

Lecture 6: Decision Trees II

How to select attribute?

Want attribute to make biggest step towards exact classification

1. one-step lookahead (find attribute that would lead to fewest errors if tree stopped there)

But may not lead to best tree – why?

$x1$	$x2$	$x3$	y
0	0	0	-
0	0	1	-
0	1	0	+
0	1	1	+
1	0	0	+
1	0	0	+
1	1	1	-
1	1	1	-

2. information theory: measure information provided by each attribute, choose max

Example: information measured by entropy:

$$Entropy(C) \equiv H(C) \equiv - \sum_c P(C = c) \log_2 P(C = c)$$

Measuring entropy (“impurity”)

entropy of class C of set of examples S

$$H(C) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

where $p_+ = \frac{p_i}{p_i + n_i}$; p_i positive and n_i negative examples in S

number of bits required on average to encode class C of example in S
(shorter codes assumed for more likely cases)

entropy can be used to measure gain in information about class by branching on attribute A – compute reduction in entropy of class C caused by knowing A value

Information Gain

$$Gain(C, A) = I(C, A) = H(C) - \sum_{v \in V(A)} P(A = v)H(C|A = v)$$

second term is entropy of C after partitioning based on A , averaged across values $V(A)$

note: this measures the mutual information between A and C : the amount of information we learn about C by knowing value of A

can also use this measure to prune decision trees – decide that attribute does not contribute enough information

Normalizing gain ratio

Entropy/gain favors attributes with many values

Instead can use gain ratio – normalize by SplitInformation which penalizes attributes with lots of values

$$\text{SplitInformation}(S, A) = - \sum_{v \in V(A)} P(v) \log P(v)$$

this measures entropy of S wrt $V(A)$ (instead of wrt class C)

$$\text{GainRatio}(S, A) = \text{Gain}(S, A) / \text{SplitInformation}(S, A)$$

Pruning methods

Combat **over-fitting** by limiting number of leaf nodes – must trade off against class entropy at leaves

Often use validation set for **early stopping**, or to guide pruning

1. **rule-based**: convert tree into set of if-then rules (one per root-leaf path); remove pre-conditions; sort rules by decreasing accuracy
2. **reduced-error**: find node whose replacement with leaf (majority class) yields highest increase in accuracy on validation set
3. **sub-tree**: find min $[(H(C) \text{ at node} - H(C) \text{ at sub-tree leaves})/\text{number-of-leaves}]$; replace with leaf (majority class)

Decision tree algorithms

model of human concept learning (Hunt)

1. ID3, by Quinlan, (original, standard algorithm): entropy as the selection rule; no pruning
2. CART, by Breiman and others: binary splits; sub-tree pruning
3. C4.5, C5.0 – also by Quinlan: binary splits; entropy as selection; pruned via rule conversions; handles noise, continuous attributes, missing-data

Continuous Attributes

can be discretized a priori into intervals

or develop splits using threshold – chosen to max information gain

- sort examples based on value for attribute
- check midpoints between items of different classes

Comparison to k -NN

k -NN vs. decision trees

1. decision boundaries:

piece-wise linear vs. axis-aligned, tree-structured

2. test complexity:

non-parametric, few parameters besides (all!) training examples vs.
attributes and splits

What are decision trees good for?

Can express any Boolean function, but most useful when functions depend critically on few attributes

Bad on: parity, majority functions; also not suited to real-valued attributes

Practical application example: flight simulator

- 20 state variables
- 90K examples based on expert pilots' actions
- tree → auto-pilot, out-performed experts

Advanced decision trees:

1. other selection and pruning methods
2. rule derivation
3. attributes as combinations of input features