Decion Tree Learning

Chapter 3

Recap from Chapter 2

- Concept learning
- consistent hypothesis (h(x)= c(x))and satisfying hypothesis (h(x) = 1)
- Find-S algorithm
- Candidate Elimination algorithm
- Inductive Bias

Topics

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

Decision Tree Definition

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.

Decision trees classify instances by sorting them down the tree from theroot to some leaf node, which provides the classification of the instance.
Example:

$$F : \langle Outlook, Humidity, Wind, Temp \rangle \rightarrow PlayTennis?$$

Decision Tree for $PlayTennis$

Decion Tree Learning

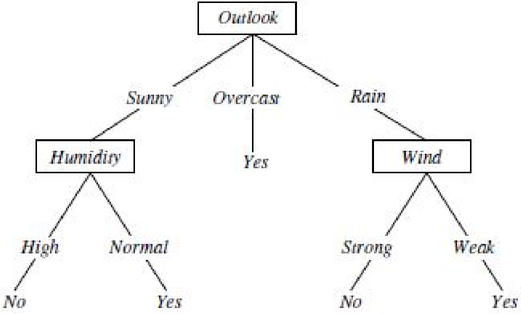
Overview

Entropy

Information Gain

Inductive Bias in ID3

Occam's Razor



This corresponds to

$$(Outlook = Sunny \land Humidity = Normal) \lor (Outlook = Overcast) \lor (Outlook = Rain \land Wind = Weak) \tag{1}$$

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

The algorithm to construct decision tree (IDB3) is greedy search top down approach.

Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - $x = \langle x1, x2 \ldots xn \rangle$.
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H = h || h : X \rightarrow Y$
  - each hypothesis h is a decision tree

Input: Training examples $\langle x(i), y(i) \rangle$ of unknown target function f.
Output: Hypothesis $h \epsilon H$ that best approximates target function f

Decision Tree

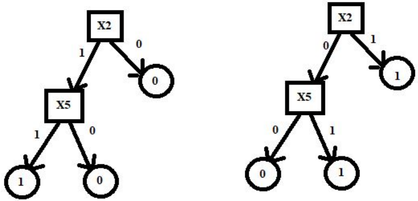
Suppose

$$X = \langle X1, \dots Xn \rangle$$

where Xi are boolean variables.
How would you represent

$$Y = X2X5?$$

$$Y = X2 \vee X5$$



When to Consider 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 of Decision Trees

Main loop:

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

Which attribute is best?

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:

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

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

Entropy

Decion Tree Learning

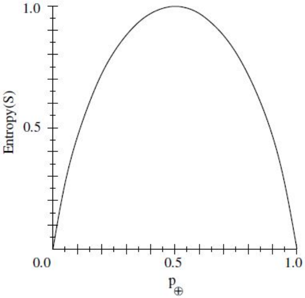
Suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-1 to summarize such a sample of data). Then the entropy of S relative to this boolean classification is

$$Entropy([9+, 5-]) \equiv -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

$$\equiv 0.940$$

Note: Entropy is 0 if all members of S belong to the same class. entropy is 1 when the collection contains an equal number of positive and negative examples.

Entropy



- 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

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

Information Gain

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

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

Gain - Example

For example, suppose S is a collection of training-example days described by attributes including *Wind*, which can have the values *Weak* or *Strong*. As before, assume S is a collection containing 14 examples, $[9+, 5-]$. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have *Wind* = *Weak*, and the remainder have *Wind* = *Strong*. The information gain due to sorting the original 14 examples by the attribute *Wind* may then be calculated as

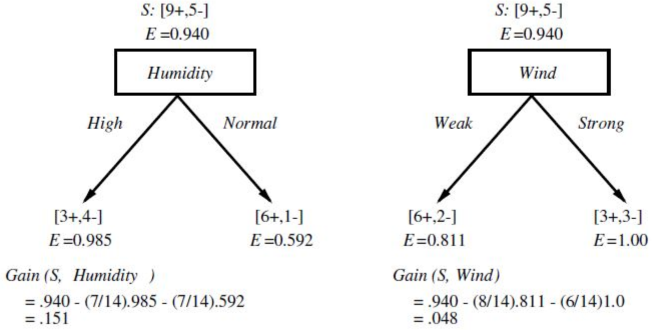
$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

Training Examples

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?

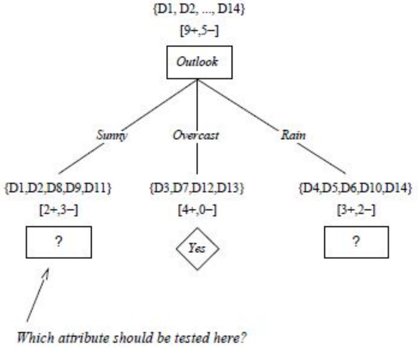


Selecting the Next Attribute

Which attribute is the best classifier?
Gain(S, Outlook) $= 0.246$
Gain(S, Humidity) $= 0.151$
Gain(S, Wind) $= 0.048$
Gain(S, Temperature) $= 0.029$

Note: Process of selecting new attribute continues for each new leaf node until either of two conditions is met:
(1) every attribute has already been included along this path through the tree, or
(2) the training examples associated with this leaf node all have the same target attribute value

Which attribute to be tested



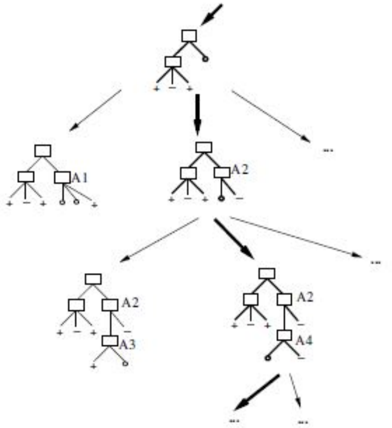
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Hypothesis Space Search by ID3



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...
- Statisically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

$\rightarrow$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.

$\rightarrow$ a short hyp that fits data unlikely to be coincidence

$\rightarrow$ a long hyp that fits data might be coincidence

Argument opposed:

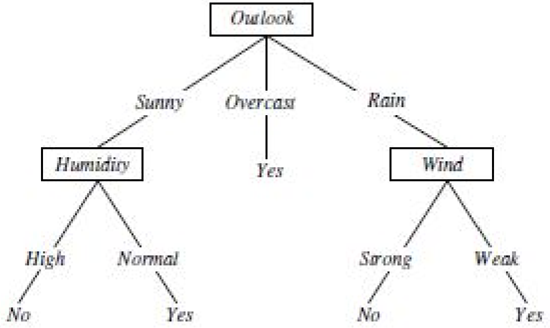
- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on size of hypothesis??

Overfitting in Decision Tree Learning

Consider adding noisy training example #15:

$Sunny, \; Hot, \; Normal, \; Strong, \; PlayTennis = No$

What effect on earlier tree?



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
- MDL: minimize $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

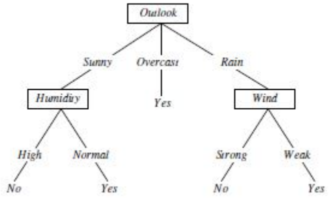
- Evaluate impact on *validation* set of pruning each possible node (plus those below it)
- Greedily remove the one that most improves *validation* set accuracy

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

Rule Post-Pruning

- Convert tree to equivalent set of rules
- Prune each rule independently of others
- Sort final rules into desired sequence for use

Converting A Tree 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

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

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

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

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

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

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

Attributes with 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?
One approach: replace gain by

- Tan and Schlimmer (1990)

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

- Nunez (1988)

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

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

Unknown Attribute 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