Designing a Learning Agent

- What type of performance element?
- Which functional component to be learned?
- How that functional component is represented
- What type of feedback is available?

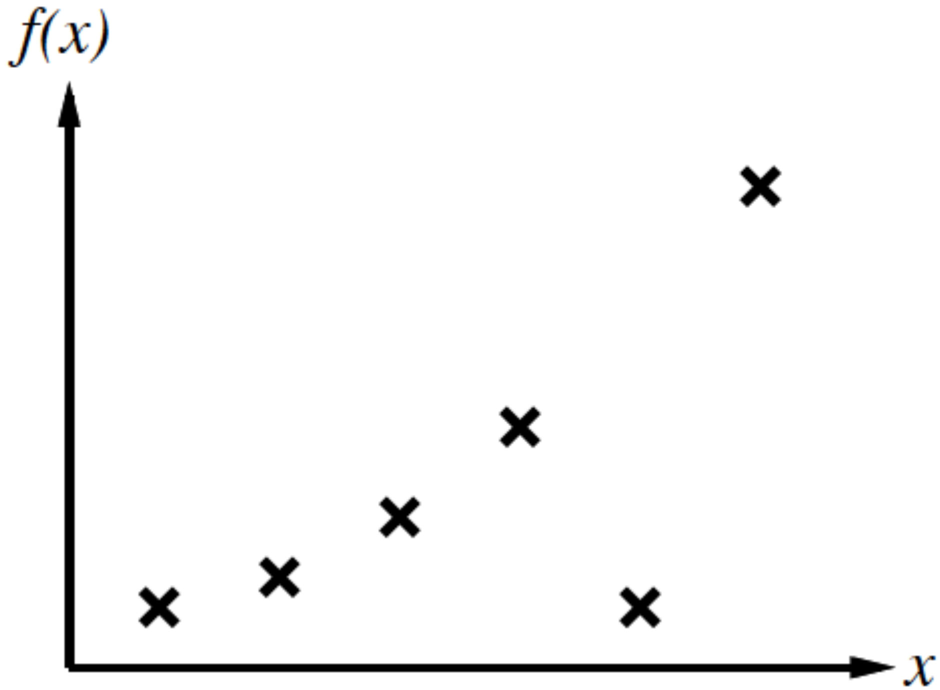
Performance Element	Component	Representation	Feedback
Alpha-beta search	Eval. fn	Weighted linear fn.	Win/loss
Logical agent	Transition model	Successor-state axioms	outcome
Utility-based agent	Transition model	Dynamic bayes net	outcome
Simple-reflex agent	Percept-action fn.	Neural network	Correct action

Types of Learning

- **Supervised learning**
  - Give correct answer for each instance
  - Learn a function from examples of inputs/outputs
- **Unsupervised learning**
  - No correct answers known
  - Can learn patterns in the input
  - Can’t learn what to do w/o feedback (don’t know whether states are desirable/undesirable)
  - But you can learn a probability distribution
- **Reinforcement learning**
  - Sometimes you get a reward, sometimes you get punished
  - Example: a waiter will learn to prefer certain behaviors because he gets bigger tips
  - Typically, trying to learn how the environment works

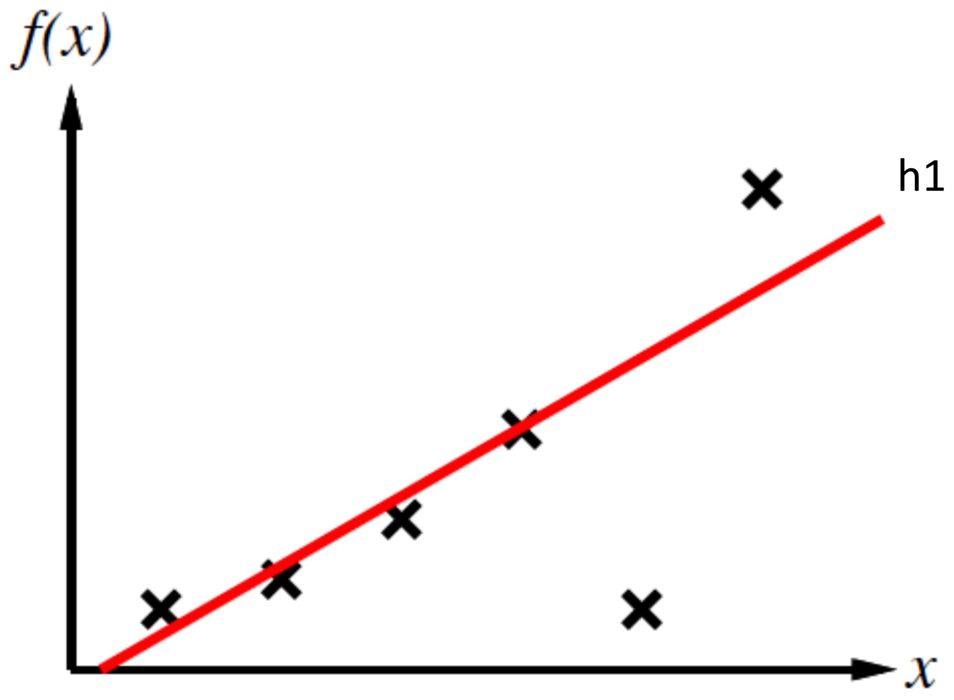
Induction

- Example: curve fitting



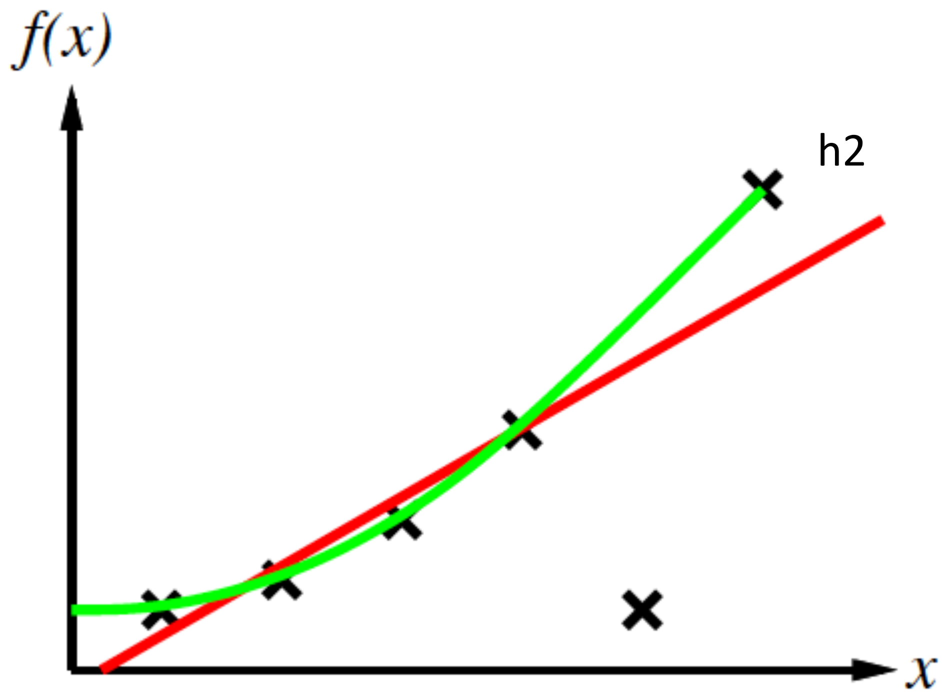
Induction

- Example: curve fitting



Induction

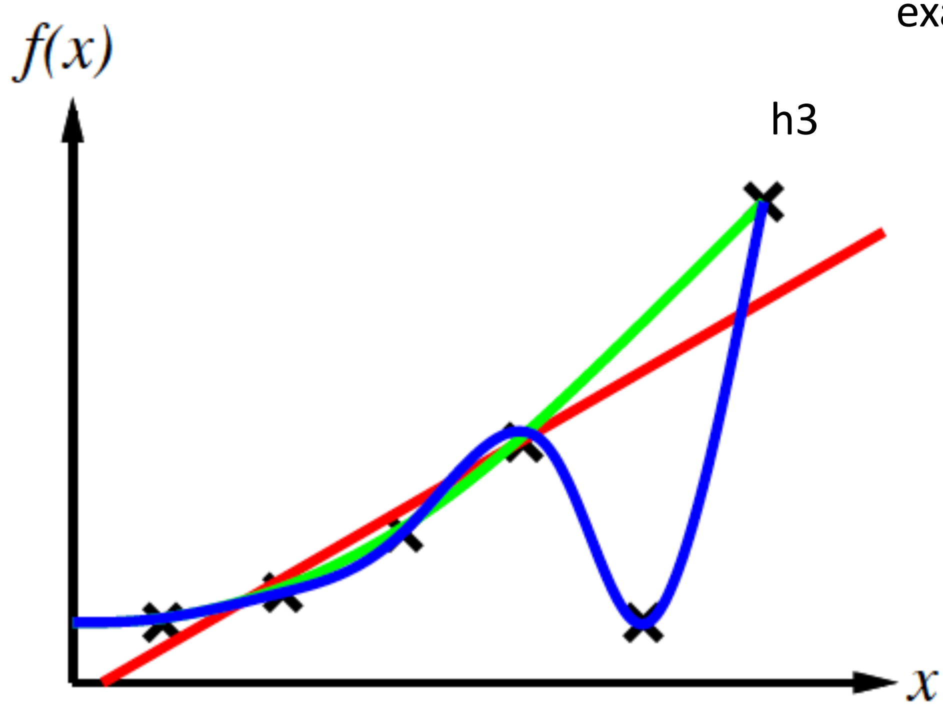
- Example: curve fitting



Induction

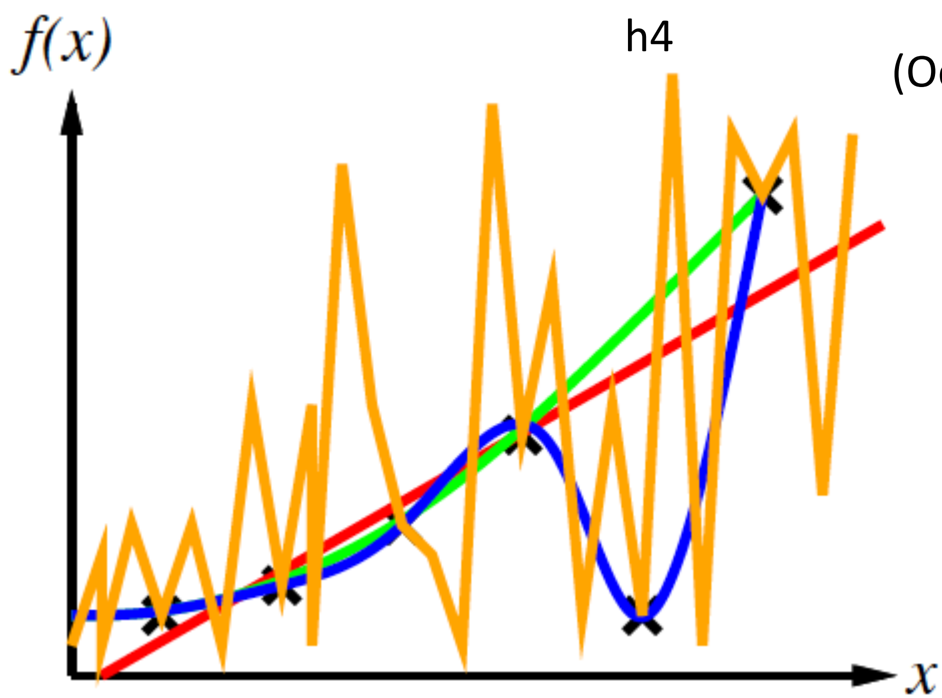
- Example: curve fitting

H is consistent if it agrees with all examples



Induction

- Example: curve fitting



Given multiple consistent hypotheses, pick the simplest one

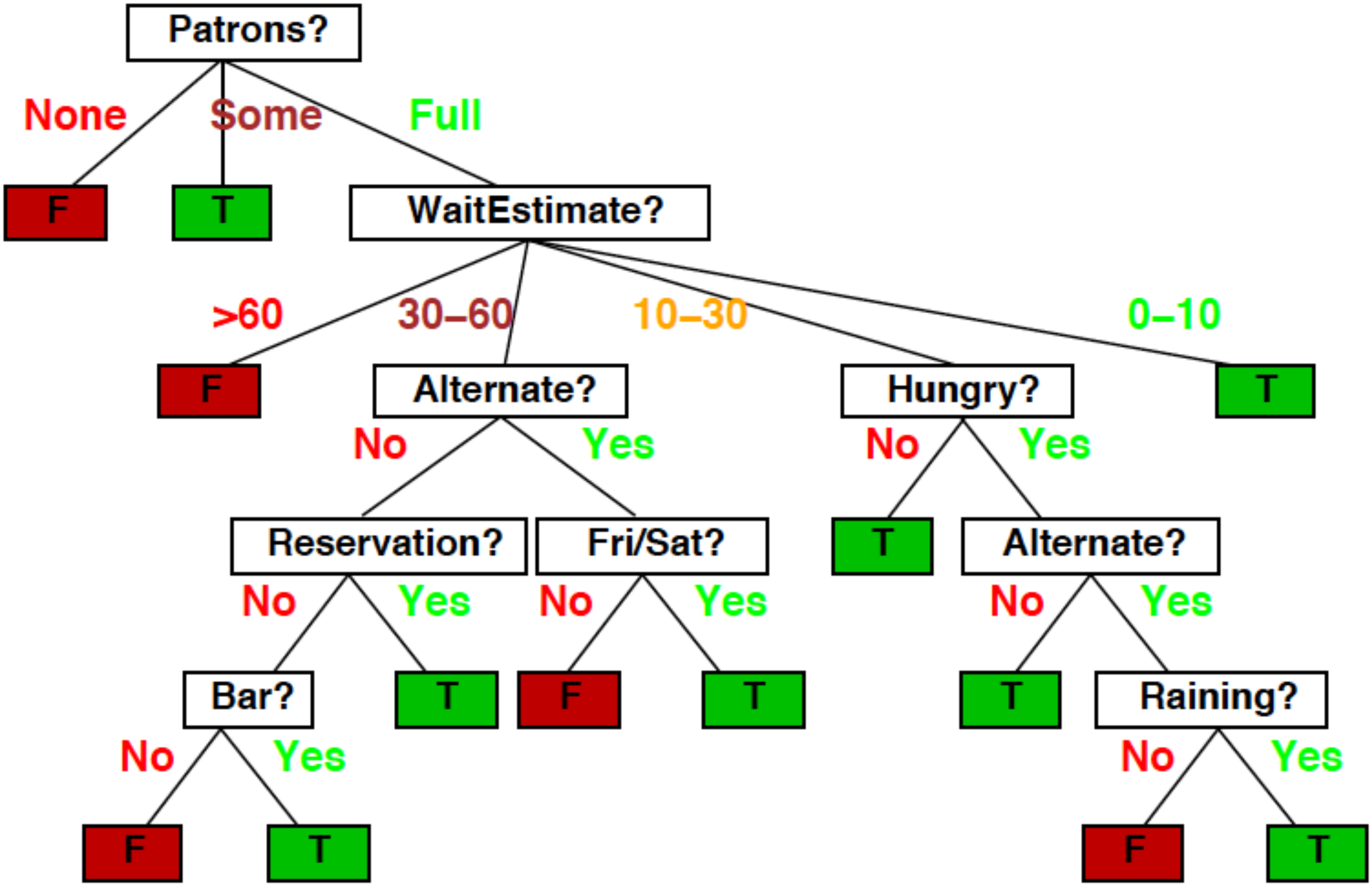
(Ockham's razor)

Learning Decision Trees

- A simple technique whereby the computer learns to make decisions that emulate human decision-making
- Can also be used to learn to classify
  - A decision can be thought of as a classification problem
- An object or situation is described as a set of attributes
  - Attributes can have discrete or continuous values
- Predict an outcome (decision or classification)
  - Can be discrete or continuous
  - We assume positive (true) or negative (false)

Eat at a restaurant?

- **Attributes:**
  - **Alternate**: suitable alternate restaurant nearby (y/n
  - **Bar**: A bar to wait in (y/n)
  - **Fri/Sat**: it’s a Friday or Saturday (y/n)
  - **Hungry**: y/n
  - **Price**: price range ($, $$, $$$)
  - **Raining**: y/n
  - **Reservation**: we made a reservation (y/n)
  - **Type**: french, italian, thai, burger
  - **WaitEstimate**: 0-10, 10-30, 30-60, >60
  - **Patrons**: none, some, full

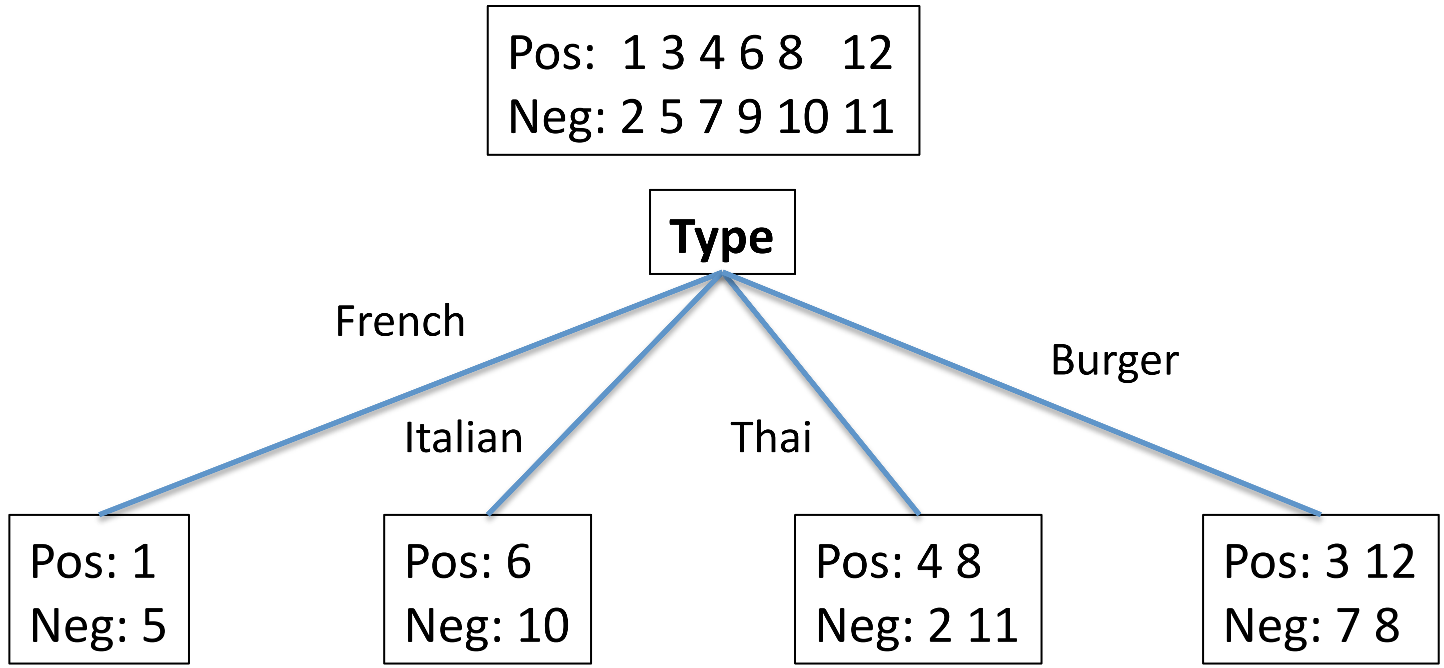


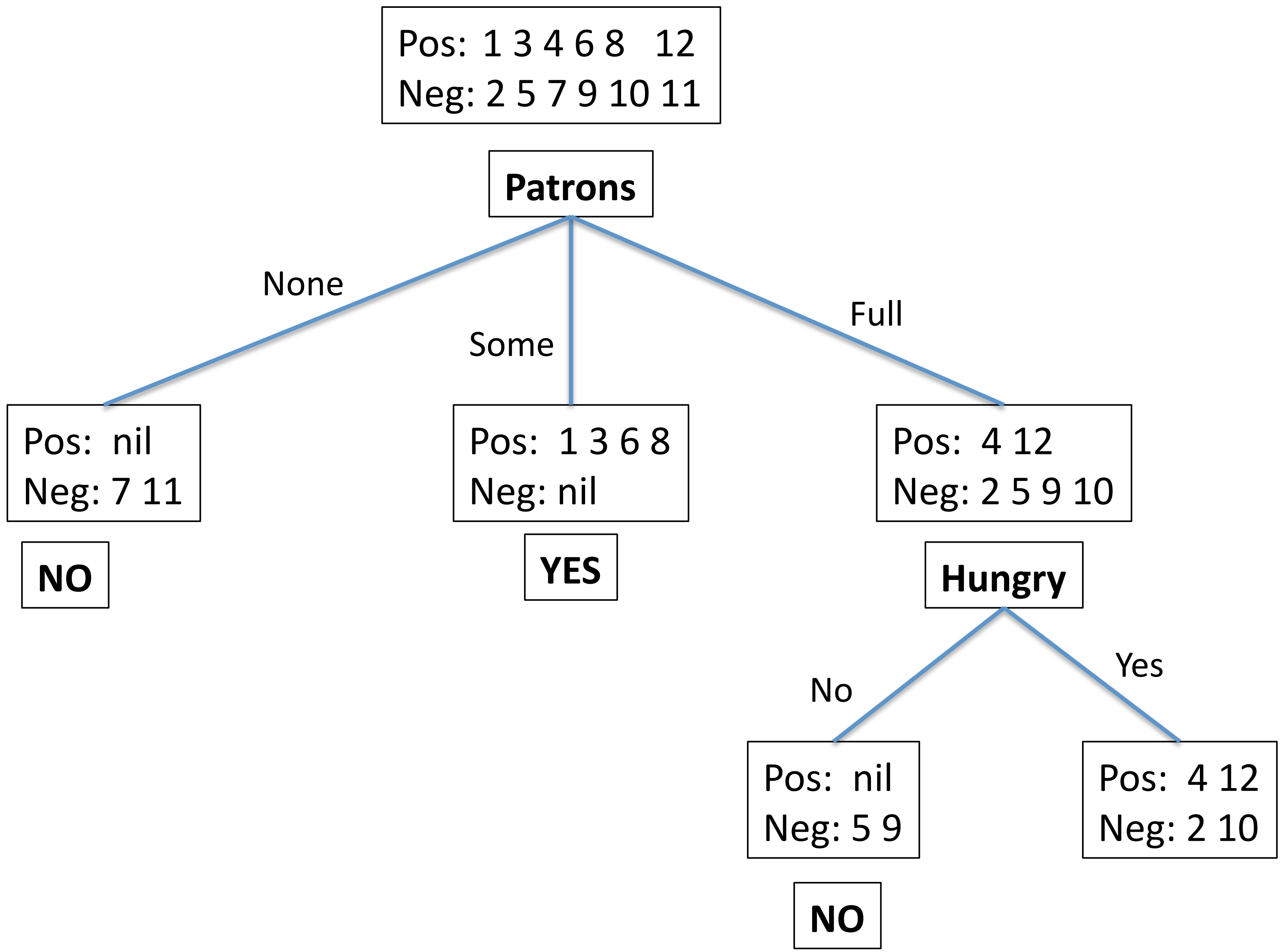
Patrons?
None
Some
Full
F
T
WaitEstimate?
>60
30–60
10–30
0–10
F
Alternate?
Hungry?
T
No
Yes
No
Yes
Reservation?
Fri/Sat?
T
Alternate?
No
Yes
No
Yes
No
Yes
Bar?
T
F
T
T
Raining?
No
Yes
No
Yes
F
T
F
T

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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*

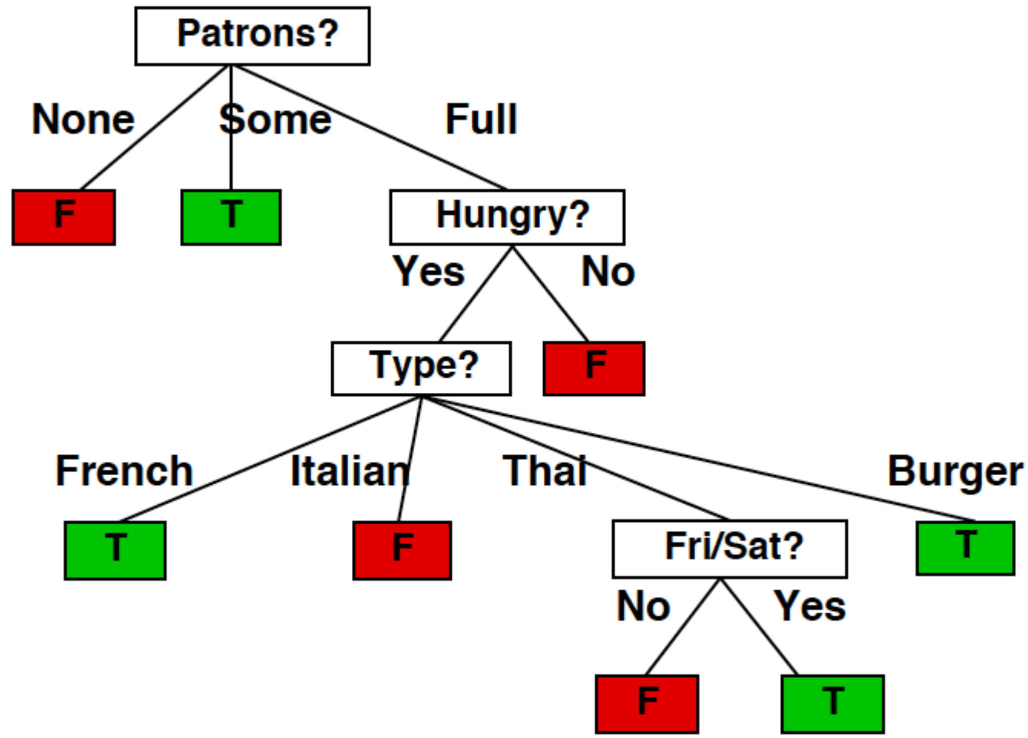
Supervised Learning

- Training set
- Test set


Pos: 1 3 4 6 8 12
Neg: 2 5 7 9 10 11
Type
French
Italian
Thai
Burger
Pos: 1
Neg: 5
Pos: 6
Neg: 10
Pos: 4 8
Neg: 2 11
Pos: 3 12
Neg: 7 8


Pos: 1 3 4 6 8 12
Neg: 2 5 7 9 10 11
Patrons
None
Some
Full
Pos: nil
Neg: 7 11
NO
Pos: 1 3 6 8
Neg: nil
YES
Pos: 4 12
Neg: 2 5 9 10
Hungry
No
Yes
Pos: nil
Neg: 5 9
NO
Pos: 4 12
Neg: 2 10

- Learned from the 12 examples
- Why doesn't it look like the previous tree?
  - Not enough examples
  - No reason to use rain or reservations
  - Hasn't seen all cases
- Learning is only as good as your training data



Which attribute to choose?

- The one that gives you the most information (aka the most diagnostic)
- Information theory
  - Answers the question: how much information does something contain?
  - Ask a question
  - Answer is information
  - Amount of information depends on how much you already knew
- Example: flipping a coin
  - If you don't know that coin flipping is random: 1 bit of information is gained
  - If you do know: 0 bits of information is gained

- If there are n possible answers, $v_1 \ldots v_n$ and v_i has probability $P(v_i)$ of being the right answer, then the amount of information is:

$$I\big(P(v_1),\ldots,P(v_n)\big) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- Example: coin toss

- For a training set:

  p = # of positive examples

  n = # of negative examples

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Probability of a positive example

Probability of a negative example

- For our restaurant behavior
  - p = n = 6
  - I() = 1
  - Would not be 1 if training set weren't 50/50 yes/no, but the point is to arrange attributes to increase information gain

Pos:	1 3 4 6 8 12
Neg:	2 5 7 9 10 11

Measuring attributes

- Information gain is a function of how much more information you need after applying an attribute
  - If I use attribute A next, how much more information will I need?
  - Use this to compare attributes

Instances of the attribute

Positive examples for this answer

Negative examples for this answer

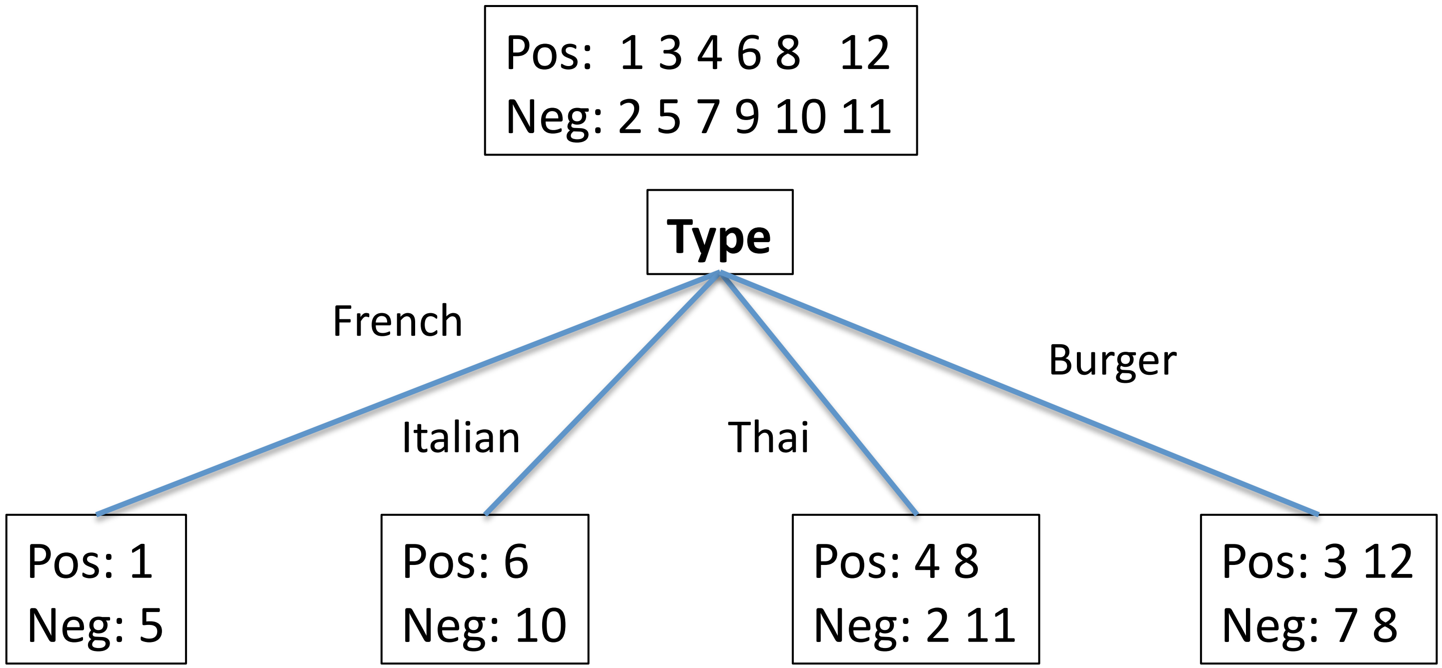
$$\text{Remainder}(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

attribute

Different answers

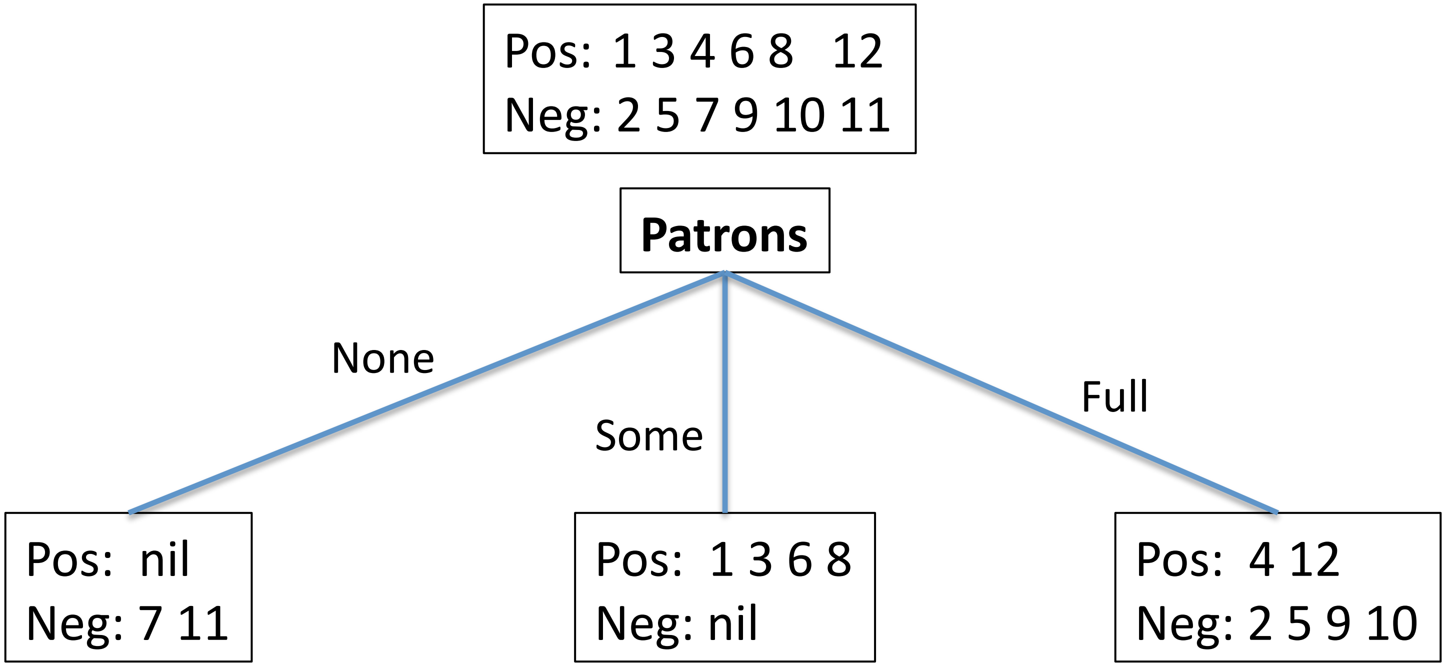
Total answers

Examples classified by A



$$\text{Remainder(type)} = \frac{2}{12} I\left(\frac{1}{2},\frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2},\frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4},\frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4},\frac{2}{4}\right) = 1 \text{ bit}$$

French Italian Thai Burger



$$\text{Remainder(patrons)} = \underbrace{\frac{2}{12} I\left(\frac{0}{2},\frac{2}{2}\right)}_{\text{none}} + \underbrace{\frac{4}{12} I\left(\frac{4}{4},\frac{0}{4}\right)}_{\text{some}} + \underbrace{\frac{6}{12} I\left(\frac{2}{6},\frac{4}{6}\right)}_{\text{full}} \approx 0.459 \text{ bit}$$

- Not done yet
- Need to measure information **gained** by an attribute

$$\text{Gain(A)} = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder(A)}$$

- Pick the biggest
- Example:
  - Gain(type) = I(½,½) – $\left(\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right)\right)$

    = 0 bits

  - Gain(patrons) = I(½,½) – $\left(\frac{2}{12} I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} I\left(\frac{4}{4}, \frac{0}{4}\right) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right)\right)$

    ≈ 0.541 bits

Pos: 1 3 4 6 8 12
Neg: 2 5 7 9 10 11

Patrons

Full

Pos: 4 12
Neg: 2 5 9 10

Hungry

No

Pos: nil
Neg: 5 9

Yes

Pos: 4 12
Neg: 2 10

Patrons=full, hungry=yes

Patrons=full, hungry=no

$$\text{gain(hungry)} = I\left(\frac{2}{6},\frac{4}{6}\right) - \left[\frac{2}{6} I\left(\frac{0}{2},\frac{2}{2}\right) + \frac{4}{6} I\left(\frac{2}{4},\frac{2}{4}\right)\right]$$

no

yes

= 0.9182958 – [0 + (4/6)(1)]

≈ 0.251 bits

Decision-tree-learning (examples, attributes, default)

IF examples is empty THEN RETURN default

ELSE IF all examples have same classification THEN RETURN classification

ELSE IF attributes is empty RETURN majority-value(examples)

ELSE

    best = choose(attributes, example) ← Where info gain happens

    tree = new decision tree with best as root

    m = majority-value(examples)

    FOREACH answer v_i of best DO

        examples_i = {elements of examples with best=v_i}

        subtree_i = decision-tree-learning(examples_i, atributes-{best}, m)

        add a branch to tree based on v_i and subtree_i

    RETURN tree

How many hypotheses?

- How many distinct trees?
  - N attributes

    = # of boolean functions

    = # of distinct truth tables with 2^n rows

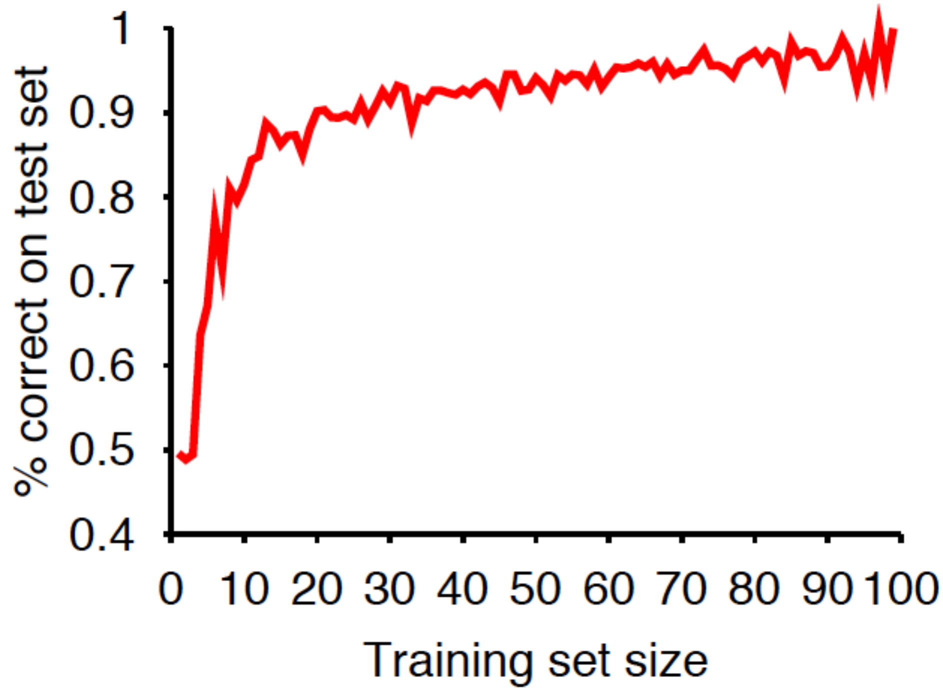
    = 2^2^n
  - With 6 attributes: > 18 quintillion possible trees

How do we assess?

- How do we know $h \approx f$?
- A learning algorithm is good if it produces hypotheses that do a good job of predicting decisions/classifications from unseen examples

1. Collect a large set of examples (with answers)
2. Divide into training set and test set
3. Use training set to produce hypothesis h
4. Apply h to test set (w/o answers)
   – Measure % examples that are correctly classified
5. Repeat 2-4 for different sizes of training sets, randomly selecting examples for training and test
   – Vary size of training set m
   – Vary which m examples are training

- Plot a learning curve
  – % correct on test set, as a function of training set size



- As training set grows, prediction quality should increase
  – Called a “happy graph”
  – There is a pattern in the data AND the algorithm is picking it up!

Noise

- Suppose 2 or more examples with same description (Same assignment of attributes) have different answers
- Examples: on two identical* situations, I do two different things
- You can't have a consistent hypothesis (it must contradict at least one example)
- Report majority classification or report probability

Overfitting

- Learn a hypothesis that is consistent using irrelevant attributes
  - Coincidental circumstances result in spurious distinctions among examples
  - Why does this happen?
    - You gave a bunch of attributes because you didn't know what would be important
    - If you knew which attributes were important, you might not have had to do learning in the first place
- Example: Day, month, or color of die in predicting a die roll
  - As long as no two examples are identical, we can find an exact hypothesis
  - Should be random 1-6, but if I roll once every day and each day results in a different number, the learning algorithm will conclude that day determines the roll
- Applies to all learning algorithms