Effective Rule Induction from Molecular Structures Represented by Labeled Graphs

Susanne Hoche[1⋆], Tamás Horváth[2,1⋆⋆], and Stefan Wrobel[1,2]

[1] Fraunhofer AiS, Institute for Autonomous intelligent Systems, Schloss Birlinghoven, D-53754 Sankt Augustin,Germany
[2] University of Bonn, Department of Computer Science III, Römerstraße 164, D-53117 Bonn, Germany
{susanne.hoche,tamas.horvath,stefan.wrobel}@ais.fraunhofer.de

Abstract. Acyclic conjunctive queries form a polynomially evaluable fragment of definite nonrecursive first-order Horn clauses. Labeled graphs, a special class of relational structures, provide a natural way for representing chemical compounds. We propose an algorithm specific to learning acyclic conjunctive queries predicting certain properties of molecules represented by labeled graphs. To compensate for the reduced expressive power of the hypothesis language and thus the potential decrease in classification accuracy, we combine acyclic conjunctive queries with constrained confidence-rated boosting. Preliminary experimental results indicate the potential of the method for problems involving labeled graphs.

1 Introduction

Machine Learning is traditionally concerned with the problem of approximating an unknown target function $f : X \to Y$, where the domain or *instance space* X is the Cartesian product of a fixed set of *attributes*. Attributes are usually unordered or linearly ordered sets. Despite the number of successful real-world applications using attribute-value representation of the instances, the need of applying other representation languages in machine learning has long been recognized. One obvious argument has been the problem that attribute-value representation is not appropriate for describing learning tasks involving instances with complex structures. Multi-relational learning, also referred to as Inductive Logic Programming (ILP) [22, 30], is one of the most successful directions among the approaches of considering more expressive representation languages in learning.

In ILP, various classes of first-order languages are used to describe the input (i.e., examples and background knowledge) and output (i.e., hypotheses) components of the learning algorithms. First-order languages, on the one hand, provide a natural way for describing learning problems over structurally complex instance spaces. In addition, hypotheses in this language are relatively easy to understand for users. On the other

⋆ Partially supported by the DFG project (WR 40/1-3) *Nachhaltige Informationsfusion: Aktives Lernen*
⋆⋆ Partially supported by the DFG project (WR 40/2-1) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume*

hand, however, serious decidability and complexity problems may arise from their use during the learning process. As an example, the membership problem, i.e., the problem of deciding whether an instance belongs to the concept represented by an hypothesis, becomes undecidable in first-order logic. To control such problems, different techniques (e.g., hypothesis language and search biases) have been proposed in ILP.[1]

Labeled graphs are one of the most important tools describing objects and the way they are connected. They are relational structures defined usually over vocabularies consisting of a single binary and a finite set of unary predicate symbols. They provide, in particular, a natural way for representing chemical compounds. Although ILP is concerned with learning from relational structures, and many ILP applications have been devoted to computational chemistry, surprisingly there are only few results (see, e.g., [12]) in the direction of restricting instances to labeled graphs. Such a structural assumption could then be exploited in the learning process to control decidability and complexity problems mentioned above.

In this work, we propose a boosted algorithm designed to learn acyclic conjunctive queries predicting unknown properties of chemical compounds. Our algorithm assumes that compounds are represented by relational structures corresponding to labeled graphs. We consider learning problems of the following form: *Given* disjoint sets E^+ and E^- of labeled graphs representing chemical compounds, *find* a set of definite first-order Horn clauses consistent (within some error) with E^+ and E^-. Since examples are disjoint labeled graphs, we use the learning from interpretations ILP setting [7] as the most plausible model for our purpose. In the algorithm presented, we apply top-down induction, a popular technique based on refinement operators [23] for first-order clauses. In our approach, refinement operators are defined by building blocks. In this work, we assume that such building blocks are provided by an expert. We are working on automatic extraction of building blocks for labeled graphs. We will discuss this problem later on.

In computational chemistry, pattern matching is usually defined by subgraph isomorphism. Since subgraph isomorphism generalizes the Hamiltonian path problem, it is NP-complete. In planar graphs, however, it can be solved in linear time for any pattern of constant size [9]. The importance of this result is that many molecules can be represented by planar graphs. In contrast to this approach, we define pattern matching by first-order logical implication, which in turn is equivalent to homomorphism [20] between relational structures in the problem setting considered. Since isomorphisms are special homomorphisms, we thus apply a more general operator in pattern matching. This may be important e.g. in those applications, where the length of paths connecting substructures is not relevant. Homomorphism between finite relational structures generalizes the graph vertex k-coloring problem, and is thus NP-hard. It becomes, however, polynomial for patterns of small *tree-width* [26]. Intuitively, tree-width measures the degree of cyclicity of structures. In this paper, we restrict the search space to patterns of tree-width one, also referred to as acyclic patterns. We note that homomorphism for this fragment is LOGCFL-complete [14] and is therefore highly parallelizable.

[1] We note that limitations regarding expressive power are not resolved completely by first-order logic, as first-order sentences are only able to capture *local properties* of structures (see, e.g., [8]).

In [17], we have presented a greedy algorithm for learning acyclic patterns. By this restriction, however, we reduce the expressive power of the hypothesis language. To compensate for the reduced expressiveness and thus the decrease in classification accuracy potentially resulting from it, in the proposed algorithm we combine acyclic patterns with confidence-rated boosting [27]. Ensemble methods, in particular boosting, are successful tools for increasing the prediction accuracy of classification learners by combining a set of only moderately accurate base hypotheses into one highly accurate strong hypothesis. Boosting works by repeatedly calling a base learner on reweighted versions of the training data, and thereby constructing an ensemble of specialized rules, or base hypotheses, which are finally combined into one prediction by weighted majority vote. In the framework of confidence-rated boosting, each base hypothesis not only predicts a classification but also generates a confidence score for this prediction.

The rest of the paper is organized as follows. In Section 2, we first review the necessary notions and results related to acyclic conjunctive queries. Section 3 is devoted to constrained confidence-rated boosting. In Section 4, we present our algorithm, and in Section 5, we empirically evaluate it on the domain of mutagenicity [19]. Finally, in Section 6 we conclude and discuss directions for future works.

2 Acyclic Conjunctive Queries

In this work, we restrict the hypothesis space to acyclic conjunctive queries, a practically relevant, efficiently evaluable fragment of first-order definite Horn-clauses. As an advantage over other ILP approaches using standard PROLOG evaluation techniques, we note that acyclic conjunctive queries allow evaluation of a set of instances in one single step. In this section we repeat the necessary notions related to acyclic conjunctive queries from our previous work [17]. In the Appendix, we give further details on acyclic conjunctive queries. For a detailed introduction to acyclic conjunctive queries the reader is referred to e.g. [1].

Throughout this section, we consider vocabularies consisting of a set of constant symbols, a distinguished predicate symbol called the target predicate, and a set of predicates called the background predicates. Thus, (non-constant) function symbols are not included in the vocabulary. Examples are ground atoms of the target predicate, and the background knowledge is an extensional database consisting of ground atoms of the background predicates. Furthermore, we assume that hypotheses are definite non-recursive first-order clauses, or in the terminology of relational database theory, conjunctive queries of the form $L_0 \leftarrow L_1, \ldots, L_l$, where L_0 is a target atom, and L_i is a background atom for $i = 1, \ldots, l$.

In order to define a special class of conjunctive queries, called *acyclic* conjunctive queries, we first need the notion of acyclic hypergraphs. A *hypergraph* (or *set-system*) $H = (V, E)$ consists of a finite set V called *vertices*, and a family E of subsets of V called *hyperedges*. A hypergraph is α-acyclic [10], or simply *acyclic*, if one can remove all of its vertices and edges by deleting repeatedly either a hyperedge that is empty or is contained by another hyperedge, or a vertex contained by at most one hyperedge [15, 31]. Note that acyclicity as defined here is not a *hereditary* property, in contrast to e.g. the standard notion of acyclicity in ordinary undirected graphs, as it may happen that

an acyclic hypergraph has a cyclic subhypergraph. For example, consider the hypergraph $H = (\{a, b, c\}, \{e_1, e_2, e_3, e_4\})$ with $e_1 = \{a, b\}$, $e_2 = \{b, c\}$, $e_3 = \{a, c\}$, and $e_4 = \{a, b, c\}$. This is an acyclic hypergraph, as one can remove step by step first the hyperedges e_1, e_2, e_3 (as they are subsets of e_4), then the three vertices, and finally, the empty hypergraph is obtained by removing the empty hyperedge that remained from e_4. On the other hand, the hypergraph $H' = (\{a, b, c\}, \{e_1, e_2, e_3\})$, which is a subhypergraph of H, is cyclic, as there is no vertex or edge that could be deleted by the above definition. In [10], other degrees of acyclicity are also considered, and it is shown that among them, α-acyclic hypergraphs form the largest class properly containing the other classes.

Using the above notion of acyclicity, now we are ready to define the class of acyclic conjunctive queries. Let Q be a conjunctive query and L be a literal of Q. We denote by Var(Q) (resp. Var(L)) the set of variables occurring in Q (resp. L). We say that Q is acyclic if the hypergraph $H(Q) = (V, E)$ with $V = \text{Var}(Q)$ and $E = \{\text{Var}(L) : L \text{ is a literal in } Q\}$ is acyclic. For instance, from the conjunctive queries

$$P(X, Y, X) \leftarrow R(X, Y), R(Y, Z), R(Z, X)$$
$$P(X, Y, Z) \leftarrow R(X, Y), R(Y, Z), R(Z, X)$$

the first one is cyclic, while the second one is acyclic.

3 Constrained Confidence-Rated Boosting

Boosting has established itself as a successful method for improving the classification accuracy of a learning system by combining the predictions of several base classifiers learned in iterative calls to the underlying learner. Numerous algorithms have emerged which demonstrate superior performance on a broad range of application problems (see, e.g., [11, 25, 5, 24, 16]).

The idea common to all boosting algorithms is to "boost" a weak learner performing only slightly better than random guessing into an arbitrarily accurate learner by repeatedly calling it on reweighted versions of the training data, and thereby constructing an ensemble of specialized rules, or base hypotheses. Predictions are based on all members of the learned ensemble by combining the individual predictions by weighted majority vote into one strong hypothesis.

The reweighted versions of the training set $E = E^+ \cup E^-$ on which the base learner is repeatedly called are obtained by maintaining a probability distribution D^t over E modeling the weight D_i^t associated with each training example e_i in the t-th iteration of boosting. D_i^t indicates the influence of an instance e_i when learning a base classifier C_t. Initially, the influence of all the instances is identical, i.e., the probability distribution D^1 is uniform. In each iterative call t of the base learner, a base hypothesis C_t with an associated weight $\tilde{c}_t$ is learned based on E weighted according to the current distribution D^t.

In the framework of confidence-rated boosting, the prediction of a base hypothesis C_t is confidence-rated. The sign of $\tilde{c}_t$ indicates the label predicted by C_t to be assigned to an instance, whereas the absolute value of $\tilde{c}_t$ is interpreted as C_t's prediction confidence, or the reliability of C_t's prediction. A base hypothesis' prediction confidence is,

on the one hand, used as its vote in the final, strong, hypothesis H, and, on the other hand, to update the distribution D^t for the next iteration of the base learner. The distribution is modified such that the weights of misclassified instances are increased while the weights of correctly classified instances are decreased. This way, the learner has to focus on those examples which are not correctly classified by the current ensemble.

Depending on the exact framework, a base hypothesis can apply distinct prediction confidences to different examples. Here, we employ a form of confidence-rated boosting in which a base hypothesis is restricted to make a prediction only for those examples which are covered by it, and to abstain otherwise. We furthermore restrict, following Cohen and Singer's approach to constrained confidence-rated boosting [5], the base hypotheses to either of two forms. A hypothesis either predicts, in the binary case we deal with here, the positive class with a positive prediction confidence, or it is the default hypothesis, just comprising the target predicate to be learned and satisfying all examples, with an assigned negative confidence.

We note that hypotheses obtained by boosting algorithms are potentially more complex than those generated by standard ILP learning systems. However, constraining the base hypotheses to either of the above two forms improves comprehensibility of strong hypotheses.

As suggested by [5], we aim at minimizing the ensemble's training error by searching in each round of boosting for a base hypothesis maximizing the objective function $\tilde{Z}$ which is, for a base hypothesis C_t, defined based on the collective weight of all positive and negative instances covered by C_t (in what follows, $x \in C$ denotes that instance x is covered by hypothesis C):

$$\tilde{Z}(C_t) = \sqrt{\sum_{x_i \in E^+, x_i \in C_t} D_i^t} - \sqrt{\sum_{x_i \in E^-, x_i \in C_t} D_i^t} \ . \tag{1}$$

After the last iteration of the base learner, the strong hypothesis H is formed on the basis of all hypotheses C_t learned over the course of iterations, and their assigned prediction confidences $\tilde{c}_t$ defined by

$$\tilde{c}_t = \frac{1}{2} \ln \left(\frac{\sum_{x_i \in E^+, x_i \in C_t} D_i^t + \frac{1}{2m}}{\sum_{x_i \in E^-, x_i \in C_t} D_i^t + \frac{1}{2m}} \right) \ , \tag{2}$$

where $m = |E|$ (see also [5]). To classify an instance x, the prediction confidences of all base hypotheses covering x are summed up. If this sum is positive, the strong hypothesis classifies x as positive, otherwise x is classified as negative:

$$H(x) = sign \left(\sum_{h_t} h_t(x) \right) \ , \tag{3}$$

where $h_t : X \rightarrow \Re$ is defined by

$$h_t(x) = \begin{cases} \tilde{c}_t & \text{if } x \in C_t \\ 0 & \text{otherwise} \ . \end{cases} \tag{4}$$

Algorithm 1 BACQ

Require: set E of $+/-$ labeled ground P-atoms and a labeled graph represented by ground A and B atoms

Ensure: a set of confidence-rated acyclic conjunctive queries

let C_{default} be the unit clause $P(X_1, \ldots, X_{m_P}) \leftarrow$
let $D(x_i) = 1/m$ for $i = 1, \ldots, m$ // where $m = |E|$
for $t = 1, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    $C_k = C_{\text{default}}$
    while $\exists C \in \mathcal{R}(C_k, N)$ such that $\widetilde{Z}(C) > \widetilde{Z}(C_k)$, // see (1) for the definition of $\widetilde{Z}$
        where $\mathcal{R}(C_k, N)$ is a set containing (at most)
        N randomly selected acyclic refinements of C_k **do**
      $C_k = C$
    end while
  end for
  let $C_t = C_j$ satisfying $\widetilde{Z}(C_j) = \max_{k=1,\ldots,K} \widetilde{Z}(C_k)$
  let $R_t = \begin{cases} C_t & \text{if } \widetilde{Z}(C_t) > |\widetilde{Z}(C_{\text{default}})| \\ C_{\text{default}} & \text{otherwise} \end{cases}$
  let $Cov \subseteq E$ be the set of examples covered by R_t
  // Cov is computed in a single step (see Algorithm EVALUATE in the Appendix)
  for $i = 1, \ldots, m$ **do**
    if $x_i \in Cov$ **then**
      let $D(x_i) = D(x_i) \cdot e^{-y_i \cdot \widetilde{c}_{R_t}}$
      // where $y_i \in \{+1, -1\}$ according to the label of x_i, and $\widetilde{c}_{R_t}$ is defined in (2)
    end if
  end for
  let $D(x_i) = D(x_i)/Z_t$ for $i = 1, \ldots, m$, where $Z_t = \sum_{i=1,\ldots,m} D(x_i)$
end for
return $\{(R_1, \widetilde{c}_{R_1}), \ldots, (R_T, \widetilde{c}_{R_T})\}$

4 Boosting Acyclic Conjunctive Queries

In this section, we present an algorithm designed to learn acyclic conjunctive queries predicting unknown properties of chemical compounds. Our algorithm assumes that compounds are represented by relational structures corresponding to labeled graphs. More precisely, we assume without loss of generality that the vocabulary consists of a target predicate P of arity m_P, and predicates A and B of arities m_A and m_B, respectively. For each compound, we have a ground target atom of the form $P(a_1, \ldots, a_{m_P})$, where a_1 is the identifier of the compound, and $a_2, \ldots, a_{m_P}$ are attribute values for the whole compound (e.g., ϵ-lumo). Each (chemical) atom is described by a fact of the form $A(b_1, \ldots, b_{m_A})$, where b_1 is the identifier of the compound containing the atom, b_2 is the atom's identifier, and $b_3, \ldots, b_{m_A}$ are attribute values for the atom. Finally, each bond is represented by a ground atom of the form $B(c_1, \ldots, c_{m_B})$, where c_1 is the compound identifier, c_2, c_3 are the identifiers of the atoms connected by the bond, and $c_4, \ldots, c_{m_B}$ are attribute values for the bond. Thus, the ground A and B atoms

represent the (labeled) vertices and (labeled) edges of the labeled graphs. We note that our approach assumes that molecules are represented in the learning from interpretation setting [7].

The algorithm combining top-down induction of acyclic conjunctive queries with constrained confidence-rated boosting is given in Algorithm 1. In Steps 1 and 2 of the algorithm, we first initialize the target clause and the distribution over the set of training examples. Then, we learn T weak hypotheses (Steps 3–19 of the Algorithm), where T is a user defined parameter.

To find a weak hypothesis, i.e., an acyclic conjunctive query, we apply top-down induction using the following refinement operator. We first select at random a literal with one of the predicate symbols P, A, or B from the clause to be refined. Then, depending on its predicate symbol, we add a set of literals to the clause as follows. If the selected literal is a P-literal (i.e., it is the head of the clause), with the same probability,

- either an atom or an acyclic building block (e.g., a benzene ring) is added to the clause,
- or one of the attributes of the P-atom is selected at random, and the best specialization for this attribute with respect to $\widetilde{Z}$ defined in Eq. (1) is computed.

If an A-atom (i.e., a labeled vertex in the graph) has been selected, we add to the clause

- either literals representing a labeled edge ending in this vertex,
- or an acyclic building block containing the selected vertex,
- or constraints specializing one of the attributes of the vertex in a similar fashion as described above.

Finally, for B-atoms, i.e., for labeled edges,

- we add either a set of literals defining an acyclic building block,
- or compute the best value for one of its attributes.

We note that none of the above refinements violates the acyclicity property. In particular, building blocks are restricted to be acyclic, and thus, as they share at most one edge with the labeled acyclic graph corresponding to C_k in Step 6, adding such a building block always results in an acyclic clause.

At the same time of adding a set of literals (i.e., atom, bond, or acyclic building block) to the clause, for each attribute of the literals we compute the best value with respect to $\widetilde{Z}$ and specialize the attribute with the value for which $\widetilde{Z}$ is maximal. In contrast to the greedy search used in [5], in Steps 5–8 of the algorithm, we apply simple local search for finding an acyclic conjunctive query. That is, we start the local search with the default clause, and refine it as long as its randomly selected refinement improves the quality measured by $\widetilde{Z}$. We repeat the random local search algorithm K times (see Step 4 of the algorithm) and select the acyclic conjunctive query of the best quality (Step 10). In Steps 12–18, we then update the distribution over the training set.

4.1 Building Blocks

In the work presented here, we assume that buildings blocks are provided by an expert. However, we are working on the automatic extraction of acyclic building blocks. Since structures are restricted to labeled chemical graphs, we are going to consider only cycle and tree patterns. To extract cycle patterns, we are going to compute the set of cycles of length ℓ, for every $\ell = 2, \ldots, L$. L is a user defined parameter bounding the length of cycles. For a fixed ℓ, this can be done by evaluating first the acyclic conjunctive query

$$\begin{aligned} cycle(Y_1, \ldots, Y_\ell) \leftarrow\ & B(X, Y_1, Y_2, Z_{1,1}, \ldots, Z_{1,m_B-3}), \\ & B(X, Y_2, Y_3, Z_{2,1}, \ldots, Z_{2,m_B-3}), \\ & \vdots \\ & B(X, Y_\ell, Y_1, Z_{\ell,1}, \ldots, Z_{\ell,m_B-3}) \end{aligned}$$

and then removing from the answer set the tuples $(a_1, \ldots, a_\ell)$ satisfying $a_i = a_j$ for some $1 \leq i < j \leq \ell$. By this step we filter those closed walks that are not cycles. If the remaining set is nonempty then for all possible subsets $\{V_1, \ldots, V_k\}$ of the nominal attributes of A and B, and for all possible combinations $\{v_1, \ldots, v_k\}$ of the values of these attributes, we can check in a similar way, whether there is a cycle of length ℓ such that $V_i = v_i$ in each atom and in each bond in the cycle for every $i = 1, \ldots, k$. This method is exponential in the number of nominal attributes of A and B. However, it is effective if the number of nominal attributes is small. To extract tree patterns, we are going to investigate the labeled graph obtained by removing all edges occurring in a cycle.

5 Empirical Results on the Domain of Mutagenicity

We evaluated our approach on the ILP-benchmark problem of *Mutagenicity* [19]. The learning task is to predict the mutagenicity of nitroaromatic compounds. Mutagenic compounds are often known to be carcinogenic and to cause damage to the DNA. Not all compounds can be empirically tested for mutagenicity, and the prediction of mutagenicity is vital to understanding and predicting carcinogenesis.

Of the several relational descriptions that are available for the domain [29], we use the strongly structured description $\mathcal{B}_4$ which comprises a description of the atoms of the molecules and the bonds between these atoms; global properties of the molecule as e.g. their hydrophobicity; chemical structures present in the molecules as e.g. benzenic or methylic groups.

Here, we consider the subset of 188 so called regression-friendly compounds 125 of which are classified as having positive levels of mutagenicity. The predictive accuracy is estimated by 10-fold-cross-validation, where we use the same folds as [29] for their experiments with Progol. The accuracy and standard deviation obtained in our experiments with BACQ is displayed in Table 1 for various numbers of iterations ranging from 50 to 400, together with reference results on the same dataset using background knowledge $\mathcal{B}_4$, and the sources from which these results are reported. ACQ is an ILP

Table 1. Accuracy ± standard deviation for the Mutagenicity domain for BACQ with different numbers of iterations ranging from 50 to 400 (i.e., T in Algorithm 1), in comparison to other systems

ACQ [17]	C^2RIB [16]	FOIL [19]	Fors [18]	G-Net [2]	Progol [19]	STILL [28]
87.0	88.0	82.0	89.0	92.0	88.0	90.0
n/a	±3.4	±3.0	±6.0	±8.0	±2.0	±5.0

BACQ 50	BACQ 100	BACQ 150	BACQ 200	BACQ 250	BACQ 300	BACQ 350	BACQ 400
89.9 ±4.6	89.9 ±4.6	91.5 ±3.8	90.4 ±4.9	92.0 ±3.8	92.0 ±3.8	91.5 ±3.8	91.5 ±4.5

learner based on acyclic conjunctive queries which we previously introduced [17] and which serves as a starting point for the work presented in this paper.

The classification accuracy obtained after only T=50 iterations with BACQ is lower than, however in the range of the standard deviation of, the best result reported so far for the Mutagenicity domain, accomplished with the system G-Net [2]. The result is also on par with the second best result reported for the domain, achieved with the system STILL [28]. For increasing T=250, and T=300, respectively, the classification accuracy is identical to the best one reported for this domain, G-Net [2], however with only half the standard deviation. Irrespective of the number of boosting iterations, our results with BACQ lie well in the range of the standard deviations reported for the learning systems most successful on the Mutagenicity domain. The classification accuracy of ACQ is significantly outperformed by any result of BACQ.

In our experiments, carbon_5_aromatic_ring, carbon_6_ring, carbon_5_ring, hetero_aromatic_6_ring, hetero_aromatic_5_ring, ring6, ring5 were used as cycle building blocks, and nitro and methyl as tree building blocks (see also [19]). As an example, the acyclic building block defining carbon_5_aromatic_ring is

$$\begin{aligned}&\text{carbon\_5\_aromatic\_ring}(X, Y_1, \ldots, Y_5) \leftarrow \\ &\quad \text{atom}(X, Y_1, c, U_1, V_1), \ldots, \text{atom}(X, Y_5, c, U_5, V_5), \\ &\quad \text{bond}(X, Y_1, Y_2, 7), \text{bond}(X, Y_2, Y_3, 7), \ldots, \text{bond}(X, Y_5, Y_1, 7)\end{aligned}$$

6 Conclusion

In this paper, we have presented an algorithm specific to learning acyclic conjunctive queries predicting unknown properties of chemical compounds. Here, chemical compounds have been represented by relational structures corresponding to labeled graphs. In our work, building blocks have been used for top-down induction of acyclic conjunctive queries. In the experiments, we have assumed that such buildings blocks were

provided by an expert. Although this seems to be a reasonable assumption when considering chemical graphs, we are working on the automatic extraction of cycle and tree patterns as building blocks.

In ILP, examples are usually evaluated one by one (by some PROLOG system). One of the major advantages of our approach is that acyclic conjunctive queries allow, in contrast to the standard ILP evaluation approach, examples to be evaluated in one step.

Restricting the search space to acyclic patterns implies, however, a reduced expressiveness and a potential decrease in classification accuracy. These shortcomings are counteracted by applying constrained confidence-rated boosting. Our first experiments indicate that combining acyclic conjunctive queries with constrained confidence-rated boosting has indeed a potential for real-world problems involving labeled graphs. As future work, we are going to evaluate the method on further such domains.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, Mass., 1995.
2. C. Anglano, A. Giordana, G. Lo Bello, and L. Saitta. An experimental evaluation of coevolutive concept learning. *Proc. of the 15th Int. Conf. on Machine Learning*, 1998.
3. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983.
4. P. A. Bernstein and N. Goodman. The power of natural semijoins. *SIAM Journal on Computing*, 10(4):751–771, 1981.
5. W. Cohen and Y. Singer. A Simple, Fast, and Effective Rule Learner. *Proc. of 16th National Conference on Artificial Intelligence*, 1999.
6. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., 1990.
7. L. De Raedt and S. Dzeroski. First-Order jk-Clausal Theories are PAC-Learnable. *Artificial Intelligence*, 70(1-2): 375-392, 1994.
8. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory* Springer-Verlag,Berlin,1995.
9. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms & Applications*, 3(3):1–27,1999.
10. R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30(3):514–550, 1983.
11. Y. Freund, and R.E. Schapire. Experiments with a New Boosting Algorithm. *Proc. of 13th International Conference on Machine Learning*, 1996.
12. T. Gärtner, P. A. Flach, and S. Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. *The Sixteenth Annual Conference on Computational Learning Theory and The Seventh Kernel Workshop (COLT-2003)*. To Appear.
13. G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.
14. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 706–715. IEEE Computer Society Press, 1998.
15. M. Graham. On the universal relation. Technical report, Univ. of Toronto, Toronto, Canada, 1979.

16. S. Hoche and S. Wrobel. Relational Learning Using Constrained Confidence-Rated Boosting. *Proc. 11th Int. Conf. on Inductive Logic Programming (ILP)*, 2001.
17. T. Horvath and S. Wrobel. Toward Discovery of Deep and Wide First-Order Structures: A Case Study in the Domain of Mutagenicity. *Proc. Discovery Science*, 2001.
18. A. Karalic. *First Order Regression*. PhD thesis, University of Ljubljana, Faculty of Computer Science, Ljubljana, Slovenia, 1995.
19. A. Srinivasan, S. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85:277-299, 1996.
20. Kolaitis and Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS: Journal of Computer and System Sciences*, 61(2):302–332, 2000.
21. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3-4):245–286, 1995.
22. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19/20:629–680, 1994.
23. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer, Berlin, 1997.
24. D. Opitz, and R. Maclin. Popular Ensemble Method: An Empirical Study. *Journal of Artificial Intelligence Research 11, pages 169-198*, 1999.
25. J.R. Quinlan. Bagging, boosting, and C4.5. *Proc. of 14th Nat. Conf. on AI*, 1996.
26. N. Robertson, and P.D. Seymour. Graph minors II: algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322,1986.
27. R. E. Schapire, and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Proceedings of COLT'98*, pages 80–91, 1998.
28. M. Sebag. Distance Induction in First Order Logic. *Proc. 7th Int. Workshop on Inductive Logic Programming (ILP)*, 1997.
29. A. Srinivasan, S. Muggleton, and R. King. Comparing the use of background knowledge by inductive logic programming systems. *Proceedings of the 5th International Workshop on Inductive Logic Programming*, 1995.
30. S. Wrobel. Inductive logic programming. In G. Brewka, editor, *Advances in Knowledge Representation and Reasoning*, pages 153–189. CSLI-Publishers, Stanford, CA, USA, 1996. Studies in Logic, Language and Information.
31. C. T. Yu and Z. M. Ozsoyoglu. On determining tree query membership of a distributed query. *INFOR*, 22(3), 1984.

Appendix: Acyclic Conjunctive Queries

In this appendix we give an algorithm for acyclic conjunctive query evaluation. In [3] it is shown that the class of acyclic conjunctive queries is identical to the class of conjunctive queries that can be represented by *join forests* [4]. Given a conjunctive query Q, the join forest $JF(Q)$ representing Q is an ordinary undirected forest such that its vertices are the set of literals of Q, and for each variable $x \in \text{Var}(Q)$ it holds that the subgraph of $JF(Q)$ consisting of the vertices that contain x is connected (i.e., it is a tree).

Now we show how to use join forests for efficient acyclic query evaluation. Let E be a set of ground target atoms, B be a set of ground atoms, and let Q be an acyclic conjunctive query with join forest $JF(Q)$. In order to find the subset $E' \subseteq E$ implied by Q with respect to B, we can apply the following method. Let $T_0, T_1, \ldots, T_k$ ($k \geq 0$)

algorithm EVALUATE

```
input: extensional database D and join tree T with root
       labeled by n_0
output: {n_0θ: θ is a substitution mapping the nodes of T into D}

let R = {n_0θ: θ is a substitution mapping n_0 into D}
let the children of n_0 be labeled by n_1, ..., n_k (k ≥ 0)
for i = 1 to k
  S = evaluate(D, T_i)     // T_i is the subtree of T rooted at n_i
  R = the natural semijoin of R and S wrt. n_0 and n_i
return R
```

denote the set of connected components of $JF(Q)$, where T_0 denotes the tree containing the head of Q, and let $Q_i \subseteq Q$ denote the query represented by T_i for $i = 0, \ldots, k$. The definition of the Q_i's implies that they form a partition of the set of literals of Q such that literals belonging to different blocks do not share common variables. Therefore, the subqueries $Q_0, \ldots, Q_k$ can be evaluated separately; if there is an i, $1 \leq i \leq k$, such that the Boolean conjunctive query Q_i (i.e., a conjunctive query with empty head) is false with respect to B then Q implies *none* of the elements of E with respect to B, otherwise Q and Q_0 imply the same subset of E with respect to B. By definition, Q_0 implies an atom $e \in E$ if there is a substitution mapping the head of Q_0 to e and the atoms in its body into B, and Q_i ($1 \leq i \leq k$) is true with respect to B if there is a substitution mapping Q_i's atom into B. That is, using algorithm EVALUATE given below, Q implies E' with respect to B if and only if

$$(E' \subseteq \text{EVALUATE}(B \cup E, T_0)) \wedge \left(\bigwedge_{i=1}^{k} (\text{EVALUATE}(B, T_i) \neq \emptyset) \right) .$$

It remains to discuss the problem of how to compute a join forest for an acyclic conjunctive query. Using maximal weight spanning forests of ordinary graphs, in [4] Bernstein and Goodman give the following method to this problem. Let Q be an acyclic conjunctive query, and let $G(Q) = (V, E, w)$ be a weighted graph with vertex set $V = \{L : L \text{ is a literal of } Q\}$, edge set $E = \{(u, v) : \text{Var}(u) \cap \text{Var}(v) \neq \emptyset\}$, and with weight function $w : E \to \mathbb{N}$ defined by $w : (u, v) \mapsto |\text{Var}(u) \cap \text{Var}(v)|$. Let $MSF(Q)$ be a maximal weight spanning forest of $G(Q)$. Note that maximal weight spanning forests can be computed in polynomial time (see, e.g., [6]). It holds that if Q is acyclic then $MSF(Q)$ is a joint forest representing Q. In addition, given a maximal weight spanning forest $MSF(Q)$ of a conjunctive query Q, instead of using the method given in the definition of acyclic hypergraphs, in order to decide whether Q is acyclic, one can check whether the equation

$$\sum_{(u,v) \in MSF(Q)} w(u, v) = \sum_{x \in \text{Var}(Q)} (\text{Class}(x) - 1) \tag{5}$$

holds, where $\text{Class}(x)$ denotes the number of literals in Q that contain x (see also [4]).