The SeCo-Framework for Rule Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Knowledge Engineering

Frederik Janssen and Markus Zopf, Technische Universität Darmstadt, Germany

Motivation

- most rule learning algorithms use the separate-and-conquer strategy
- algorithms of this kind can be decomposed into different components according to [3]
- using an abstract algorithm lets the user configure many existing and new algorithms
- often new work consists only of the exchange of a single component of an algorithm

Components

interface	purpose
BinarizationMode	used to determine the type of binarization (ordered or unordered 1vs-all)
ClassificationMode	selects the type of classification (decision list or (weighted) voting)
RuleInitializer	used to initialize the rule (either empty or by selecting a random example)
CandidateSelector	selects the candidate rules for the next iteration (for implementing various search algorithms)
Heuristic	the heuristic to evaluate all candidate rules
RuleRefiner	creates refinements of rules (bottom up, top down, or bidirectional)
RuleFilter	filters out unpromising candidates
StoppingCriterion	used to stop the refinement of a candidate rule (usually when it does not cover any negatives)
RuleStoppingCriterion	stops the induction of rules (e.g., significance tests)
PostProcessor	implements the post processing method of RIPPER [2]

Example configuration of *Ripper*

```
<seco growingSetSize="2/3" minNo="2" weighted="false" seed="0">
  <binarizationmode classname="OrderedBinarization"/>
  <classificationmode classname="DecisionList"/>
  <ruleinitializer classname="TopDownRuleInitializer"/>
  <candidateselector classname="SelectAllCandidatesSelector"/>
  <heuristic classname="FoilGain"/>
  <rulerefiner classname="TopDownRefiner"
    nominal.cmpmode="equal"/>
  <stoppingcriterion classname="NoNegativesCoveredStop"/>
  <rulestoppingcriterion classname="MDLStoppingCriterion"/>
  <rulefilter classname="BeamWidthFilter" beamwidth="1"/>
  <postprocessor classname="PostProcessorRipper"
    optimizations="2"/>
</seco>
```

Capabilities

- Rule learning algorithms
  - currently: Ripper [2], CN2 [1], AQ [5], and SimpleSeCo [4]
  - but nearly every separate-and-conquer rule learning algorithm can be configured
- Weighted covering, search strategies, and heuristics
  - weighted covering is possible
  - different search strategies: top-down, bottom-up, bidirectional, hill-climbing, beam search, and exhaustive search
  - heuristics (excerpt): Laplace, Weighted Relative Accuracy, Correlation, Odds Ratio, Relative Cost Measure, *F*-Measure, *m*-estimate, Klösgen Measure, LinearRegression [4]
- Evaluation package
  - including various output formats, serialization of rule sets, and statistical tests (Friedmann & Nemenyi)

Conclusions and future work

- building block architecture eases the implementation of new rule learning algorithms
- Evaluation package simplifies the comparison of different algorithms
- Future Work
  - implement more rule learning algorithms
  - include an interactive component
  - graphical user interface
  - **publish the framework**

References

- [1] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–283, 1989.
- [2] William W. Cohen. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 115–123, 1995.
- [3] Johannes Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.
- [4] Frederik Janssen and Johannes Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379, March 2010. DOI 10.1007/s10994-009-5162-2.
- [5] R. S. Michalski. On the Quasi-Minimal Solution of the Covering Problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3 (Switching Circuits), pages 125–128, Bled, Yugoslavia, 1969.