Refinement of Rule Sets with JoJo

Dieter Fensel and Markus Wiese

Institut für Angewandte Informatik und Formale Beschreibungsverfahren,
University of Karlsruhe, P.O. Box 6980, 7500 Karlsruhe, Germany
c-mail: fensel@aifb.uni-karlsruhe.de

Abstract. In the paper we discuss a new approach for learning classification rules
from examples. We sketch out the algorithm JoJo and its extension to a four step
procedure which can be used to incrementally refine a set of classification rules.
Incorrect rules are refined, the entire rule set is completed, redundant rules are de-
leted and the rule set can be minimized. The first two steps are done by applying
JoJo which searches through the lattice of rules by generalization and specializa-
tion.

1 Introduction

Learning from example deals with the task of learning general descriptions (decision
trees or rules) from examples. In the paper we discuss a four step procedure which can
be used to refine a set of classification rules. First, the rules which become incorrect be-
cause of new negative examples are refined. Each incorrect rule is replaced by a set of
rules which covers the positive examples but not the new negative ones. In a second
step, the rule set is extended to cover the new positive examples. Third, redundancy
of the rule set is corrected by deleting rules. In a fourth step, a minimal subset can be com-
puted which covers all positive examples. Steps one and two are carried out by applying
the JoJo-algorithm. The main feature of JoJo is that it integrates generalization and spe-
cialization into one heuristic search procedure.

The version space algorithm is one of the earliest algorithms which obtained classifica-
tion rules from a set of examples [13]. It applies a dual search strategy in the lattice of
possible rules. If a negative example is given, all general rules which cover it are spe-
cialized. If a positive example is given, all special rules which do not cover it are gen-
eralized. The search procedure starts at the top and the bottom of the lattice and
converges to the final rule which covers all positive and no negative examples. The ver-
sion space algorithm does a complete search and can therefore only be applied to small
data sets because it is impossible to find the minimal hypothesis (Occam’s Razor) consis-
tent with the given examples in polynomial time [1].

Heuristic (i.e. incomplete) search procedures like AQ [14], C4 [16], CN2 [3], CABRO
[11], FOIL [16], and PRISM [2] work by specialization only. They start with very gen-
eral descriptions and specialize them until they are correct. This is done by adding ad-
tional premises to the rule or by restricting the range of an attribute which is used in a
premise.

In [5] we discussed the heuristic search procedure RELAX which works by generaliza-
tion only. It starts with very special descriptions and generalizes them as long as they
are not incorrect. It regards every example as a very specific rule which is generalized.
This is done by deleting premises from the rule. The motivation for this procedure are
algorithms used for minimizing electronic circuits [12].

Algorithms which use specialization as search strategy generally have the problem of
overspecialization. In an i-th specialization, a specialization can performed which could
Procedures which can only specialize have a predefined starting point for the search. They must start as generally as possible, because a possible solution could not be found if it is more general than the starting point. Similarly, procedures working only with specialization must start as specifically as possible. JoJo is able to start at an arbitrary point in the lattice because it can use both search directions.

A starting point can be described by two parameters, its vertical position (the length of the description) and its horizontal position (the chosen attributes).

Criteria for choosing a vertical position:
- An expert can approximate the possible length of the rules or has some experience from earlier program runs.
- Rules are produced randomly for every length and the distribution of their quality is used to decide the position.
- The procedure starts with a small sample or very limited resources and arbitrary positions in order to find a good starting point.
- The starting point is randomly chosen. In the average case this is no worse than always starting with the bottom or top element as other procedures do.
- Heuristic: Few positive examples and maximal-specific descriptions indicate long rules, whereas few negative examples and maximal-general descriptions indicate short rules.

Criteria for choosing a horizontal position:
- If the vertical position is chosen, the premises with the highest correlation to the goal concept (or the combination of premises if this is not to expensive) can be selected.

In general, it is possible to carry out several program runs with different starting points. Rules which were already produced by JoJo or other algorithms can be used as starting points for further refinement and improvement.

2.2 Search Process in the Lattice

The gist of JoJo consists of three components: a generalizer, a specializer, and a scheduler.

The generalizer computes, validates and orders the descriptions which can be reached by the next generalization step using a predefined generalization strategy and a predefined preference criterion (g-preference). An example of a simple generalizer is H-RE-LAX [6]:
- Conjunctions are generalized by deleting a premise.
- The g-preference applied is:
  \[
  1 - \frac{\text{number of covered negative examples}}{\text{number of all covered examples}} + 0.5
  \]

The specializer computes, validates and orders the descriptions which can be reached by the next specialization step using a predefined specialization strategy and a predefined preference criterion (s-preference). An example of a simple specializer is:
- Conjunctions are specialized by adding a premise.

---

1. A possible quality criterion is the average correctness of the rules with the same length.
2. It is possible to check rules for overspecialization when that rules are produced by an algorithm using specialization as search direction.
3. 0.5 is added in order to prevent division by zero and to favor rules which cover more examples when several rules do not cover any negative example.
Alternatively, a rule can be replaced by a set of rules which covers a subset of the examples which were covered by the old rule. Therefore, we apply JoJo for the sub-space which is defined by the incorrect rule.

![Cover of the incorrect rule](image)

- Covered negative examples + Covered positive examples

**Figure 2.** Specializing an incorrect rule by a set of rules.

This is done for each incorrect rule. Then the incorrect rules are replaced by the corresponding rule sets.

### 3.2 Completeness

After this step every rule is correct in the sense that no rule covers more negative examples than is specified by the threshold. In a second step, the set of rules must be completed. This is done by applying JoJo to the set of uncovered positive example and all negative examples. JoJo computes new rules which are correct. These rules are computed as long as the number of positive examples which remain uncovered is larger than that which is specified by a threshold.

### 3.3 Non-Redundancy and Minimality

In this step, rules are deleted which are more special than other rules. This is done by checking the subset relation between the sets of premises of the rules. A rule is more special than a second one if its set of premises is a superset of the second one. In a fourth (optional) step a minimal set of complexes is computed. A minimized set of rules has the same cover as the original one but consists of a minimal number of complexes. Because this problem is NP-complete we apply the following heuristic search:

1. Look for the best rule of the rule set (i.e. the one which covers most examples).
2. Remove all positive examples which are covered by this rule.
3. Add the best rule to the final rule set.
4. If more examples remain than are specified by a threshold go to step 1.

### 4 Implementation and Empirical Evaluation of JoJo

JoJo is implemented in C and available under Sun-Unix and MS-DOS. It is integrated into the RJ-environment [9] which preprocesses unknown-values, ordinal, multi-valued, and continuous attributes. It has been tested with several data sets of the machine learning library and with data sets of the ESPRIT project StatLog (data sets with more than 10,000 objects and 40 attributes). The major result of these tests is its ability to learn very brief, i.e. general, descriptions of classes (cf. [8]) than algorithms like CN2 or C4.