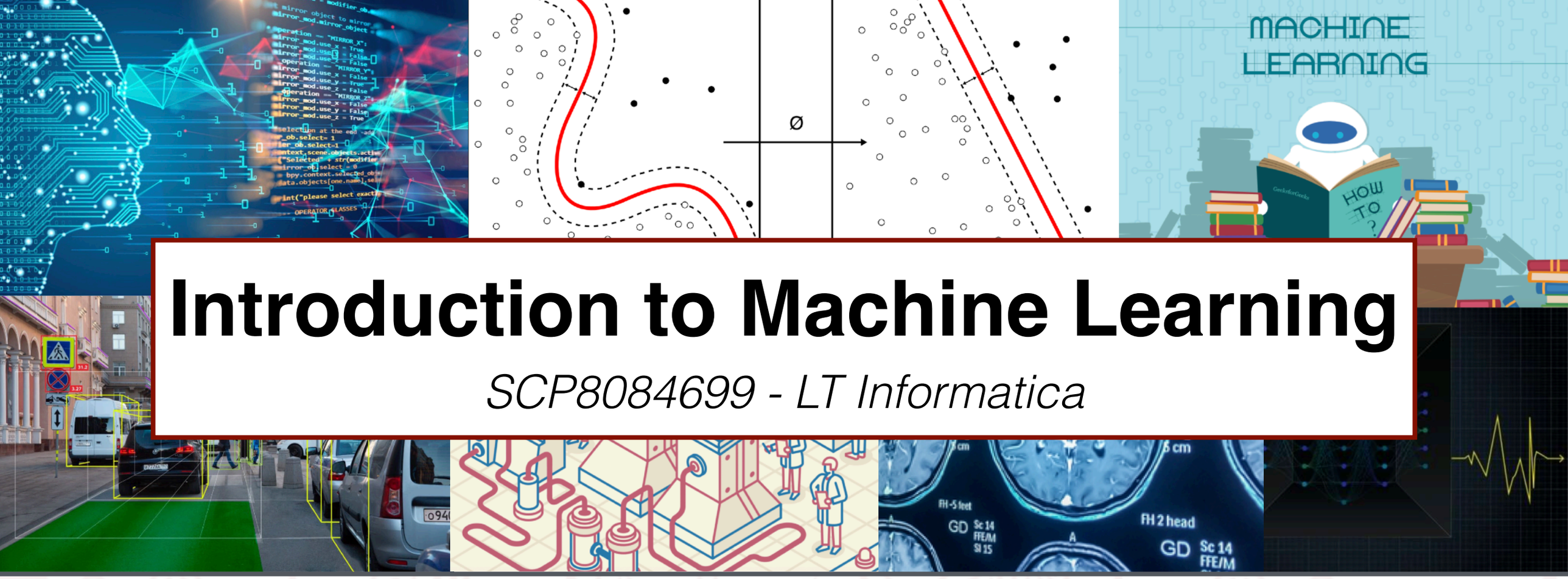



# Introduction to Machine Learning

*SCP8084699 - LT Informatica*

## Decision Trees

Prof. Lamberto Ballan

University of Padova, A.Y. 2021/22 May 5, 2022

DIPARTIMENTO MATEMATICA
Department of Mathematics “Tullio Levi-Civita”

# Course calendar: update

- Next lectures:

| | | | |
|---|---|---|---|
| L10 | W10 | Tuesday, 3 May 2022 | Non-parametric Models: kNN |
| L11 | W10 | Thursday, 5 May 2022 | Decision Trees |
| | W11 | Tuesday, 10 May 2022 | |
| Lab6 | W11 | Thursday, 12 May 2022 | Lab6: kNN, Random Forests |
| L12 | W12 | Tuesday, 17 May 2022 | Probabilistic models, NLP & information retrieval |
| L13 | W12 | Thursday, 19 May 2022 | Probabilistic models, NLP & information retrieval |
| S1 | W13 | Tuesday, 24 May 2022 | *Panel: Mathematical challenges in an AI driven world* |
| | W13 | Thursday, 26 May 2022 | |
| L14 | W14 | Tuesday, 31 May 2022 | ML & CV: Teaching machines to see |
| | W14 | Thursday, 2 June 2022 | |

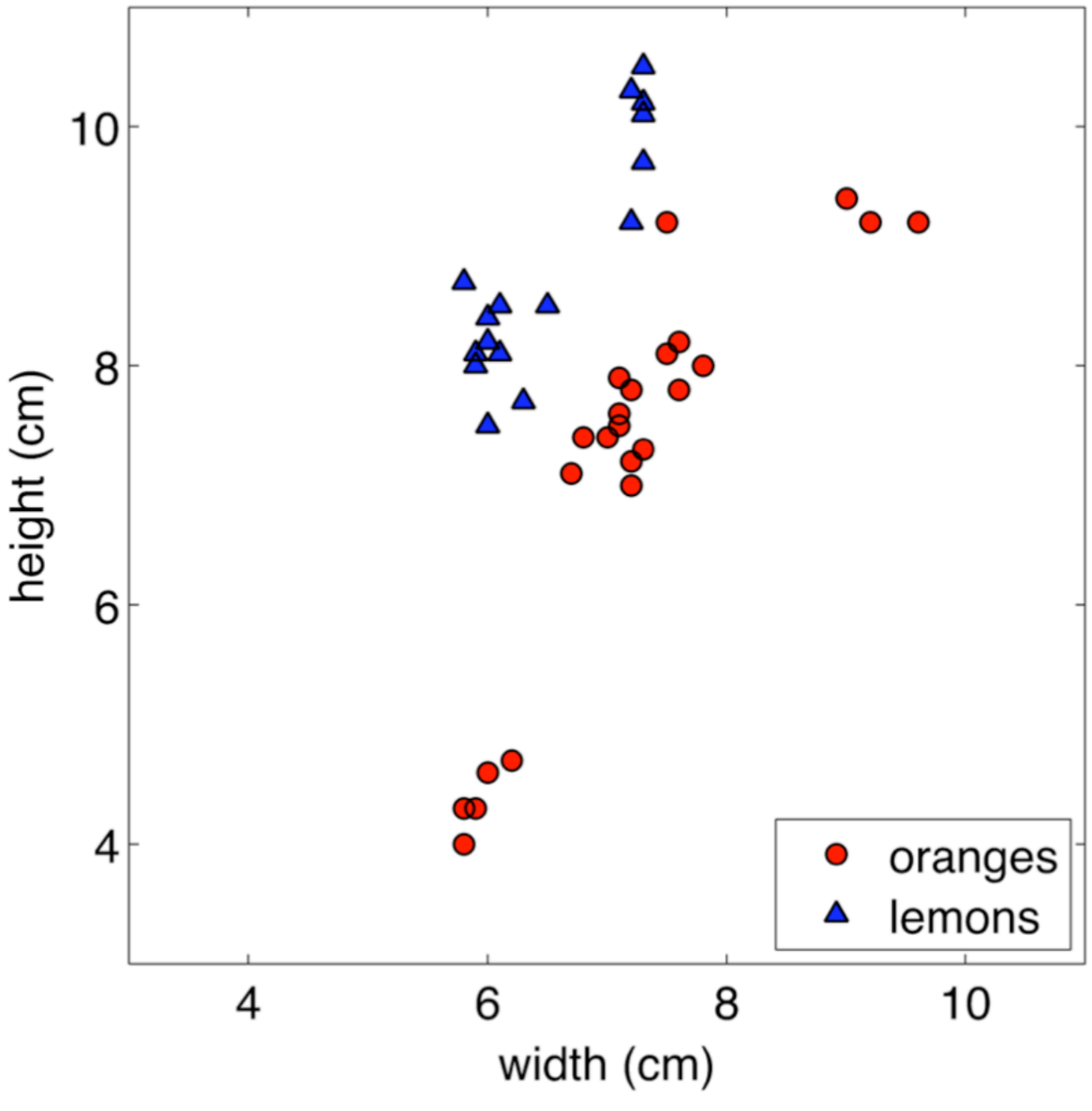
- Next exams:
  - June 30 (together with “II compitino”), 9:30 in Lum250
  - July 15, 9:30 in Lum250

# Decision Trees

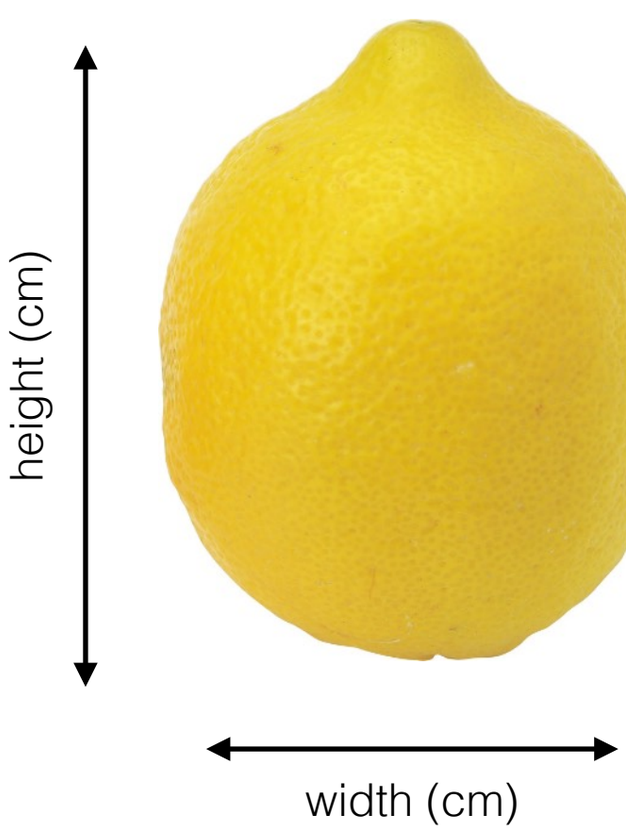
- A *decision tree* is a structure in which:
  - Internal nodes represent attributes (features)
  - Leaf nodes represent class labels or target values
- Decision trees are widely used in operations research to support decision analysis
- They are also a popular algorithm in AI/ML
  - A good property of decision trees is that they provide interpretable results
  - A decision tree represents the hypothesis function $h(\mathbf{x})$ we want to model through machine learning

# Decision Trees - Intuition

- Let's take another look at our toy classification task:

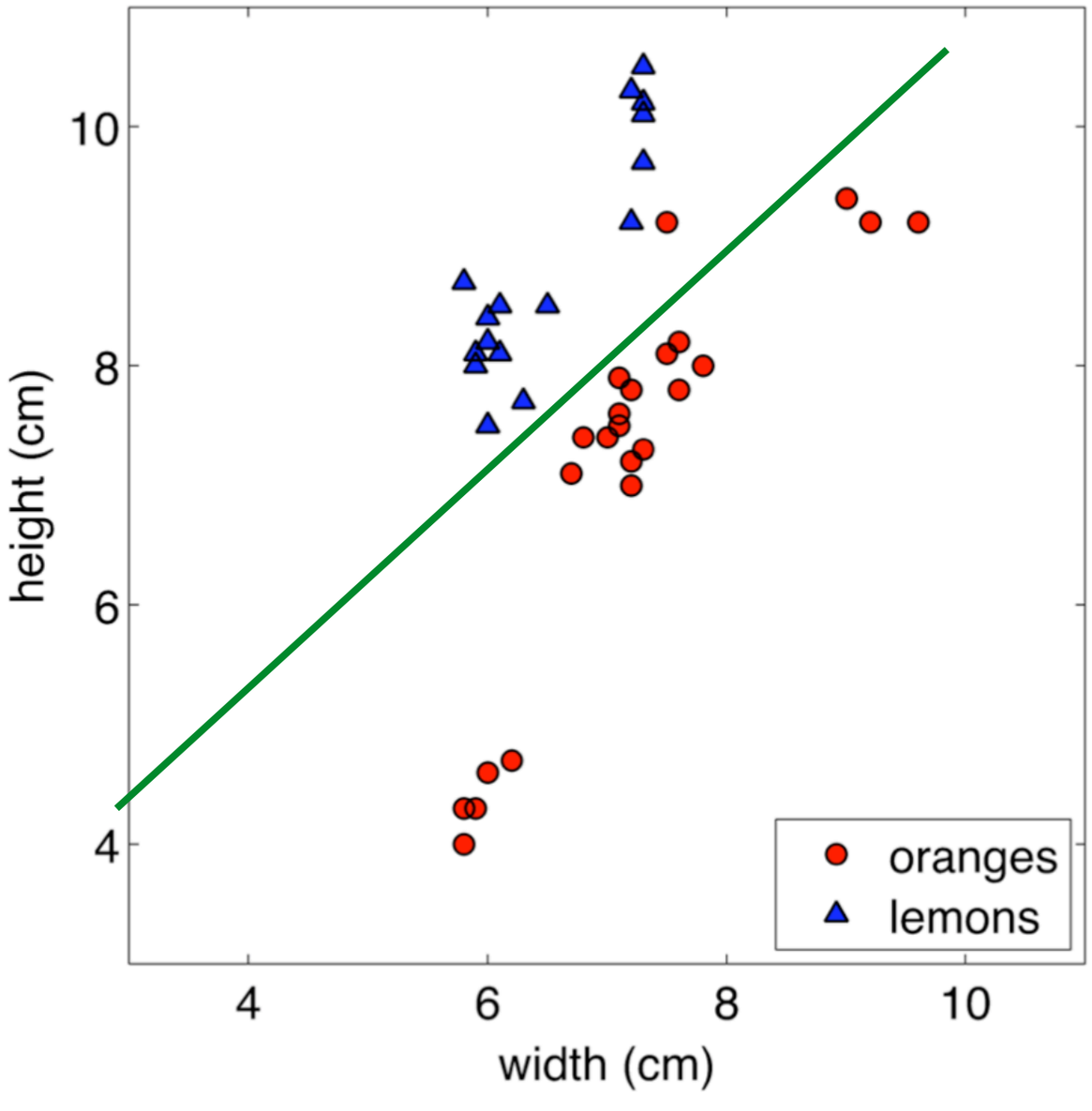



Binary classifier based on two simple features:



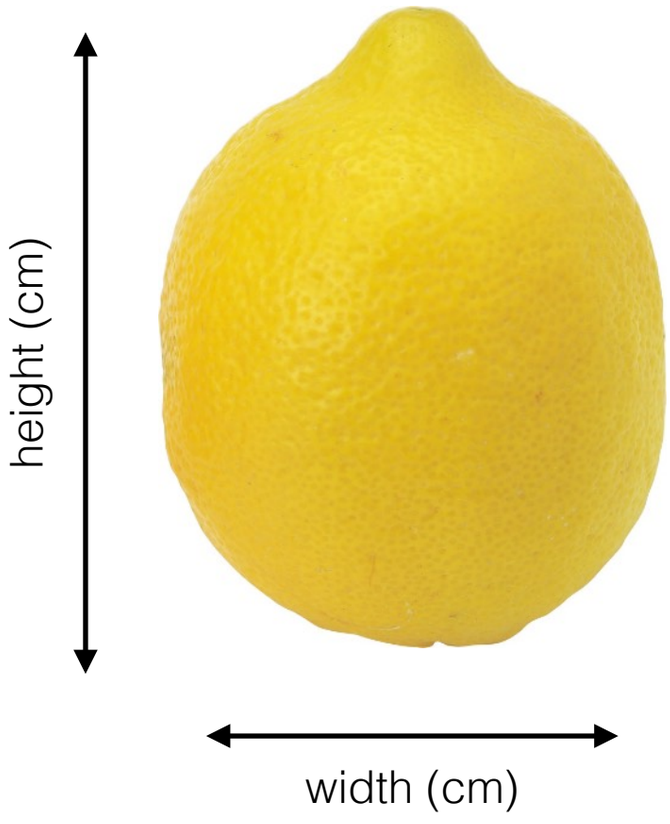
# Decision Trees - Intuition

- Linear (binary) classifier: “oranges” vs “lemons”



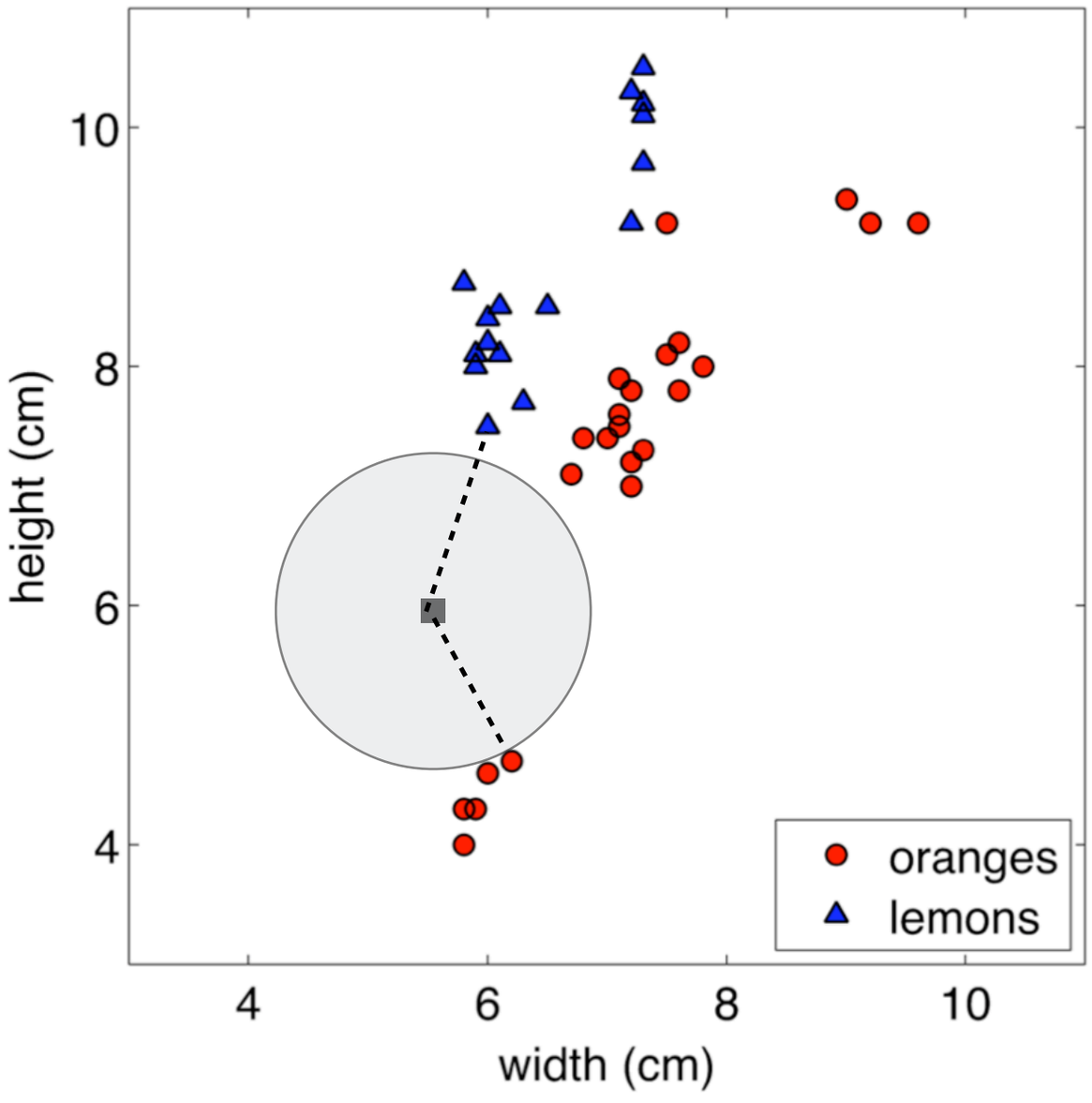

We can construct simple linear decision boundary:

$$y = \text{sign}(h_\theta(\mathbf{x})), \quad h_\theta(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\,\mathbf{x}$$

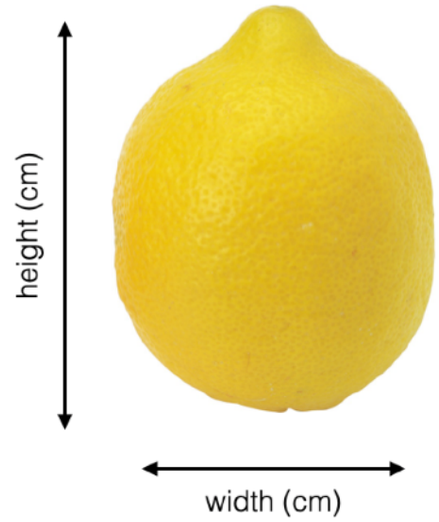


# Decision Trees - Intuition

- Non-parametric k-NN: “oranges” vs “lemons”




Binary classifier based on two simple features:




Which distance?

$$\|x_j^{(a)} - x_j^{(b)}\|_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

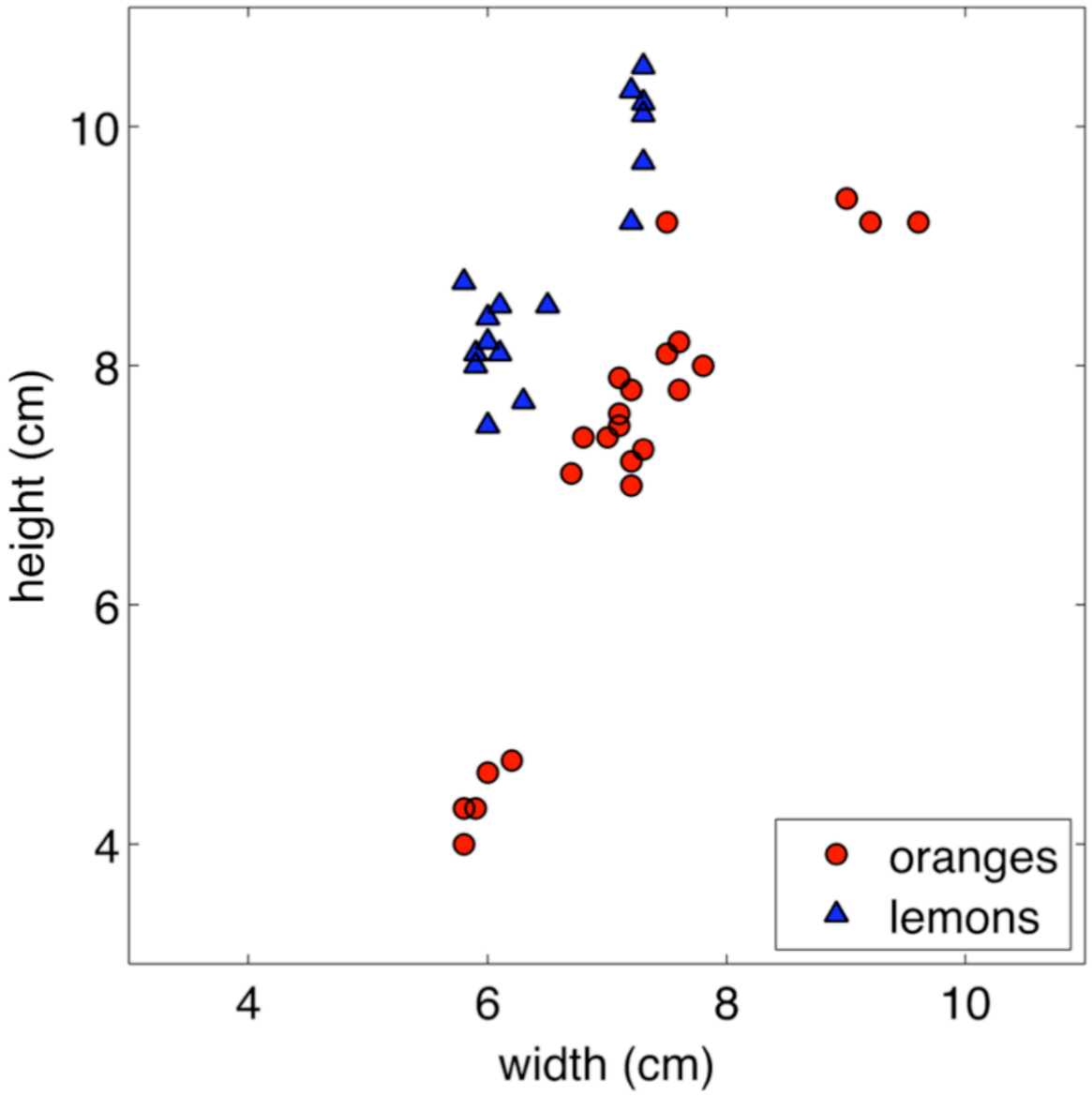
*L2 (Euclidean)*

$$\|x_j^{(a)} - x_j^{(b)}\|_1 = \sum_{j=1}^{d} |x_j^{(a)} - x_j^{(b)}|$$

*L1*

# Decision Trees - Intuition

- Decision tree classifier: “oranges” vs “lemons”




General idea:

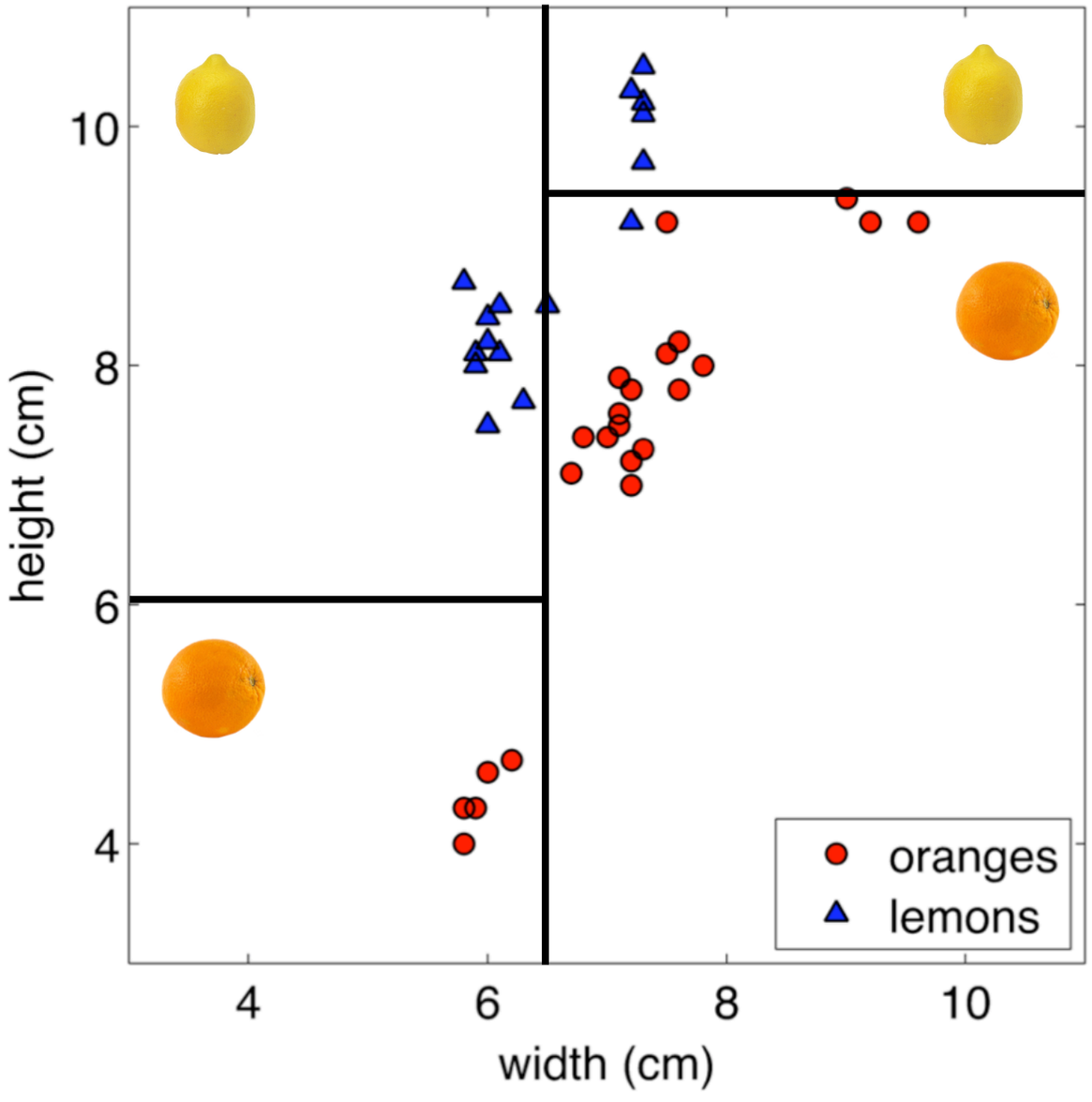
- Pick an attribute and do a simple test
- Conditioned on a choice, pick another attribute
- In the leaves, assign a class with majority vote
- Do other branches as well

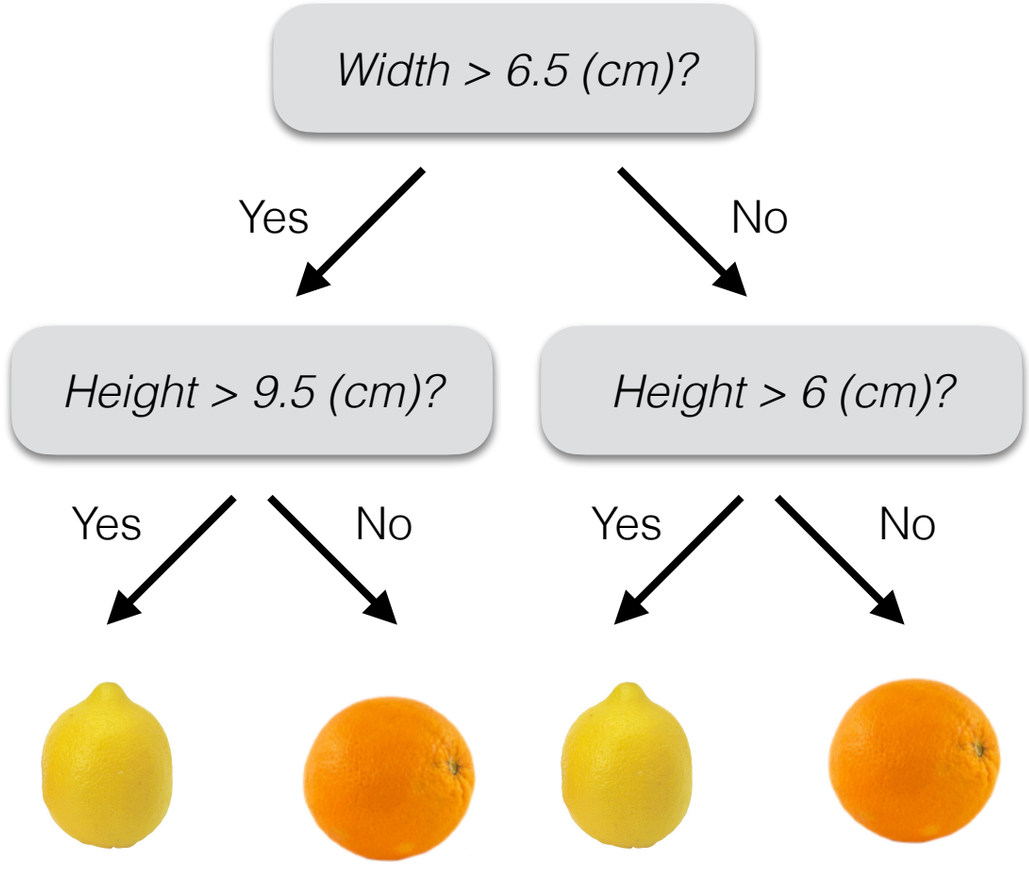


Adapted from slide by S. Fidler

# Decision Trees - Intuition

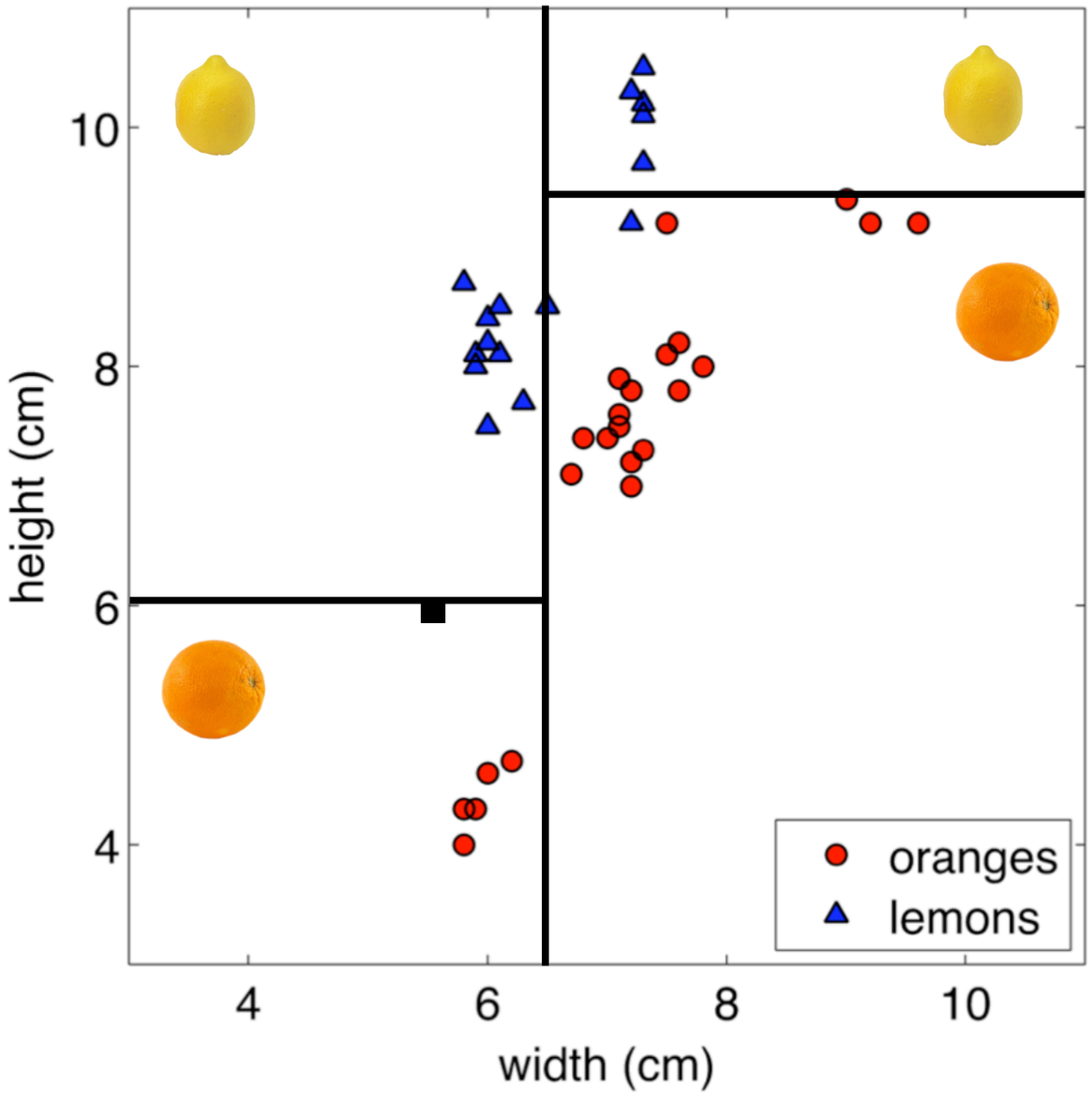
- Decision tree classifier: “oranges” vs “lemons”

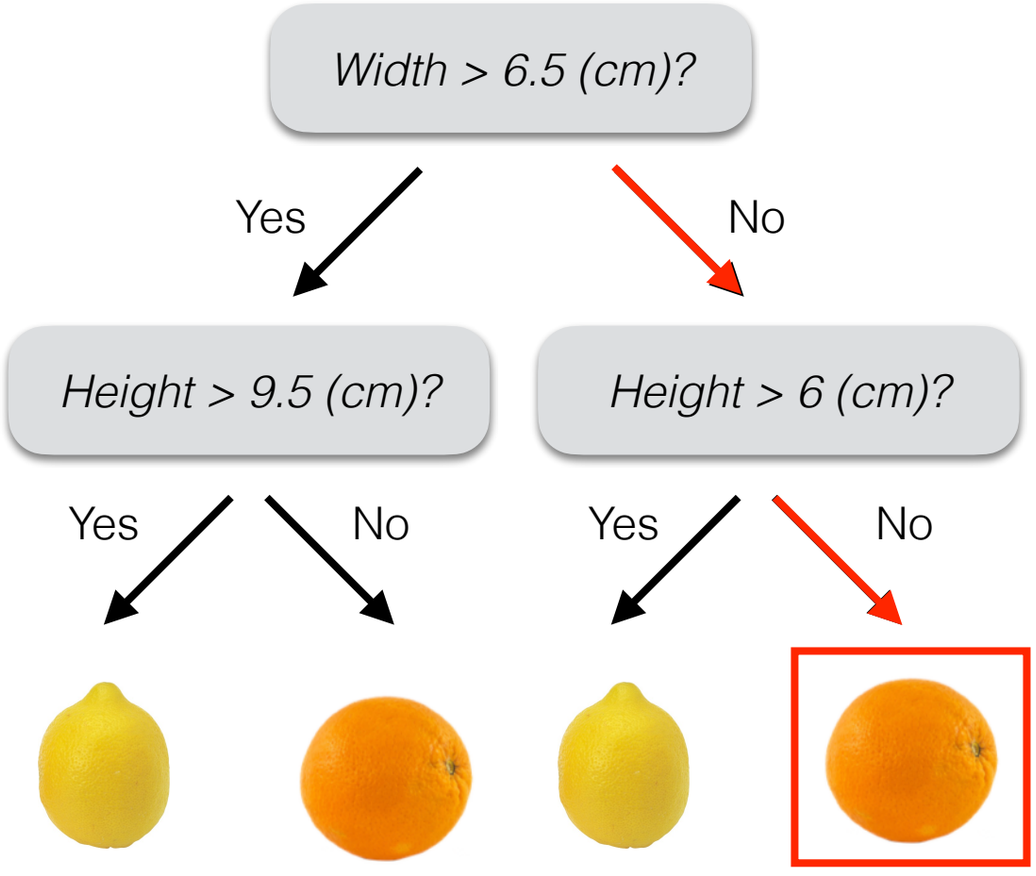


# Decision Trees - Intuition

- At test time: ■ =  → “orange”



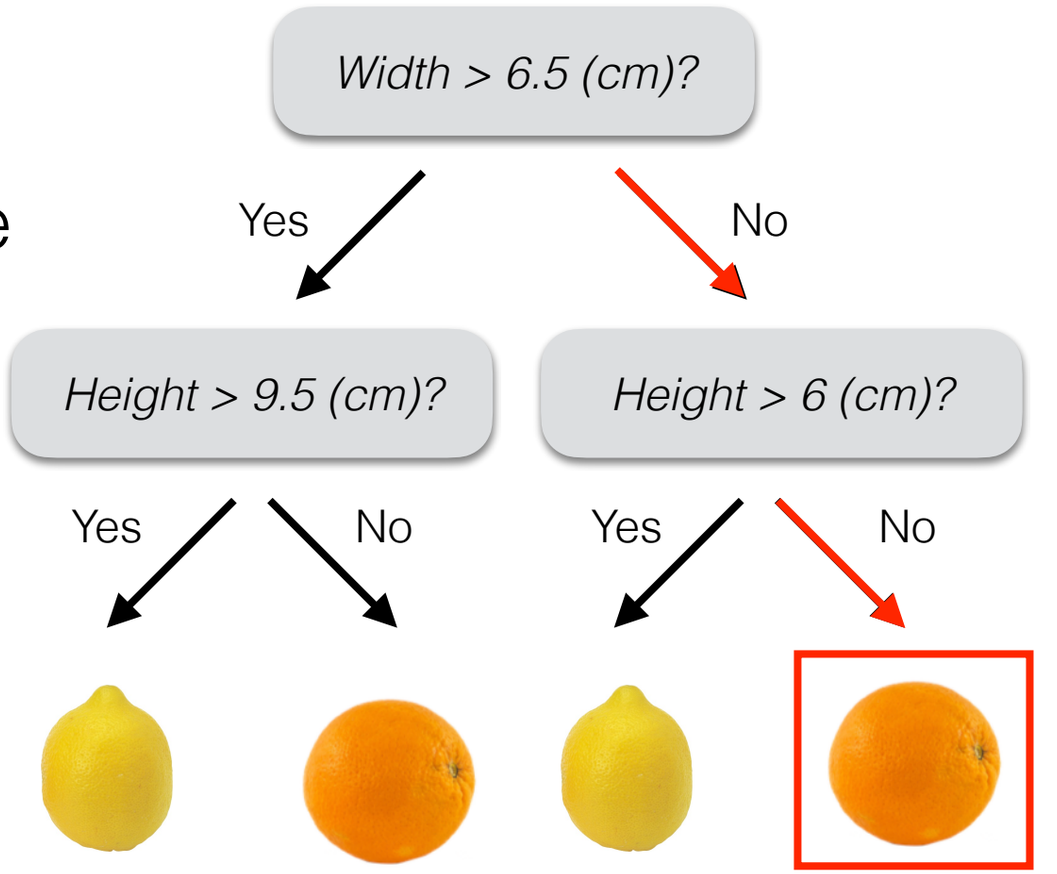






Adapted from slide by S. Fidler

# Decision Trees: ingredients

- Internal nodes
  - Related to test attributes
- Branching
  - It's determined by attribute value
- Leaf nodes
  - Outputs / Class assignments




*Note: decision trees can also be seen as generative models of induction rules from empirical data*

# Decision Trees - An example

- Let's see another example where the inputs have discrete values and the output is binary
  - We will build a decision tree to decide whether to wait for a table at a restaurant
  - This can be represented as a boolean function and our goal is to learn a definition for the goal predicate *WillWait*
  - A boolean decision tree is equivalent to the assertion that the goal attribute is true if and only if the input attributes satisfy one of the paths leading to a leaf with value true

# Decision Trees - An example

- First we list the attributes that we consider as inputs:

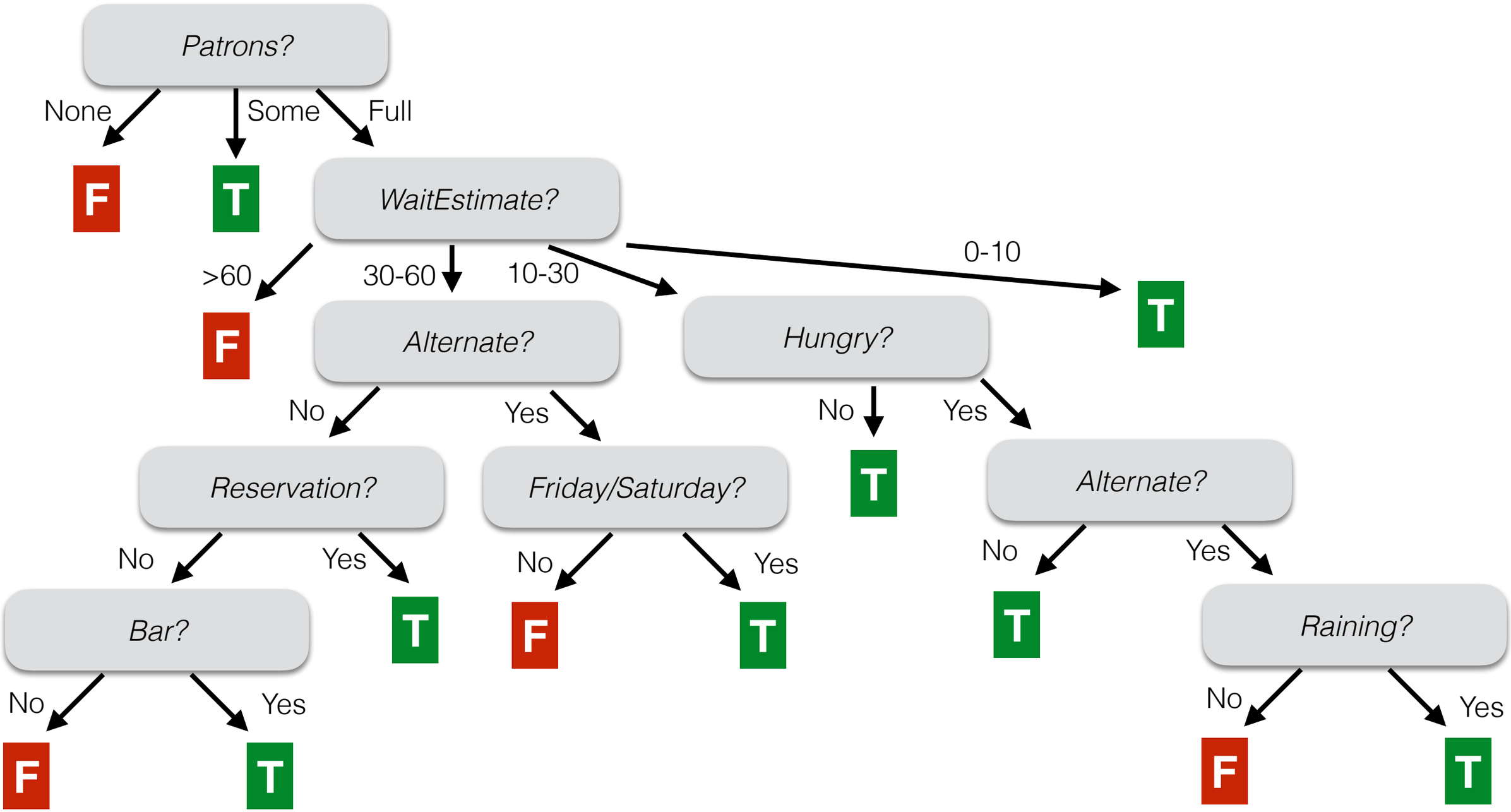
| Example | Input Attributes | | | | | | | | | | Target (*WillWait*) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | |
| $x^{(1)}$ | *Yes* | *No* | *No* | *Yes* | *Some* | *$$$* | *No* | *Yes* | *French* | *0–10* | $y^{(1)}$ |
| $x^{(2)}$ | *Yes* | *No* | *No* | *Yes* | *Full* | *$* | *No* | *No* | *Thai* | *30–60* | $y^{(2)}$ |
| $x^{(3)}$ | *No* | *Yes* | *No* | *No* | *Some* | *$* | *No* | *No* | *Burger* | *0–10* | $y^{(3)}$ |
| $x^{(4)}$ | *Yes* | *No* | *Yes* | *Yes* | *Full* | *$* | *Yes* | *No* | *Thai* | *10–30* | $y^{(4)}$ |
| $x^{(5)}$ | *Yes* | *No* | *Yes* | *No* | *Full* | *$$$* | *No* | *Yes* | *French* | *>60* | $y^{(5)}$ |
| $x^{(6)}$ | *No* | *Yes* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Italian* | *0–10* | $y^{(6)}$ |
| $x^{(7)}$ | *No* | *Yes* | *No* | *No* | *None* | *$* | *Yes* | *No* | *Burger* | *0–10* | $y^{(7)}$ |
| $x^{(8)}$ | *No* | *No* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Thai* | *0–10* | $y^{(8)}$ |
| $x^{(9)}$ | *No* | *Yes* | *Yes* | *No* | *Full* | *$* | *Yes* | *No* | *Burger* | *>60* | $y^{(9)}$ |
| $x^{(10)}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$$$* | *No* | *Yes* | *Italian* | *10–30* | $y^{(10)}$ |
| $x^{(11)}$ | *No* | *No* | *No* | *No* | *None* | *$* | *No* | *No* | *Thai* | *0–10* | $y^{(11)}$ |
| $x^{(12)}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$* | *No* | *No* | *Burger* | *30–60* | $y^{(12)}$ |

m=12 “training” examples

| | |
|---|---|
| 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 Trees - An example

- The tree to decide wether to wait (T) or not (F)



# Decision Tree: algorithm

- A tree is built by splitting the (training) dataset, into subsets which constitute the successor children
  - Choose an attribute on which to descend at each level
  - Condition on earlier (higher) scores
  - Restrict only one dimension at a time
    - Note: in the lemon/orange example, we only split each dimension once, but that is not required
  - Declare an output when you get to the bottom, i.e. the subset at a node has all the same value of the target $y$
- This procedure is repeated on each derived subset in a recursive manner (*recursive partitioning*)

# Decision Tree: classification & regression

- Each path from the root to a leaf defines a region $R_n$ of the input space
  - Let $\{(x^{(n_1)}, t^{(n_1)}), \ldots, (x^{(n_k)}, t^{(n_k)})\}$ be the training examples falling into the region $R_n$
  - Classification Tree:
    - Discrete output (class labels)
    - Leaf $y^{(k)}$ is set to the most common value in $\{(t^{(n_1)}, \ldots, t^{(n_k)})\}$
  - Regression Tree:
    - Continuous output
    - Leaf $y^{(k)}$ is set to the mean value in $\{(t^{(n_1)}, \ldots, t^{(n_k)})\}$



Adapted from slide by S. Russell

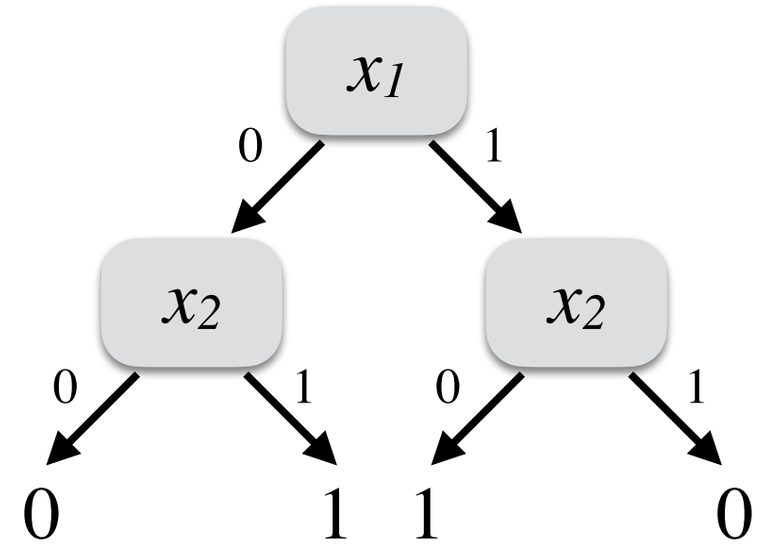
# Decision Tree: expressiveness

- Discrete-input, discrete output:
  - A tree can express any function of the input attributes
  - E.g., for logical functions: truth table = path to leaf

$x_1, x_2 \in \{0, 1\}$

$y = x_1$ XOR $x_2$

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




- Continuous-input, continuous-output:
  - A tree can approximate any function (arbitrarily closely)

# Decision Trees & Logical Functions

- A decision tree represents a boolean function
  - Each path from root to leaf represents a conjunction of tests on attributes
  - Different paths leading to the same classification represent a disjunction of conjunctions

- These two rules define a series of DNF (Disjunctive Normal Form), one for each class

# Decision Trees & Logical Functions

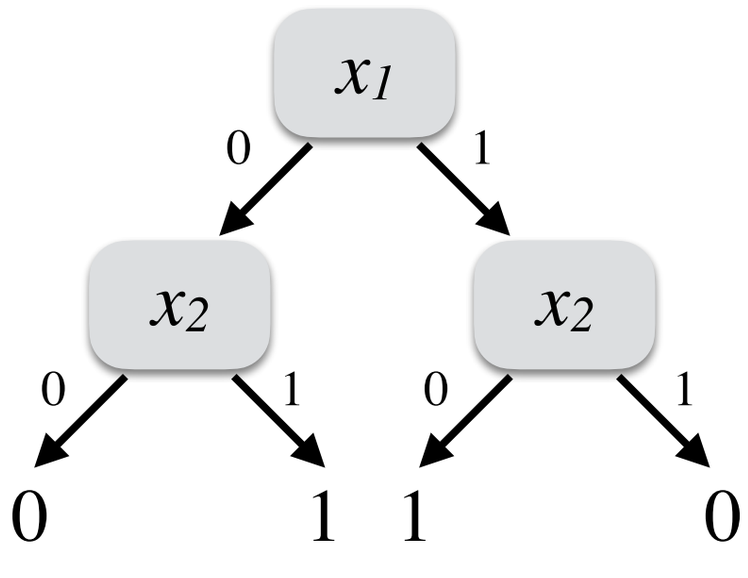
- A decision tree represents a boolean function
  - Each path from root to leaf represents a conjunction of tests on attributes
  - Different paths leading to the same classification represent a disjunction of conjunctions

- An example:

$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ XOR } x_2$$

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




DNF for $y = 1$

$(x_1=0 \text{ AND } x_2=1) \text{ OR } (x_1=1 \text{ AND } x_2=0)$

# Learning Decision Trees

- Trivially, there is a decision tree for any training set with one path to leaf for each sample

  *(unless $f$ non deterministic in $x$)*

  - However, it probably won't generalize to new examples
  - We need some kind of regularization to ensure more compact (and useful) decision trees

- How do we construct a useful decision tree?

# Learning Decision Trees

- Learning the simplest (smallest) decision tree is a NP complete problem [check: Hyafil & Rivest '76]
  - Resort to a greedy heuristic:
    - Start from an empty decision tree
    - Split on next best attribute
    - Recurse
  - What is best attribute?
    - We use *information theory* to guide us

# ID3 Algoritm

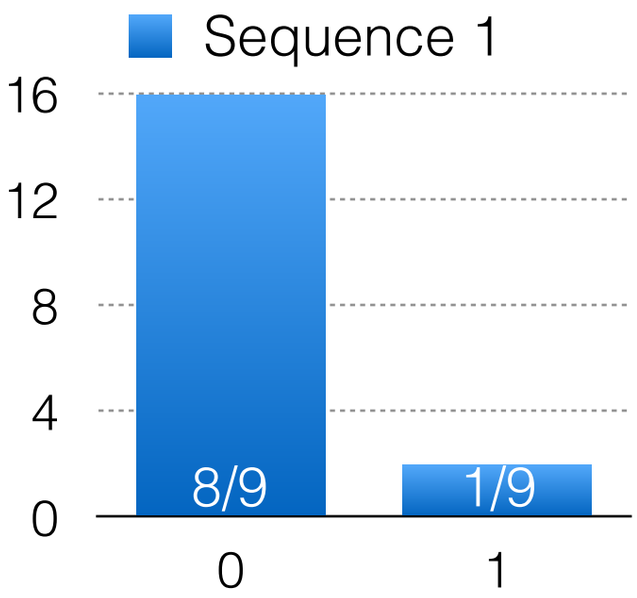
- ID3 is a popular algorithm for learning decision trees
  *(S: training examples, A: attributes)*
  - It starts by creating a root node $r$ for the tree
  - If all examples in $S$ belong to the same class $y$, return the root note $r$ with label $y$
  - If $A$ is empty, return $r$ labeled as the majority class in $S$
  - Otherwise, select the "optimal" attribute $a \in A$
    - Partition/split the set $S$ into subsets using $a$ for which the resulting *entropy* is minimized (or *infogain* is maximized)
    - Make a decision tree node containing that attribute $a$
    - Recurse on subsets using the remaining attributes

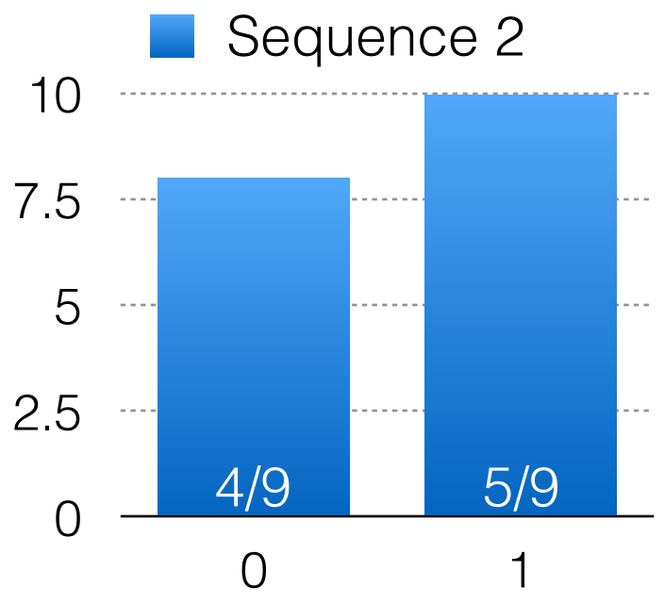
# Choosing a good attribute

- Let's take a slight detour and recall some concepts from information theory
  - Quantifying uncertainty (a simple example):

Sequence 1: 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

Sequence 2: 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 1 0 0




# Quantifying uncertainty: entropy

- Entropy $H$ is a measure of the amount of uncertainty (or randomness) in the dataset $\mathbf{S}$
  - It allows us to answer to questions such as:
    - How surprised are we by a new value in the sequence?
    - How much information does it convey?
  - Entropy is defined as:

$$H(\mathbf{S}) = -\sum_{\mathbf{c} \in \mathbf{C}} p(\mathbf{c}) \log_2 p(\mathbf{c})$$ , where:

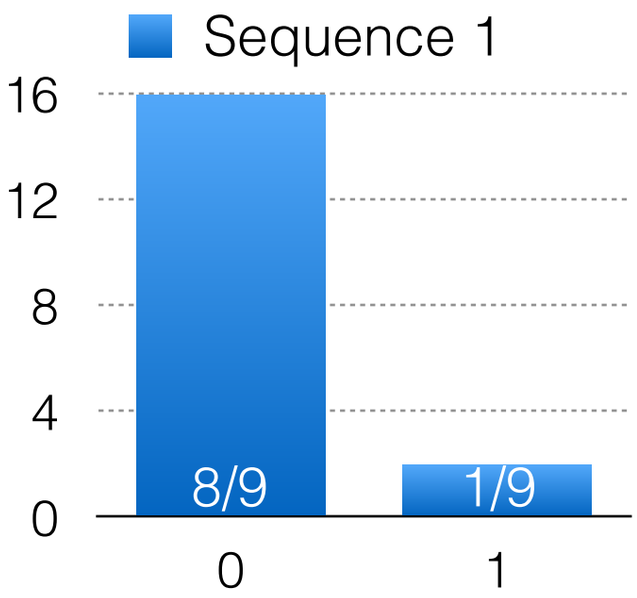
$\mathbf{C}$ is the set of classes in $\mathbf{S}$

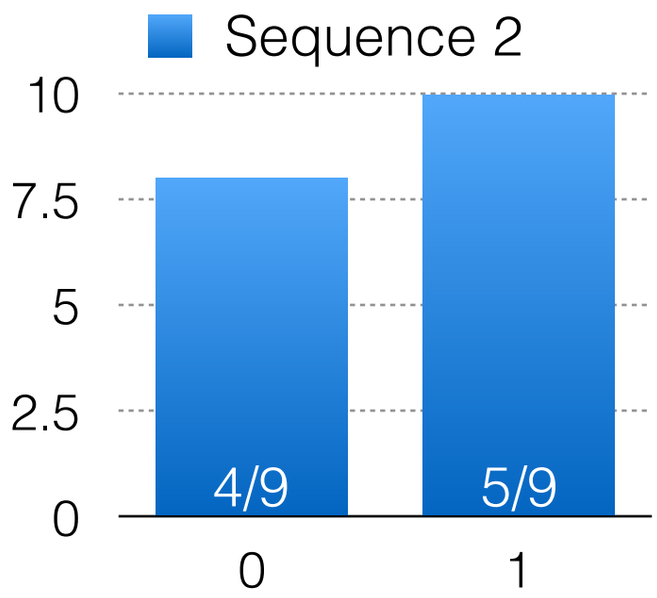
$p(\mathbf{c})$ is the proportion of elements in $\mathbf{c}$ to the elements in $\mathbf{S}$

# Quantifying uncertainty: entropy

- Entropy $H$ is a measure of the amount of uncertainty (or randomness) in the dataset $\mathrm{S}$
  - Entropy is defined as: $H(\mathrm{S}) = -\sum_{\mathrm{c} \in \mathrm{C}} p(\mathrm{c}) \log_2 p(\mathrm{c})$

$$H(\mathrm{S}_1) = -\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx 0.5 \qquad H(\mathrm{S}_2) = -\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$




# Quantifying uncertainty: entropy

- Entropy $H$ is a measure of the amount of uncertainty (or randomness) in the dataset $\mathrm{S}$
  - Entropy is defined as: $H(\mathrm{S}) = -\sum_{\mathrm{c}\in\mathrm{C}} p(\mathrm{c}) \log_2 p(\mathrm{c})$

This is a binary classification problem: $\mathrm{c}\in\{\mathrm{c}_1, \mathrm{c}_2\}$


Therefore, it will help to define $B(\mathrm{q})$ as the entropy of a Boolean random variable that is true with probability $\mathrm{q}$:

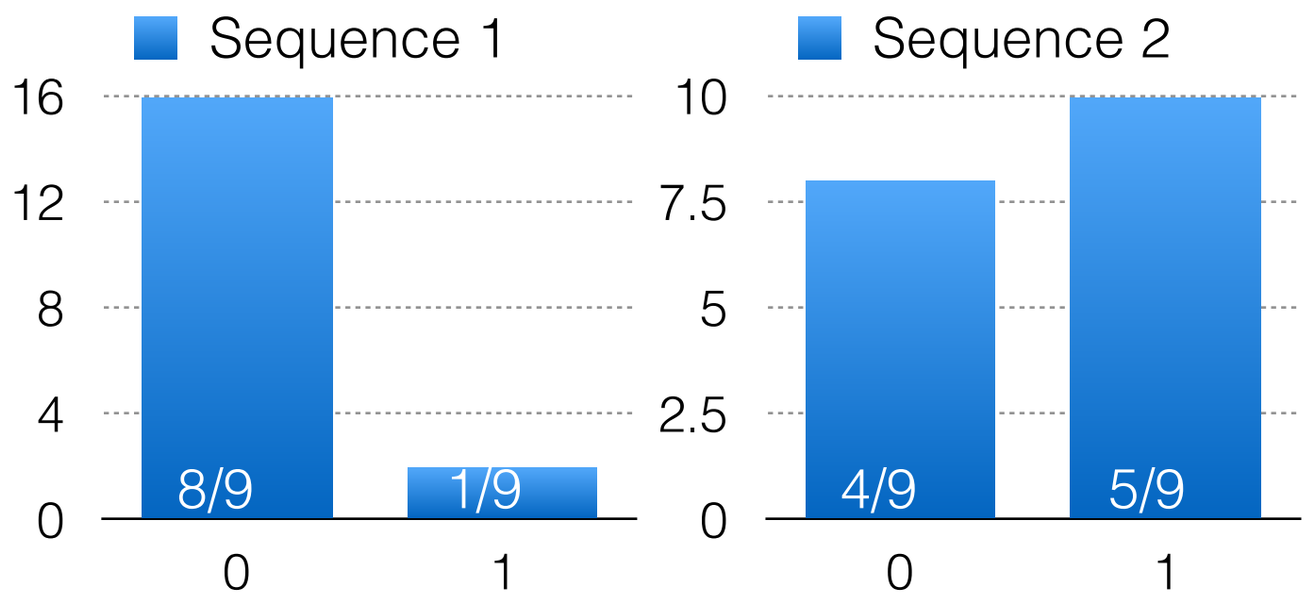
$$B(\mathrm{q}) = -(\mathrm{q}\log_2\mathrm{q} + (1-\mathrm{q})\log_2(1-\mathrm{q}))$$

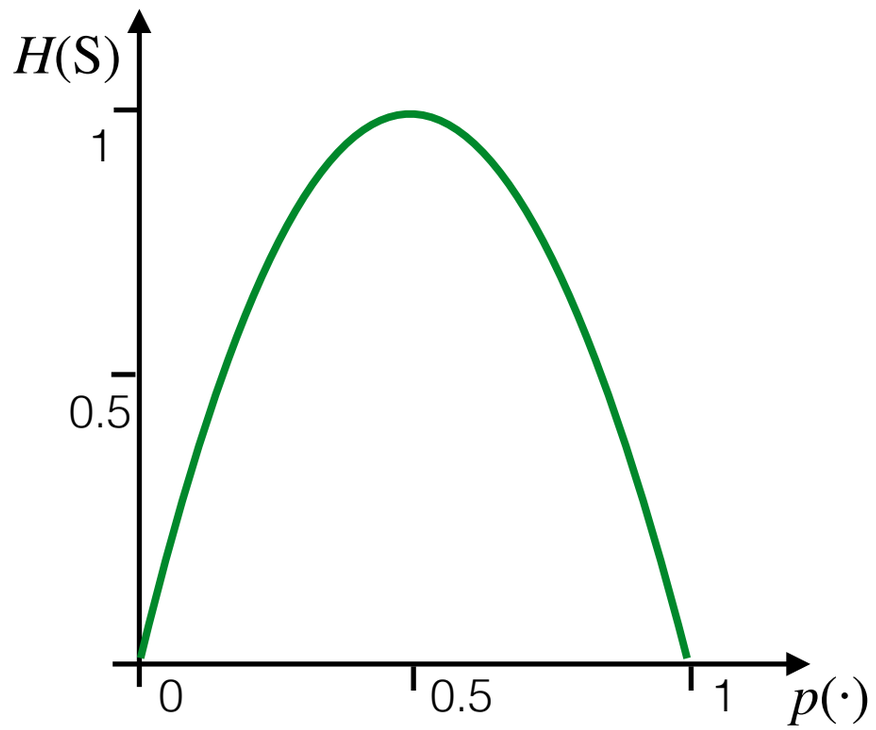
$$H(\mathrm{S}_1) = B(\frac{8}{9}) \approx 0.5 \qquad H(\mathrm{S}_2) = B(\frac{4}{9}) \approx 0.99$$

# Quantifying uncertainty: entropy

- Entropy $H$ is a measure of the amount of uncertainty (or randomness) in the dataset $\mathbf{S}$
  - Entropy is defined as: $H(\mathbf{S}) = -\sum_{c \in \mathbf{C}} p(c) \log_2 p(c)$

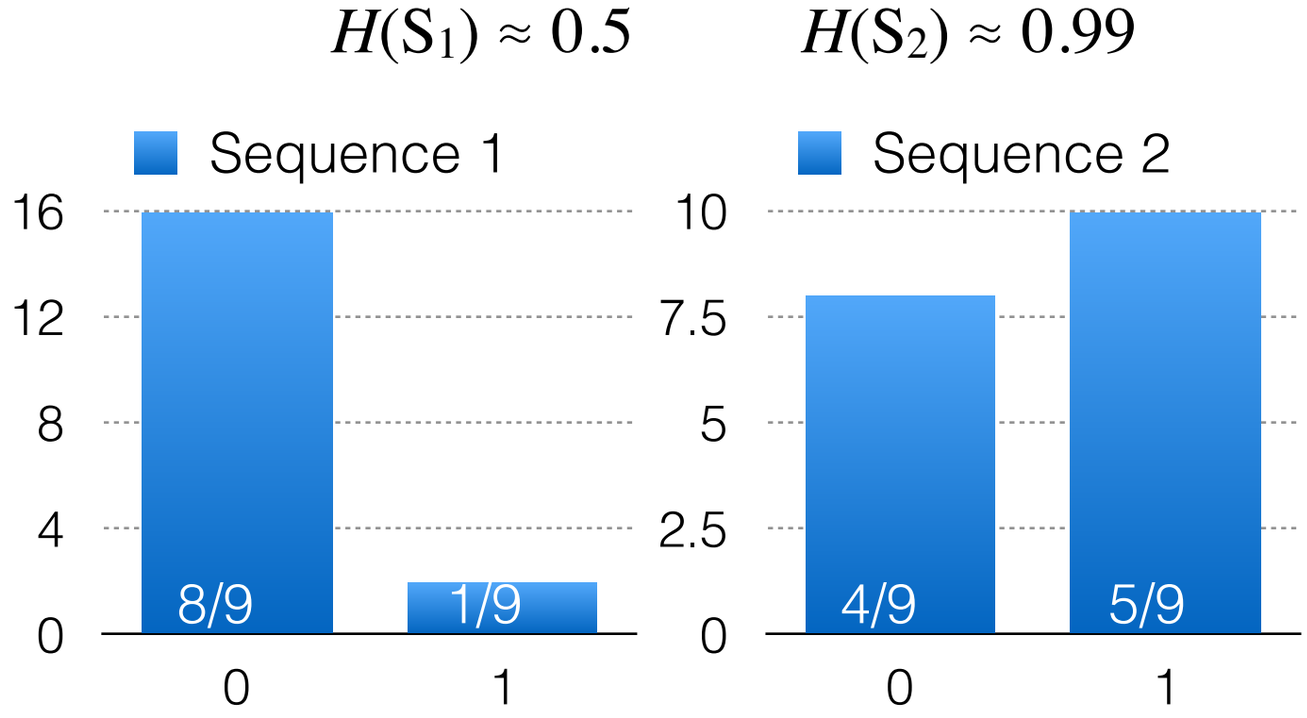
$$H(\mathbf{S}_1) = B(\frac{8}{9}) \approx 0.5 \qquad H(\mathbf{S}_2) = B(\frac{4}{9}) \approx 0.99$$

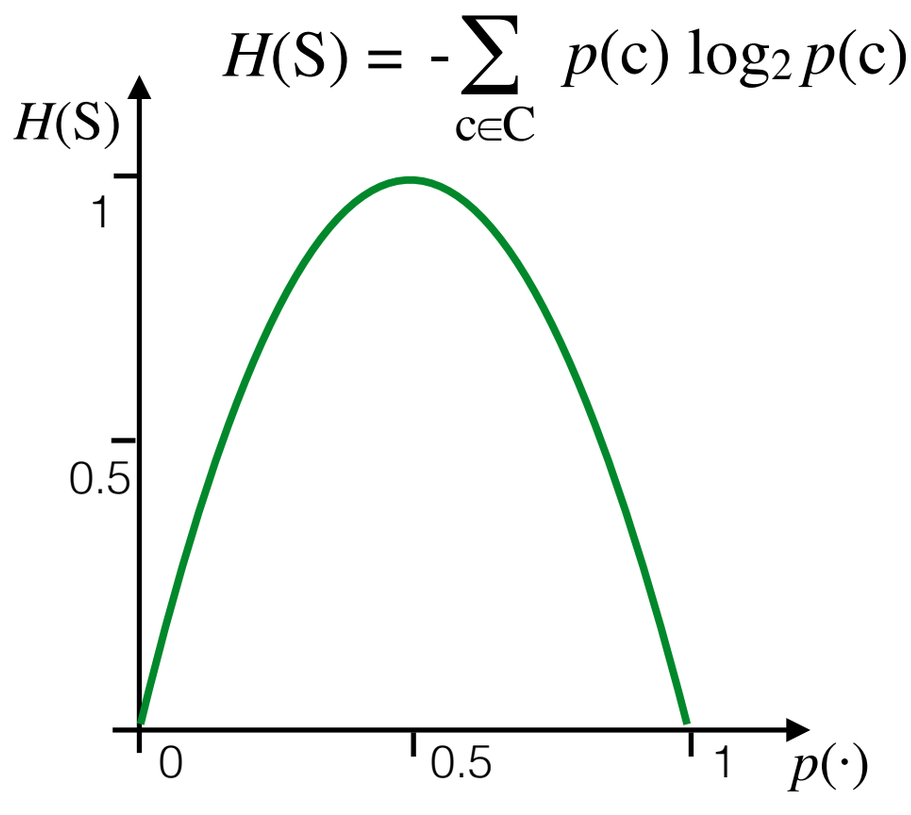


# Quantifying uncertainty: entropy

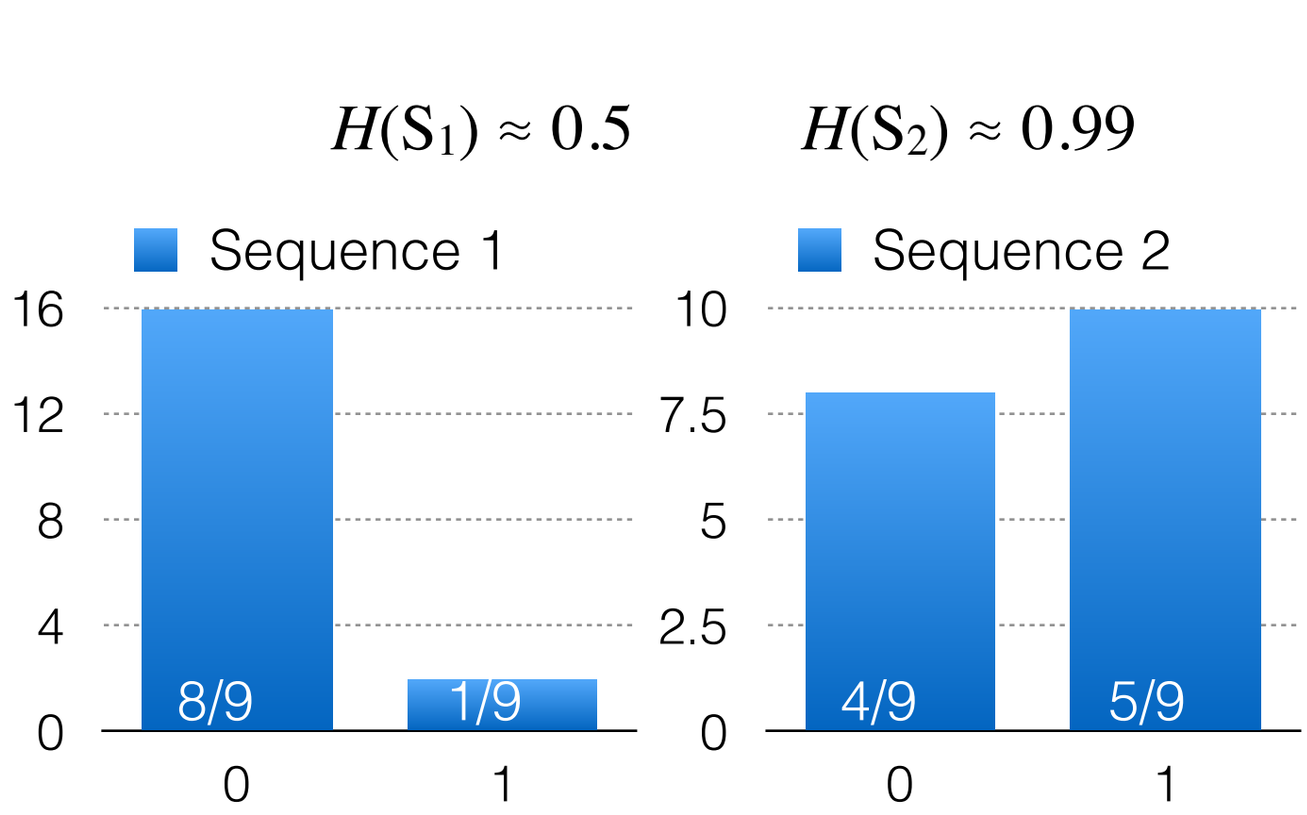
- High entropy:
  - Variable has a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable

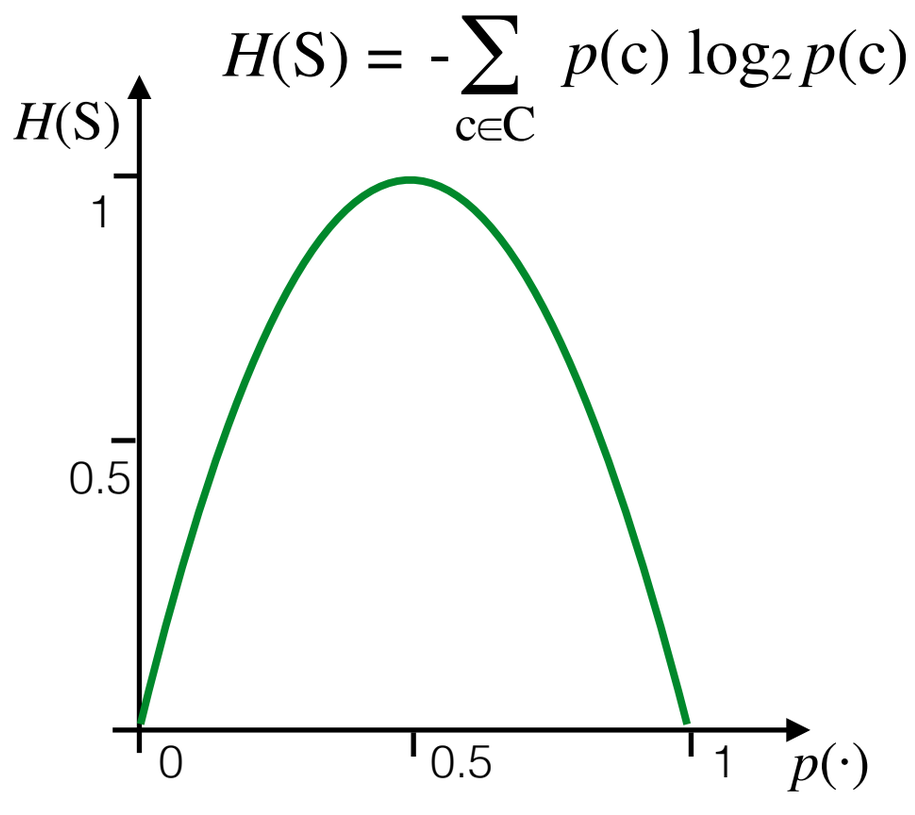


# Quantifying uncertainty: entropy

- Low entropy:
  - Distribution of variable has (many) peaks and valleys
  - Histogram has (many) lows and highs
  - Values sampled from it are more predictable



# Information Gain

- Information Gain measures the difference in entropy from before to after the set $S$ is split on an attribute $a$
  - In other words, IG measures how much uncertainty in $S$ was reduced after splitting set $S$ on attribute $a$
  - IG is defined as: $IG(S, a) = H(S) - \sum_{t \in T} p(t)\, H(t) = H(S) - H(S \mid a)$

$H(S)$ is the entropy of dataset $S$

$T$: the subsets created from splitting set $S$ by attribute $a$

$p(t)$ is the proportion of elements in $t$ to the elements in $S$

$H(t)$ is the entropy of subset $t$

*In ID3, infogain is calculated for each remaining attribute; the attribute with the largest IG is used to split the set on this iteration*

# Learning Decision Trees with ID3

- ID3 is a popular algorithm for learning decision trees
  *($\mathbf{S}$: training examples, $\mathbf{A}$: attributes)*
  - It starts by creating a root node $\mathbf{r}$ for the tree
  - If all examples in $\mathbf{S}$ belong to the same class $y$, return the root note $\mathbf{r}$ with label $y$
  - If $\mathbf{A}$ is empty, return $\mathbf{r}$ labeled as the majority class in $\mathbf{S}$
  - Otherwise, select the "optimal" attribute $\mathbf{a} \in \mathbf{A}$
    - Partition/split the set $\mathbf{S}$ into subsets using $\mathbf{a}$ for which the resulting *information gain* is maximized
    - Make a decision tree node containing that attribute $\mathbf{a}$
    - Recurse on subsets using the remaining attributes

# Back to our example…

- First we list the attributes that we consider as inputs:

| Example | Input Attributes | | | | | | | | | | Target (*WillWait*) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | |
| $x^{(1)}$ | *Yes* | *No* | *No* | *Yes* | *Some* | *$$$* | *No* | *Yes* | *French* | *0–10* | $y^{(1)}$ = True |
| $x^{(2)}$ | *Yes* | *No* | *No* | *Yes* | *Full* | *$* | *No* | *No* | *Thai* | *30–60* | $y^{(2)}$ = False |
| $x^{(3)}$ | *No* | *Yes* | *No* | *No* | *Some* | *$* | *No* | *No* | *Burger* | *0–10* | $y^{(3)}$ = True |
| $x^{(4)}$ | *Yes* | *No* | *Yes* | *Yes* | *Full* | *$* | *Yes* | *No* | *Thai* | *10–30* | $y^{(4)}$ = True |
| $x^{(5)}$ | *Yes* | *No* | *Yes* | *No* | *Full* | *$$$* | *No* | *Yes* | *French* | *>60* | $y^{(5)}$ = False |
| $x^{(6)}$ | *No* | *Yes* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Italian* | *0–10* | $y^{(6)}$ = True |
| $x^{(7)}$ | *No* | *Yes* | *No* | *No* | *None* | *$* | *Yes* | *No* | *Burger* | *0–10* | $y^{(7)}$ = False |
| $x^{(8)}$ | *No* | *No* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Thai* | *0–10* | $y^{(8)}$ = True |
| $x^{(9)}$ | *No* | *Yes* | *Yes* | *No* | *Full* | *$* | *Yes* | *No* | *Burger* | *>60* | $y^{(9)}$ = False |
| $x^{(10)}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$$$* | *No* | *Yes* | *Italian* | *10–30* | $y^{(10)}$ = False |
| $x^{(11)}$ | *No* | *No* | *No* | *No* | *None* | *$* | *No* | *No* | *Thai* | *0–10* | $y^{(11)}$ = False |
| $x^{(12)}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$* | *No* | *No* | *Burger* | *30–60* | $y^{(12)}$ = True |

m=12 “training” examples

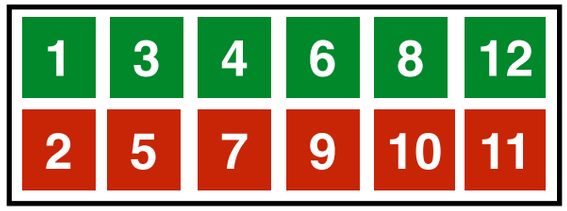
| | |
|---|---|
| 1. | Alternate: whether there is a suitable alternative restaurant nearby. |
| 2. | Bar: whether the restaurant has a comfortable bar area to wait in. |
| 3. | Fri/Sat: true on Fridays and Saturdays. |
| 4. | Hungry: whether we are hungry. |
| 5. | Patrons: how many people are in the 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). |

# Back to our example…

- Attribute selection:

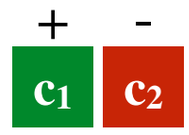
[ recall: $IG(\mathrm{S}, \mathrm{a}) = H(\mathrm{S}) - H(\mathrm{S} \mid \mathrm{a})$ ]

$$H(\mathrm{S}) = -\sum_{\mathrm{c} \in \mathrm{C}} p(\mathrm{c}) \log_2 p(\mathrm{c})$$




Dataset S of m=12 examples

This is a binary classification problem: $\mathrm{c} \in \{\mathrm{c}_1, \mathrm{c}_2\}$




Therefore it will help to define $B(\mathrm{q})$ as the entropy of a Boolean random variable that is true with probability q:

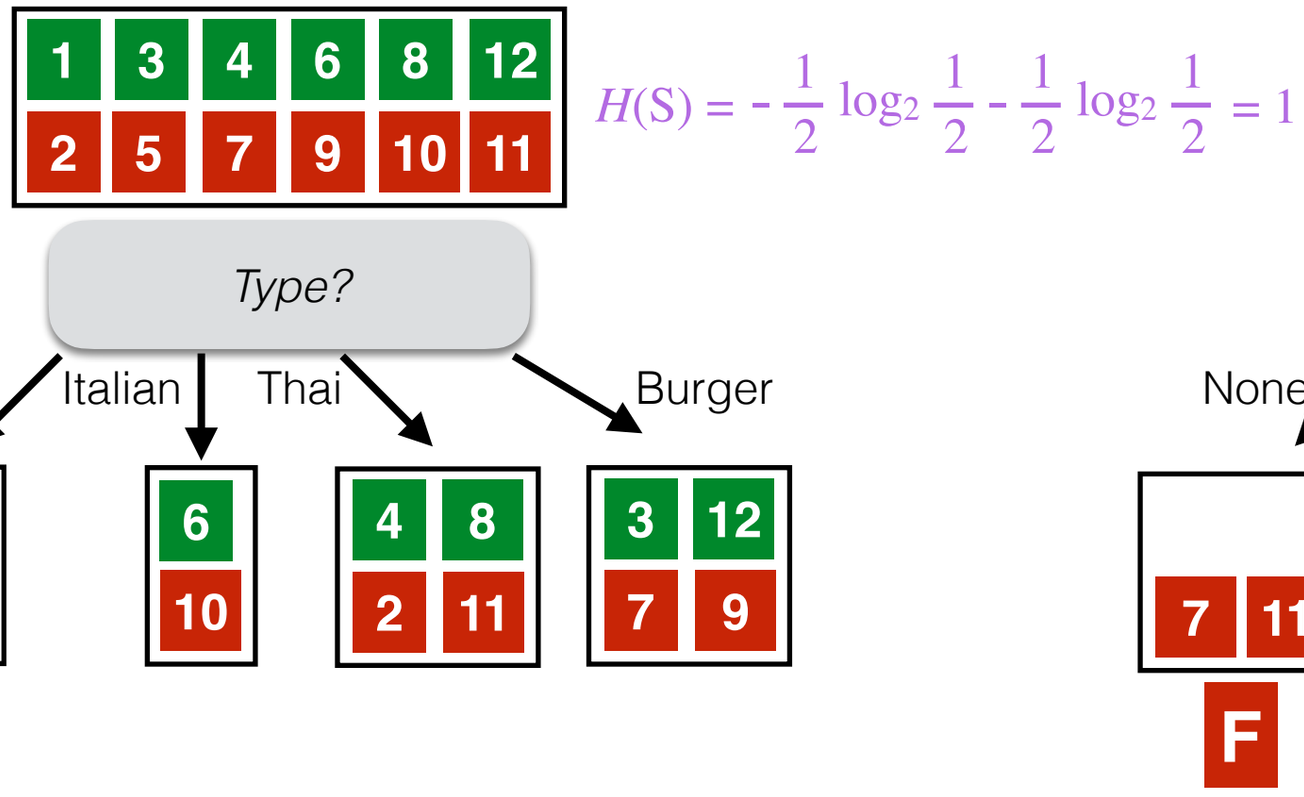
$$B(\mathrm{q}) = -(\mathrm{q} \log_2 \mathrm{q} + (1-\mathrm{q}) \log_2 (1-\mathrm{q})) \text{ , where } \mathrm{q} = \frac{|\mathrm{c}_1|}{|\mathrm{c}_1| + |\mathrm{c}_2|}$$

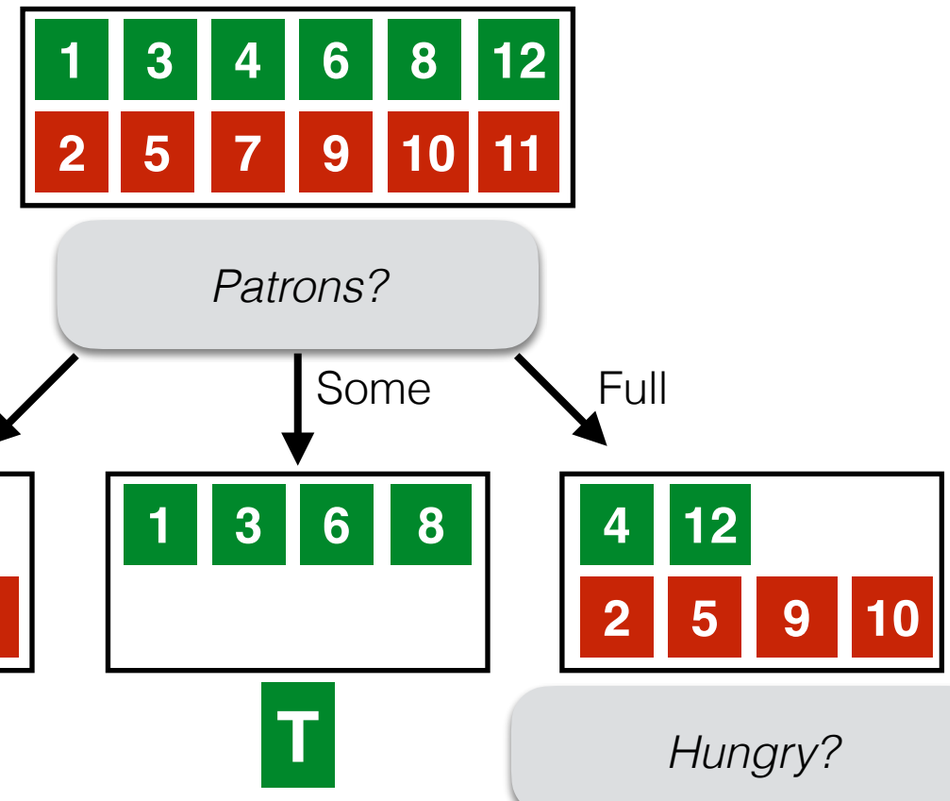
$$H(\mathrm{S}) = B(\tfrac{1}{2}) = -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{2} \log_2 \tfrac{1}{2} = 1$$

# Back to our example…

- Attribute selection:

[ recall: $IG(\mathrm{S}, \mathrm{a}) = H(\mathrm{S}) - H(\mathrm{S} \mid \mathrm{a})$ ]



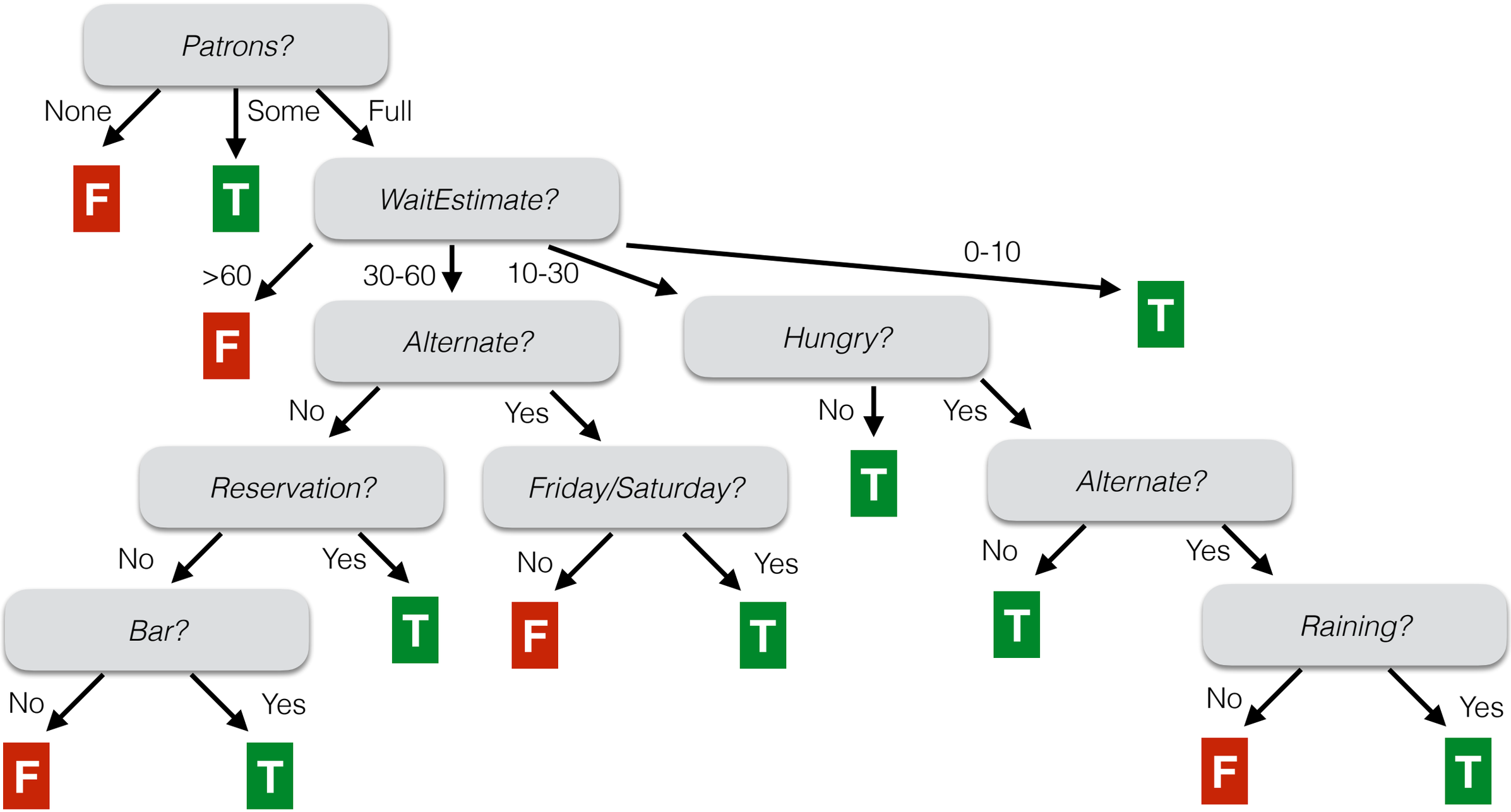

$$IG(\text{Type}) = 1 - [\frac{2}{12}H(\mathrm{Y} \mid \text{Fre}) + \frac{2}{12}H(\mathrm{Y} \mid \text{Ita}) + \frac{4}{12}H(\mathrm{Y} \mid \text{Tha}) + \frac{4}{12}H(\mathrm{Y} \mid \text{Bur})]$$

$$IG(\text{Type}) = 1 - [\frac{2}{12}B(\frac{1}{2}) + \frac{2}{12}B(\frac{1}{2}) + \frac{4}{12}B(\frac{2}{4}) + \frac{4}{12}B(\frac{2}{4})] = 0$$

$$IG(\text{Patrons}) = 1 - [\frac{2}{12}B(\frac{0}{2}) + \frac{4}{12}B(\frac{4}{4}) + \frac{6}{12}B(\frac{2}{6})] \approx 0.54$$

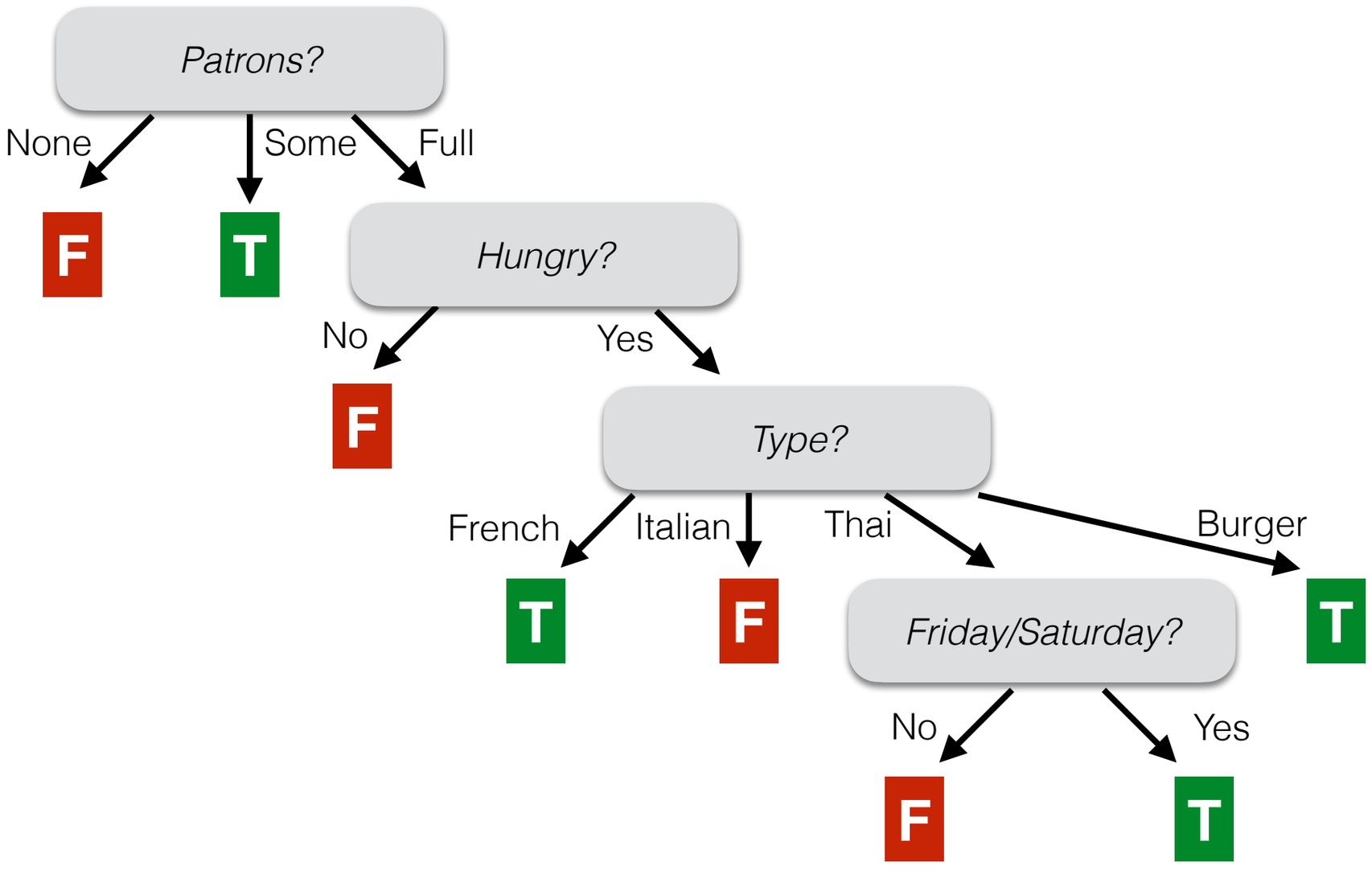
# Which tree is better?

- The tree to decide wether to wait (T) or not (F)



# Which tree is better?

- The tree to decide wether to wait (T) or not (F)



# What makes a good tree?

- Not too small
  - Need to handle important but subtle distinctions in data
- Not too big
  - Computational efficiency (avoid redundant attributes)
  - Avoid overfitting training examples
- Occam's Razor: find the simplest hypothesis
  *(i.e. the smallest tree that fits the observations)*
- Inductive Bias:
  - Small trees that place high information gain attributes close to the root are preferred

# Decision Trees: issues, challenges

- Problems:
  - You have exponentially less data at lower levels
  - Too big of a tree can overfit the data
  - Greedy algorithms usually don't yield the global optimum
  - Information gain privileges the attributes that assume a broad range of values

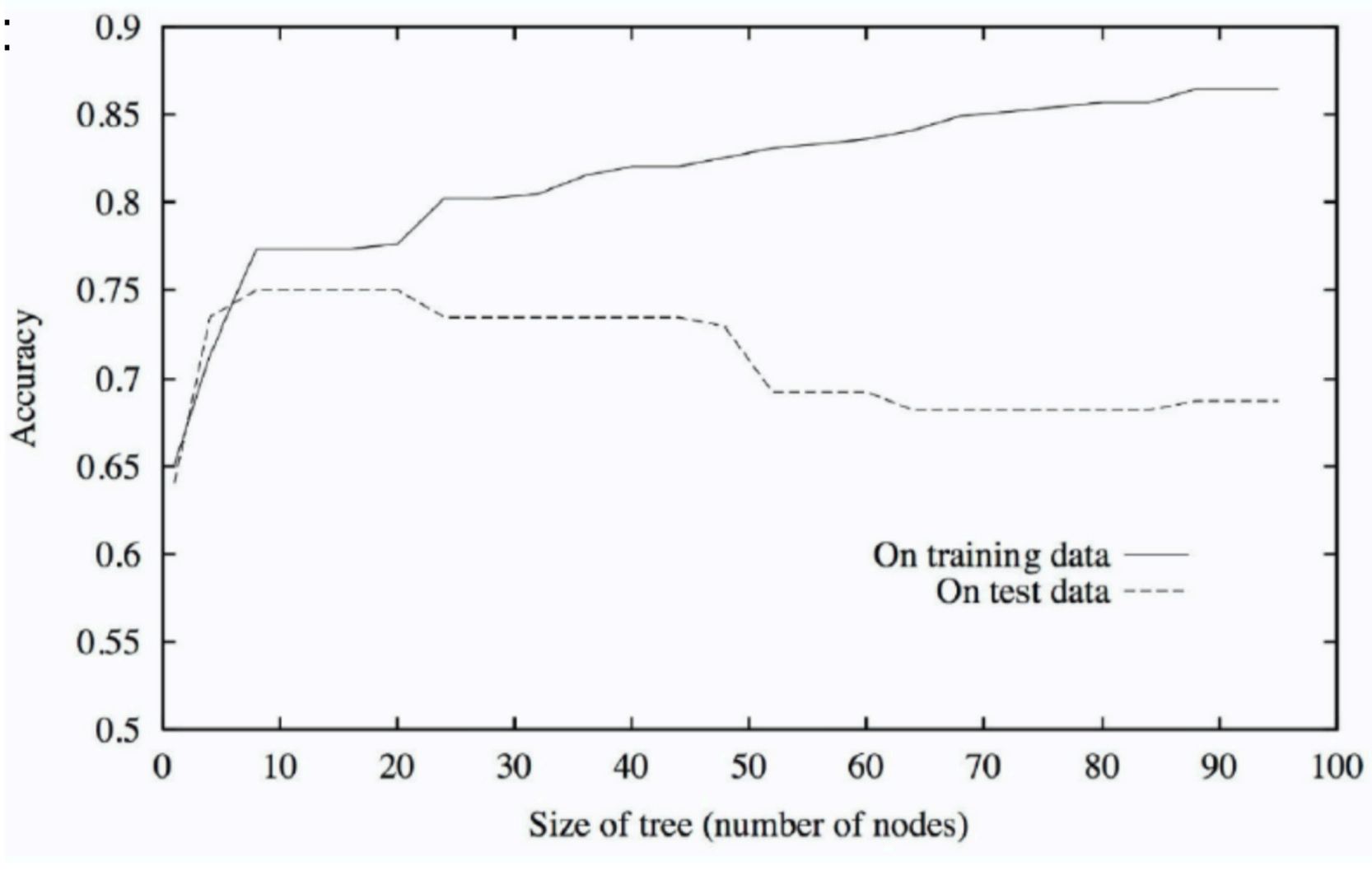
- In practice, one often regularizes the construction process to get small but highly informative trees



Adapted from slide by S. Fidler

# Decision Trees: issues, challenges

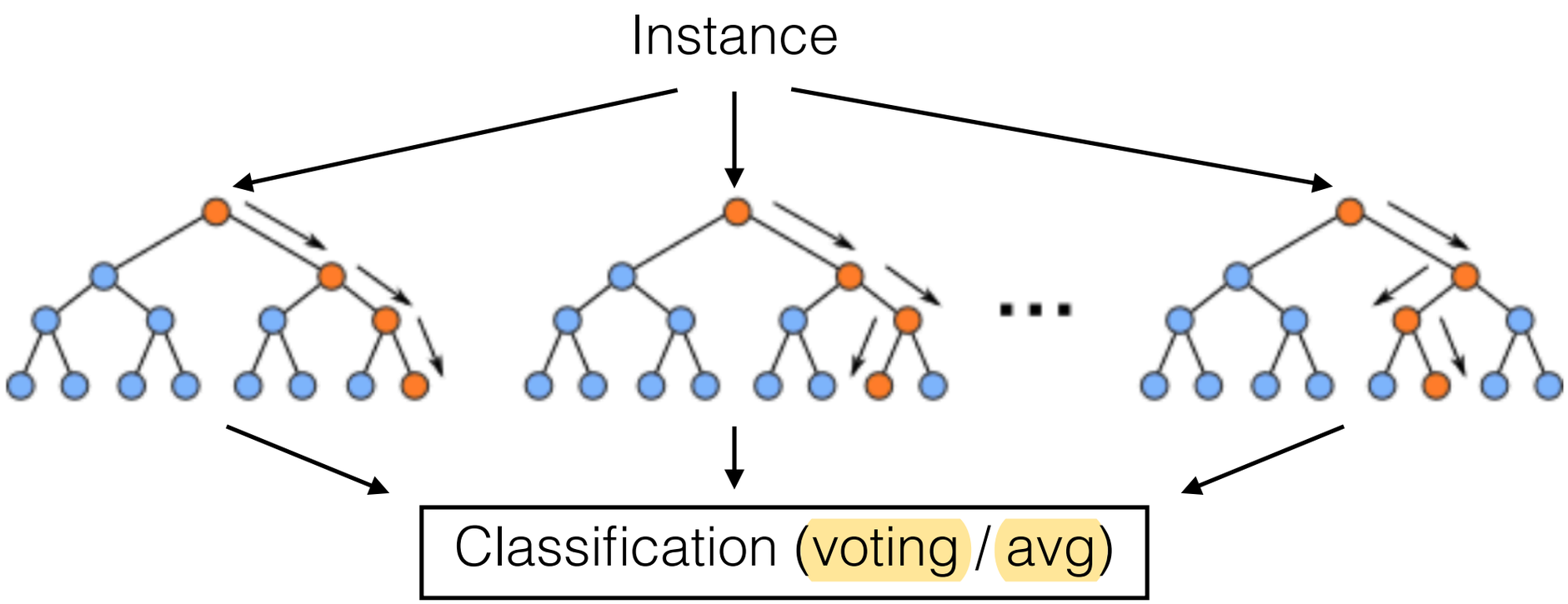
- Towards preventing overfitting (through pruning)
  - E.g.:




  - Note: in `sklearn` you can set a threshold on the number of examples in the leaf nodes or on the depth of the tree

# Random Forests

- The random forest is a classification algorithm consisting of many decisions trees
  - It uses bagging and feature randomness to combine trees
  - This way it reduces the variance of the "original" classifier (decision tree) and then making an ensemble out of it



# Contact

- **Office:** Torre Archimede 6CD, room 622
- **Office hours** (ricevimento)**:** Friday 11:00-13:00


✉ lamberto.ballan@unipd.it

🏠 http://www.lambertoballan.net

🏠 http://vimp.math.unipd.it

@ twitter.com/lambertoballan