COS 781 (Data Mining)

Rule Induction
Further Reading

What is Rule Induction?

- Builds up a set of decision rules
  - A rule for each class
  - Rules are progressively refined
  - Usually there is a default rule
Rule Induction Terminology

- Rule components
  - Conditions (expressions)
    - Basic attribute test
      - Selector
    - Conjunct of selectors
      - Complex
    - Disjunct of complexes
      - Cover
  - Classification
    - Class label
Rule Induction Terminology

- Examples of selectors
  - \( \langle \text{Cloudy} = \text{yes} \rangle \)
  - \( \langle \text{Weather} = \text{wet} \land \text{stormy} \rangle \)
  - \( \langle \text{Temp} > 60 \rangle \)
Rule Induction Terminology

- An expression “covers” an example if it is true for the example

- Axioms:
  - Empty complex (conjunct of zero attribute tests) covers all examples
  - Empty cover (disjunct of zero complexes) covers no examples
AQR

- A generic form of the original AQ algorithm by Michalski
- Implemented by Clark and Niblett for comparison to CN2
AQR: Algorithm Output

- **Unordered** set of decision rules
  - One for each class
  - A cover that predicts a class label

- **Classification of new examples**
  - Find set of rules that cover the new example
    - If this set contains one rule, its class is predicted
    - If this set contains more than one rule, the most common class of training example covered by the set is predicted
  - If no rules cover the example, default rule predicts most commonly occurring class in the set of training examples
AQR: Algorithm Overview

- Iteratively builds a cover for each class
  - Stage 1: Select an un-covered example. Build a cover, one complex at a time, that covers the example, but no negative examples
  - Stage 2: Remove training examples covered by the complex
  - Repeat until complexes are found to cover all examples of the class (but no negative examples)
AQR: Algorithm Overview

- **A beam search:**
  - A series of hill-climb searches in parallel
  - Only specialisations that exclude negative examples while still covering a “seed” positive example are considered
AQR: Algorithm Overview

**Procedure AQR**(POS, NEG) returning COVER:

- let COVER be the empty cover;
- while COVER does not cover all positive examples in POS
  - select a SEED, i.e. a positive example not covered by COVER;
  - call procedure STAR(SEED, NEG) to generate the STAR (a set) of
    - complexes that cover SEED but no examples in NEG;
  - select the best complex BEST from the star according to user-defined criteria;
  - add BEST as an extra disjunct to COVER;
- return COVER.

**Procedure STAR**(SEED, NEG) returning STAR:

- let STAR be the set containing the empty complex;
- while one or more complexes in STAR covers some negative examples in NEG,
  - select a negative example $E_{neg}$ covered by a complex in STAR;
  - Specialize complexes in STAR to exclude $E_{neg}$ by:
    - let EXTENSION be all selectors that cover SEED but not $E_{neg}$;
    - let STAR be the set $\{ x \land y | x \in STAR, y \in EXTENSION \}$;
    - remove all complexes in STAR subsumed by other complexes in STAR;
    - Remove the worst complexes from STAR
      - until size of STAR is less than or equal to user-defined maximum (maxstar);
- return STAR.
AQR: Heuristic Functions

- Heuristic for determining “best complex” to add as a new disjunct to COVER:
  - Maximise number of positive examples covered
  - Because we know that no negative examples are covered and at least one positive example is covered

- Heuristic for determining “worst complexes” to remove from STAR:
  - Minimum sum of positive examples covered and negative examples excluded
  - Optimises search by excluding poorly performing complexes

- In the case of ties:
  - Complexes with fewer selectors are considered better
  - Favours less complex conditions
AQR: Heuristic Functions

- Positive example (SEED) selection:
  - Randomly chosen

- Negative example ($E_{neg}$) selection:
  - Closest to SEED selected first
  - Distance = number of different attribute values in SEED and $E_{neg}$
  - If SEED and $E_{neg}$ have same attribute values (distance = 0), $E_{neg}$ is ignored
- Evolved from earlier AQ algorithm
- Current Version (6.1)
  - [http://www.cs.utexas.edu/users/pclark/software.html](http://www.cs.utexas.edu/users/pclark/software.html)
- Designers (1989):
  - Peter Clark
  - Tim Niblett
- Implementers:
  - Robin Boswell
  - Rick Kufrin
  - Johannes Fuernkranz
Further Reading

**CN2: Algorithm Output**

- **Ordered** set of decision rules (decision lists)
  - Same structure as for AQR
  - Unordered rules also possible (see further reading)

- **Classification of new examples**
  - Apply rules in order to example
    - If a rule covers the example, its class is predicted
    - Therefore, order is important
  - If no rules cover the example, default rule predicts most commonly occurring class in the set of training examples
Problem with AQ algorithm:
- Doesn’t deal with noise well
- Only searches space of complexes completely consistent with training data

Beam search of AQR algorithm, BUT
- Remove dependence on specific examples during the search
- Extend search space to include rules that do not perform perfectly on training data
Achieve this by:

- Broadening specialisation
  - Consider all specialisations of a complex
  - Not just those excluding all negative examples while including a seed positive example

- Allows for a cut-off method
  - Stop specialisation once no further statistically significant specialisations are possible
CN2: Algorithm Overview

- Performed for entire training set
- Build a statistically significant “best” complex for a class C
- Remove examples covered by “best” complex
- No more significant “best” complexes to be found
- STAR covers all examples
- Specialise every complex in STAR using every possible condition
- Find “best” complex of all that are statistically significant
- Prune “worst” complexes before specialising again

```
Let: E be a set of training examples;

Procedure CN2(E) returning RULE_LIST:
let RULE_LIST be the empty list;
repeat
  let BEST_CPX be Find_Best_Complex(E);
  if BEST_CPX is not nil then
    Let E' be the examples covered by BEST_CPX;
    Remove from E the examples E' covered by BEST_CPX;
    Let C be the most common class of examples in E';
    Add the rule ‘if BEST_CPX then class=C’ to the end of RULE_LIST;
  until BEST_CPX is nil or E is empty.
return RULE_LIST.

Procedure Find_Best_Complex(E) returning BEST_CPX:
let the set STAR contain only the empty complex;
let BEST_CPX be nil;
let SELECTORS be the set of all possible selectors;
while STAR is not empty, specialize all complexes in STAR as follows:
  let NEWSTAR be the set {x AND y | x ∈ STAR, y ∈ SELECTORS};
  Remove all complexes in NEWSTAR that are either in STAR (i.e., the unspecialized ones) or are null (e.g., big = y AND big = n);
  for every complex C in NEWSTAR:
    if C is statistically significant when tested on E and better than
    the best complex according to user-defined criteria when tested on E,
    then replace the current value of BEST_CPX by C;
  repeat remove worst complexes from NEWSTAR
  until size of NEWSTAR is <= user-defined maximum;
let STAR be NEWSTAR;
return BEST_CPX.
```
CN2: Algorithm Overview

- Specialisation of continuous attributes
  - Divide attribute range into discrete sub-ranges
  - Selectors test whether a value is > or ≤ values at sub-range boundaries

- Unknown values
  - Replace unknown values with their attribute’s most commonly occurring value in training data
  - For continuous values, replace with mid-range of most commonly occurring sub-range for the attribute, in training data
Quality of complexes

- Indicates accuracy when predicting majority class covered
- Used when BEST_CPX is evaluated against new complexes and when pruning STAR

Set $E'$ of examples which complex covers

- Probability distribution $P = (p_1, \ldots, p_n)$ of examples in $E'$ among classes

- \( Entropy = -\sum_{i} p_i \log_2(p_i) \)
- Lower entropy = better complex
**Significance of a complex**

- Used when BEST_CPX is evaluated against new complexes
- Indicates whether complex represents a pattern unlikely to have occurred by chance
  - i.e. Reliability of complex
- Comparison of observed class distribution of examples satisfying complex with expected class distribution if complex selected examples randomly
  - If difference too great to be put down to chance, CN2 concludes that the complex is significant
CN2: Heuristics

- Significance of a complex
  - Likelihood ratio statistic
    - \[ \sum_{i=1}^{n} f_i \log(f_i/e_i) \]
    - \( F = (f_1, f_2, \ldots, f_n) \) is observed class frequency distribution of examples covered by complex
    - \( E = (e_1, e_2, \ldots, e_n) \) is expected class frequency distribution of examples covered by randomly classifying complex
    - \( E \) calculated according to the probability distribution of examples from entire training set
    - Higher score for statistic indicates a complex with greater significance