Machine Learning

Inductive Learning and Decision Trees

CSE 473

Why Learning?

- Learning is essential for unknown environments
  e.g., when designer lacks omniscience
- Learning is necessary in dynamic environments
  Agent can adapt to changes in environment not foreseen at design time
- Learning is useful as a system construction method
  Expose the agent to reality rather than trying to approximate it through equations etc.
- Learning modifies the agent's decision mechanisms to improve performance

Types of Learning

- Supervised learning: correct answers for each input is provided
  E.g., decision trees, backprop 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
  E.g., Q learning, MDPs

Inductive learning

A form of Supervised Learning:
Learn a function from examples

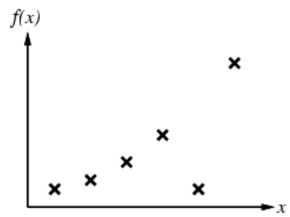
f is the target function. Examples are pairs (x, $f(x)$)

Problem: learn a function ("hypothesis") h
such that $h \approx f$ (h approximates f as best as possible)
given a training set of examples

(This is a highly simplified model of real learning:
Ignores prior knowledge
Assumes examples are given)

Inductive learning example

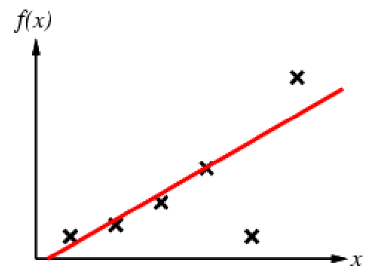
- Construct *h* to agree with *f* on training set
  h is consistent if it agrees with *f* on all training examples
- E.g., curve fitting (regression):



× = Input data point (a training example)

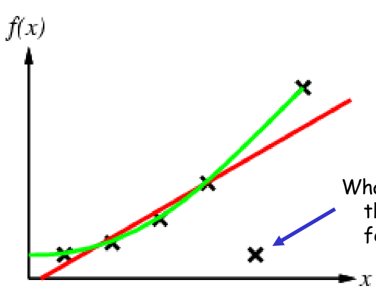
Inductive learning example

h = Straight line?



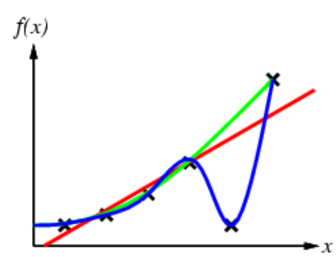
Inductive learning example

What about a quadratic function?



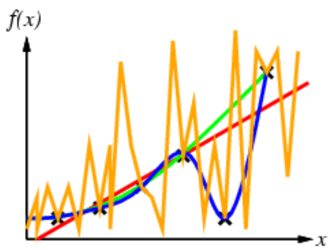
Inductive learning example

Finally, a function that satisfies all!

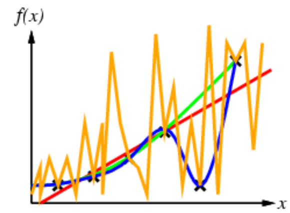


Inductive learning example

But so does this one…



Ockham's razor principle



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

- Smooth blue function preferable over wiggly yellow one
- If noise known to exist in this data, even linear might be better (the lowest x might be due to noise)

Decision Trees

Input: Description of an object or a situation through a set of **attributes**.

Output: a **decision**, that is the predicted output value for the input.

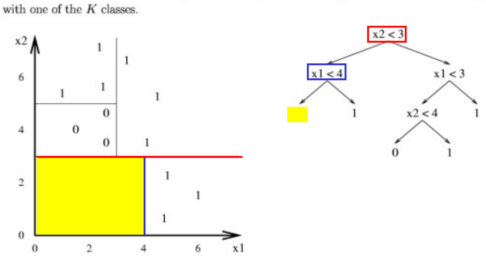
Both, **input and output can be discrete or continuous**.

Discrete-valued functions lead to **classification problems**.

Learning a **continuous function** is called **regression**.

Decision Tree Decision Boundaries

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



Experience: "Good day for tennis"

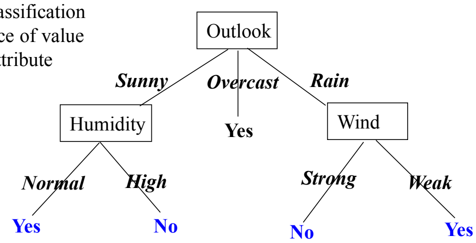
Day	Outlook	Temp	Humid	Wind	PlayTennis?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h	w	y
d4	r	m	h	w	y
d5	r	c	n	w	y
d6	r	c	n	s	y
d7	o	c	n	s	y
d8	s	m	h	w	n
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

Decision Tree Representation

Good day for tennis?

Leaves = classification

Arcs = choice of value for parent attribute



Decision tree is equivalent to logic in disjunctive normal form

G-Day $\Leftrightarrow$ (Sunny $\wedge$ Normal) $\vee$ Overcast $\vee$ (Rain $\wedge$ Weak)

DT Learning as Search

- Nodes
  - **Decision Trees**
- Operators
  - **Tree Refinement: Sprouting the tree**
- Initial node
  - **Smallest tree possible: a single leaf**
- Heuristic?
  - **Information Gain**
- Goal?
  - **Best tree possible (???)**

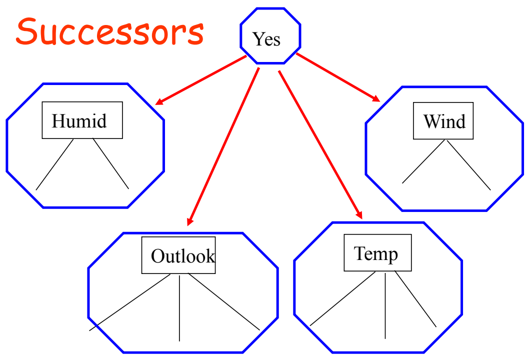
What is the Simplest Tree?

Day	Outlook	Temp	Humid	Wind	Play?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h	w	y
d4	r	m	h	w	y
d5	r	c	n	w	y
d6	r	c	n	s	y
d7	o	c	n	s	y
d8	s	m	h	w	n
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

How good?

[10+, 4-] Means:
correct on 10 examples
incorrect on 4 examples

Successors

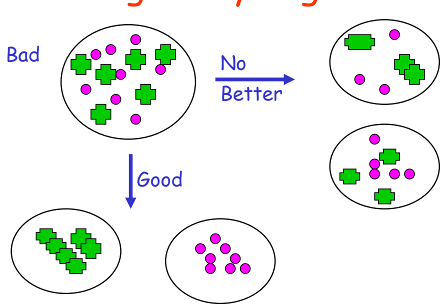


Which attribute should we use to split?

To be decided:

- How to choose best attribute?
  - Information gain
  - Entropy (disorder)
- When to stop growing tree?

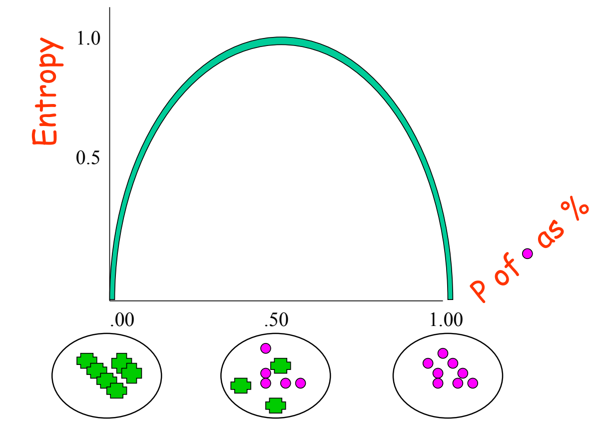
Disorder is bad
Homogeneity is good



Using information theory to quantify uncertainty

- Entropy measures the amount of uncertainty in a probability distribution
- Entropy (or Information Content) of an answer to a question with possible answers $v_1, \ldots, v_n$:

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$



Entropy (disorder) is bad Homogeneity is good

- Let S be a set of examples
- Entropy(S) = -P $\log_2$(P) - N $\log_2$(N)
  where **P** is proportion of pos example
  and **N** is proportion of neg examples
  and 0 log 0 == 0
- Example: S has 10 pos and 4 neg
  Entropy([10+, 4-]) = -(10/14) $\log_2$(10/14) - (4/14)$\log_2$(4/14)
  = 0.863

Information Gain

- ***Measure of expected*** ***reduction*** ***in entropy***
- ***Resulting from splitting along an attribute***

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

Where Entropy(S) = -P $\log_2$(P) - N $\log_2$(N)

Gain of Splitting on Wind

Values(wind)=weak, strong
S = [10+, 4-]
S_{weak} = [6+, 2-]
S_s = [3+, 3-]

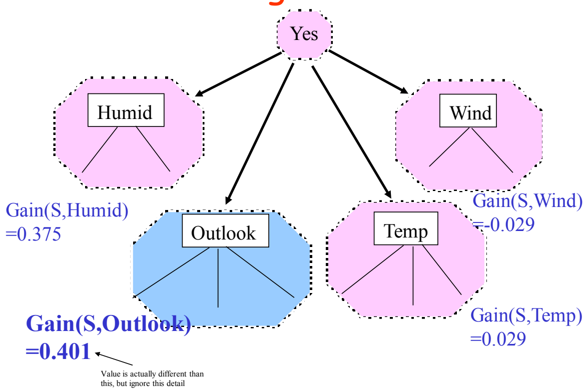
Gain(S, wind)

$$= \text{Entropy}(S) - \sum_{v \in \{weak,\ s\}} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

= Entropy(S) - 8/14 Entropy(S_{weak}) - 6/14 Entropy(S_s)
= 0.863 - (8/14) 0.811 - (6/14) 1.00
= -0.029

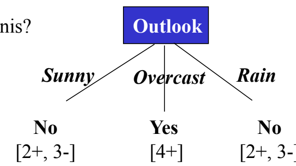
Day	Wind	Tennis?
d1	weak	n
d2	s	n
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	s	yes
d7	s	yes
d8	weak	n
d9	weak	yes
d10	weak	yes
d11	s	yes
d12	s	yes
d13	weak	yes
d14	s	n

Evaluating Attributes

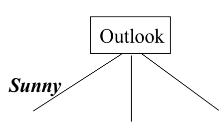


Resulting Tree ....

Good day for tennis?

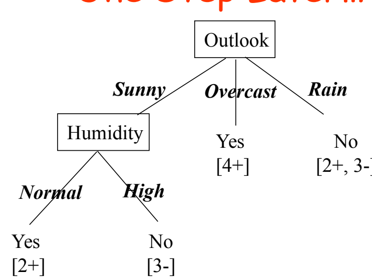


Recurse!



Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	n	weak	yes
d11	m	n	s	yes

One Step Later...

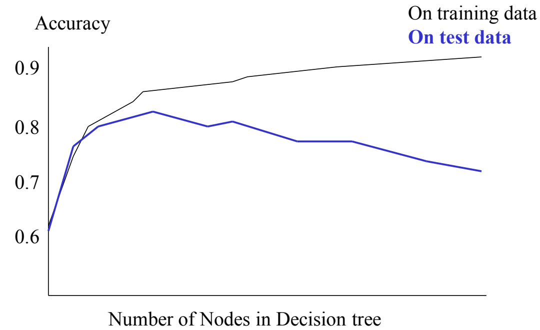


Decision Tree Algorithm

BuildTree(TrainingData)
    Split(TrainingData)

Split(D)
    If (all points in D are of the same class)
        Then Return
    For each attribute A
        Evaluate splits on attribute A
    Use best split to partition D into D1, D2
    Split (D1)
    Split (D2)

Overfitting



Overfitting 2

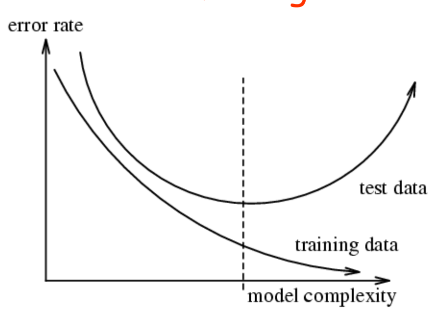


Figure from w.w.cohen

Overfitting…

- DT is *overfit* when exists another DT' and
  DT has ***smaller*** error on training examples, but
  DT has ***bigger*** error on test examples
- Causes of overfitting
  Noisy data, or
  Training set is too small

Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

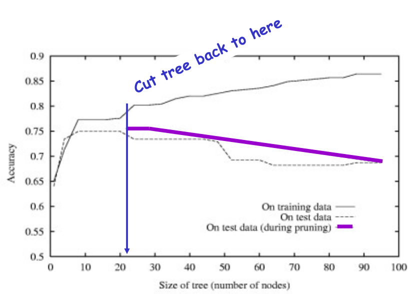
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

Effect of Reduced-Error Pruning



Other Decision Tree Features

- Can handle continuous data
  - Input: Use threshold to split
  - Output: Estimate linear function at each leaf
- Can handle missing values
  - Use expectation taken from other samples