# COMP 472: Artificial Intelligence
# Machine Learning part #2
# Decision Trees video #4

- Russell & Norvig: Sections 19.3

# Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees 

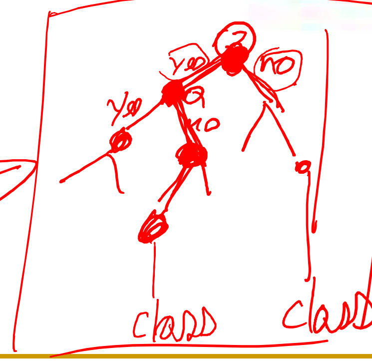
4. ( Evaluation
5. Unsupervised Learning )
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

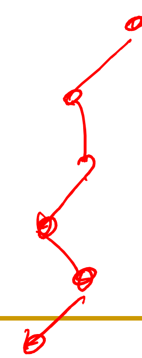
# Guess Who?


P1

P2

DT

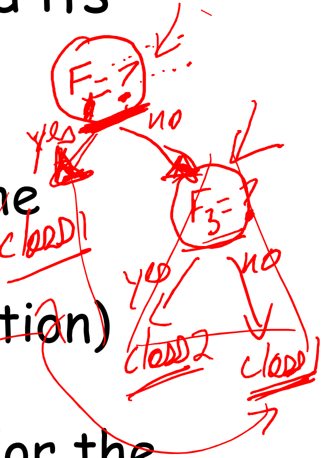


# Decision Trees

Ross Quinlan

- Simplest, but most successful form of learning algorithm
- Very well-know algorithm is ID3 (Quinlan, 1987) and its successor C4.5 5-48 ...




1. Rank features based on how good they are to indicate the result
2. Put the most discriminating feature as a node (as a question) of a tree
3. Split the training examples so that those with different values for the chosen feature are in a different set
4. Repeat the same process with the nest most discriminating feature

http://www.rulequest.com/Personal/

# Example 1

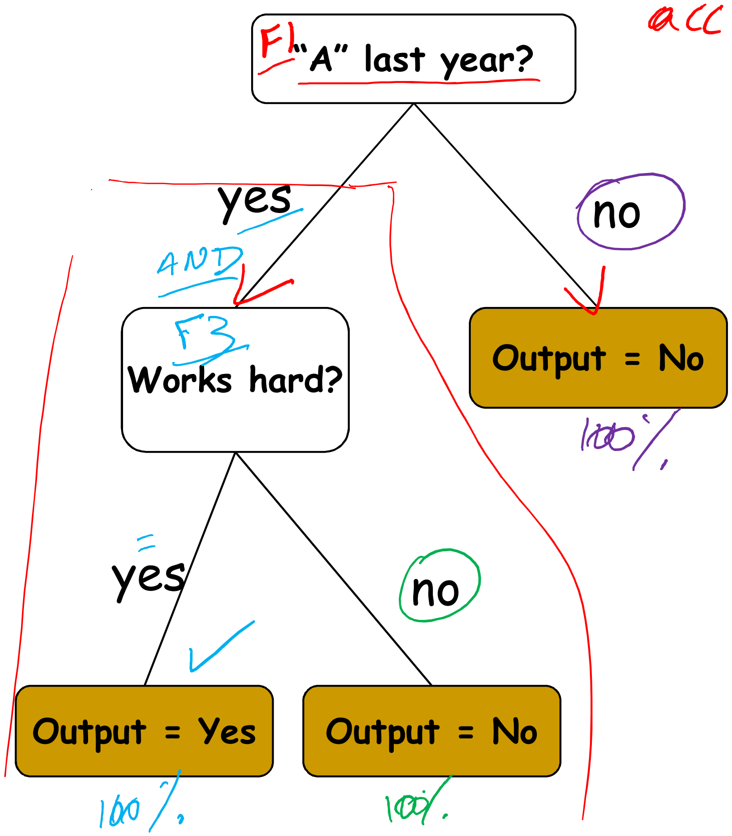
Info on last year's students to determine if a student will get an 'A' this year

| | Features (X) | | | | Output f(X) class |
|---|---|---|---|---|---|
| **Student** | **'A' last year?** F1 | **Black hair?** F2 | **Works hard?** F3 | **Drinks?** F4 | **'A' this year?** |
| X1: Richard | Yes | Yes | No | Yes | **No** |
| X2: Alan | Yes | Yes | Yes | No | **Yes** |
| X3: Alison | No | No | Yes | No | **No** |
| X4: Jeff | No | Yes | No | Yes | **No** |
| X5: Gail | Yes | No | Yes | Yes | **Yes** |
| X6: Simon | No | Yes | Yes | Yes | **No** |

# Example 1

A random decision tree that fits the dataset with 100% accuracy

same data set as previous slide




| | Features | | | | Output f(X) |
|---|---|---|---|---|---|
| **Student** | **'A' last year? F1** | **Black hair?** | **Works hard? F3** | **Drinks ?** | **'A' this year?** |
| Richard | Yes | Yes | No | Yes | No |
| Alan | Yes | Yes | Yes | No | Yes |
| Alison | No | No | Yes | No | No |
| Jeff | No | Yes | No | Yes | No |
| Gail | Yes | No | Yes | Yes | Yes |
| Simon | No | Yes | Yes | Yes | No |

- fits the training set with 100% accuracy
- but the features are chosen at random so there might be a shorter DT // better

# Example 2: The Restaurant

- Goal: learn whether one should wait for a table
- Attributes

10 features

1. Alternate: another suitable restaurant nearby
2. Bar: comfortable bar for waiting
3. Fri/Sat: true on Fridays and Saturdays
4. Hungry: whether one is hungry
5. Patrons: how many people are present (none, some, full)
6. Price: price range ($, $$, $$$)
7. Raining: raining outside
8. Reservation: reservation made
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated wait by host (0-10 mins, 10-30, 30-60, >60)

# Example 2: The Restaurant

- Training data:

10 features.

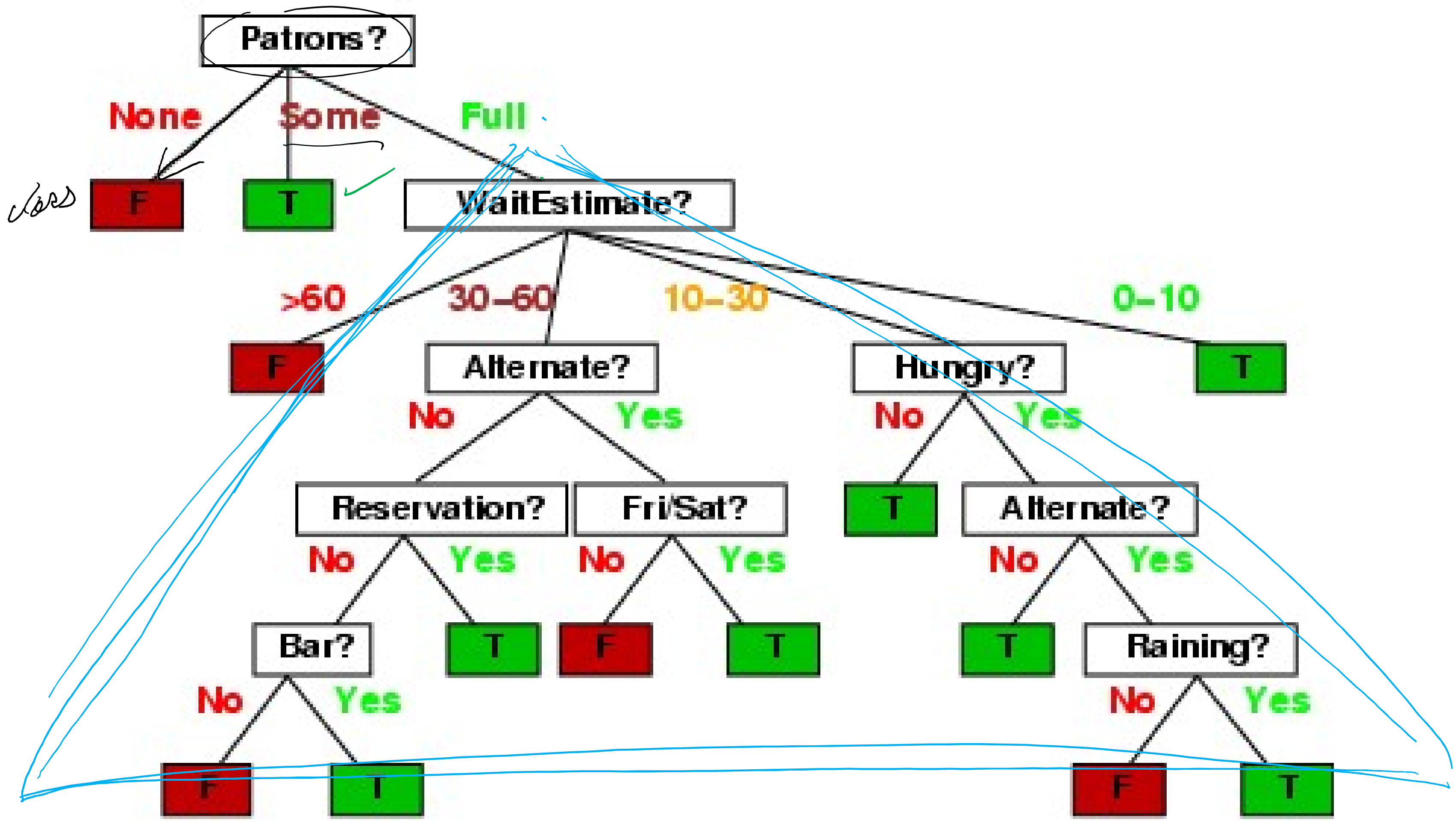
f(x) class

12 instances

| 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 |

source: Norvig (2003)

# A First Decision Tree




- But is it the best decision tree we can build?

source: Norvig (2003)

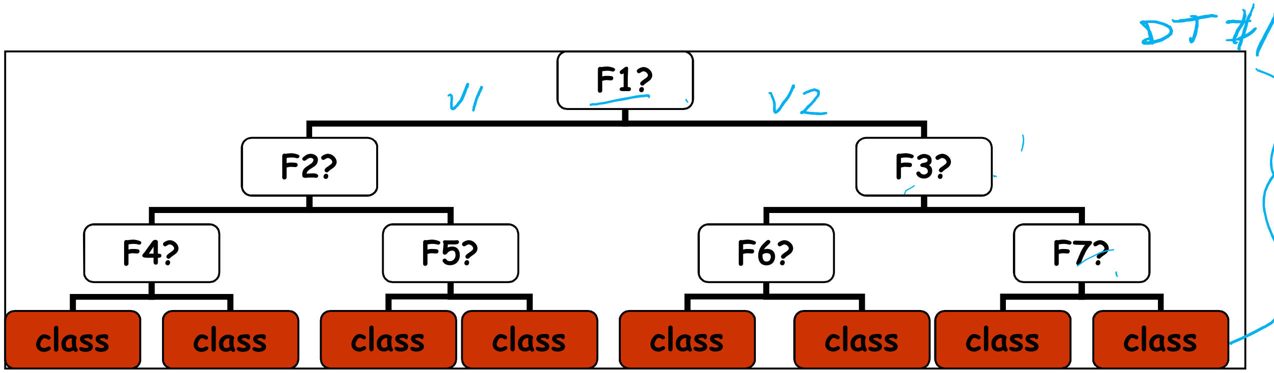
# Ockham's Razor

*It is vain to do more than can be done with less… Entities should not be multiplied beyond necessity. [Ockham, 1324]*

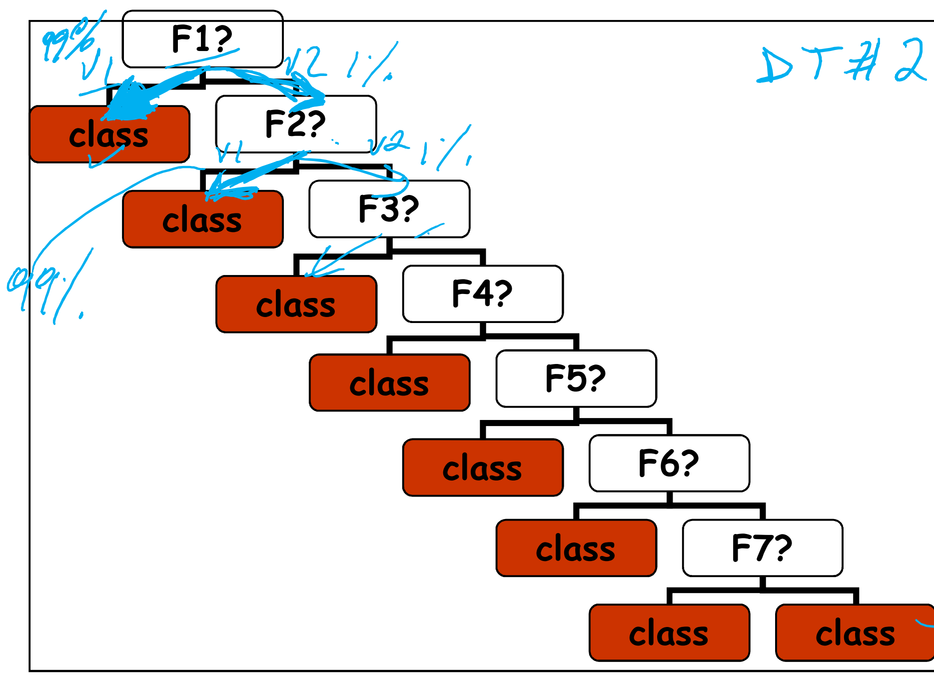
- In other words… always favor the simplest answer that correctly fits the training data
- i.e. the smallest tree on average
- This type of assumption is called inductive bias
  - inductive bias = making a choice beyond what the training instances contain

https://www.britannica.com/topic/Occams-razor

# Which Tree is Best?



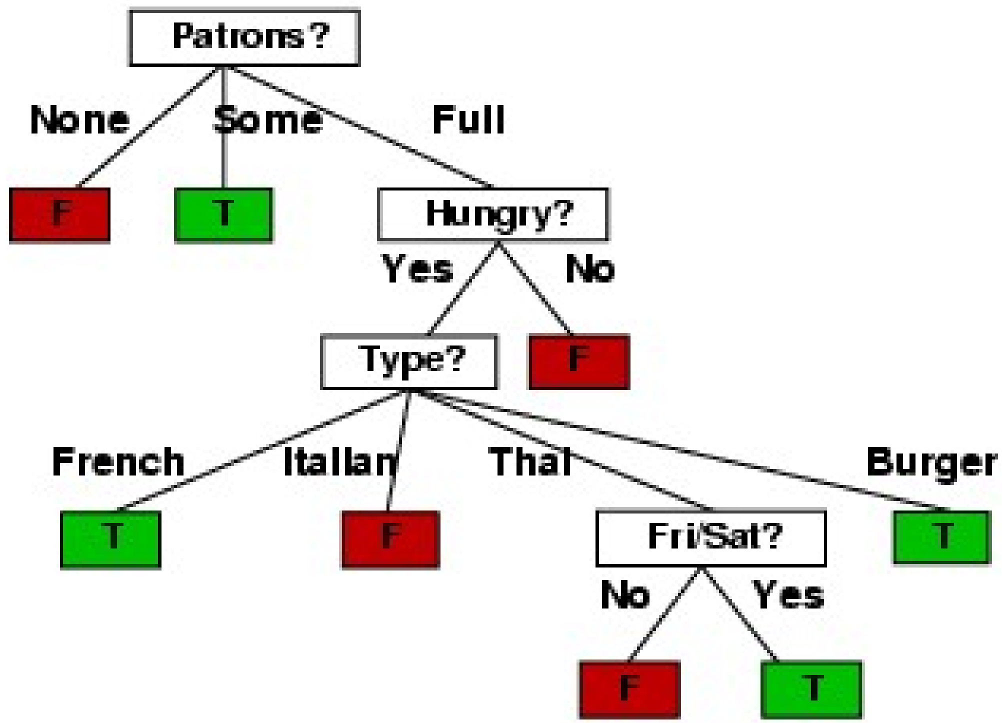

always asks 3 questions




may be the shortest tree on average depending on the probability of each branch

# A Better Decision Tree

- 4 tests instead of 9
- 11 branches instead of 21




source: Norvig (2003)

# Choosing the Next Feature

- The key problem is choosing which feature to split a given set of examples
- Different measures have been proposed
- ID3 uses Maximum Information-Gain
  - i.e. we choose the feature that has the largest information gain
  - we expect this feature to result in the smallest tree on average
  - based on information theory

# Essential Information Theory

- Developed by Claude Shannon in the 40s
- Shannon developed the notion of entropy (aka information content) of a random variable (RV)

- Entropy measures how "predictable" a RV is
  - If you already have a good idea about the answer (e.g. 90/10 split)
    → low entropy // sure thing
  - If you have no idea about the answer (e.g. 50/50 split)
    → high entropy // total chaos

# Entropy

- Let X be a discrete RV with *i* possible outcomes $x_i$
- Entropy (or information content) of X

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

base 2 always

$x_i$: 'e'

$x_n$: n

z

x

hello

- measures:
  - the *amount of information* in a RV
  - *average uncertainty* of a RV
  - *average length of a message* needed to transmit an outcome $x_i$ of that RV when encoded optimally over a binary channel
- measured in bits

# Entropy of a Coin Toss

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

Entropy (or information content)

prob $x_1$ = "head"    prob $x_2$ = "tail"

$$H(\text{fair coin toss}) = -\sum_{x_i \in X} p(x_i)\log_2 p(x_i) = H\left(\frac{1}{2}, \frac{1}{2}\right)$$

$$= -\left(\underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{head}} + \underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{tail}}\right) = 1\text{ bit}$$

entropy of a fair coin toss (the RV) with 2 possible outcomes, each with a probability of 1/2

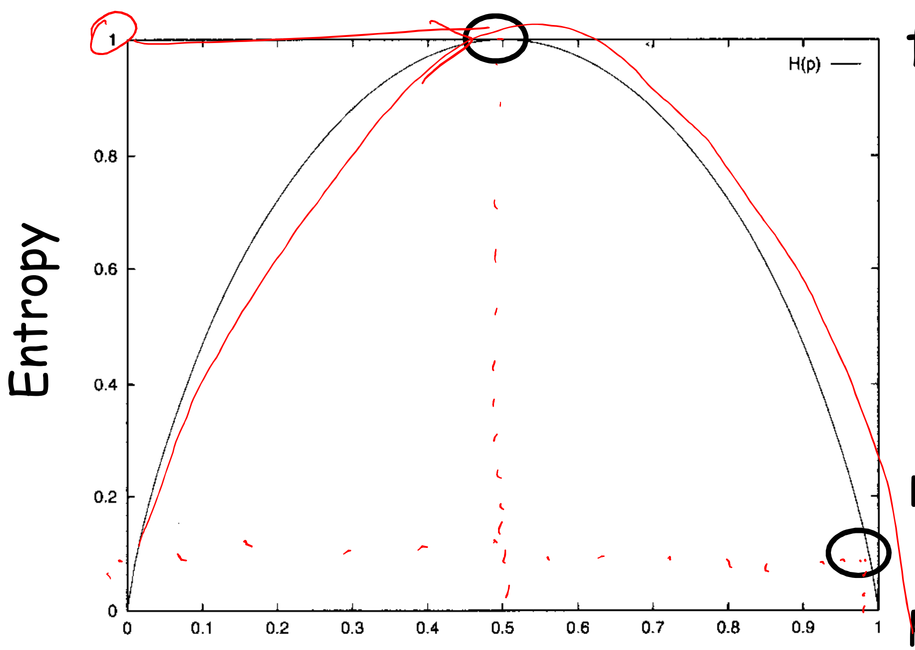
a RV with only 2 outcomes $x_1$ and $x_2$ will have $1 \geq H(X) \geq 0$

// sure thing

// total chaos

# Example: The Coin Toss

- Fair coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{head}} + \underbrace{\frac{1}{2}\log_2\frac{1}{2}}_{\text{tail}}\right) = 1\text{ bit}$
- Rigged coin: $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = -\left(\underbrace{\frac{99}{100}\log_2\frac{99}{100}}_{\text{head}} + \frac{1}{100}\log_2\frac{1}{100}\right) = 0.08\text{ bits}$




fair coin -> high entropy

rigged coin -> low entropy

P(head)

# So what?

for each attribute/feature I will compute how much knowing its values will reduce the entropy of the class RV

- Training data:

10 features

| 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 |

$x_1$

$x_2$

entropy(Target Wait) = 1

source: Norvig (2003)

# Information Gain

- information gain
  - measure the entropy reduction of a RV, once a piece of information is known
  - used to measure the "discriminating power" of an attribute A given a data set S

  - Let Values(A) = the set of values that attribute A can take
  - Let $S_v$ = the set of examples in the data set which have value v for attribute A (for each value v from Values(A) )

// see slide 15

information gain (or entropy reduction)

$$\text{gain}(S, A) = H(S) - H(S \mid A)$$

$$= H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \times H(S_v)$$

# Some Intuition

$H(\text{output}) = 1$ // total chaos

3 features

| Size | Color | Shape | Output |
|---|---|---|---|
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |

- *Size* is the least discriminating attribute (i.e. smallest information gain)
- *Shape* and *color* are the most discriminating attributes (i.e. highest information gain)

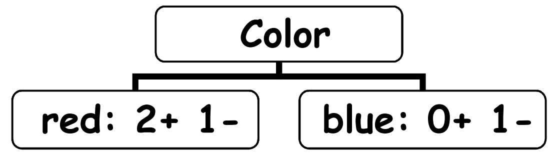
shape, color } more discriminating than size

# A Small Example (1)

let's try F2 = color

| Size | Color (F2) | Shape | Output |
| --- | --- | --- | --- |
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |

Values(Color) = {red,blue}




$$gain(S, Color) = H(S) - \sum_{v \in values(Color)} \frac{|S_v|}{|S|} \times H(S_v)$$

entropy of output

$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$ // total chaos, not knowing anything

for each v of Values(Color)

- $H(S|Color = red) = H\left(\frac{2}{3}, \frac{1}{3}\right) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) = 0.918$
- $H(S|Color = blue) = H(1,0) = -\left(\frac{1}{1}\log_2\frac{1}{1}\right) = 0$ // sure thing

$H(S|Color) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = \boxed{0.6885}$ knowing the color

$gain(Color) = H(S) - H(S|Color) = 1 - 0.6885 = 0.3115$

Note: by definition,

- $\log_2 0 = -\infty$
- $0\log_2 0$ is 0

1 time out of 4 we have a blue

3 times out of 4 we have a red

# A Small Example (2)

F3

| Size | Color | Shape | Output |
|---|---|---|---|
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |

Shape

circle: 2+ 1-

square: 0+ 1-

$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

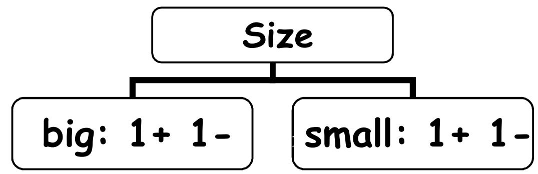
$$H(S\,|\,Shape) = \frac{3}{4}(0.918) + \frac{1}{4}(0) = 0.6885$$

$$gain(Shape) = H(S) - H(S\,|\,Shape) = 1 - 0.6885 = 0.3115$$

# A Small Example (3)

F1

| Size | Color | Shape | Output |
|---|---|---|---|
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |




$$H(S) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1$$

H(Output) without knowing any feature

$$H(S|Size) = \frac{2}{4}(1) + \frac{2}{4}(1)$$

$$gain(Size) = H(S) - H(S|Size) = 1 - 1 = 0$$

H(output/Size) knowing the size

# A Small Example (4)

| Size | Color | Shape | Output |
|---|---|---|---|
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |

$$gain(Shape) = 0.3115$$

$$gain(Color) = 0.3115$$

$$gain(Size) = 0$$

- So first separate according to either *color* or *shape* (root of the tree)

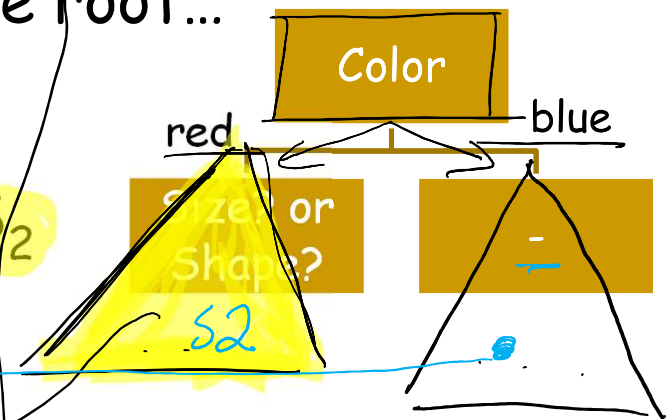
# A Small Example (4)

- Let's assume we pick *Color* for the root…

all instances in the data set

| Size | Color | Shape | Output |
|---|---|---|---|
| Big | Red | Circle | + |
| Small | Red | Circle | + |
| Small | Red | Square | - |
| Big | Blue | Circle | - |

S2 (rows 1–3), $S_2$




only use the instances with color = red

$$H(S_2) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)$$

only use the instances with color = blue

for each v of Values(Size)

$$H(S_2 \mid Size = big) = H\left(\frac{1}{1}, \frac{0}{1}\right) = 0$$

$$H(S_2 \mid Size = small) = H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$$

$$H(S_2 \mid Size) = \frac{1}{3}(0) + \frac{2}{3}(1)$$

$$gain(Size) = H(S_2) - H(S_2 \mid Size)$$

for each v of Values(Shape)

$$H(S_2 \mid Shape = circle) = H\left(\frac{2}{2}, \frac{0}{2}\right) = 0$$

$$H(S_2 \mid Shape = square) = H\left(\frac{0}{1}, \frac{1}{1}\right) = 0$$

$H(S_2 \mid Shape)$

$$gain(Shape) = H(S_2) - H(S_2 \mid Shape)$$

# Back to the Restaurant

pick max info gain

- Training data:

| 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 |

source: Norvig (2003)

# The Restaurant Example

∀ features

$\text{gain}(\text{alt}) = \ldots$ $\quad$ $\text{gain}(\text{bar}) = \ldots$ $\quad$ $\text{gain}(\text{fri}) = \ldots$ $\quad$ $\text{gain}(\text{hun}) = \ldots$

$$\text{gain}(\text{pat}) = 1 - \left( \frac{2}{12} \times H\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} \times H\left(\frac{0}{4}, \frac{4}{4}\right) + \frac{6}{12} \times H\left(\frac{2}{6}, \frac{4}{6}\right) \right)$$

$$= 1 - \left( \frac{2}{12} \times -\left( \frac{0}{2} \log_2 \frac{0}{2} + \frac{2}{2} \log_2 \frac{2}{2} \right) + \frac{4}{12} \times -\left( \frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4} \right) + \ldots \right) \approx 0.541 \text{bits}$$

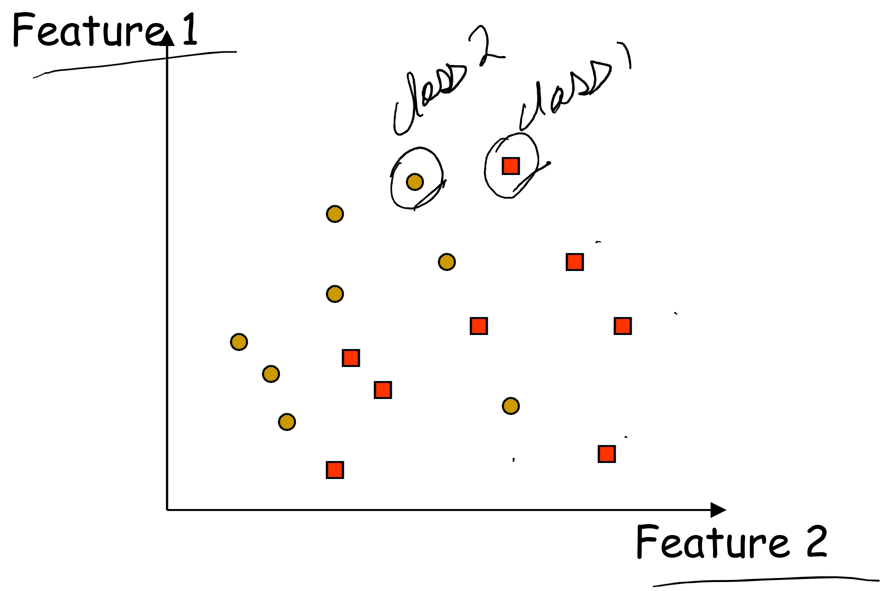
$\text{gain}(\text{price}) = \ldots$ $\quad$ $\text{gain}(\text{rain}) = \ldots$ $\quad$ $\text{gain}(\text{res}) = \ldots$

$$\text{gain}(\text{type}) = 1 - \left( \frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} \times H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} \times H\left(\frac{2}{4}, \frac{2}{4}\right) \right) = 0 \text{ bits}$$

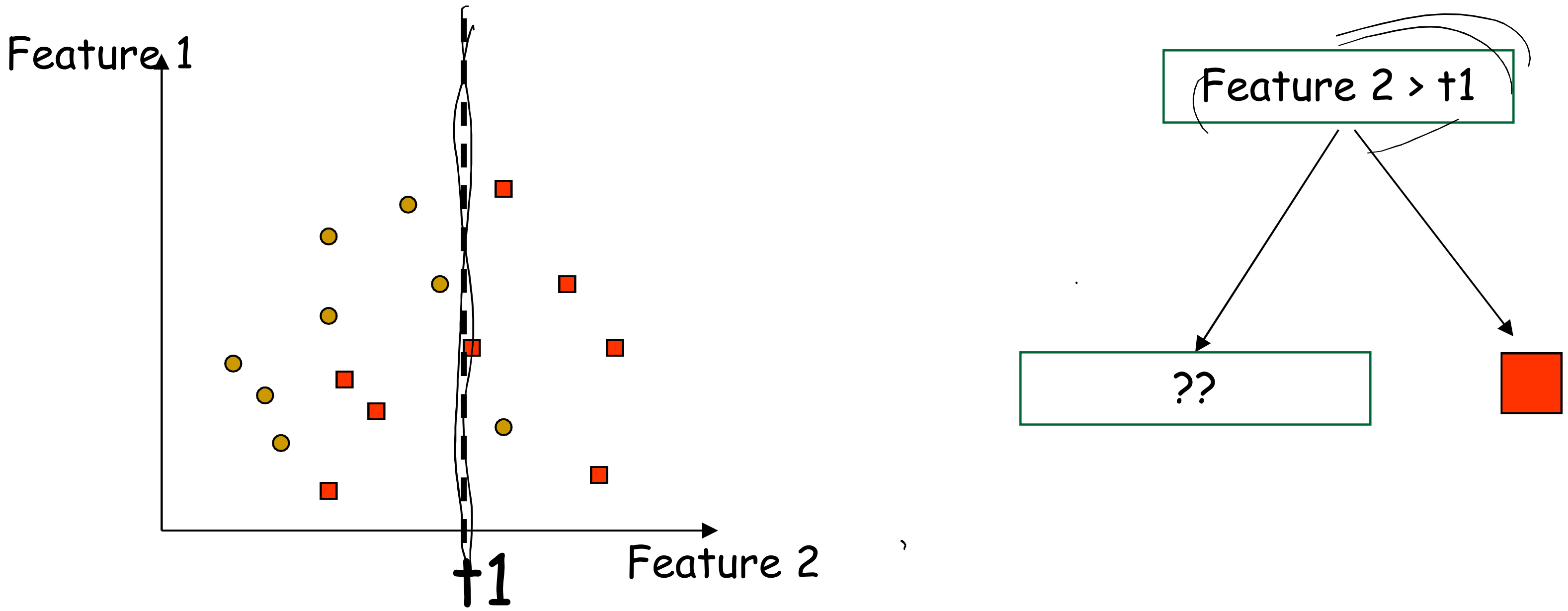
$\text{gain}(\text{est}) = \ldots$

- Attribute pat (Patron) has the highest gain, so root of the tree should be attribute *Patrons*
- do recursively for subtrees

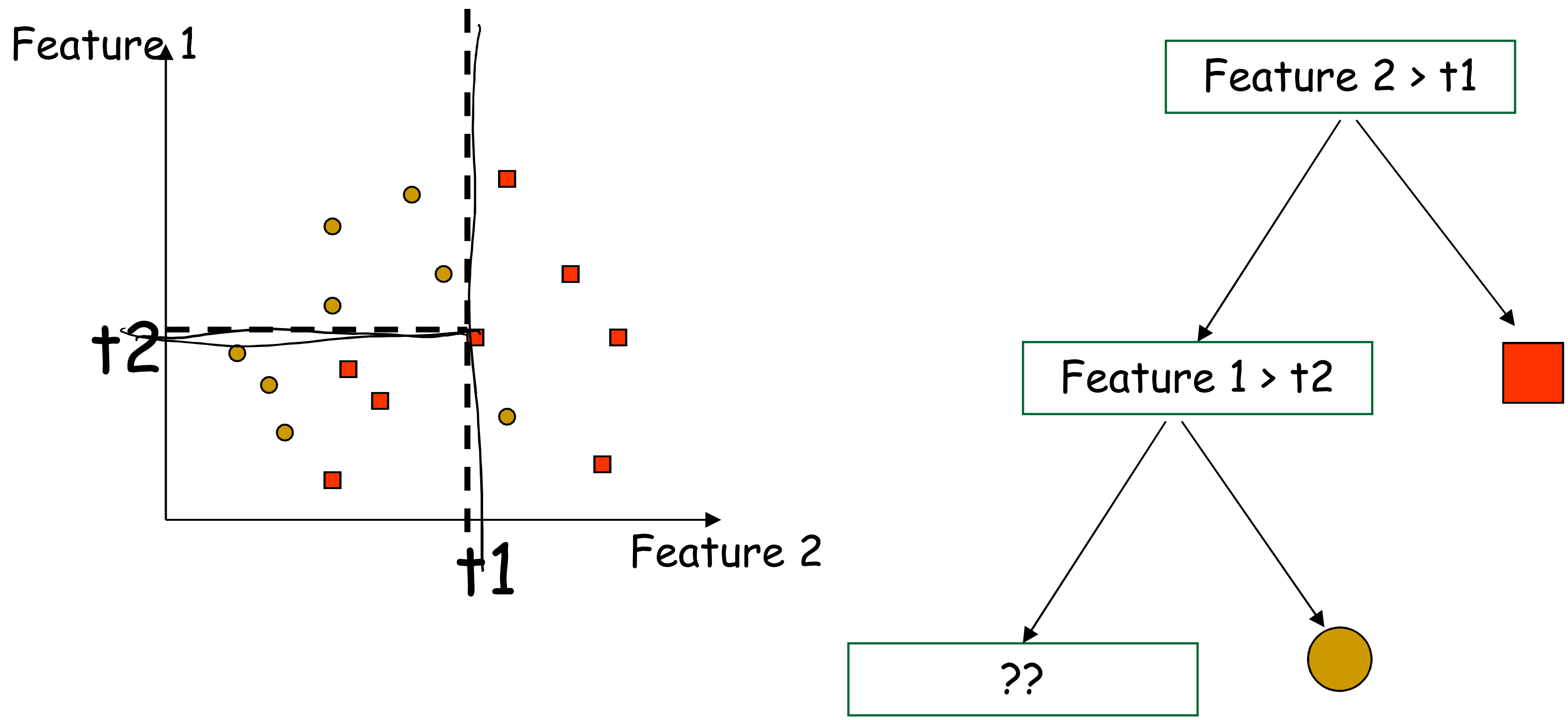
# Decision Boundaries of Decision Trees



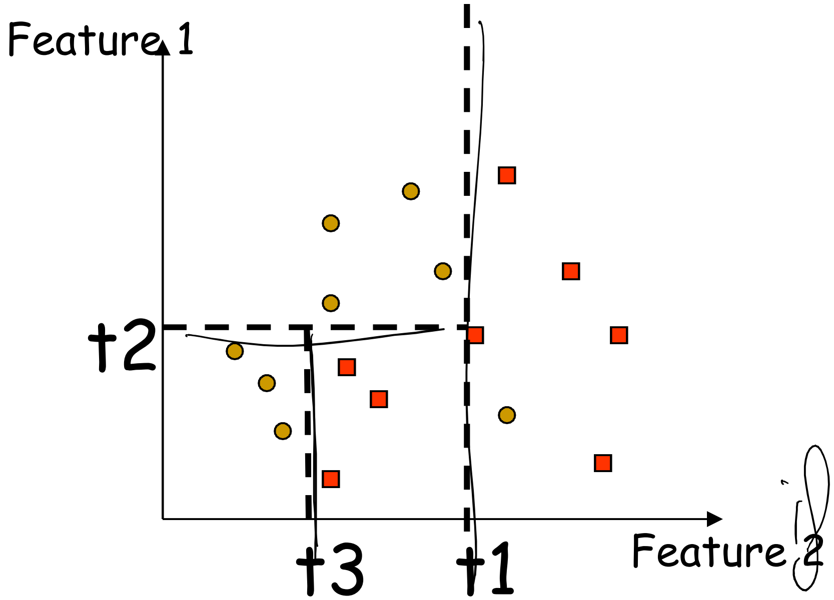
# Decision Boundaries of Decision Trees

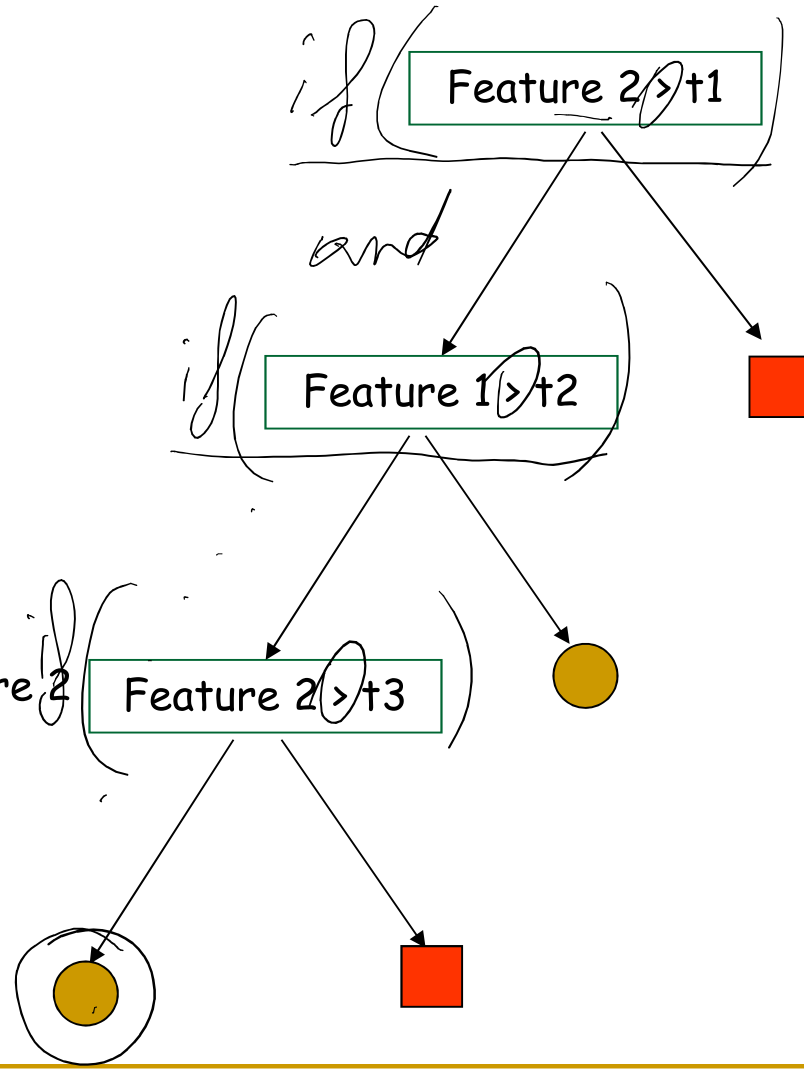


# Decision Boundaries of Decision Trees



# Decision Boundaries of Decision Trees



# Applications of Decision Trees

- One of the most widely used learning methods in practice
  - Fast
  - Simple
  - Traceable (<-- very important!)

# Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees

video #4

4. ( Evaluation
5. Unsupervised Learning )
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

# Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. ( Evaluation
5. Unsupervised Learning )
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks

supervised learning