A Random Forest Approach to Relational Learning

Anneleen Van Assche ANNELEEN.VANASSCHE@CS.KULEUVEN.AC.BE
Celine Vens CELINE.VENS@CS.KULEUVEN.AC.BE
Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.AC.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Sašo Džeroski SASO.DZEROSKI@IJS.SI
Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

Random forest induction is an ensemble method that uses a random subset of features to build each node in a decision tree. The method has been shown to work well when many features are available. This certainly is the case in relational learning, especially when aggregate functions, combined with selection conditions on the set to be aggregated, are included in the feature space. This paper presents an initial exploration of the use of random forests in a relational context. We experimentally validated our approach both in a business domain, and on a structurally complex data set.

1. Introduction

The motivation for this paper is based on two observations. First, random forest induction, an ensemble method that builds decision trees using only a random subset of the feature set at each node, has been shown to work well when many features are available (Breiman, 2001). Because relational learning typically deals with large feature sets, it seems worthwhile to investigate whether random forests perform well in this context. Second, using random forests allows an extension of the feature space by including aggregate functions, possibly combined with selection conditions. So far, this combining has been considered a difficult task (Blockeel & Bruynooghe, 2003), because the feature set grows quickly, and because the search space is less well-behaved due to the non-monotonicity problem (Knobbe et al., 2002).

The paper is organized as follows. In Section 2, we discuss random forests. Section 3 illustrates the problem of combining aggregates with selection conditions in relational learning. Our method, which is a random forest approach to relational learning, is presented in Section 4. In Section 5, we experimentally evaluate our method in both a structurally complex domain, and a (highly non-determinate) business domain. Finally, we formulate conclusions and some ideas for future research in Section 6.

2. Random Forests

Random forest induction (Breiman, 2001) is an ensemble method. An ensemble learning algorithm constructs a set of classifiers, and then classifies new data points by taking a vote of the predictions of each classifier. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members, is that the classifiers are accurate and diverse (Hansen & Salamon, 1990). An accurate classifier does better than random guessing on new examples. Two classifiers are diverse if they make different errors on new data points. There are different ways for constructing ensembles: bagging (Breiman, 1996) and boosting (Freund & Schapire, 1996) for instance, introduce diversity by manipulating the training set. Several other approaches attempt to increase variability by manipulating the input features or the output targets, or by introducing randomness in the learning algorithm (Dietterich, 2000). Random forests try to increase diversity among the classifiers by resampling the data, and by changing the feature sets over the different (tree) model induction processes. The exact procedure is as follows:

- **for** $i = 1$ **to** k **do**:
  - build dataset D_i by sampling with replacement from dataset D
  - learn a decision tree T_i from D_i *using randomly restricted feature sets*

- make predictions according to the majority vote of the set of k trees

The part of the algorithm where random forests differ from the normal bagging procedure is emphasized. When inducing a decision tree, the best feature is selected from a fixed set of features F in each node. In bagging, this set of features does not vary over the different runs of the induction procedure. In random forests however, a different random subset of size $f(|F|)$ is considered in each node (e.g. $f(x) = 0.1x$ or $f(x) = \sqrt{x}$, ...), and the best feature from this subset is chosen. This obviously increases variability. Assume for instance that $f(x) = \sqrt{x}$, and that two tests t_1 and t_2 are both good features for the root of the tree, say t_1 is the best and t_2 is the second best feature. With a regular bagging approach t_1 is consistently selected for the root, whereas with random forests both t_1 and t_2 will occur in the root nodes of the different trees respectively with frequency $1/\sqrt{|F|}$ and $1/\sqrt{|F|} - 1/|F|$. So t_2 will occur only with slightly lower frequency than t_1.

Random forests have some interesting properties (Breiman, 2001). They are more efficient since only a sample of $f(|F|)$ features needs to be tested in each node, instead of all features. They also do not overfit as more trees are added. Furthermore, they are relatively robust to outliers and noise, and they are easily parallellized.

The efficiency gain makes random forests 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, relational data mining offers an interesting test suite for random forests, exactly because the advantage of random forests is expected to become more clear for very large feature spaces. In relational data mining, such data sets abound. Moreover, using random forests allows us to even enlarge the feature set by including aggregate functions, possibly refined with selection conditions. We discuss this in the next section.

3. Aggregates and Selection in Relational Learning

When considering multi-relational learning, sets (or more general, bags) need to be handled. They are represented by one-to-many or many-to-many relationships in or between relations. Blockeel and Bruynooghe (2003) categorize current approaches to relational learning with respect to how they handle sets. Whereas ILP (Muggleton, 1992) is biased towards selection of specific elements, other approaches, such as PRM's (Koller, 1999) or propositionalisation approaches (e.g. the one by Krogel and Wrobel (2001)) use aggregate functions, which compute a feature of the set that summarizes the set. The latter methods are optimized for highly non-determinate (e.g. business) domains, whereas the former is geared more towards structurally complex domains, e.g. molecular biology, language learning, etc.

Perlich and Provost (2003) present a hierarchy of relational concepts of increasing complexity, where the complexity depends on that of any aggregate functions used. They argue that ILP is currently the only approach that can explore all concepts in the hierarchy. However, combining aggregates and selection in ILP is not a trivial task. Until now ILP-like feature construction and the use of aggregation features have mostly been combined in relatively restricted ways, e.g., without allowing complex conditions within the aggregation operator.

There are several difficulties that arise. First of all, while approaches such as ILP by themselves already suffer from a large search space, integrating aggregation operators into them further increases the hypothesis space (Blockeel & Bruynooghe, 2003). Moreover, the search space is less well-behaved because of the problem of non-monotonicity (Knobbe et al., 2002), which renders traditional pruning methods in ILP inapplicable. Clearly, the idea of performing a systematic search for a good hypothesis has to be given up; the search will have to be more heuristic.

Let us explain the reduction in pruning opportunities. Suppose refining a hypothesis h consists of adding a condition into its body. Then refining h can only decrease its coverage, so if any h is unsuitable because it has too low coverage, then any refinements of h must be unsuitable as well; hence we can prune the search space at h. If conditions are added within an aggregate, though, the coverage of the hypothesis may increase instead of decrease, hence such pruning cannot safely be applied any more.

We illustrate the above with an example:

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

adding a literal inside the count aggregate gives

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

which must have at most the same coverage (people with more than two daughters must be a subset of people with more than two children). If we consider

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

and its refinement

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

then the examples covered by the refinement are a superset of the original set (people with at most two children certainly have at most two daughters, but people with at most two daughters may have more than two children).

Knobbe et al. (2002) present an approach to combine aggregates with reasonably complex selection conditions by generalizing the selection graph pattern language. Selection graphs are a graphical description of sets of objects in a multi-relational database.

In this paper we explore an alternative approach which is based on random forests and explores the same feature space as Knobbe et al. (2002).

4. A Random Forest Approach to Relational Learning

Knobbe et al. (2002) provide some definitions that are useful for our explanation. An *aggregate condition* is a triple (f, o, v) with f being an aggregate function, o a comparison operator and v a value of the domain of f. An aggregate condition is *monotone* if, given sets of records S and S', $S' \subseteq S, f(S')\ o\ v \Rightarrow f(S)\ o\ v$. The example in the previous section showed that $(count, >, v)$ is a monotone aggregate condition, while $(count, <, v)$ is not.

As mentioned in Section 3, Knobbe et al. (2002) describe a method for including aggregate functions in selection graphs. If an aggregate condition is monotone, selection conditions on the set to be aggregated are allowed.

We implemented a similar method in Tilde (Blockeel & De Raedt, 1998), which is included in the ACE data mining system (Blockeel et al., 2002). Tilde is a relational top-down induction of decision trees (TDIDT) instantiation, and can be viewed as a first order upgrade of C4.5 (Quinlan, 1993), using logical queries in tree nodes.

We expanded the feature set considered at each node in the tree to consist of the regular features (with aggregate conditions included), augmented with any refinements of aggregate conditions (such as the ones discussed in the example of Section 3) used on the path from the root to the current node, when the "yes"- branch was taken.

Moreover, we built in a filter that allows only a random subset of the tests to be considered at each node[1]. Around this procedure we built a wrapper in order to get bagging. These two mechanisms together result in a random forest induction method.

We want to point out that the use of random forests tackles the difficulties that arise when combining aggregation and selection in ILP. First, the size of the search space is limited because we only consider a subset of the possible features at each node in a tree. 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. However, even if we would allow non-monotone aggregate conditions, the refinements of these aggregates wouldn't be chosen to split a node, because they don't add any extra information to the tree.

Another advantage of using random forests is that more randomly generated features can be included in the feature space. This includes the construction of aggregate conditions, where the aggregate clause consists of a number of conjuncts. This way, tests as

```
count(Y, (child(X,Y),female(Y),blonde(Y)), C), C>3
```

could be immediately considered at the root node, while now they can only be found as a refinement of an aggregate already occuring in the tree. Non-monotone aggregate functions could then be refined by deleting a number of conjuncts. This would lead to a bottom-up search strategy included in our top-down approach, and would be very interesting to explore. However this is not yet implemented in our system.

5. Experimental results

The aim of our experiments is to investigate the benefit of random forests in a relational setting where the feature set is expanded with aggregates and even refinements of aggregates, both in a business domain and a structurally complex data set. In the following sections we first describe our experimental setup, and then present results on two well-known data sets: the Mutagenesis data set (Srinivasan et al., 1999) and the Financial data set (Berka, 2000).

5.1. Experimental setup

All experiments were performed using a 5-fold cross-validation and were carried out five times (with the same folds). Afterwards the results were averaged in order to obtain a more reliable estimate of the per-

[1] This actually differs from the definition in (Breiman, 2001) where a random subset of the attributes, instead of the tests, is chosen.

formance of the random forests. Different parameters needed to be set. First of all, the number of random features in each node: we chose to consider random subsets of 100%, 75%, 50%, 25%, 10%, and the square root of the number of tests at each node in the trees. We have also examined the influence of the number of trees in the random forests, experimenting with 3, 11, and 33 trees. To investigate the performance of random forests in the context of aggregation, we have performed experiments with and without the use of aggregates, and also with refinement of aggregates, where conditions on the set to be aggregated are added.

5.2. Mutagenesis

For our first experiment we used the Mutagenesis data set. This ILP benchmark data set, introduced to the ILP community by Srinivasan et al. (1999), consists of structural descriptions of 230 molecules that are to be classified as mutagenic (60%) or not. The description consists of the atoms and the bonds that make up the molecule. For this data set we expect only a slight gain in accuracy using aggregates, since Mutagenesis does not have many numerical attributes nor is it very non-determinate.

Table 1 shows the results obtained on the Mutagenesis data. For each column the best result is in bold. For the experiments with refined aggregates, we can see that the best result is always one where random subsets of tests are used. In fact, we see that applying random forests using random samples with down to 25% of the possible tests gives either an improvement or at least no significant decrease in accuracy over bagging[2], while it certainly results in an efficiency gain. As expected it also clearly outperforms Tilde.

For the results without the use of aggregates we see that random forests don't seem to enhance accuracy over bagging, while still performing better than Tilde. This may show that random forests indeed profit from large feature sets. These results also support Breiman's (2001) statement that random forests do not tend to overfit. The average difference between training and test set error for random forests using randomly 10% of the tests at each node is 6% while for Tilde (using all features) it is 16%. As was expected, we also found that adding more trees to the forest, clearly increases accuracy in all cases. In Figure 1 the results for a random forest with 33 trees either making use of refined aggregates (RA) or aggregates without refinement (WA) or using no aggregates (NA) are compared for different sizes of random subsets in the nodes

[2]In Table 1 bagging is corresponding to random forests with 100% of the tests used in the node.

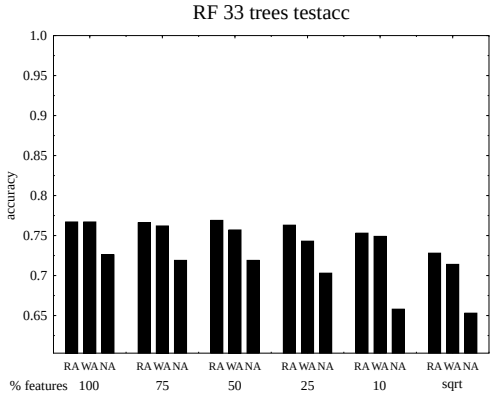

Figure 1. Accuracy for random forests consisting of 33 trees, either not using aggregates (NA), using aggregates (WA) or refined aggregates (RA) on the Mutagenesis data. Results are shown for different numbers of randomly chosen features.

of the trees (100% down to square root of the number of features at the node). As we can see from Figure 1 there is always (no matter which number of features is used) an improvement when aggregates are added. When we allow refinements of the aggregates in the queries, in many cases another slight improvement is observed. An example of a test which was frequently found across the different trees was the following aggregate

```
count(BId, (bond(BId,Mol,At1,At2,Tp)), C), C>28.
```

with refinement

```
count(BId, (bond(BId,Mol,At1,At2,Tp),
           atom(Mol,At1,carbon)), C), C>28.
```

The first part of the example represents the set of all molecules that have at least 28 bonds. This aggregate was also found to be a good test by Knobbe et al. (2002). The refinement of that aggregate describes all molecules that have at least 28 bonds connected to an atom of type carbon.

5.3. Financial Data Set

Our second experiment deals with the Financial data set, from the discovery challenge organized at PKDD '99 and PKDD '00 (Berka, 2000). This data set involves learning to classify bank loans into good (86%) and bad loans. The data set consists of 8 relations and contains for each loan customer information and account information which includes permanent orders and several hundreds of transactions per account. This

	Tilde			RF 3 trees			RF 11 trees			RF 33 trees		
	RA	WA	NA	RA	WA	NA	RA	WA	NA	RA	WA	NA
1	0.717	0.730	0.683	0.721	0.710	**0.704**	0.754	0.745	**0.718**	0.767	**0.767**	**0.726**
0.75	0.723	0.728	**0.687**	0.725	0.730	0.677	0.758	**0.750**	0.704	0.766	0.762	0.719
0.50	**0.728**	0.710	0.678	**0.735**	0.736	0.690	**0.763**	0.743	0.710	**0.769**	0.757	0.719
0.25	0.724	**0.744**	0.670	0.728	**0.737**	0.700	0.753	0.743	0.703	0.763	0.743	0.703
0.10	0.717	0.727	0.653	0.723	0.720	0.657	0.753	0.735	0.651	0.753	0.749	0.658
sqrt	0.701	0.719	0.654	0.693	0.700	0.638	0.707	0.710	0.668	0.728	0.714	0.653

Table 1. Accuracy results on the Mutagenesis data set. The rows describe the different proportions of random test subsets in the nodes. The columns compare accuracies for experiments using refined aggregates (RA), aggregates without refinement (WA) and no aggregates (NA) for Tilde and random forests with different number of trees.

	Tilde			RF 3 trees			RF 11 trees			RF 33 trees		
	RA	WA	NA	RA	WA	NA	RA	WA	NA	RA	WA	NA
1	0.978	0.978	0.746	0.978	0.985	0.839	0.988	0.986	0.849	0.987	0.988	0.853
0.75	0.980	0.984	0.768	0.981	0.980	0.833	0.985	0.984	0.847	0.987	0.988	0.853
0.50	**0.985**	0.981	0.766	0.983	0.983	0.823	0.990	**0.989**	0.854	0.987	0.988	0.862
0.25	0.978	**0.984**	0.752	**0.991**	**0.990**	0.847	**0.990**	0.986	0.855	**0.992**	**0.990**	0.859
sqrt	0.954	0.949	0.766	0.959	0.970	0.847	0.972	0.983	0.865	0.984	0.986	0.864
0.10	0.950	0.972	**0.827**	0.957	0.947	**0.863**	0.964	0.964	**0.866**	0.964	0.957	**0.866**

Table 2. Accuracy results on the Financial data set. The rows describe the different proportions of random test subsets in the nodes. The columns compare accuracies for experiments using refined aggregates (RA), aggregates without refinement (WA) and no aggregates (NA) for Tilde and random forests with different number of trees.

problem is thus a typical business data set which is very non-determinate.

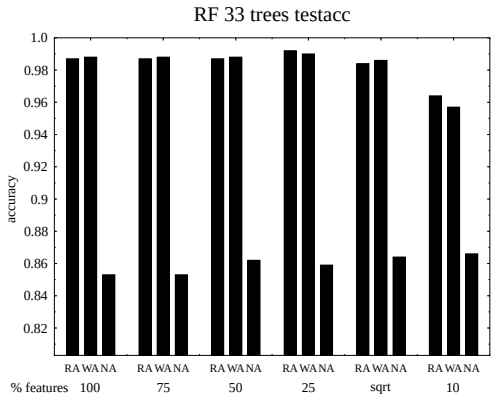

Figure 2. Accuracy for random forests consisting of 33 trees, either not using aggregates (NA), using aggregates (WA) or refined aggregates (RA) on the Financial data. Results are shown for different numbers of randomly chosen features.

Table 2 provides the results obtained on the Financial data. For this data set we want to point out that the data distribution is quite skewed; 86% of the examples are positive. We also note that in this case the square root of the number of tests is on average larger than 10%, so we switched these two rows in Table 2. We observe that for all experiments, using only a random part of the tests to select the best test (certainly down to 25% of the tests) seems to be advisable since there is no drop in accuracy (no significant gain either) and we profit from the efficiency gain by testing fewer features in each node. Contrary to the results on the Mutagenesis data, when no aggregates are used, the accuracy seems to increase while adding more randomness. This is actually due to the skewness of the Financial data: without randomness the predictive accuracy of the forests is below the accuracy of allways predicting the default value (86%). When adding more randomness, some trees tend to allways predict the default value, or at least predict a very small proportion of the examples to be negative. So this might improve accuracy up to the default, but the trees become certainly less informative.

Adding aggregates resulted in much more compact trees, and, as can be seen in Figure 2, yielded a large improvement of accuracy. Allowing refinements in these aggregates didn't seem to improve accuracy further though. This may be due to the fact that the theory that needs to be learned, can be represented very well without complex selection within the aggregates. We also found that among the trees aggregates were rarely refined.

5.4. Experimental conclusions

For the experiments using aggregates, decreasing the number of random tests considered in a node to a certain threshold slightly increases the predictive accuracy. If we go below that threshold the accuracy de-

creases again. However it seems quite difficult to determine an optimal value for this threshold. For the chosen data sets, sampling randomly around 25% of the tests at each node, gives either an improvement or at least no significant decrease in accuracy. Breiman (2001) on the other hand, obtained improvements with even much smaller proportions of features in the propositional case. Also the average improvement was much higher. This difference might be due to fact that in our approach a random subset of tests is taken while Breiman takes a random subset of the attributes, and selects the best test using these attributes. Or it might show that in relational learning random forests do not work as well as in propositional learning.

We also found that the results without aggregates differ a lot over the two data sets: on Mutagenesis decreasing the random sample of tests decreases accuracy while on the Financial data set the accuracy increases. This last effect was because of the skewness of the data.

As was expected, adding more trees to the forest indeed increases accuracy. Of course more trees mean longer runtimes, so there is always a trade-off between efficiency and accuracy.

Concerning aggregation and selection, allowing aggregate functions in the tests always gave an improvement, whereas additional refinement of the aggregates didn't consistently increase accuracy.

While in general performing at least as well as bagging, random forests are computationally more efficient than bagging. If we compare random forests, consisting of 33 trees and using 25% of the features, with bagging using 33 trees, we found an efficiency gain of factor 2.6 on the Mutagenesis data and a factor 1.3 on the Financial data. Again this factor is smaller than in the propositional case, where there is no need to generate tests at each node, since the set of tests remains the same for all nodes. In the relational case on the other hand constructing a new node in the tree consists of two steps: first all possible refinement queries need to be generated and then secondly those queries are executed on the remaining examples and for each query a heuristic is computed. Afterwards, based on those heuristics the best test is placed in the new node. When using random forests the time for executing the queries and calculating the heuristics is reduced, since only a sample of all generated refinements is considered. But in our current implementation the time for the first step remains the same, because all possible refinements are still generated and only afterwards a random sample of that set is taken.

6. Conclusions and Future Work

In this paper, we have explored a random forest approach to relational learning. Random forests have been shown to work well when many features are available. This is often the case in relational learning. Thus the investigation of random forests in this context seems worthwhile.

Moreover, when using random forests, we can further enlarge the feature space by including aggregate conditions and refining them with selection conditions on the set to be aggregated. This combination of aggregation with selection conditions is not a trivial task, because the feature space grows quickly and the search space is less well-behaved due to the non-monotonicity problem. However, the use of random forests overcomes these problems.

We experimentally validated the strength of random forests and the use of (refined) aggregates in the relational case. Our results show that, 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, this level turned out to be 25% of the features, below this threshold classification performance decreased. The benefit of including aggregate functions was clear from the results. Refinements of aggregates sometimes yielded another slight performance improvement. However, some effects are still unclear and therefore, more data sets should be explored.

Now we turn to three possible directions for future work. As mentioned in Section 4, we would like to introduce more randomly generated features, e.g. aggregate functions with a number of selection conditions on the set to be aggregated. At this moment, these features can only be found as refinements of other aggregate functions. These randomly generated aggregate functions would increase diversity between features, and could be refined by deleting some selection conditions, if the function is non-monotone. This would lead to research into the benefits of bottom-up search strategies included in a top-down approach.

Alternative search heuristics could also be explored. Instead of always choosing the best feature (out of the features in the randomly chosen subset) at each node in a tree, features could be chosen according to a Boltzman distribution. This way, even less good features would have a chance to be chosen; this chance being proportional to their quality. This would increase diversity among the trees, and probably improve the strength of the random forest.

It would also be interesting to investigate the effect of autocorrelation and degree diversity (Jensen et al., 2003) in the context of random forests, since these may influence the extent to which different trees in the forest are independent.

Acknowledgements

Anneleen Van Assche is supported by the Institute for the Promotion of Innovation by Science and Technology in Flanders (I.W.T.-Vlaanderen). Celine Vens is supported by the GOA/2003/08(B0516) on Inductive Knowledge Bases. Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen).

References

Berka, P. (2000). Guide to the financial data set. *The ECML/PKDD 2000 Discovery Challenge.*

Blockeel, H., & Bruynooghe, M. (2003). Aggregation versus selection bias, and relational neural networks. *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, SRL-2003, Acapulco, Mexico, August 11, 2003.*

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence, 101*, 285–297.

Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., & Vandecasteele, H. (2002). Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research, 16*, 135–166.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Dietterich, T. (2000). Ensemble methods in machine learning. *Proceedings of the 1th International Workshop on Multiple Classifier Systems* (pp. 1–15).

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Morgan Kaufmann.

Hansen, L., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Patern Analysis and Machine Intelligence, 12*, 993–1001.

Jensen, D., Neville, J., & Hay, M. (2003). Avoiding bias when aggregating relational data with degree disparity. *Proceedings of the 20th International Conference on Machine Learning.*

Knobbe, A., Siebes, A., & Marseille, B. (2002). Involving aggregate functions in multi-relational search. *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th European Conference* (pp. 287–298). Springer-Verlag.

Koller, D. (1999). Probabilistic relational models. *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 3–13). Springer-Verlag.

Krogel, M.-A., & Wrobel, S. (2001). Transformation-based learning using multi-relational aggregation. *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 142–155).

Muggleton, S. (Ed.). (1992). *Inductive logic programming.* Academic Press.

Perlich, C., & Provost, F. (2003). Aggregation-based feature invention and relational concept classes. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 167–176). ACM Press.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann series in Machine Learning. Morgan Kaufmann.

Srinivasan, A., King, R., & Bristol, D. (1999). An assessment of ILP-assisted models for toxicology and the PTE-3 experiment. *Proceedings of the Ninth International Workshop on Inductive Logic Programming* (pp. 291–302). Springer-Verlag.