A Random Forest Approach to Relational Learning

Anneleen Van Assche[a] Celine Vens[a] Hendrik Blockeel[a]

Sašo Džeroski[b]

[a] Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
[b] Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

The full version of this paper appeared in: *Proceedings of ICML-2004 workshop on Statistical Relational Learning*. Banff, Canada, 2004.

Introduction This paper presents an initial exploration of the use of random forests in a relational context. Random forest induction is a bagging method that builds decision trees by selecting in each node the "best" feature, not out of all available features, but out of a random subset of features (which may be different for each node). The motivation for this work is based on two observations.

On the one hand, random forests have been shown to work well when many features are available. Hence, their use seems especially interesting for relational data mining, for which it is typical that there is a large number of features, many of which are expensive to compute.

On the other hand, using random forests allows an extension of the feature space by including aggregate functions, possibly refined with selection conditions on the set to be aggregated.

This combination of aggregation and selection in relation learning is not a trivial task, because the feature set grows quickly, and because the search space is less well-behaved due to the non-monotonicity problem. However, the use of random forests tackles both problems.

A Random Forest Approach to Relational Learning Our implementation started from Tilde, which is included in the ACE data mining system. Tilde is a relational top-down induction of decision trees (TDIDT) algorithm, using logical queries in the tree nodes.

A filter that allows only a random subset of the tests to be considered at each node, was built in. Around this procedure a bagging wrapper was built. These two mechanisms together result in a random forest induction method.

Moreover, the feature set considered at each node in the tree was expanded to consist of the regular features (with aggregate conditions included), augmented with any refinements of aggregate conditions used on the path from the root to the current node. For example, consider the following concept, which covers persons X if they have more than two children:

```
a(X) :- count(Y, child(X,Y), C), C>2.
```

The concept can be refined by adding a literal inside the count aggregate, to cover persons with more than two daughters:

```
a(X) :- count(Y, (child(X,Y), female(Y)), C), C>2.
```

We want to point out that the use of random forests tackles the difficulties that arise when combining aggregation and selection in relational learning. First, the size of the search space is limited because only a subset of the possible features at each node in a tree is considered. Second, using decision trees, the comparison operators "<" and ">" are equivalent up to switching branches. This means that we can restrict ourselves to using monotone aggregate conditions.

Experimental Results and Conclusions The strength of random forests and the use of (refined) aggregates were experimentally validated in a relational setting. Experiments were performed along three dimensions, both on a business domain (Financial) and a structurally complex data set (Mutagenesis). To examine the influence of the number of trees in the random forests, we experimented with 3, 11, and 33 trees. To determine the optimal degree of randomness, we used a sample size of 100% down to 10% and the square root of the number of features. To investigate the performance of random forests in the context of aggregation, we performed experiments without aggregates, with (simple) aggregates and with refined aggregates, where conditions on the set to be aggregated are added.

Figure 1 clearly shows the benefit of adding more trees. Concerning the size of the sample set, if we randomly decrease the feature set at each node in a tree down to a certain level, the classification performance is at least as good as bagging, so we profit from the gain in efficiency. In our chosen data sets, the optimal level turned out to be 25% of the features, below this threshold classification performance decreased. Including aggregates yielded a large performance boost. Refining aggregates added another slight improvement. However, some effects are still unclear and therefore, further investigation is needed.

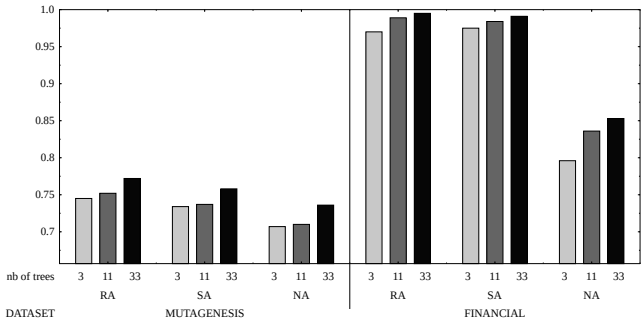

Figure 1: Accuracy for random forests using 25% of the features. Results are shown for refined aggregates (RA), simple aggregates (SA) and not using aggregates (NA) on the two data sets, both with 3, 11 and 33 trees.