

# CS 188: Artificial Intelligence

## Machine Learning I: Decision trees and linear regression



Instructor: Dan Klein and Stuart Russell

University of California, Berkeley

# Lectures on learning

---

- **Learning**: a process for improving the performance of an agent through experience (RL was one example)
  - Machine Learning I (today):
    - The general idea: generalization from experience
    - Supervised learning: classification with decision trees; linear regression
  - Machine Learning II: Naive Bayes and Bayesian learning
  - Deep Learning I and II: Perceptrons and neural networks
- (We cover just a small fraction of ML ideas; see also 189, 182, 180,...)

# Learning: Why?

---

- *The baby, assailed by eyes, ears, nose, skin, and entrails at once, feels it all as one great blooming, buzzing confusion ...*  
[William James, 1890]
- Learning is **essential** in unknown environments – when the agent designer lacks omniscience

# Learning: Why?

---

- *Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationer's. Rather little mechanism, and lots of blank sheets.*

[Alan Turing, 1950]

- Learning is *useful* as a system construction method, i.e., expose the system to reality rather than trying to write it down
  - Also, humans may be able to do something without knowing how!

# Learning: How?



Mr. & Mrs., Skinner view daughter  
Debbie in a Skinner Box

# Learning: How?



# Key questions when building a learning agent

---

- What is the ***agent design*** that will implement the desired performance?
- Improve the performance of ***what piece*** of the agent system and ***how is that piece represented?***
- ***What data*** are available relevant to that piece? (In particular, do we know the right answers?)
- ***What knowledge*** is already available?

# Examples

Agent design	Component	Representation	Feedback	Knowledge
MCTS search	Evaluation function	Linear polynomial	Win/loss	Rules of game
MDP agent	Transition model (observable envt)	Transition matrix	Action outcomes	Available actions, possible states
Utility-based patient monitor	Physiology/sensor model	Dynamic Bayesian network	Observation sequences	Human physiology; Sensor design
Chatbot	Context-based word predictor	Transformer (deep neural network)	Known next word	Tokenization, vector space

***Supervised learning***: correct answers for each training instance

***Reinforcement learning***: reward sequence, no correct answers

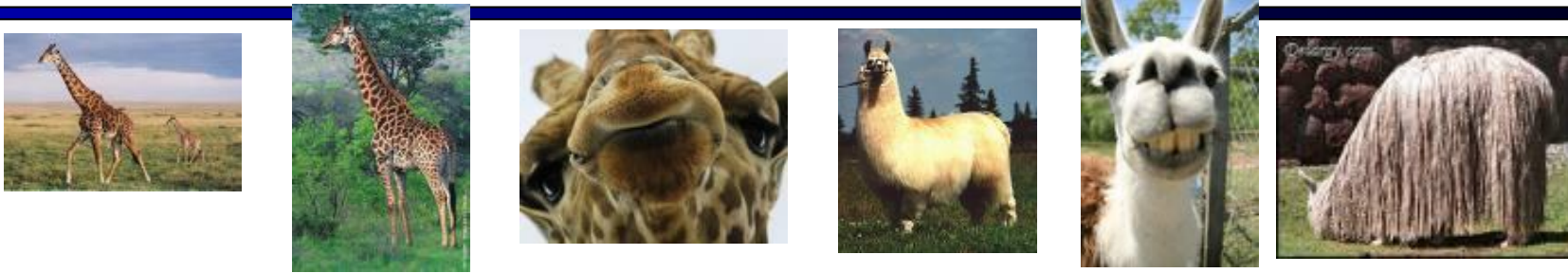
***Unsupervised learning***: “just make sense of the data”

# Supervised learning

- To learn an unknown **target function**  $f$
- Input: a **training set** of **labeled examples**  $(x_j, y_j)$  where  $y_j = f(x_j)$ 
  - E.g.,  $x_j$  is an image,  $f(x_j)$  is the label “giraffe”
  - E.g.,  $x_j$  is a seismic signal,  $f(x_j)$  is the label “explosion”
- Output: **hypothesis**  $h$  that is “close” to  $f$ , i.e., predicts well on unseen examples (“**test set**”)
- Many possible hypothesis families for  $h$ 
  - Linear models, logistic regression, neural networks, decision trees, examples (nearest-neighbor), grammars, Python programs, etc etc
- **Classification** = learning  $f$  with discrete output value
- **Regression** = learning  $f$  with real-valued output value

# Classification example: Object recognition

x



f(x)

giraffe

giraffe

giraffe

llama

llama

llama

x=



f(x)=?

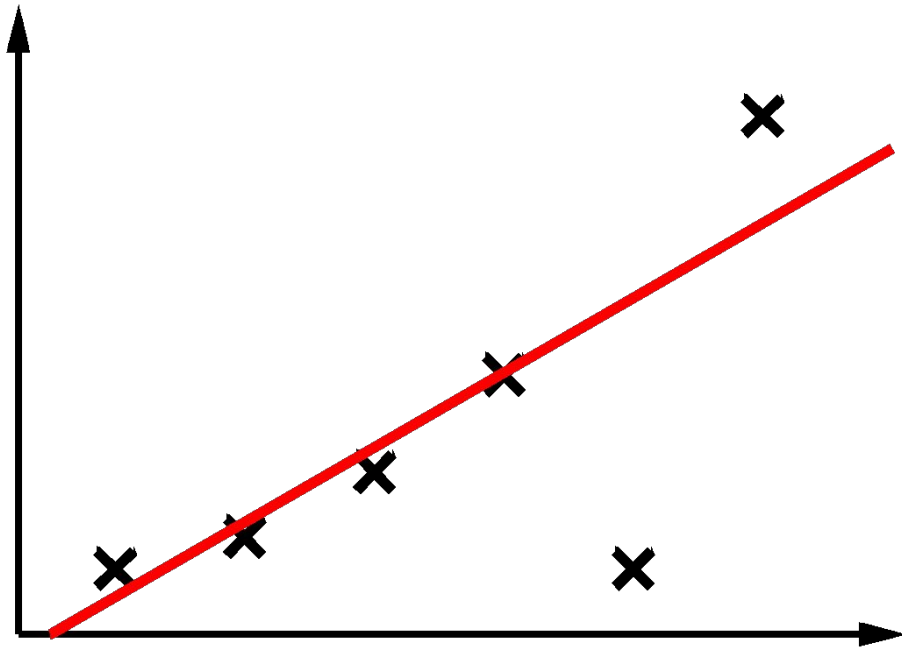
# Regression example: Curve fitting

---



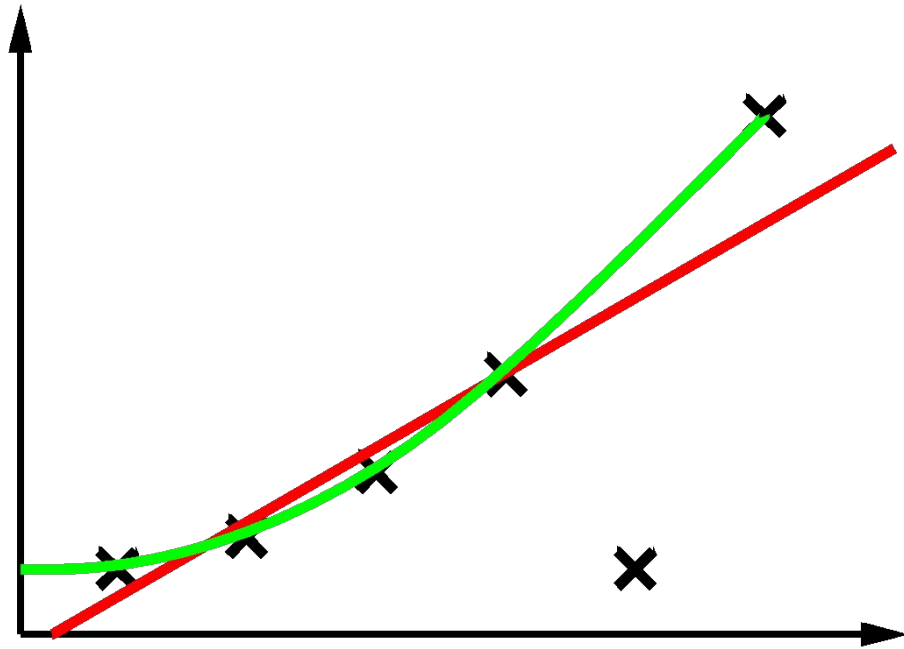
# Regression example: Curve fitting

---



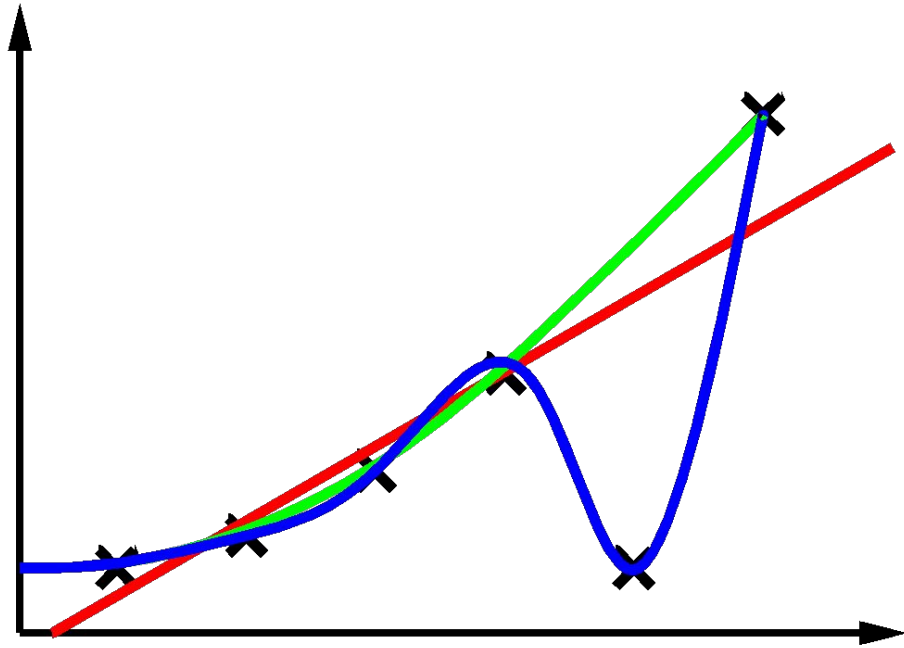
# Regression example: Curve fitting

---



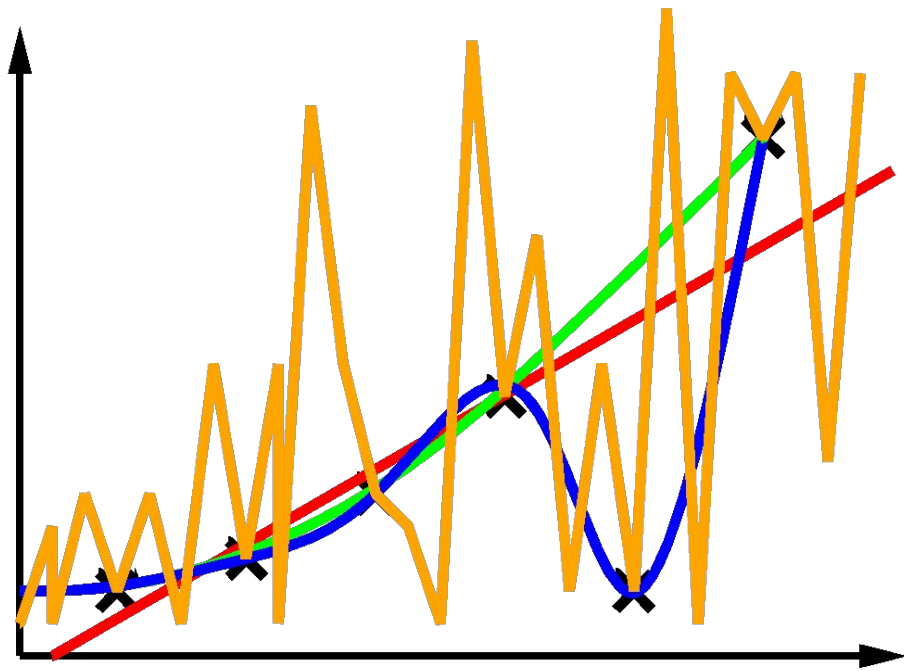
# Regression example: Curve fitting

---



# Regression example: Curve fitting

---



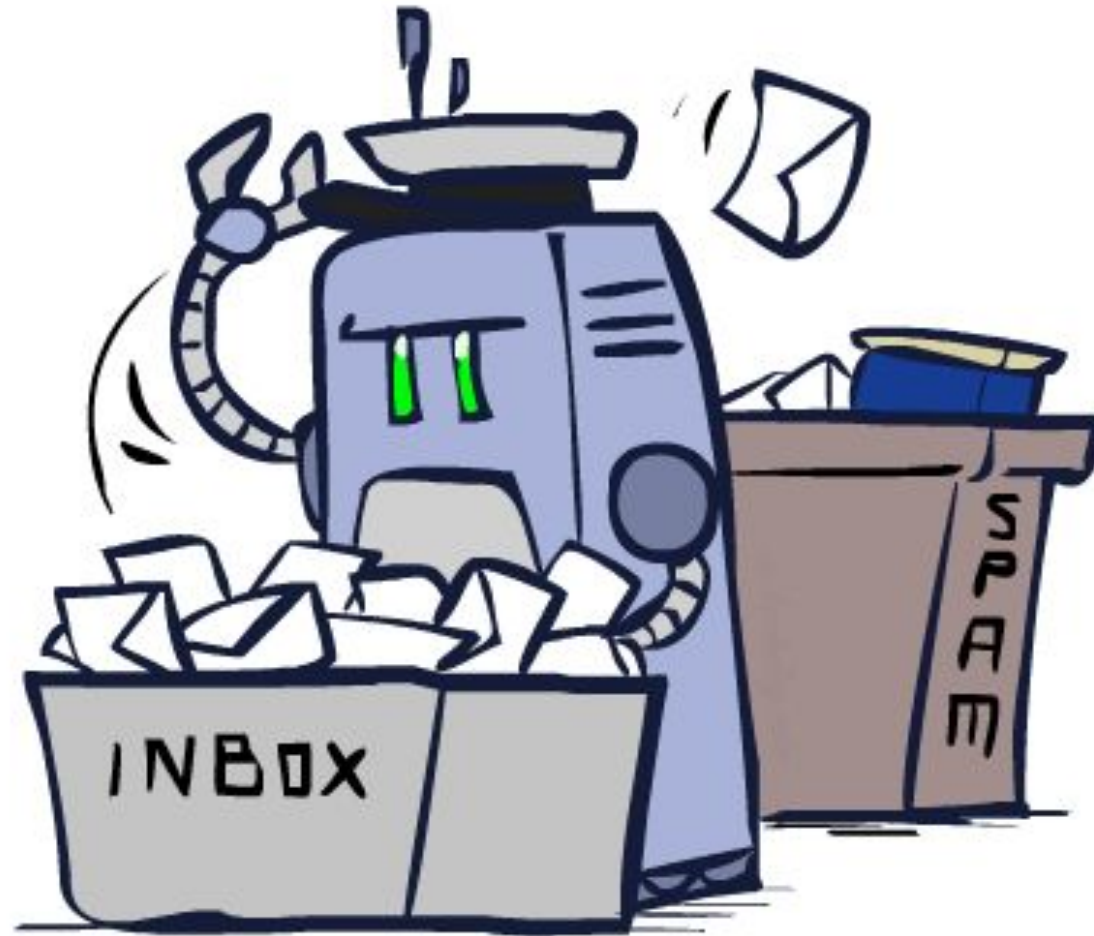
# Basic questions

---

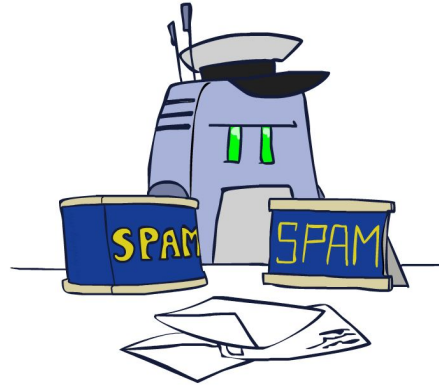
- Which hypothesis space  $H$  to choose?
- How to measure degree of fit?
- How to trade off degree of fit vs. complexity?
  - “*Ockham’s razor*”
- How do we find a good  $h$ ?
- How do we know if a good  $h$  will predict well?

# Classification

---



# Example: Spam Filter



- Input: an email
- Output: spam/ham
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham” (by hand)
  - Learn to predict labels of new incoming emails
  - Classifiers reject 200 billion spam emails per day
- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: \$dd, CAPS
  - Non-text: SenderInContacts, AnchorLinkMismatch
  - ...



Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top secret. ...



TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

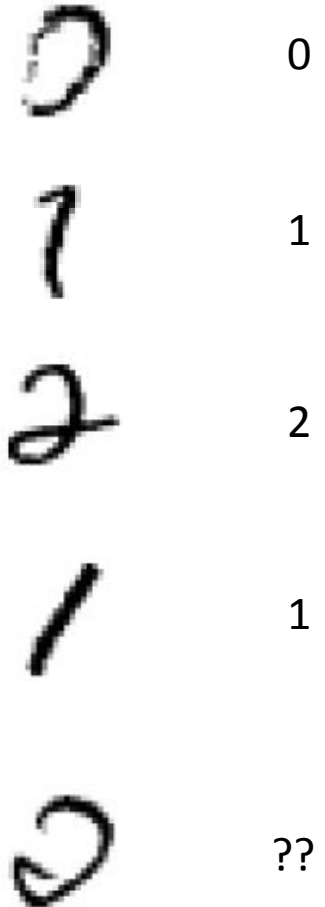
99 MILLION EMAIL ADDRESSES FOR ONLY \$99



Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

# Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9
- Setup:
  - MNIST data set of 60K collection hand-labeled images
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images
- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - Various filter response maps ...



# Other Classification Tasks

- Medical diagnosis
  - input: symptoms
  - output: disease
- Automatic essay grading
  - input: document
  - output: grades
- Fraud detection
  - input: account activity
  - output: fraud / no fraud
- Email routing
  - input: customer complaint email
  - output: which department needs to ignore this email
- Fruit and vegetable inspection
  - input: image (or gas analysis)
  - output: moldy or OK
- ... many more



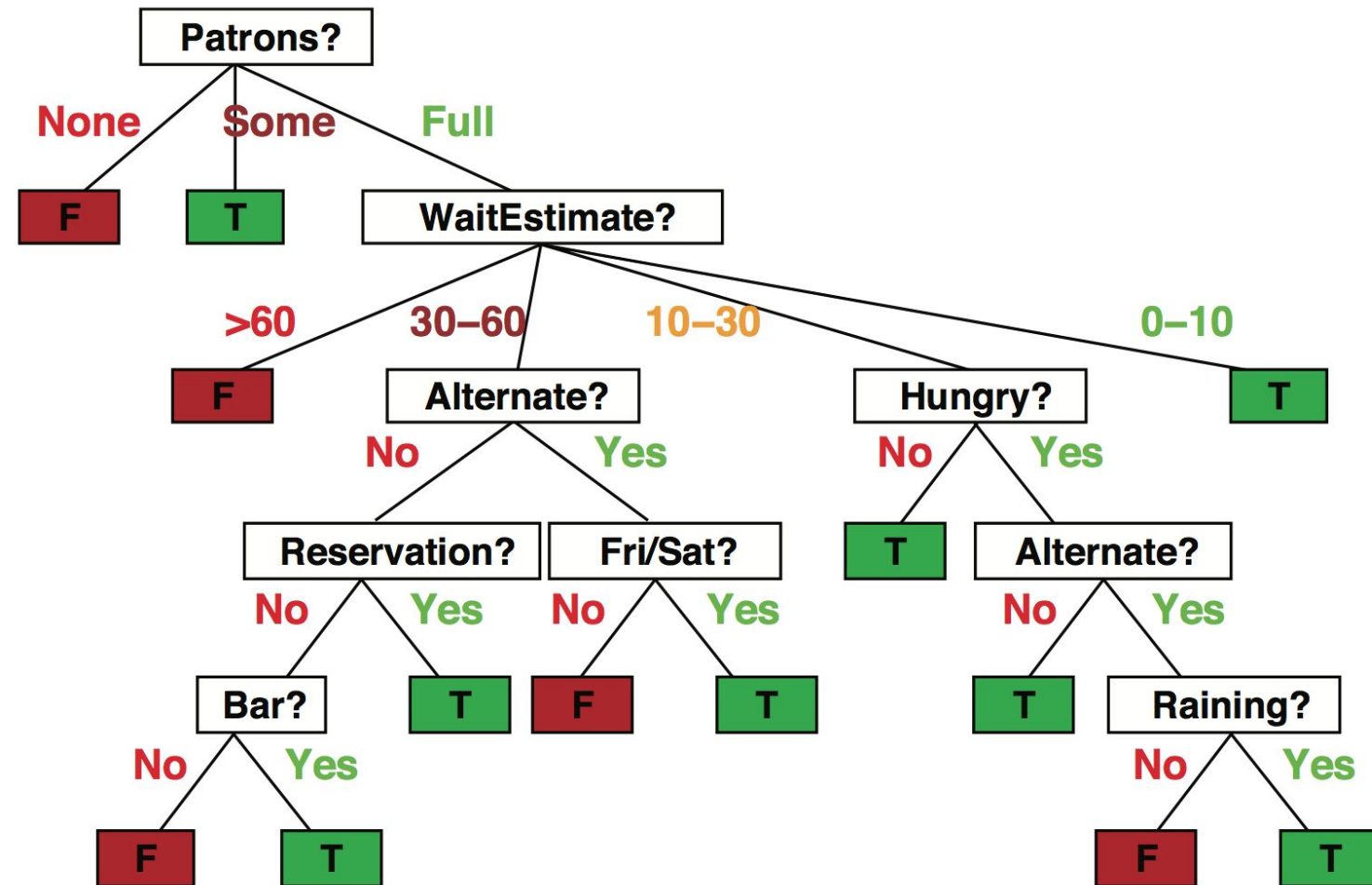
# Decision tree learning

---

- Decision tree models
- Tree construction: keeping it simple
- Measuring learning performance

# Decision trees

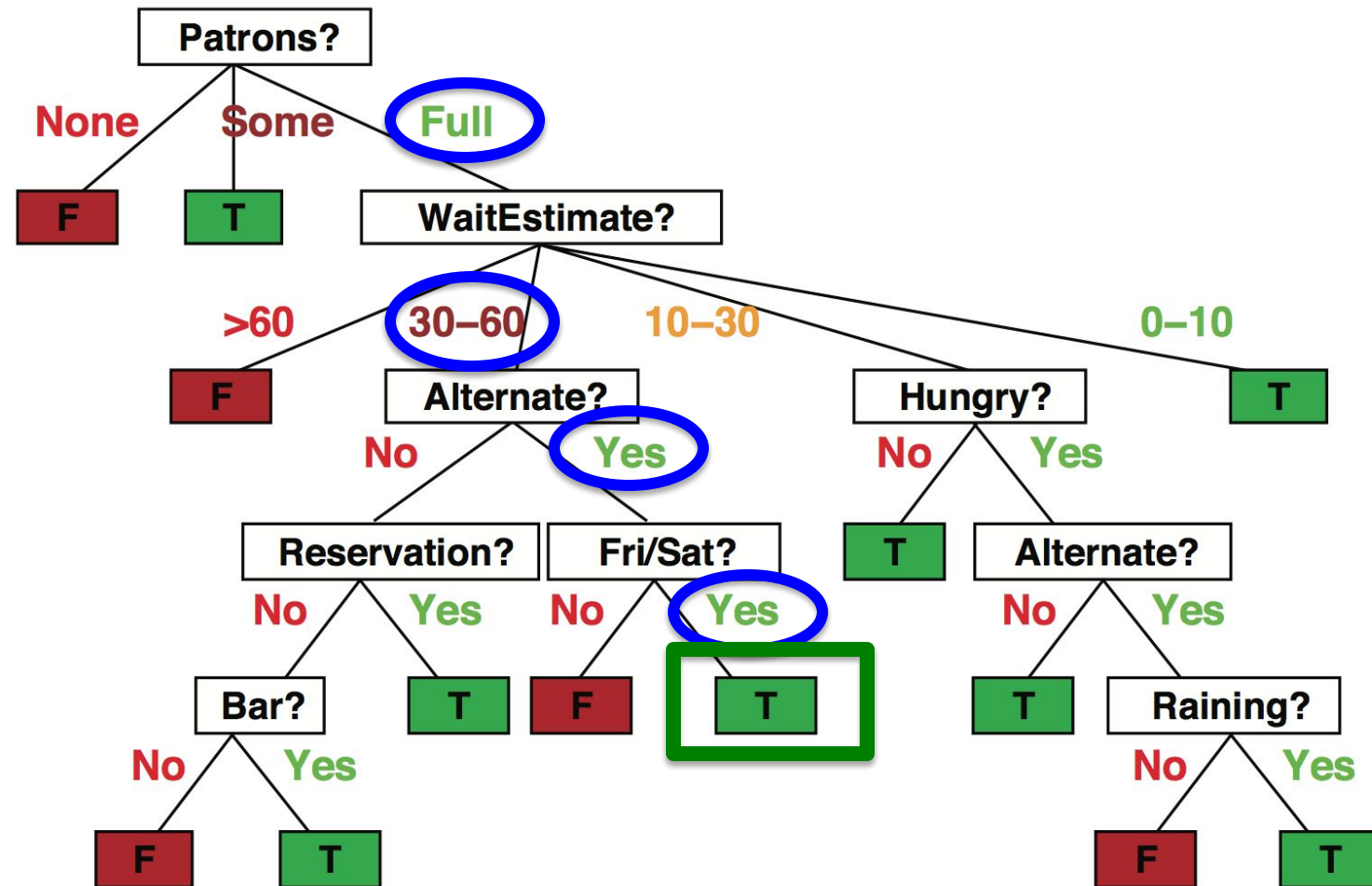
- Popular representation for classifiers
  - Even among humans!
- I've just arrived at a restaurant: should I stay (and wait for a table) or go elsewhere?



# Decision trees

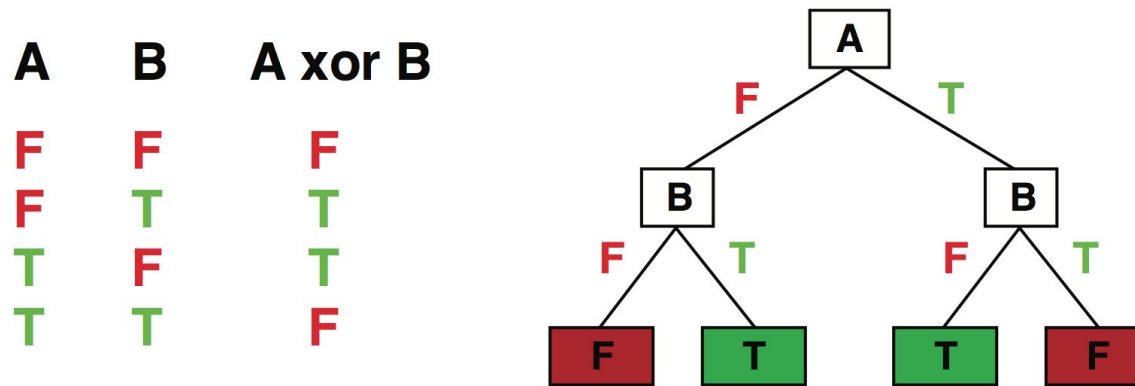
- It's Friday night and you're hungry
- You arrive at your favorite cheap but really cool happening burger place
- It's full up and you have no reservation but there is a bar
- The host estimates a 45 minute wait
- There are alternatives nearby but it's raining outside

Decision tree *partitions* the input space, assigns a label to each partition



# Expressiveness

- Discrete decision trees can express *any function* of the input
- E.g., for Boolean functions, build a path from root to leaf for each row of the truth table:



- Trivially there is a consistent decision tree that fits any training set exactly (unless true function is nondeterministic)
- But a tree that simply records the examples is essentially a lookup table
- To get generalization to new examples, need a *compact* tree

# Quiz

---

- How many distinct decision trees with  $n$  Boolean attributes?
  - = number of distinct Boolean functions with  $n$  inputs
  - = number of truth tables with  $n$  inputs
  - = number of ways of filling in output column with  $2^n$  entries
  - =  $2^{2n}$
  - For  $n=6$  attributes, there are 18,446,744,073,709,551,616 trees

# Hypothesis spaces in general

- Increasing the expressiveness of the hypothesis language
  - Increases the chance that the true function can be expressed
  - Increases the number of hypotheses that are consistent with training set
    - => many consistent hypotheses have large test error
    - => may **reduce** prediction accuracy!
- With  $2^{2n}$  hypotheses, all but an exponentially small fraction will require  $O(2^n)$  bits to express in **any** representation (even brains!)
- I.e., any given representation can represent only an exponentially small fraction of hypotheses concisely; no universal compression
- E.g., decision trees are bad at “k-out-of-n” functions

# Training data

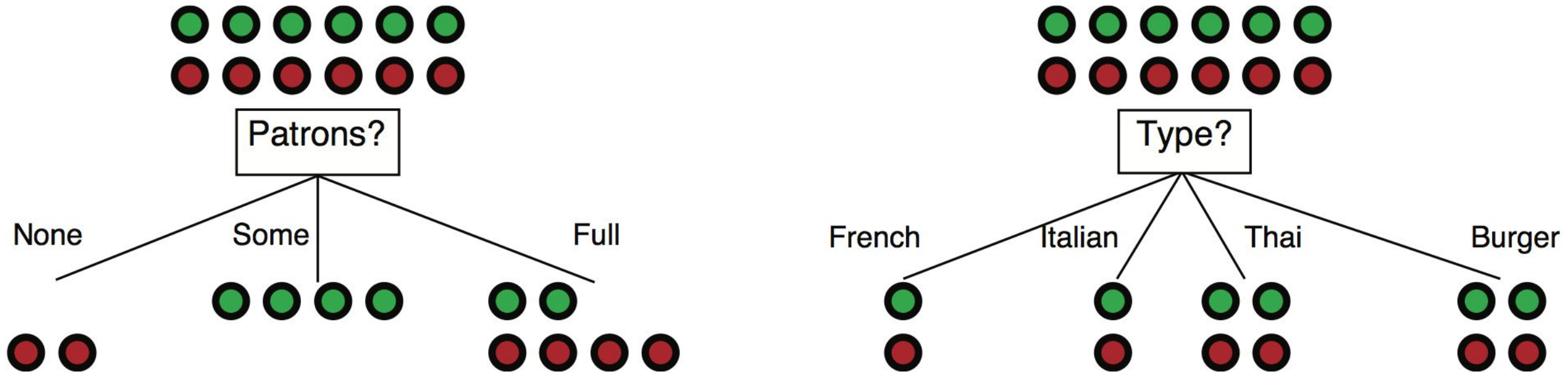
Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
$x_1$	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	$y_1 = \text{Yes}$
$x_2$	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	$y_2 = \text{No}$
$x_3$	No	Yes	No	No	Some	\$	No	No	Burger	0-10	$y_3 = \text{Yes}$
$x_4$	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10-30	$y_4 = \text{Yes}$
$x_5$	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 = \text{No}$
$x_6$	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	$y_6 = \text{Yes}$
$x_7$	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	$y_7 = \text{No}$
$x_8$	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	$y_8 = \text{Yes}$
$x_9$	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 = \text{No}$
$x_{10}$	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	$y_{10} = \text{No}$
$x_{11}$	No	No	No	No	None	\$	No	No	Thai	0-10	$y_{11} = \text{No}$
$x_{12}$	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	$y_{12} = \text{Yes}$

# Decision tree learning

```
function Decision-Tree-Learning(examples, attributes, parent_examples) returns a tree
if examples is empty then return Plurality-Value(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return Plurality-Value(examples)
else  $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{Importance}(a, \text{examples})$ 
      tree  $\leftarrow$  a new decision tree with root test  $A$ 
      for each value  $v$  of  $A$  do
        exs  $\leftarrow$  the subset of examples with value  $v$  for attribute  $A$ 
        subtree  $\leftarrow$  Decision-Tree-Learning(exs, attributes -  $A$ , examples)
        add a branch to tree with label  $(A = v_k)$  and subtree subtree
return tree
```

# Choosing an attribute: Information gain

- Idea: measure contribution of attribute to increasing “purity” of labels in each subset of examples



- Patrons is a better choice: gives *information* about classification (i.e., reduces *entropy* of the distribution of labels)

# Information

- Information answers questions
- The more clueless I am about the answer initially, the more information is contained in the answer
- Scale: **1 bit** = answer to Boolean question with prior  $\langle 0.5, 0.5 \rangle$
- Information in an answer when prior is  $\langle p_1, \dots, p_n \rangle$  is
$$H(\langle p_1, \dots, p_n \rangle) = \sum_i -p_i \log p_i$$
  - This is the **entropy** of the prior
- Convenient notation:  $B(p) = H(\langle p, 1-p \rangle)$

# Information gain from splitting on an attribute

- Suppose we have  $p$  positive and  $n$  negative examples at the root

- $\Rightarrow B(p/(p+n))$  bits needed to classify a new example

- E.g., for 12 restaurant examples,  $p = n = 6$  so we need 1 bit

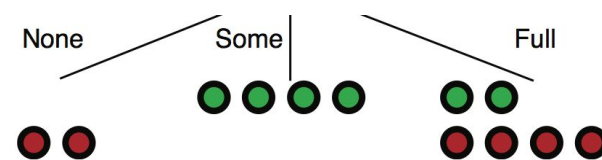


- An attribute splits the examples  $E$  into subsets  $E_k$ , each of which (we hope) needs less information to complete the classification

- For an example in  $E_k$  we expect to need  $B(p_k/(p_k+n_k))$  more bits

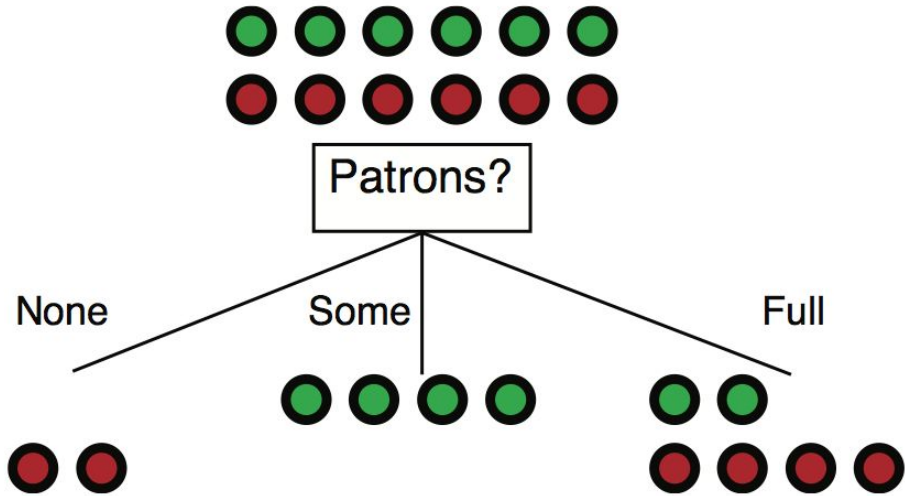
- Probability a new example goes into  $E_k$  is  $(p_k+n_k)/(p+n)$

- Expected* number of bits needed after split is  $\sum_k (p_k+n_k)/(p+n) B(p_k/(p_k+n_k))$

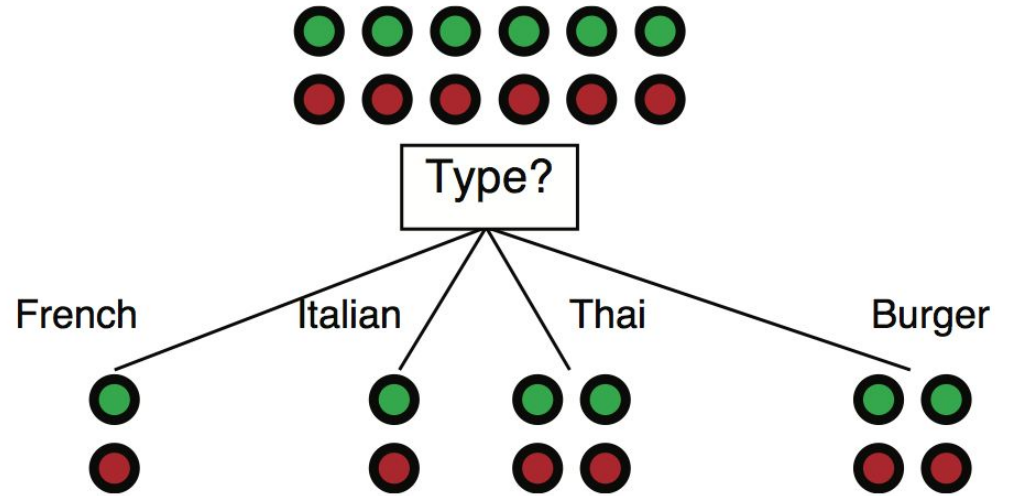


- Information gain =  $B(p/(p+n)) - \sum_k (p_k+n_k)/(p+n) B(p_k/(p_k+n_k))$

# Example



$$1 - [(2/12)B(0) + (4/12)B(1) + (6/12)B(2/6)]$$
$$= 0.541 \text{ bits}$$

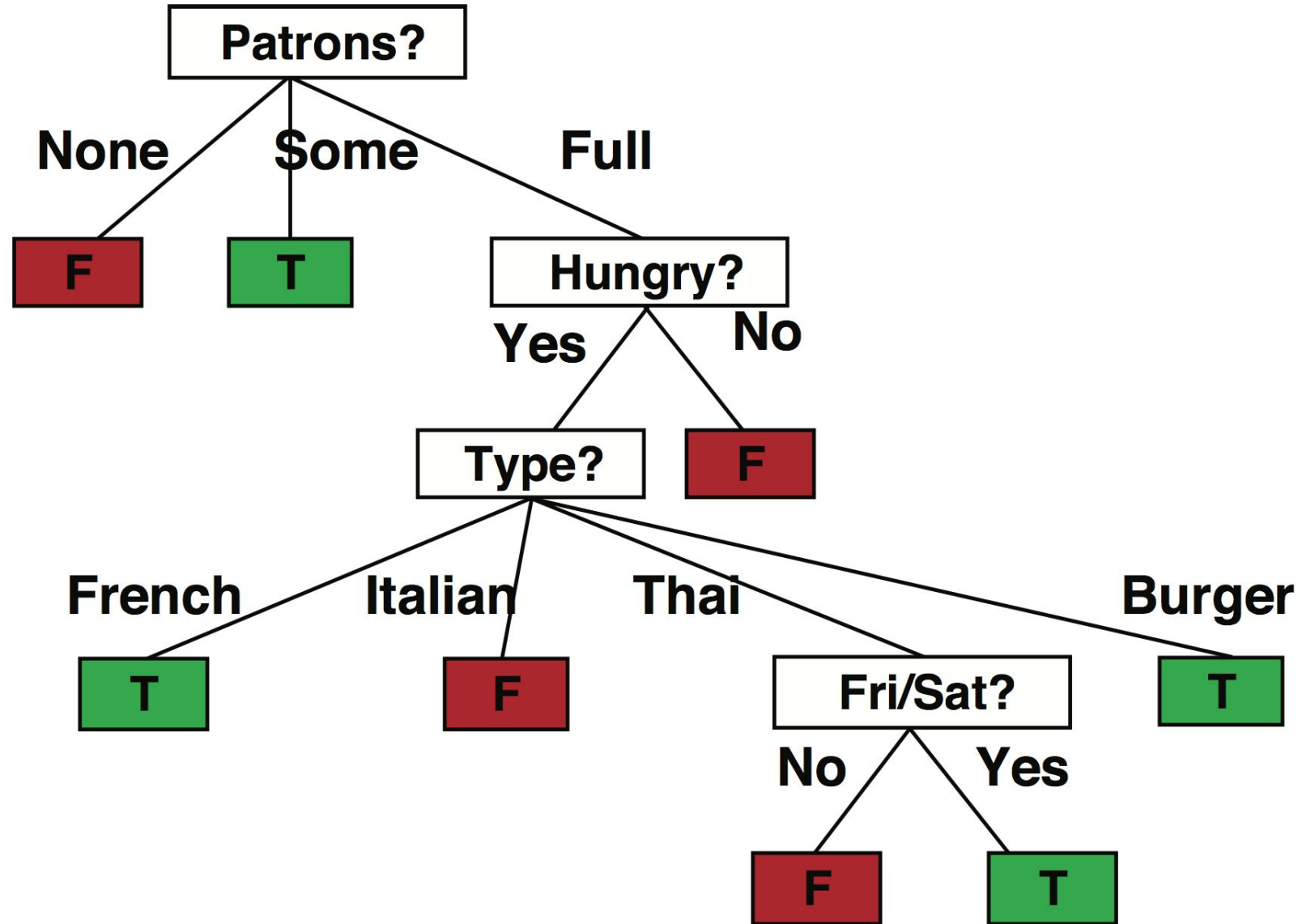


$$1 - [(2/12)B(1/2) + (2/12)B(1/2) +$$
$$(4/12)B(2/6) + (4/12)B(1/2)]$$
$$= 0 \text{ bits}$$

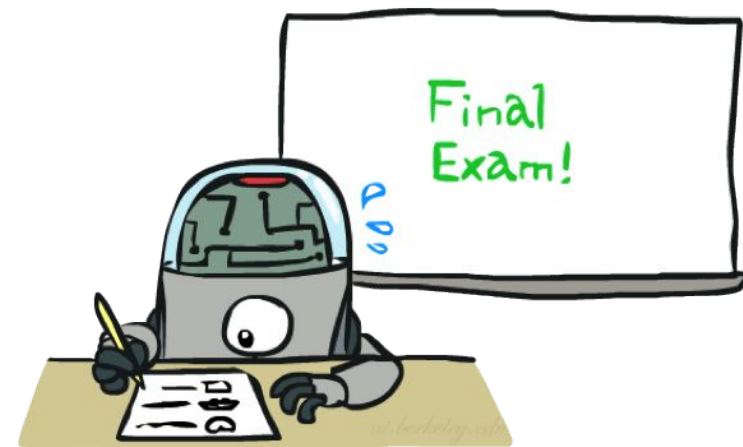
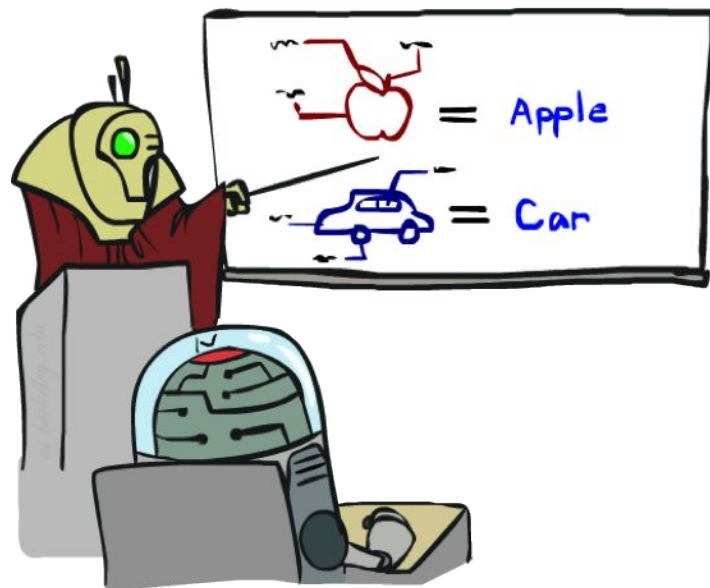
# Results for restaurant data

Decision tree learned from the 12 examples:

- Simpler than “true” tree!

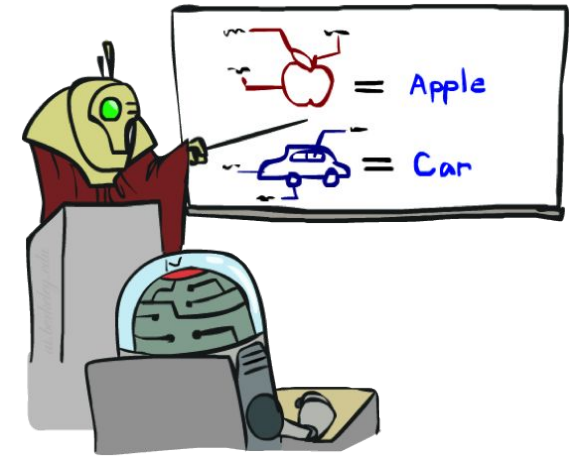
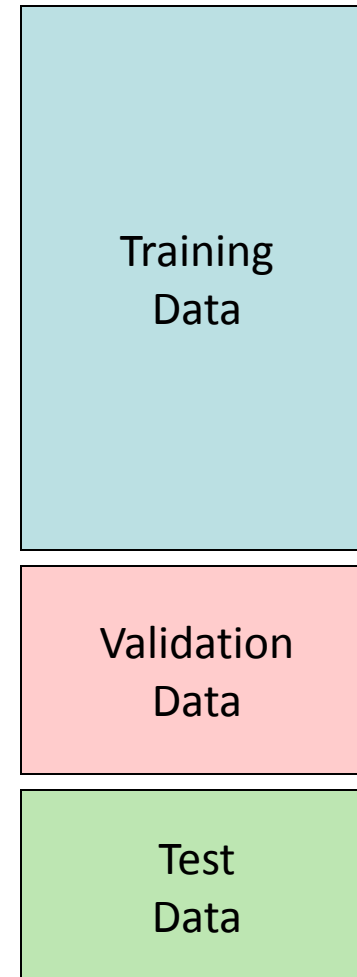


# Training and Testing

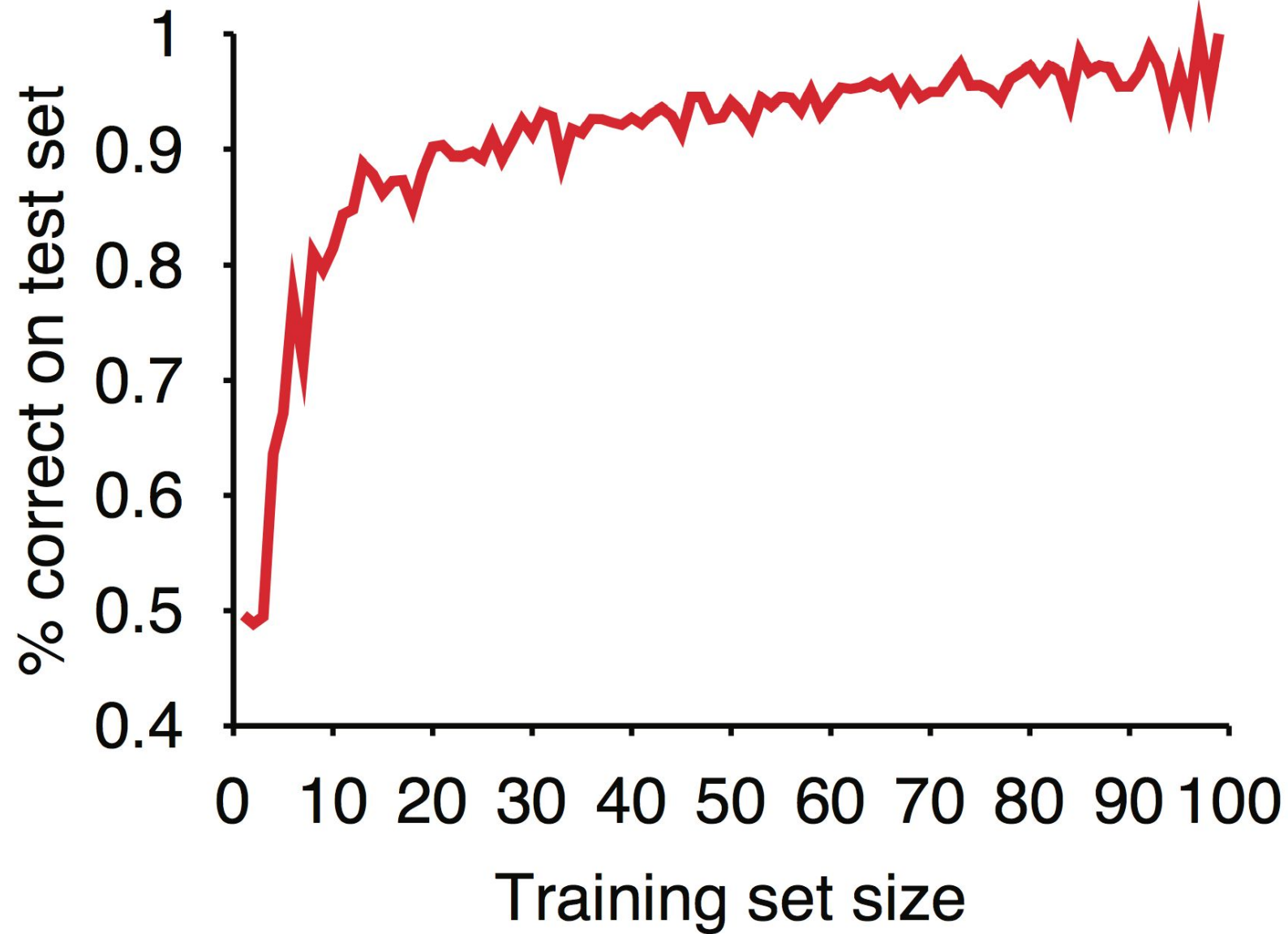


# Basic Concepts

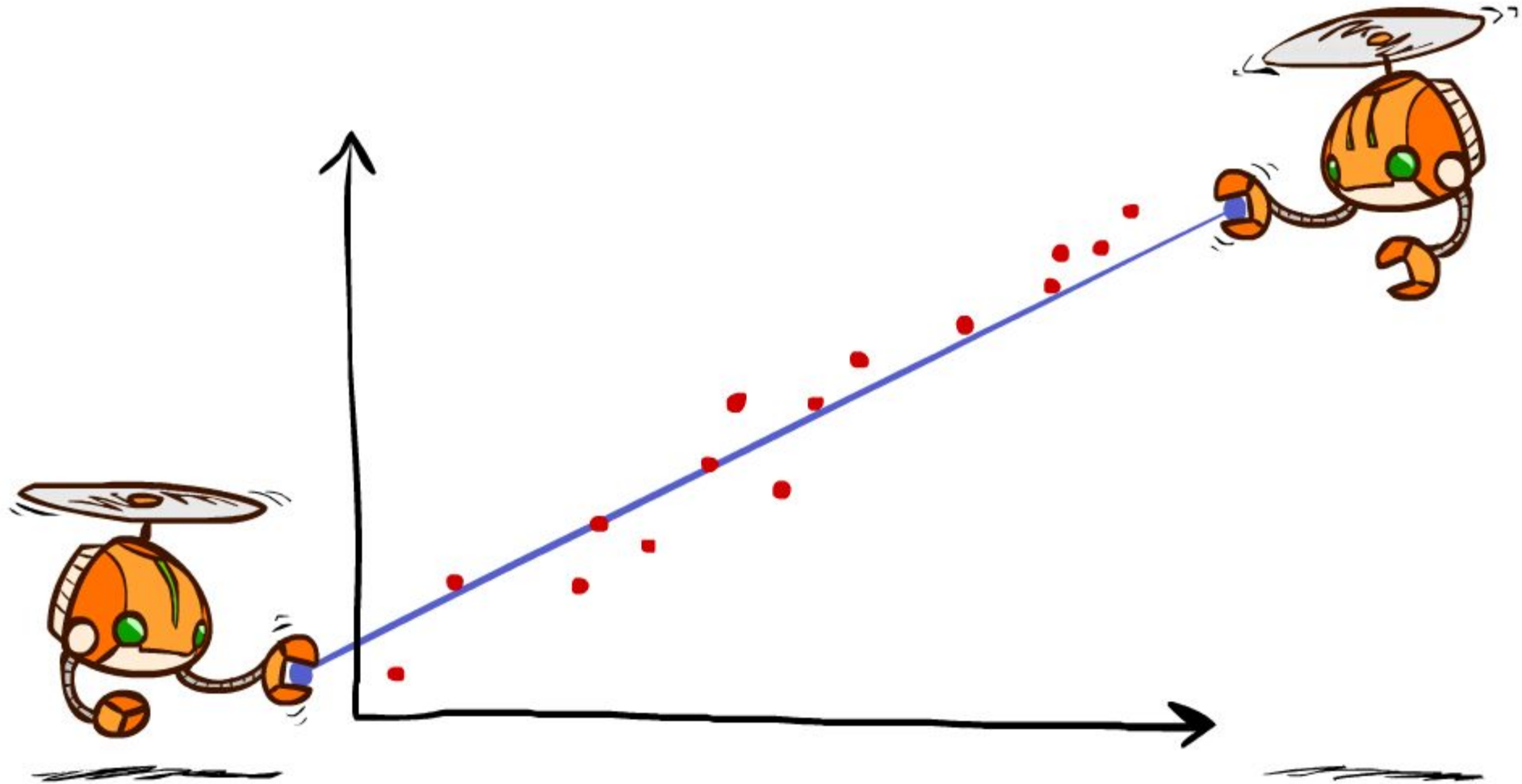
- Data: labeled instances, e.g. restaurant episodes
  - **Training set**
  - **Validation set**
  - **Test set**
- Experimentation cycle
  - Generate hypothesis  $h$  from training set
  - (Possibly choose best  $h$  by trying out on validation set)
  - Finally, compute accuracy of  $h$  on test set
  - Very important: never **peek** at the test set!
- Evaluation
  - **Accuracy**: fraction of instances predicted correctly
  - **Learning curve**: test set accuracy as a function of training set size



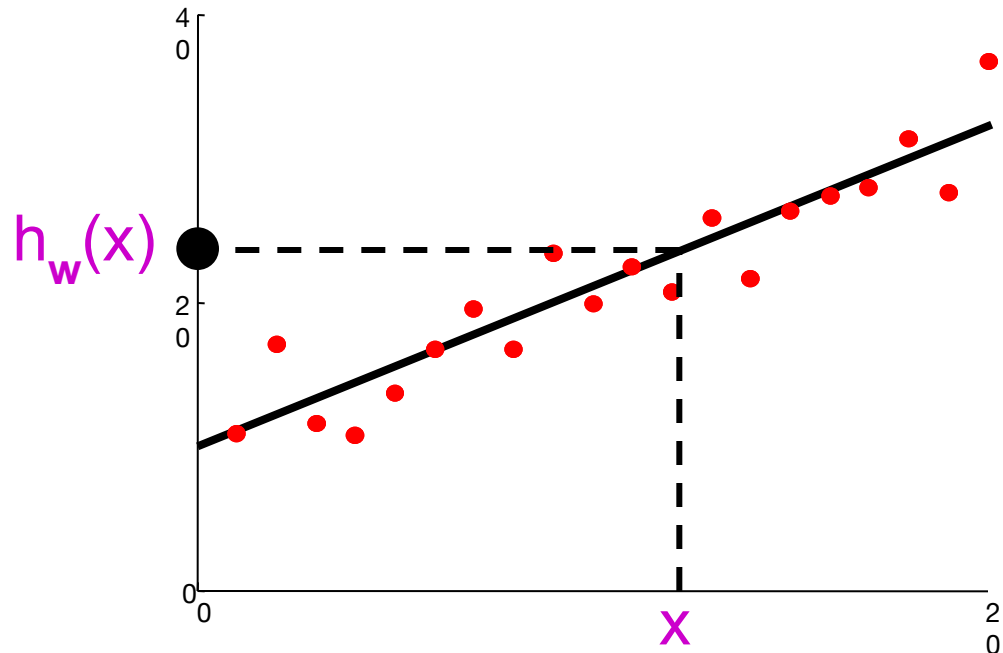
# Results for restaurant data



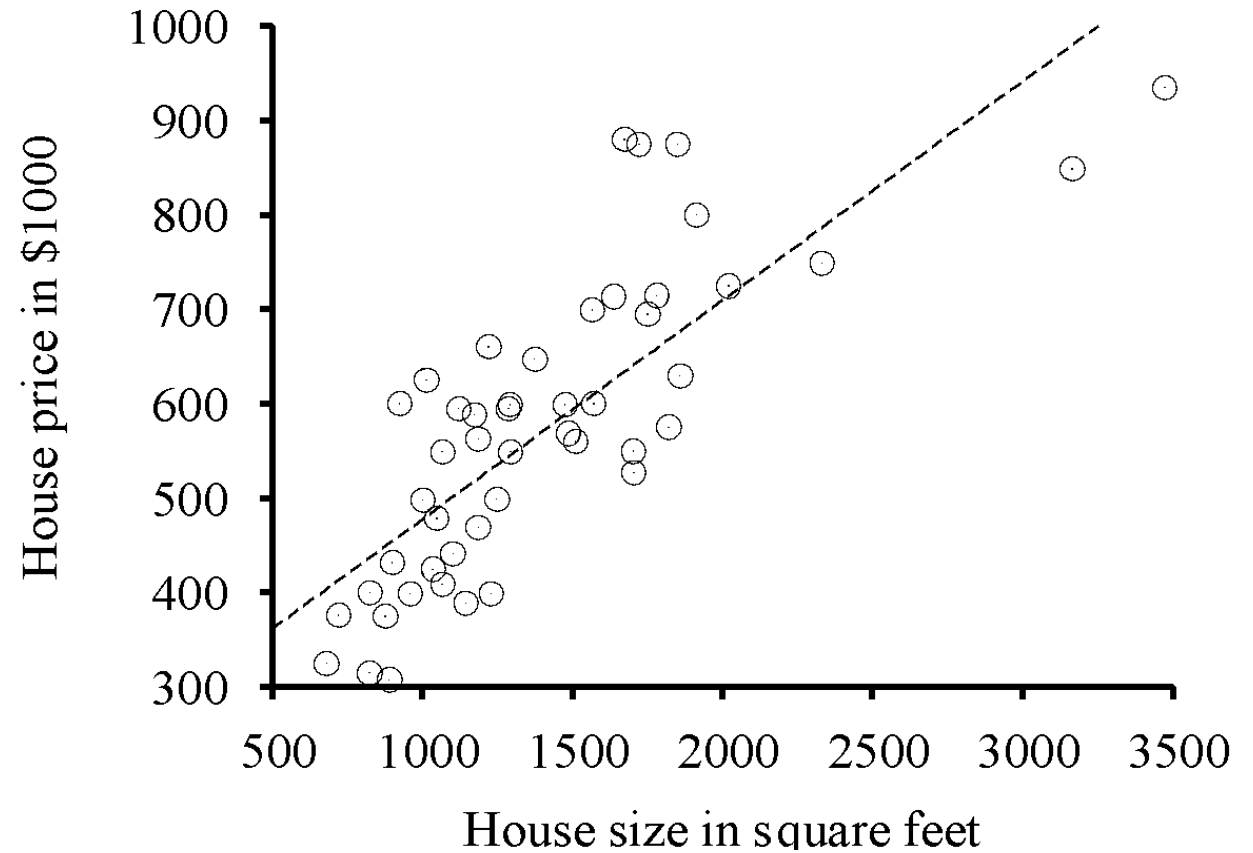
# Linear Regression



# Linear regression = fitting a straight line/hyperplane



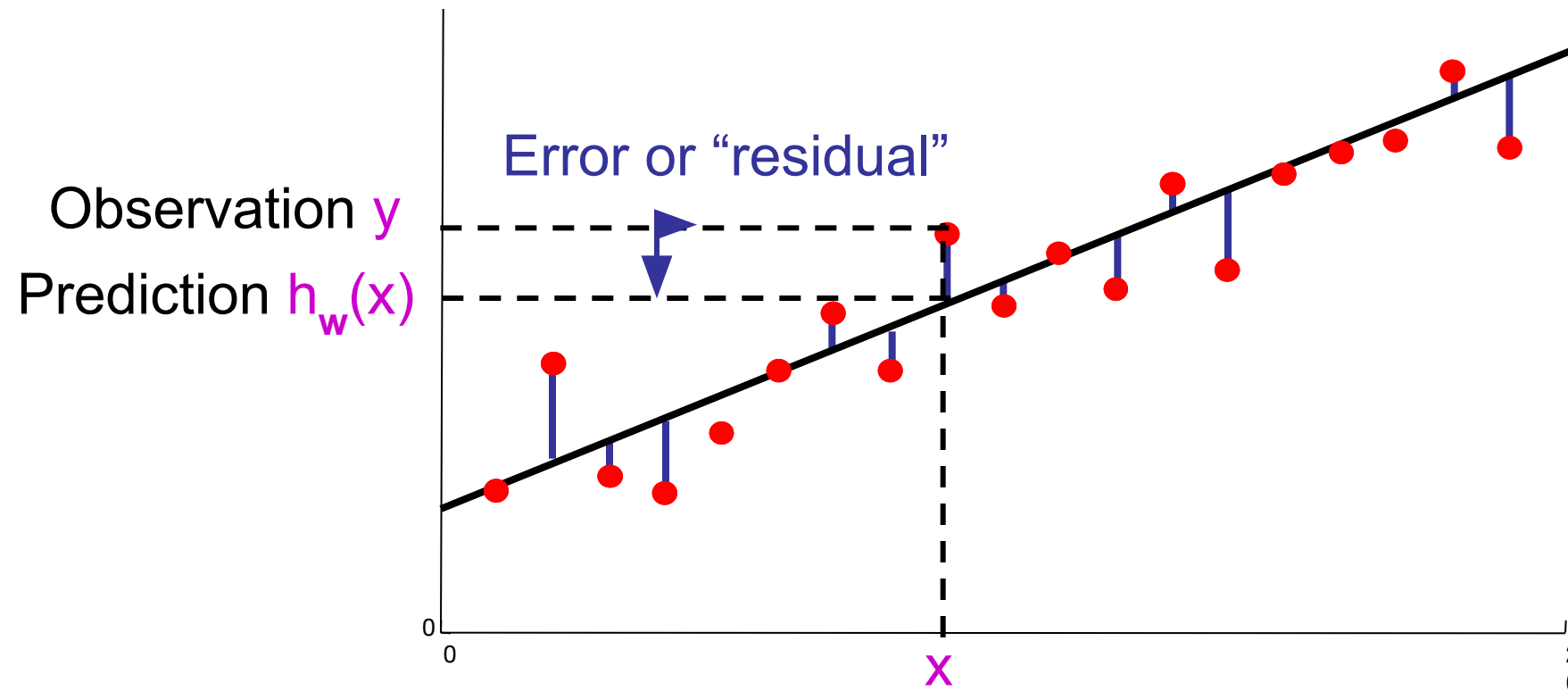
Prediction:  $h_w(x) = w_0 + w_1x$



Berkeley house prices, 2009

# Prediction error

Error on one instance:  $y - h_w(x)$



# Least squares: Minimizing squared error

- L2 loss function: sum of squared errors over all examples
  - $\text{Loss} = \sum_j (y_j - h_w(x_j))^2 = \sum_j (y_j - (w_0 + w_1 x_j))^2$
- We want the weights  $w^*$  that minimize loss
- At  $w^*$  the derivatives of loss w.r.t. each weight are zero:
  - $\partial \text{Loss} / \partial w_0 = -2 \sum_j (y_j - (w_0 + w_1 x_j)) = 0$
  - $\partial \text{Loss} / \partial w_1 = -2 \sum_j (y_j - (w_0 + w_1 x_j)) x_j = 0$
- Exact solutions for  $N$  examples:
  - $w_1 = [N \sum_j x_j y_j - (\sum_j x_j)(\sum_j y_j)] / [N \sum_j x_j^2 - (\sum_j x_j)^2]$  and  $w_0 = 1/N [\sum_j y_j - w_1 \sum_j x_j]$
- For the general case where  $x$  is an  $n$ -dimensional vector
  - $X$  is the data matrix (all the data, one example per row);  $y$  is the column of labels
  - $w^* = (X^T X)^{-1} X^T y$

# Summary

---

- Learning is *essential* in unknown environments, *useful* in many others
- The nature of the learning process depends on
  - agent design
  - what part of the agent you want to improve
  - what prior knowledge and new experiences are available
- Supervised learning: learning a function from labeled examples
  - Concise hypotheses generalize better; trade off conciseness and accuracy
  - Classification: discrete-valued function; example: decision trees
  - Regression: real-valued function; example: linear regression
- Next: Bayesian learning