

Informatics

Decision Trees

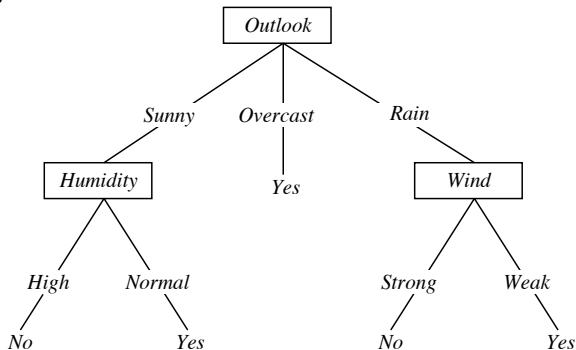
Francisco Iacobelli

Department of Computer Science
Northeastern Illinois University

September 18, 2011

Decision Tree Representation

Will I play tennis?



Disjunction of Conjunctions (what's my Function?)

$$h(x) = \text{yes} \iff$$

$$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$$

\vee

$$(\text{Outlook} = \text{Overcast})$$

\vee

$$(\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$$

Problems for Decision Trees

- ▶ Rulebased decisions (attribute / value pairs)
- ▶ Equipment diagnostics
- ▶ Credit risk
- ▶ Scheduling

Iterative Dichotomiser 3: ID3

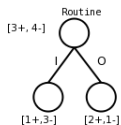
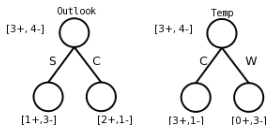
(Quinlan 1986)

Building a tree Top - Down

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

Best classifier attribute?

Outlook	Temp	Routine	WearCoat
Sunny	Cold	InDoors	NO
Sunny	Warm	OutDoors	NO
Cloudy	Warm	InDoors	NO
Sunny	Warm	InDoors	NO
Cloudy	Cold	InDoors	YES
Cloudy	Cold	OutDoors	YES
Sunny	Cold	OutDoors	YES

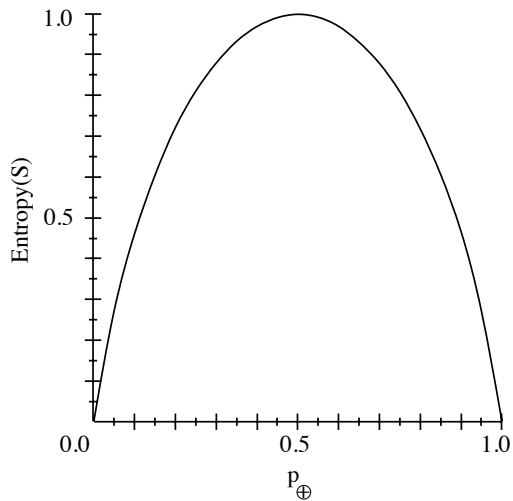


Entropy

- ▶ Measures the impurity of examples
- ▶ S is a sample of training examples
- ▶ p_{\oplus} is the proportion of positive examples in S
- ▶ p_{\ominus} is the proportion of negative examples in S
- ▶ Entropy measures the impurity of S

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy



Entropy

$Entropy(S)$ = expected number of bits needed to encode class (\oplus or \ominus) of 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 \ominus of random member of S :

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

More generally:

$$Entropy(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

Information Gain

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

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Information Gain

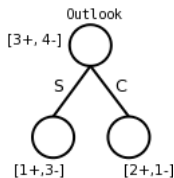
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Values(Outlook) = \{Sunny, Cloudy\}$

$S = [3+, 4-]$

$S_{Sunny} = [1+, 3-]$

$S_{Cloudy} = [2+, 1-]$



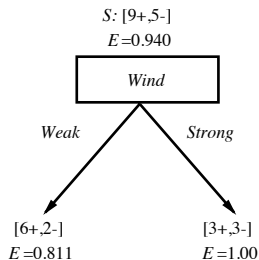
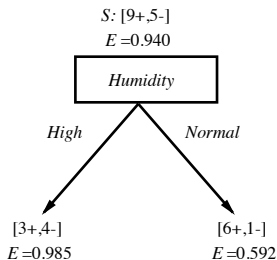
$$Gain(S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Cloudy\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

Example

Day	Outlook	Temperature	Humidity	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	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute

Which attribute is the best classifier?

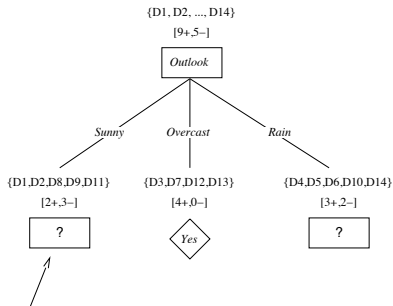


$$\begin{aligned} \text{Gain}(S, \text{Humidity}) &= .940 - (7/14).985 - (7/14).592 \\ &= .151 \end{aligned}$$

$$\begin{aligned} \text{Gain}(S, \text{Wind}) &= .940 - (8/14).811 - (6/14)1.0 \\ &= .048 \end{aligned}$$

- ▶ $\text{Gain}(S, \text{Outlook}) = 0.246$
- ▶ $\text{Gain}(S, \text{Humidity}) = 0.151$
- ▶ $\text{Gain}(S, \text{Wind}) = 0.048$
- ▶ $\text{Gain}(S, \text{Temperature}) = 0.029$

Selecting the Next Attribute



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

Hypothesis Space Search by ID3

- ▶ Hypothesis Space is *complete*: Contains the target function
- ▶ Only outputs a single hypothesis: Can't play 20 questions...
- ▶ No backtracking: It may get stuck in a local minima
- ▶ Statistically-based search using all examples: Robust to noise

Inductive Bias

“Prefer shortest tree”

→ 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

Note H is the power set of instances X

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

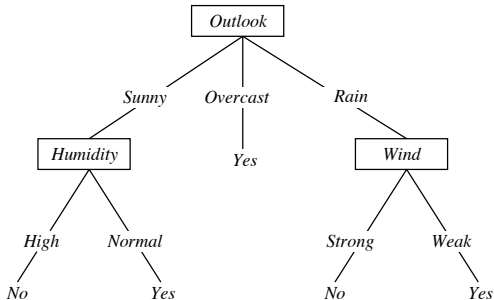
Argument opposed:

- ▶ 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??

Consider adding noisy training example #15:

Sunny, Hot, Normal, Strong, PlayTennis = No

What effect on earlier tree?



How does this new hypothesis h compare h' given above?

Overfitting

Consider error of hypothesis h over

- ▶ training data: $error_{train}(h)$
- ▶ entire distribution \mathcal{D} of data: $error_{\mathcal{D}}(h)$
- ▶ Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

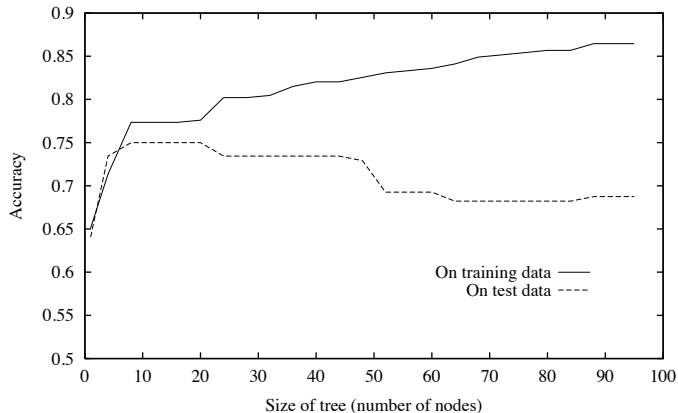
$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

- ▶ In other words: there's an alternative hypothesis h' that is not as good a fit as h for the training data, but fits better with the test/real data.

Overfitting



Diabetes diagnosis using decision trees

Avoid 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
- ▶ Minimum Description Length (MDL): minimize $size(tree) + size(misclassifications(tree))$

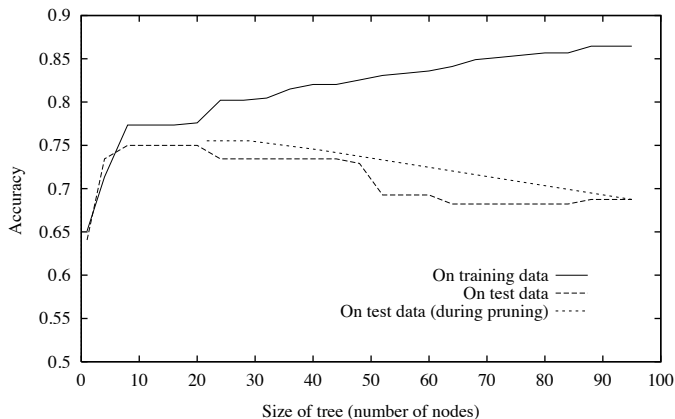
Reduced-Error Pruning

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)
 - by removing subtrees and making them leaf nodes with the most likely value
2. Greedily remove the one that most improves *validation* set accuracy

Reduced-Error Pruning



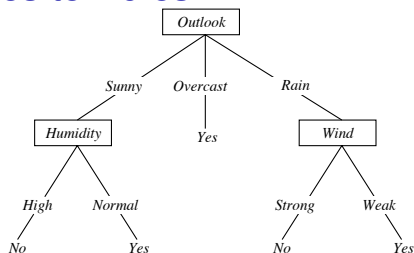
- ▶ produces smallest version of most accurate subtree
- ▶ What if data is limited?

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 Trees to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

Continuous Valued Attributes

Create a discrete attribute to test continuous

- ▶ $Temperature = 82.5$
- ▶ $(Temperature > 72.3) = t, f$
- ▶ Or obtain some reasonable thresholds and keep the one that maximizes information gain.

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

Attributes with too many values

Problem:

- ▶ If attribute has many values, *Gain* will select it
- ▶ Imagine using *Date = Sept₁3₂011* as attribute

One approach: use *GainRatio* instead

$$\text{GainRatio}(S, A) \equiv \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

$$\text{SplitInformation}(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

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

Attributes with Cost

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?

One approach: replace gain by

- ▶ Tan and Schlimmer (1990)

$$\frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}.$$

- ▶ Nunez (1988)

$$\frac{2^{\text{Gain}(S, A)} - 1}{(\text{Cost}(A) + 1)^w}$$

where $w \in [0, 1]$ determines importance of cost

Missing Values

What if some examples missing values of A ?

Use training example anyway, sort through tree

- ▶ 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
 - ▶ assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion