General Q&A

No questions.

Theory Tasks

Chapter 6: Association Rules

Why can support/confidence for association rules be misleading?

- support:
  - measure that can also be applied to plain itemsets, not only association rules
  - quantifies how often itemsets on LHS and RHS of association rule occur together
  - i.e., tells how often rule satisfied (relative to number of transactions or absolute)
  - but does not consider how often rule is invalid (only LHS satisfied) → use confidence
- confidence:
  - normalizes support of association rule with support of LHS
  - conceptually similar to metric *precision* in classification
  - asymmetric measure: does not consider support of RHS
    - might be high even if there is negative correlation between LHS and RHS
    - in particular, danger if support of RHS is high already without LHS
    - see basketball/cereal example from lecture
  - in contrast, *lift* is symmetric quality measure (but has other weaknesses)

Do you see a relationship between association rules and decision trees?

- at first glance, very different approaches
  - association rules are unsupervised (no target variable)
    - decision trees are supervised
  - association rules work with transactions of (feature-less and target-less) items
    - decision trees work with feature-target representation of data objects
  - association rules might not apply (if LHS of rule not contained in transaction)
    - decision trees classify each data object (they partition data space fully)
  - association rules are individual rules
    - decision trees are a hierarchical structure of rules (applied in combination)
- main similarity: both approaches use if-then rules
  - association rules: if transaction contains LHS, then it also should contain RHS
  - decision tree: if feature has certain value, follow corresponding path and either test another feature (internal node) or assign class label (leaf node)
- you can apply association rules to tabular feature-target data
  - step 1: transform dataset into transaction-item representation
    - each data object becomes a transaction
    - each feature value or target value becomes an item
    - thus, each transaction has same length
    - don't forget feature/target name → multi-dimensional itemset (see the lecture)
    - need to discretize features (unlike decision trees, which support numeric features)
    - exemplary transaction for a data object from the `iris` dataset: {sepal.length=short, sepal.width=long, petal.length=short, petal.width=short, species=setosa}
    - i.e., values of four features and a target transformed to transaction with five items

  - step 2: mine multi-dimensional association rules
    * rules like {petal.width=short} $\rightarrow$ {petal.width=long} cannot occur
    * also, we want rules where feature values are on LHS and a target value is on RHS
    * (we could also mine rules to explore relationships between features)
    * these constraints might make mining (candidate generation) more efficient
  - step 3: post-process rules
    * problem: zero, one, or multiple rules might apply to features of a data object
    * need to define default prediction if no rule matches
    * need to decide what to do if multiple rules match, e.g., bring rules into an order
  - exemplary algorithm: CBA (Classification Based on Associations) from paper "Integrating Classification and Association Rule Mining"
  - exemplary Python package: `pyarc`

How does sorting help in FP-trees?

- sorting by frequency is integral component of original FP-tree algorithm
  - done between first scan of database (count items) and second scan (build tree)
- *prefix path property* of algorithm depends on frequency sorting of items in transactions
  - bonus task: does algorithm work with arbitrary (potentially fixed) order of items?
- sorting by frequency improves efficiency
  - FP-tree smaller (has less paths) if transactions share first few items (so-called prefix)
  - more frequent items more likely to be shared, thus sorted to beginning of transactions