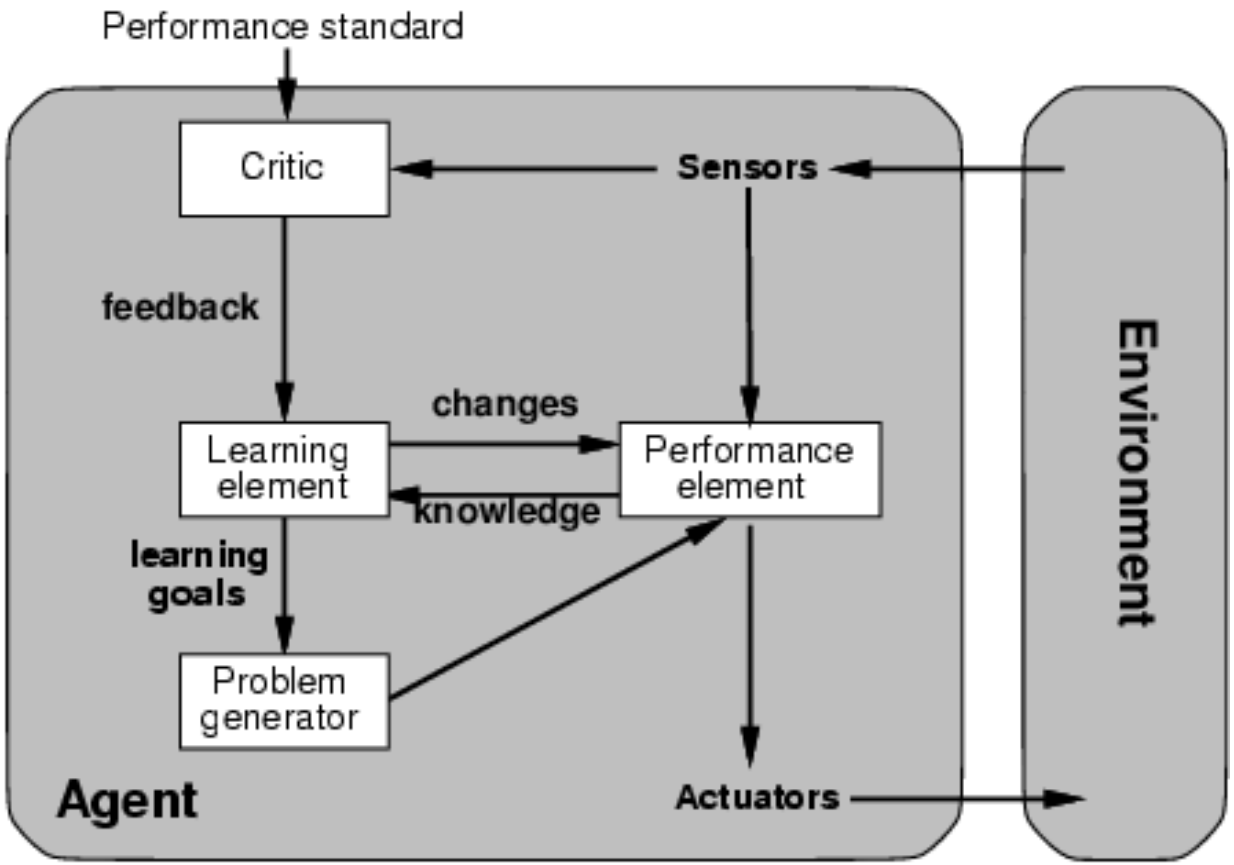
Machine Learning



Recap of last class

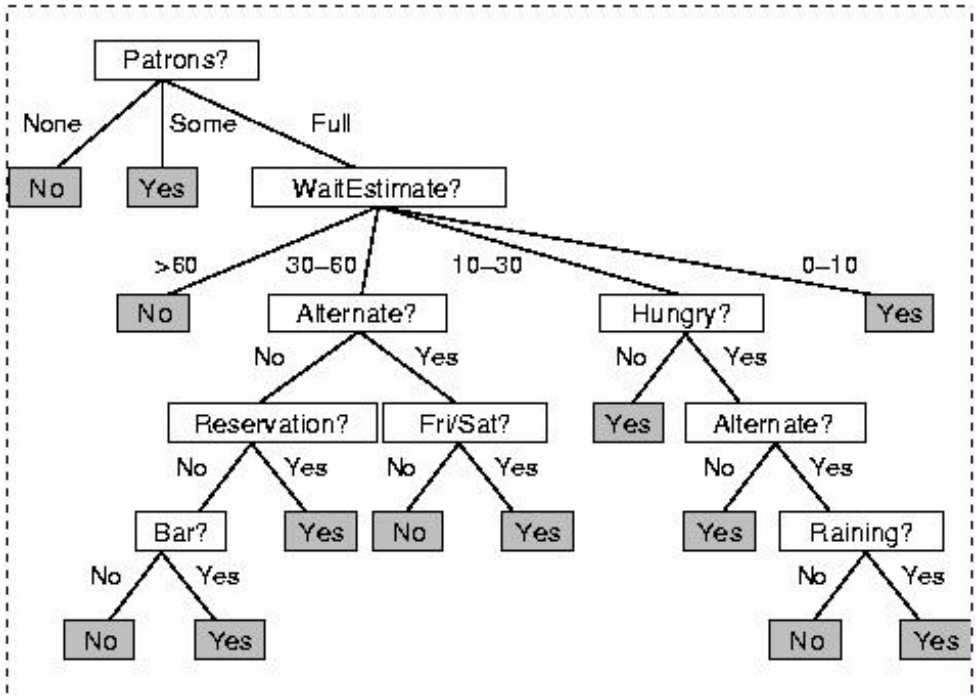
- intro to machine learning concepts
- categories of ML techniques:
  - clustering
  - data reduction
  - classification
  - regression
- different types of learning:
  - supervised, reinforcement, and unsupervised
- how to avoid overlearning

Today's Agenda

- Decision Tree Learning
- Overfitting
- Ensemble Learning
- Boosting

Simple Supervised Learning:
Decision Trees

- examples (training set) described by:
  - input: the values of attributes
  - output: the classification (yes/no)
- can represent any Boolean function



Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	$$$	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	$	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	$$$	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	$	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	$	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$$$	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	$	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Burger*	*30–60*	*Yes*

Inducing Decision Trees

- option #1: build full look-up table
  - one path for each training example
  - going down path tests each attribute in turn
  - leaf assigned the classification of example
- problems:
  - does not learn patterns
  - size is 2^n—hardly the most concise description
  - other problems -- we'll see this soon

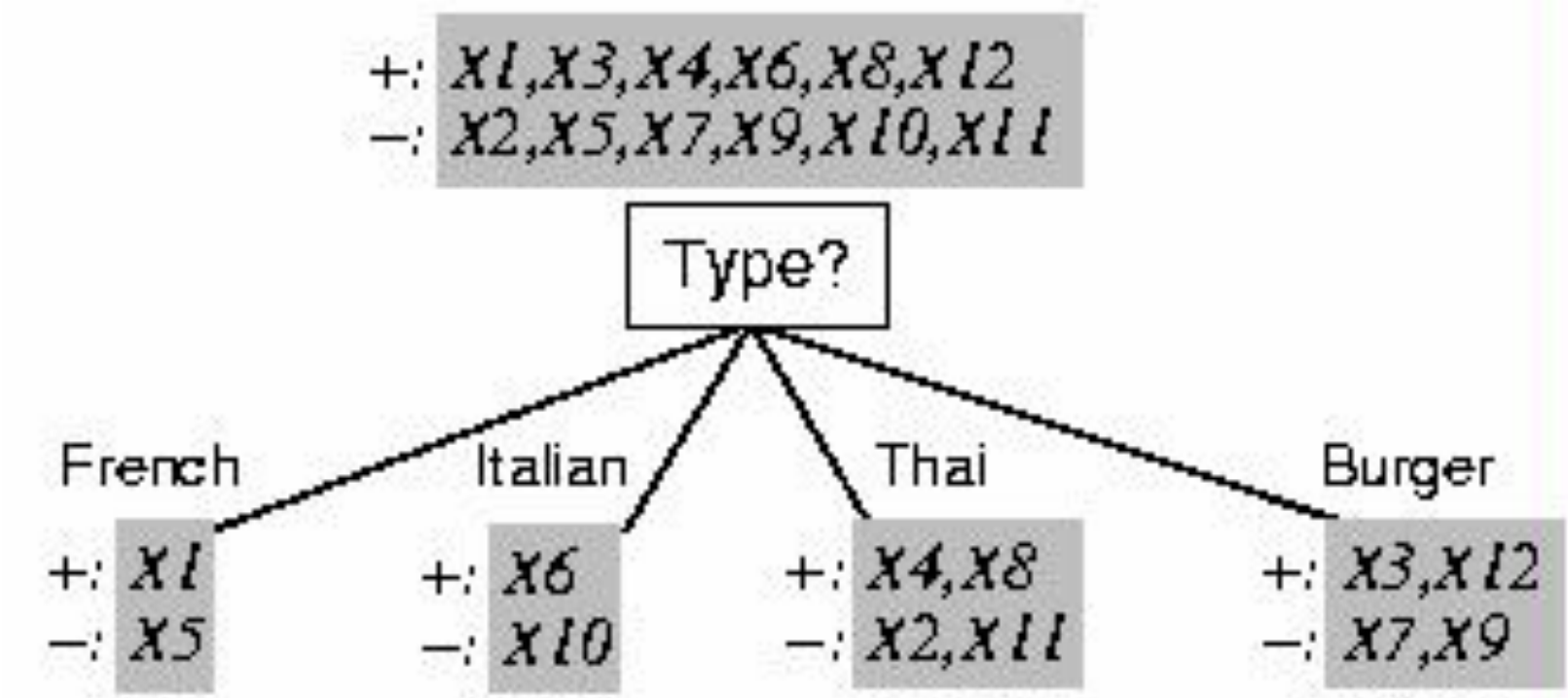
Fixing the 2^n problem

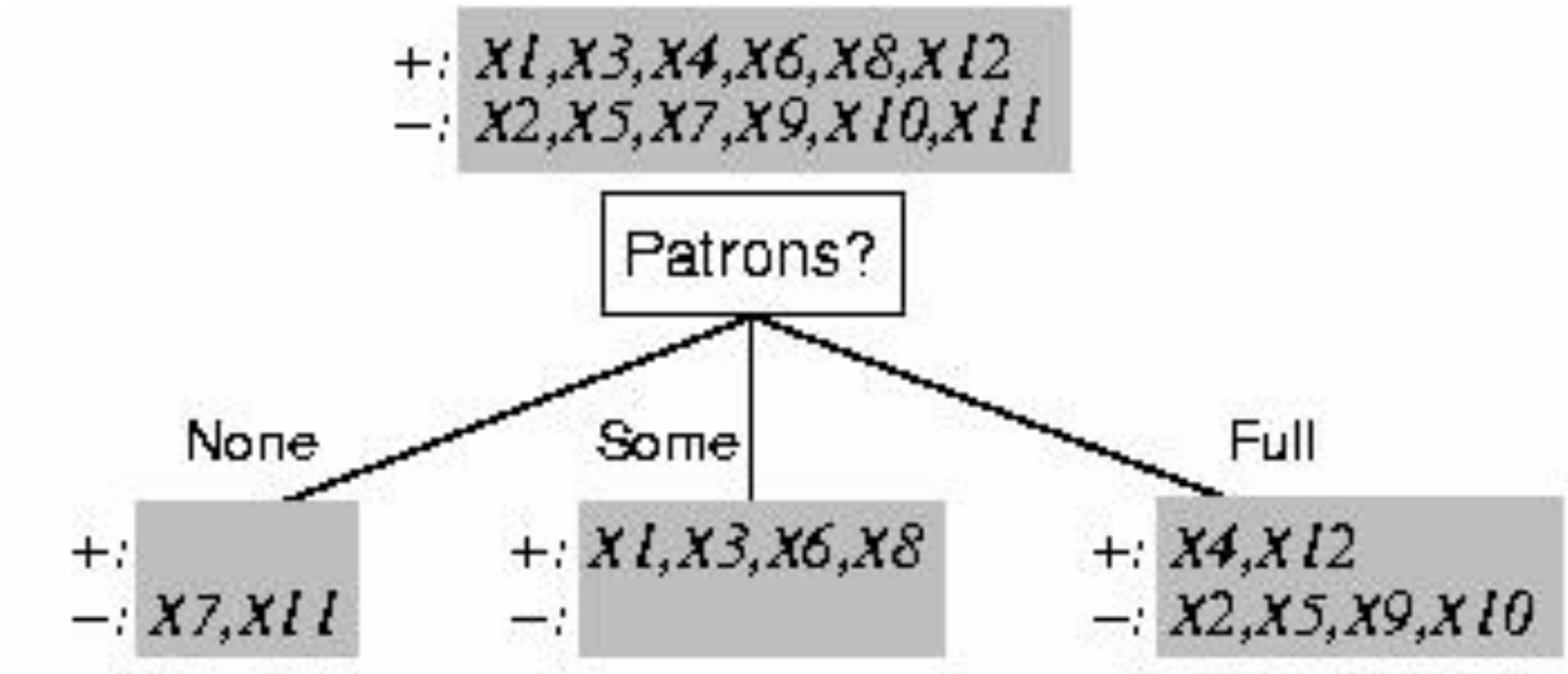
- form more compact representation
  - look for redundancy of attributes
  - but not always possible, e.g.,
    - parity function
    - majority function

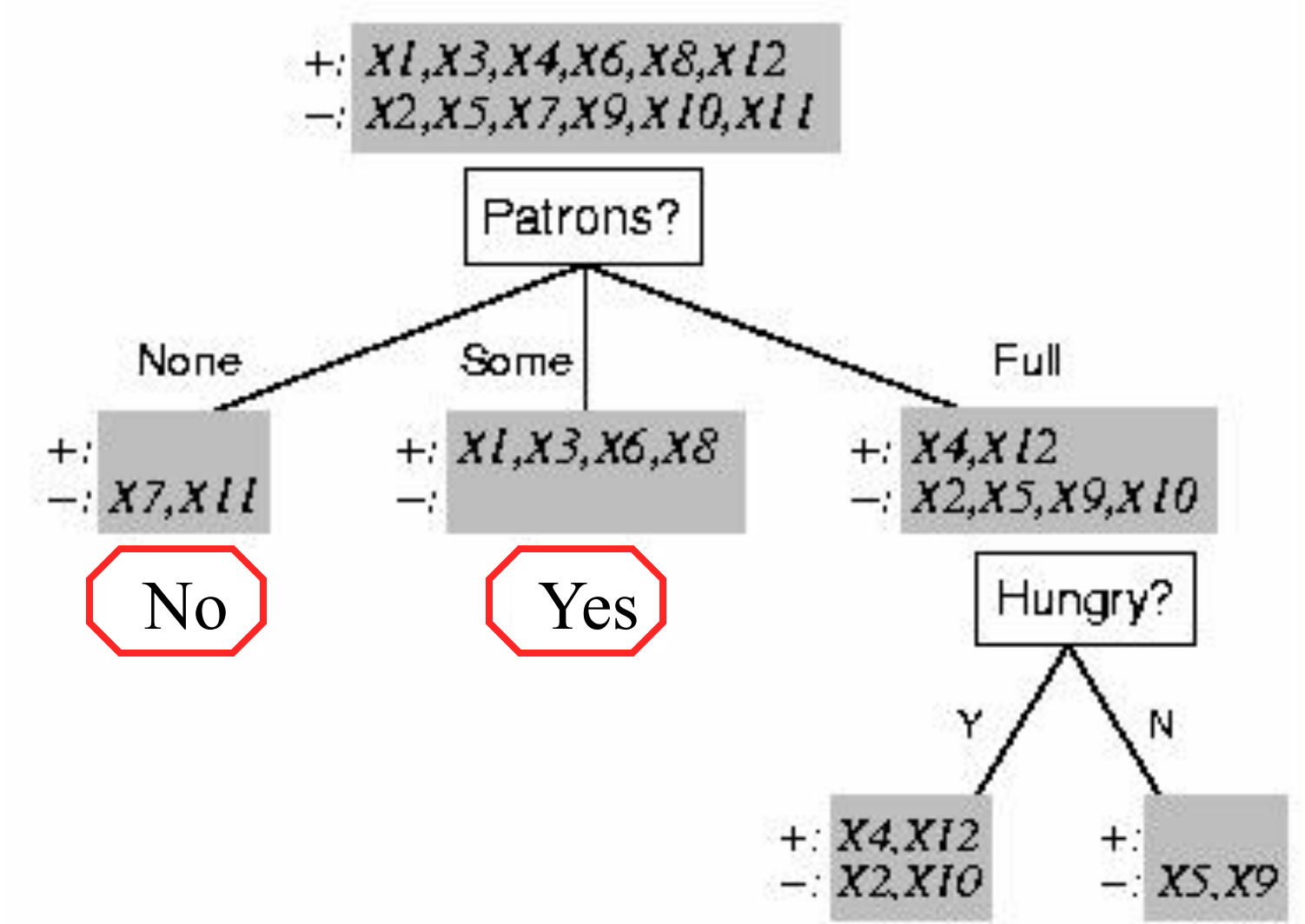
Recursive Decision Tree Induction

- **Case 1:** if we have some +, some - examples, split based on *most important* attribute
  - defined by information theoretic measures
- **Case 2:** if remaining examples all + (or -) then done
  - can answer yes or no
- **Case 3:** if there is some value for a particular attribute but no examples available to classify it
  - use default value (majority classification) from node's parent
- **Case 4:** if no attributes left, but we still have some +, some -, examples, problem:
  - data noise, insufficient information, or nondeterministic

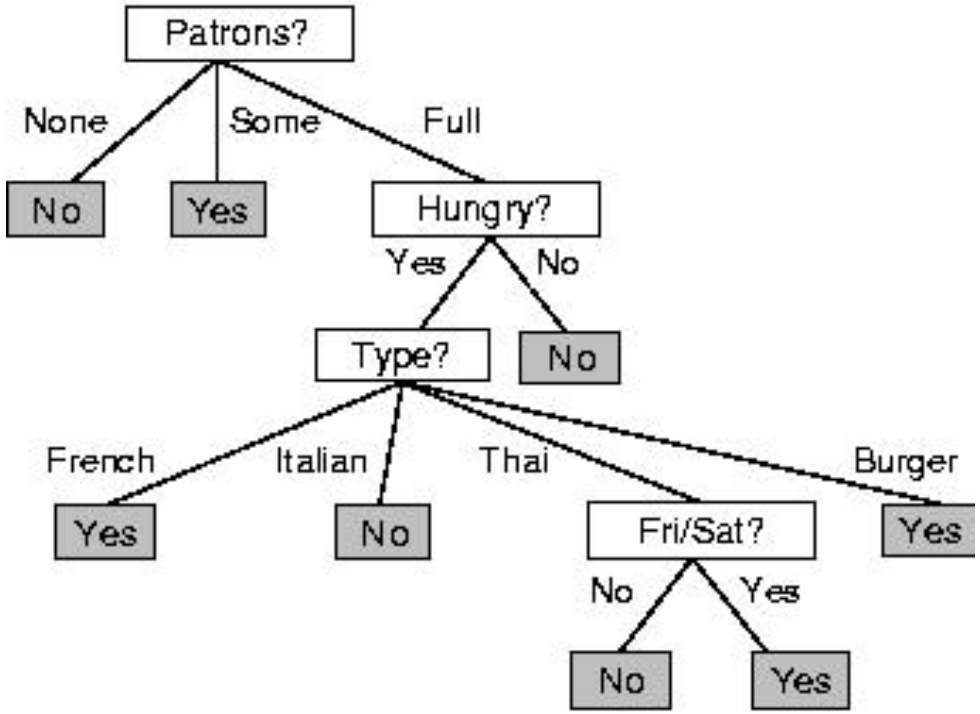
Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	$$$	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	$	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	$$$	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	$	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	$	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$$$	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	$	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Burger*	*30–60*	*Yes*

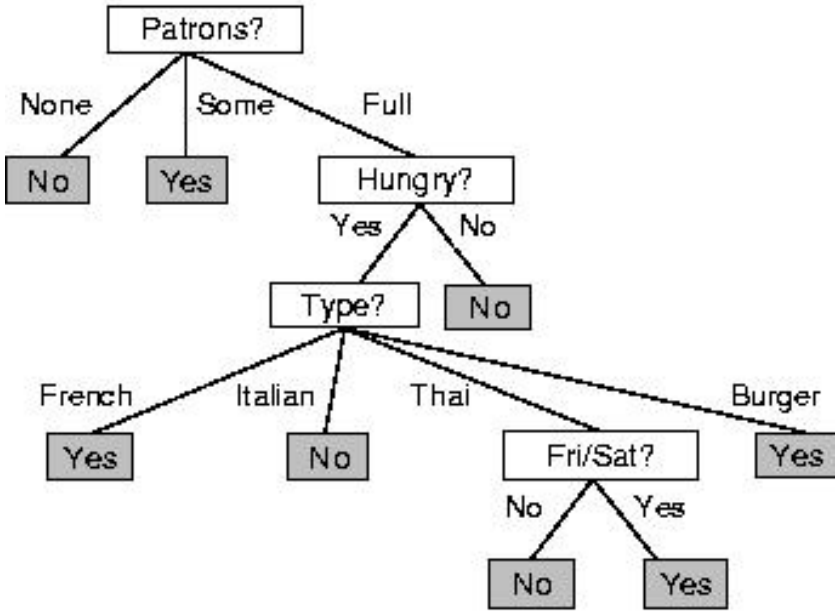

+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Type?
French
Italian
Thai
Burger
+: X1
−: X5
+: X6
−: X10
+: X4,X8
−: X2,X11
+: X3,X12
−: X7,X9


+: x1,x3,x4,x6,x8,x12
−: x2,x5,x7,x9,x10,x11
Patrons?
None
Some
Full
+:
−: x7,x11
+: x1,x3,x6,x8
−:
+: x4,x12
−: x2,x5,x9,x10

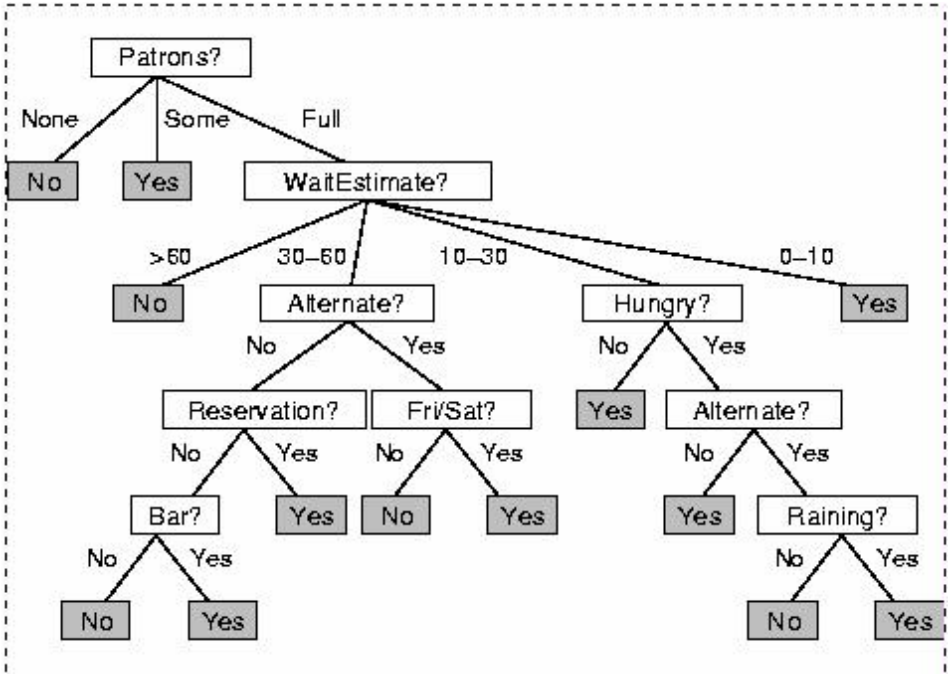

+: X1,X3,X4,X6,X8,X12
–: X2,X5,X7,X9,X10,X11
Patrons?
None
Some
Full
+:
–: X7,X11
+: X1,X3,X6,X8
–:
+: X4,X12
–: X2,X5,X9,X10
No
Yes
Hungry?
Y
N
+: X4,X12
–: X2,X10
+:
–: X5,X9

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	$$$	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	$	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	$$$	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	$	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	$	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$$$	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	$	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Burger*	*30–60*	*Yes*





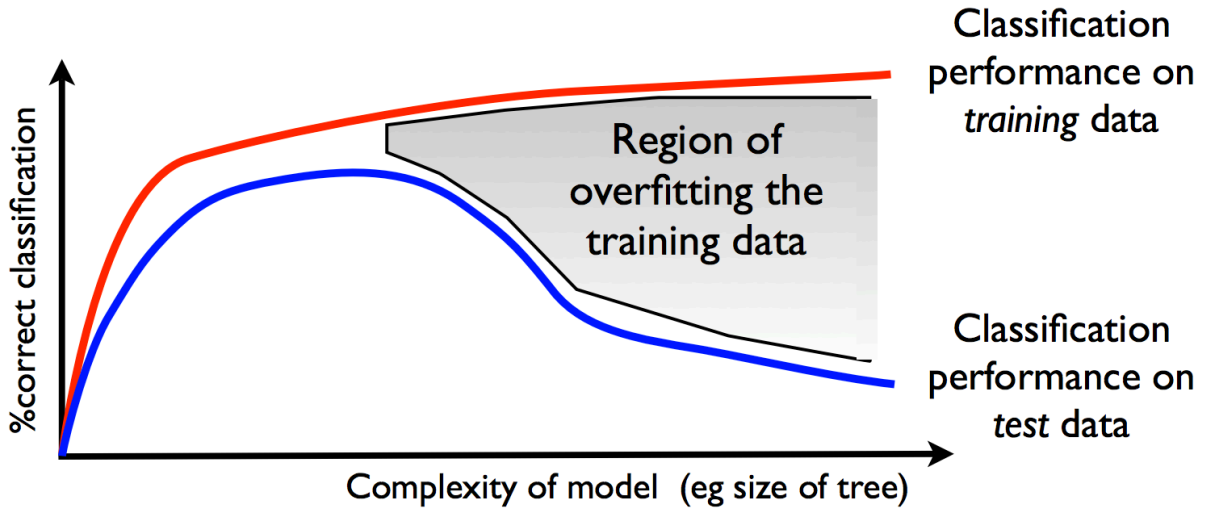
Induced tree



Original tree

- induced decision tree looks nothing like the original
  - it's much smaller, but is it wrong?
- doesn't necessarily learn original function
  - ignores *Raining* and *Reservation*
- what happens if *restaurant is full*, *wait = 5 minutes*, *hungry=no*?

Overfitting



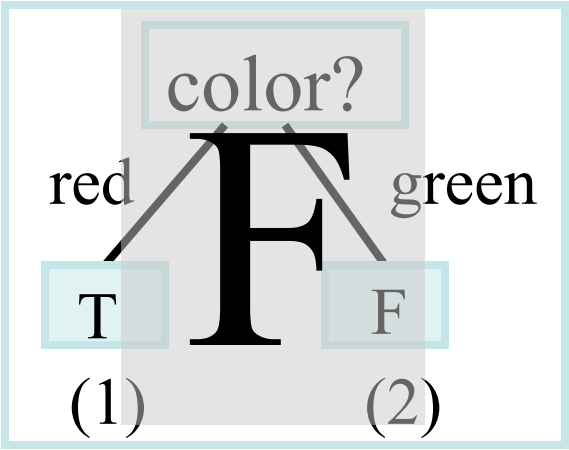
- Problem: might be fitting noise rather than data
- How to avoid overfitting in decision trees?
  - Pruning: avoid recursive split on irrelevant attributes
  - But how do we determine irrelevance?

Recursive Decision Tree Induction

- **Case 1:** if we have some +, some - examples, split based on *most important* attribute
  - defined by information theoretic measures
- **Case 2:** if remaining examples all + (or -) then done
  - can answer yes or no
- **Case 3:** if there is some value for a particular attribute but no examples available to classify it
  - use default value (majority classification) from node's parent
- **Case 4:** if no attributes left, but we still have some +, some -, examples, problem:
  - data noise, insufficient information, or nondeterministic

Cross-validation

- how well does the prediction fare on unseen data?
- use a set of independent data to “validate” the tree
- what happens if validation set gives errors?
- Prune: replace subtree by a single (approximate) decision node (“most popular decision”)



TrainingSet Results

color	decision	
red	T	√
green	F	√
green	F	√

Validation Set Results

color	decision	
green	F	√
green	T	x
red	F	x

Statistical Significance Test

- take a sample of size *v*, consisting of *p* positive and *n* negative examples
- divide the sample into subsets based on classification of the attribute

color	decision
green	F
green	F
green	T
green	F

$p_i=1, n_i=3$

color	decision
red	T
red	F

$p_i=1, n_i=1$

- for each subset, let p_i and n_i be the number of positive and negative examples

Statistical Significance Test

color	decision
green	F
green	F
green	T
green	F

$p_i=1, n_i=3$

color	decision
red	T
red	F

$p_i=1, n_i=1$

$p=2, n=4$

- Calculate expected # of positive and negative examples, assuming attribute is irrelevant

$$\hat{p}_i = p \times \frac{p_i + n_i}{p + n} \qquad \hat{n}_i = n \times \frac{p_i + n_i}{p + n}$$

Statistical Significance Test

- now calculate total deviation:

$$D = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

- D measures how much the split deviates from what we would expect from random data
- if attribute is “irrelevant”, i.e., the total deviation from the null hypothesis is statistically insignificant,can prune it from tree
- is D small in this case?

Real-world decision tree algorithms

- handle continuous data (e.g., temperature)
- avoid overfitting the data
- deal with:
  - weighted attribute costs
  - data with missing attribute values
- perform pruning

Example: Golf Decision

Outlook	Temperature	Humidity	Windy	Decision
sunny	85	85	false	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

Approach

- start by considering binary variables (e.g., outlook, windy)
- when they no longer help provide unambiguous classification to all remaining examples:
  - consider continuous variables
  - choose a useful split point in their range (e.g. temperature > 83)

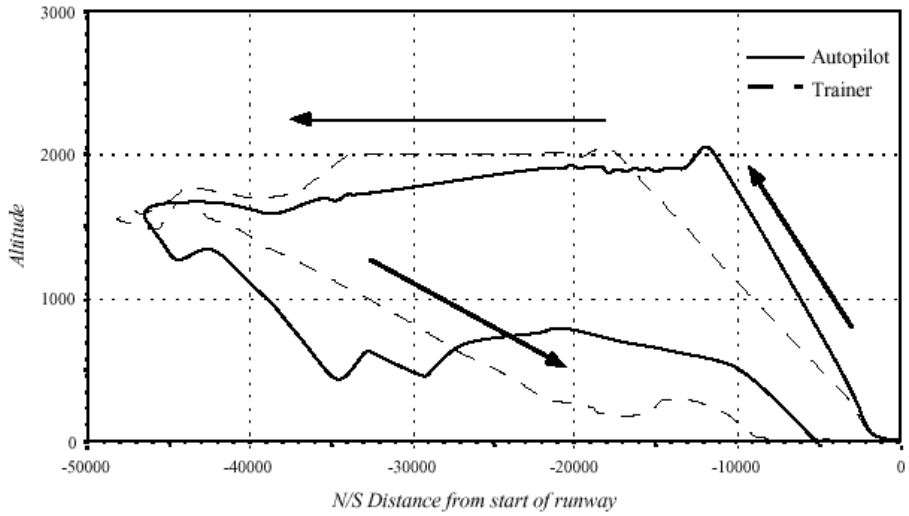
C4.5 Decision Tree Output

```
Decision Tree:
outlook = overcast: Play (4.0)
outlook = sunny:
      |   humidity <= 75 : Play (2.0)
      |   humidity > 75 : Don't Play (3.0)
outlook = rain:
      |   windy = true: Don't Play (2.0)
      |   windy = false: Play (3.0)

Evaluation on training data (14 items):
        Before Pruning           After Pruning
       ----------------   ---------------------------
       Size      Errors   Size      Errors   Estimate
         8    0( 0.0%)      8    0( 0.0%)    (38.5%)   <<
```

Sammut: Learning to Fly

- take off and fly to altitude of 2000 feet
- level out, fly to distance of 32000 feet
- turn right to 330^0
- at 42,000 feet, turn back to runway
- line up on the runway
- descend on the runway, keeping in line
- land



Issues Faced

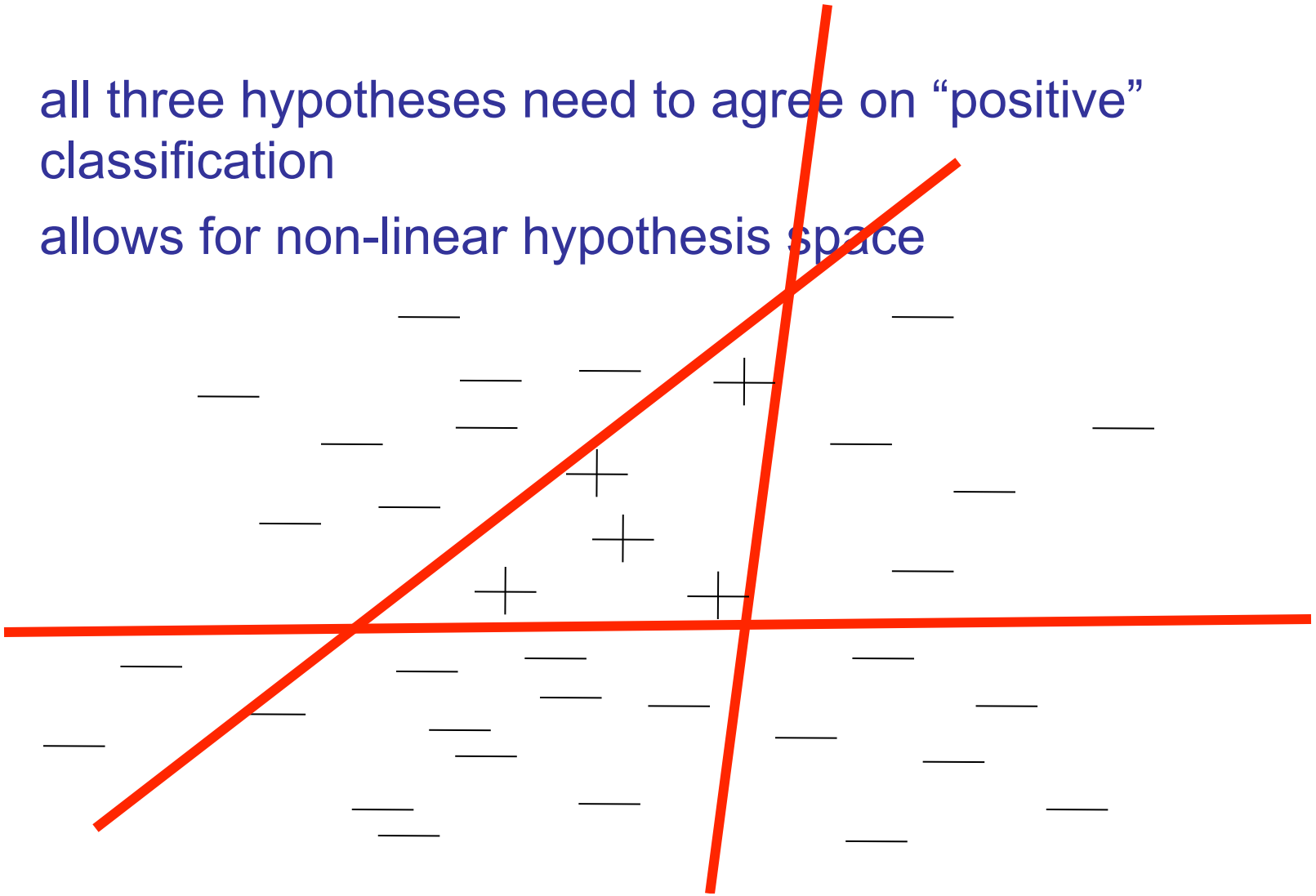
- pruning
- branching
- dealing with real-world issues
  - noise in data
  - causality
  - delay between sensing and reaction
  - different strategies accomplish same goal

Ensemble Learning

- We've seen how to learn to make predictions from a single hypothesis, e.g., decision trees
- What if we generated an ensemble of hypotheses and used their combination to make predictions?
- If errors made by each hypothesis are independent, the probability that a majority of them will make the same error is very small

Three linear hypotheses in ensemble

- all three hypotheses need to agree on “positive” classification
- allows for non-linear hypothesis space



Boosting

- incrementally build an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified

Boosting

- Weighted training set: each example has a weight $w_{ij} \geq 0$ representing its importance during learning
- Start learning:
  - Use $w_{ij}=1$ for all examples in training set

  $\Rightarrow$ hypothesis h_1
- Next iteration:
  - Increase w_{ij} for all misclassified examples in h_1
  - Decrease w_i for others; learn again

  $\Rightarrow$hypothesis h_2

  …
- Repeat until *M* hypotheses generated

How Boosting Works

height of rectangle
indicates weight