法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

决策树和随机森林

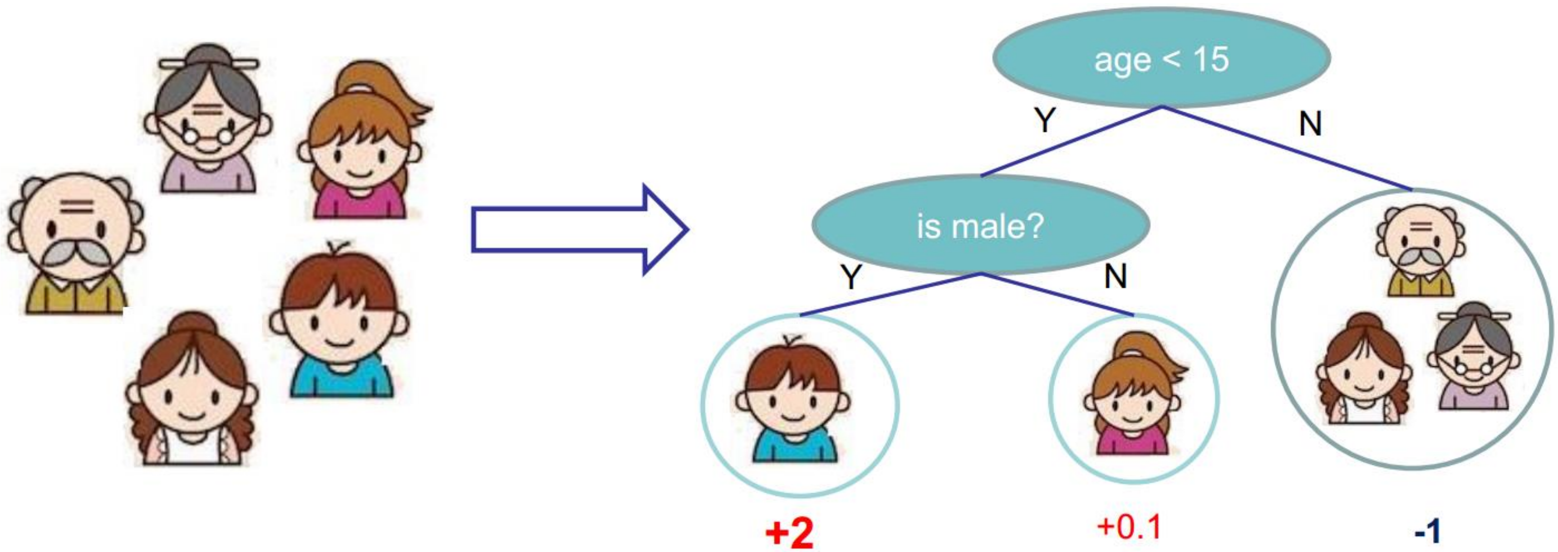
小象学院 ChinaHadoop.cn 邹博

目标任务与主要内容

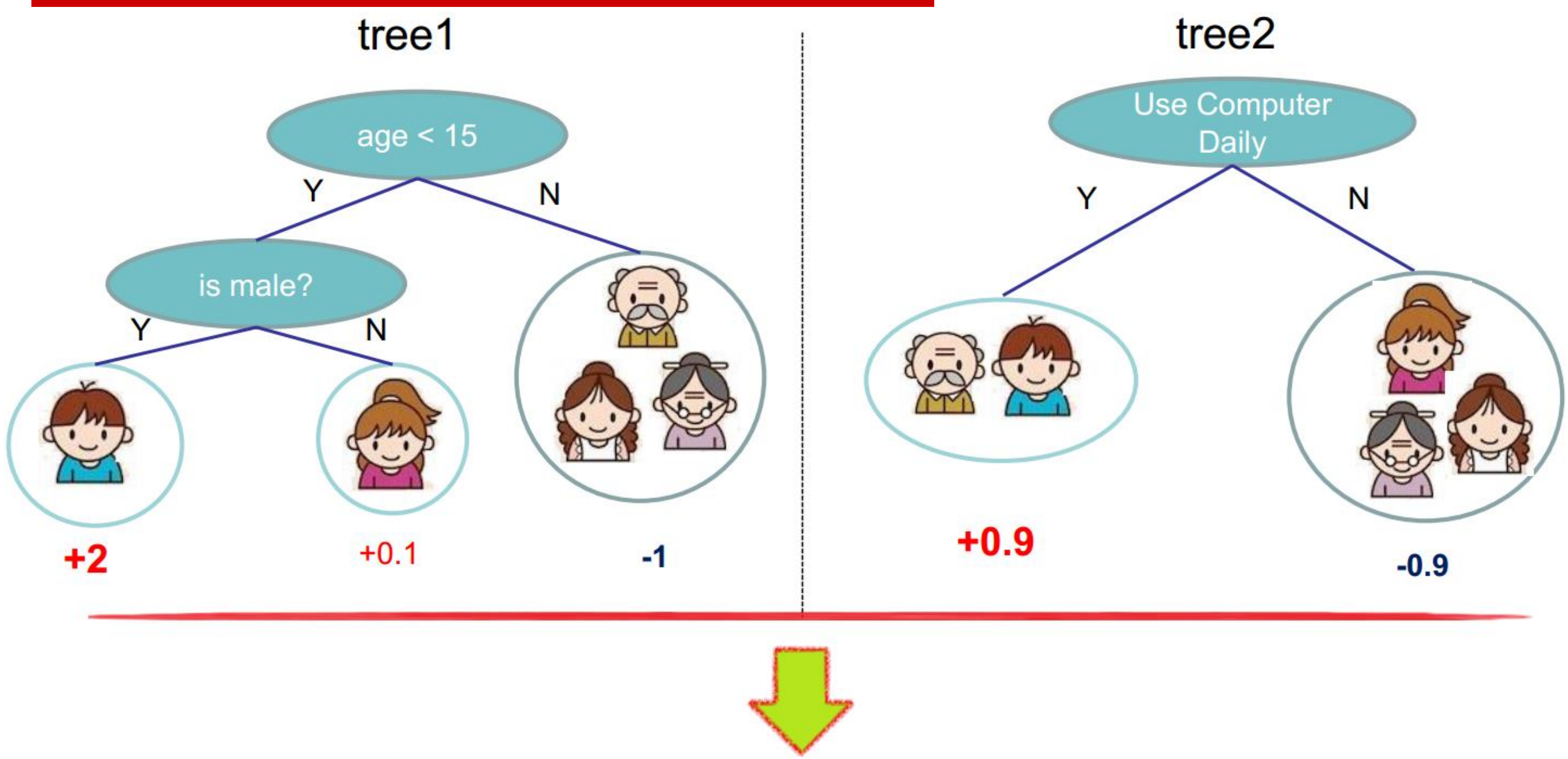
- 复习信息熵
  - 熵、联合熵、条件熵、互信息
- 决策树学习算法
  - 信息增益
  - ID3、C4.5、CART
- Bagging与随机森林

CART

- 输入数据x：M个样本数据，每个数据包括年龄、性别、职业、每日使用计算机时间等
- 输出y：该样本是否喜欢计算机游戏

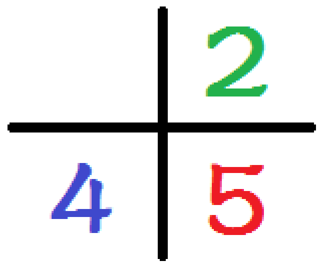


随机森林

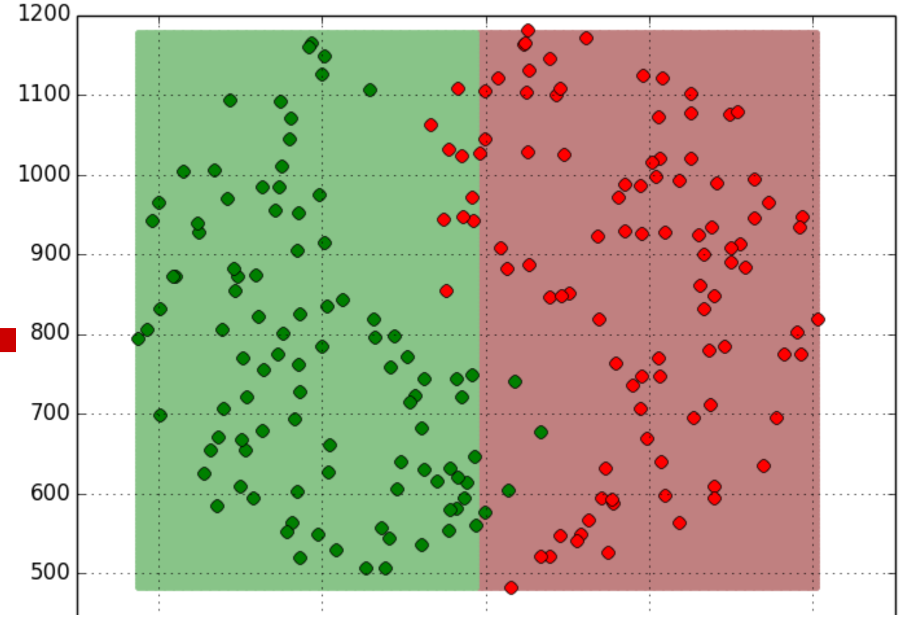


f() = 2 + 0.9= 2.9 f()= -1 + 0.9= -0.1

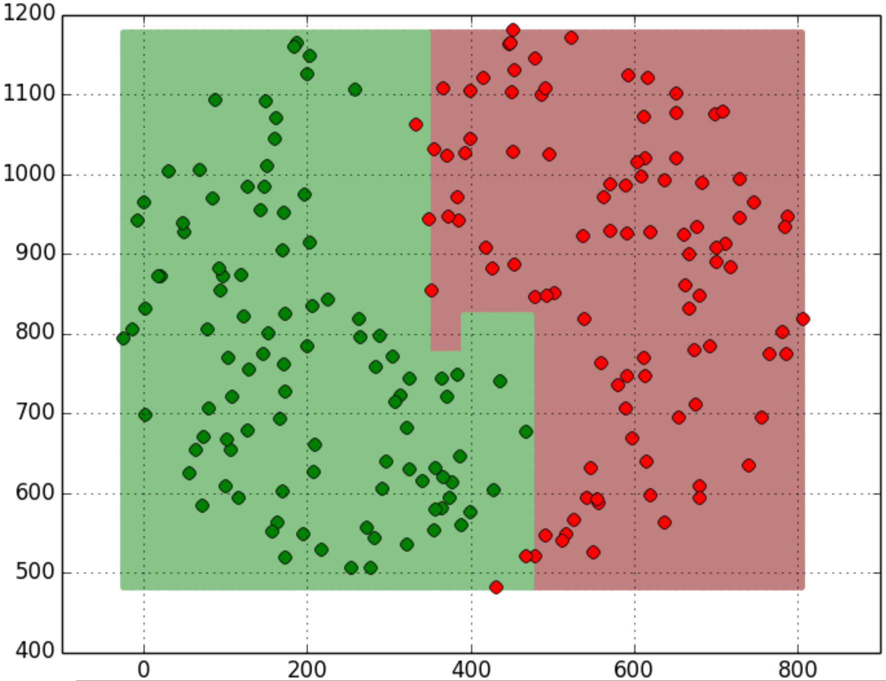
决策树：Level



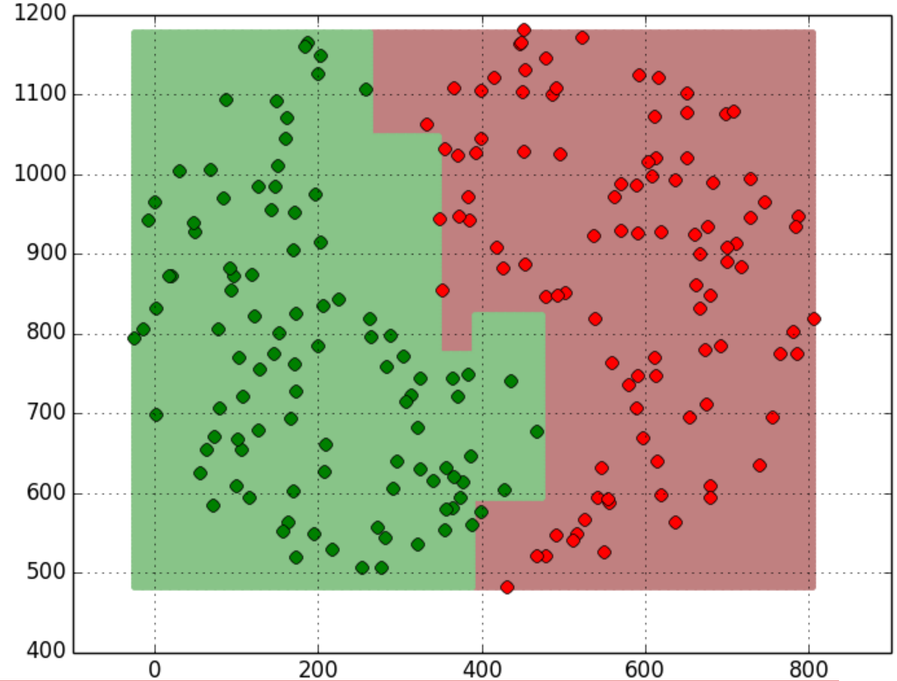
2
4
5



1200
1100
1000
900
800
700
600
500



1200
1100
1000
900
800
700
600
500
400
0
200
400
600
800



1200
1100
1000
900
800
700
600
500
400
0
200
400
600
800

复习：条件熵

- H(X,Y) – H(X)
  - (X,Y)发生所包含的熵，减去X单独发生包含的熵：在X发生的前提下，Y发生“新”带来的熵
  - 该式子定义为X发生前提下，Y的熵：
    - 条件熵H(Y|X)

复习：推导条件熵的定义式

$$
\begin{aligned}
& H(X,Y)-H(X) \\
&= -\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x} p(x)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x}\left(\sum_{y} p(x,y)\right)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x,y} p(x,y)\log p(x) \\
&= -\sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)} \\
&= -\sum_{x,y} p(x,y)\log p(y\mid x)
\end{aligned}
$$

复习：根据条件熵的定义式，可以得到

$$
\begin{aligned}
H(X,Y)-H(X) &= -\sum_{x,y} p(x,y)\log p(y|x) \\
&= -\sum_{x}\sum_{y} p(x,y)\log p(y|x) \\
&= -\sum_{x}\sum_{y} p(x)p(y|x)\log p(y|x) \\
&= -\sum_{x} p(x)\sum_{y} p(y|x)\log p(y|x) \\
&= \sum_{x} p(x)\left(-\sum_{y} p(y|x)\log p(y|x)\right) \\
&= \sum_{x} p(x)H\left(Y|X=x\right)
\end{aligned}
$$

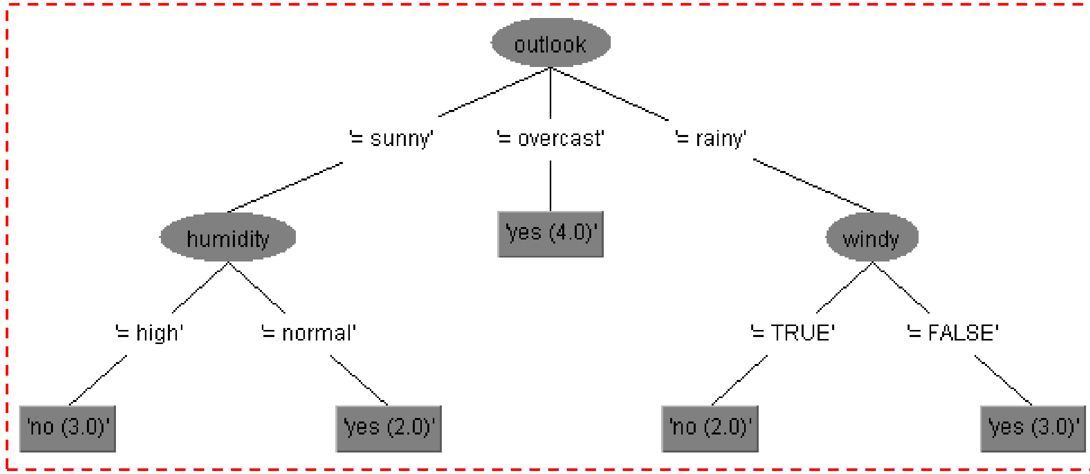
决策树的实例(WEKA The University of Waikato 自带测试数据)

Viewer

Relation: weather.symbolic

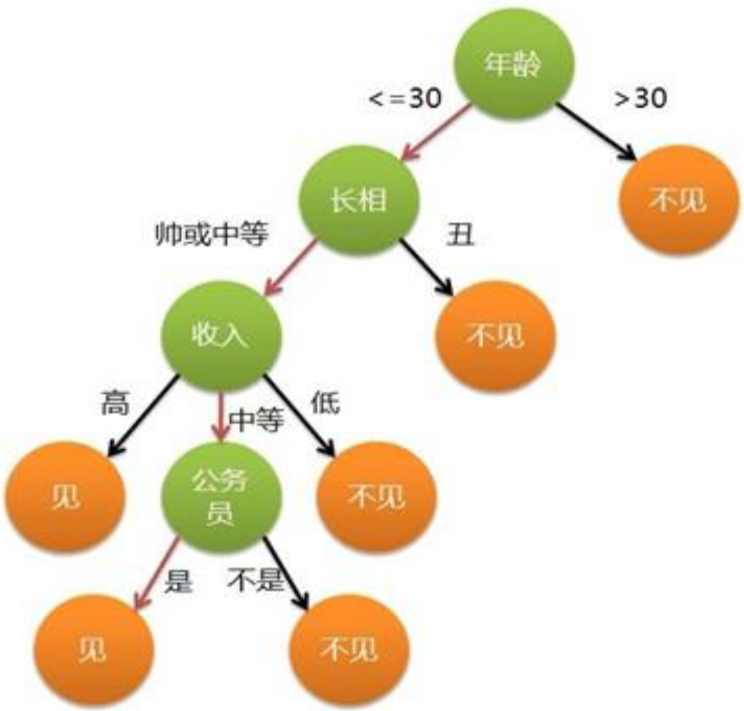
No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Undo OK Cancel

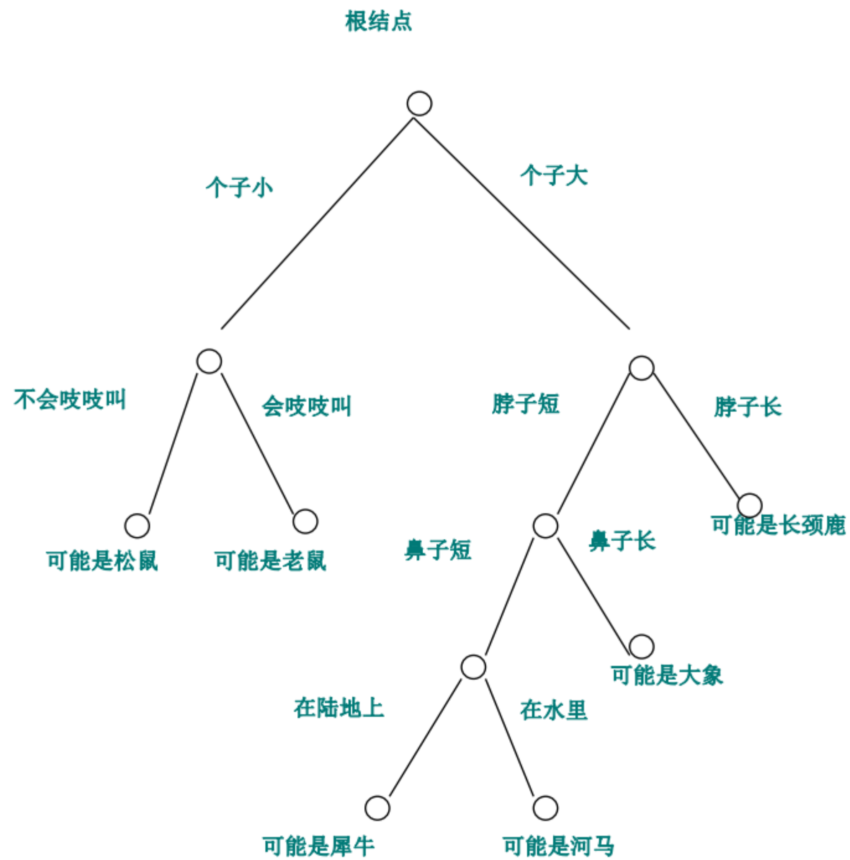


注：Weka的全名是怀卡托智能分析环境(Waikato Environment for Knowledge Analysis)，是一款免费的，非商业化(与之对应的是SPSS公司商业数据挖掘产品--Clementine)的，基于JAVA环境下开源的机器学习(machine learning)以及数据挖掘(data minining)软件。它和它的源代码可在其官方网站下载。

决策树示意图



年龄
<=30
>30
长相
不见
帅或中等
丑
收入
不见
高
中等
低
见
公务员
不见
是
不是
见
不见



根结点
个子小
个子大
不会吱吱叫
会吱吱叫
脖子短
脖子长
可能是松鼠
可能是老鼠
鼻子短
鼻子长
可能是长颈鹿
可能是大象
在陆地上
在水里
可能是犀牛
可能是河马

决策树 (Decision Tree)

- 决策树是一种树型结构，其中每个内部结点表示在一个属性上的测试，每个分支代表一个测试输出，每个叶结点代表一种类别。
- 决策树学习是以实例为基础的归纳学习。
- 决策树学习采用的是自顶向下的递归方法，其基本思想是以信息熵为度量构造一棵熵值下降最快的树，到叶子节点处的熵值为零，此时每个叶节点中的实例都属于同一类。

决策树学习算法的特点

- 决策树学习算法的最大优点是，它可以自学习。在学习的过程中，不需要使用者了解过多背景知识，只需要对训练实例进行较好的标注，就能够进行学习。
  - 显然，属于有监督学习。
  - 从一类无序、无规则的事物(概念)中推理出决策树表示的分类规则。

决策树学习的生成算法

- 建立决策树的关键，即在当前状态下选择哪个属性作为分类依据。根据不同的目标函数，建立决策树主要有一下三种算法。
  - ID3
    - Iterative Dichotomiser
  - C4.5
  - CART
    - Classification And Regression Tree

信息增益

- 概念：当熵和条件熵中的概率由数据估计(特别是极大似然估计)得到时，所对应的熵和条件熵分别称为**经验熵**和**经验条件熵**。
- 信息增益表示得知特征A的信息而使得类X的信息的不确定性减少的程度。
- 定义：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，即：
  - $g(D,A)=H(D)-H(D|A)$
  - 显然，这即为训练数据集D和特征A的互信息。

基本记号

- 设训练数据集为 D，$|D|$ 表示样本个数。
- 设有K个类 $C_k, k=1,2\cdots K$，$|C_k|$为属于类 C_k 的样本个数，有：$\sum_k |C_k| = |D|$
- 设特征A有n个不同的取值 $\{a_1, a_2 \cdots a_n\}$，根据特征A的取值将D划分为n个子集 $D_1, D_2 \cdots D_n$，$|D_i|$为 D_i 的样本个数，有：$\sum_i |D_i| = |D|$
- 记子集 D_i 中属于类 C_k 的样本的集合为 D_{ik}，$|D_{ik}|$ 为 D_{ik} 的样本个数。

信息增益的计算方法

- 计算数据集D的经验熵 $H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log\frac{|C_k|}{|D|}$
- 遍历所有特征，对于特征A：
  - 计算特征A对数据集D的经验条件熵H(D|A)
  - 计算特征A的信息增益：g(D,A)=H(D) – H(D|A)
  - 选择信息增益最大的特征作为当前的分裂特征

经验条件熵H(D|A)

$$H(D|A) = -\sum_{i,k} p(D_k, A_i) \log p(D_k | A_i)$$

$$= -\sum_{i,k} p(A_i) p(D_k | A_i) \log p(D_k | A_i)$$

$$= -\sum_{i=1}^{n} \sum_{k=1}^{K} p(A_i) p(D_k | A_i) \log p(D_k | A_i)$$

$$= -\sum_{i=1}^{n} p(A_i) \sum_{k=1}^{K} p(D_k | A_i) \log p(D_k | A_i)$$

$$= -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log \frac{|D_{ik}|}{|D_i|}$$

其他目标

- 信息增益率：$g_r(D,A) = g(D,A) / H(A)$
- Gini系数：

$$Gini(p)=\sum_{k=1}^{K} p_k(1-p_k)=1-\sum_{k=1}^{K} p_k^2$$

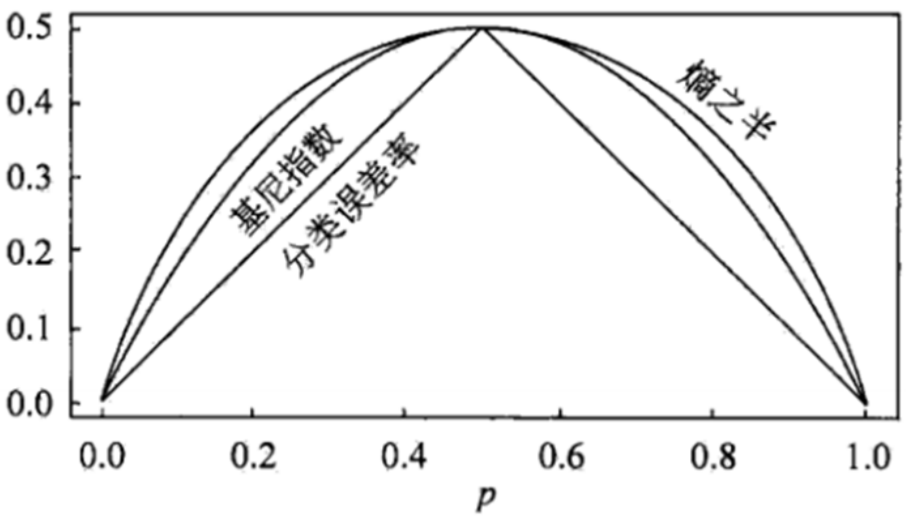
$$=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2$$

关于Gini系数的讨论(一家之言)

- 考察Gini系数的图像、熵、分类误差率三者之间的关系
  - 将f(x)=-lnx在x=1处一阶展开，忽略高阶无穷小，得到f(x)≈1-x

$$H(X) = -\sum_{k=1}^{K} p_k \ln p_k$$

$$\approx \sum_{k=1}^{K} p_k (1 - p_k)$$



Gini系数的第二定义

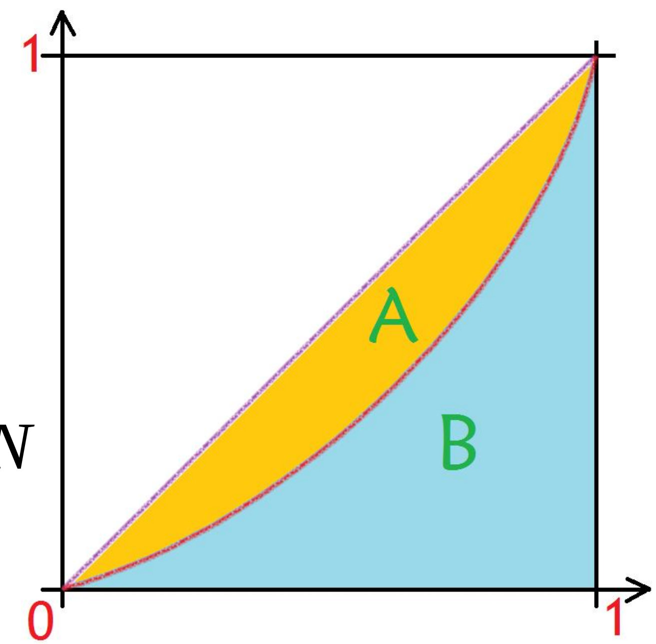
- 给定M个样本，计算样本最大值max和最小值min，等分成N份，计算每份的样本数目$x_i(i=1,2,\ldots,N)$，则每份的近似概率为

$$p_i = \frac{x_i}{M}, \quad i = 1,2,\cdots,N$$

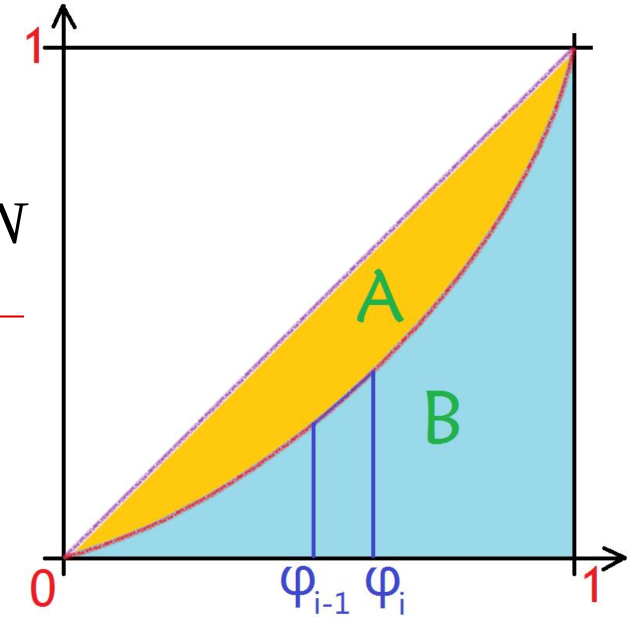
- 计算累积概率

$$\varphi_i = \sum_{i=1}^{i} p_i = \sum_{i=1}^{i} \frac{x_i}{M} = \frac{1}{M}\sum_{i=1}^{i} x_i, \; i = 1,2,\cdots,N$$

- 另：$\varphi_0 = 0, \; \varphi_N = 1$



推导Gini系数 $\varphi_i = \frac{1}{M}\sum_{i=1}^{i} x_i,\ i = 1,2\cdots N$



$$\begin{cases} \varphi_0 = 0 \\ \varphi_N = 1 \end{cases}$$

- 阴影B的面积：

$$S_B = \sum_{i=1}^{N}\left(\frac{\varphi_{i-1}+\varphi_i}{2}\cdot\frac{1}{N}\right) = \frac{1}{2N}\sum_{i=1}^{N}(\varphi_{i-1}+\varphi_i)$$

$$= \frac{1}{2N}\left(\sum_{i=1}^{N}\varphi_i + \sum_{i=1}^{N}\varphi_{i-1}\right) = \frac{1}{2N}\left(\sum_{i=1}^{N}\varphi_i + \left(\sum_{i=1}^{N}\varphi_i\right) - 1\right)$$

$$= \frac{1}{2N}\left(2\sum_{i=1}^{N}\varphi_i - 1\right)$$

- gini系数：

$$Gini(x) = \frac{S_A}{S_A+S_B} = \frac{1/2 - S_B}{1/2} = 1 - 2S_B = 1 - \frac{1}{N}\left(2\sum_{i=1}^{N}\varphi_i - 1\right)$$

三种决策树学习算法

- ID3：使用信息增益/互信息g(D,A)进行特征选择
  - 取值多的属性，更容易使数据更纯，其信息增益更大。
  - 训练得到的是一棵庞大且深度浅的树：不合理。
- C4.5：信息增益率 $g_r(D,A) = g(D,A) / H(A)$
- CART：基尼指数

- 一个属性的信息增益(率)/gini指数越大，表明属性对样本的熵减少的能力更强，这个属性使得数据由不确定性变成确定性的能力越强。

决策树的评价

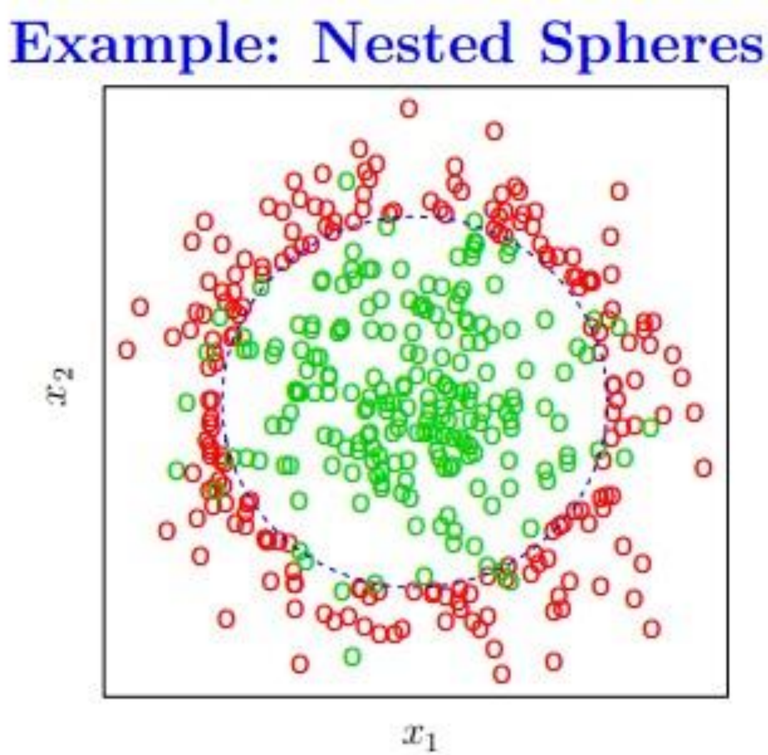
- 假定样本的总类别为K个。
- 对于决策树的某叶结点，假定该叶结点含有样本数目为n，其中第k类的样本点数目为n_k，$k=1,2,\ldots,K$。
  - 若某类样本$n_j=n$而$n_1,\ldots,n_{j-1},n_{j+1},\ldots,n_K=0$，称该结点为纯结点；
  - 若各类样本数目$n_1=n_2=\ldots=n_k=n/K$，称该样本为均结点。
- 纯结点的熵$H_p=0$，最小
- 均结点的熵$H_u=\ln K$，最大
- 对所有叶结点的熵求和，该值越小说明对样本的分类越精确。
  - 各叶结点包含的样本数目不同，可使用样本数加权求熵和
- 评价函数：

$$C(T)=\sum_{t\in leaf} N_t\cdot H(t)$$

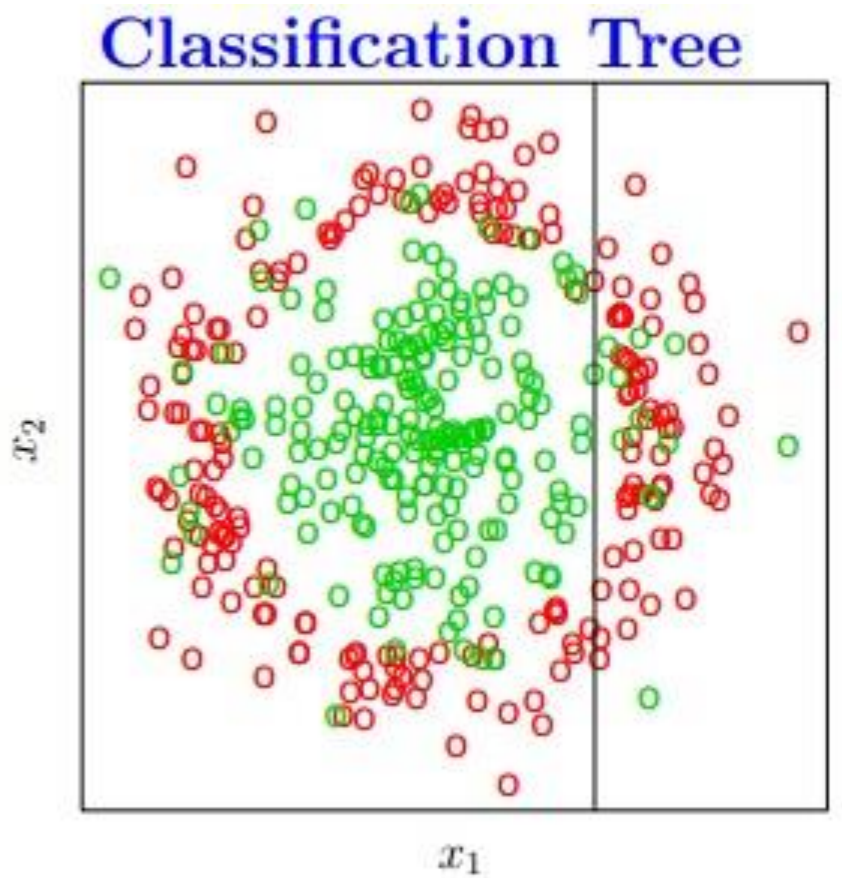
  - 由于该评价函数越小越好，所以，可以称之为“损失函数”。

决策树的例子

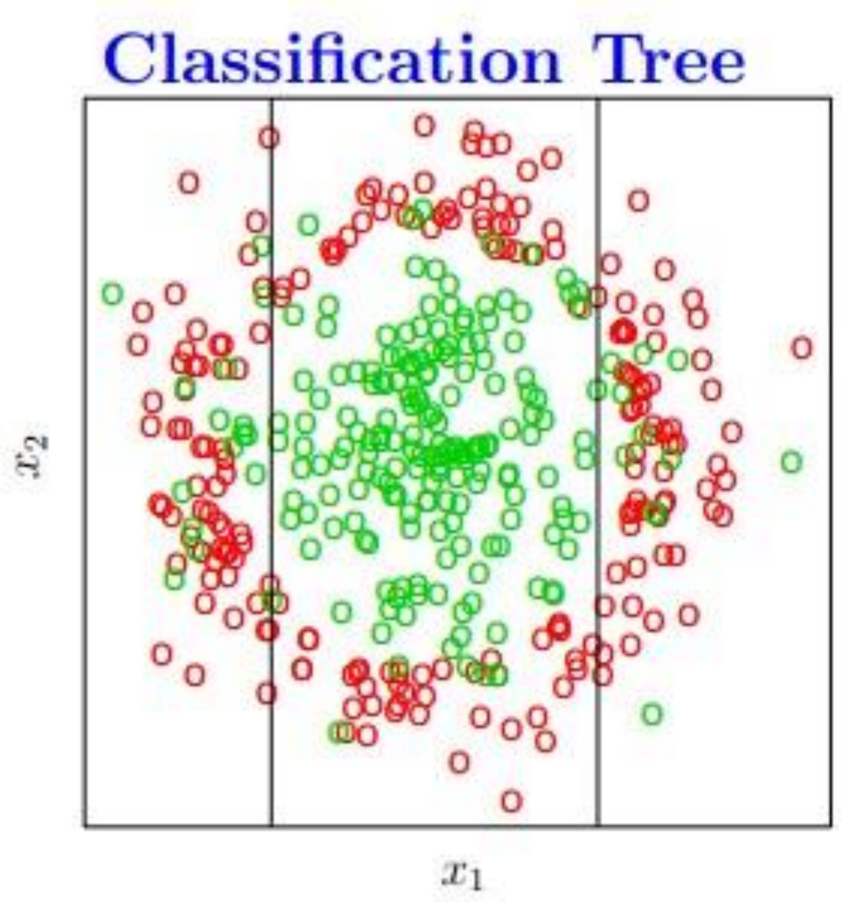
- 对于下面的数据，希望分割成红色和绿色两个类



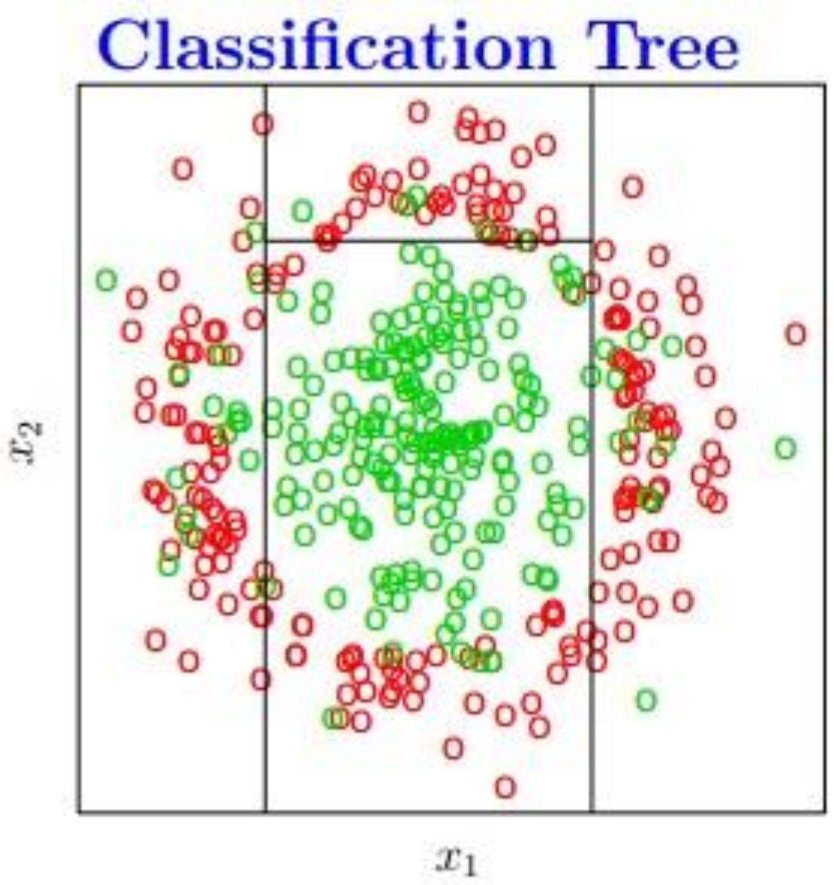
决策树的生成过程



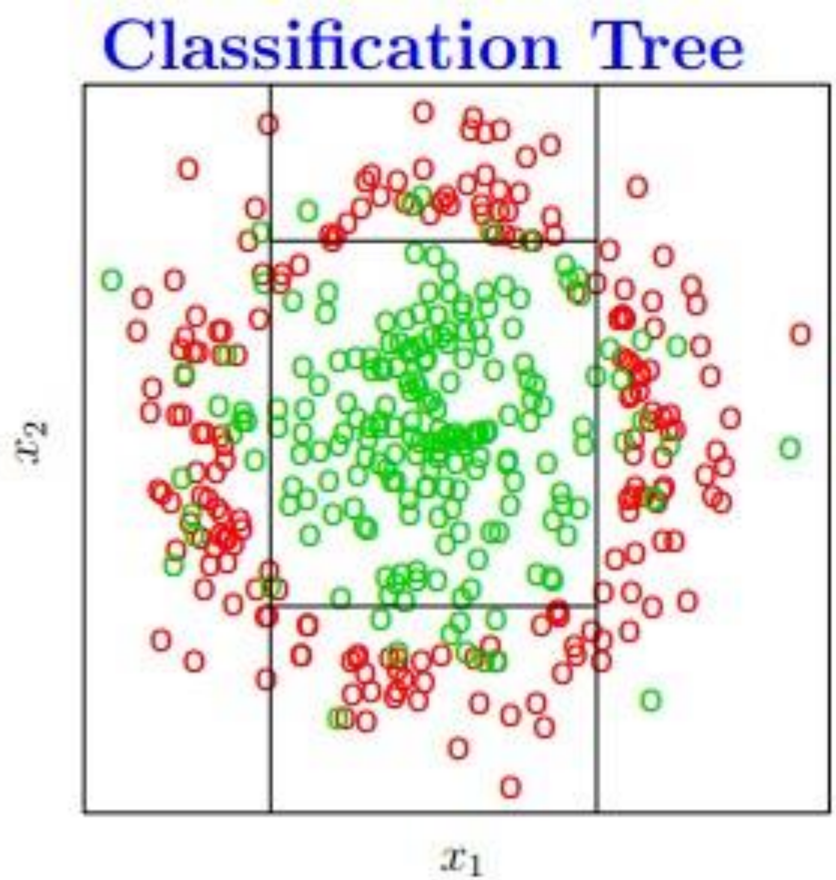
决策树的生成过程



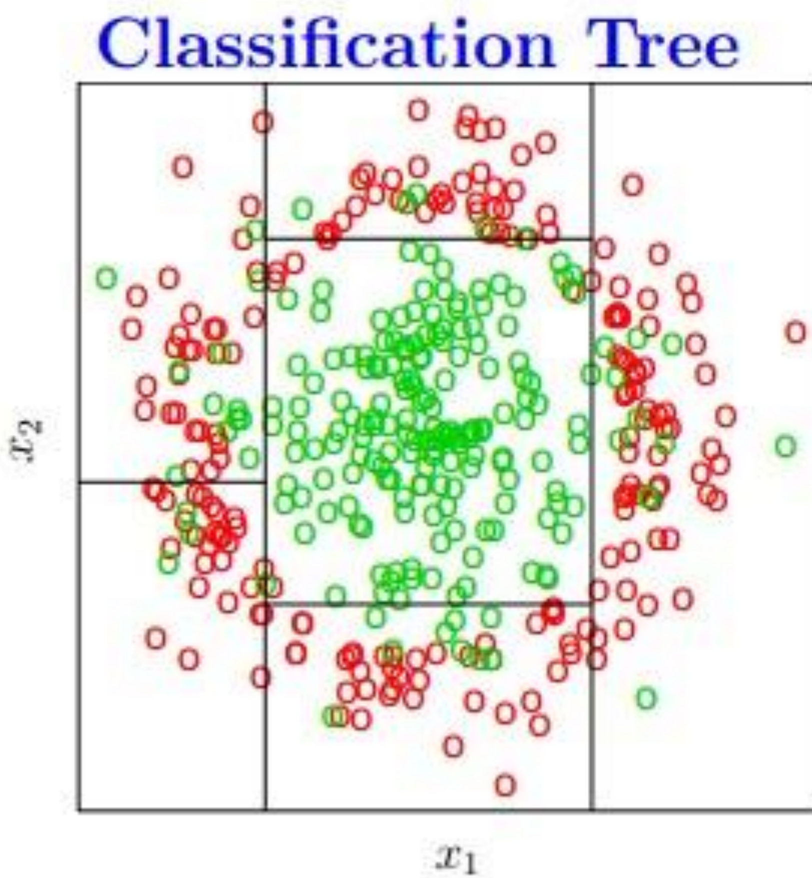
决策树的生成过程



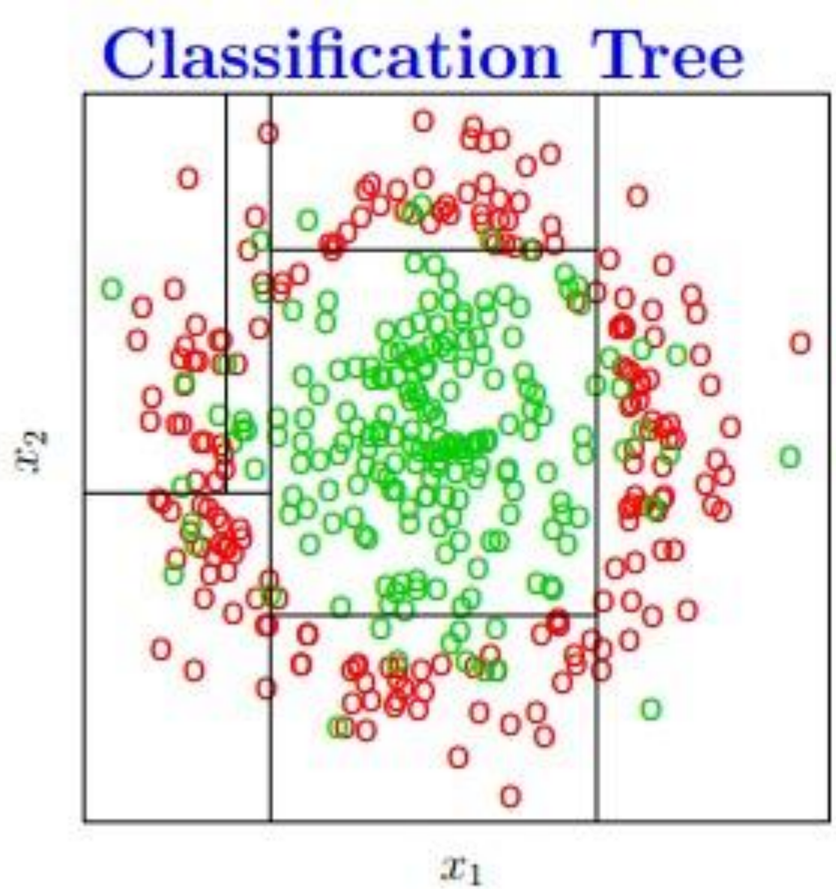
决策树的生成过程



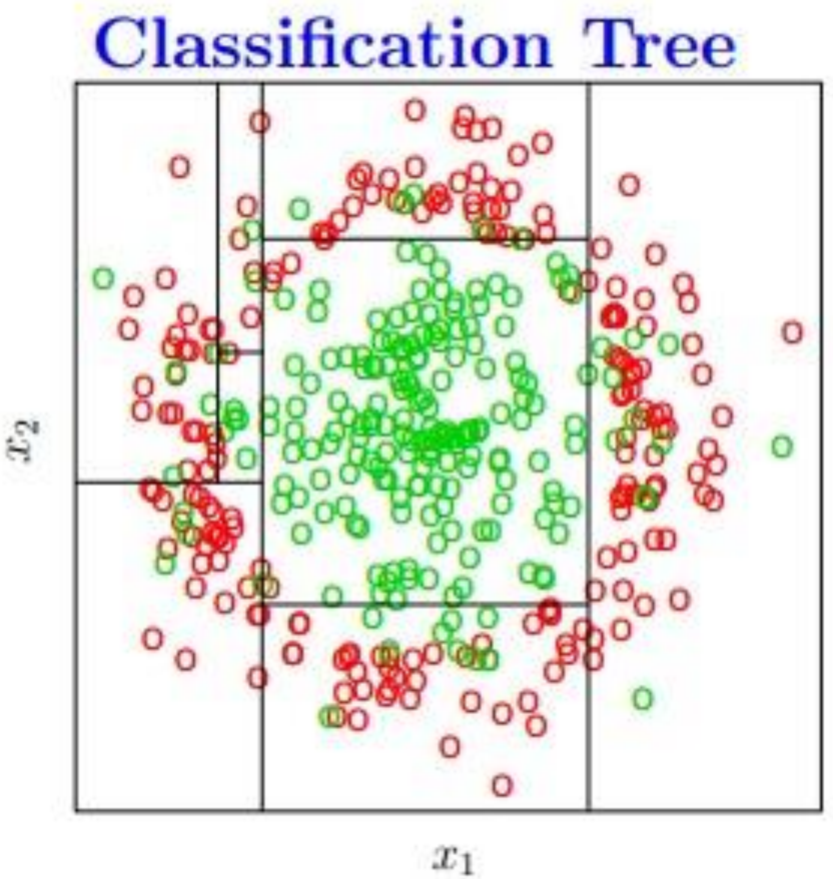
决策树的生成过程



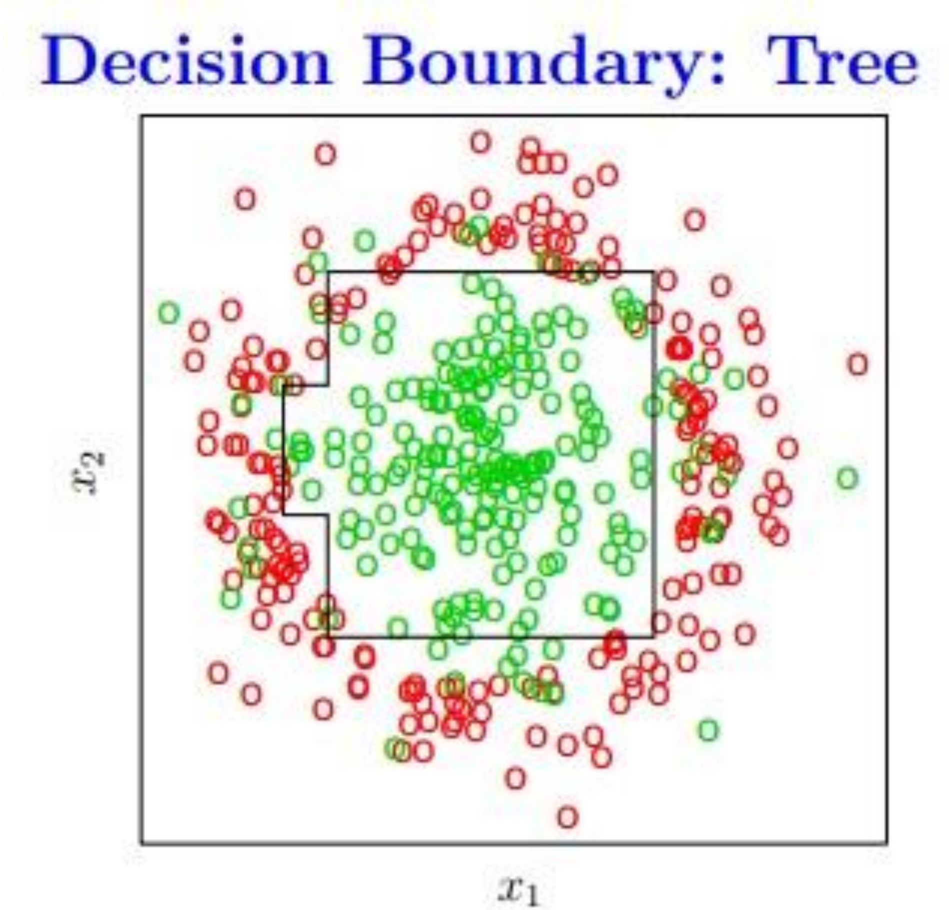
决策树的生成过程



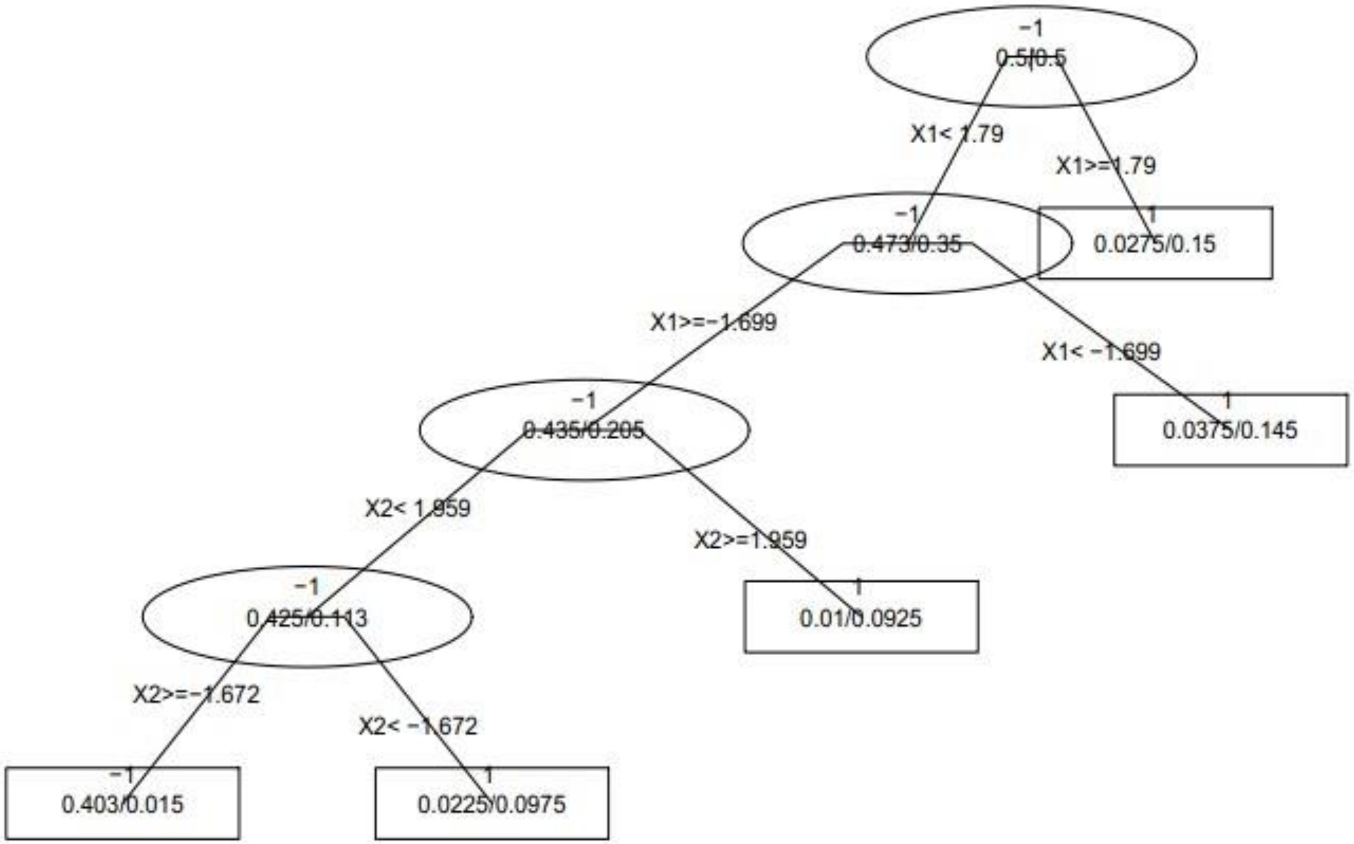
决策树的生成过程



决策树的生成过程



决策树的生成过程



−1
0.5/0.5
X1< 1.79
X1>=1.79
−1
0.473/0.35
1
0.0275/0.15
X1>=−1.699
X1< −1.699
−1
0.435/0.205
1
0.0375/0.145
X2< 1.959
X2>=1.959
−1
0.425/0.113
1
0.01/0.0925
X2>=−1.672
X2< −1.672
−1
0.403/0.015
1
0.0225/0.0975

决策树的过拟合

- 决策树对训练属于有很好的分类能力，但对未知的测试数据未必有好的分类能力，泛化能力弱，即可能发生过拟合现象。
  - 剪枝
  - 随机森林

剪枝

- 三种决策树的剪枝过程算法相同，区别仅是对于当前树的评价标准不同。
  - 信息增益、信息增益率、基尼系数
- 剪枝总体思路：
  - 由完全树T_0开始，剪枝部分结点得到T_1，再次剪枝部分结点得到T_2…直到仅剩树根的树T_k；
  - 在验证数据集上对这k个树分别评价，选择损失函数最小的树T_α

剪枝系数的确定

- 根据原损失函数 $C(T)=\sum_{t\in leaf} N_t \cdot H(t)$
- 叶结点越多，决策树越复杂，损失越大，修正：
  - 当α=0时，未剪枝的决策树损失最小； $C_\alpha(T)=C(T)+\alpha\cdot\left|T_{leaf}\right|$
  - 当α=+∞时，单根结点的决策树损失最小。
- 假定当前对以r为根的子树剪枝：
  - 剪枝后，只保留r本身而删掉所有的叶子
- 考察以r为根的子树：
  - 剪枝后的损失函数： $C_\alpha(r)=C(r)+\alpha$
  - 剪枝前的损失函数： $C_\alpha(R)=C(R)+\alpha\cdot\left|R_{leaf}\right|$
  - 令二者相等，求得： $\alpha=\dfrac{C(r)-C(R)}{<R_{leaf}>-1}$
- α称为结点r的剪枝系数。

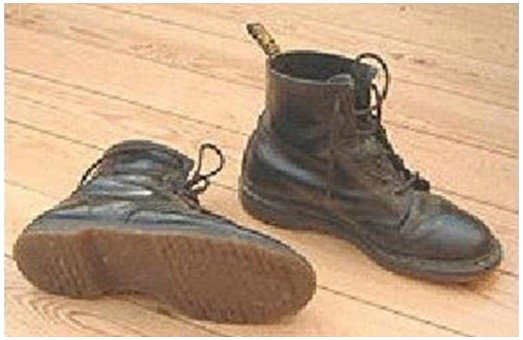
剪枝算法

- 对于给定的决策树T_0：
  - 计算所有内部节点的剪枝系数；
  - 查找最小剪枝系数的结点，剪枝得决策树T_k；
  - 重复以上步骤，直到决策树T_k只有1个结点；
  - 得到决策树序列$T_0T_1T_2...T_K$；
  - 使用验证样本集选择最优子树。
- 使用验证集做最优子树的标准，可以使用评价函数：

$$C(T)=\sum_{t\in leaf} N_t\cdot H(t)$$

Bootstraping

- Bootstraping的名称来自成语"pull up by your own bootstraps"，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法。



  - 注：Bootstrap本义是指高靴子口后面的悬挂物、小环、带子，是穿靴子时用手向上拉的工具。"pull up by your own bootstraps"即"通过拉靴子让自己上升"，意思是"不可能发生的事情"。后来意思发生了转变，隐喻"不需要外界帮助，仅依靠自身力量让自己变得更好"。

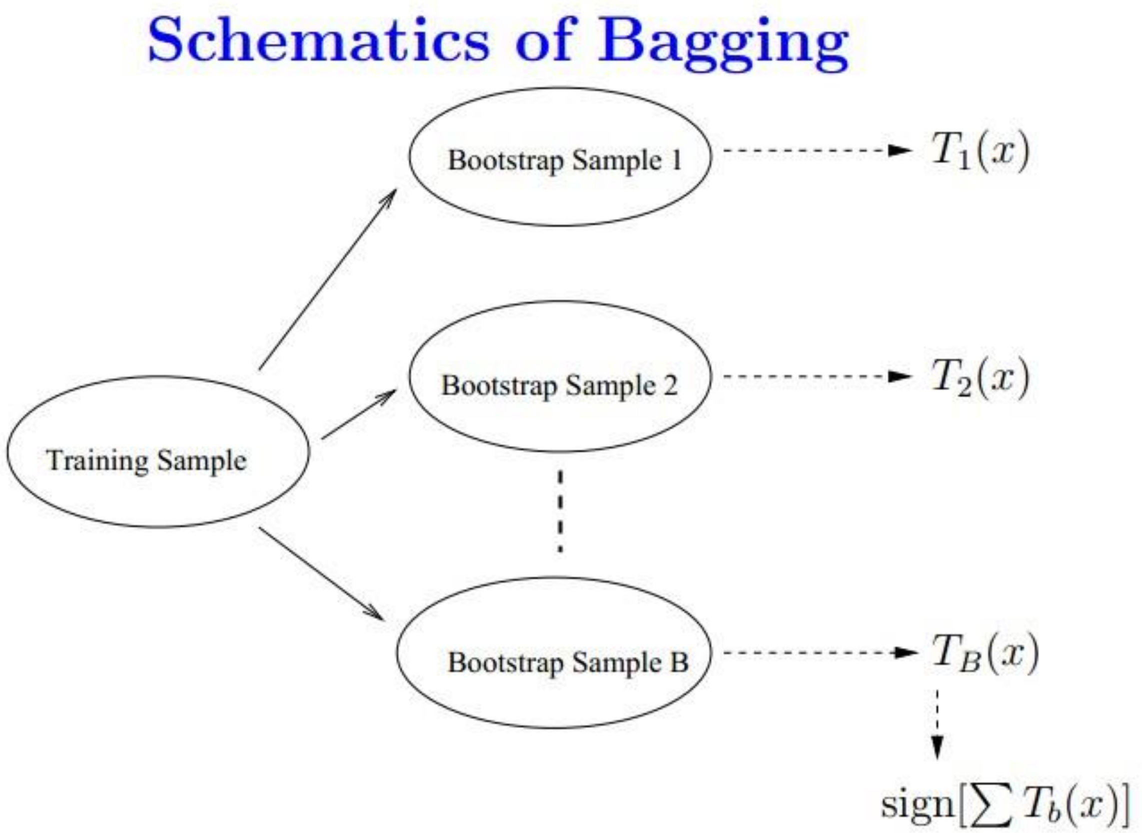
Bagging的策略

- bootstrap aggregation
- 从样本集中重采样(有重复的)选出n个样本
- 在所有属性上，对这n个样本建立分类器(ID3、C4.5、CART、SVM、Logistic回归等)
- 重复以上两步m次，即获得了m个分类器
- 将数据放在这m个分类器上，最后根据这m个分类器的投票结果，决定数据属于哪一类

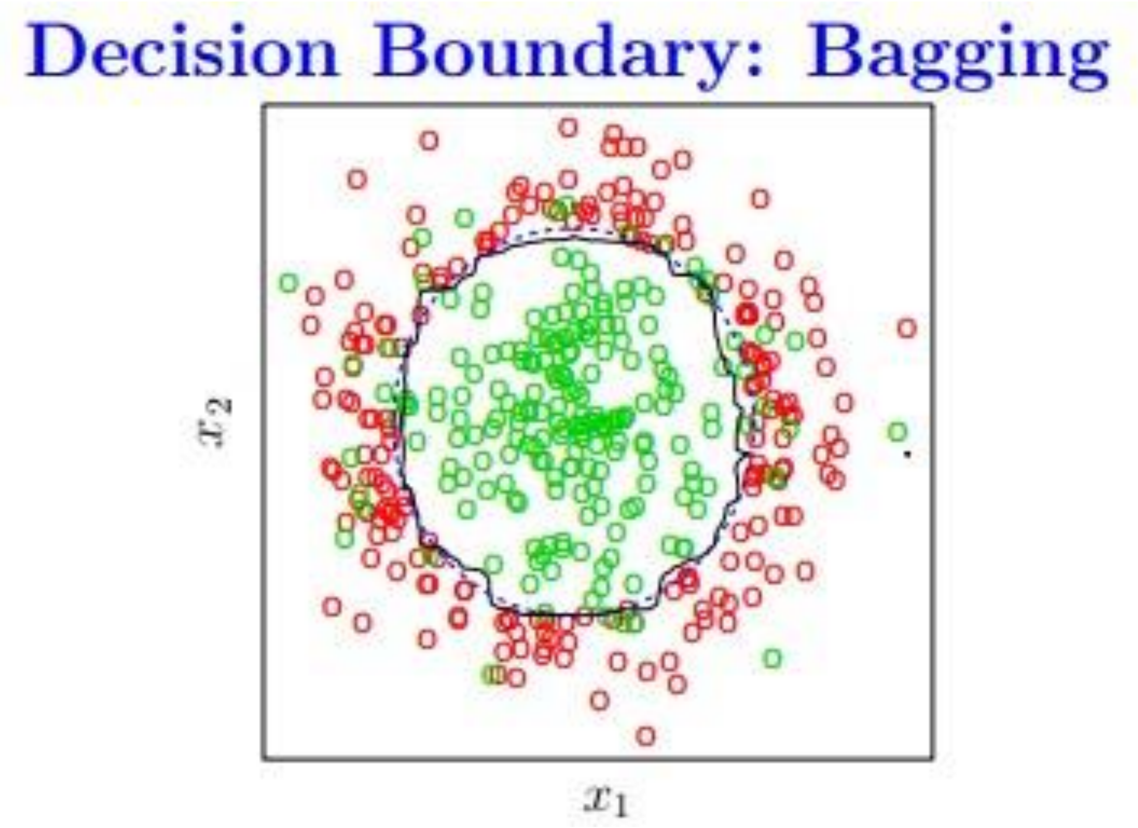
Another description of Bagging

Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n' n, by sampling examples from D uniformly and with replacement. By sampling with replacement, it is likely that some examples will be repeated in each D_i. If $n'=n$, then for large n the set D_i is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates.[1] This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

Bagging


Schematics of Bagging
Bootstrap Sample 1
Bootstrap Sample 2
Bootstrap Sample B
Training Sample
$T_1(x)$
$T_2(x)$
$T_B(x)$
$\text{sign}[\sum T_b(x)]$

Bagging的结果



随机森林

- 随机森林在bagging基础上做了修改。
  - 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，选择最佳分割属性作为节点建立CART决策树；
  - 重复以上两步m次，即建立了m棵CART决策树
  - 这m个CART形成随机森林，通过投票表决结果，决定数据属于哪一类

应用实例：Kinect

3.3. Randomized decision forests

Decision forests are considered effective multi-class classifiers. Forest is a group of T decision trees, where each split node is represented by a feature and a threshold τ. The procedure starts at the root node of a tree, evaluating equation 1 at each split node and branching left or right according to the comparison to threshold τ. The leaf node consists of a learned distribution $P_t(c|I,x)$ over body part label c, in the tree t. Figure IV illustrates the forest approach.

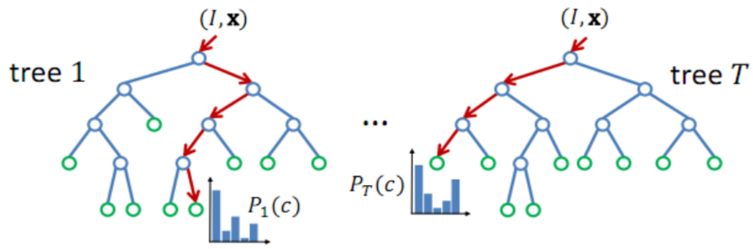
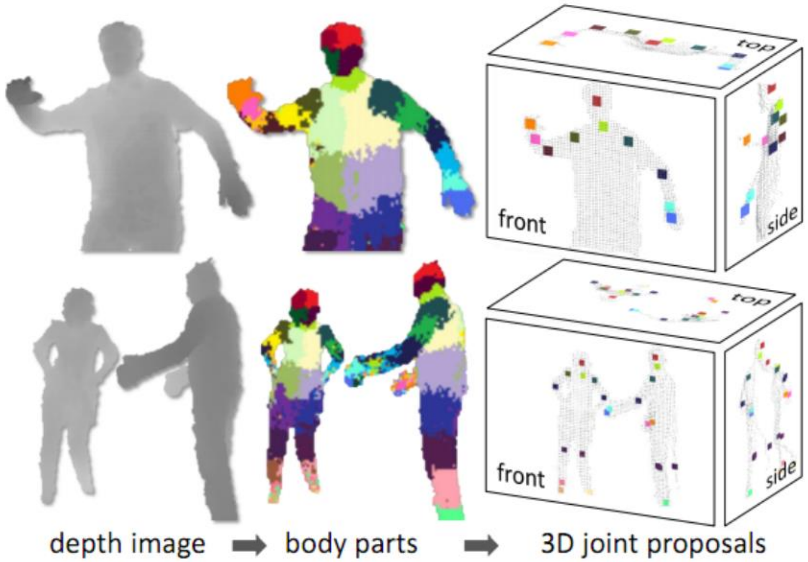


Figure IV. Decision forest.

The final classification is given by averaging all the distributions together in the forest. Equation 2 represents this classification.

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x) \quad (2)$$

The final classification is given by averaging all the distributions together in the forest. Equation 2 represents this classification.



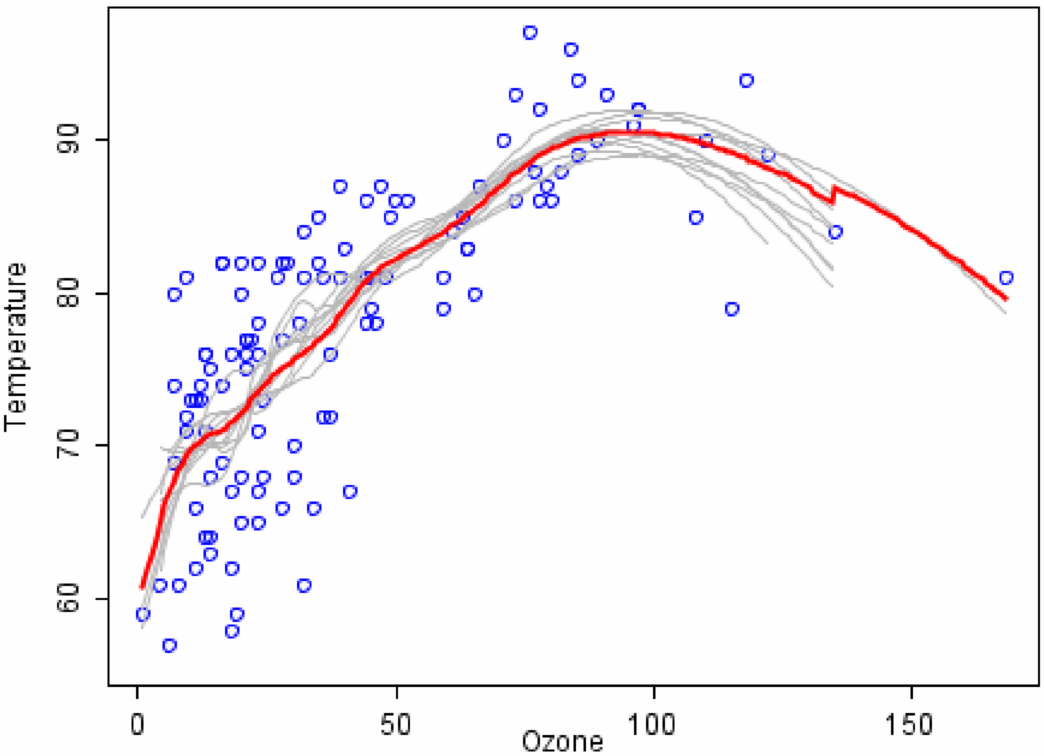
Real-Time Human Pose Recognition in Parts from Single Depth Images, Jamie Shotton etc,2001,

随机森林/Bagging和决策树的关系

- 当然可以使用决策树作为基本分类器
- 但也可以使用SVM、Logistic回归等其他分类器，习惯上，这些分类器组成的“总分类器”，仍然叫做随机森林。
- 举例
  - 回归问题

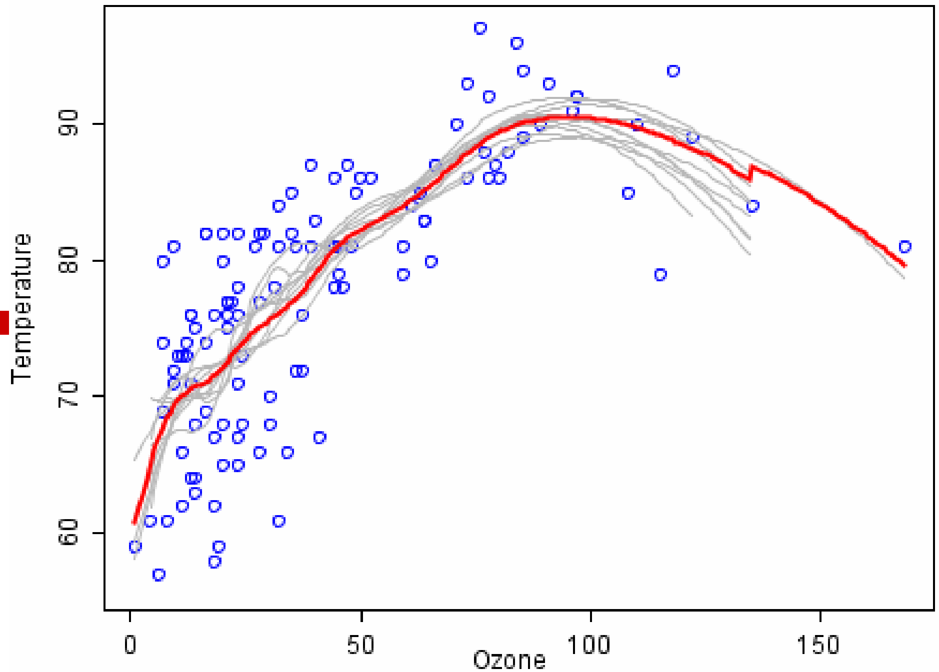
回归问题

- 离散点为臭氧(横轴)和温度(纵轴)的关系
- 试拟合变化曲线



使用Bagging

记原始数据为D，长度为N(即图中有N个离散点)



- 算法过程
  - 做100次bootstrap，每次得到的数据Di，Di的长度为N
  - 对于每一个Di，使用局部回归(LOESS)拟合一条曲线(图中灰色线是其中的10条曲线)
  - 将这些曲线取平均，即得到红色的最终拟合曲线
  - 显然，红色的曲线更加稳定，并且没有过拟合明显减弱

投票机制

- 简单投票机制
  - 一票否决(一致表决)
  - 少数服从多数
    - 有效多数(加权)
  - 阈值表决
- 贝叶斯投票机制

投票机制举例

- 假定有N个用户可以为X个电影投票(假定投票者不能给同一电影重复投票)，投票有1、2、3、4、5星共5档。
- 如何根据用户投票，对电影排序？
  - 本质仍然是分类问题：对于某个电影，有N个决策树，每个决策树对该电影有1个分类(1、2、3、4、5类)，求这个电影应该属于哪一类(可以是小数：分类问题变成了回归问题)

一种可能的方案

$$WR = \frac{v}{v+m}R + \frac{m}{v+m}C$$

- WR：加权得分(weighted rating)
- R：该电影的用户投票的平均得分(Rating)
- C：所有电影的平均得分
- v：该电影的投票人数(votes)
- m：排名前250名的电影的最低投票数
  - 根据总投票人数，250可能有所调整
  - 按照v=0和m=0分别分析

Code

```
def split(self, tree):
    f = self.select_feature()
    self.choose_value(f, tree)

def select_feature(self):   # 返回当前
    n = len(data[0])
    if rf:
        return random.randint(0,n-2)
    gini_f = 1          # gini指数最大是
    f = -1              # gini指数最小的
    for i in range(n-1):
        g = self.gini_feature(i)
        if gini_f > g:
            gini_f = g
            f = i
    return f
```

```
class TreeNode:
    def __init__(self):
        self.sample = []     # 该结点拥有哪些样本
        self.feature = -1    # 用几号特征划分
        self.value = 0       # 该特征的取值
        self.type = -1       # 该结点的类型
        self.left = -1       # 该节点的左孩子
        self.right = -1      # 该节点的右孩子
        self.gini = 0

    def gini_coefficient(self):
        types = {}
        for i in self.sample:
            type = data[i][-1]
            if types.has_key(type):
                types[type] += 1
            else:
                types[type] = 1
        pp = 0
        m = float(len(self.sample))
        for t in types:
            pp += (float(types[t]) / m) ** 2
        self.gini = 1-pp
        max_type = 0
        for t in types:
            if max_type < types[t]:
                max_type = types[t]
                self.type = t
```

Code

```python
def choose_value(self, f, tree):
    f_max = self.calc_max(f)
    f_min = self.calc_min(f)
    step = (f_max - f_min) / granularity
    if step == 0:
        return f_min
    x_split = 0
    g_split = 1
    for x in numpy.arange(f_min+step, f_max, step):
        if rf:
            x = random.uniform(f_min, f_max)
        g = self.gini_coefficient2(f, x)
        if g_split > g:
            g_split = g
            x_split = x
    if g_split < self.gini: # 分割后gini系数要变小才有意义
        self.value = x_split
        self.feature = f
        t = TreeNode()
        t.sample = self.choose_sample(f, x_split, True)
        t.gini_coefficient()
        self.left = len(tree)
        tree.append(t)
        t = TreeNode()
        t.sample = self.choose_sample(f, x_split, False)
        t.gini_coefficient()
        self.right = len(tree)
        tree.append(t)
```

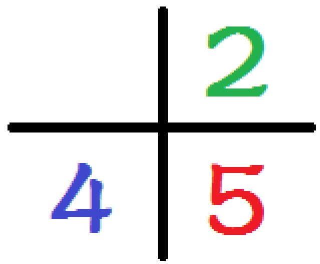
Code

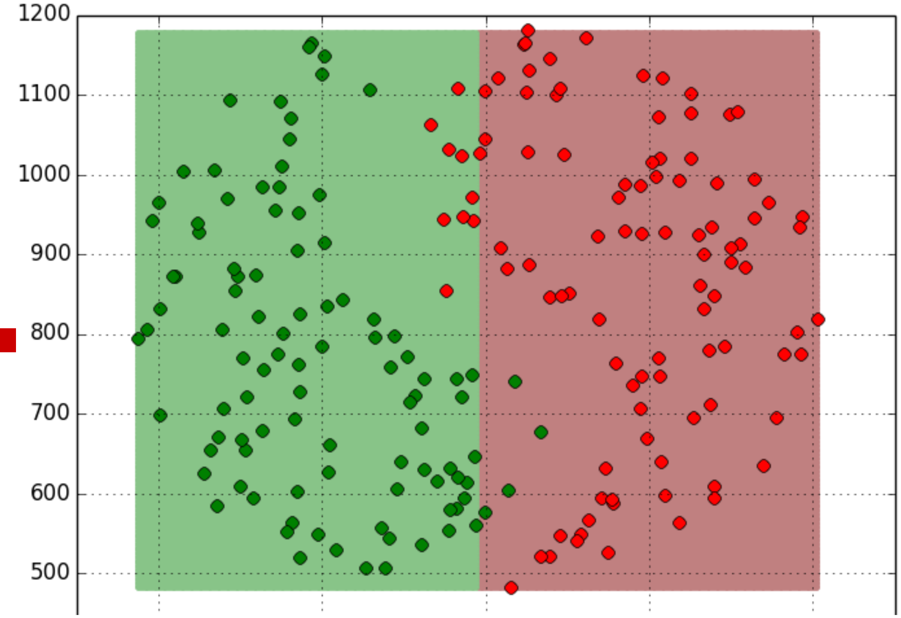
```python
def decision_tree():
    m = len(data)
    n = len(data[0])
    tree = []
    root = TreeNode()
    if rf:
        root.sample = random_select(alpha)
    else:
        root.sample = [x for x in range(m)]
    root.gini_coefficient()
    tree.append(root)
    first = 0
    last = 1
    for level in range(max_level):
        for node in range(first, last):
            tree[node].split(tree)
        first = last
        last = len(tree)
        print level+1, len(tree)
    return tree
```

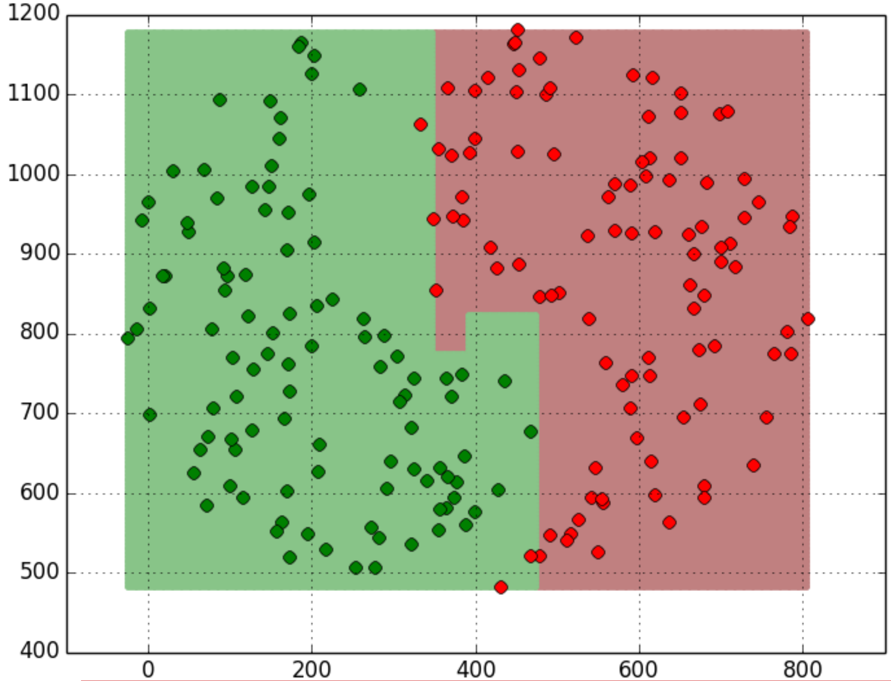
```python
def predict_tree(d, tree):
    node = tree[0]
    while node.left != -1 and node.right != -1:
        if d[node.feature] < node.value:
            node = tree[node.left]
        else:
            node = tree[node.right]
    return node.type
```

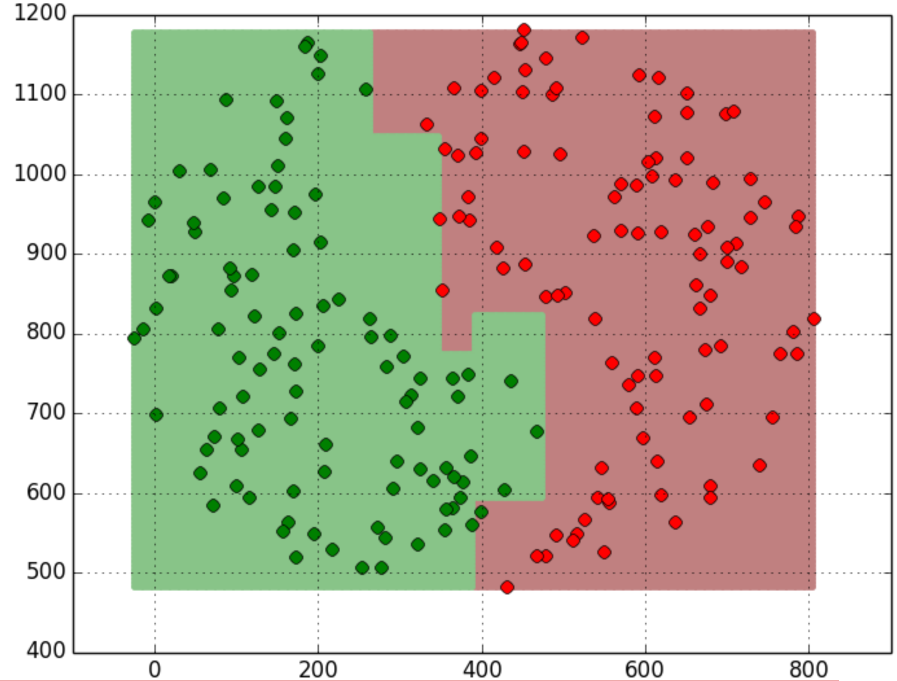
```python
def predict(d, forest):
    pd = {}
    for tree in forest:
        type = predict_tree(d, tree)
        if pd.has_key(type):
            pd[type] += 1
        else:
            pd[type] = 1
    number = 0
    type = 0.0
    for p in pd:
        if number < pd[p]:
            number = pd[p]
            type = p
    return type
```

决策树：Level

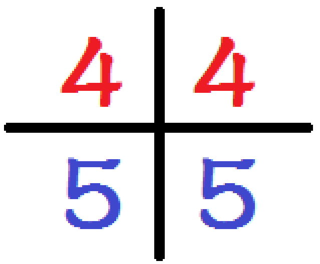

2
4
5

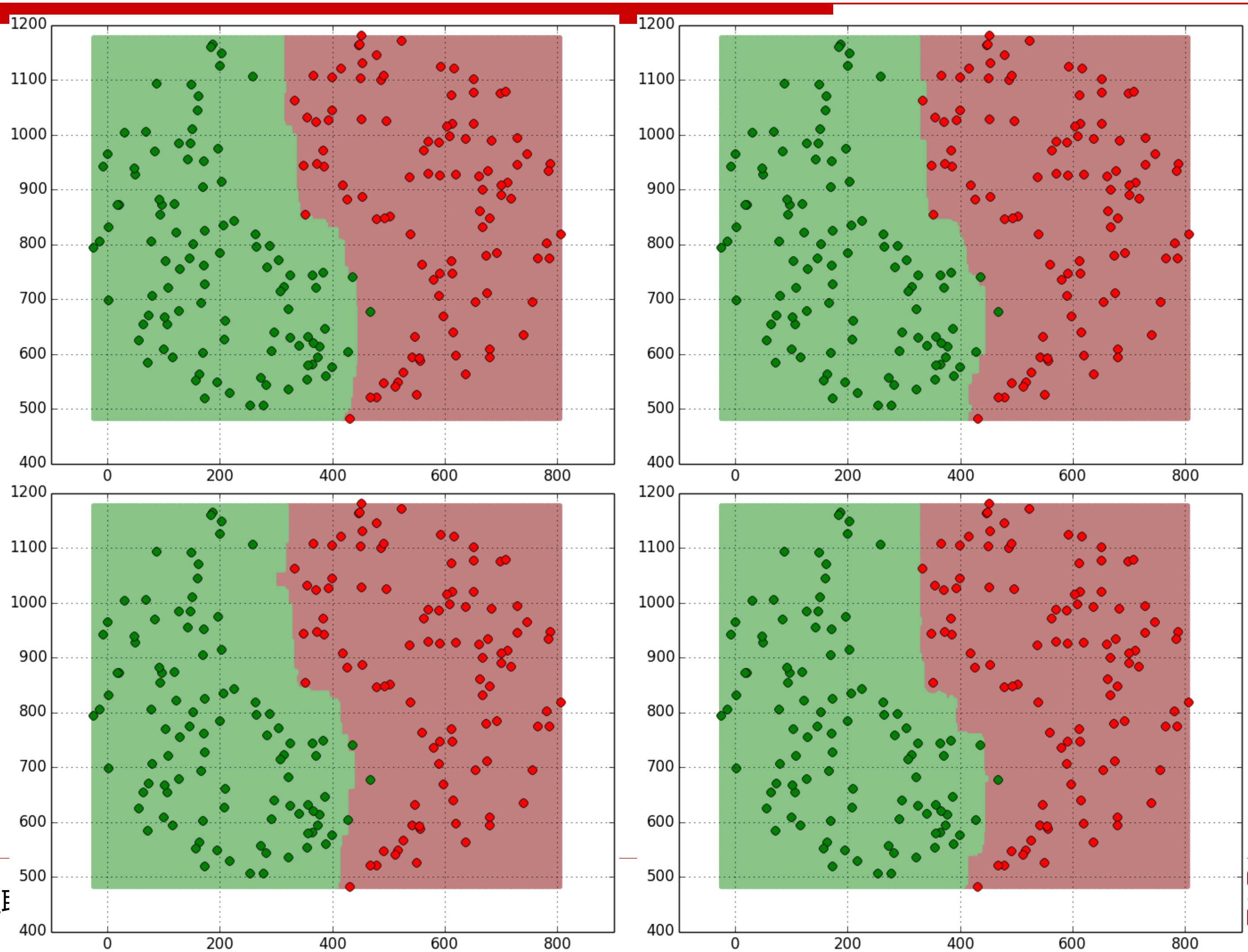

1200
1100
1000
900
800
700
600
500


1200
1100
1000
900
800
700
600
500
400
0
200
400
600
800


1200
1100
1000
900
800
700
600
500
400
0
200
400
600
800

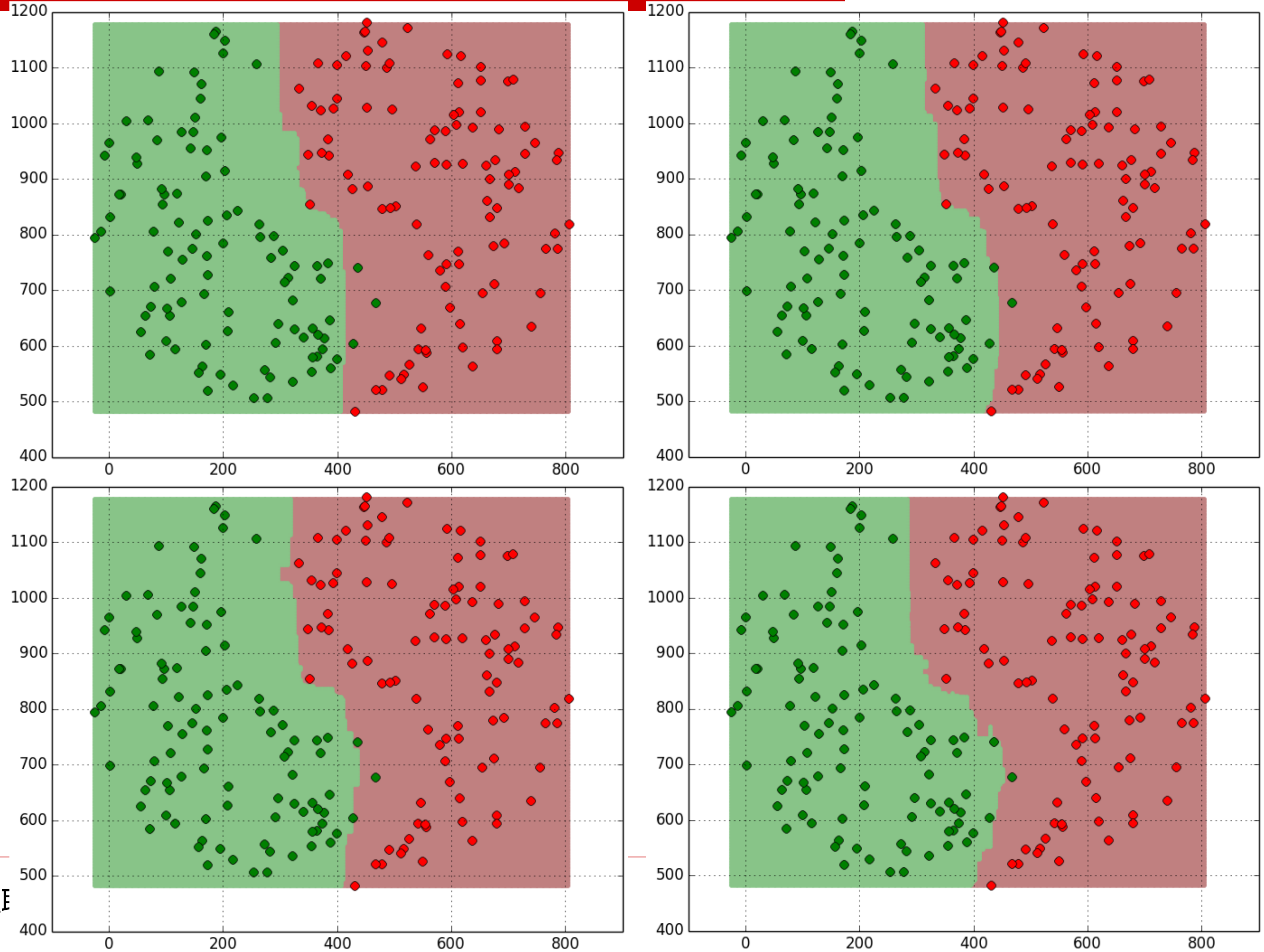
随机森林：30，重现





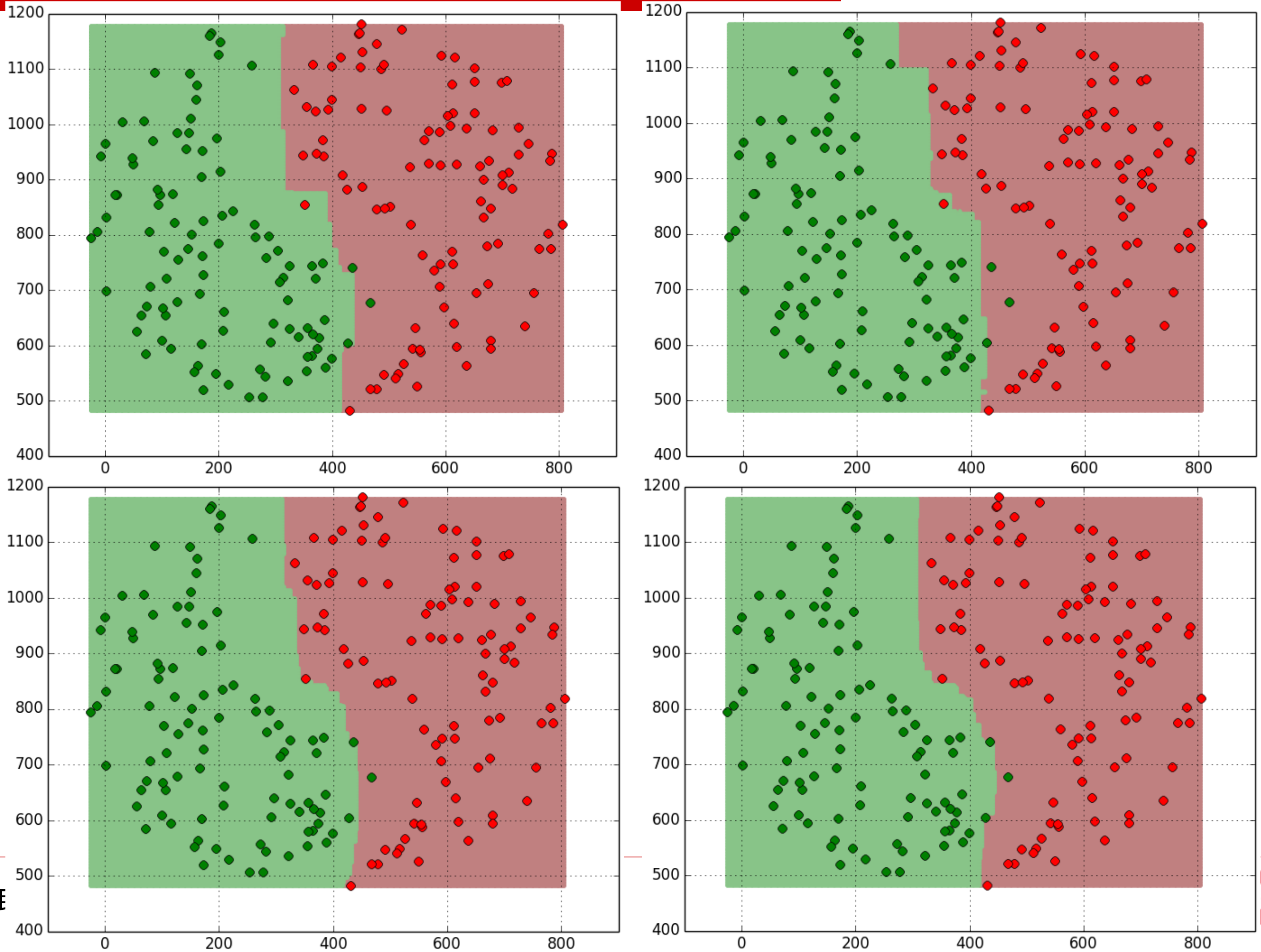
随机森林：30，Level

3 | 4
5 | 6



随机森林：4，Tree

10	20
30	40



互I

总结

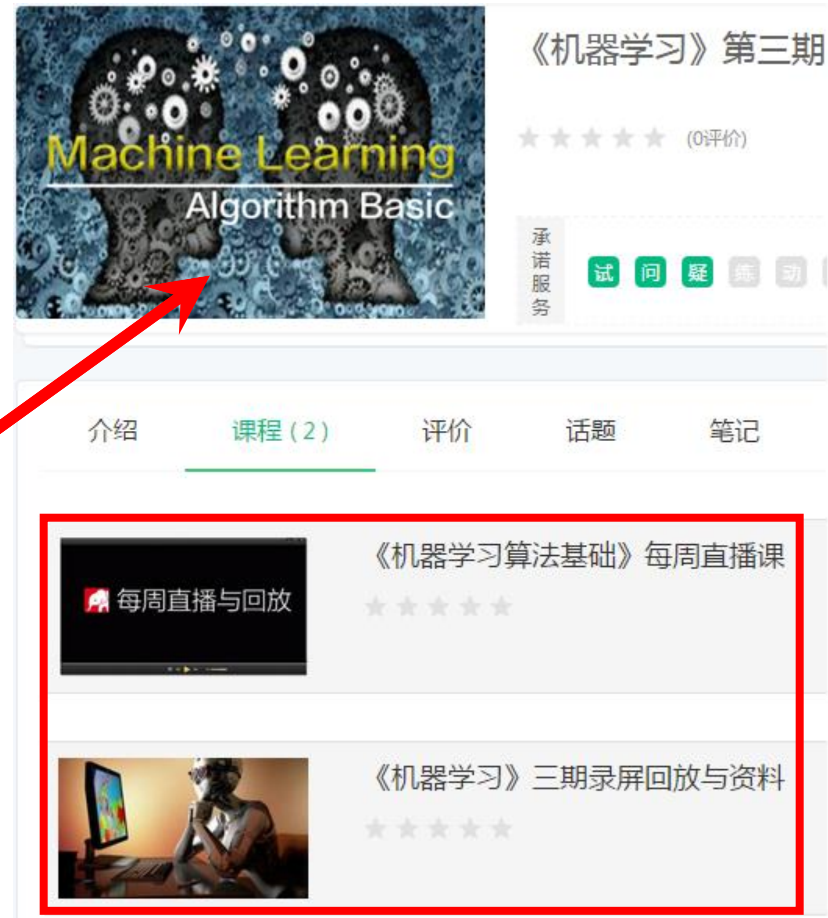
- 决策树/随机森林的代码清晰简单、训练时间非常快，在胜任分类问题的同时，往往也可以作为对数据分布探索的首要尝试算法。
- 随机森林的集成思想也可以用在其他分类器的设计中。
- 如果正负样本数量差别很大，如何处理？
- 思考：在得到新决策树后，对样本的权值进行合理的调整——分类正确的则降低权值，分类错误的则增大权值——是否可行？

参考文献

- Thomas M. Cover, Joy A. Thomas. *Elements of Information Theory*. 2006
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag. 2006
- 李航，统计学习方法，清华大学出版社，2012
- Jamie Shotton, Andrew Fitzgibbon, etc. *Real-Time Human Pose Recognition in Parts from Single Depth Images*. 2011

课程资源

- 直播课的入口
- 录播视频和讲义资料



感谢大家！

恳请大家批评指正！