可取值数目不再有限，因此，不能直接根据连续特征的可取值来对结点进行划分。此时，应对连续特征进行**离散化。**
- 最简单的策略是采用**二分法**对连续特征进行处理。

# 处理特征连续值

- 有监督离散化常用的有**二分法**(bi-partition)，即将连续取值的特征按选定的阈值分割成布尔特征（二值特征）：
  - 按照某个连续特征A排列训练样本，找出类标记不同的相邻样本
  - 计算类标记不同的相邻样本的特征A的取值的中间值，产生一组候选阈值
  - 计算每个候选阈值对应的信息增益，选择具有最大信息增益的阈值来离散化连续特征A

$$
\begin{aligned}
Gain(D,a) &= \max_{t\in T_a} Gain(D,a,t) \\
&= \max_{t\in T_a}\left[Ent(D) - \sum_{\lambda\in\{-,+\}} \frac{D_t^{\lambda}}{D} Ent(D_t^{\lambda})\right]
\end{aligned}
$$

# 处理特征缺失值

- 现实任务中常会遇到不完整样本，即样本的某些值缺失，尤其是在特征数目较多的情况下，往往会有大量样本出现缺失值。
- 面对缺失值，决策树学习会面临两个方面的问题：
  - 如何计算含缺失值的结点分裂度量、并进行最优特征的选择？
  - 选择好最优特征后，若样本在该特征上的值缺失，如何对样本进行划分？

# 处理特征缺失值

- 处理缺失值，通常有两个办法：
  - **放弃**含缺失值的样本，仅使用无缺失值的样本来进行学习，这种方法造成了数据信息的浪费。
  - 根据此特征取值已知的其他样本，来**估计**这个缺失的特征取值。
    - 赋给它当前结点所有样本中该特征最常见的值
    - 赋给它当前结点同类样本中该特征最常见的值
    - 为含缺失值的特征的每个可能值赋予一个概率，而不是简单地将最常见的值赋给它（即让该样本以不同的概率划分到不同的子结点中去）

# 叶结点的判定

- 上述问题都是围绕树构建的过程来展开的，那到底什么时候才***终止***树的生长？也就是递归过程什么时候返回且当前结点会被判定为叶结点？
- 如果我们暂且不考虑树的规模过大而导致的过拟合问题，在决策树学习基本算法中，有三种情形会被判定为叶子结点：
  - 当前结点中的样本集合为空，即空叶子
  - 当前结点中的所有样本归属于同一类别，即纯叶子
  - 特征用完：1. 分类任务完成 2. 分类没有完成，比如，在判断为“是”的结点集合中，有 8 个正类 3 个负类，此时我们将采用**占比最大的类**作为当前结点的归属类。

# 决策树

- 决策树的生成由两个阶段组成
  - 树构建
    - 开始时，所有的训练样本都在根结点
    - 递归地通过选定的特征来划分样本 （必须是离散值）
  - 树**剪枝**
    - 许多分枝反映的是训练数据中的噪声和孤立点，树剪枝试图检测和剪去这种分枝
- 决策树的使用：
  - 对未知样本进行分类：通过将样本的特征取值与决策树相比较

# 怎样解决过拟合问题——剪枝

- 剪枝是决策树学习算法处理“**过拟合**”的主要手段。
  - 在决策树学习中，为了尽可能正确分类训练样本，结点分裂过程不断重复，有时会造成决策树分支过多，这时就可能因为对训练样本学习得“太好”了，以致于把训练集自身的一些特点当作所有数据都具有的一般性质而导致***过拟合***。因此，可通过主动去掉一些分支来降低过拟合的风险。

- 决策树剪枝的基本策略有***预剪枝***和***后剪枝***：
  - 预剪枝：在结点分裂前就进行剪枝判断，如果判断结果是需要剪枝，则不进行分裂。
  - 后剪枝：允许树过度拟合训练数据，然后对树进行修剪。

# 剪枝

- 要点：判断什么时候需要进行剪枝操作
- ——如果剪枝后决策树模型在测试集验证上得到有效的提升，就判断其需要剪枝，否则不需要。

# 预剪枝

- **预剪枝**是指在决策树生成过程中，在结点分裂前先进行估计，若当前结点的划分不能带来决策树性能的提升，则停止分裂并将当前结点标记为叶结点。

- 常用方法：
  - 定义一个深度，当决策树达到该深度时就停止决策树的生长
  - 定义一个阈值，当结点的样本个数小于阈值时就停止决策树的生长
  - 定义一个阈值，当叶子结点的个数大于阈值时就停止决策树的生长

# 后剪枝

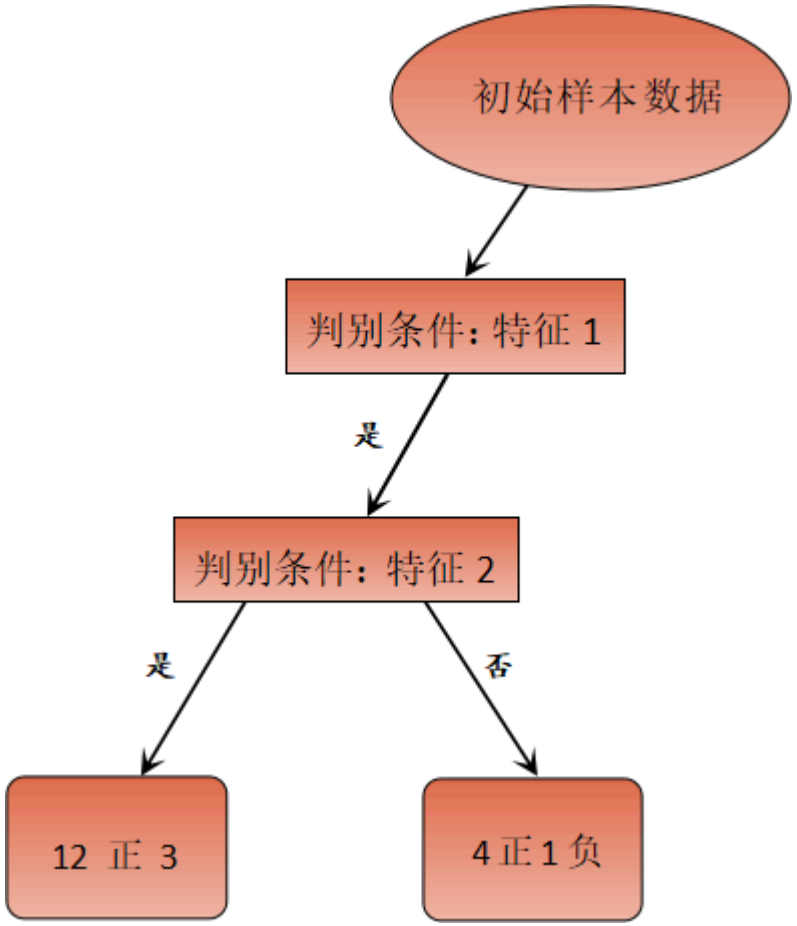
- 后剪枝则是先从训练集生成一棵完整的决策树，然后**自下而上**地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来决策树泛化性能的提升，则将该子树替换为叶结点。
- 相比于预剪枝，***后剪枝更常用***，因为在预剪枝中精确地估计何时停止树的生长很困难。

- 后剪枝的步骤：构建完全生长的决策树->判断是否删除分支->子树替换

# 例

- 某个样本数据集有两个类别（正类与负类），2 个特征，现在我们对 20 个样本进行分类。
- 首先，在应用全部特征的情况下对样本进行分类。如图所示：

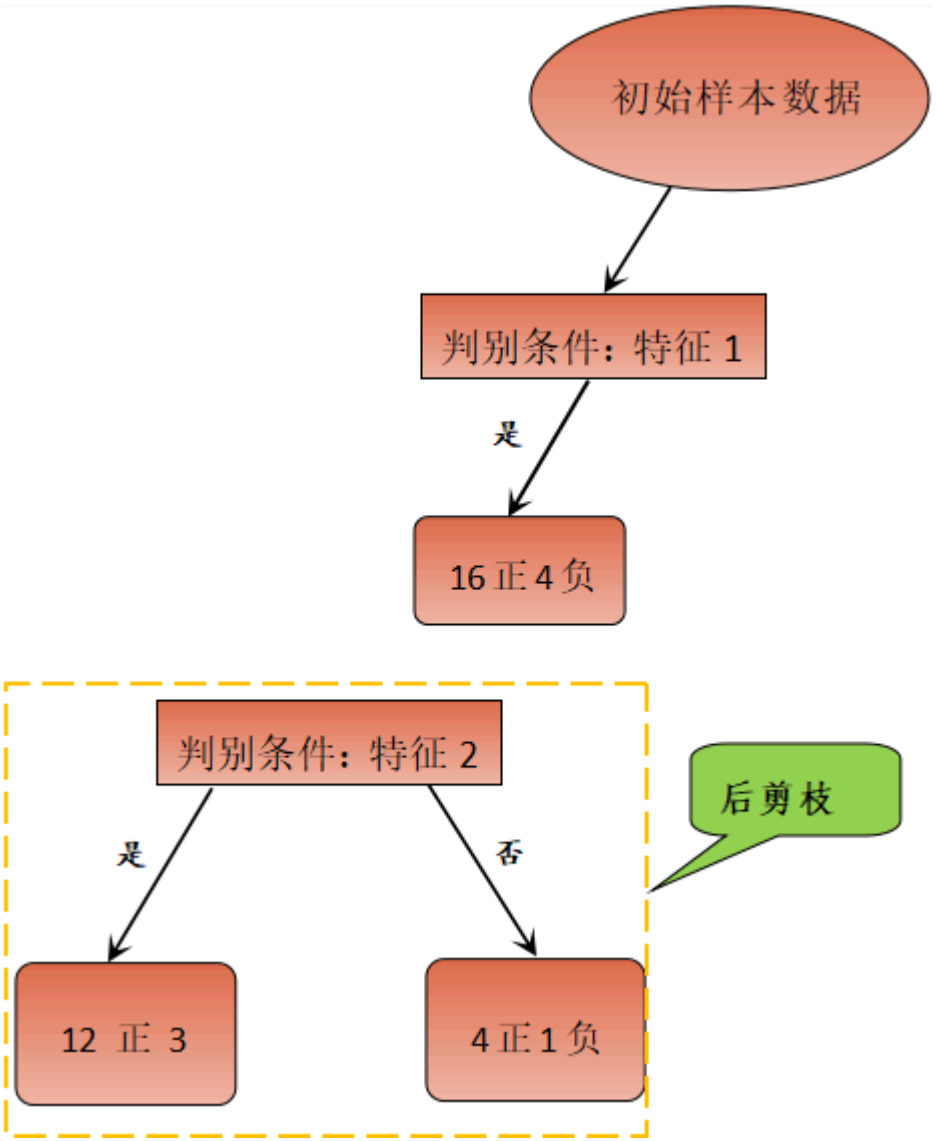
- 图中使用了两个特征对样本集合进行分类，最后正确分类的概率是 12/20。



# 例

- 如果只通过特征 1 进行分类，也就是剪掉冗余特征 2，最后的结果又是如何呢?

- 通过后剪枝策略，正确分类概率变成了 16/20。显而易见，剪枝策略使得正确分类的概率得到提高。



# 后剪枝

- **错误率降低剪枝**（REP，Reduced-Error Pruning）方法将可用数据集分成两个样本集合：训练集用于形成学习到的假设；验证集用于评估这个假设在后续数据上的精度。
  - 使用与训练样本分离的验证样本来评估通过后剪枝从树上修剪结点的效果。
  - 当**修剪后的树对于验证集的性能不比修剪前的树的性能差**时，则确认删除该结点，否则恢复该结点
  - 通常的做法是，所有样例的三分之二作训练集，三分之一作验证集。

- 此外，还有悲观错误剪枝（PEP，Pesimistic-Error Pruning）、代价复杂度剪枝（CCP，Cost-Complexity Pruning）等方法。

# 剪枝的数学理解

- 机器学习的实现方式：**极小化损失函数**。
- 决策树算法的损失函数：

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha|T|$$

- 损失函数的第一项是样本的训练误差，第二项是模型的复杂度。
  - 我们希望让每一棵子树的损失函数值尽可能小，损失函数最小化就是用正则化的极大似然估计进行模型选择的过程。

- 可以看出，***决策树的生成只是考虑通过提高信息增益对训练数据进行更好的拟合，而决策树剪枝通过优化损失函数还考虑了减小模型复杂度***。

# 决策树算法小结

- 决策树是一种类似二叉树或多叉树的树结构模型。
  - 树中的每个非叶结点（包括根结点）对应于训练样本集中的一个特征
  - 非叶结点的每一个分支对应特征的一个取值
  - 每个叶结点代表一个类或类分布
  - 从根结点到叶结点的一条路径形成一条分类规则
  - 决策树可以很方便地转化为分类规则，是一种非常直观的分类模型表示形式
  - 决策树是**有监督**的学习方法

# 决策树算法的优缺点

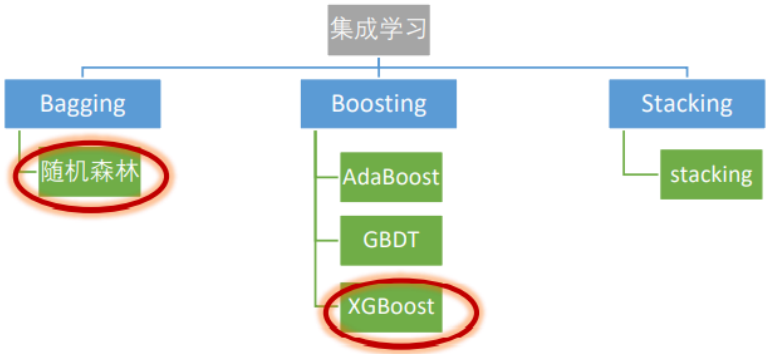
- 决策树的**优点**:
  - 模型直观清晰，计算量相对不大
  - 可以生成可理解的、易于解释的分类规则
  - 可以处理连续和离散特征
  - 可以清晰地显示哪些特征比较重要
- 决策树的**不足**:
  - 较难预测连续特征
  - 当类别太多时，错误可能会增加地比较快
  - 不是全局最优解

# 决策树方法的适用问题

- 适用问题的特征
  - 样本实例由带分类标签的“特征-值”对表示
  - 目标函数具有**离散的输出值**
  - 训练数据可以包含错误
  - 训练数据可以包含缺失值
- 分类问题
  - 核心任务是把样本分类到各个可能的类中
- 问题举例
  - 医学应用（如根据疾病分类患者、疾病分析与预测）
  - 故障诊断（如根据起因分类设备故障）
  - 信用评估（如根据拖欠支付的可能性分类贷款申请）

# 集成学习（Ensemble learning）

- 背景
  - 有监督学习算法的目标是学习出一个稳定的且预测性能较好的模型
  - 通常学习算法产生的模型有偏好，对某些数据预测效果较好（也称为弱学习器）
- 集成学习思想
  - 构建多个不同的弱学习器
  - 整合弱学习器得到一个更强大的模型（强学习器），来做最后的决策
- 多种集成方法
  - Bagging
  - Boosting
  - Stacking

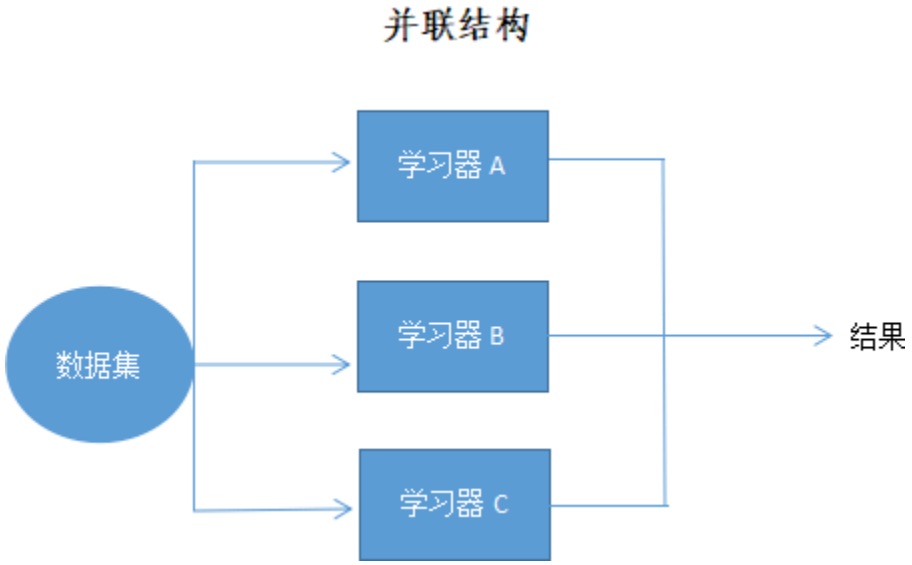


# 集成学习（Ensemble learning）

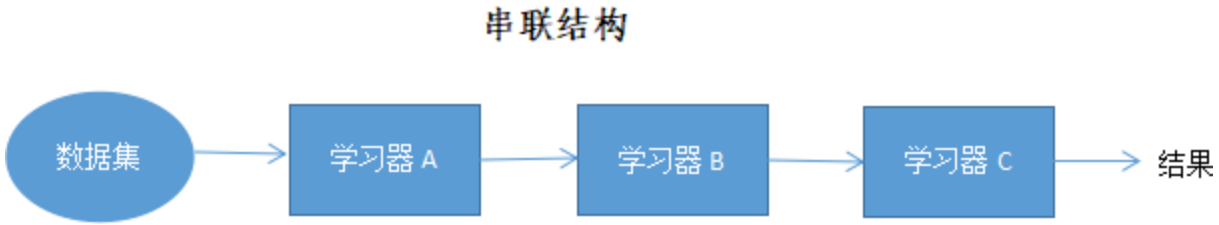
- 集成学习算法并非一种机器学习算法，它更像是一种模型优化方法。集成学习算法可以理解成是一套组合了多种机器学习算法模型的框架，它关注的是框架内各个模型之间的组织关系，而非某个模型的具体内部结构。
- 集成学习算法具有更广泛的适应场景，比如分类问题、回归问题、特征选取和异常点检测等各类机器学习任务。

# 集成学习组织方式

- 并行式集成学习（Bagging）




- 串行式集成学习（Boosting）




*串联与并联的最大区别在于，并联的学习器彼此独立，而串联则是把预测结果传递给后面的学习器。*

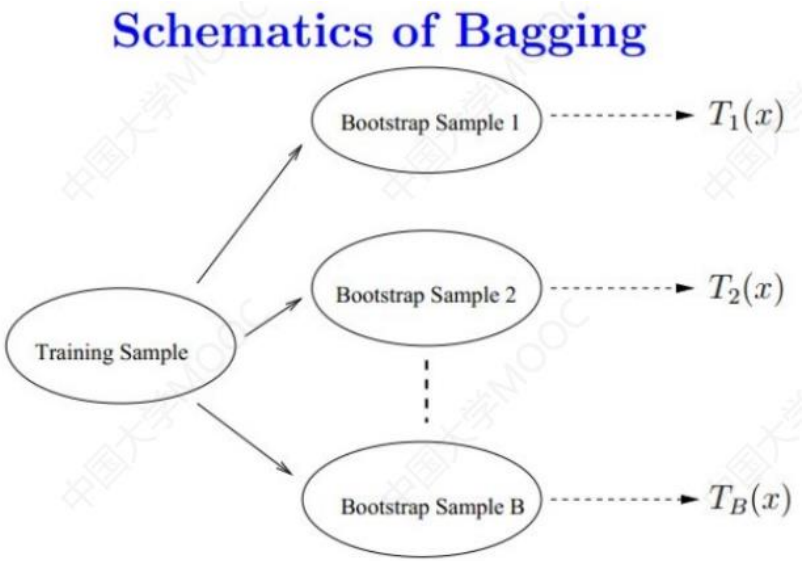
# 随机森林（Random Forest）

- **集成学习**通过建立几个模型组合的来解决单一预测问题。它的工作原理是生成多个分类器/模型，各自独立地学习和作出预测。这些预测最后结合成单预测，因此优于任何一个单分类的做出预测。
- **随机森林**是由Leo Breiman于2001年提出的一种集成学习模型，结合了其在1996年提出的Bagging集成学习理论与Ho在1998年提出的随机子空间方法。
- 它包含多个由**Bagging集成学习技术**训练得到的决策树，当输入待分类的样本 时，最终的分类结果由基决策树的输出结果**投票**决定。

# 随机森林的基本思想：Bagging策略

- Bagging的名称来源于 Bootstrap Aggregating，意为自助抽样集成

1. 从训练样本集重采样m个样本
2. 对m个样本构建基分类器
3. 重复上述步骤n次，得到n个分类器
4. 根据n个分类器的输出结果投票决定数据属于哪一类别

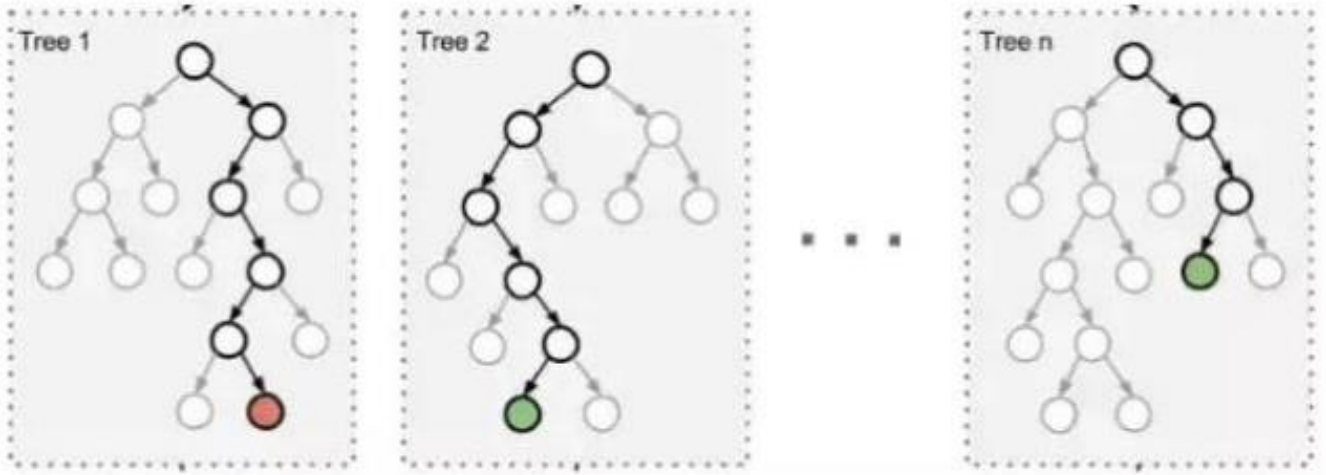



- bagging的代价是不用单分类器来做预测，具体哪个变量起到重要作用变得未知，所以bagging改进了预测准确率但损失了解释性。

# 随机森林（Random Forest）

- 随机森林相较于bagging的修改：

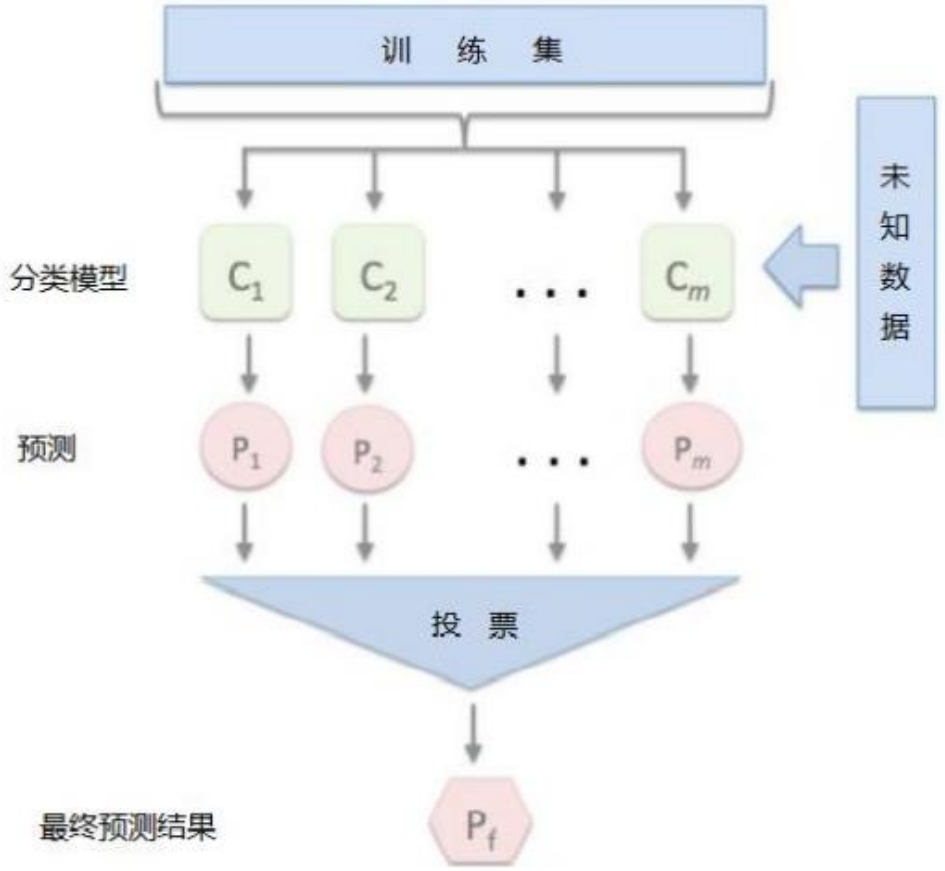
1. 重采样m个样本
2. 随机选择k个属性
3. 建立n棵决策树
4. 投票表决结果



# 随机森林（Random Forest）

- 通过bootstrapping，对于每一棵决策树，大概有 $\lim_{m\to\infty}(1-\frac{1}{m})^m=\frac{1}{e}\approx 0.368$ 比例的样本始终未被采到，可将其作为该树的验证集，进行袋外估计（out-of-bag estimate）。

- 以右图的方式训练得到K个决策树组合起来，就可以得到一个随机森林。当输入待分类的样本时，分类结果由所有输出结果进行投票决定。



# 随机森林的分类效果

- 随机森林的泛化误差界:

$$PE^* \leq \frac{\bar{\rho}(1-s^2)}{s^2}$$

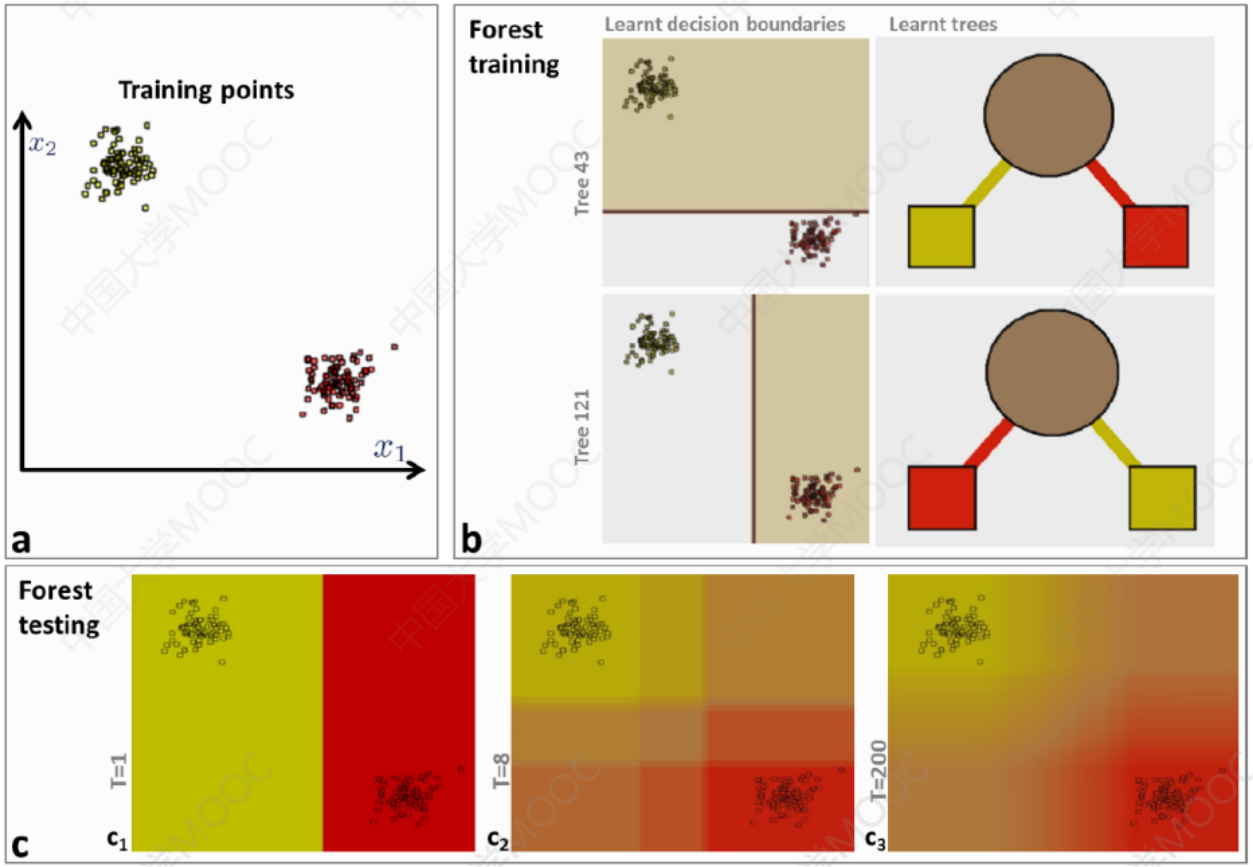
s：代表单个树的分类强度

$\bar{\rho}$：代表树间的相关性

  - 森林中任意两棵树的相关性越强，整个森林的错误率越高;
  - 森林中每棵树的分类能力越强，整个森林的错误率越低。

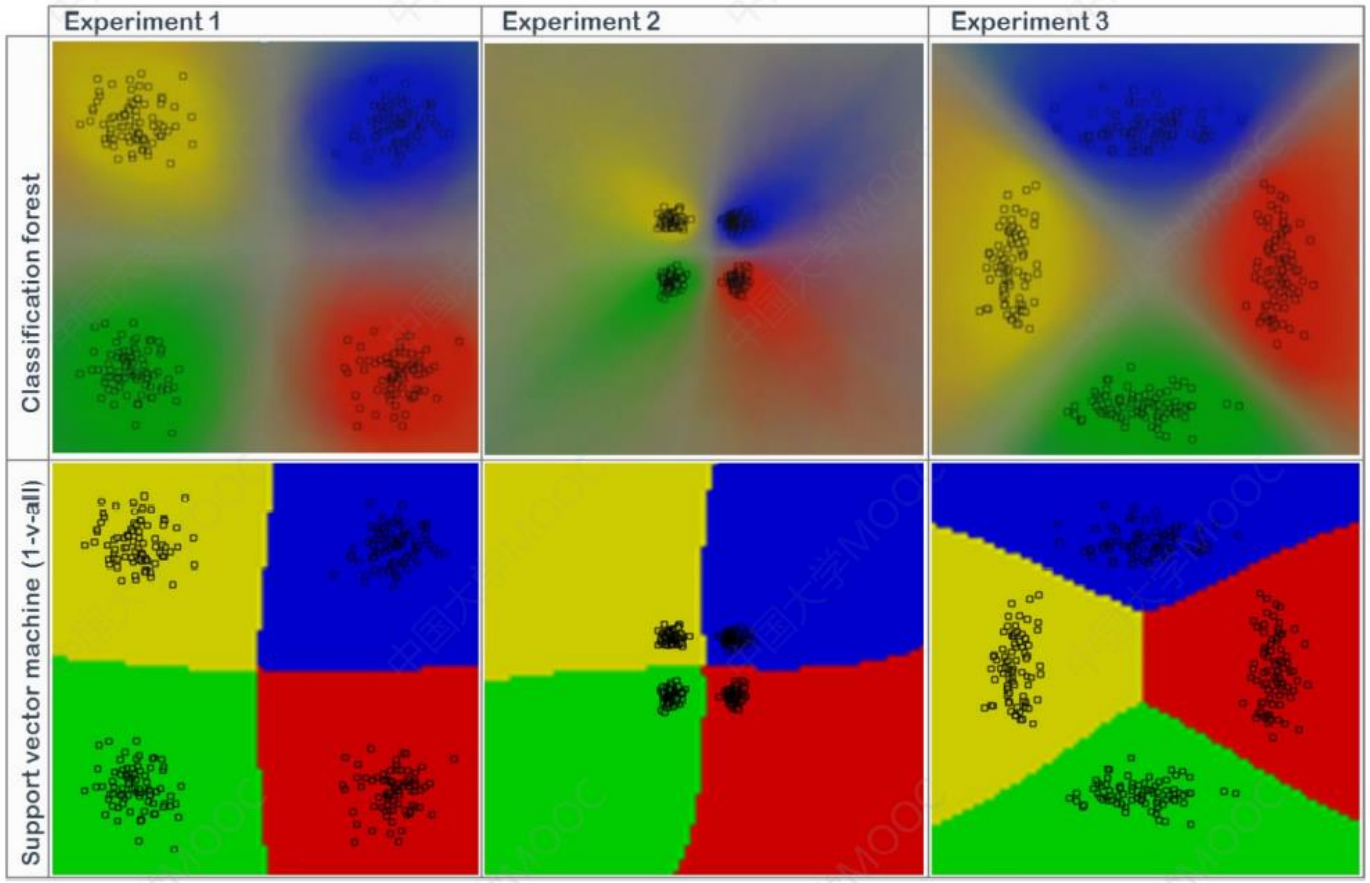
- 减小特征选择个数k，树的相关性和分类能力也会相应的降低；增大k，两者也会随之增大。因此k是一个重要的参数。

# 随机森林中树的数量影响




当只有一棵树时，模型展示出**非常确定**的分类效果。随着树数量的增加，决策平面变得逐渐**平滑**，越接近训练样本的位置颜色越深，提供了对结果的**置信度**。

# 随机森林 vs. SVM




随机森林往往可以在特征空间中提供一个**概率性**的结果，而SVM往往只能给出**确定性**的结果，且在不同类的训练样本靠近或远离时，决策平面**比较相似**。

# 随机森林算法的优缺点

- 优点:
- 模型准确率高
- 能够处理数量庞大的高维度的特征，且不需要进行降维（因为特征子集是随机选择的）；
- 能够评估各个特征在分类问题上的重要性
- 对异常值、缺失值不敏感。

- 缺点:
- 随机森林解决回归问题的效果不如分类问题;
- 树之间的相关性越大，错误率就越大;
- 当训练数据噪声较大时，容易产生过拟合现象。

# 小结

- 随机森林/Bagging和决策树的关系
  - 当然可以用决策树作为基本分类器
  - 也可以使用SVM、Logistic回归等其他分类器，这些分类器组成的“总分类器”，仍在随机森林的框架下。树模型的优势在于可以在每个结点上都增加模型随机性，有利于增大基分类器之间的差异。

- 要点：
  - 森林：集成思想
  - 随机：样本随机和特征随机
  - 采用随机采样和随机特征的办法，随机森林足以保证不同的基分类器之间具有足够的不相关性，因而**不再需要对个体决策树进行剪枝**

# Sklearn vs. 分类

- 与聚类算法被统一封装在sklearn.cluster 模块不同，sklearn库中的分类算法并未被统一封装在一个子模块中，因此对分类算法的 import 方式各有不同
- Sklearn提供的分类函数包括：

| 分类模型 | 加载模块 |
| --- | --- |
| 最近邻算法 | neighbors.NearestNeighbors |
| 支持向量机 | svm.SVC |
| 朴素贝叶斯 | naive_bayes.GaussianNB |
| 决策树 | tree.DecisionTreeClassifier |
| 集成方法 | ensemble.BaggingClassifier |
| 神经网络 | neural_network.MLPClassifier |

- 这其中有线性分类器，也有非线性分类器

# 决策树分类实现

- Scikit-learn决策树：DecisionTreeClassifier类
  - 支持二分类和多分类问题

**模型初始化：clf = tree.DecisionTreeClassifier()**

**模型学习：clf.fit(X, y)**

**Accuracy计算：clf.score(X,y)**

**模型预测：predicted_y = clf.predict(X)**

| 参数说明： | |
|---|---|
| *X*[m,n] | 样本特征二维数组，*m* 样本数，*n* 特征项个数，数值型 |
| *y*[n] | 分类标签的一维数组，必须为整数 |

**混淆矩阵计算：metrics.confusion_matrix(y, predicted_y)**

**分类性能报告：metrics.classification_report(y, predicted_y)**

# 集成学习实现

- Scikit-learn： ensemble包

| | 随机森林 |
|---|---|
| 分类 | RandomForestClassifier |
| 回归 | RandomForestRegressor |

# 离散数据处理

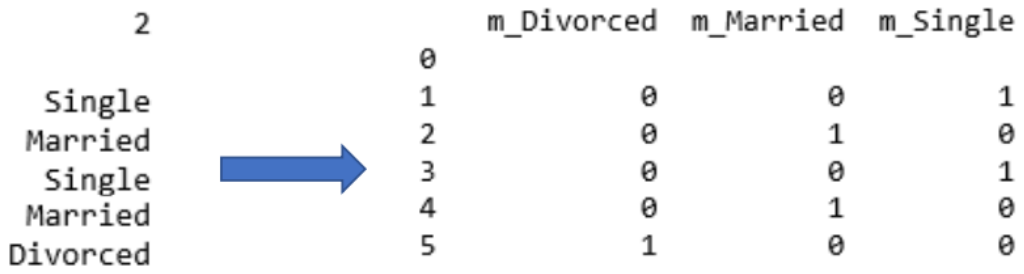
- 数据集中的非数值型数据
  - 房产：**Yes/No**，婚姻状况：**Married/Single/Divorced**
  - 需要转换为数值型数据处理

- 替换为指定值
  - ‘Yes’ → 1，‘No’ → 0

```
>>> data.loc[data[1] == 'Yes',1 ] = 1
>>> data.loc[data[1] == 'No',1 ] = 0
```


- 转换为one-hot（独热）矩阵

```
>>> dumm_marital = pd.get_dummies( data[2], prefix='marital' )
```



# 分类模型性能评估

- 使用分类器计算样本分类结果，得到预测类
- 计算每个样本真实类（ground truth）对应的预测类，得到混淆矩阵（confusion matrix）

| 真实类 \ 预测类 | Class=Yes | Class=No |
|---|---|---|
| Class=Yes | a | b |
| Class=No | c | d |

- 准确率Accuracy

$$Accuracy = \frac{a+d}{a+b+c+d}$$

- 实际应用关心模型对某一特定类别（Yes）的预测能力
  - 精确率（Precision）、召回率（Recall）和F1-measure

$$Precision = \frac{a}{a+c} \qquad Recall = \frac{a}{a+b} \qquad F1 = \frac{2a}{2a+b+c}$$

# Thank you