Econ 425 Week 4

Classification II: decision trees

Grigory Franguridi

UCLA Econ

USC CESR

franguri@usc.edu

Reminder: classification problem

Setup:

- set of possible **instances** $\mathcal{X}$
- set of possible **labels** $\mathcal{Y}$
- unknown **target** function $f : \mathcal{X} \to \mathcal{Y}$
- **model**, i.e. set of hypotheses $H = \{h : \mathcal{X} \to \mathcal{Y}\}$

Input: training data with $y_i = f(x_i)$ (possibly with noise),

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \cdots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Goal: construct a hypothesis $\hat{f} \in H$ that best approximates f, i.e. predicts well labels of unseen instances (*generalizes*)

Example: classification

- features $\boldsymbol{x}_i$ describe weather
- label $y_i =$ "tennis game is played"
- dataset: labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision trees: motivation

So far in this class: models (predictors, hypotheses) of the form

$$f(\boldsymbol{x}) = \eta(\boldsymbol{x}'\beta),$$

Single linear

e.g., logistic $\eta(z) = \frac{e^z}{1+e^z}$, linear $\eta(z) = z$,
perhaps trained with regularization like lasso/ridge

?

But is it optimal to use (i) single index $\boldsymbol{x}'\beta$ (ii) original features $\boldsymbol{x}$?

How about

- interactions, e.g., $x_1 x_6$
- powers, e.g., x_1^2
- **splits**, e.g., $1(x_1 \leq 0.47)$
- **non-single-index** hypotheses

} **decision trees**

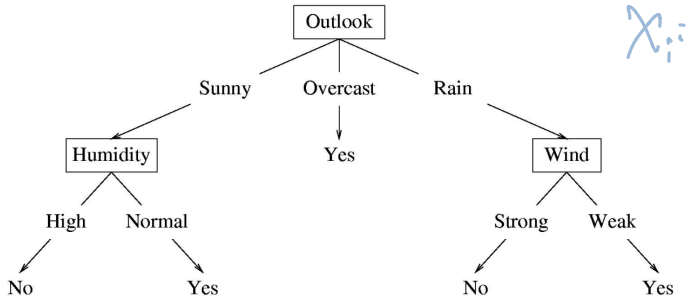
↓ ?

↓ compress Xs into a single number with linear ...

Hard to choose transformation.

Decision trees: example

One possible predictor (hypothesis) is a **decision tree**

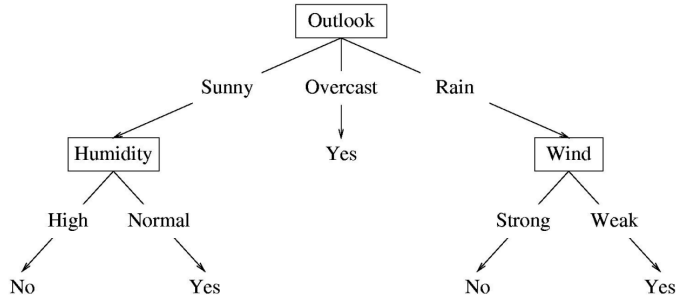


X_i: outlook, Humidity, wind, temperature ...

- Each **internal node**: test one attribute x_k (e.g., humidity)
- Each **branch**: selects one value for x_k (e.g., "high" humidity)
- Each **leaf node**: predict Y (e.g., "yes")

Decision trees: example

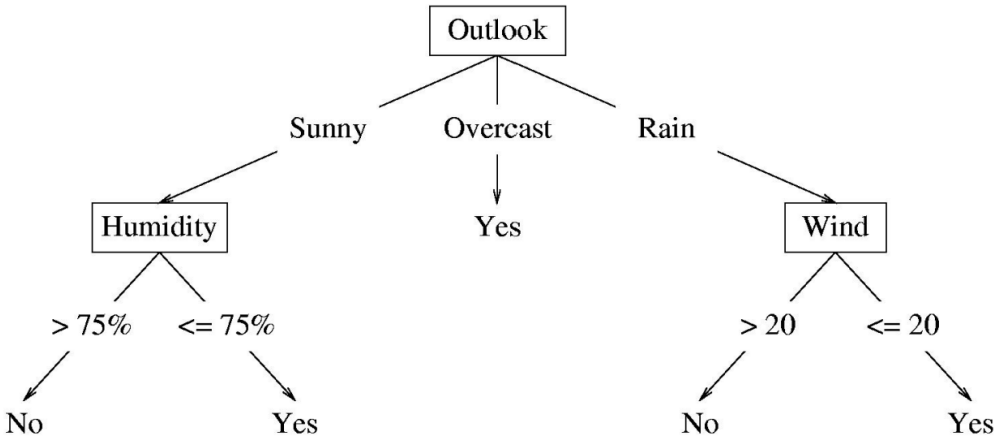
One possible predictor (hypothesis) is a **decision tree**



- What does this DT predict for $\boldsymbol{x} = \langle$outlook=sunny, temperature=hot, humidity=high, wind=weak$\rangle$?

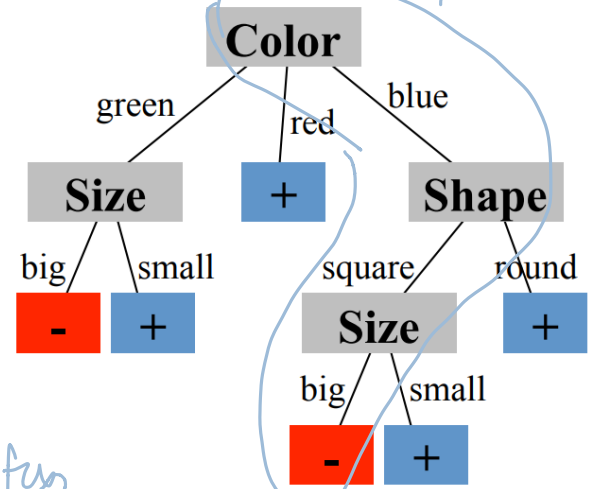
Decision trees: example

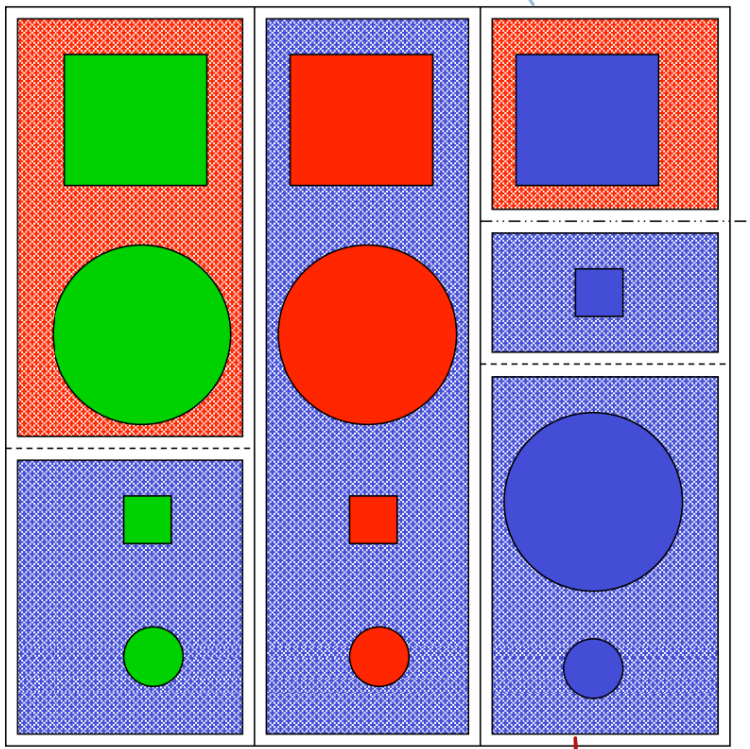
If some feature is continuous, test its value against a threshold:



Decision tree-induced partition of $\mathcal{X}$

$\boldsymbol{x}$ =(shape, size, color), $y \in \{+, -\}$





$y = f(x)$

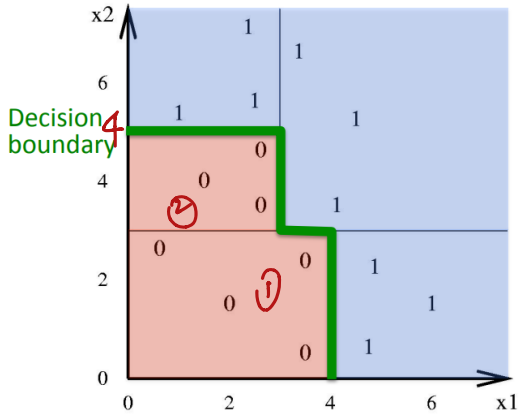
$f^{-1}(1) = \{x : f(x) = -1\}$

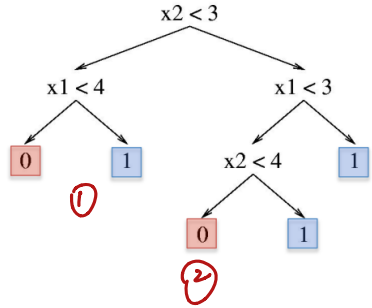
$f^{-1}(+1) = \{x : f(x) = +\} \longrightarrow$ blue background

(非连续性函数?)

Decision boundary

- Decision trees divide the feature space $\mathcal{X}$ into axis-parallel (hyper-)rectangles
- Each rectangle is labeled with one label (or a probability distribution over labels)



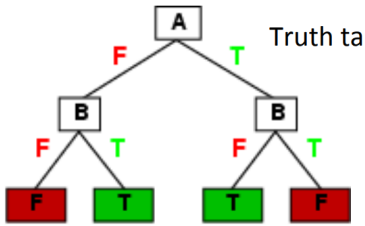


Expressiveness

Decision trees can represent **any** function
f from $\boldsymbol{x} \in \{0,1\}^d$ to $y \in \{0,1\}$

Example: let $\boldsymbol{x} = (A, B) \in \{T, F\}^2$ and $y = f(\boldsymbol{x}) = A \operatorname{xor} B$
(exclusive OR)

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



- In the worst case, the tree is exponentially large

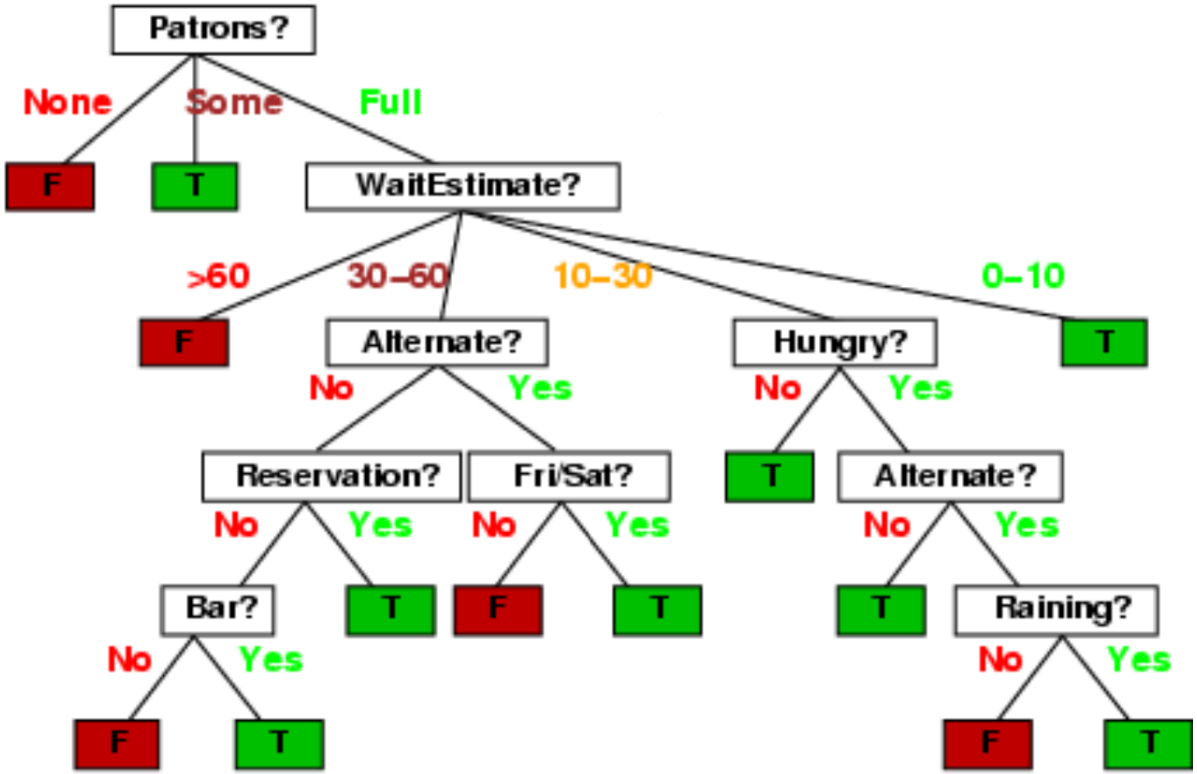
Decision trees: example

Target y is client's decision to wait for a table at a restaurant (Russell & Norvig)

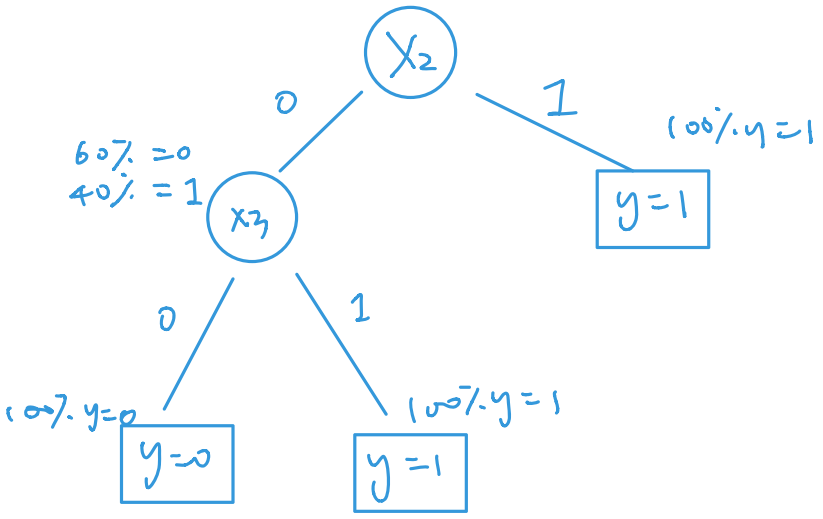
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

~7,000 possible cases

Decision tree from introspection (eyeballing)



- Is this **the best** decision tree?



depth = 2

train error = 0

Training decision trees

Basic algorithm (ID3):

> *node* = root of decision tree
>
> Main loop:
>
> 1. $A \leftarrow$ the "best" decision attribute for the next node.
> 2. Assign A as decision attribute for *node*.
> 3. For each value of A, create a new descendant of *node*.
> 4. Sort training examples to leaf nodes.
> 5. If training examples are perfectly classified, stop.
>   Else, recurse over new leaf nodes.

- how to define/choose an **attribute** to split?
- for a continuous attribute, how to choose its **split threshold**?
- is perfect classification (zero training error) good? (no)

Choosing attribute to split

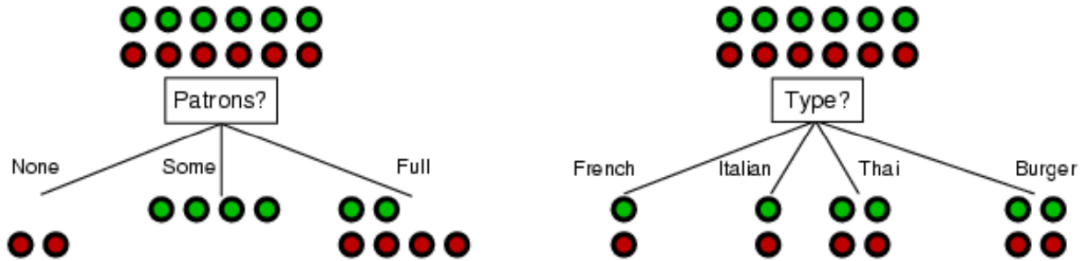
Some possibilities:

- **Random**
- **Least-values**: the attribute with the smallest number of possible values
- **Most-values**: the attribute with the largest number of possible values
- **Max-gain**: the attribute that has the largest expected information gain

The ID3 algorithm uses **max-gain**

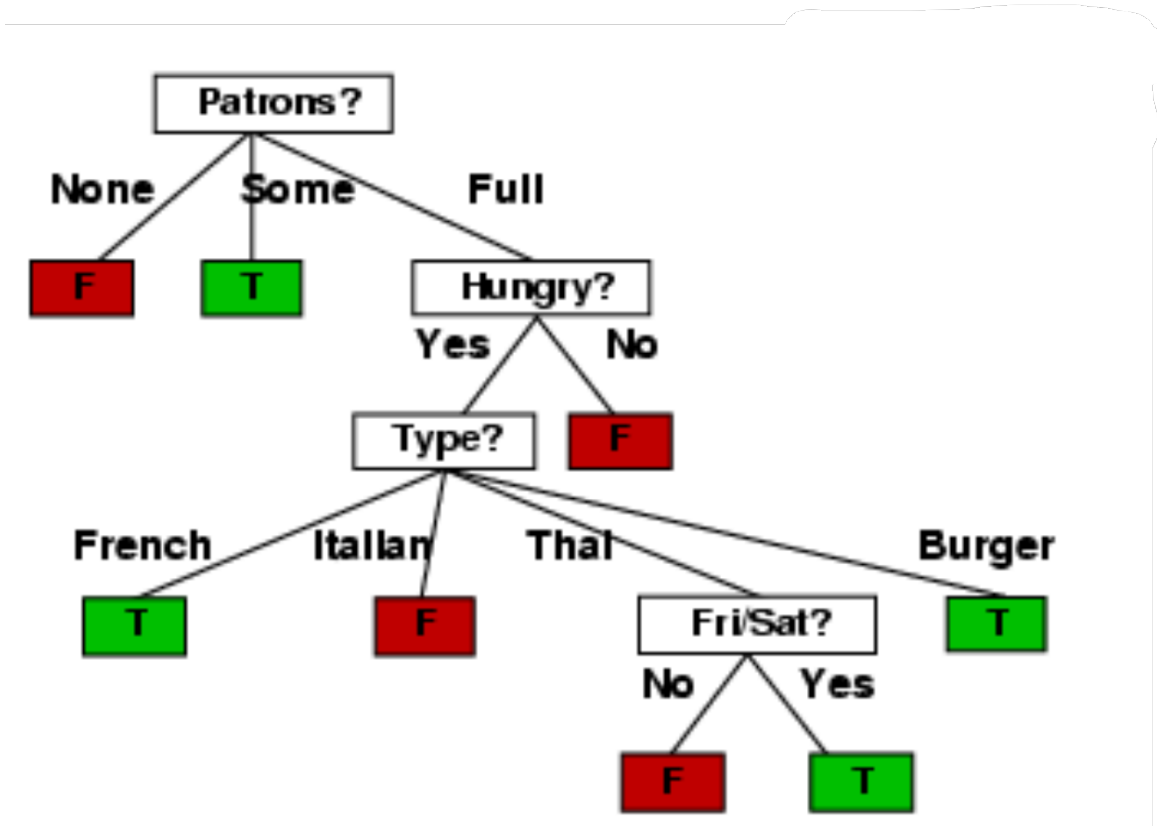
Choosing attribute to split

Idea: a good attribute splits the examples into subsets that are ideally "all positive" or "all negative" (i.e. have **low impurity**)

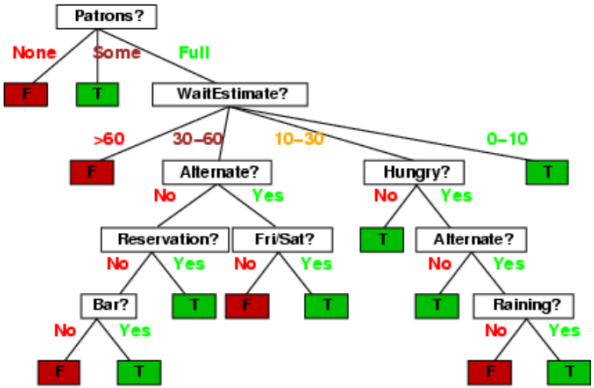


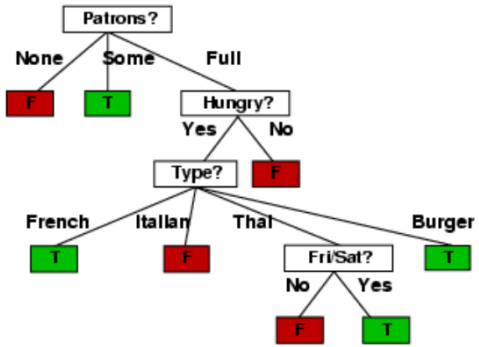
- which attribute is more informative: Patrons or Type?

ID3-trained decision tree



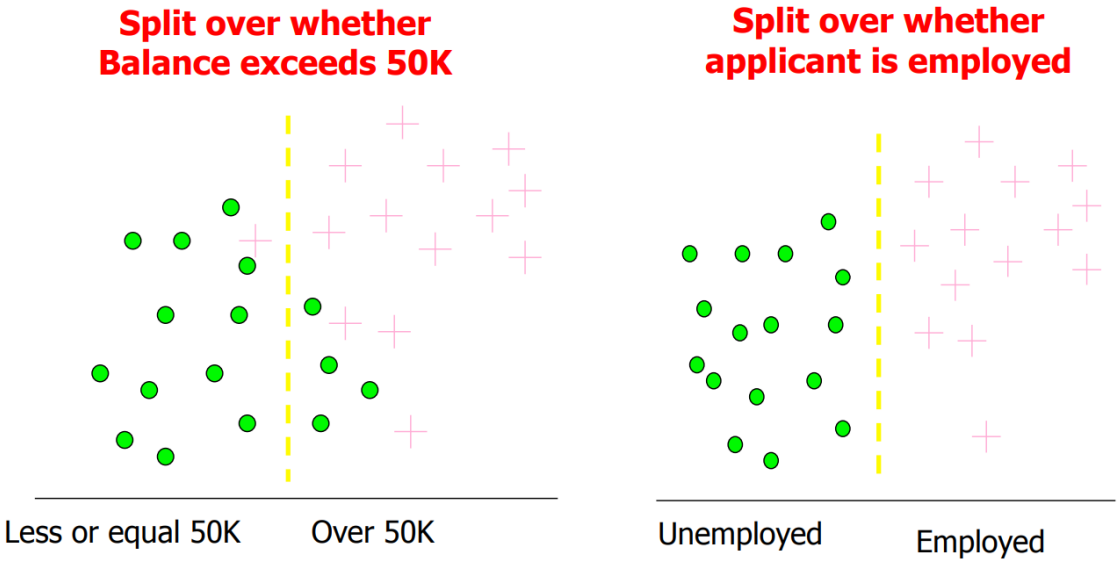
The two decision trees: introspection vs ID3





Choosing attribute to split

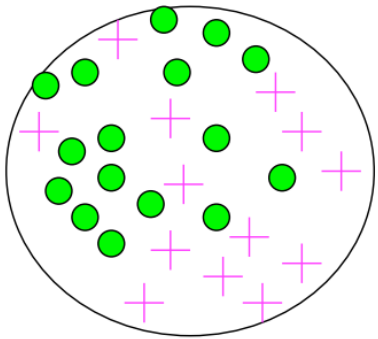
Which attribute is **better** (more informative)?



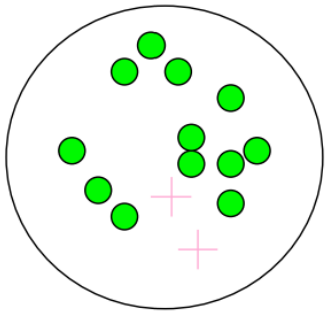
- employment status is better:
  more **information** / lower **impurity** of subclasses

Impurity

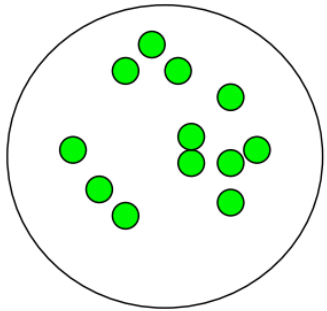
Very impure group



Less impure



Minimum impurity



Measuring impurity: entropy

Entropy $H(X)$ of a random variable $X \in \mathcal{X}$ is

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

$= -p \cdot \log p - (1-p) \log(1-p)$

X is treated as a signal

- rare realizations x reveal more information
- **amount of information** in x is $-\log_2 P(X = x)$
  (roughly, the number of bits needed to encode info in x)
- hence entropy is the expected amount of information in X
- $X = const \Rightarrow H(X) = 0$
- $X \sim \text{Ber}(p) \Rightarrow H(X) = -p \log p - (1-p) \log p$
  - when is this entropy maximal? why?
- only depends on the distribution (not the values) of X

From entropy to information gain

Entropy $H(X)$ of a random variable $X \in \mathcal{X}$ is

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

Conditional entropy $H(X|Y = y)$ of X given $Y = y$ is

$$H(X|Y = y) = -\sum_{x \in \mathcal{X}} P(X = x|Y = y) \log_2 P(X = x|Y = y)$$

From entropy to information gain

Conditional entropy $H(X|Y)$ of X given Y is

$$H(X|Y) = \sum_{y \in \mathcal{Y}} P(Y = y) H(X|Y = y)$$

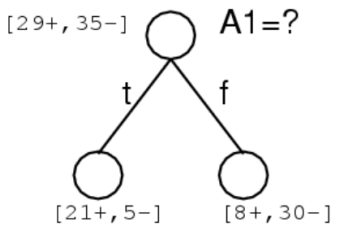
Information gain (aka mutual information) of X and Y is

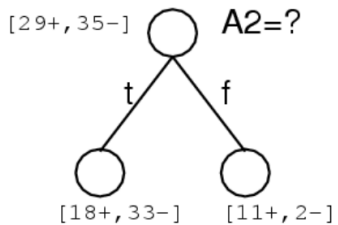
$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Sample information gain

- suppose A is an attribute (a component of $\boldsymbol{X}$)
- high $I(A, Y) \Leftrightarrow A$ is a good predictor of Y
- for a sample S, we can estimate $I(A, Y)$ by **sample IG**, i.e. the reduction in the sample entropy of Y due to sorting on variable A

$$I_S(A, Y) = H_S(Y) - H_S(Y|A)$$



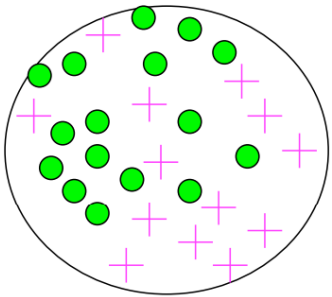


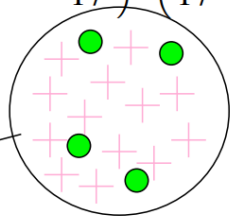
Calculating IG

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

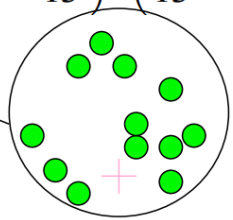
Entire population (30 instances)





17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



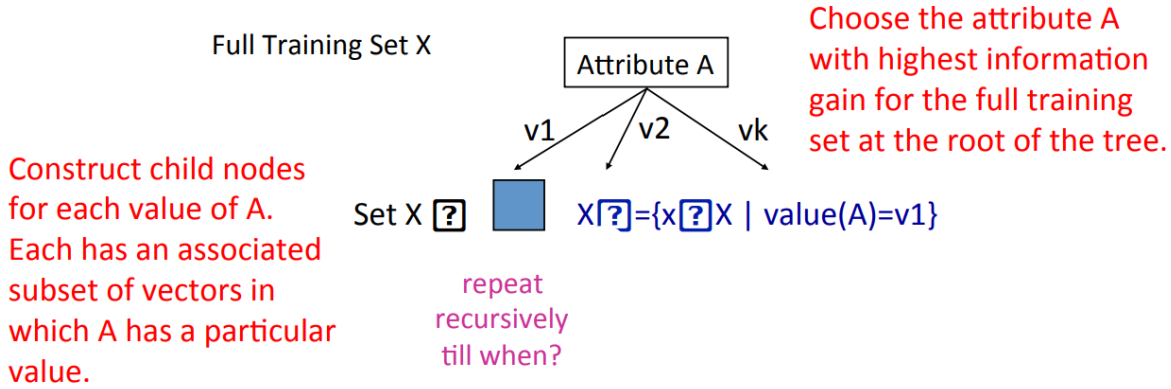
13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Using IG for training decision trees



Main drawback of IG:

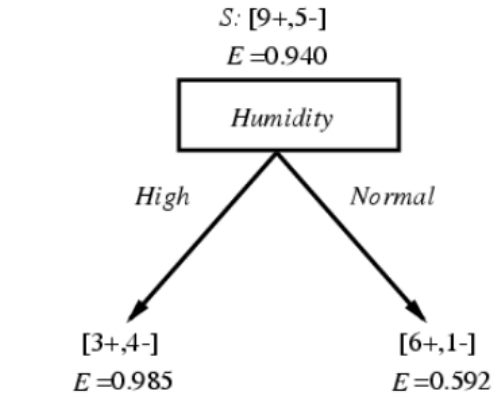
- prefers attributes that take many values and split the data into **small, pure subsets**
  $\Rightarrow$ potential **overfitting**

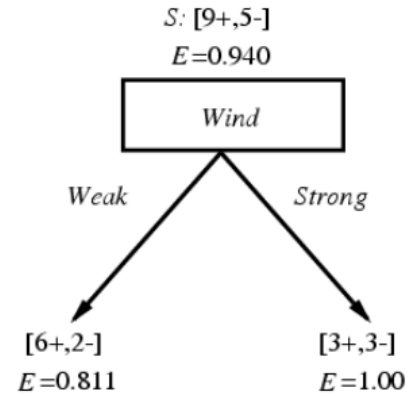
Back to example

Day	Outlook	Temperature	Humidity	Wind	PlayTenr
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the next attribute

Which attribute is the best classifier?



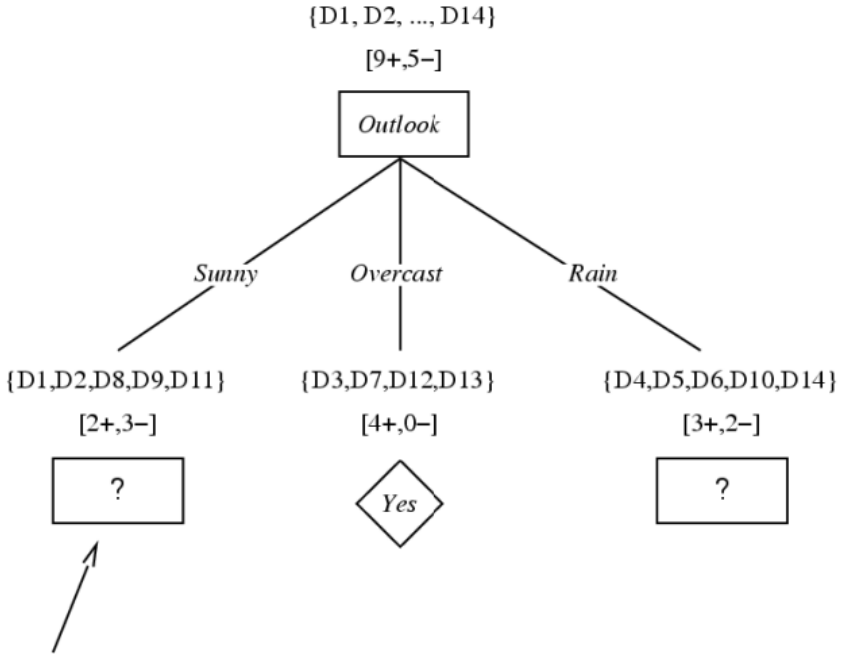


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019