Chapter 3: Decision Tree Learning

CS 536: Machine Learning

Littman (Wu, TA)

Decision-Tree Learning

[read Chapter 3]

[some of Chapter 2 might help...]

[recommended exercises 3.1, 3.2]

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

Classification Learning

Instances are vectors of attribute values.

A *concept* is a function that maps instances to categories (T, F, say).

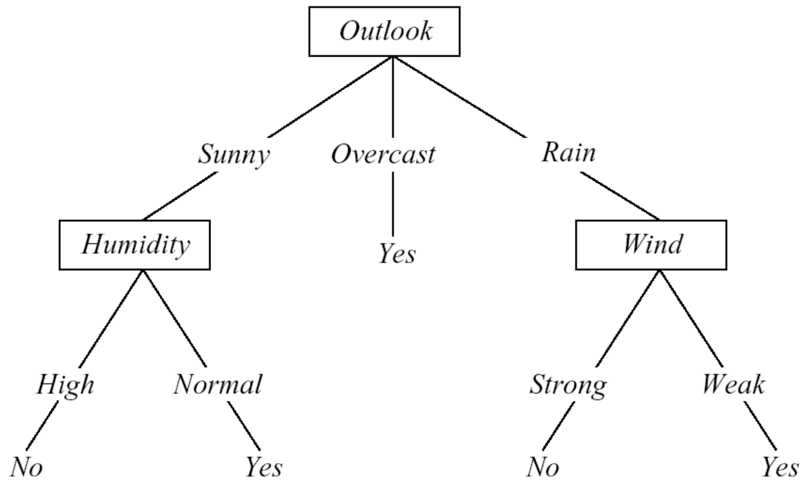
A *target concept* is the concept we want to learn.

A *hypothesis class* is the set of concepts we consider.

A *sample* of instances (or *training set*) is our source of information about the target concept.

Our *candidate concept* is usually evaluated by how well it classifies a separate sample of instances (the *testing set*).

Decision Tree for *PlayTennis*



Predicting C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge$, $\vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

Whence Decision Trees?

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

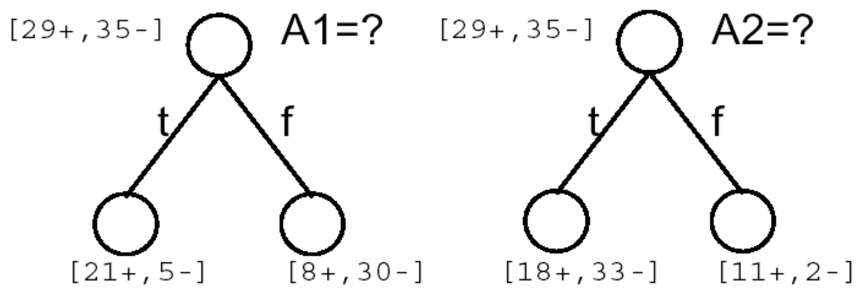
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-Down Induction

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which Attribute is Best?



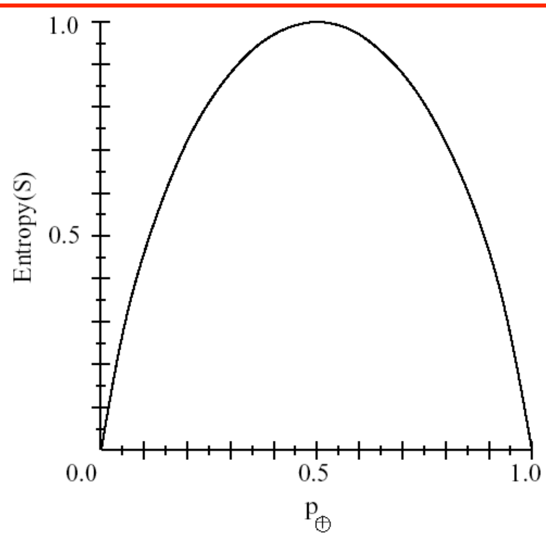
Measuring Entropy

- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\otimes}$ is the proportion of negative examples in S

Entropy measures the impurity of S

$$Entropy(S) = -p_{\oplus} \log p_{\oplus} - p_{\otimes} \log p_{\otimes}$$

Entropy Function



Entropy

Entropy(S) = expected number of bits needed to encode class ($\oplus$ or $\otimes$) of a randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\otimes$ of a random member of S:

$$p_{\oplus} (-\log p_{\oplus} -) + p_{\otimes} (-\log p_{\otimes})$$

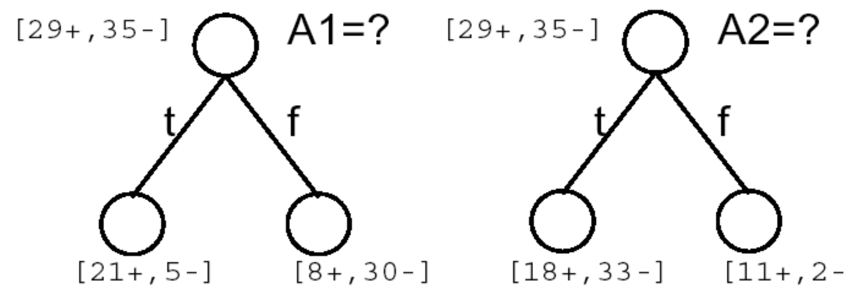
Information Gain

Gain(S, A) = expected reduction in entropy due to sorting S on A

$$Gain(S, A) \equiv Entropy(S) - \Sigma_{v \text{ in } Values(A)} |S_v|/|S| \; Entropy(S_v)$$

Here, S_v is the set of training instances remaining from S after restricting to those for which attribute A has value v.

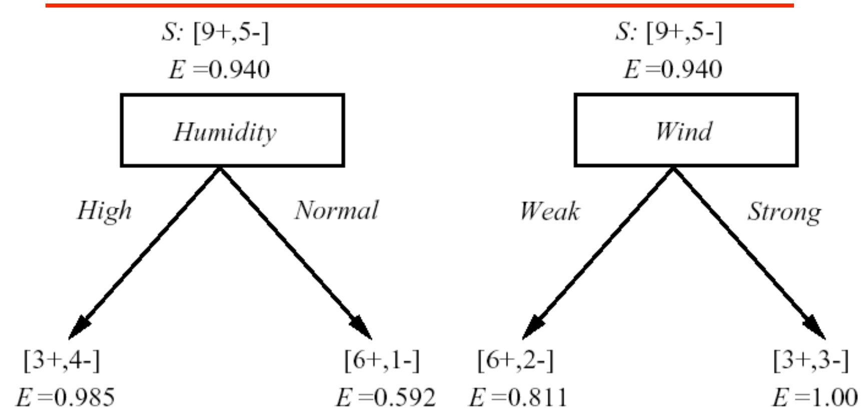
Which Attribute is Best?



Training Examples

Day	Outlook	Temp	Hum.	Wind	*PlayTennis*
D1	Sunny	Hot	High	Weak	*No*
D2	Sunny	Hot	High	Strong	*No*
D3	Overcast	Hot	High	Weak	*Yes*
D4	Rain	Mild	High	Weak	*Yes*
D5	Rain	Cool	Nml	Weak	*Yes*
D6	Rain	Cool	Nml	Strong	*No*
D7	Overcast	Cool	Nml	Strong	*Yes*
D8	Sunny	Mild	High	Weak	*No*
D9	Sunny	Cool	Nml	Weak	*Yes*
D10	Rain	Mild	Nml	Weak	*Yes*
D11	Sunny	Mild	Nml	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D13	Overcast	Hot	Nml	Weak	*Yes*
D14	Rain	Mild	High	Strong	*No*

Selecting the Next Attribute

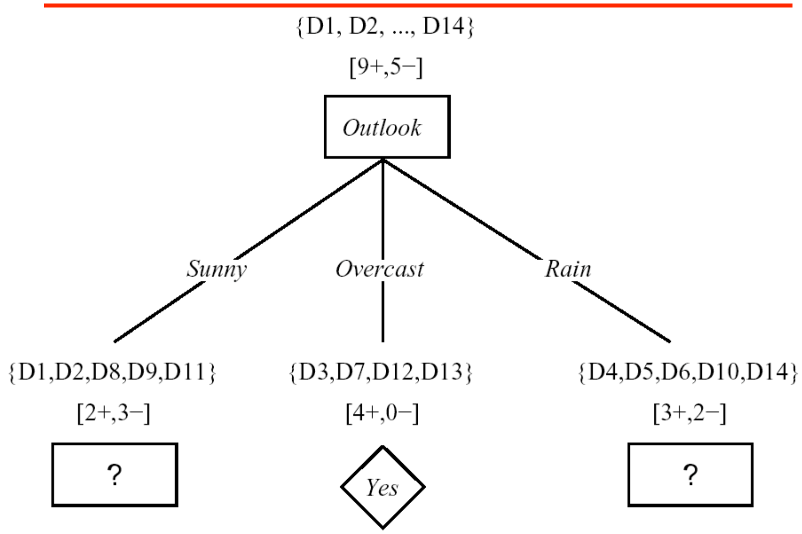


Which attribute is the best classifier?

Gain(S, *Humidity*) = .940 - (7/14).985 - (7/14).592 = .151

Gain(S, *Wind*) = .940 - (8/14).811 - (6/14)1.0 = .048

Attribute Bottom Left?



Comparing Attributes

S_{sunny} = {D1,D2,D8,D9,D11}

- *Gain (S_{sunny} , Humidity)*
  = .970 - (3/5) 0.0 - (2/5) 0.0 = .970
- *Gain (S_{sunny} , Temp)*
  = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570
- *Gain (S_{sunny} , Wind)*
  = .970 - (2/5) 1.0 - (3/5) .918 = .019

What is ID3 Optimizing?

How would you find a tree that minimizes:

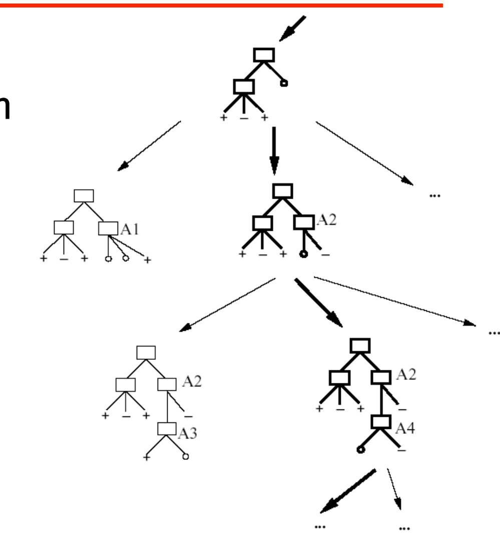
- misclassified examples?
- expected entropy?
- expected number of tests?
- depth of tree given a fixed accuracy?
- etc.?

How decide if one tree beats another?

Hypothesis Space Search by ID3

ID3:

- representation
  : trees
- scoring
  : entropy
- search
  : greedy



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias ≈ "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

- Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space H
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hyps. than long hyps.
  - a short hyp that fits data unlikely to be coincidence
  - a long hyp that fits data might be coincidence

Argument opposed:

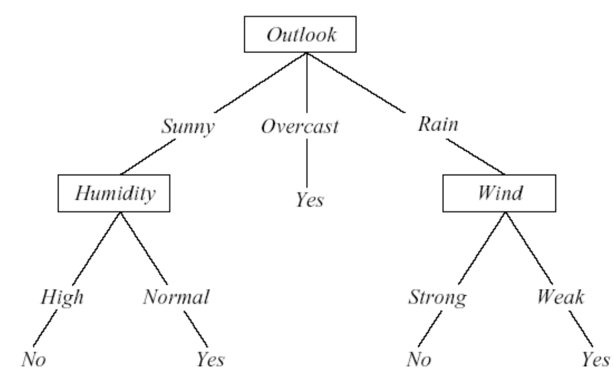
- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on size of hypothesis??

Overfitting

Consider adding noisy training example #15:

Sunny, Hot, Normal, Strong, PlayTennis = No

What effect on earlier tree?



Overfitting

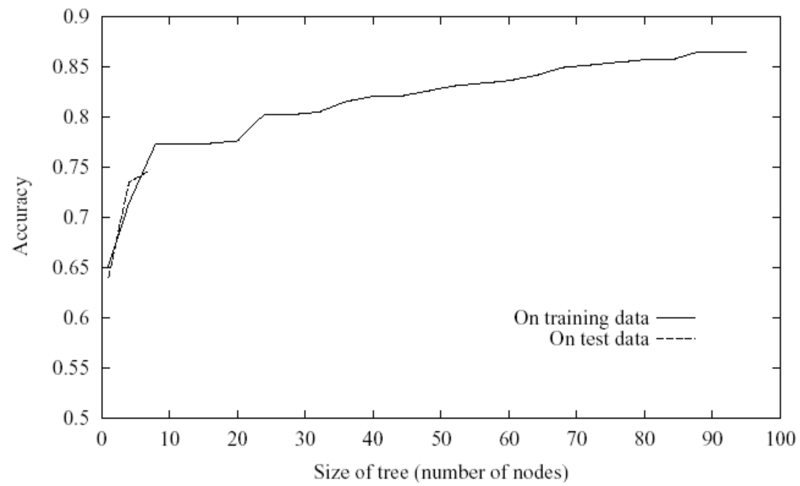
Consider error of hypothesis *h* over

- training data: $error_{train}(h)$
- entire distribution *D* of data: $error_D(h)$

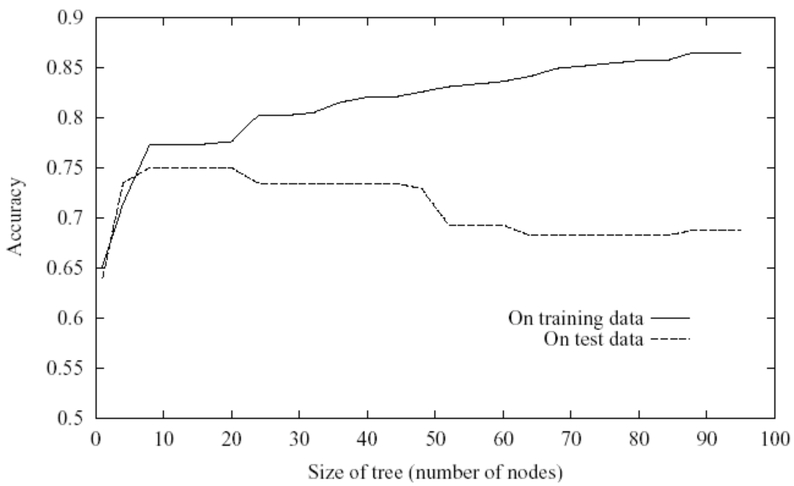
Hypothesis *h* in *H* **overfits** training data if there is an alternative hypothesis *h'* in *H* such that

- $error_{train}(h) < error_{train}(h')$, and
- $error_D(h) > error_D(h')$

Overfitting in Learning



Overfitting in Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune (DP alg!)

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize
  size(*tree*) + *size*(*misclassifications*(*tree*))

Reduced-Error Pruning

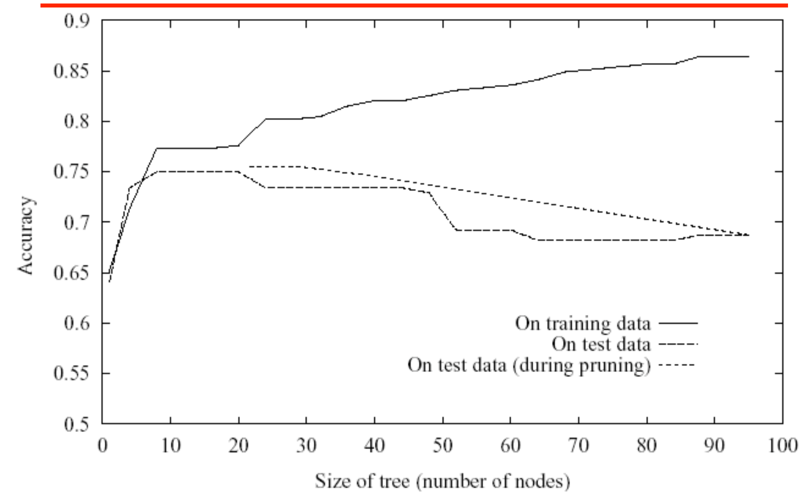
Split data into *training* and *validation* set

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

- produces smallest version of most accurate subtree
- What if data is limited?

Effect of Pruning

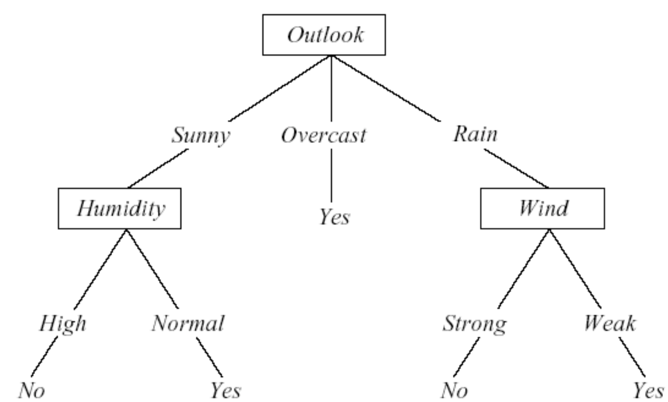


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Converting Tree to Rules



The Rules

IF (Outlook = Sunny) ∧ (Humidity = High)
THEN PlayTennis = No
IF (Outlook = Sunny) ∧ (Humidity = Normal)
THEN PlayTennis = Yes
...

Continuous Valued Attributes

Create a discrete attribute to test continuous

- *Temp* = 82.5
- (*Temp* > 72.3) = *t*, *f*

Temp:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Attributes with Many Values

Problem:

- If one attribute has many values compared to the others, *Gain* will select it
- Imagine using *Date* = *Jun_3_1996* as attribute

One approach: use *GainRatio* instead

$$GainRatio(S,A) \equiv Gain(S,A) \ / \ SplitInfo(S,A)$$

$$SplitInfo(S,A) \equiv -\Sigma_{i=1}{}^{c} \ |S_i|/|S| \ \log_2 |S_i|/|S|$$

where S_i is subset of S for which A has value v_i

Attributes with Costs

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, *Width_from_1ft* has cost 23 sec.

How to learn a consistent tree with low expected cost? Find min cost tree.

Another approach: replace gain by

- Tan and Schlimmer (1990)
  $Gain^2(S,A)/Cost(A)$
- Nunez (1988) [w in [0,1]: importance)
  $(2^{Gain(S,A)}-1)/(Cost(A)+1)^w$

Unknown Attribute Values

Some examples missing values of *A*?

Use training example anyway, sort it

- If node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*
- assign most common value of *A* among other examples with same target value
- assign probability p_i to each possible value v_i of *A* (perhaps as above)
  - assign fraction p_i of example to each descendant in tree
- Classify new examples in same fashion