

# Machine Learning

CS 786

University of Waterloo

Lecture 4: May 10, 2012

## What is Machine Learning?

- Definition:
  - A computer program is said to **learn** from **experience**  $E$  with respect to some class of **tasks**  $T$  and **performance measure**  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .

[T Mitchell, 1997]

## Examples

- **Backgammon (reinforcement learning):**
  - T: playing backgammon
  - P: percent of games won against an opponent
  - E: playing practice games against itself
- **Handwriting recognition (supervised learning):**
  - T: recognize handwritten words within images
  - P: percent of words correctly recognized
  - E: database of handwritten words with given classifications
- **Customer profiling (unsupervised learning):**
  - T: cluster customers based on transaction patterns
  - P: homogeneity of clusters
  - E: database of customer transactions

CS786 Lecture Slides (c) 2012 P. Poupart

3

## Inductive learning (aka concept learning)

- **Induction:**
  - Given a **training set** of **examples** of the form  $(x, f(x))$ 
    - $x$  is the input,  $f(x)$  is the output
  - Return a function  $h$  that approximates  $f$ 
    - $h$  is called the **hypothesis**

CS786 Lecture Slides (c) 2012 P. Poupart

4

# Classification

- Training set:

STAT231 statistics	CS341 algorithms	CS350 OS	CS485 ML	CS486 AI	CS786 PI+ML
A	A	B	A	A	A
A	B	B	B	A	A
B	B	B	B	B	B
B	A	B	A	A	A

$x$

$f(x)$

- Possible hypotheses:

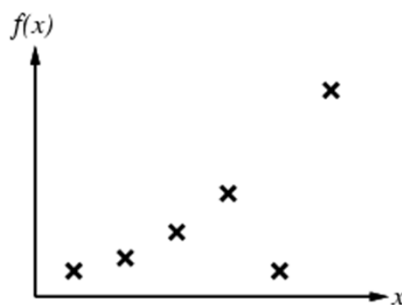
- $h_1: CS485=A \rightarrow CS786=A$
- $h_2: CS485=A \vee STAT231=A \rightarrow CS786=A$

CS786 Lecture Slides (c) 2012 P. Poupart

5

# Regression

- Find function  $h$  that fits  $f$  at instances  $x$

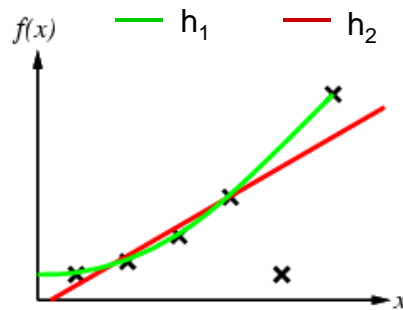


CS786 Lecture Slides (c) 2012 P. Poupart

6

# Regression

- Find function  $h$  that fits  $f$  at instances  $x$



CS786 Lecture Slides (c) 2012 P. Poupart

7

# Hypothesis Space

- Hypothesis space  $H$ 
  - Set of all hypotheses  $h$  that the learner may consider
  - Learning is a search through hypothesis space
- Objective:
  - Find hypothesis that agrees with training examples
  - But what about unseen examples?

CS786 Lecture Slides (c) 2012 P. Poupart

8

## Generalization

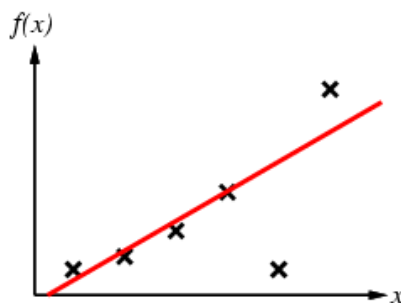
- A good hypothesis will **generalize well** (i.e. predict unseen examples correctly)
- Usually...
  - Any hypothesis  $h$  found to approximate the target function  $f$  well over a sufficiently large set of training examples will also approximate the target function well over any unobserved examples

CS786 Lecture Slides (c) 2012 P. Poupart

9

## Inductive learning

- Construct/adjust  $h$  to agree with  $f$  on training set
- ( $h$  is **consistent** if it agrees with  $f$  on all examples)
- E.g., curve fitting:

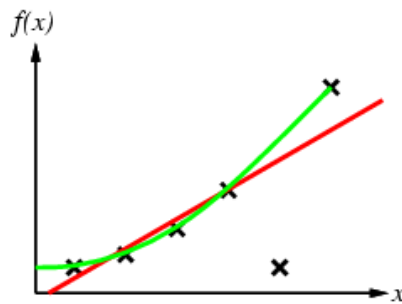


CS786 Lecture Slides (c) 2012 P. Poupart

10

# Inductive learning

- Construct/adjust  $h$  to agree with  $f$  on training set
- ( $h$  is **consistent** if it agrees with  $f$  on all examples)
- E.g., curve fitting:

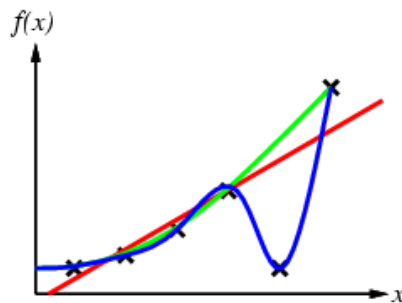


CS786 Lecture Slides (c) 2012 P. Poupart

11

# Inductive learning

- Construct/adjust  $h$  to agree with  $f$  on training set
- ( $h$  is **consistent** if it agrees with  $f$  on all examples)
- E.g., curve fitting:

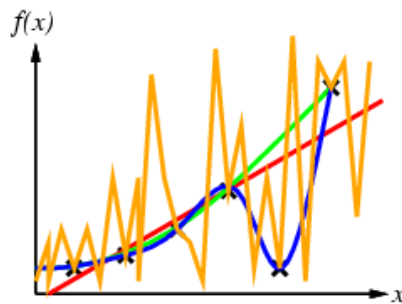


CS786 Lecture Slides (c) 2012 P. Poupart

12

# Inductive learning

- Construct/adjust  $h$  to agree with  $f$  on training set
- ( $h$  is **consistent** if it agrees with  $f$  on all examples)
- E.g., curve fitting:

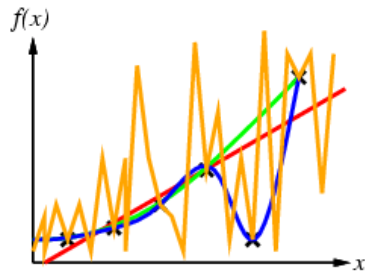


CS786 Lecture Slides (c) 2012 P. Poupart

13

# Inductive learning

- Construct/adjust  $h$  to agree with  $f$  on training set
- ( $h$  is **consistent** if it agrees with  $f$  on all examples)
- E.g., curve fitting:



- **Ockham's razor:** prefer the simplest hypothesis consistent with data

CS786 Lecture Slides (c) 2012 P. Poupart

14

## Inductive learning

- Finding a **consistent** hypothesis depends on the hypothesis space
  - For example, it is not possible to learn exactly  $f(x)=ax+b+x\sin(x)$  when  $H$ =space of polynomials of finite degree
- A learning problem is **realizable** if the hypothesis space contains the true function, otherwise it is **unrealizable**
  - Difficult to determine whether a learning problem is realizable since the true function is not known

CS786 Lecture Slides (c) 2012 P. Poupart

15

## Inductive learning

- It is possible to use a very large hypothesis space
  - For example,  $H$ =class of all Turing machines
- But there is a **tradeoff** between **expressiveness** of a hypothesis class and **complexity** of finding simple, consistent hypothesis within the space
  - Fitting straight lines is easy, fitting high degree polynomials is hard, fitting Turing machines is very hard!

CS786 Lecture Slides (c) 2012 P. Poupart

16



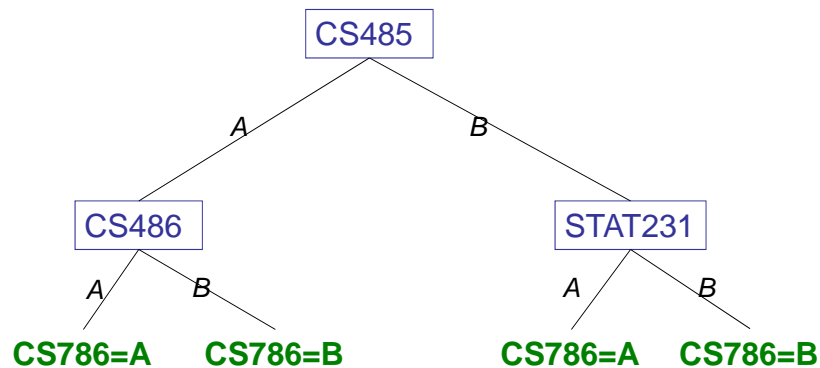
# Decision trees

- Decision tree classification
  - Nodes: labeled with attributes
  - Edges: labeled with attribute values
  - Leaves: labeled with classes
- Classify an instance by starting at the root, testing the attribute specified by the root, then moving down the branch corresponding to the value of the attribute
  - Continue until you reach a leaf
  - Return the class

CS786 Lecture Slides (c) 2012 P. Poupart

17

## Decision tree (grade prediction for CS786)



An instance  
⟨CS485=A, CS486=A, STAT231=B, CS341=B⟩

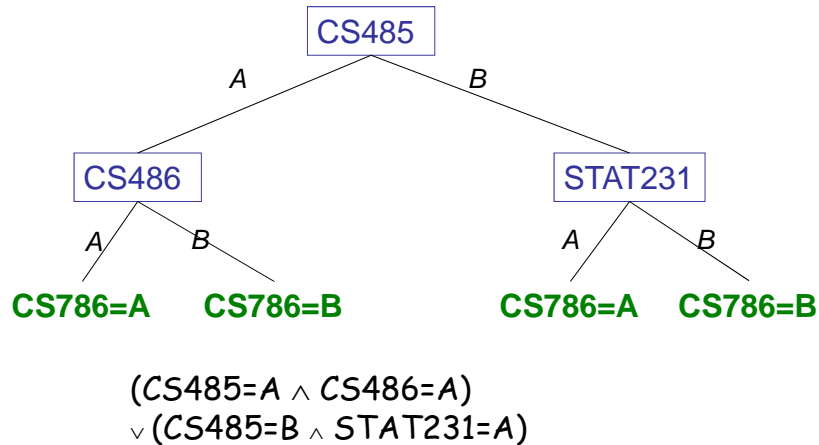
Classification: CS786=A

CS786 Lecture Slides (c) 2012 P. Poupart

18

## Decision tree representation

- Decision trees can represent disjunctions of conjunctions of constraints on attribute values



CS786 Lecture Slides (c) 2012 P. Poupart

19

## Decision tree representation

- Decision trees are fully expressive within the class of propositional languages
  - Any Boolean function can be written as a decision tree
    - Trivially by allowing each row in a truth table correspond to a path in the tree
    - Can often use small trees
    - Some functions require exponentially large trees (majority function, parity function)
  - However, there is no representation that is efficient for all functions

CS786 Lecture Slides (c) 2012 P. Poupart

20

## Inducing a decision tree

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

CS786 Lecture Slides (c) 2012 P. Poupart

21

## Decision Tree Learning

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
      examplesi ← {elements of examples with best =  $v_i$ }
      subtree ← DTL(examplesi, attributes – best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
    return tree
```

CS786 Lecture Slides (c) 2012 P. Poupart

22

## Choosing attribute tests

- The central choice is deciding which attribute to test at each node
- We want to choose an attribute that is most useful for classifying examples

CS786 Lecture Slides (c) 2012 P. Poupart

23

## Example -- Restaurant

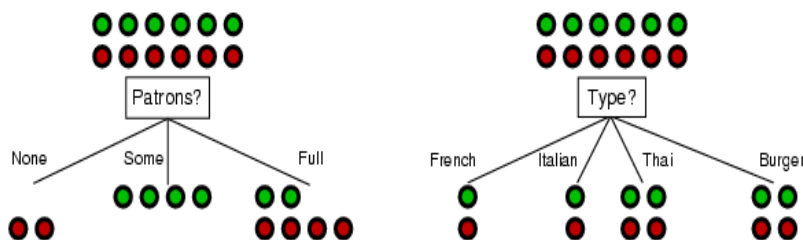
Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>Wait</i>
$X_1$	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
$X_2$	T	F	F	T	Full	\$	F	F	Thai	30-60	F
$X_3$	F	T	F	F	Some	\$	F	F	Burger	0-10	T
$X_4$	T	F	T	T	Full	\$	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

CS786 Lecture Slides (c) 2012 P. Poupart

24

## Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- Patrons?* is a better choice

CS786 Lecture Slides (c) 2012 P. Poupart

25

## Using information theory

- To implement **Choose-Attribute** in the DTL algorithm
- Measure uncertainty (Entropy):  

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$
- For a training set containing  $p$  positive examples and  $n$  negative examples:

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

CS786 Lecture Slides (c) 2012 P. Poupart

26

## Information gain

- A chosen attribute  $A$  divides the training set  $E$  into subsets  $E_1, \dots, E_v$  according to their values for  $A$ , where  $A$  has  $v$  distinct values.

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

- Information Gain (IG) or reduction in uncertainty from the attribute test:

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

- Choose the attribute with the largest IG

## Information gain

For the training set,  $p = n = 6$ ,  $I(6/12, 6/12) = 1$  bit

Consider the attributes *Patrons* and *Type* (and others too):

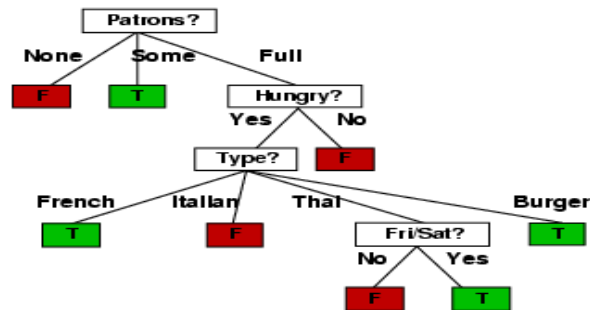
$$IG(\text{Patrons}) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .541 \text{ bits}$$

$$IG(\text{Type}) = 1 - \left[ \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

*Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

## Example

- Decision tree learned from the 12 examples:



- Substantially simpler than "true" tree---a more complex hypothesis isn't justified by small amount of data

CS786 Lecture Slides (c) 2012 P. Poupart

29

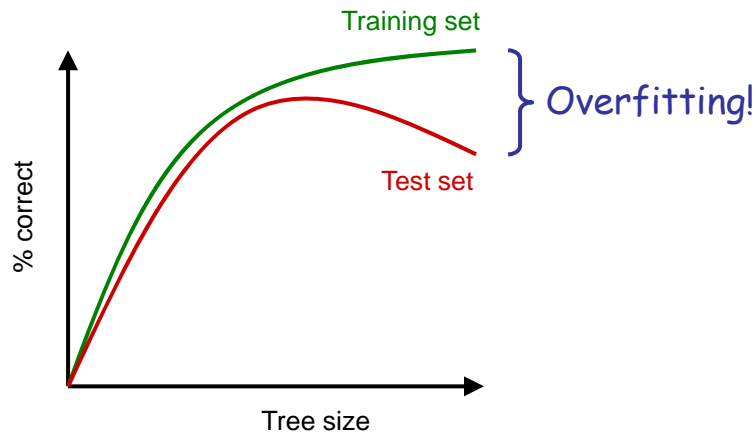
## Performance of a learning algorithm

- A learning algorithm is good if it produces a hypothesis that does a good job of predicting classifications of unseen examples
- Verify performance with a **test set**
  1. Collect a large set of examples
  2. Divide into 2 disjoint sets: training set and test set
  3. Learn hypothesis  $h$  with training set
  4. Measure percentage of correctly classified examples by  $h$  in the test set
  5. Repeat 2-4 for different randomly selected training sets of varying sizes

CS786 Lecture Slides (c) 2012 P. Poupart

30

## Learning curves



CS786 Lecture Slides (c) 2012 P. Poupart

31

## Overfitting

- Decision-tree grows until all training examples are perfectly classified
- But what if...
  - Data is noisy
  - Training set is too small to give a representative sample of the target function
- May lead to **Overfitting!**
  - Common problem with most learning algo

CS786 Lecture Slides (c) 2012 P. Poupart

32



# Overfitting

- **Definition:** Given a hypothesis space  $H$ , a hypothesis  $h \in H$  is said to overfit the training data if there exists some alternative hypothesis  $h' \in H$  such that  $h$  has smaller error than  $h'$  over the training examples but  $h'$  has smaller error than  $h$  over the entire distribution of instances
- Overfitting has been found to decrease accuracy of decision trees by 10-25%

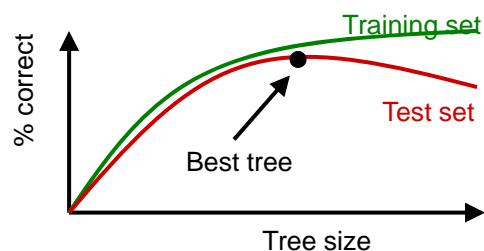
CS786 Lecture Slides (c) 2012 P. Poupart

33

# Avoiding overfitting

Two popular techniques:

1. Prune statistically irrelevant nodes
  - Measure irrelevance with  $\chi^2$  test
2. Stop growing tree when test set performance starts decreasing
  - Use cross-validation



CS786 Lecture Slides (c) 2012 P. Poupart

34

## Cross-validation

- Split data in two parts, one for training, one for testing the accuracy of a hypothesis
- K-fold cross validation means you run  $k$  experiments, each time putting aside  $1/k$  of the data to test on