中国科学院大学人工智能学院硕士课《模式识别》

决策树方法

张煦尧(xyz@nlpr.ia.ac.cn)

2019年12月18日

Top 10 algorithms in data mining

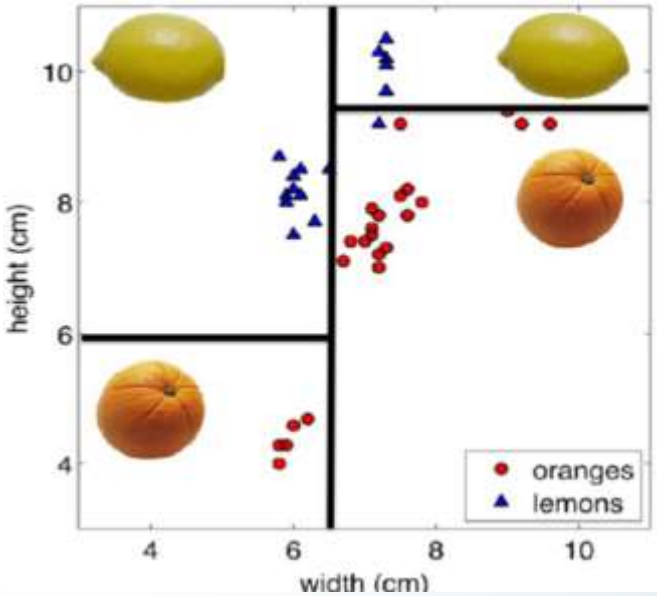
IEEE Int'l Conf. on Data Mining 2006, Hong Kong

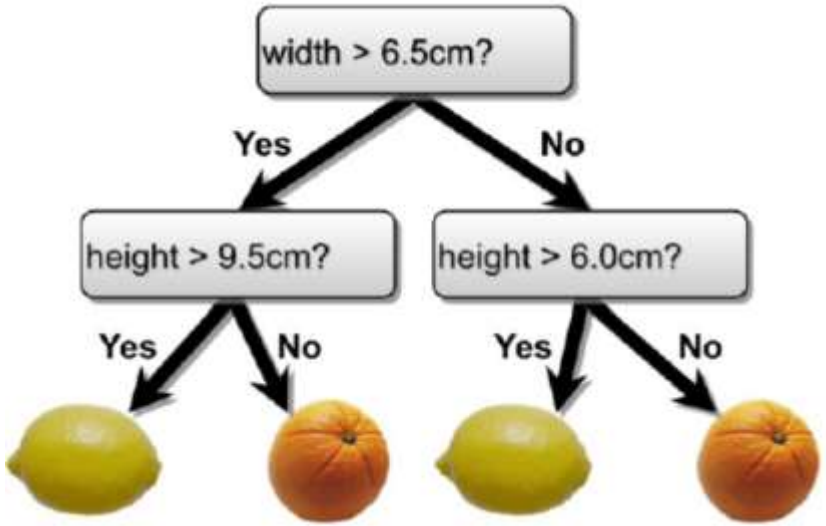
1. C4.5: decision trees classifiers
2. K-Means: simplest clustering algorithm
3. SVM: support vector machine for classification
4. Apriori: a seminal algorithm for finding frequent itemsets
5. EM: maximum likelihood estimation of mixture distribution
6. PageRank: search ranking algorithm using hyperlinks (Google)
7. AdaBoost: one of the most important ensemble methods
8. KNN: simplest classifier
9. Naïve Bayes: simplest Bayes, independent Bayes
10. CART: classification and regression Trees

决策树的基本概念

Another Classification Idea

- Classification models: Bayes decision rule, SVM, nearest neighbors … **Any other idea?**
- **Decision Tree**
  - Pick an attribute, do a simple test
  - Conditioned on a choice, pick another attribute, do another test
  - In the leaves, assign a class with majority vote
  - Do other branches as well





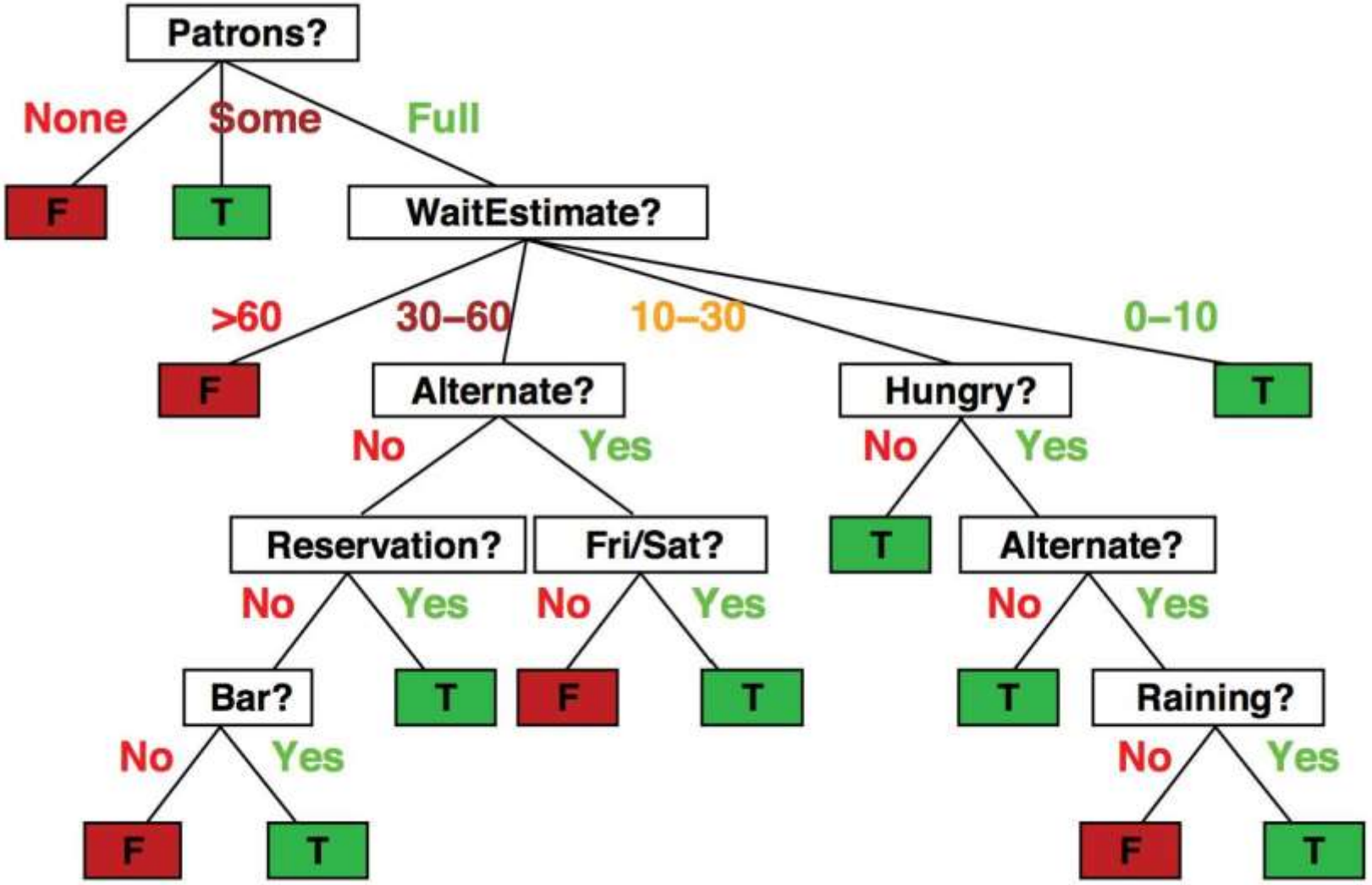
Example with Discrete Inputs

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Example with Discrete Inputs

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

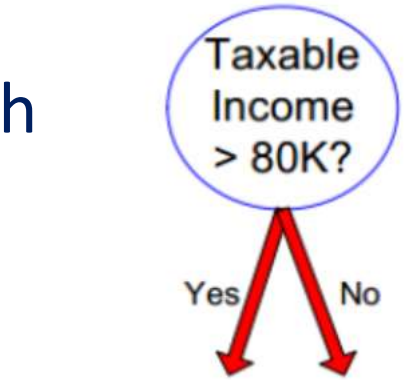
Decision Tree



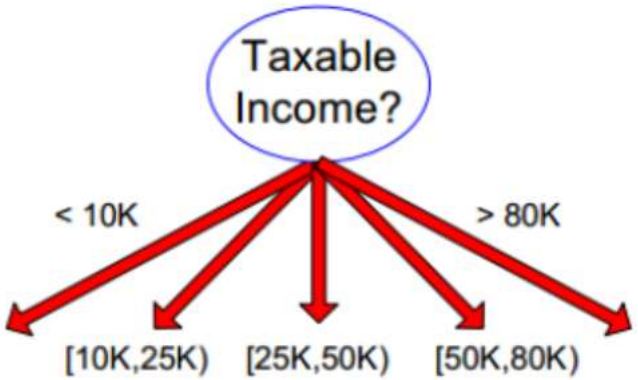
Splitting Continuous Attributes

How to split continuous attributes such as Age, Income etc

- **Discretization** to form an ordinal categorical attribute
  - Static: discretize once at the beginning
  - Dynamic: find ranges by equal interval bucketing, equal frequency bucketing, percentiles, clustering etc
- **Binary Decision**: (A < v)or (A >= v)
  - Consider all possible split and find the best cut
  - Often, computationally intensive



(i) Binary split

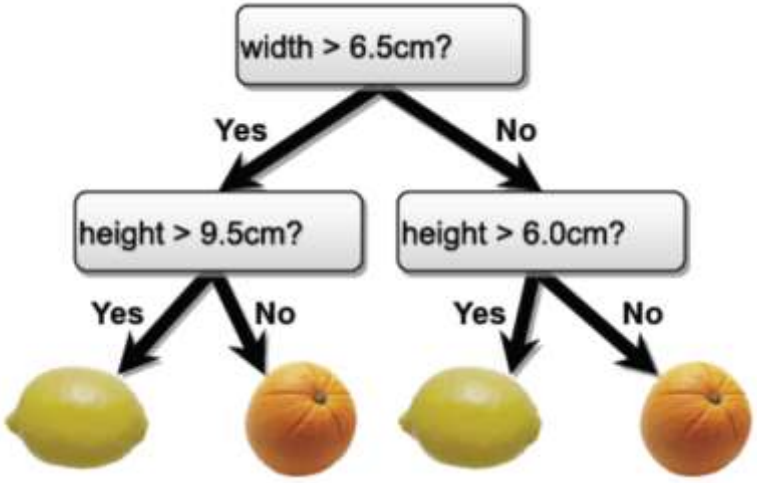


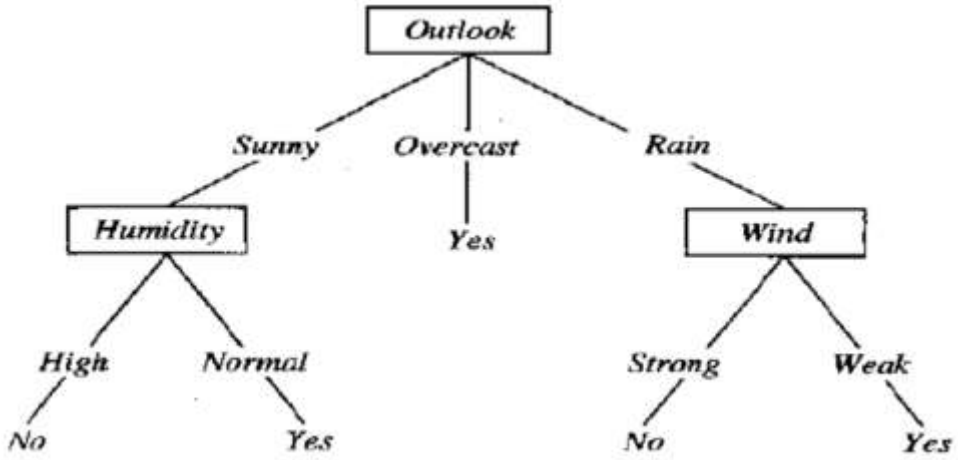
(ii) Multi-way split

Decision Tree

A decision tree is a tree in which

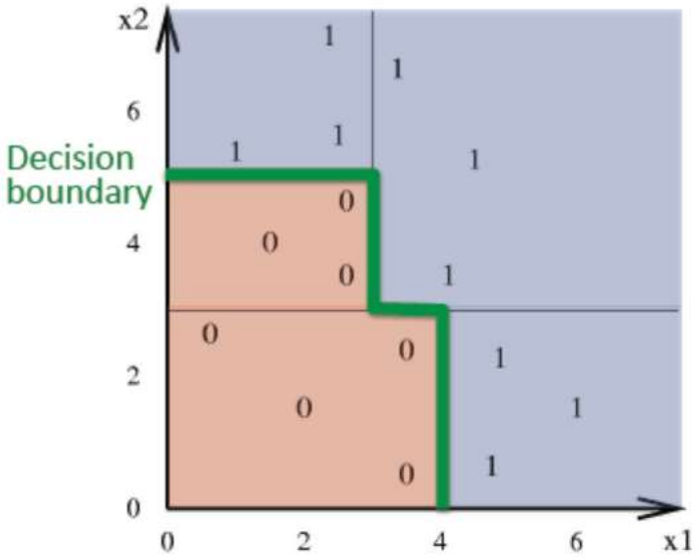
- each ***internal node*** represents a choice between a number of alternatives
- each ***leaf node*** represents a classification or decision

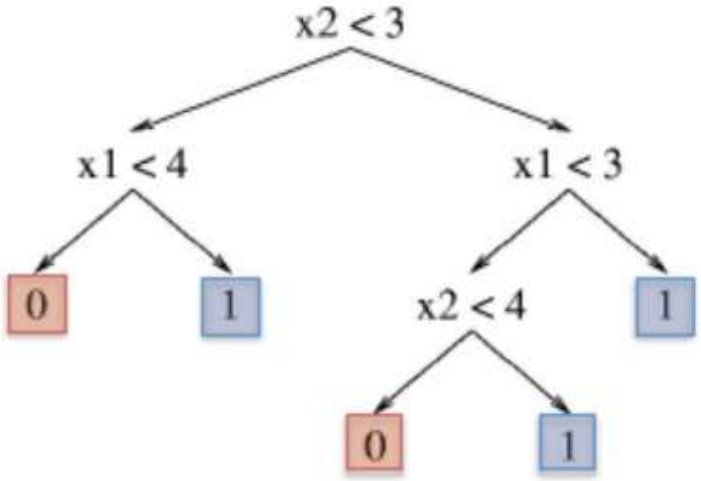




Decision Tree: Boundary

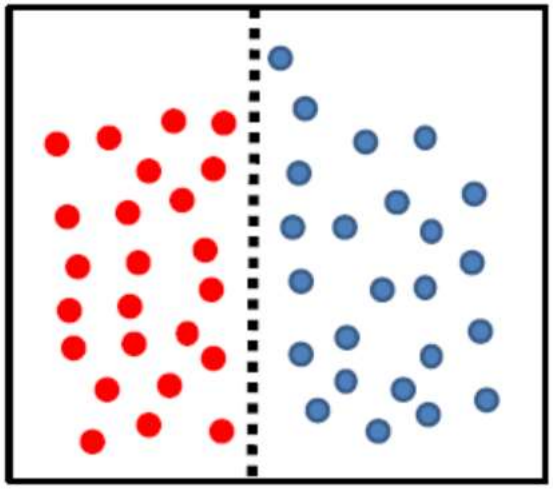
- Decision trees divide the feature space into axis-parallel (hyper-) rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels

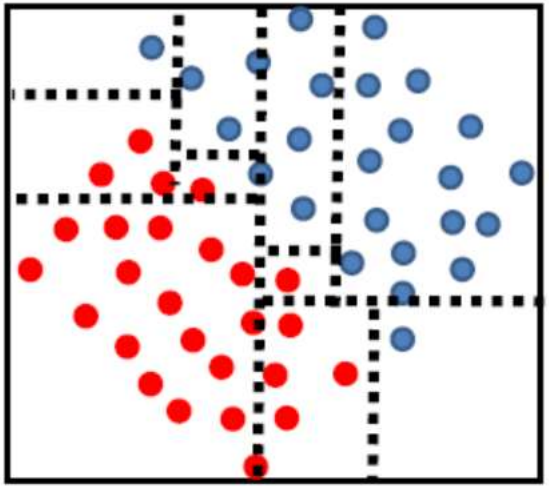




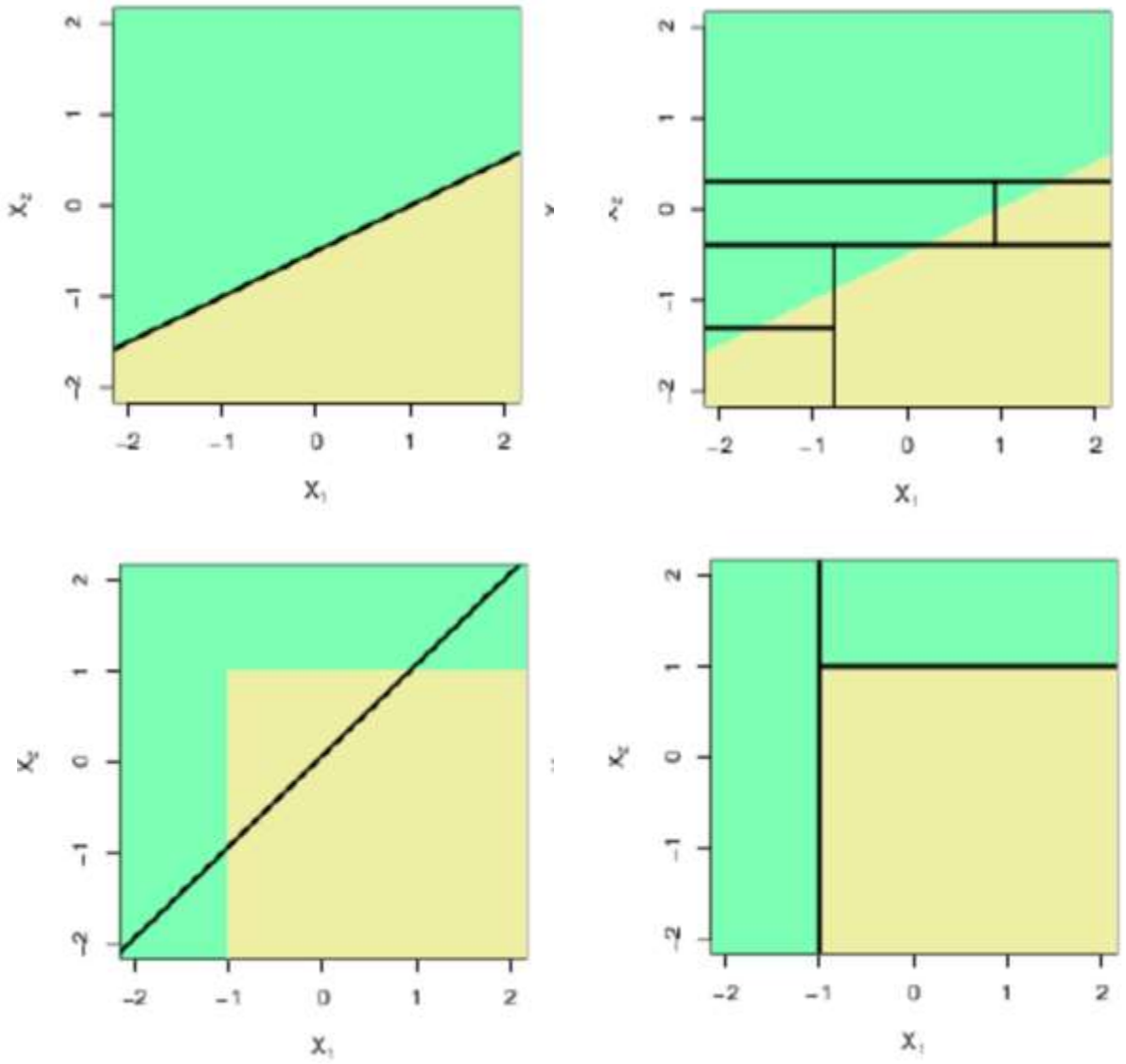
Decision Tree: Boundary

- Tree complexity is highly dependent on data geometry in the feature space





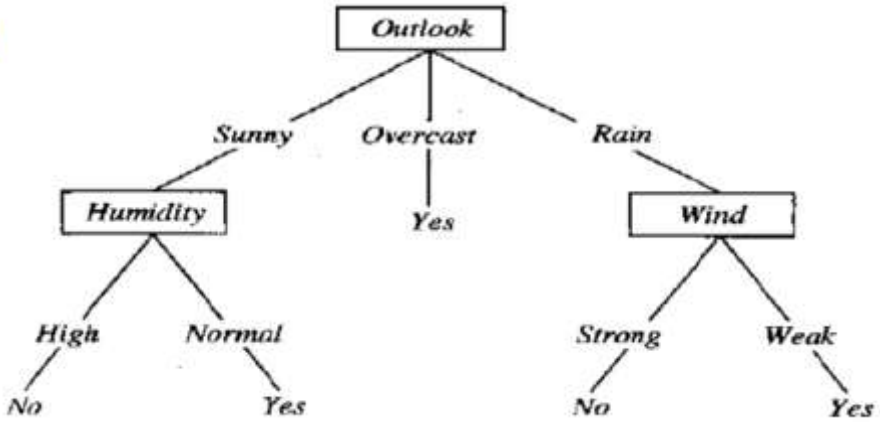
Trees vs Linear Model



基本概念

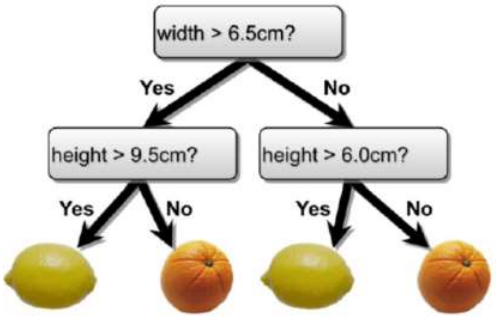
- 每一个节点表示对样本实例的某个属性值进行测试，该节点后相连接的各条路径上的值表示该属性的可能取值（二叉，三叉，…）
- 每一个叶子产生一条规则，规则由根到该叶子的路径上所有节点的条件，规则的结论是叶子上标注的结论（决策，分类，判断）
- 决策树规则代表实例属性值约束的合取（交集）的析取（并集）式。从树根到树叶的每一条路径对应一组属性测试的合取，树本身对应这些合取的析取

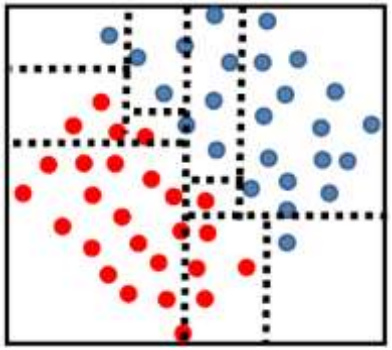
$$(Outlook = sunny \wedge Humidity = normal)$$
$$\vee (Outlook = overcast)$$
$$\vee (Outlook = rain \wedge Wind = weak)$$



Decision Tree: Algorithm

- Choose an attribute on which to descend at each level.
- Condition on earlier (higher) choices.
- Generally, restrict only **one dimension** at a time.
- Declare an output value when you get to the bottom
- In the orange/lemon example, we only split each dimension once, but that is not required





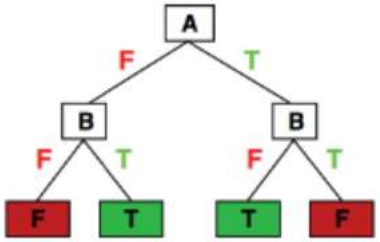
Decision Tree: Classification and Regression

- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- **Classification tree**:
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$
- **Regression tree**:
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$

Expressiveness

- Discrete-input, discrete-output case:
  - Decision trees can express any function of the input attributes.
  - E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Continuous-input, continuous-output case:
  - Can approximate any function arbitrarily closely
- Trivially, there is a consistent decision tree for any training set (one path to leaf for each example) but it probably won't generalize to new examples (**consider 1-NN**)
- Need some kind of regularization to ensure more compact decision trees

Advantages and Disadvantages

- Very easy to explain to people
- Some people believe that decision trees more closely mirror human decision-making
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small)
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Inexpensive to construct
- Extremely fast at classifying new data
- Unfortunately, trees generally do not have the same level of predictive accuracy as other classifiers

Learning Decision Trees

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recursive
    - Iterative Dichotomizer" (ID3)
    - C4.5 (ID3+improvements)

- What is the best attribute?
- We use **information theory** to guide us

信息论相关

Choosing a Good Attribute

- Which attribute is better to split on?
- Use counts at leaves to define probability distributions, so we can measure **uncertainty**

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Flip Two Different Coins

Sequence 1:

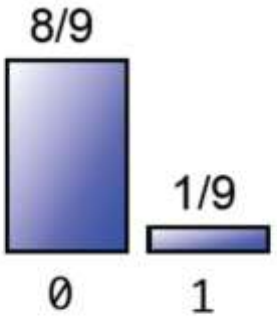
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:

0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

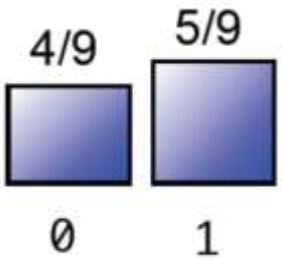
Entropy

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$



$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

- **"Low Entropy"**
  - Distribution of variable has many peaks and valleys
  - Histogram has many lows and highs
  - Values sampled from it are more predictable



$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- **"High Entropy"**:
  - Variable has a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable

熵与不确定性

在信息论与概率统计中，熵（entropy）是表示随机变量不确定性的度量．设 X 是一个取有限个值的离散随机变量，其概率分布为

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

则随机变量 X 的熵定义为

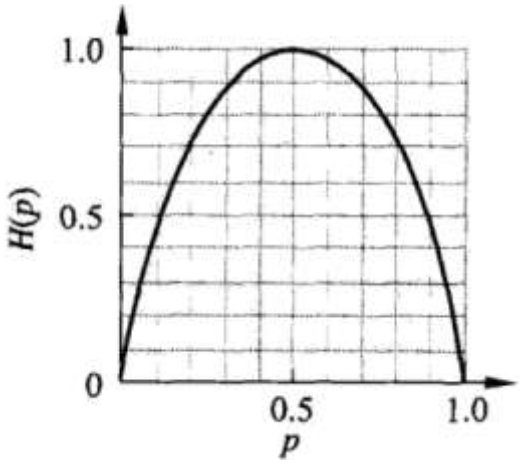
$$H(X)=-\sum_{i=1}^{n}p_i\log p_i \tag{5.1}$$

当随机变量只取两个值，例如 1，0 时，即 X 的分布为

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

熵为

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



分布为贝努利分布时熵与概率的关系

条件熵

设有随机变量(X,Y)，其联合概率分布为

$$P(X=x_i,Y=y_j)=p_{ij}\,,\quad i=1,2,\cdots,n\,;\quad j=1,2,\cdots,m$$

条件熵$H(Y\mid X)$表示在已知随机变量X的条件下随机变量Y的不确定性. 随机变量X给定的条件下随机变量Y的条件熵（conditional entropy）$H(Y\mid X)$，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

$$H(Y\mid X)=\sum_{i=1}^{n}p_iH(Y\mid X=x_i) \tag{5.5}$$

这里，$p_i=P(X=x_i)$，$i=1,2,\cdots,n$.

信息增益（information gain）表示得知特征X的信息而使得类Y的信息的不确定性减少的程度.

信息增益（互信息）

定义 5.2（信息增益） 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

$$g(D,A)=H(D)-H(D|A) \tag{5.6}$$

一般地，熵 $H(Y)$ 与条件熵 $H(Y|X)$ 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.

信息增益算法

根据信息增益准则的特征选择方法是：对训练数据集（或子集）D，计算其每个特征的信息增益，并比较它们的大小，选择信息增益最大的特征.

算法 5.1（信息增益的算法）

输入：训练数据集 D 和特征 A；

输出：特征 A 对训练数据集 D 的信息增益 $g(D,A)$.

（1）计算数据集 D 的经验熵 $H(D)$

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|} \tag{5.7}$$

（2）计算特征 A 对数据集 D 的经验条件熵 $H(D|A)$

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|} \tag{5.8}$$

（3）计算信息增益

$$g(D,A)=H(D)-H(D|A) \tag{5.9}$$ ■

信息增益计算举例

例 5.2 对表 5.1 所给的训练数据集 D，根据信息增益准则选择最优特征.

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

信息增益计算举例

解 首先计算经验熵 $H(D)$.

$$H(D)=-\frac{9}{15}\log_2\frac{9}{15}-\frac{6}{15}\log_2\frac{6}{15}=0.971$$

然后计算各特征对数据集 D 的信息增益. 分别以 A_1，A_2，A_3，A_4 表示年龄、有工作、有自己的房子和信贷情况 4 个特征，则

(1)

$$\begin{aligned}g(D,A_1)&=H(D)-\left[\frac{5}{15}H(D_1)+\frac{5}{15}H(D_2)+\frac{5}{15}H(D_3)\right]\\&=0.971-\left[\frac{5}{15}\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)\right.\\&\quad\left.+\frac{5}{15}\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)+\frac{5}{15}\left(-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5}\right)\right]\\&=0.971-0.888=0.083\end{aligned}$$

这里 D_1，D_2，D_3 分别是 D 中 A_1（年龄）取值为青年、中年和老年的样本子集. 类似地，

信息增益计算举例

（2）

$$g(D,A_2)=H(D)-\left[\frac{5}{15}H(D_1)+\frac{10}{15}H(D_2)\right]$$

$$=0.971-\left[\frac{5}{15}\times 0+\frac{10}{15}\left(-\frac{4}{10}\log_2\frac{4}{10}-\frac{6}{10}\log_2\frac{6}{10}\right)\right]=0.324$$

（3）

$$g(D,A_3)=0.971-\left[\frac{6}{15}\times 0+\frac{9}{15}\left(-\frac{3}{9}\log_2\frac{3}{9}-\frac{6}{9}\log_2\frac{6}{9}\right)\right]$$

$$=0.971-0.551=0.420$$

（4）

$$g(D,A_4)=0.971-0.608=0.363$$

最后，比较各特征的信息增益值．由于特征 A_3（有自己的房子）的信息增益值最大，所以选择特征 A_3 作为最优特征．■

ID3与C4.5学习算法

ID3算法

- ID3是Quinlan于1986年提出的，它的提出开创了决策树算法的先河，而且是国际上最早的决策树方法，在该算法中，引入了信息论中熵的概念，利用分割前后的熵来计算信息增益，作为判别能力的度量。

ID3 算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树．具体方法是：从根结点（root node）开始，对结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为结点的特征，由该特征的不同取值建立子结点；再对子结点递归地调用以上方法，构建决策树；直到所有特征的信息增益均很小或没有特征可以选择为止．最后得到一个决策树．ID3 相当于用极大似然法进行概率模型的选择．

ID3算法

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

（1）若 D 中所有实例属于同一类 C_k，则 T 为单结点树，并将类 C_k 作为该结点的类标记，返回 T；

（2）若 $A=\varnothing$，则 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；

（3）否则，按算法 5.1 计算 A 中各特征对 D 的信息增益，选择信息增益最大的特征 A_g；

（4）如果 A_g 的信息增益小于阈值 ε，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；

（5）否则，对 A_g 的每一可能值 a_i，依 $A_g=a_i$ 将 D 分割为若干非空子集 D_i，将 D_i 中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T，返回 T；

（6）对第 i 个子结点，以 D_i 为训练集，以 $A-\{A_g\}$ 为特征集，递归地调用步 (1) ～步 (5)，得到子树 T_i，返回 T_i. ■

ID3算法举例

例 5.3 对表 5.1 的训练数据集，利用 ID3 算法建立决策树.

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

ID3算法举例

解 利用例 5.2 的结果，由于特征 A_3（有自己的房子）的信息增益值最大，所以选择特征 A_3 作为根结点的特征．它将训练数据集 D 划分为两个子集 D_1（A_3 取值为“是”）和 D_2（A_3 取值为“否”）．由于 D_1 只有同一类的样本点，所以它成为一个叶结点，结点的类标记为“是”.

对 D_2 则需从特征 A_1（年龄），A_2（有工作）和 A_4（信贷情况）中选择新的特征．计算各个特征的信息增益：

$$g(D_2, A_1) = H(D_2) - H(D_2 \mid A_1) = 0.918 - 0.667 = 0.251$$
$$g(D_2, A_2) = H(D_2) - H(D_2 \mid A_2) = 0.918$$
$$g(D_2, A_4) = H(D_2) - H(D_2 \mid A_4) = 0.474$$

选择信息增益最大的特征 A_2（有工作）作为结点的特征．由于 A_2 有两个可能取值，从这一结点引出两个子结点：一个对应“是”（有工作）的子结点，包含 3 个样本，它们属于同一类，所以这是一个叶结点，类标记为“是”；另一个是对应“否”（无工作）的子结点，包含 6 个样本，它们也属于同一类，所以这也是一个叶结点，类标记为“否”.

ID3算法举例

这样生成一个如图 5.5 所示的决策树．该决策树只用了两个特征（有两个内部结点）．

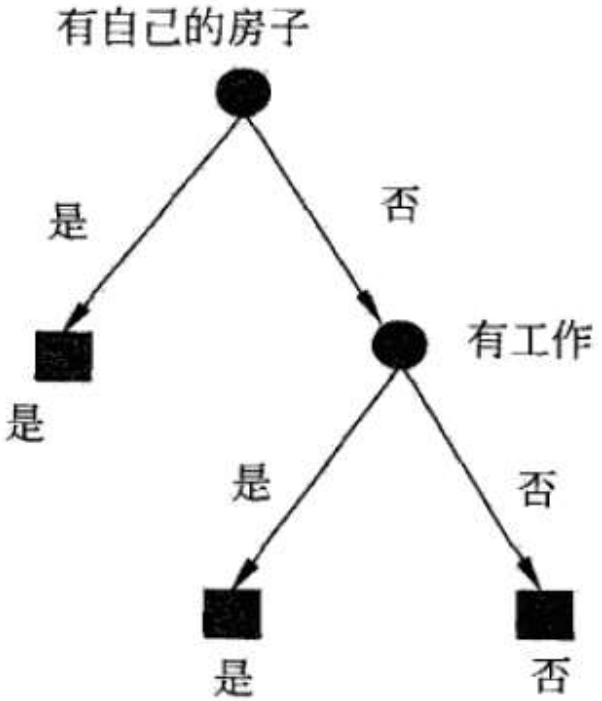


图 5.5　决策树的生成

C4.5算法

- 属性选择度量（分裂规则）
  - 信息增益（ID3）

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \qquad Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

    - 偏向于具有大量值的属性。在训练集中，某个属性所取的不同值的个数越多，那么越有可能拿它来作为分裂属性。

  - 信息增益率 (C4.5)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

C4.5 vs ID3

- C4.5中使用信息增益率（Gain-ratio）
- ID3算法使用信息增益（Info-Gain）
- Info-Gain在面对类别较少的离散数据时效果较好，之前的outlook、temperature等数据都是离散数据，而且每个类别都有一定数量的样本，这种情况下使用ID3与C4.5的区别并不大。
- 但如果面对**连续的数据**（如体重、身高、年龄、距离等），或者每列数据没有明显的类别之分（最极端的例子的该列所有数据都独一无二），在这种情况下，ID3算法倾向于把每个数据自成一类（将每一个样本都分到一个节点当中去），程序会倾向于选择这种划分，这样划分效果极差。
- 为了解决这个问题，引入了信息增益率（Gain-ratio）的概念，**减轻了划分行为本身的影响**。

C4.5 vs ID3

- 对于取值多的属性，尤其一些连续型数值，比如两条地理数据的距离属性，这个单独的属性就可以划分所有的样本，使得所有分支下的样本集合都是“纯的”（最极端的情况是每个叶子节点只有一个样本）。
- 一个属性的信息增益越大，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。
- 对于ID（就比如是姓名），用信息增益划分，每一个名字都是一个类
- 所以如果是取值更多的属性，更容易使得数据更“纯”（尤其是连续型数值），其信息增益更大，决策树会首先挑选这个属性作为树的顶点。
- 结果训练出来的形状是一棵庞大且深度很浅的树，这样的划分是极为不合理的。
- C4.5使用了信息增益率，在信息增益的基础上除了一项split information,来惩罚值更多的属性
- 采取信息增益方式，若某属性取值数目远大于类别数量时信息增益可以变得很大，对训练集分割尽管不错，但泛化能力同时也会变差。这种分割方式导致在每个分割空间内数据较纯净，甚至能使熵为0，但未利用其它实际上可能更加有效的分类信息。信息增益率引入了分裂信息，取值数目多的属性分裂信息也会变大，将增益除以分裂信息，再加上一些额外操作，可以有效控制信息增益过大的问题。

C4.5算法

- C4.5算法是机器学习算法中的一种分类决策树算法，其核心算法是ID3算法，C4.5继承了ID3算法的优点，并在以下几方面对ID3算法进行了改进：

- 用**信息增益率**来选择属性，克服了用**信息增益**选择属性时偏向选择取值多的属性的不足
- 在树构造过程中进行剪枝
- 能够完成对连续属性的离散化处理
- 能够对不完整数据进行处理

Classification and Regression Trees (CART, C&RT)

CART

分类与回归树（classification and regression tree，CART）模型由 Breiman 等人在 1984 年提出，是应用广泛的决策树学习方法．CART 同样由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归．以下将用于分类与回归的树统称为决策树．

CART 是在给定输入随机变量 X 条件下输出随机变量 Y 的条件概率分布的学习方法．CART 假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支．这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布．

CART算法

（1）决策树生成：基于训练数据集生成决策树，生成的决策树要尽量大；

（2）决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时用损失函数最小作为剪枝的标准.

决策树的生成就是递归地构建二叉决策树的过程.对回归树用平方误差最小化准则，对分类树用基尼指数（Gini index）最小化准则，进行特征选择，生成二叉树.

假设 X 与 Y 分别为输入和输出变量，并且 Y 是连续变量，给定训练数据集

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

考虑如何生成回归树.

CART回归树

算法 5.5（最小二乘回归树生成算法）

输入：训练数据集 D；

输出：回归树 $f(x)$.

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式 (5.21) 达到最小值的对 (j,s).

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x\mid x^{(j)}\leqslant s\},\quad R_2(j,s)=\{x\mid x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1,R_2,\cdots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

CART分类树

分类树用基尼指数选择最优特征，同时决定该特征的最优二值切分点.

定义 5.4（基尼指数） 分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为

$$\text{Gini}(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{5.22}$$

对于二类分类问题，若样本点属于第 1 个类的概率是 p，则概率分布的基尼指数为

$$\text{Gini}(p)=2p(1-p) \tag{5.23}$$

对于给定的样本集合 D，其基尼指数为

$$\text{Gini}(D)=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2 \tag{5.24}$$

这里，C_k 是 D 中属于第 k 类的样本子集，K 是类的个数.

CART分类树

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分，即

$$D_1 = \{(x,y) \in D \mid A(x) = a\}, \quad D_2 = D - D_1$$

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$\text{Gini}(D,A) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2) \tag{5.25}$$

基尼指数 $\text{Gini}(D)$ 表示集合 D 的不确定性，基尼指数 $\text{Gini}(D,A)$ 表示经 $A=a$ 分割后集合 D 的不确定性．基尼指数值越大，样本集合的不确定性也就越大，这一点与熵相似．

CART算法

输入：训练数据集 D，停止计算的条件；

输出：CART 决策树.

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

（1）设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数. 此时，对每一个特征 A，对其可能取的每个值 a，根据样本点对 $A=a$ 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 两部分，利用式（5.25）计算 $A=a$ 时的基尼指数.

（2）在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点. 依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去.

（3）对两个子结点递归地调用（1），（2），直至满足停止条件.

（4）生成 CART 决策树.

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值（样本基本属于同一类），或者没有更多特征.

基尼指数

- $gini(t)=1-\Sigma_j p^2(j|t)$

p(red|t)=11/25,
p(green|t)=14/25,
gini(t)=1-0.194-0.314=0.492

$p(red|t_1)=2/13$,
$p(green|t_1)=11/13$,
$gini(t_1)=1 - 0.024 - 0.726 = 0.25$

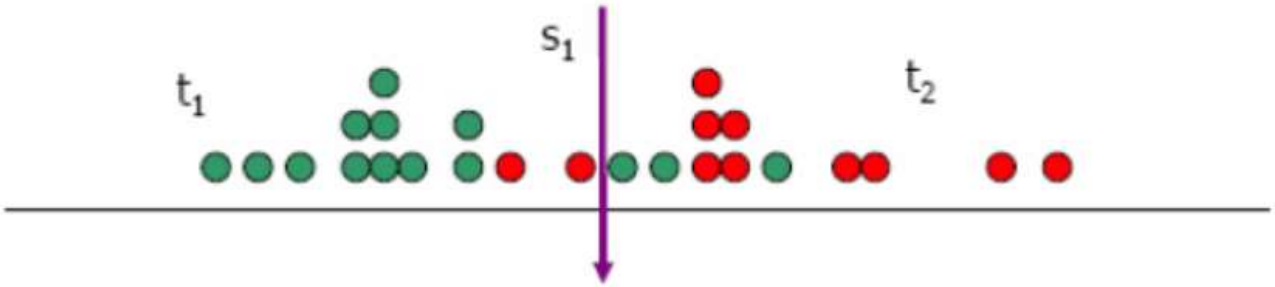
$p(red|t_2)=9/12$,
$p(green|t_2)=3/12$,
$gini(t_2)=1 - 0.563 - 0.063 = 0.374$

Weighted average = 0.25*(13/25) +0.374*(12/25) = 0.31

划分的优度

基尼指数的变化量：

$$= \text{gini}(t) - (n_1/n) * \text{gini}(t_1) - (n_2/n) * \text{gini}(t_2)$$



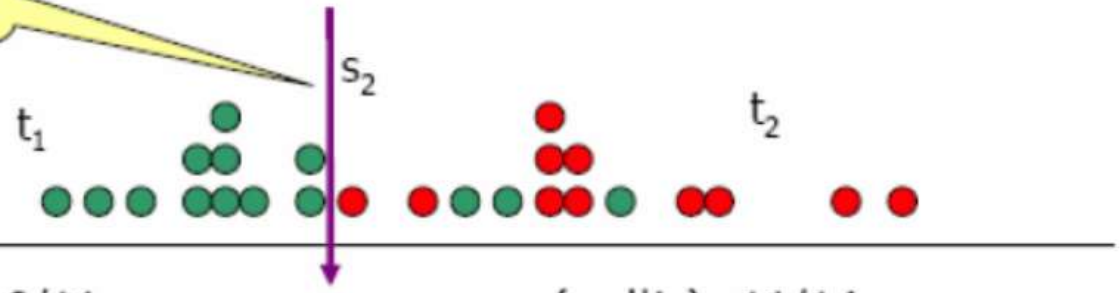
$$\text{Improvement}(s_1) = 0.492 - 0.31 = 0.182$$

$$gini(t)=1-\Sigma_j p^2(j|t)$$

p(red|t)=11/25,
p(green|t)=14/25,
gini(t)=1-0.194-0.314=0.492

数据
混杂度

另一个
划分



$p(red|t_1)=0/11$,
$p(green|t_1)=11/11$,
$gini(t_1) = 1 - 0.0 - 1.0 = 0.0$

$p(red|t_2)=11/14$,
$p(green|t_2)=3/14$,
$gini(t_2) = 1 - 0.617 - 0.046 = 0.337$

Weighted average = 0.0*(11/25) +0.337*(14/25) = 0.189

$Improvement(s_2) = 0.492 - 0.189 = 0.303$

$Improvement(s_1) = 0.492 - 0.31 = 0.182$

S_2 是更好
的划分

基尼指数划分的特点

- 基尼指数关注的目标变量里面最大的类，它试图找到一个划分把它和其它类别区分开来。
- 完美的系列划分将会得到k个纯粹的子节点，每一个节点对应目标变量的一个类别。
- 如果误分类代价因素被加入，基尼指数试图把代价最大的类别区分开来。

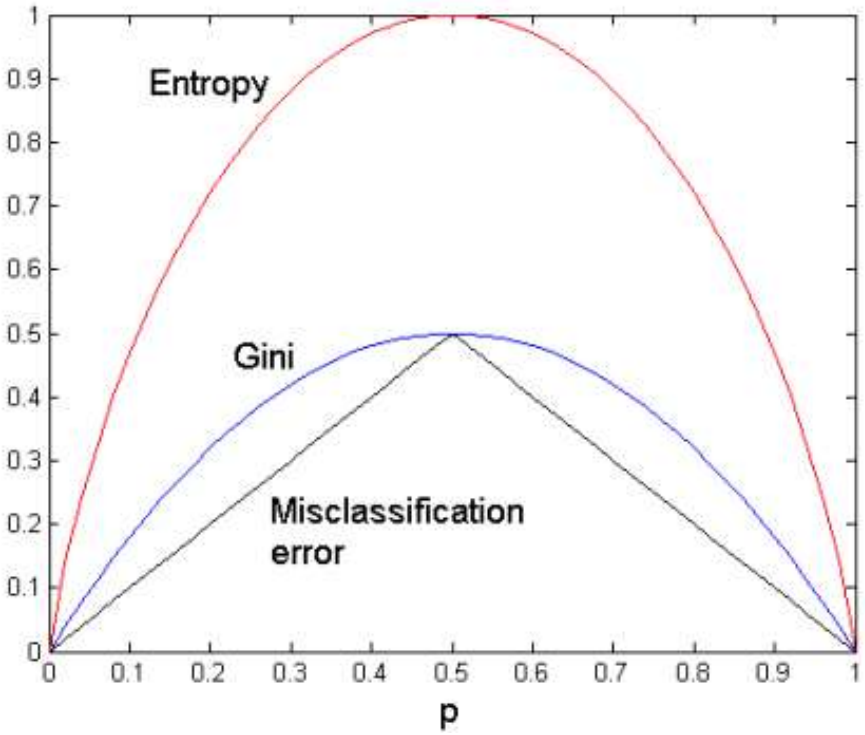
不同评价标准

For a 2-class problem:

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

$$Error(t) = 1 - \max_i P(i \mid t)$$



剪枝

在终止条件被满足，划分停止之后，下一步是剪枝：

- 给树剪枝就是剪掉“弱枝”，弱枝指的是在验证数据上误分类率高的树枝
- 为树剪枝会增加训练数据上的错误分类率，但精简的树会提高新记录上的预测能力
- **剪掉的是最没有预测能力的枝**

Decision Tree Overfitting

Noisy Data

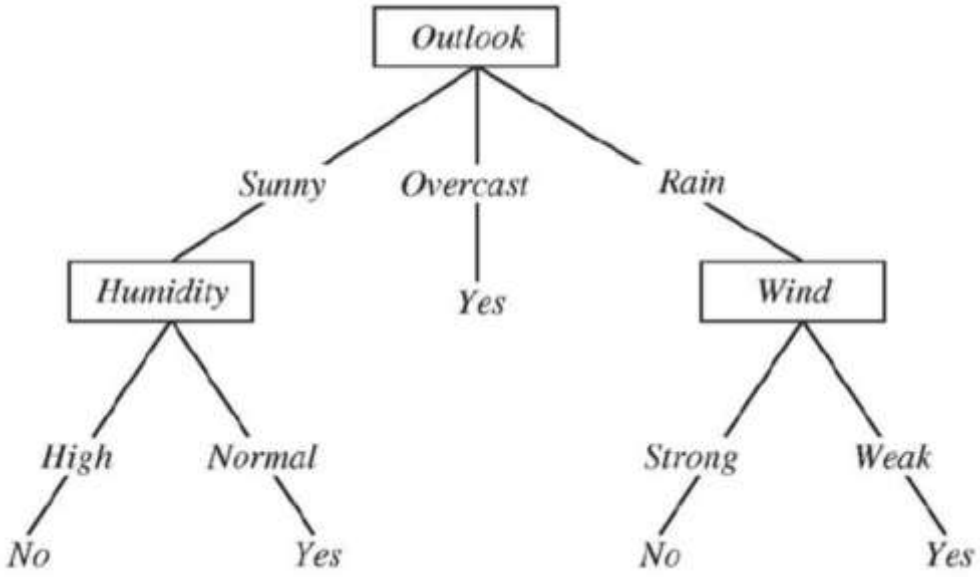
Many kinds of **noise** can occur in the examples

- **Erroneous** training data
  - two examples have same attribute/value pairs, but different classifications
  - feature noise
    - some values of attributes are incorrect because of errors in data acquisition process or preprocessing phase
  - label noise
    - instance was labeled incorrectly (+ instead of -)

➔ **means training error not guaranteed to be 0**

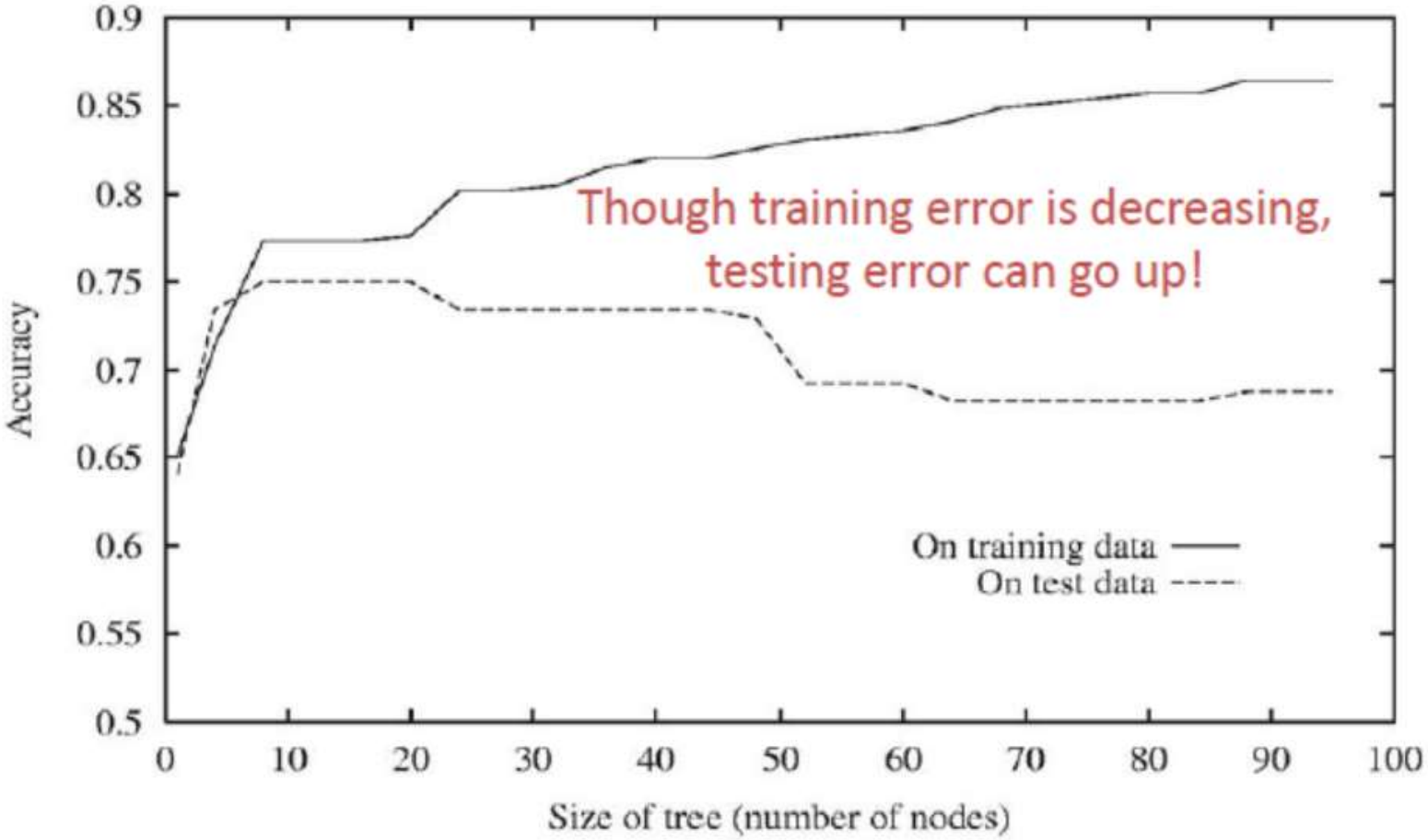
Overfitting in Decision Tree

- Consider adding a noisy training example to the following tree:



- **What would be the effect of adding:**
  - **(outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No)?**

Overfitting in Decision Tree



Overfitting

Much more significant sources of "noise"

- Some attributes are **irrelevant** to decision making process
  - if hypothesis space has many dimensions (large # of attributes), may find **meaningless regularity** in data that is irrelevant to true, important, distinguishing features

- Target function is **non-deterministic** in attributes
  - in general, we cannot measure all variables needed to do perfect prediction → target function is not uniquely determined by attribute values
  - if too little training data, even a reasonable hypothesis will "overfit"

Avoid Overfitting

How can we avoid overfitting?

- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)

Decision Tree Pruning

- Prune while building tree (**stopping early**)
- Prune after building tree (**post pruning**)

How to select ”best” tree

- Statistical tests
- Measure performance over separate validation set
  - Split data into training and validation sets
  - Grow tree with max depth d based on training set
  - Evaluate each tree on validation set
  - Pick tree with best performance

Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same

- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Post-Pruning (Reduced-Error Pruning)

- Grow decision tree to its entirety
- Split data into training and validation set
- Do until further pruning is harmful:
  - Evaluate impact on validation set of pruning each possible node (plus those below it)
  - Greedily remove the one that most improves validation set accuracy
- Produces smallest version of most accurate subtree.

Rule Post-Pruning

- Convert tree to equivalent set of rules
- Prune each rule independently of others
- Sort final rules into desired sequence for use

- Perhaps most frequently used method (e.g, C4.5)

Missing Values

- Examples are classified as usual
  - if we are lucky, attributes with missing values are not tested by the tree
- If an attribute with a missing value needs to be tested:
  - split the instance into fractional instances (pieces)
  - one piece for each outgoing branch of the node
  - a piece going down a branch receives a weight proportional to the popularity of the branch
  - weights sum to 1
- Info gain or gain ratio work with fractional instances
  - use sums of weights instead of counts
- During classification, split the instance in the same way
  - Merge probability distribution using weights of fractional instances

Random Forests (随机森林)

主要作者

- Leo Breiman was a distinguished **statistician** at the University of California, Berkeley.
- Breiman‘s work helped to bridge the gap between statistics and computer science, particularly in the field of **machine learning**.
- His most important contributions were his work on **classification and regression trees (CART)** and **ensembles of trees fit to bootstrap samples**.
- Bootstrap aggregation was given the name **bagging** by Breiman.
- Another of Breiman's ensemble approaches is the **random forest**.

随机森林的通俗理解

- Random Forest，顾名思义，Random就是**随机抽取**，Forest就是说这里不止一棵树，而由**一群决策树组成的一片森林**，连起来就是用随机抽取的方法训练出一群决策树来完成分类任务。

- RF用了**两次随机**抽取，一次是对**训练样本的随机抽取**；另一次是对**变量（特征）的随机抽取**。这主要是为了解决样本数量有限的问题。

- RF的核心是**由弱变强思想**的运用。每棵决策树由于只用了部分变量、部分样本训练而成，可能单个的分类准确率并不是很高。但是当一群这样的决策树组合起来分别对输入数据作出判断时，可以带来较高的准确率。有点类似于俗语 三个臭皮匠顶个诸葛亮。

- 随机森林有**两个重要参数**：
  - 一是树节点预选的变量个数；
  - 二是随机森林中树的个数

随机森林的思想来源

在说明random forest的算法之前，我先了解了一下它的思想来源，主线条可以由下面这个发展线来表示。

PAC→Bootstrap→Bagging→Random Forest←CART

来源1: PAC

PAC（Probably Approximately Correct）

- Kearns和Valiant提出的一种学习模型。
- 在该模型中，若存在一个多项式级的学习算法来识别一组概念，并且识别正确率很高，那么这组概念是强学习算法；而如果学习算法识别一组概念的正确率仅比随机猜测略好，那么这组概念是弱学习算法。
- 如果可以将**弱学习算法提升成强学习算法**，那么我们就只要找到一个弱学习算法，然后把它提升成强学习算法，而不必去找通常情况下很难获得的强学习算法。

来源2: Bootstrap

- 根据PAC由弱得到强的思想，统计学著名学者Bradley Efron在1979年提出了Bootstraps算法，这个名字来自于成语“pull up by your own bootstraps”，意思是依靠自己的资源，称为**自助法**。

- 它的思想就是**当样本数量不大，分布情况未知时，可以从原始样本中随机抽取的多个样本情况（弱学习）来估计原样本真实的分布情况**。它是非参数统计中一种重要的估计统计量方差进而进行区间估计的统计方法。

- 其基本步骤如下：
  - ①从原始数据集中，有放回地抽样一定数量的样本
  - ②根据抽出的样本计算给定的统计量T
  - ③重复上述N次（一般大于1000），得到N个统计量T
  - ④计算上述N个统计量T的样本方差，得到统计量的方差

来源2: Bootstrap (续)

- 这里举例说明其中一种最常用的方法：0.632自助法。
- 假设给定的数据集包含d个样本。该数据集有放回地抽样d次，产生d个样本的训练集。
- 原数据中的某些样本很可能在该样本集中出现多次
- 显然每个样本被选中的概率是1/d
- 因此未被选中的概率就是(1-1/d)
- 这样一个样本在训练集中没出现的概率就是d次都未被选中的概率，即(1-1/d)^d。当d趋于无穷大时，这一概率就将趋近于e^-1=0.368
- 所以留在训练集中的样本大概就占原来数据集的63.2%。

来源3: Bagging

- Bagging又叫bootstrap aggregation，是Breiman在1993年提出的方法，第一步就是根据Bootstrap进行抽样。

- 基本的步骤：
  - ①从样本集中用Bootstrap采样选出n个样本
  - ②在所有属性上，对这n个样本建立分类器（CART or SVM or ...）
  - ③重复以上两步m次，i.e.建立 m个分类器（CART or SVM or ...）
  - ④将数据放在这m个分类器上跑，最后vote看到底分到哪一类

- 这种方法可以大大降低每个分类器的不稳定性，从而带来较高的预测准确率。从这个方法再往下发展就是随机森林了。

From Bagging to Random Forest

- 随机森林是以决策树为基本分类器的一个集成学习模型，它包含多个由Bagging集成学习技术训练得到的决策树。

- Random Forests不同的是：在Bagging的基础上，使用一种**改进的树学习算法**，在每个候选分裂的学习过程中，选择特征值的一个随机子集。有时被称为“feature bagging”。

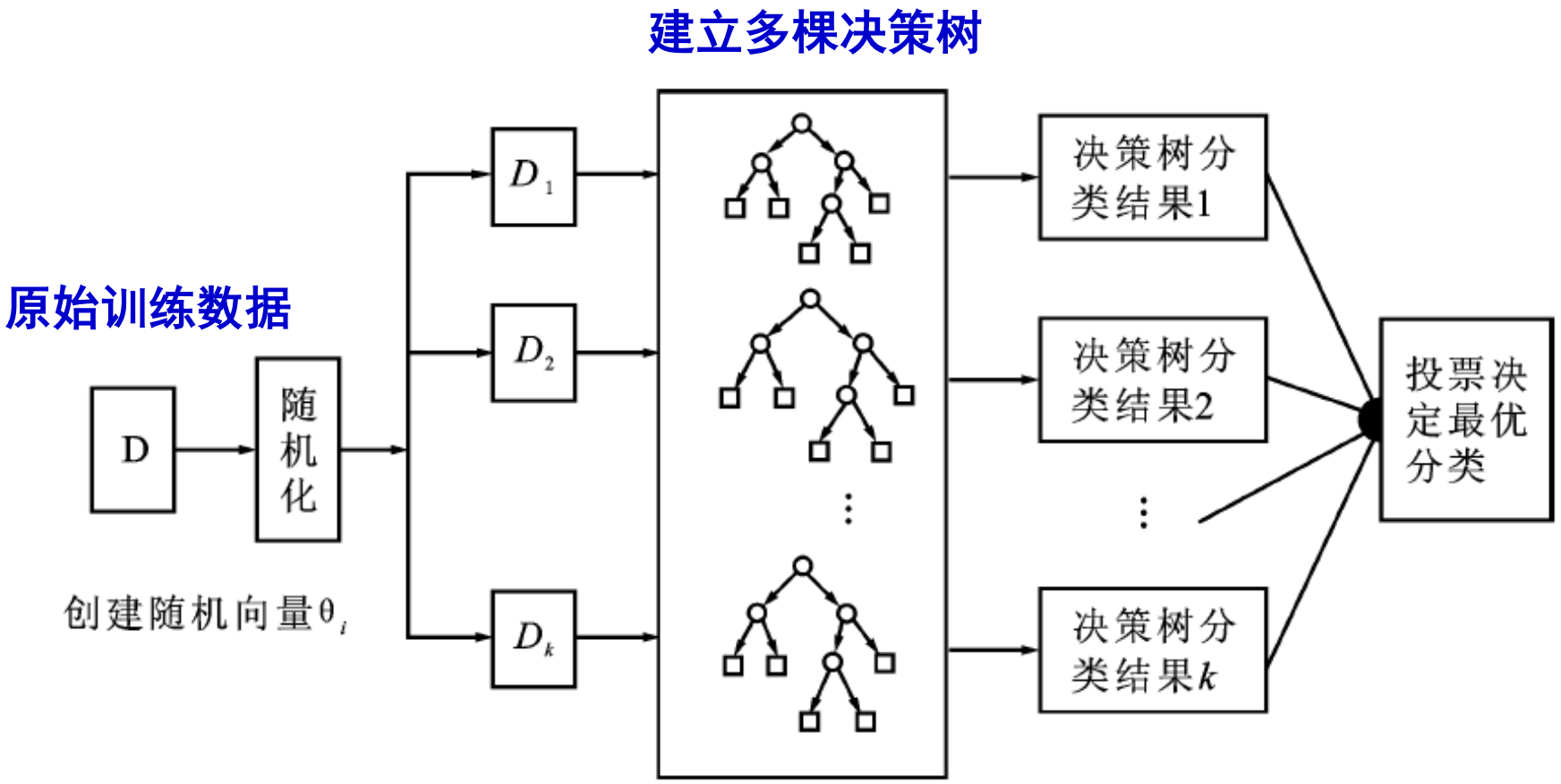
来源4: CART

回顾一下RF里面用到的决策树算法CART（Classification and Regression Trees）。CART是以自顶向下递归方式构造的二叉树。基本的步骤包括构造和剪枝两部分。

- 对于构造决策树方面：
  - ①创建一个结点N
  - ②如果样本T都在同一个类C中，返回N作为叶结点，以类C做标记
  - ③从候选属性A集中找出GINI系数最小的属性
  - ④将样本T划分为T1，T2两个子集
  - ⑤对T1重复①-④
  - ⑥对T2重复①-④

- 生成的树是完全分裂的，肯定存在过度拟合的现象，CART使用事后剪枝的方式对决策树进行剪枝。

随机森林算法流程图示



随机森林分类过程图

随机森林算法流程

- 该算法用随机的方式建立起一棵棵决策树，然后由这些决策树组成一个森林，其中每棵决策树之间没有关联，当有一个新的样本输入时，就让每棵树独立的做出判断，按照多数原则决定该样本的分类结果。

分为

- 构建随机森林
- 使用随机森林预测

构建随机森林

①从样本集中用bagging采样选出n个样本，预建立CART

②在树的每个节点上，从所有属性中随机选择k个属性，选择出一个最佳分割属性作为节点（RI 和 RC）

③重复以上两步m次，i.e. 构建m棵CART(不剪枝)

④这m个CART形成Random Forest

与bagging方法不同，随机森林存在两次随机过程：

- 使用Bootstrap随机选择子样本；
- 最佳分割属性不是从完全的属性集上挑选出来，而是从随机选出的部分属性集中挑选的。

对于样本采样，Breiman用的就是前面介绍过的0.632自助法。

对于属性的采样，Breiman设计了两种方法，分别是RI(随机输入选择)和RC（随机线性组合）。

- RI就是随机的从完全属性集中选择一定数量的属性形成候选属性集。
- RC会先从所有随机变量里面选择L个变量，然后把这些变量通过线性组合形成一个新的组合变量，使用的系数来自于[-1,1]上的随机数。用这种方法得到F个组合变量形成候选分割属性集。

利用随机森林预测

基本步骤（分类）：

①向建立好的随机森林中输入一个新样本

②随机森林中的每棵决策树都独立的做出判断

③将**得到票数最多**的分类结果作为该样本最终的类别

上面是针对分类而言的，对于回归预测，最简单的做法就是将所有决策树的**预测结果取平均值**作为最终的结果。

影响随机森林分类性能的主要因素

- 森林中单颗树的分类强度（Strength）：每颗树的分类强度越大，则随机森林的分类性能越好。

- 森林中树之间的相关度（Correlation）：树之间的相关度越大，则随机森林的分类性能越差。

随机森林的几个理论要点

- 收敛定理（详见论文）
  - 它度量了随机森林对给定样本集的分类错误率
- 泛化误差界
  - 单个决策树的分类强度越大，相关性越小，则泛化误差界越小，随机森林分类准确度越高。
- 袋外估计
  - Breiman在论文中指出袋外估计是无偏估计，袋外估计与用同训练集一样大小的测试集进行估计的精度是一样的。
  - Using out-of-bag estimates to monitor error, strength, and correlation

Out-of-Bag (OOB) Estimation (袋外估计)

- 在bootstrapping的过程中，有些数据可能没有被选择，这些数据称为out-of-bag(OOB) examples，对于训练每一个决策树 g_t ，其中用星号标注的数据即是g_t的OOB examples

Bagging

function Bag($\mathcal{D}, \mathcal{A}$)
For $t = 1, 2, \ldots, T$

1. request size-N' data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain base g_t by $\mathcal{A}(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_2$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	$\star$		$\star$

$\star$ in t-th column: not used for obtaining g_t
—called **out-of-bag (OOB) examples** of g_t

Number of OOB Examples

OOB (in $\star$) $\Longleftrightarrow$ not sampled after N' drawings

if $N' = N$

- probability for $(\mathbf{x}_n, y_n)$ to be OOB for g_t: $\left(1 - \frac{1}{N}\right)^N$
- if N large:

$$\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(\frac{N}{N-1}\right)^N} = \frac{1}{\left(1 + \frac{1}{N-1}\right)^N} \approx \frac{1}{e}$$

OOB size per $g_t \approx \frac{1}{e}N$

上面公式是经过 N'次选择之后没有被选择的数据，大约有(1/e)N多的数据没有被选择到，即大约有三分之一的数据没有被选择，这些数据由于没有用来训练模型，故可以用于模型的验证。

OOB vs Validation

Validation

g_1^-	g_2^-	$\cdots$	g_M^-
$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{train}}$		$\mathcal{D}_{\text{train}}$
$\mathcal{D}_{\text{val}}$	$\mathcal{D}_{\text{val}}$		$\mathcal{D}_{\text{val}}$
$\mathcal{D}_{\text{val}}$	$\mathcal{D}_{\text{val}}$		$\mathcal{D}_{\text{val}}$
$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{train}}$		$\mathcal{D}_{\text{train}}$

OOB

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_2$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\star$	$\star$		$\star$

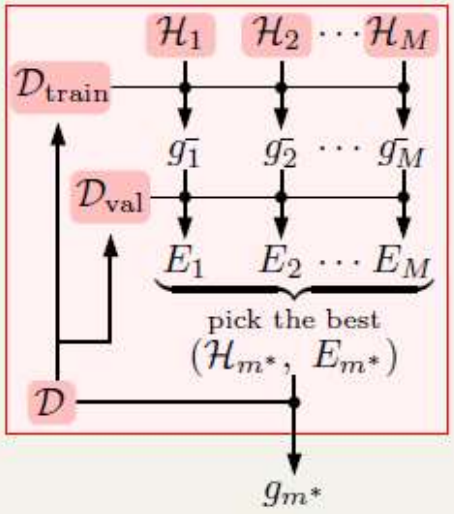
- $\star$ like $\mathcal{D}_{\text{val}}$: 'enough' random examples unused during training
- use $\star$ to validate g_t? easy, but **rarely needed**
- use $\star$ to validate G? $E_{\text{oob}}(G) = \frac{1}{N}\sum_{n=1}^{N} \text{err}(y_n, G_n^-(\mathbf{x}_n))$, with G_n^- contains only trees that $\mathbf{x}_n$ is OOB of, such as $G_N^-(\mathbf{x}) = \text{average}(g_2, g_3, g_T)$

E_{oob}: self-validation of bagging/RF

Model Selection by OOB Error

Previously: by Best E_{val}

$$g_{m^*} = \mathcal{A}_{m^*}(\mathcal{D})$$
$$m^* = \operatorname*{argmin}_{1 \le m \le M} E_m$$
$$E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}}))$$



RF: by Best E_{oob}

$$G_{m^*} = \text{RF}_{m^*}(\mathcal{D})$$
$$m^* = \operatorname*{argmin}_{1 \le m \le M} E_m$$
$$E_m = E_{\text{oob}}(\text{RF}_m(\mathcal{D}))$$

- use E_{oob} for self-validation —of RF **parameters** such as d''
- no re-training needed

E_{oob} often **accurate** in practice

随机森林的优点

①可以处理高维数据，可以不用进行特征筛选

②非常容易进行分布式处理，实现起来非常高效

③可以利用OOB(out of bag)数据评价算法的误差率，而不用像一般算法那样还需要选择validation set测误差

④可以利用OOB和permutation test(把某个特征的值随机进行重排)计算每个feature的重要性，也可以作为其他非线性算法特征选择阶段的算法

⑤因为最终结果通过voting by majority得到，相对光滑，像SVM那样“large-margin like”的边界,受噪音干扰小，鲁棒性好

⑥CART的优点：容易理解，处理非线性的问题，可以很好的处理数值型和序列型变量，处理缺失值

随机森林的缺点

①树越多，随机森林的表现才会越稳定。

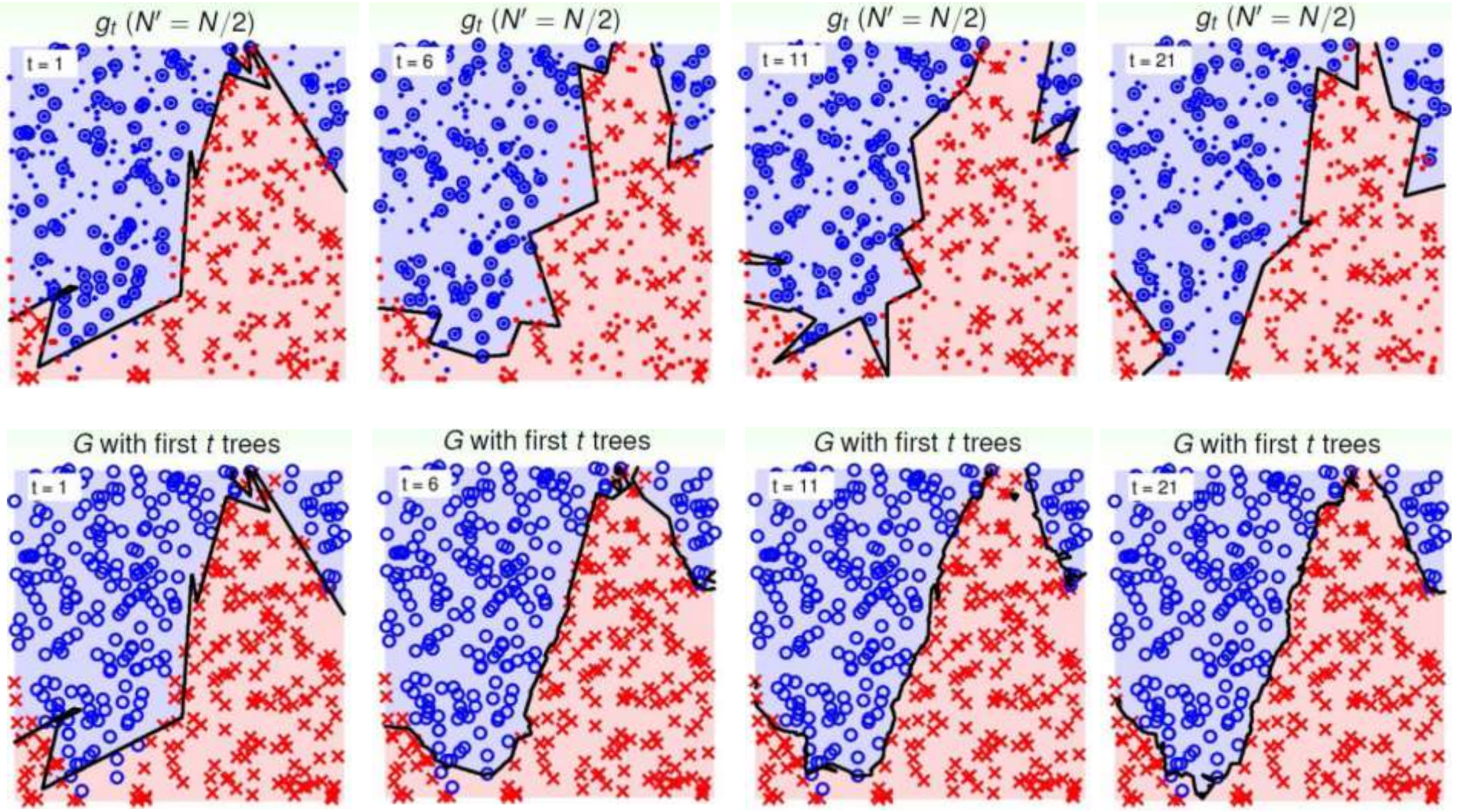
所以在实际使用随机森林的时候需要注意如果树不够多的时候，可能会导致不稳定的情况。

②不平衡数据集。分类结果会倾向于样本多的类别，所以训练样本中各类别的数据必须相同。

Breiman在实现该算法的时候考虑到了该问题，采取了根据样本类别比例对决策树的判断赋予不同权值的方法，比如类A:类B的样本量比例=3:1,,则对新样本，判断为类A的决策结果乘以25%，判断为类B的结果乘以75%，然后再比较两者大小，得到最终的结果。但是Breiman的实验结果也表明了该方法还是不能很好的解决由样本数量不平衡造成的结果不准确的问题。

Example

almost every theory: the more, **the 'better'**
assuming **good** $\bar{g} = \lim_{T\to\infty} G$



Random Forest 应用举例

Microsoft Research

Classification forests in practice

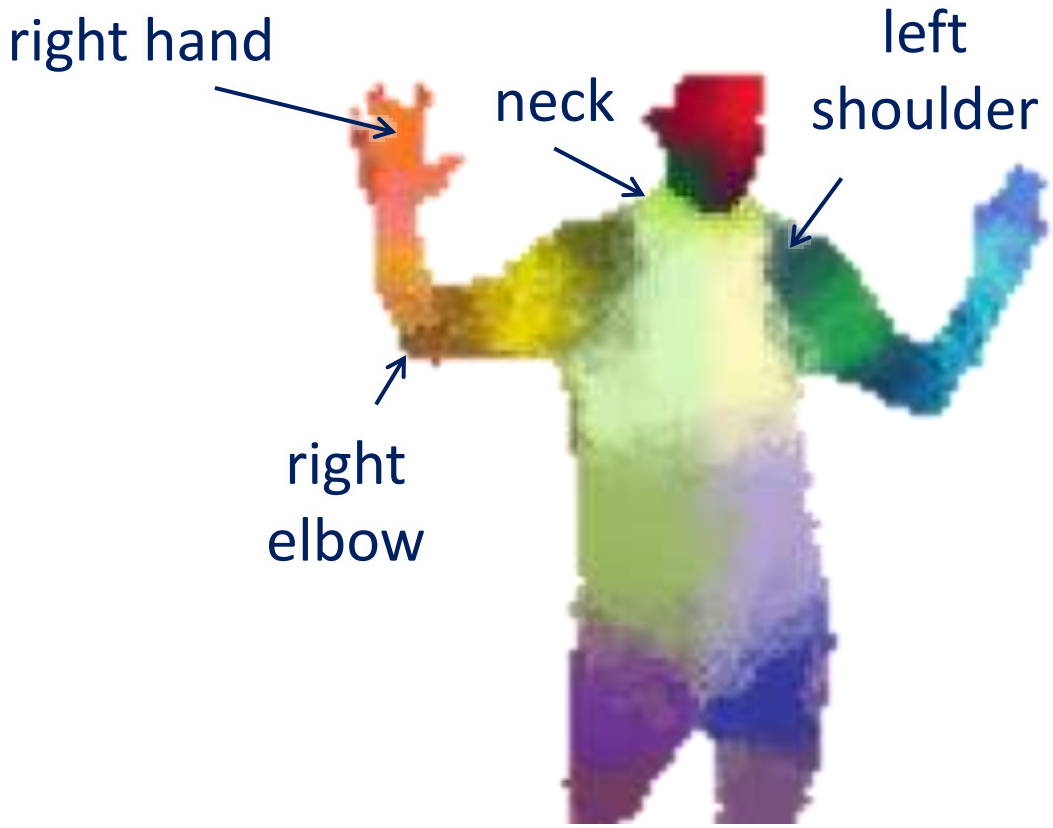
Body Part Recognition for Kinect

Jamie Shotton
Sebastian Nowozin

ICCV 2013 Tutorial

[J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake.
Real-Time Human Pose Recognition in Parts from a Single Depth Image. *In Proc. IEEE CVPR*, June 2011.]

Body Part Recognition



Body Part Recognition

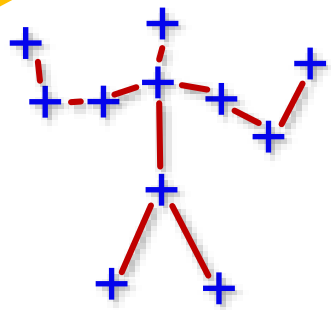
- No temporal information
  - frame-by-frame
- Local pose estimate of parts
  - each pixel & each body joint treated independently
  - reduced training data and computation time
- Very fast
  - simple depth image features
  - parallel decision forest classifier

Kinect Pipeline

capture
depth image &
remove bg

infer
body parts
per pixel

cluster pixels to
hypothesize
body joint
positions



fit model &
track skeleton

Classifying Pixels

- Compute $\mathrm{P}(c_i \mid w_i)$
  - pixels $i = (x, y)$
  - body part c_i
  - image window w_i

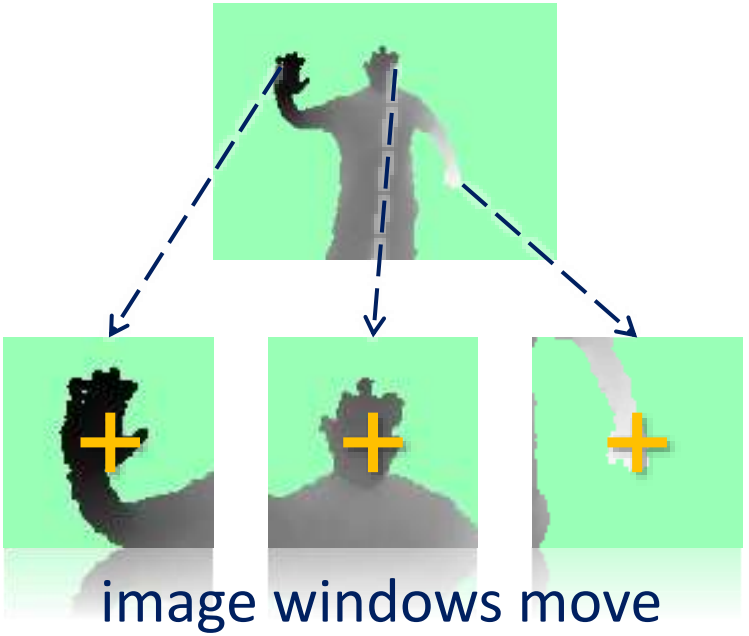


image windows move with classifier

- Discriminative approach
  - learn classifier $\mathrm{P}(c_i \mid w_i)$ from training data

Synthetic Training Data

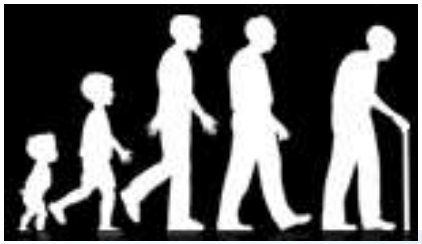
Record mocap

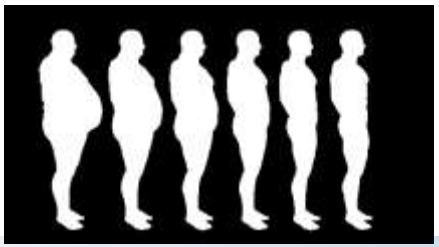
500k frames
distilled to 100k poses

Retarget to several models

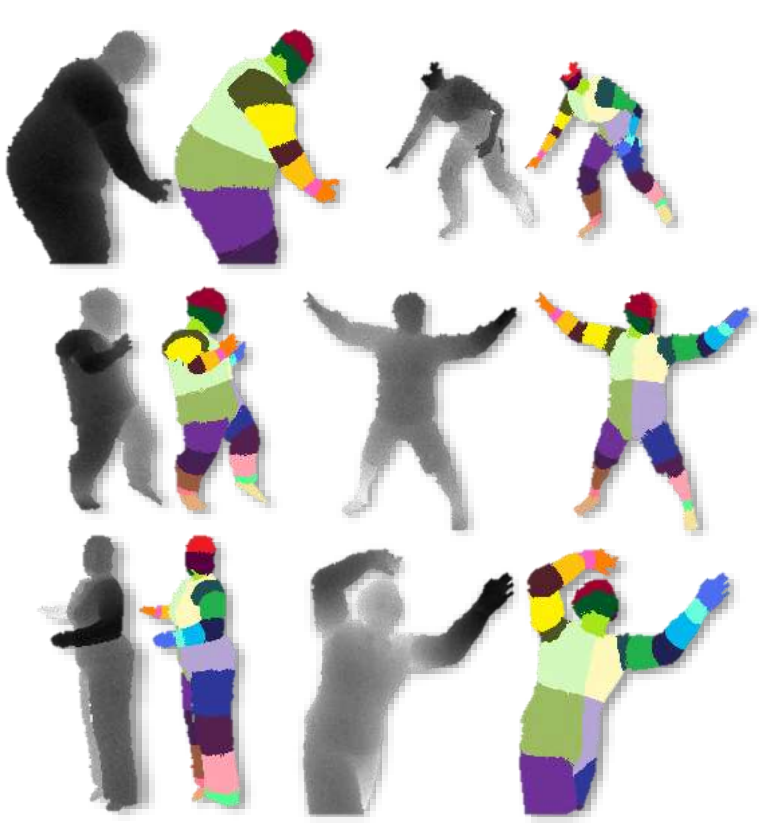
Render (depth, body parts) pairs

Train invariance to:

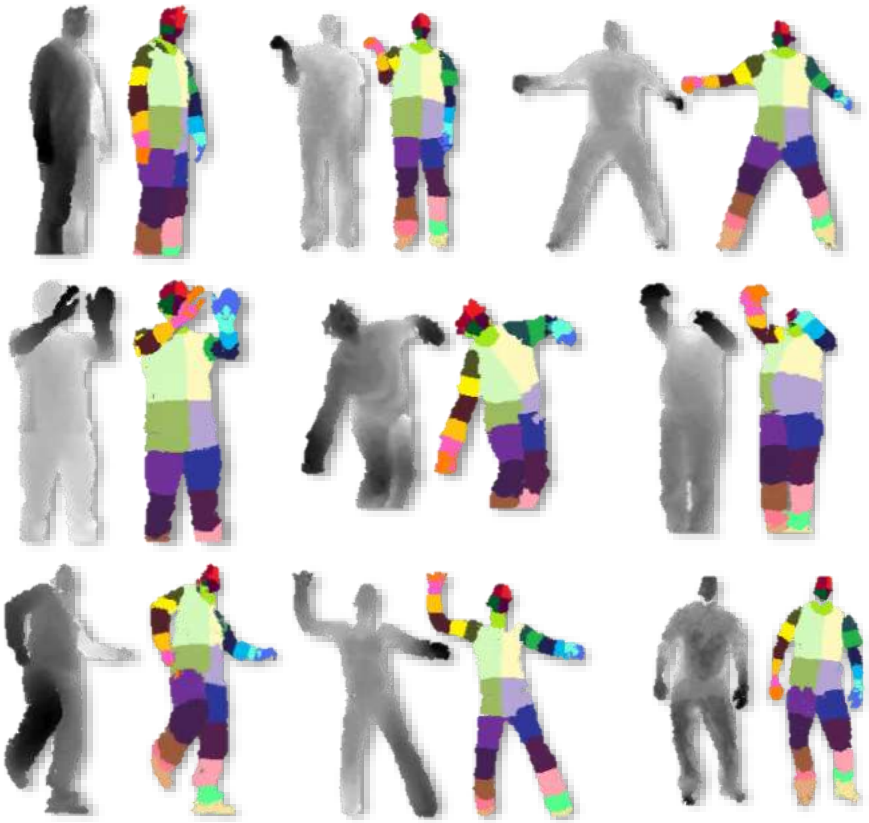




Synthetic vs Real Data



synthetic
(train & test)



real
(test)

Fast Depth Image Features

- Depth comparisons
  - very fast to compute

feature response

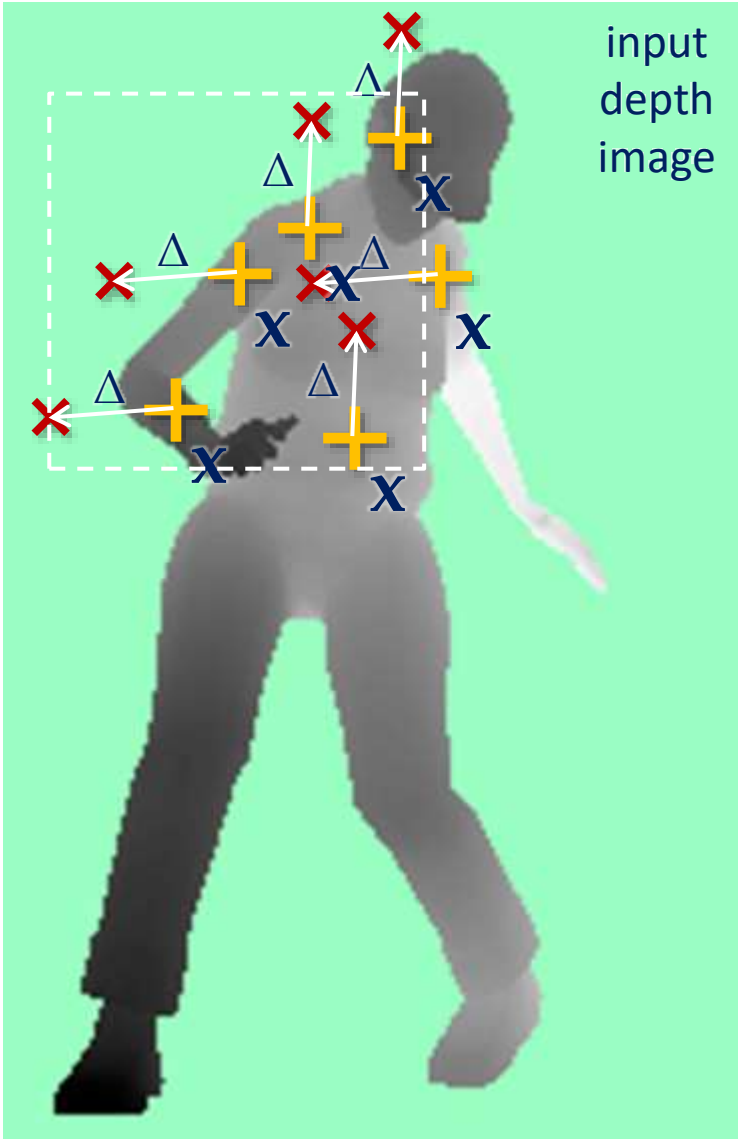
$$f(I, \mathbf{x}) = d_I(\mathbf{x}) - d_I(\mathbf{x} + \Delta)$$

image coordinate; image; depth; offset depth

$$\Delta = \frac{\mathbf{v}}{d_I(\mathbf{x})}$$

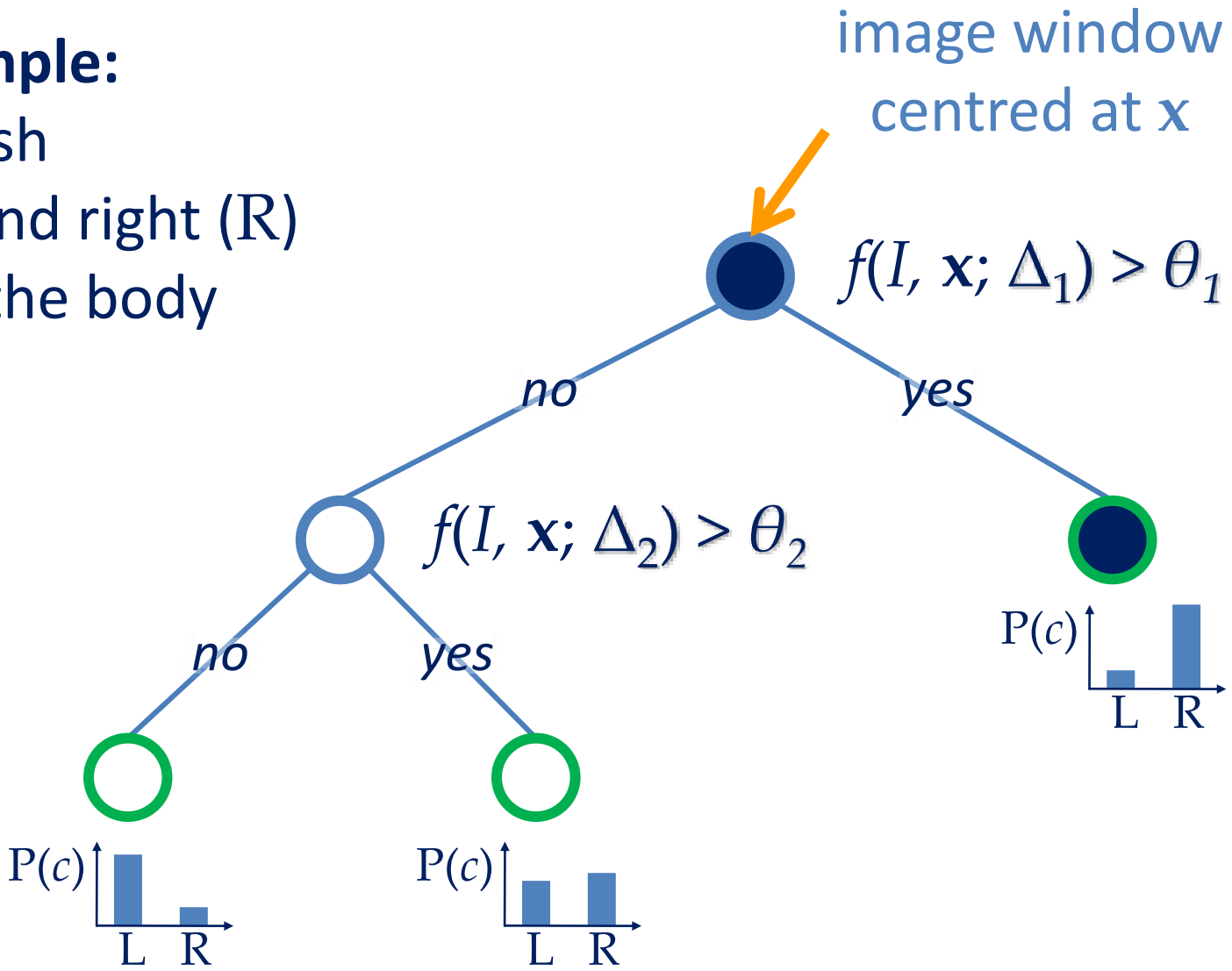
scales inversely with depth

Background pixels
d = large constant



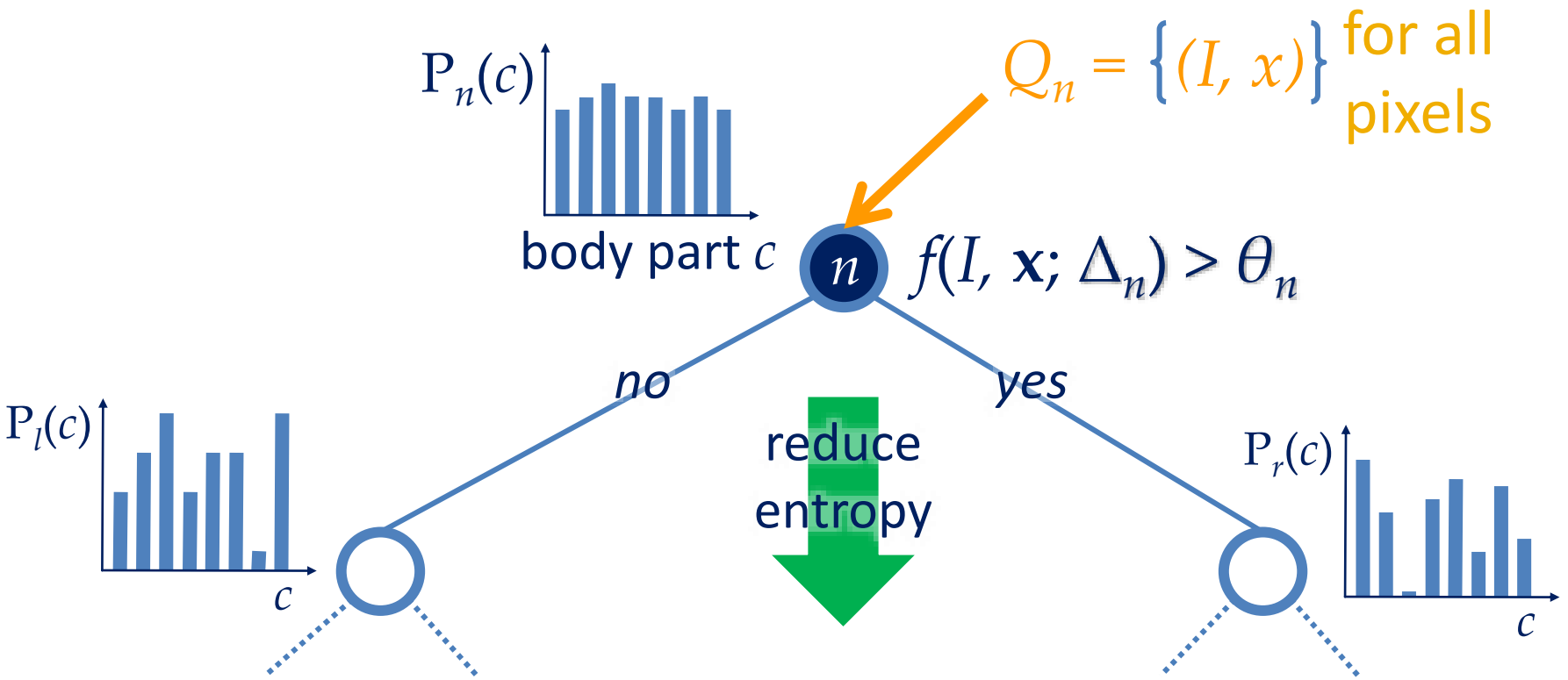
Decision Tree Classification

Toy example:
distinguish
left (L) and right (R)
sides of the body



Training Decision Trees

[Breiman *et al.* 84]



Take (Δ, θ) that maximises information gain:

$$\Delta E = -\frac{|Q_l|}{|Q_n|} E(Q_l) - \frac{|Q_r|}{|Q_n|} E(Q_r)$$

Goal: drive entropy at leaf nodes to zero

Depth of Trees

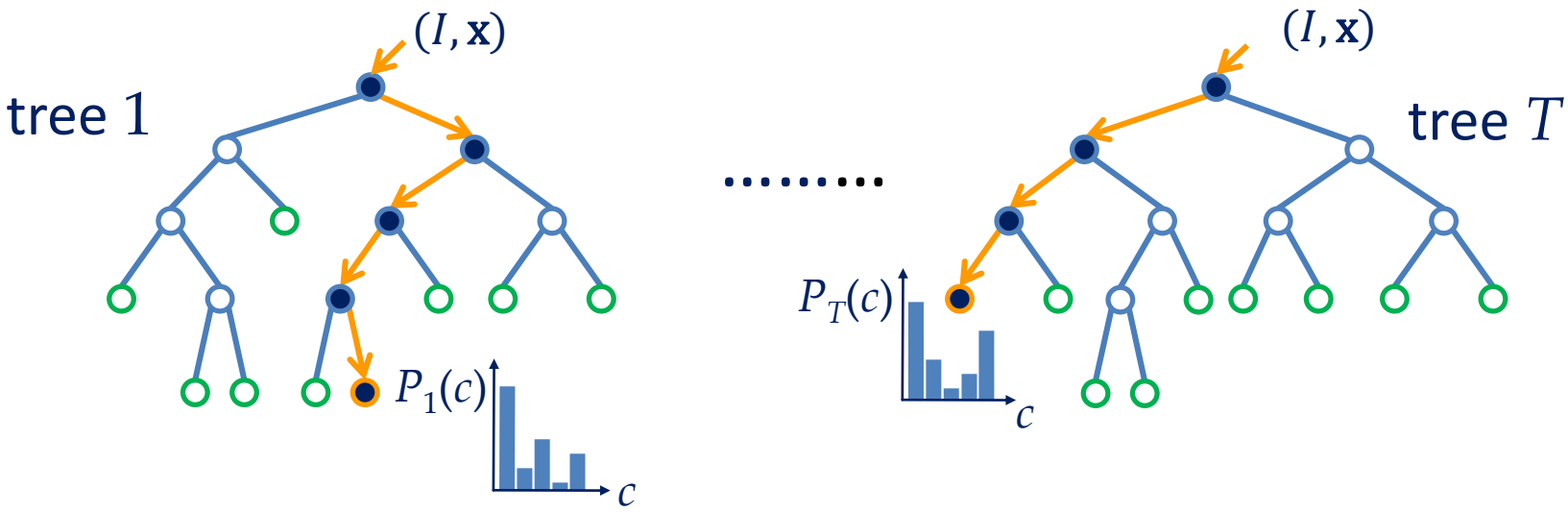
input depth

ground truth parts

inferred parts (soft)

depth 18

Decision Forest Classifier

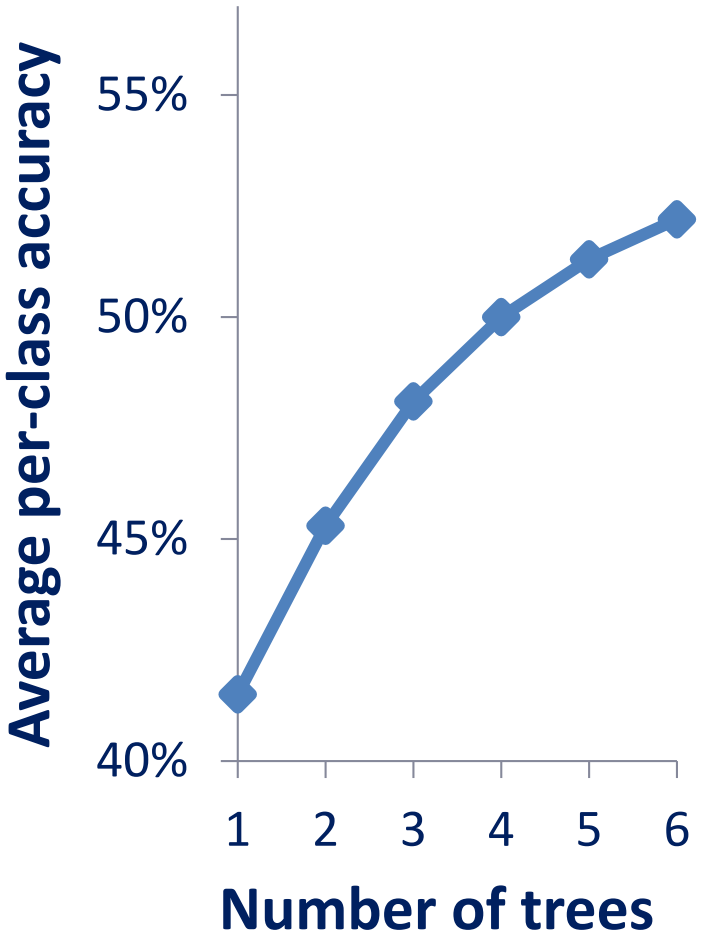


- Trained on different random subset of images
  - "bagging" helps avoid over-fitting

- Average tree posteriors

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

Number of Trees



ground truth

inferred body parts (most likely)

1 tree

3 trees

6 trees

Body Parts to Joint Hypotheses

- Define 3D world space density:

3D coord

pixel weight

3D coord of *i* th pixel

$$f_c(\hat{\mathbf{x}}) \propto \sum_{i=1}^{N} w_{ic} \exp\left(-\left\|\frac{\hat{\mathbf{x}} - \hat{\mathbf{x}}_i}{b_c}\right\|^2\right)$$

pixel index *i*

bandwidth

$$w_{ic} = P(c|I, \mathbf{x}_i) \cdot d_I(\mathbf{x}_i)^2$$

inferred probability

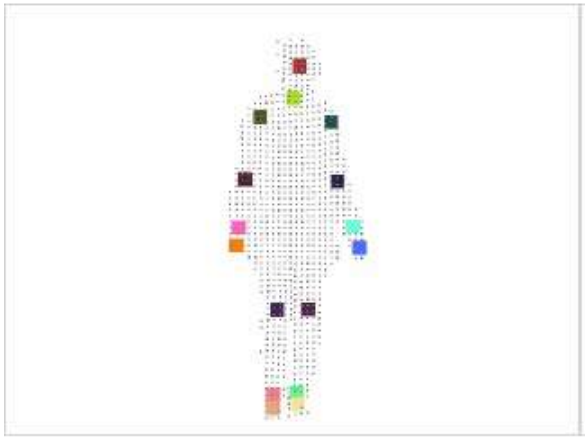
depth at *i* th pixel

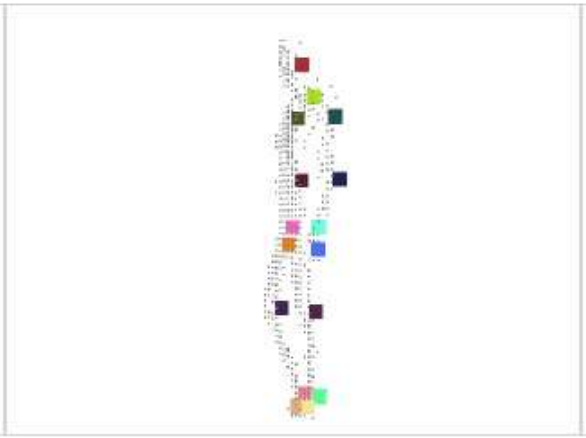
3. hypothesize body joints

- **Mean shift** for mode detection

input depth

inferred body parts





front view

side view

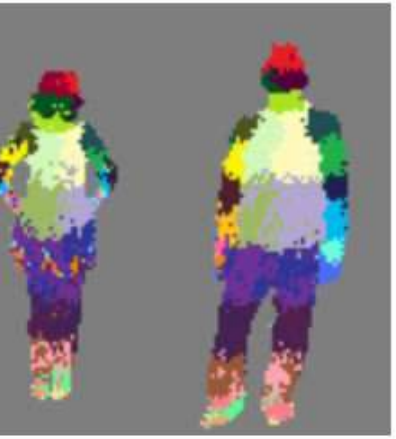
top view

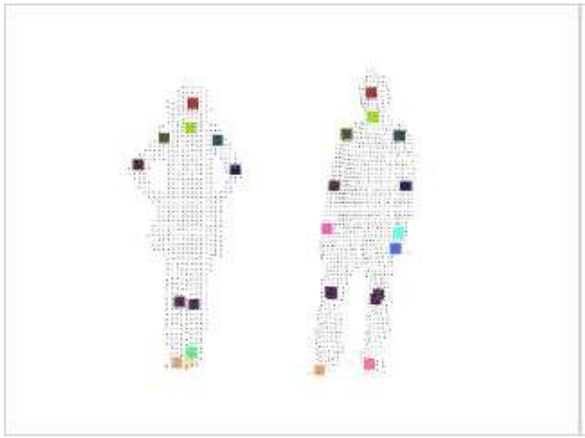
inferred joint positions

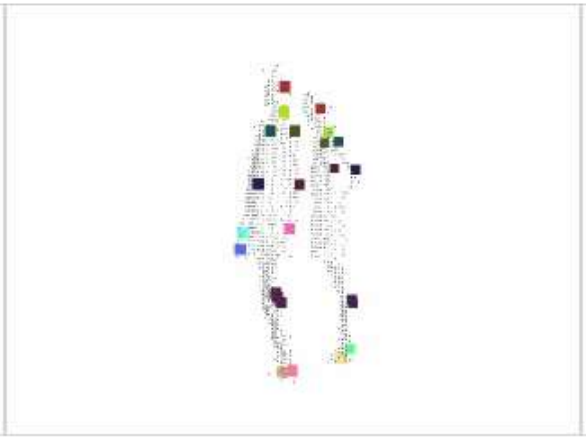
no tracking or smoothing

input depth

inferred body parts







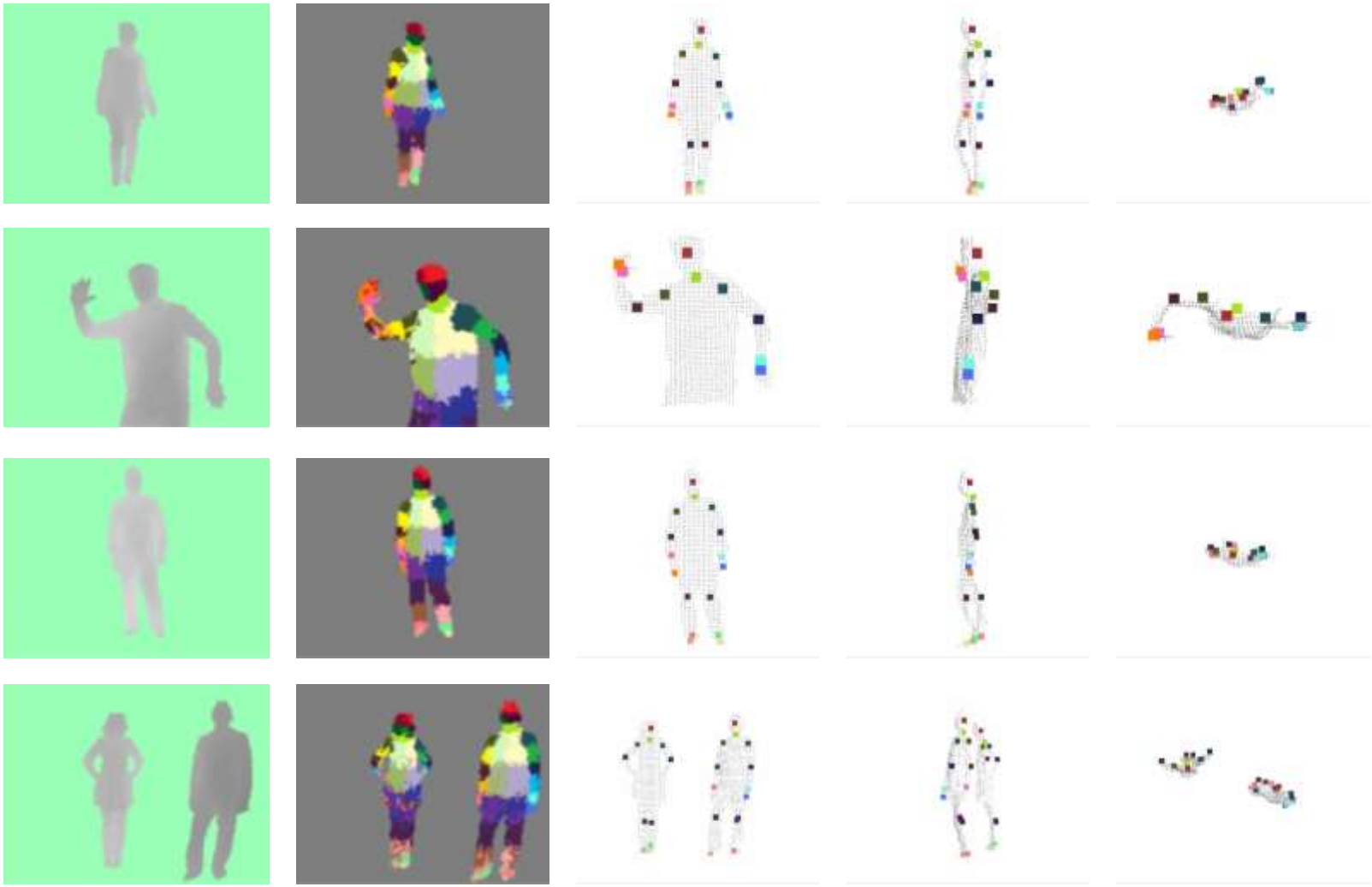
front view

side view

top view

inferred joint positions

no tracking or smoothing



Microsoft Research

XBOX 360

Thank You!

Q&A