Decision Trees

CS4881 Aritificial Intelligence
Jay Urbain

Credits: Tom Mitchell, Machine Learning

Outline

- Widely used, practical method for inductive inference.
- Method for approximating discrete valued functions.
- Exhaustively search hypothesis space.
- Learned trees can be used for representing if-then-else rules.

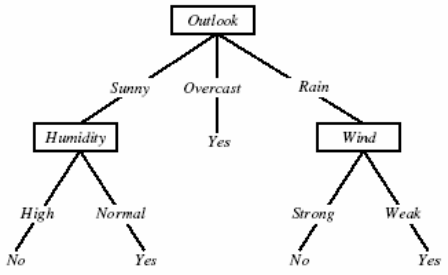
DT Representation

- Classify instances by sorting them down a tree from the root to some leaf node.
- Leaf node provides classification instance.
- Each node in tree specifids a test of some attribute of the instance.
- Each branch descending from that node represents one of the possible values for this attribute.

DT Classification

- An “instance” is classified by:
  - starting at the root node of the tree
  - testing the attribute specified by this node
  - Moving down the tree branch corresponding to the value of the attribute in the given example.
  - Process is repeated for each subtree rooted at the new node.
  - Each leaf represents classification of instance.

DT for Play Tennis



Play Tennis Example

- How to classify:
  - {outlook=sunny, temperature=hot, humidity=high, wind=strong}
- Decision trees represent a *disjunction of conjunctions* (sound familiar?) of constraints of the attribute values of instances.
- The previous decision tree corresponds to:

(outlook=sunny ^ humidity=high) v (outlook=overcast) v (outlook=rain ^ wind=weak)

Predict C-Section Risk Example

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-
```

Representation Questions

How would you represent?:

- ^, v, XOR
- (A^B)v(C^!D^E)
- M of N

Appropriate problems DT's

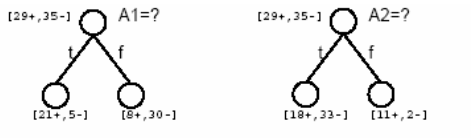
- Instances decribable by attribute-value pairs
- Target function is discrete values
- Disjunctiove hopthesis may be required
- Possibly noisy training data
- Examples:
  - Equipment of medical diagnosis
  - Credit risk analysis
  - Modelig calendar scheduling preferences

Top-down induction of DTs

Main loop:

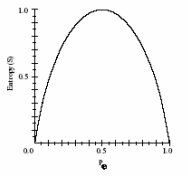
1. *A* <- the "best" decision attribute for next *node*
2. Assign *A* as decision attribute for *node*
3. For each value of *A,* create new descendnat of *node*
4. Sort training examples to leaf *nodes*
5. If training examples perfectly classified, then STOP
6. Else iterate over new leaf nodes

Which attribute is best?



Entropy

- Given, *S*, a set of training examples.
- Entropy measres the impurity of S
- p+ is proportion of positive examples of S
- P- is proportion of negative examples of S



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

Entropy

Entropy(S) = expected number of bits needed to encode class(+/-) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

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

So, expected number of bits to encode + or – examples of random member of S:

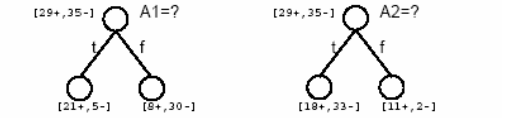
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

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

Information Gain

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

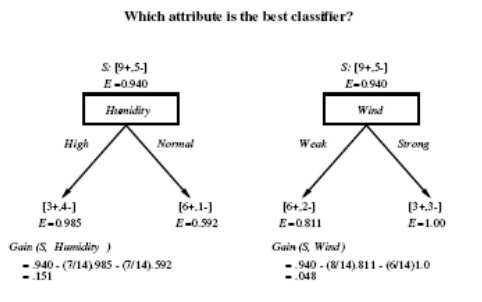
$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

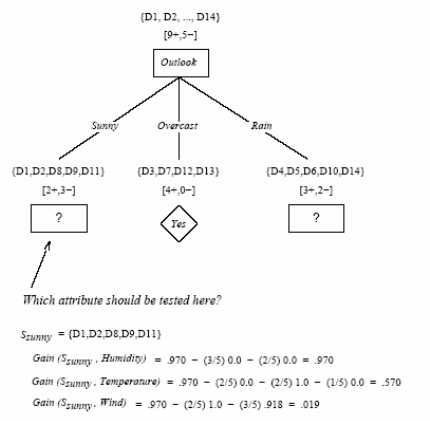


Training Examples

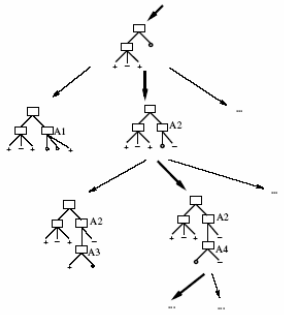
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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





Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function is surely in there…
- Outputs a single hypothesis (which one?)
- No backtracing
  - Local minuma
- Statiscally=based search choices
  - Robust to noisy data
- Inductive bias: approx "prefer shortest tree"

Inductive Bias of ID3

Note *H* is the power set of instances *X*

- Unbiased?

Note really…

- Prefer short trees, and for those with high info gain attributes near the root
- Bias is a preference for some hypothesis, 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 hypo's than long hypo's
- A short hypo that fits data unlikely to be coincidence
- A long hypo that fist data might be coincidence

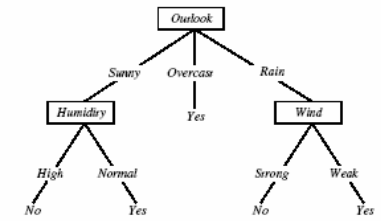
Argument opposed:

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

Overfitting in Decision Trees

Consider adding noisy traing example D15:

- Sunny, Hot, Normal, Strong, Playtennis=No



Overfitting

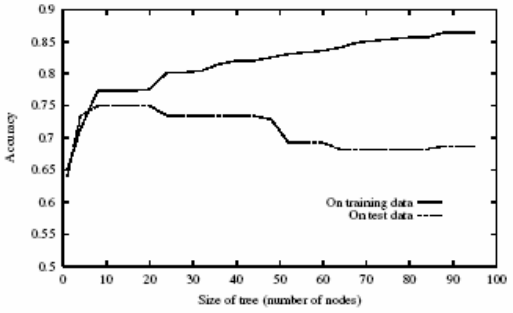
- Consider error of hypothesis *h* over:
  - Training data: $error_{train}(h)$
  - Entire distribution *D* of data: $error_D(h)$
- Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that:

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

and

$$error_D(h) > error_D(h')$$

Overfitting



Avoiding 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 set
- *MDL: minimize size(tree)+size(missclassification)*

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

Rule Post-Pruning

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

- Most frequently used method, e.g., C4.5

Decision tree learning

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

```
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
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```