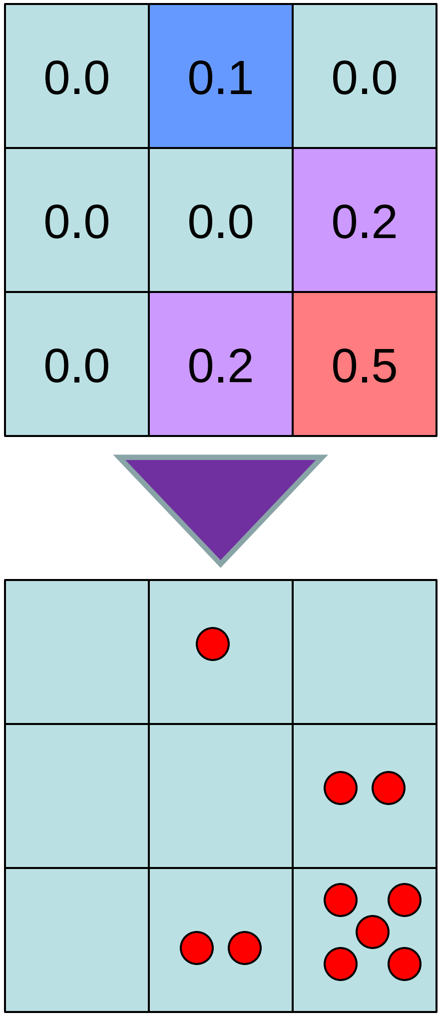
CSE 473

Lecture 24
(Chapter 18)

From Particle Filtering to Supervised Learning and Decision Trees

Last Time: Particle Filtering

- Sometimes |X| is too big for exact inference
  - |X| may be too big to even store $P(X_t \mid e_{1:t})$
    E.g. when X is continuous
- Solution: Approximate inference
  - Track a set of *samples* of X
  - Samples are called ***particles***
  - Number of samples for X=x is proportional to probability of x



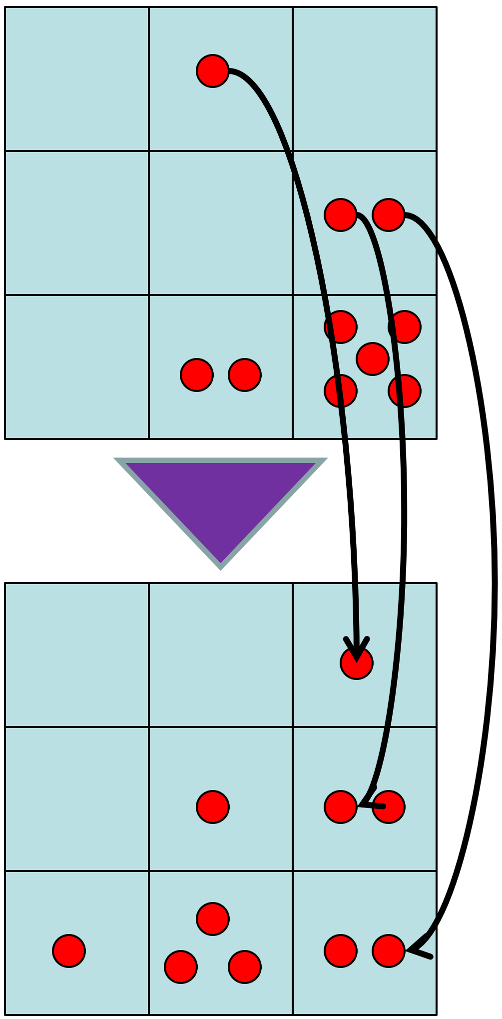
Particle Filtering
Step 1: Elapse Time

- Each particle x is moved by sampling its next position using the transition model

$$x' = \mathsf{sample}(P(X'|x))$$

  - Samples' frequencies reflect the transition probabilities
  - In example, most samples move clockwise, but some move in another direction or stay in place

- This step captures passage of time



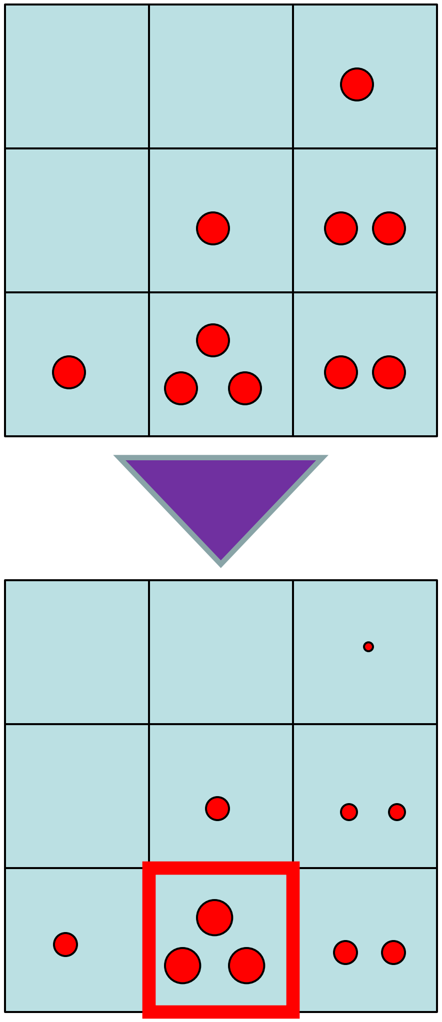
Particle Filtering
Step 2: Observe

Weight particles according to evidence

- ***Assign weights*** w ***to samples*** based on the new observed evidence e

$$w(x) = P(e|x)$$

- In example, true ghost position is shown in red outline; samples closer to ghost get higher weight (bigger size of circles) based on noisy distance emission model



Particle Filtering
Step 3: Resample

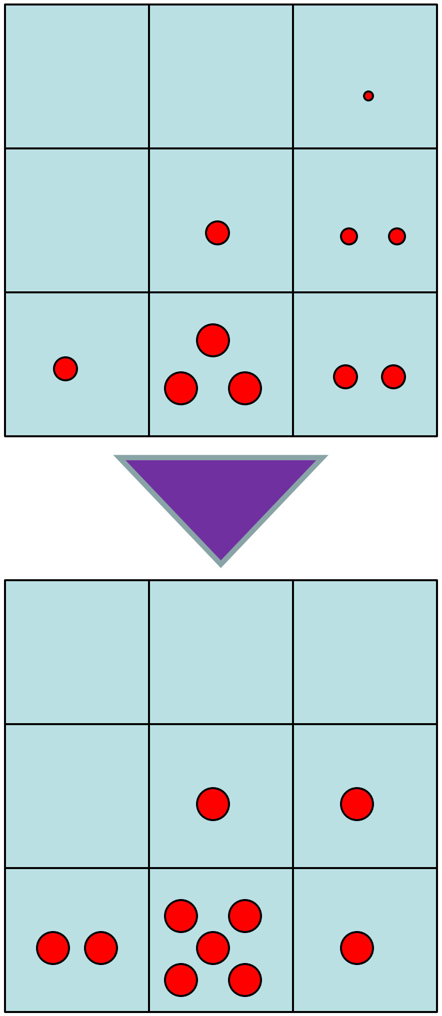
- N times, we choose from our weighted sample distribution (i.e. randomly select with replacement)
  - Each sample selected with probability proportional to its weight
- Now the update is complete for this time step, continue with the next one

Old Particles:

(1,3) w=0.1
(3,2) w=0.9
(3,2) w=0.9
(3,1) w=0.4
(2,3) w=0.3
(2,2) w=0.4
(3,3) w=0.4
(3,3) w=0.4
(3,2) w=0.9
(2,3) w=0.3

New Particles:

(3,2) w=1
(3,2) w=1
(3,2) w=1
(2,3) w=1
(2,2) w=1
(3,2) w=1
(3,1) w=1
(3,3) w=1
(3,2) w=1
(3,1) w=1



Particle Filtering Summary

- Represent current belief P(X | evidence to date) as set of *N* samples (actual values x)
- For each new observation e:

  1. *Sample transition*, once for each current particle x

$$x' = \text{sample}(P(X'|x))$$

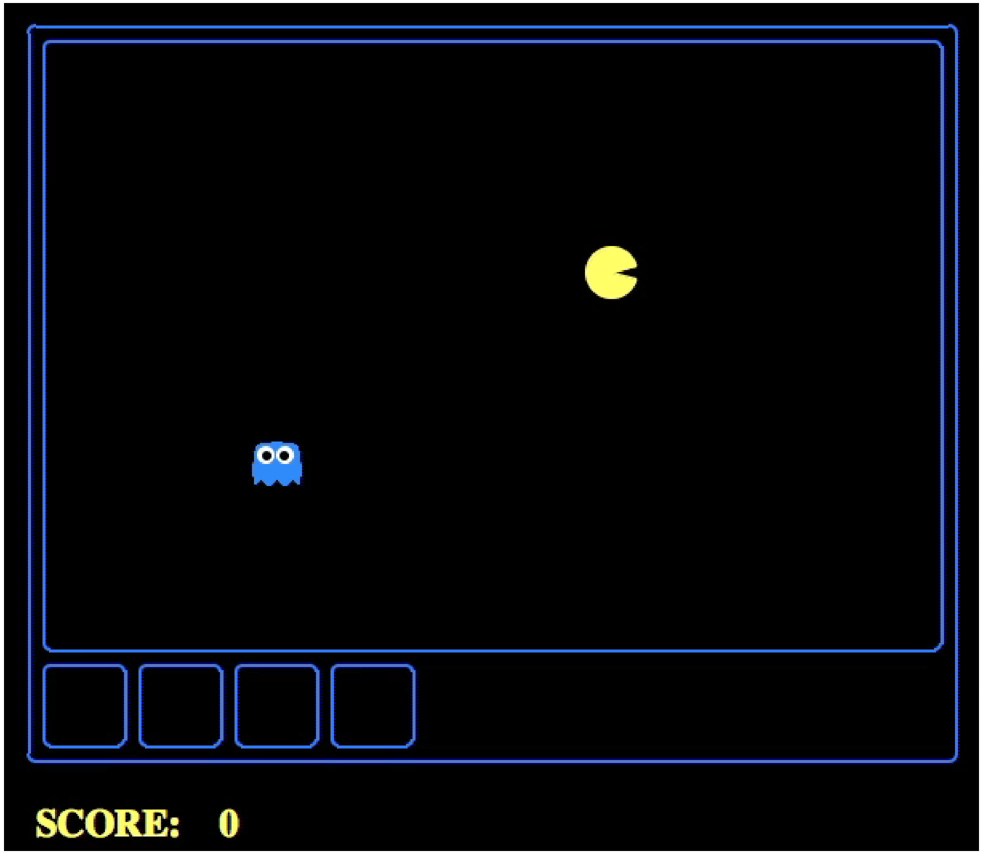
  2. For each new sample x', *compute importance weights* for the new evidence e:

$$w(x') = P(e|x')$$

  3. Finally, *resample* the importance weights to create N new particles

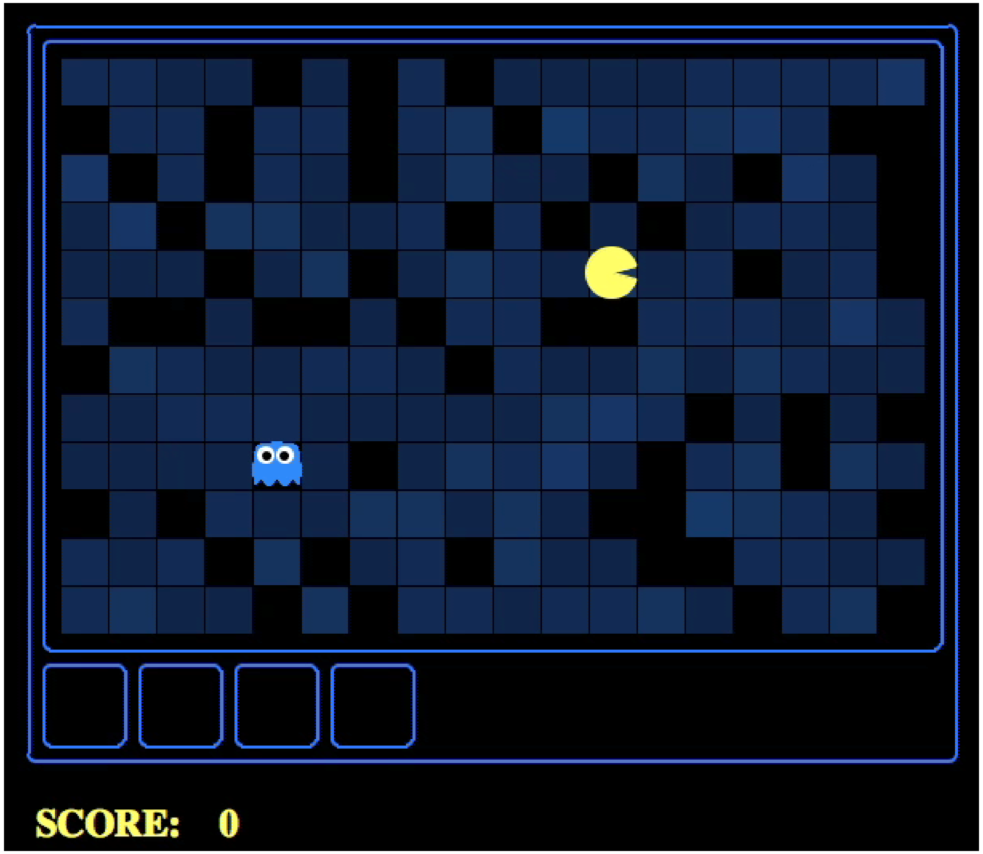
Example 1

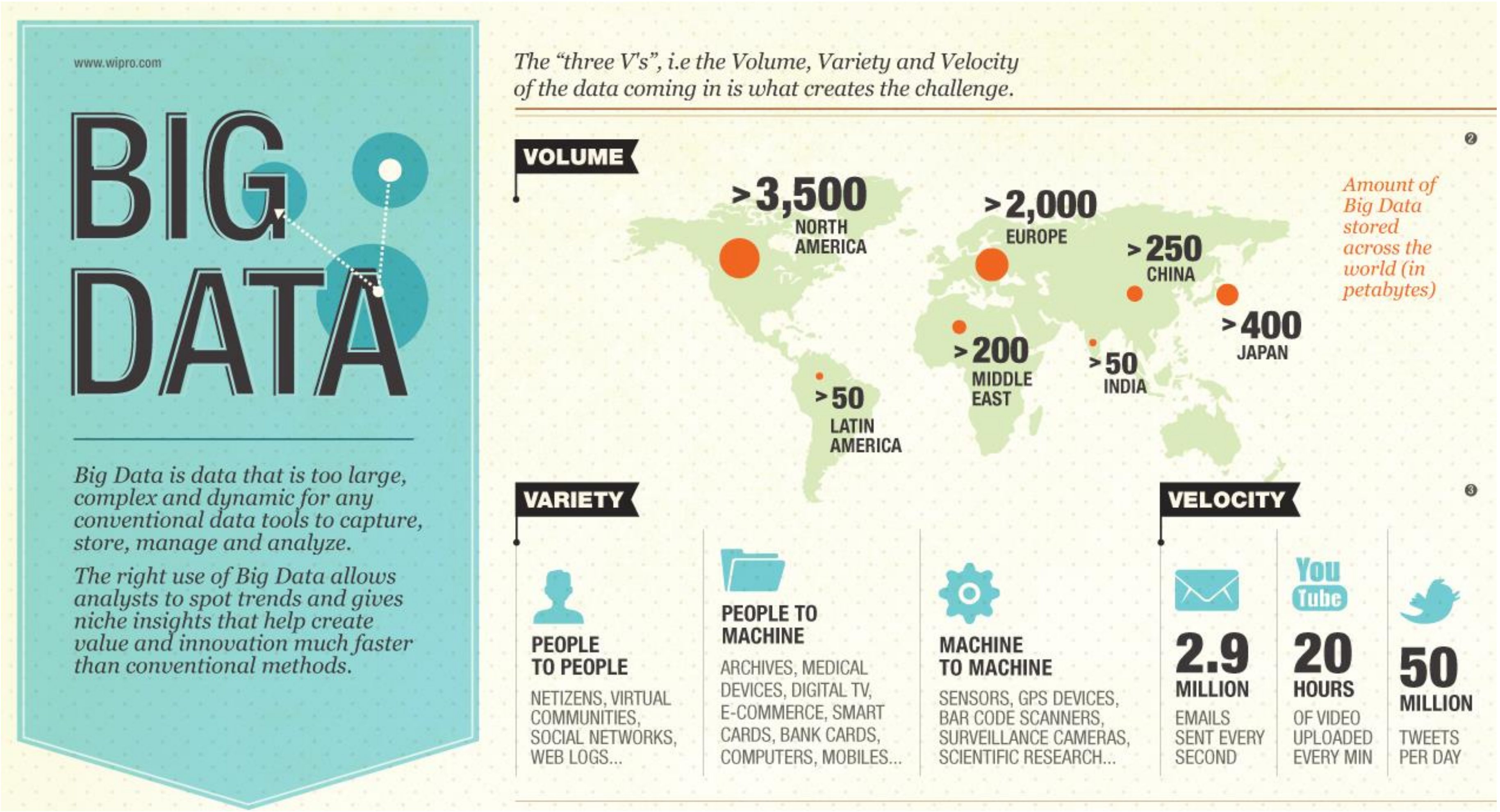
Particle filter, uniform initial beliefs, 25 particles



Example 2

Particle filter, uniform initial beliefs, 300 particles





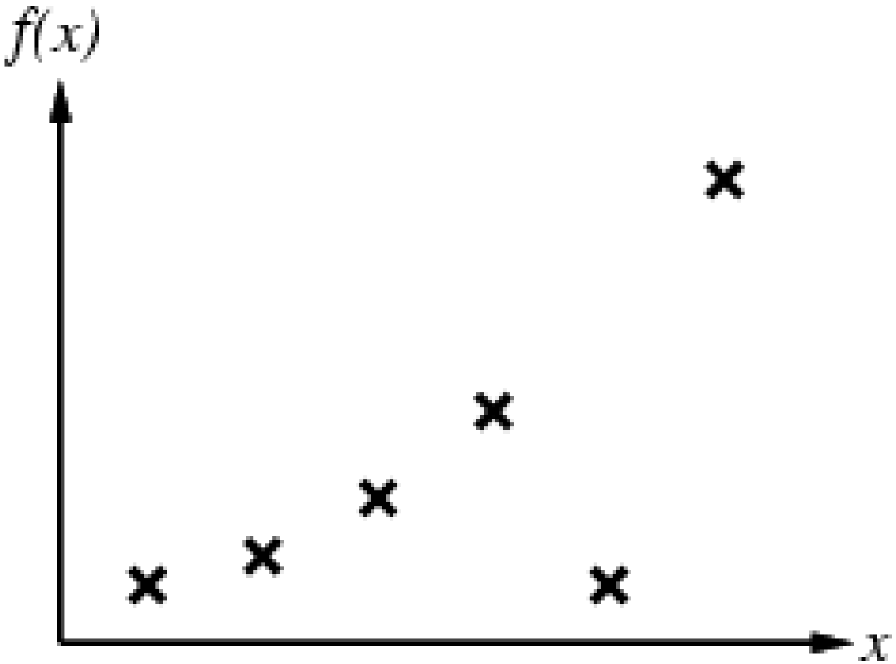
Enter…Machine Learning

Varieties of Machine Learning

- Supervised learning: correct answers for each input is provided, goal is to *generalize* to new data
  - E.g., decision trees, neural networks
- Unsupervised learning: correct answers not given, must *discover patterns* in input data
  - E.g., clustering, principal component analysis
- Reinforcement learning: occasional *rewards* (or punishments) given to guide behavior
  - We've covered this already! (Q-learning, MDPs)

Supervised learning

- Goal: Construct a function *h* from training data to approximate the hidden function *f* that is generating the data
  - *h* is consistent if it agrees with *f* on all training examples



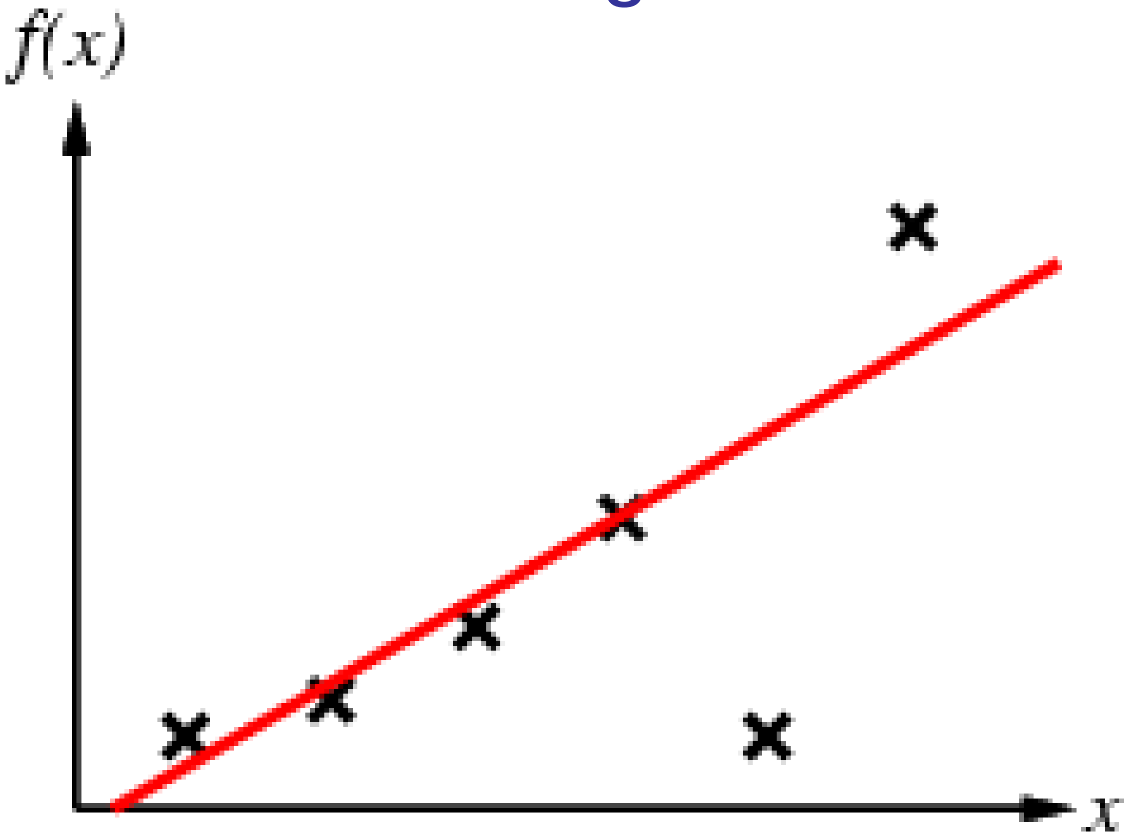
Curve fitting (aka regression)

Given: Data points $(x, f(x))$ (training examples)

What kind of function would you fit?

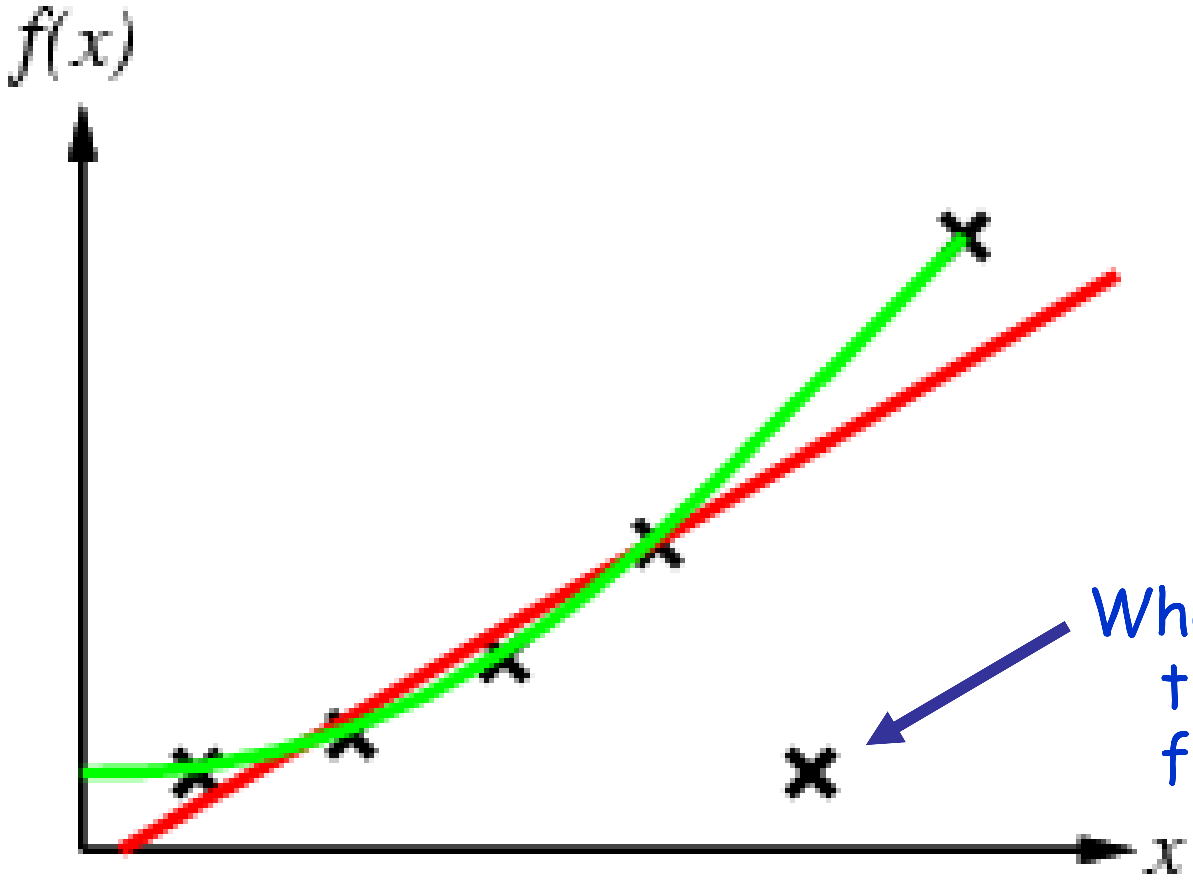
Supervised learning example

h = Straight line?



Supervised learning example

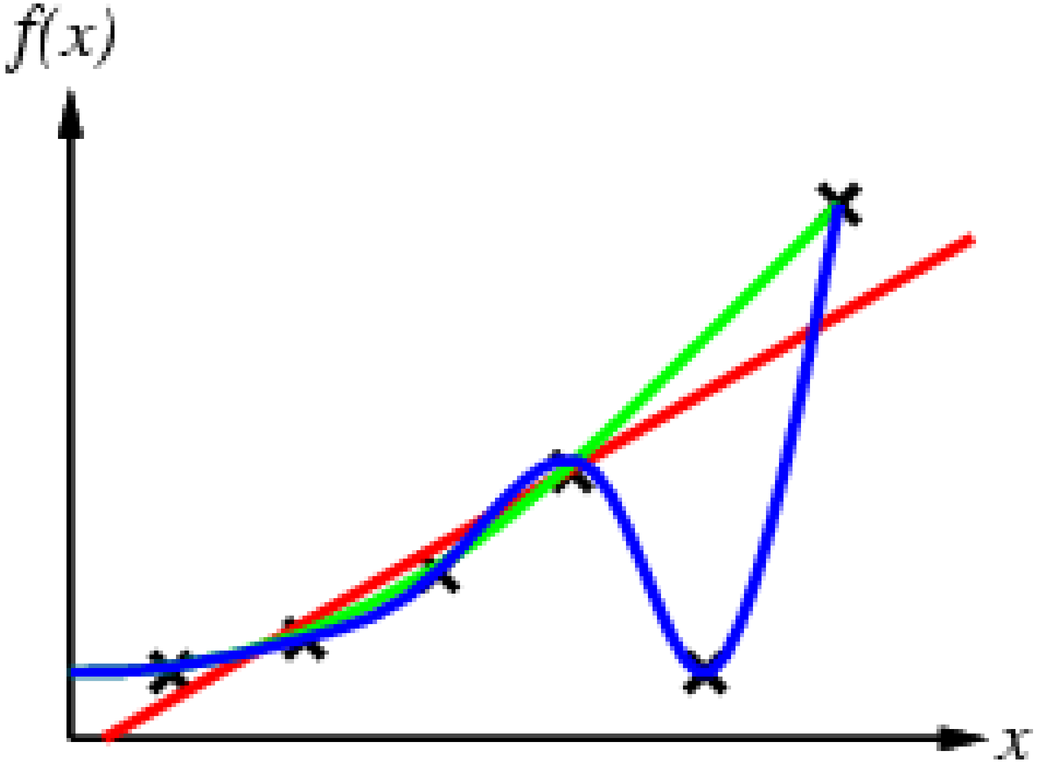
What about a quadratic function?



What about this little fella?

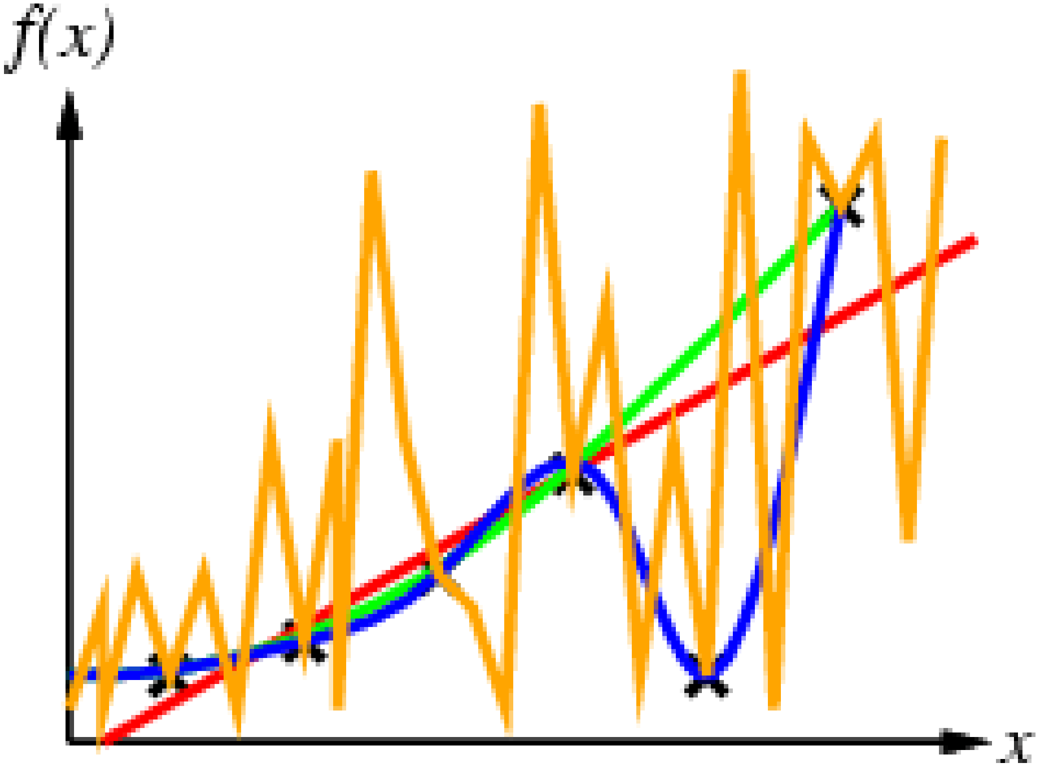
Supervised learning example

Finally, a function that satisfies all!
(consistent function)

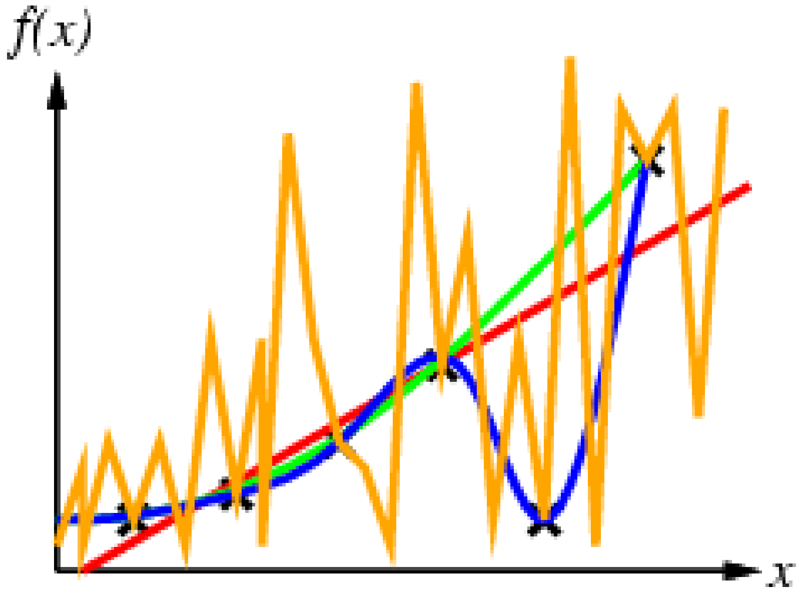


Supervised learning example

But so does this one…



Ockham's Razor Principle



Prefer the simplest hypothesis consistent with data

- Related to KISS principle ("keep it simple stupid")
- *Smooth* blue function preferable over wiggly yellow one
- If noise known to exist in data, even linear might be better (the lowest x might be due to noise)

Types of Supervised Learning

- Classification: Output is discrete (e.g., Yes/No, Class 1 or Class 2 or Class 3, etc.)
  - Decision trees
  - K-nearest neighbor
  - Linear Classifiers
  - Support Vector Machines (SVMs)
  - Cross validation
- Regression: Output is continuous
  - Linear regression and Neural networks
    - Backpropagation learning algorithm

Goal: Learn the function "PlayTennis?" from example data

	Input Attributes			*Output*
Day	Outlook	Humid	Wind	**PlayTennis**? "yes" (y) or "no" (n)
d1	s	h	w	n
d2	s	h	s	n
d3	o	h	w	y
d4	r	h	w	y
d5	r	n	w	y
d6	r	n	s	y
d7	o	n	s	y
d8	s	h	w	n
d9	s	n	w	y
d10	r	n	w	y
d11	s	n	s	y
d12	o	h	s	y
d13	o	n	w	y
d14	r	h	s	n

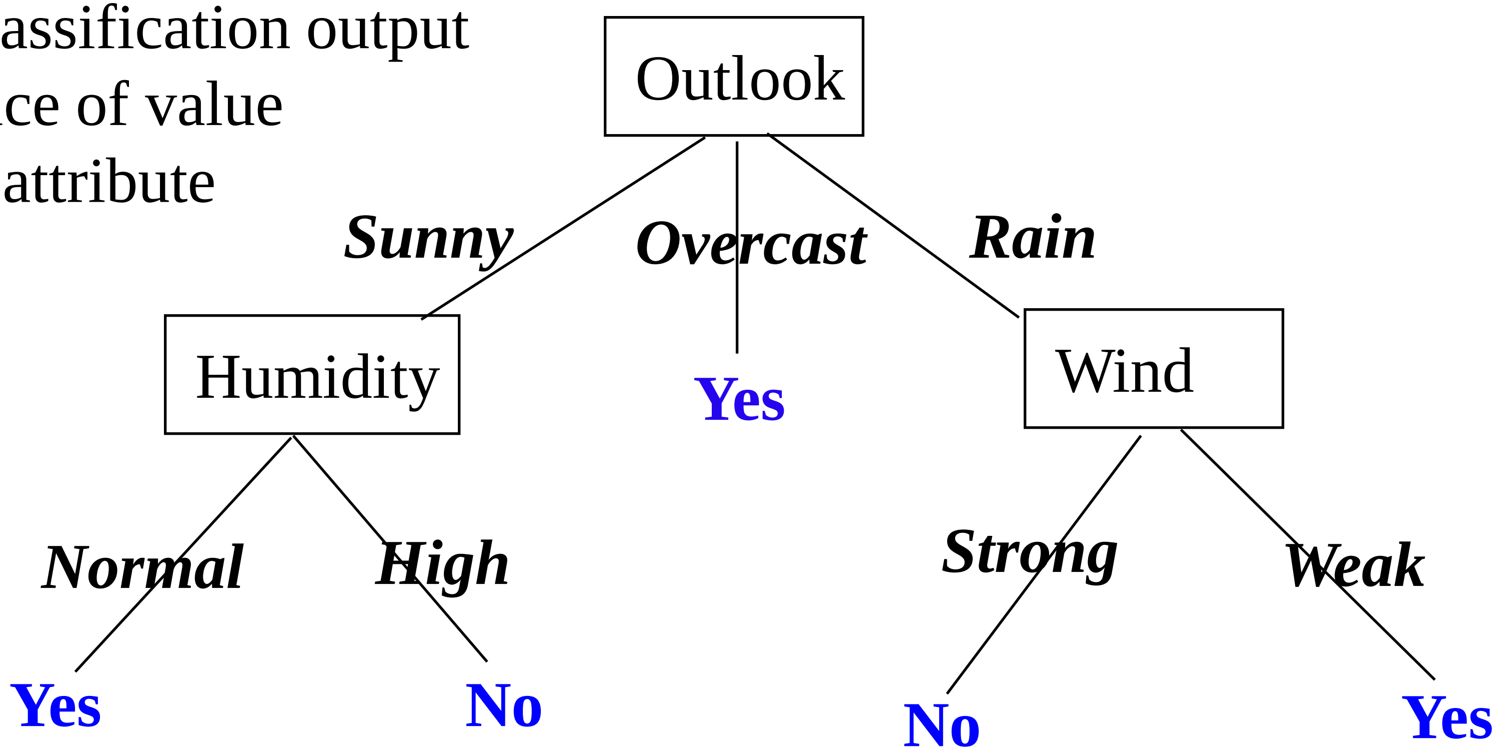
- Outlook = sunny (s), overcast (o), or rain (r)
- Humidity = high (h), or normal (n)
- Wind = weak (w) or strong (s)

A Decision Tree for the Same Data

Decision Tree for “PlayTennis?”

Leaves = classification output

Arcs = choice of value
for parent attribute



Decision tree equivalent to logical statment in disjunctive normal form

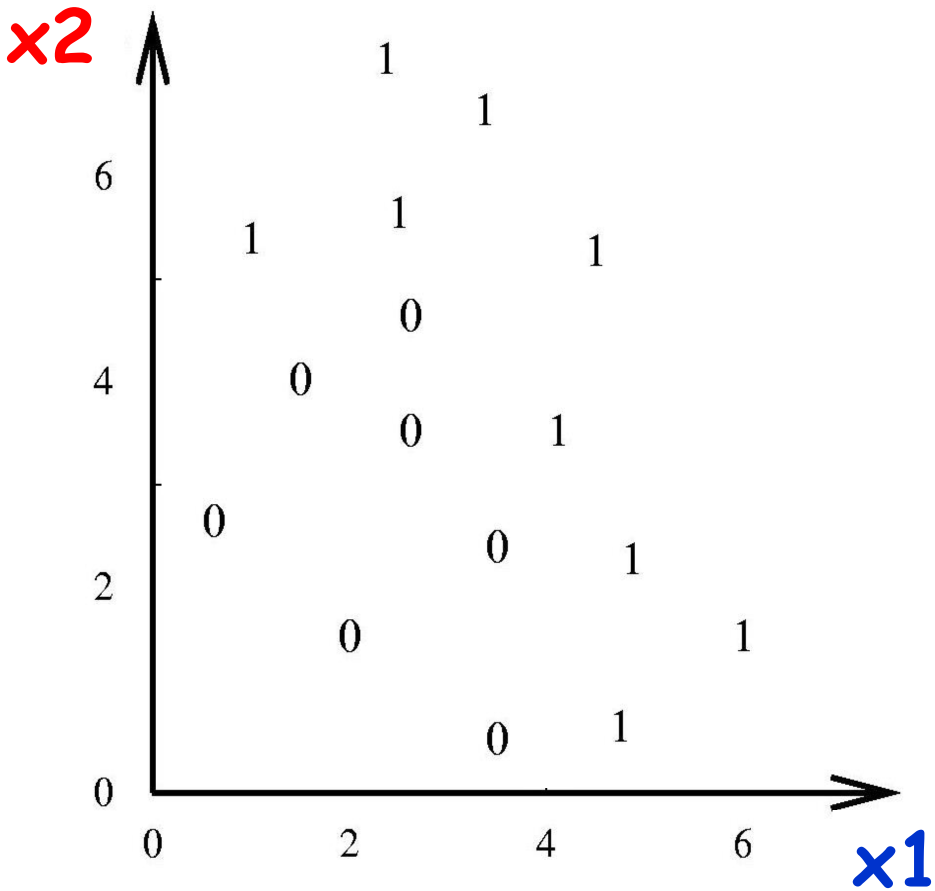
$\text{PlayTennis} \Leftrightarrow (\text{Sunny} \wedge \text{Normal}) \vee \text{Overcast} \vee (\text{Rain} \wedge \text{Weak})$

Decision Trees

- **Input:** Set of **attributes** describing an object or situation
- **Output:** Predicted output value for the input
- Decision tree is consistent if it produces the correct output on all training examples
- Input and output can be **discrete or continuous**

Example: Decision Tree for Continuous Values

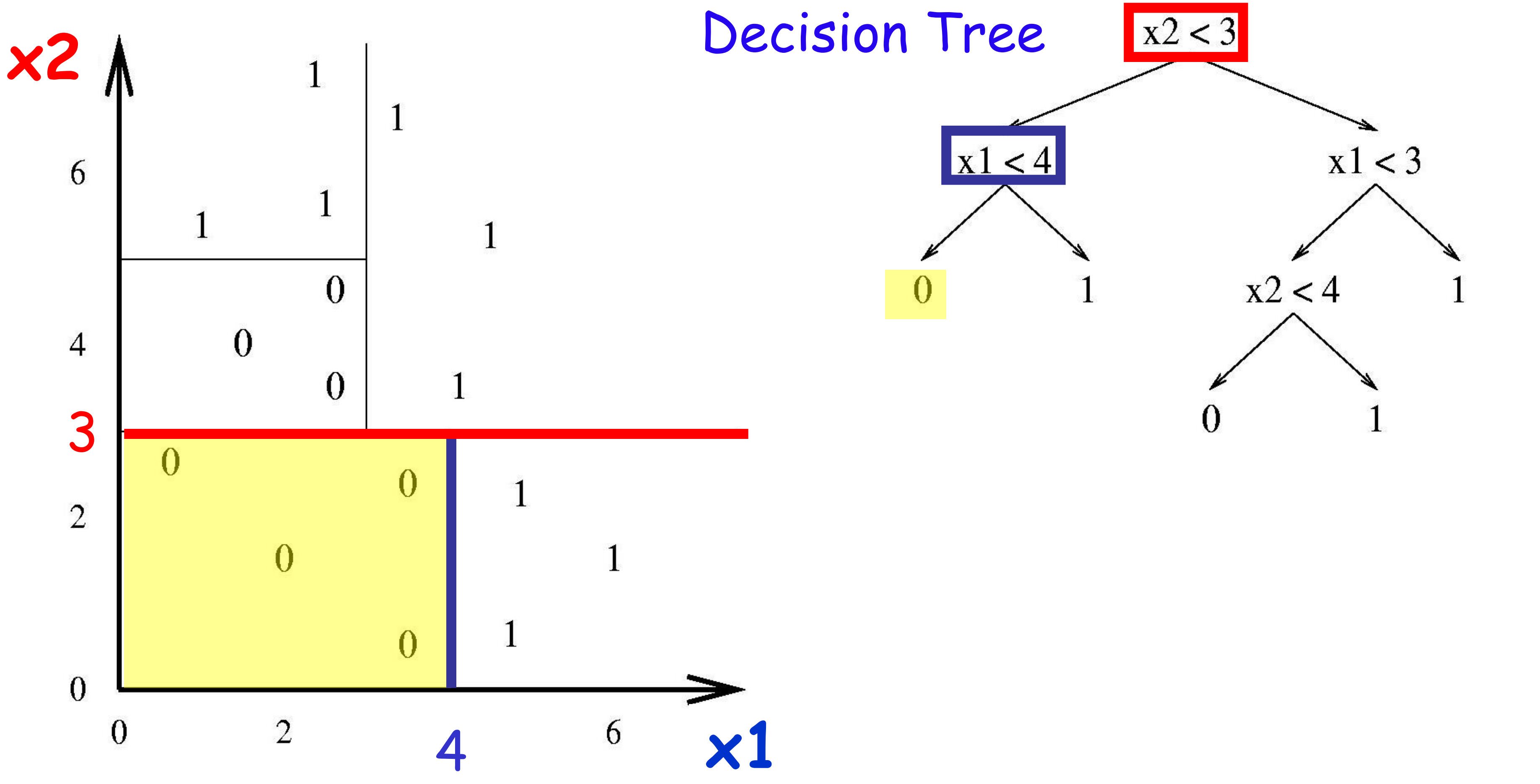
Input: Continuous-valued attributes (x1,x2)
Output: 0 or 1



How do we branch on attribute values x1 and x2 to partition the space and generate correct outputs?

Example: Classification of Continuous Valued Inputs

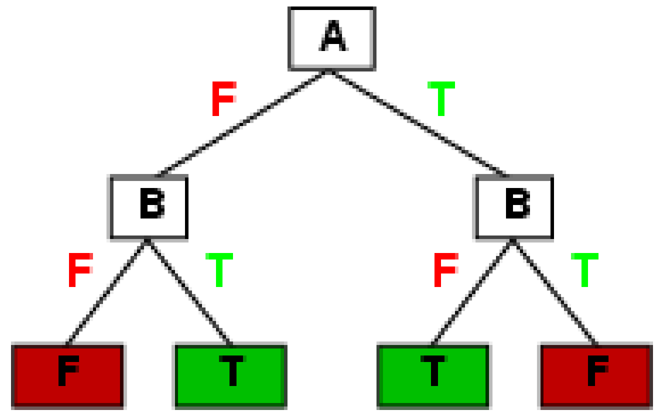
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.



Expressiveness of Decision Trees

- Decision trees can express any function of the input attributes.
- E.g., Boolean functions, truth table row = path to leaf:

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



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example
  - But most likely won't generalize to new examples
- Prefer to find more compact decision trees

Learning Decision Trees

- Example: When should I wait for a table at a restaurant?

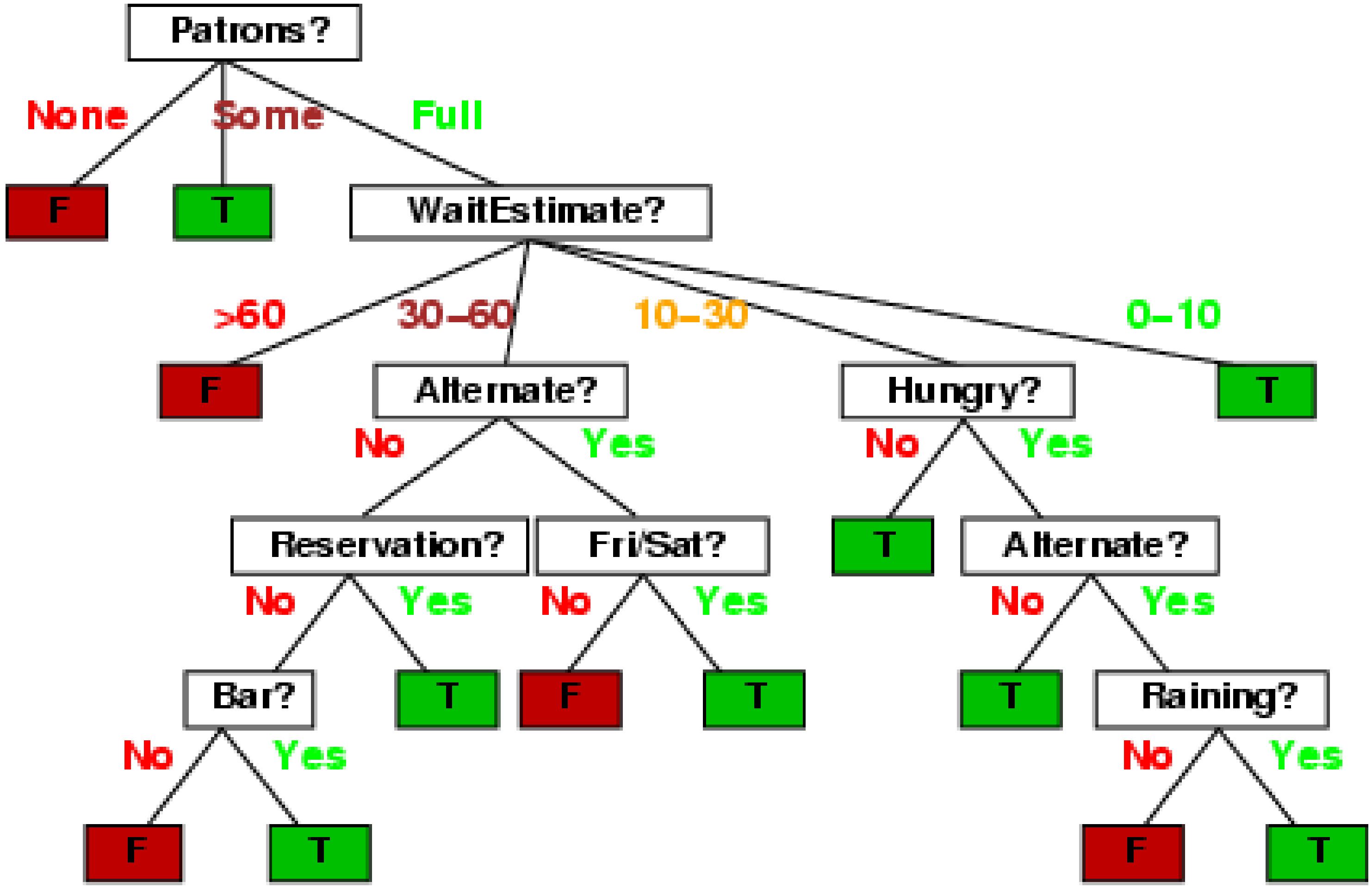
Learning Decision Trees

- Example: When should I wait for a table at a restaurant?

- Attributes (features) relevant to *Wait?* decision:
  1. Alternate: is there an alternative restaurant nearby?
  2. Bar: is there a comfortable bar area to wait in?
  3. Fri/Sat: is today Friday or Saturday?
  4. Hungry: are we hungry?
  5. Patrons: number of people in the restaurant (None, Some, Full)
  6. Price: price range ($, $$, $$$)
  7. Raining: is it raining outside?
  8. Reservation: have we made a reservation?
  9. Type: kind of restaurant (French, Italian, Thai, Burger)
  10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

A “personal” decision tree

- A decision tree for *Wait?* based on personal “rules of thumb”:



Input Data for Learning

- Past examples when I did/did not wait for a table:

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

Next Time

- Learning Decision Trees from Data
- Preventing Overfitting and Generalization
  - Cross-Validation
- To Do:
  - Project 4
  - Read Chapter 18