Day 18: Inductive Decision Trees (Basic ML)

The Classification Problem

- Given a finite set of examples of concepts, where each example is expressed as a set of values for a set of features, where the examples are classified into different groups
    - Discover rules for classification that accurately and efficiently generalize to other unseen examples in the domain
    - Approaches:
        - Symbolic approach: format the rules/reasons such that they're compact and esy to understand
    - Superproblems:
        - Non-feature-based definitions, where the data is more freeform
            - Work with pixels or soundwaves directly
        - Not be given the classes (this is unsupervised learning)
    - Subproblems:
        - Features are only clean booleans
- Historical milestones:
    - Quine-McClusky: logic minimization
    - Michaelski's predicate logic
    - **Quinlan's/Fisher's decision trees**
    - **Mitchell's LEX**: refined concept spaces
    - RHW Back Propagation: layered NNs
- Classification approaches:
    - Store the whole database

    - That's inefficient and doesn't allow generalization
    - You can combine with a similarity function (nearest neighbor, etc.) to get a still-inefficient but generalizable approach
  - Extract logical rules from the data
    - Ex: `~Parking -> Office, SatDish & ~Heliport -> Mall, Parking & ~Signage -> Hospital`
    - Determining the minimal logic required is related to the NP-complete vertex covering problem.
  - Extract a decision tree.

Decision Trees

- Decision tree example:

```
if building is tall: it's an office
  else if building has sinage: it's a mall
    else: it's a hospital
```

- Decision tree creation algorithm:

```
if all remaining data is in the same class:
  return that class
else:
  // how you pick the variable can be important! random?
  select a variable to use to make a decision ($var)
  split into 2 subproblems by removing $var and \
      dividing rows into 2 groups by the value of $var
  recursively solve those two problems
```

- How to pick the split variable to get a more efficient decision tree?
  - You want to pick the best variable at every opportunity to minimize the complexity of the decision tree
  - Fundamentally this is a heuristic

  - One option: **Mutual Information:** A variable gives us *perfect information* if it completely discriminates between classes, *some information* (or *good information*) if it is unbalanced, or no information if it doesn't help discriminate between classes
    - To do this, look at the conditional probabilities of each property. Look for the one that most accurately splits on class boundaries (either perfectly identifies one class or perfectly identifies a set of classes)
    - Mutual information with decision trees follow Occam's Razor (simpler is better)
- How to generalize?
  - Smaller decision trees (ex. mutual information) are thought to generalize better
- What about non-binary data?
  - Continuous Data: context dependent. Use average as threshold?
  - Finite set of values: use case statement/trees with branching factor >2, or use heuristics to divide values into fewer subsets
- What's the relationship to Neural Networks?
  - NNs can be used for classification
  - Features need to be converted to floats with thresholding
  - Much slower