Decision trees

Oliver W. Layton

CS251: Data analysis and visualization

Lecture 33, Spring 2019

Monday April 29

Plan

- ID3 algorithm
- Pruning

ID3 Algorithm: third iterative dichotomizer

How do we know which feature to put in which node?

1. Compute information gain for each unused feature.
2. Also use information gain to determine "best" split point WITHIN each unused feature.
3. Assign to the next node the feature with maximal information gain.
4. Stop when every branch ends with a **leaf node** (pure or "pure enough" node) OR when all the features are used up.

- Let's use ID3 to create a decision tree based on the tennis data.

Tennis data

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

When to stop splitting

- Potential problem with ID3 decision tree: **overfitting**, every detail in training set contributes to tree structure (including noise). **Our learning becomes too specific to the training set**. What do the leaves of an overfit decision tree look like?
- Ways to mitigate overfitting
  - **Split count threshold**: Don't allow feature splits if the ANY class count < T. Example: $T = 2$ (NO) [4, 1]. (YES) [3, 2]
  - Limit tree height. Smaller trees: greater training error, but generalize better.
  - **Prune tree** after training: reduce height by removing weak branches.

Reduced error pruning

- Split training set into train and validation sets.
- Build full tree on training set.
- Replace each parent of leaf nodes with a new leaf node, assign majority class of current parent.
- Check classification accuracy on validation set. No effect on error? Do another pruning pass. Error increases too much? Stop.
- Advantage: Simple and fast.

Cost complexity pruning

- Generate M trees: $T_0, T_1, \ldots, T_{M-1}$. T_0 is full tree, T_{M-1} is root only.
- i^{th} tree generated by removing a subtree from tree $i-1$ ("the weakest link").
- Removed subtree replaced with new leaf node.
- At each step starting from full tree, remove subtree that minimizes **removal error**.
- Define T_i as current tree, T_i^t as tree with subtree t removed, $\mathbf{Error}(x)$ is data misclassification rate
- Removal error that is minimized: $g(t) = \dfrac{\mathbf{Error}(T_i) - \mathbf{Error}(T_i^t)}{\mathbf{NumLeaves}(T_i^t) - 1}$
- Advantage: More thorough than reduced error pruning, but slower and memory load much higher.

Occam's razor: 1R decision trees

- Often performance may be good enough with only a single rule (1R) decision tree (**decision stump**) (root node + N leaf nodes)!
- Generalizes well and classifies data based on "most meaningful split" according to the "most meaningful feature".
- Let's write a function that determines the "most meaningful split" for one feature. We will make use of the split count threshold.
- We will need a helper function that computes...
- We will test the 1R decision tree on medical diagnostic data (*Bartter's syndrome*; genetic kidney disorder).

Wisdom in numbers

- We can make many decision stumps with different features / splits (maybe quasi-randomly).
- While one might not be best, the vote of many might give good classification results (**random forest**).
- Combining outputs of multiple weak learners called **ensemble learning**.

Why use decision trees over other machine learning methods?

- Easy to interpret structure (if-then rules).
- Fast test set processing
- Conceptually simple
- Classification rules are human-readable
- Handles categorical and numerical data
- Straight-forward approach to decrease overfitting (prune, limited height etc.)