Journal of Machine Learning Research ? (2007) ?–? Submitted 11/05; Revised 10/06; Published ??

Integrating Naïve Bayes and FOIL*

Niels Landwehr[1] LANDWEHR@INFORMATIK.UNI-FREIBURG.DE
Kristian Kersting[1] KERSTING@INFORMATIK.UNI-FREIBURG.DE
[1]*Department of Computer Science*
Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 79
79110 Freiburg, Germany
Luc De Raedt[2,1] LUC.DERAEDT@CS.KULEUVEN.BE
[2]*Department of Computer Science*
Katholieke Universiteit Leuven
Celestijnenlaan 200 A
B-3001 Heverlee, Belgium

Editor: Stefan Wrobel

Abstract

A novel relational learning approach that tightly integrates the naïve Bayes learning scheme with the inductive logic programming rule-learner FOIL is presented. In contrast to previous combinations that have employed naïve Bayes only for post-processing the rule sets, the presented approach employs the naïve Bayes criterion to guide its search directly. The proposed technique is implemented in the NFOIL and TFOIL systems, which employ standard naïve Bayes and tree augmented naïve Bayes models respectively. We show that these integrated approaches to probabilistic model and rule learning outperform post-processing approaches. They also yield significantly more accurate models than simple rule learning and are competitive with more sophisticated ILP systems.

Keywords: rule learning, naïve Bayes, statistical relational learning, inductive logic programming

1. Introduction

The study of learning schemes that lie at the intersection of probabilistic and logic or relational learning has received a lot of attention recently (De Raedt and Kersting, 2003; Getoor and Taskar, 2007). Whereas typical approaches upgrade existing probabilistic learning schemes to deal with relational or logical data (such as Probabilistic Relational Models (Getoor et al., 2001) or Stochastic Logic Programs (Muggleton, 1996)), we start from an inductive logic programming system and extend it with a probabilistic model. More specifically, we start with the simplest approaches from both domains, the inductive logic programming system FOIL (Quinlan, 1990) and naïve Bayes, and integrate them in the NFOIL system. As the strong independence assumption of naïve Bayes can be problematic in some domains, we furthermore investigate a generalization of naïve Bayes known as tree augmented naïve Bayes in the TFOIL system. Indeed, this methodology could be extended to learning full Bayesian networks; however, the advantage of combining such simple learning

*. This is a significant extension of a paper that appeared in the Proceedings of the Twentieth National Conference on Artificial Intelligence (Landwehr et al., 2005).

schemes is that the resulting probabilistic logical or relational model is easier to understand and to learn.

In relational learning or inductive logic programming, one typically induces a set of rules (so-called clauses). The resulting rule-set then defines a disjunctive hypothesis, as an instance is classified as positive if it satisfies the conditions of one of the rules. On the other hand, a probabilistic model defines a joint probability distribution over a class variable and a set of "attributes" or "features", and the type of model constrains the joint probability distributions that can be represented. A straightforward but powerful idea to integrate these two approaches is to interpret the clauses or rules as propositional "features" over which a joint probability distribution can be defined. Using naïve Bayes as the probabilistic model, this translates into the statement that "*clauses are independent*". This idea is not really new. It has been pursued by Pompe and Kononenko (1995) and Davis et al. (2004). However, in these existing approaches for combining ILP and naïve Bayes, one learns the model in two *separate* steps. First, the features or clauses are generated (for example using an existing inductive logic programming system such as ILP-R (Pompe and Kononenko, 1995)) and then the probability estimates for the naïve Bayes are determined. This actually corresponds to a *static propositionalization* approach, where the propositionalized problem is learned using naïve Bayes.

We propose a *different* and *novel* approach, in which feature construction is tightly integrated with naïve Bayes. The advantage of such a *dynamic propositionalization* is that the criterion according to which the features are generated is that of naïve Bayes. Our intention in this paper is to investigate how dynamic propositionalization compares to static propositionalization approaches that use naïve Bayes only to post-process the rule set. More precisely, we will investigate:

(Q1) Is there a gain in predictive accuracy of a dynamic propositionalization approach over its ILP baseline?

(Q2) If so, is the gain of dynamic propositionalization over its baseline larger than the gain of static propositionalization approaches?

(Q3) Can performance be improved by using a more expressive probabilistic model, such as tree augmented naïve Bayes?

(Q4) Is a dynamic propositionalization based on a simple rule learner competitive with advanced ILP approaches?

(Q5) Relational naïve Bayes methods such as 1BC2 (Flach and Lachiche, 2004) essentially employ all clauses within a given language bias as features in a probabilistic model and thus perform static propositionalization. Does dynamic propositionalization employ fewer features and perform better than these approaches?

To answer Questions **Q1**–**Q5**, we present the NFOIL and TFOIL systems. The former is the first system reported in the literature that tightly integrates feature construction and naïve Bayes (De Raedt and Kersting, 2004; Landwehr et al., 2005). It guides the structure search by the probabilistic score of naïve Bayes. This contrasts with static propositionalization approaches, in which the criteria employed for feature generation and classification are different.

NFOIL essentially performs a covering search in which one feature (in the form of a clause) is learned after the other, until adding further features does not yield improvements. The search heuristic is based on class conditional likelihood and clauses are combined with naïve Bayes. TFOIL is an

extension of NFOIL that employs a tree augmented naïve Bayes, that is, it relaxes the naïve Bayes assumption to allow additional probabilistic dependencies between clauses.

Recently, two other approaches that integrate the search for structural rules and a probabilistic model have been proposed. For STRUCTURAL LOGISTIC REGRESSION (Popescul et al., 2003), aggregated SQL queries are used as features in a logistic regression model. The authors propose an integrated search algorithm; however, the higher computational cost of training a logistic regression model allows for a limited search only. In the experimental evaluation, the implementation of Structural Logistic Regression only searches for clauses with up to two literals, where the first literal is fixed in advance. Furthermore, no experimental comparison to static propositionalization approaches is provided. In 2005, Davis et al. presented the SAYU system, which also learns a set of first-order rules that are combined in a probabilistic model. Clauses are selected according to the area under the precision-recall curve. The parameters of the probabilistic model are estimated to maximize the likelihood. Unlike naïve Bayes and NFOIL, SAYU does not handle multi-class problems (cf. the discussing of related work in Section 7).

The rest of the paper is organized as follows. Section 2 introduces some basic concepts from inductive logic programming and the simple first-order rule learning algorithm FOIL. In Section 3, we lift FOIL's problem setting to the probabilistic case and introduce a combined probabilistic/logical model for this setting. Section 4 shows how the NFOIL algorithm for learning such a model can be obtained by modifying certain components of the FOIL algorithm. Section 5 discusses how to extend the NFOIL algorithm to incorporate tree augmented naïve Bayes, resulting in the TFOIL system. In Section 6 we experimentally evaluate the proposed methods on benchmark datasets from four different domains and specifically investigate how it compares to static propositionalization approaches. Finally, we conclude and touch upon related work.

2. FOIL: First Order Inductive Learning

The problem that we tackle in this paper is a probabilistic formulation of the traditional inductive logic programming (ILP) problem. In ILP, the goal is to induce first-order clausal hypotheses from a relational description of a set of labeled examples and background knowledge. As an illustrating example, consider the task of inducing a theory that distinguishes between mutagenic and non-mutagenic chemical compounds based on the description of their structure (cf. the well-known mutagenicity problem (Srinivasan et al., 1996)):

Example 1 *Consider the following background theory B:*

$$
\begin{array}{ll}
atom(189, d189\_1, c, 22, -0.11) & bond(189, d189\_1, d189\_2, 7) \\
atom(189, d189\_2, c, 22, -0.11) & bond(189, d189\_2, d189\_3, 7) \\
atom(189, d189\_3, c, 27, 0.02) & bond(189, d189\_3, d189\_4, 7) \\
\ldots & \ldots \\
atom(189, d189\_26, o, 40, -0.38) & bond(189, d189\_18, d189\_26, 2)
\end{array}
$$

and the example $mutagenic(189)$. A possible hypothesis for this domain is

$$
\begin{array}{l}
mutagenic(X) \leftarrow atom(X, A, o, 40, C), bond(X, B, A, 2) \\
mutagenic(X) \leftarrow atom(X, A, c, 22, C), atom(X, B, E, 22, 0.02), bond(X, A, B, 7)
\end{array}
$$

In the example, $atom/5$ and $bond/4$ are *predicates* (of arity 5 and 4 respectively) that identify relations, numbers and lower-case strings like $189, c, d189\_1$ are *constants* that identify objects and upper-case letters like X, A, B, C are logical *variables*. *Logical atoms* are predicates together with their arguments, for example $bond(189, d189\_1, d189\_2, 7)$. *Definite clauses*, such as

$$mutagenic(X) \leftarrow atom(X, A, o, 40, C), bond(X, A, B, 2),$$

consist of a head $mutagenic(X)$ and a body $atom(X, A, o, 40, C), bond(X, A, B, 2)$. The logical atoms in the body are also called *literals*. *Facts* are definite clauses with an empty body, e.g., $atom(189, d189\_1, c, 22, -0.11)$. A definite clause is called *ground* if it does not contain any variables. A *hypothesis* is a set of clauses. A hypothesis is said to *entail* an example given the background knowledge if the example is a logical consequence of the definite clauses in the hypothesis and background knowledge. In the example, $mutagenic(189)$ is entailed by the hypothesis given the background knowledge.

(Probabilistic) inductive logic programming problems can be formalized in a general way as follows (De Raedt and Kersting, 2004):

Given

- a background theory B, in the form of a set of definite clauses $h \leftarrow b_1, \cdots, b_n$;
- a set of examples E, in the form of ground facts, classified into classes C;
- a language of clauses $\mathcal{L}$, which specifies the clauses that are allowed in hypotheses;
- a $covers(e, H, B)$ function, which returns the classification $covers(e, H, B)$ of an example e, w.r.t. a hypothesis H, and the background theory B;
- a $score(E, H, B)$ function, which specifies the quality of the hypothesis H w.r.t. the data E and the background theory;

Find

$$\arg\max_{H \subset \mathcal{L}} score(E, H, B) \ .$$

Traditional approaches to inductive logic programming (Muggleton and De Raedt, 1994) tackle a concept-learning problem, in which there are typically two classes, and the goal is to find a complete and consistent concept-description. This can be formalized within our framework by making the following choices for $covers$ and $score$:

- $covers(e, H, B) =$ positive if $B \cup H \models e$ (i.e., e is entailed by $B \cup H$);
  otherwise, $covers(e, H, B) =$ negative;
- $score(E, H, B) =$ training set accuracy (or a related measure).

Using this definition of coverage, the hypothesis given in Example 1 covers the example $mutagenic(189)$ together with the background knowledge. This setting is incorporated in many well-known inductive logic programming systems such as FOIL (Quinlan, 1990), GOLEM (Muggleton and Feng, 1990), PROGOL (Muggleton, 1995) and TILDE (Blockeel and De Raedt, 1997).

Algorithm 1 Generic FOIL algorithm.

```
Initialize H := ∅
repeat
   Initialize c := p(X_1, ··· , X_n) ←
   repeat
      for all c' ∈ ρ(c) do
         compute score(E, H ∪ {c'}, B)
      end for
      let c be the c' ∈ ρ(c) with the best score
   until stopping criterion
   add c to H
   E := update(E, H)
until stopping criterion
output H
```

FOIL, like many inductive logic programming systems, follows a *greedy* and *incremental* approach to induce a hypothesis. Such an algorithm is described in Algorithm 1. It repeatedly searches for clauses that score well with respect to the data set and the current hypothesis and adds them to the hypothesis. In the *update* function, the set of training examples can be updated after a clause has been learned. In the inner loop, the algorithm greedily searches for a clause that scores well. To this aim, it employs a general-to-specific hill-climbing search strategy. To generate the specializations of the current clause c, a so-called refinement operator ρ under θ-subsumption is employed. A clause c_1 θ-subsumes a clause c_2 if and only if there is a substitution θ such that $c_1\theta \subseteq c_2$. A substitution is a set $\{V_1/t_1, \ldots, V_l/t_l\}$ where the V_i are different variables and the t_i are terms, and the application of the substitution replaces the variables $V_1, \ldots, V_l$ by the corresponding terms $t_1, \ldots, t_l$. The most general clause is $p(X_1, \cdots, X_n) \leftarrow$ where p/n is the predicate being learned and the X_i are different variables. The refinement operator specializes the current clause $h \leftarrow b_1, \cdots, b_n$. This is typically realized by either adding a new literal l to the clause yielding $h \leftarrow b_1, \cdots, b_n, l$ or by applying a substitution θ yielding $h\theta \leftarrow b_1\theta, \cdots, b_n\theta$.

This type of algorithm has been successfully applied to a wide variety of problems in inductive logic programming. In classical FOIL, it is further simplified to a *separate-and-conquer* approach: examples that are covered by a learned clause are removed from the training data, and the score of a clause can be computed without respect to the current hypothesis; that is, $score(E, H \cup \{c'\}, B)$ simplifies to $score(E, c', B)$. Many different scoring functions and stopping criteria have been employed. The original FOIL algorithm uses information gain based on the number of positive and negative tuples bound to clauses as the scoring function (Quinlan, 1990), and a stopping criterion based on the minimum description length principle (Rissanen, 1978). MFOIL, a variant of FOIL that was particularly designed to cope with noise in the training data, uses the m-estimate for scoring clauses and a significance-based stopping criterion (Lavrač and Džeroski, 1994).

3. Integrating Naïve Bayes and FOIL: Problem Specification

Let us now discuss how to integrate the naïve Bayes method in FOIL's problem specification. This will be realized by modifying the $covers$ and $score$ functions in the inductive logic programming

setting. All other elements, such as the examples, background theory and language of clauses $\mathcal{L}$ will – in principle – be untouched. However, the set of clauses defining a hypothesis is augmented with a set of parameters that quantify the probabilistic model. For easy of exposition, we will first discuss this for the case that the clauses are combined using naïve Bayes, that is, the NFOIL system. The extension to the tree-augmented naïve Bayes model (TFOIL) will be presented in Section 5.

3.1 A Probabilistic $covers$ Function

In the setting of learning from probabilistic entailment (De Raedt and Kersting, 2004), the notion of coverage is replaced by a probability. We will use $\mathbf{P}$ to denote a probability distribution as in $\mathbf{P}(\mathbf{x})$, and P to denote a probability value as in $P(x)$, where x is a state of the random variable $\mathbf{x}$.

The probabilistic covers relation is then defined as the likelihood of the example, conditioned on the hypothesis and the background theory:

$$covers(e, H, B) = P(e \mid H, B), \tag{1}$$

where B is defined as before and $H = (H_C, H_\lambda)$ is an augmented hypothesis, consisting of a clause set H_C and an associated probabilistic model H_λ. An example e is of the form $p(X_1, \cdots, X_n)\theta = true$ or $p(X_1, \cdots, X_n)\theta = \ false$. Abusing notation—when the context is clear—we will sometimes refer to the example as θ, and say that the random variable $\mathbf{p}$ (class label) takes on the value $p\theta$ (true or false) for example θ.

We now still need to define $P(e \mid H, B)$. The key idea for realizing this is that we interpret the clauses in H together with the example e as queries or features. More formally, let H contain a set of clauses defining the predicate p. Then for each clause c of the form

$$p(X_1, \cdots, X_n) \leftarrow b_1, \cdots, b_n$$

we view the query $q_c = \leftarrow b_1, \cdots, b_n$ as a boolean feature or attribute. Applied to an example θ these queries become instantiated, $q_c\theta = \leftarrow b_1\theta, \cdots, b_n\theta$, and either succeed or fail in the background theory B. As for the class label, we will say that $q_c\theta$ is the observed (boolean) value of the random variable $\mathbf{q_c}$.

Example 2 *Assume that the background theory is given as in Example 1, and the query q_c under consideration is*

$$\leftarrow atom(X, A, o, 40, C), bond(X, B, A, 2)$$

For the example $\theta = \{X/189\}$, the instantiated clause

$$\leftarrow atom(189, A, o, 40, C), bond(189, B, A, 2)$$

succeeds in the background theory, so the boolean random variable $\mathbf{q_c}$ takes on value $q_c\theta = true$ for this example.

The probabilistic model H_λ of H specifies a distribution over the random variables $\mathbf{p}$ and $\mathbf{q_c}$. The observed (boolean) value of $\mathbf{q_c}$ is $q_c\theta$. We now define

$$\begin{aligned} P(e \mid H, B) &= P_\lambda(p\theta | q_1\theta, \cdots, q_k\theta) \\ &= \frac{P_\lambda(q_1\theta, \cdots, q_k\theta | p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta, \cdots, q_k\theta)} \end{aligned}$$

where $H_C = \{q_1, ..., q_k\}$ and $\mathbf{P}_\lambda(\mathbf{p}, \mathbf{q_1}, ..., \mathbf{q_k})$ is the distribution defined by H_λ.

Now it becomes possible to state the naïve Bayes assumption

$$\mathbf{P}_\lambda(\mathbf{q_1}, ..., \mathbf{q_k}|\mathbf{p}) = \prod_i \mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p})$$

and apply it to our *covers* function (Equation 1):

$$P(e \mid H, B) = \frac{\prod_i P_\lambda(q_i\theta|p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta, \cdots, q_k\theta)}$$

At the same time, this equation specifies the parameters H_λ of the augmented model H, which are the distributions $\mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p})$ and the prior class distribution $\mathbf{P}_\lambda(\mathbf{p})$ (note that $\mathbf{P}_\lambda(\mathbf{q_1}, ..., \mathbf{q_k})$ can be computed by summing out).

Example 3 *Reconsider the mutagenicity example, and assume that the hypothesis is as sketched before. Then the queries q_1 and q_2 are*

$$mutagenic(X) \leftarrow atom(X, A, o, 40, C), bond(X, B, A, 2)$$
$$mutagenic(X) \leftarrow atom(X, A, c, 22, C), atom(X, B, E, 22, 0.02), bond(X, A, B, 7)$$

and the target predicate p is "mutagenic(X)". Now assume a naïve Bayes model with probability distributions $\mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p})$ given as

$$P_\lambda(\mathbf{p} = t) = 0.6$$
$$P_\lambda(\mathbf{q_1} = t|\mathbf{p} = t) = 0.7 \qquad P_\lambda(\mathbf{q_1} = t|\mathbf{p} = f) = 0.4$$
$$P_\lambda(\mathbf{q_2} = t|\mathbf{p} = t) = 0.2 \qquad P_\lambda(\mathbf{q_2} = t|\mathbf{p} = f) = 0.1$$

Summing out yields

$$P_\lambda(\mathbf{q_1} = t, \mathbf{q_2} = t) = 0.10 \qquad P_\lambda(\mathbf{q_1} = t, \mathbf{q_2} = f) = 0.48$$
$$P_\lambda(\mathbf{q_1} = f, \mathbf{q_2} = t) = 0.06 \qquad P_\lambda(\mathbf{q_1} = f, \mathbf{q_2} = f) = 0.36$$

where t (f) denotes $true$ ($false$). For the positively labeled example $\theta = \{X/189\}$, q_1 succeeds and q_2 fails: $p\theta = true$, $q_1\theta = true$, $q_2\theta = false$. Thus,

$$P(e \mid H, B) = \frac{P_\lambda(q_1\theta|p\theta) \cdot P_\lambda(q_2\theta|p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta, q_2\theta)} = \frac{0.7 \cdot 0.8 \cdot 0.6}{0.48} = 0.7$$

3.2 The $score$ Function

As scoring function, we employ the likelihood $P(E \mid H, B)$ of the data given the model and the background knowledge, and we assume also that the instances are independently and identically distributed (i.i.d.). So, we want to find the hypothesis H that maximizes

$$P(E \mid H, B) = \prod_{e \in E} P(e|H, B) \ .$$

To evaluate a set of clauses H_C in the generic FOIL algorithm, the scoring function has to be changed accordingly, to

$$score(E, H_C, B) = P(E \mid H, B)$$

where H_C has been augmented with an optimal probabilistic model H_λ. This is discussed in more detail in Section 4.

3.3 Multi-Class Problems

Because we started from ILP, we have so far formulated the learning problem as a binary classification task. In ILP, multiple classes are usually encoded with a predicate $p(X, C)$ where C is an *output variable* that is instantiated with a class $c_i \in \{c_1, ..., c_m\}$. One hypothesis is then induced per class using target predicate $p(X, c_i)$. However, combining these hypothesis to classify new instances is generally non-trivial, as *conflict resolution* strategies are needed to resolve conflicting predictions from the individual hypotheses. The same holds for approaches that use ILP to generate a rule set (Pompe and Kononenko, 1995; Davis et al., 2004) or an ILP technique such as bottom clauses to generate candidate rules for selection (Davis et al., 2005).

On the other hand, probabilistic models such as (tree augmented) naïve Bayes can deal with multi-class problems very naturally: the binary class variable is simply replaced by a multi-valued one. This directly carries over to the integrated model presented above. For a multi-class problem with classes $\{c_1, ..., c_m\}$, the examples are of the form $p\theta = p(x, c_i)$ and the random variable $\mathbf{p}$ in the naïve Bayes model takes on some value $p\theta \in \{c_1, ..., c_m\}$. No conflict resolution strategies are needed, as the naïve Bayes model directly returns a class $c_i \in \{c_1, \ldots, c_m\}$. Handling multi-class problems is part of the NFOIL and TFOIL implementations and we report on experimental results in Section 6.

4. Integrating Naïve Bayes and FOIL: Learning

The goal of learning is to identify a hypothesis H that maximizes the *score* function. More formally, the model space under consideration is

$$\mathcal{H} = \{(H_C, H_\lambda) | H_C \subseteq \mathcal{L}, H_\lambda \in \mathcal{M}_C\}$$

where $\mathcal{M}_C$ is the space of all naïve Bayes models over the clauses in H_C. The optimization problem is to find

$$\begin{aligned} H^* &= \arg\max_H P(E \mid H, B) \\ &= \arg\max_{H_C, H_\lambda} P(E \mid (H_C, H_\lambda), B), \end{aligned}$$

a hypothesis that jointly optimizes the clause set (structure) and probabilistic model (parameters such as the $\mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p})$ appearing in Example 3). Roughly speaking, the existing approaches pursued by Pompe and Kononenko (1995) and Davis et al. (2004) solve this task in a two-step process: First, H_C^* is found (using a standard ILP system, and thus some deterministic score) and fixed; second, the parameters for the fixed structure are optimized using a probabilistic score (usually, maximum likelihood):

for all $H_C \in candidates(\mathcal{L})$ **do**
  compute *ilp-score*(E, H_C, B)
end for
$H_C^* = \arg\max_{H_C}$ *ilp-score*(E, H_C, B)
find $H_\lambda^* = \arg\max_{H_\lambda} score(E, (H_C^*, H_\lambda), B)$

As a consequence, it is unclear which score is being maximized at the *global* level.

A more principled way of solving such optimization problems relies on a nested approach: Evaluate structures H_C which are each augmented with optimal parameters, and select the best structure H_C^*. Thus, the task of structure selection involves augmenting a given structure with optimal parameters:

for all $H_C \in candidates(\mathcal{L})$ **do**
  find $H_\lambda^* = \arg\max_{H_\lambda} score(E, (H_C, H_\lambda), B)$
  $score(H_C) := score(E, (H_C, H_\lambda^*), B)$
end for
$H_C^* = \arg\max_{H_C} score(H_C)$

We will follow this more principled approach of guiding the search for the structure directly by the probabilistic objective function. The rest of this section shows how this can be realized by modifying the original search technique used in FOIL. This will first be presented for learning the basic NFOIL model. An extension of the learning algorithm that accounts for the tree augmented naïve Bayes structure in TFOIL will be discussed in Section 5.

4.1 Adapting FOIL

Like FOIL, NFOIL searches a set of clauses greedily, and a single clause in a general-to-specific manner using a refinement operator. The main difference in the search technique is that FOIL can use a *separate-and-conquer* approach. Because the final model in FOIL is the disjunction of the learned clauses (where every clause covers a certain subset of examples), it holds that

1. Examples that are already covered do not have to be considered when learning additional clauses: $update(E, H) = E \setminus covered(H)$

2. (Non-recursive) clauses already learned do not need to be considered when scoring additional clauses: $score(E, H \cup \{c'\}, B) = score(E, \{c'\}, B)$.

Here, $score(E, \{c'\}, B)$ is some accuracy-related measure such as information gain or the m-estimate. In NFOIL, such a separate-and-conquer approach is not possible because every clause can affect the likelihood of all examples. Consequently, the NFOIL algorithm can be obtained from FOIL by changing two components in the generic FOIL algorithm (see Algorithm 1):

1. The set of examples is not changed after learning a clause: $update(E, H) = E$.

2. An additional clause c' has to be scored together with the current model:

$$score(E, H_C \cup \{c'\}, B) = P(E \mid H', B)$$

   where H' has clauses $H_C \cup \{c'\}$ and optimal parameters H'_λ.

We also have to modify the stopping criterion. The most basic stopping criterion for FOIL stops when all positive examples are covered. This is replaced in NFOIL by stopping if the change in score when adding a clause falls below a certain threshold. In general, this simple criterion might lead to overfitting. We therefore also investigated post-pruning the learned hypothesis (see Section 6). In the experiments, however, this basic stopping criterion worked surprisingly well.

4.2 Parameter Estimation: An Approximation

To evaluate a set of clauses $H_C = \{q_1, ..., q_k\}$ by $score(E, H_C, B)$, one needs to solve the "inner" optimization problem of finding optimal parameters H_λ for the naïve Bayes model over the random variables $\{\mathbf{q_1}, ..., \mathbf{q_k}\}$:

$$\begin{aligned} H^*_\lambda &= \arg\max_{H_\lambda} P(E \mid H, B) \\ &= \arg\max_{\lambda} \prod_{\theta \in E} P_\lambda(p\theta | q_1\theta, \cdots, q_k\theta) \\ &= \arg\max_{\lambda} \prod_{\theta \in E} \frac{\prod_j P_\lambda(q_j\theta \mid p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta, ..., q_k\theta)} \end{aligned} \tag{2}$$

However, these are the parameters maximizing the *conditional likelihood* of the observed class labels given the model. Usually, naïve Bayes (as a generative model) is trained to maximize the *likelihood*

$$\begin{aligned} \prod_{\theta \in E} P_\lambda(p\theta, q_1\theta, ..., q_k\theta) &= \prod_{\theta \in E} P_\lambda(p\theta | q_1\theta, \cdots, q_k\theta) \cdot P_\lambda(q_1\theta, ..., q_k\theta) \\ &= \prod_{\theta \in E} \prod_j P_\lambda(q_j\theta \mid p\theta) \cdot P_\lambda(p\theta). \end{aligned} \tag{3}$$

Maximum *likelihood* parameters can be computed easily by

$$P_\lambda(\mathbf{q_i} = q_i | \mathbf{p} = p) = \frac{n(\mathbf{q_i} = q_i, \mathbf{p} = p)}{n(\mathbf{p} = p)}$$

where $n(X)$ are the *counts*, i.e., the number of examples for which the query X succeeds. However, there is no closed-form solution for maximum *conditional likelihood* parameters, and relatively slow iterative optimization algorithms have to be used.

Could we use the likelihood as defined by Equation (3) as the score? The problem is that this term is dominated by $P_\lambda(q_1\theta, ..., q_k\theta)$, the likelihood of the propositional dataset that is obtained by evaluating all features. This likelihood is easily maximized for very non-uniform distributions over the features values. In fact, it can be maximized to 1 if all features are constant (always succeed or always fail). Such features are of course completely uninformative with respect to predicting class labels. In contrast, in Equation (2) the likelihood is corrected by the term $P_\lambda(q_1\theta, ..., q_k\theta)$, and in this way informative features are selected. Example 4 illustrates this situation for a single feature. Note that in the setting of parameter estimation from propositional data this problem does not occur as the distribution over the attribute values is determined by the (fixed) training data and cannot be changed by the learning algorithm. However, similar considerations apply to feature selection problems for (propositional) probabilistic models.

Example 4 *Consider the following queries q_1 and q_2:*

$$\leftarrow true$$
$$\leftarrow \textit{perfect-literal}(X)$$

Query q_1 succeeds on all examples. Assume that query q_2 succeeds on all positive and no negative examples, and that half of the examples are positive. Given maximum likelihood parameters, the models H_1/H_2 consisting of only q_1/q_2 actually have the same *likelihood, while conditional likelihood correctly favors H_2:*
For any example θ,

$$\begin{aligned} P_\lambda(p\theta, q_1\theta) &= P_\lambda(q_1\theta|p\theta) \cdot P_\lambda(p\theta) = 1 \cdot 0.5 = 0.5 \\ P_\lambda(p\theta, q_2\theta) &= P_\lambda(q_2\theta|p\theta) \cdot P_\lambda(p\theta) = 1 \cdot 0.5 = 0.5 \end{aligned}$$

as $P_\lambda(q_1\theta|p\theta) = P_\lambda(q_2\theta|p\theta) = 1$. On the other hand,

$$\begin{aligned} P_\lambda(p\theta|q_1\theta) &= \frac{P_\lambda(q_1\theta|p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta)} = \frac{1 \cdot 0.5}{1} = 0.5 \\ P_\lambda(p\theta|q_2\theta) &= \frac{P_\lambda(q_2\theta|p\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_2\theta)} = \frac{1 \cdot 0.5}{0.5} = 1 \end{aligned}$$

as $P_\lambda(q_1\theta) = 1$ but $P_\lambda(q_2\theta) = 0.5$.

Furthermore, a similar problem has been noted by Grossman and Domingos (2004) when learning the structure of Bayesian networks. Here, likelihood maximization leads to over-connected structures, a problem which is also solved by maximizing conditional likelihood. Because finding maximum conditional likelihood parameters is computationally too expensive, Grossman and Domingos propose a "mixed" approach: use conditional likelihood as score, but set parameters to their maximum likelihood values (seen as an approximation to maximum conditional likelihood values).

For NFOIL, we follow the same approach. Parameters are estimated to maximize the likelihood (Equation (3)), while the conditional likelihood, see Equation (2), is retained as score for selecting the features. Assuming that the naïve Bayes assumption is correct, the maximum likelihood estimates of the parameters will approach the maximum conditional likelihood estimates as the number of training examples goes to infinity. This is because the maximum likelihood estimate $P_\lambda(\mathbf{p}\theta, \mathbf{q_1}\theta, ..., \mathbf{q_k}\theta)$ approaches the true joint probability $P(\mathbf{p}\theta, \mathbf{q_1}\theta, ..., \mathbf{q_k}\theta)$ which determines the true conditional $P(\mathbf{p}\theta|\mathbf{q_1}\theta, ..., \mathbf{q_k}\theta)$ that also maximizes the conditional likelihood (Friedman and Goldszmidt, 1996).

This means that parameter estimates are easily obtained by counting, and computational costs for evaluating a hypothesis are dominated by computing for each query the set of examples on which it succeeds. Given this information, the computational cost of scoring a partial clause q_{k+1} in NFOIL is $O(C \cdot n)$ where C is the number of classes and n the number of examples. This assumes appropriate caching mechanisms on the parameters of $\{q_1, ..., q_k\}$ and on partial products of the scoring function (Equation (2)). Thus, evaluating a clause in NFOIL involves basically the same computational costs as scoring in FOIL. FOIL, however, profits from its separate-and-conquer approach; the number of examples is reduced after each iteration. Furthermore, the actual runtime depends on the path in the search space that is taken by the algorithm, which can be different for NFOIL because it may learn different clauses. Nevertheless, the computational complexity of NFOIL is roughly similar to that of FOIL.

Algorithm 2 TFOIL algorithm.

Initialize $H := \emptyset$
repeat
  Initialize $c := p(X_1, \cdots, X_n) \leftarrow$
  repeat
    for all $c' \in \rho(c)$ **do**
      for all $q \in H$ **do**
        compute $s(c', q) = score(E, H \cup \{c'\}, q \rightarrow c', B)$
      end for
      let $pa(c')$ be the $q \in H$ with maximum $s(c', q)$
    end for
    let c be the $c' \in \rho(c)$ with maximum $s(c', pa(c'))$
  until stopping criterion
  add c with dependency $pa(c') \rightarrow c'$ to H
until stopping criterion
output H

5. TFOIL: Relaxing the Naïve Bayes Assumption

The naïve Bayes model employed in NFOIL corresponds to the strong assumption that the probability of an example satisfying one query is independent of its probability to satisfy another query, given the class of the example:

$$\mathbf{P}_\lambda(\mathbf{q_1}, ..., \mathbf{q_k}|\mathbf{p}) = \prod_i \mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p}) \tag{4}$$

Although it has been shown that naïve Bayes can perform well in practice even if this assumption is violated, better models can sometimes be constructed by relaxing this strict independence assumption (Friedman and Goldszmidt, 1996). Tree augmented naïve Bayes (TAN) models generalize naïve Bayes by allowing additional dependencies, but are still significantly more restrictive than full Bayesian networks. The restriction compared to a full Bayesian network is that the additional dependencies form a tree (i.e, every query node has at most one additional parent). This means that the number of parameters of a TAN model is in $O(\#nodes)$ as compared to $O(2^{\#nodes})$ for a full Bayesian network, and learning them is generally easier. In this section, we will discuss the TFOIL algorithm, which extends the NFOIL algorithm presented above to employ tree augmented naïve Bayes. Under the TAN assumption, Equation 4 is relaxed to

$$\mathbf{P}_\lambda(\mathbf{q_1}, ..., \mathbf{q_k}|\mathbf{p}) = \prod_i \mathbf{P}_\lambda(\mathbf{q_i}|\mathbf{p}, \mathbf{q_{pa(i)}})$$

where $\mathbf{q_{pa(i)}}$ is the additional parent of the node $\mathbf{q_i}$ in the TAN model. In analogy to Section 3, the *covers* function can be re-derived as

$$\begin{aligned} covers(E, H, B) =& P(e \mid H, B) \\ =& \frac{\prod_i P_\lambda(q_i\theta|p\theta, q_{pa(i)}\theta) \cdot P_\lambda(p\theta)}{P_\lambda(q_1\theta, \cdots, q_k\theta)}. \end{aligned}$$

Otherwise, the problem specification outlined in Section 3 directly carries over to TFOIL.

In contrast to NFOIL, learning a TFOIL model involves the additional task of selecting a TAN structure over the random variables representing the logical queries. In general, TFOIL follows the same integrated search strategy as outlined for NFOIL in Section 4. However, when evaluating a set of clauses $H_C = \{q_1, ..., q_k\}$ in TFOIL, for every clause $q_i \in H_C$ we have to decide on a $q_j = q_{pa(i)} \in H_C$ for which a dependency $q_{pa(i)} \rightarrow q_i$ is added. Of course, this could be accomplished by running the standard TAN structure learning algorithm, which finds a maximum-likelihood structure in polynomial time (Friedman and Goldszmidt, 1996). However, the incremental way in which a theory H is learned and query nodes are added to the probabilistic model suggests a faster, incremental (though heuristic) way of learning the TAN structure. Rather than re-learning the TAN graph from scratch every time a new candidate clause q_i is scored, the subgraph over the existing hypothesis $H = \{q_1, ..., q_{i-1}\}$ is kept fixed, and all existing clauses $q_j \in H$ are considered as possible parents for the clause q_i. Out of these candidate graph structures, the one maximizing $score(E, H, B)$ is chosen, that is, the maximum-likelihood extension of the existing graph on H.

This approach is outlined in Algorithm 2. The function $score(E, H \cup \{c'\}, q \rightarrow c', B)$ returns the score of the TAN model over $H \cup \{c'\}$, where $\mathbf{q}$ is the additional parent of $\mathbf{c}'$. For every candidate clause c', the best parent $pa(c')$ is identified and the c' with highest score is added to H with the dependency $pa(c') \rightarrow c'$. Comparing Algorithm 2 with the NFOIL algorithm which follows the template of Algorithm parent of a clause that is added to the model. The computational complexity of scoring the $(k+1)th$ clause q_{k+1} in TFOIL is $O(C \cdot k \cdot n)$, as all k existing clauses are considered as possible parents.

6. Experiments

In the following two subsections, we will describe the datasets and algorithms used to experimentally investigate the Questions **Q1–Q5** posed in the introduction (cf. Section 1). Section 6.3 then presents and discusses the results.

6.1 Datasets

We conducted experiments on eight datasets from four domains. Three domains are binary classification tasks, and one is a multi-class problem. See Table 1 for an overview of the different datasets.

Mutagenesis (Srinivasan et al., 1996) is a well-known domain for structure-activity relation prediction. The problem is to classify compounds as mutagenic or not given their chemical structure described in terms of atoms, bonds, atom charge, and information about atom and bond types that have been generated by the molecular modeling package QUANTA. No additional numeric or handcrafted features of the compounds are used. The dataset is divided into two sets: a regression friendly (**r.f.**) set with 188 entries (125 positives, 63 negatives) and a regression unfriendly (**r.u.**) set with 42 entries (13 positives and 29 negatives).

For **Alzheimer** (King et al., 1995), the aim is to compare 37 analogues of Tacrine, a drug against Alzheimer's disease, according to four desirable properties: inhibit **amine** re-uptake, low **toxicity**, high **acetyl** cholinesterase inhibition, and good **reversal** of scopolamine-induced memory deficiency. For any property, examples consist of pairs $pos(X, Y)/neg(X, Y)$ of two analogues indicating that X is better/worse than Y w.r.t. the property. The relation is transitive and anti-

Dataset	#Classes	#Examples	Majority Class	#Relations
Mutagenesis r.f.	2	188	66.5%	4
Mutagenesis r.u.	2	42	69.1%	4
Alzheimer amine	2	686	50.0%	20
Alzheimer toxic	2	886	50.0%	20
Alzheimer acetyl	2	1326	50.0%	20
Alzheimer memory	2	642	50.0%	20
DSSTox	2	232	56.5%	3
Diterpene	23	1530	23.5%	17

Table 1: Datasets used in experiments.

symmetric but not complete (for some pairs of compounds the result of the comparison could not be determined).

The **DSSTox** dataset has been extracted from the EPA's DSSTox NCTRER Database (Fang et al., 2001). It contains structural information about a diverse set of 232 natural, synthetic and environmental estrogens and classifications with regard to their binding activity for the estrogen receptor. In our experiments, only structural information, i.e., atom elements and bonds are used. Additionally, we provided a relation $linked(A_1, A_2, E, BT)$ in the background knowledge that represents that there is a bond of type BT from atom A_1 to atom A_2 and A_2 is of element E. This was done to reduce the lookahead problem for greedy search algorithms.

For **Diterpene** (Džeroski et al., 1998), the task is to identify the skeleton of diterpenoid compounds, given their C-NMR spectra that include the multiplicities and the frequencies of the skeleton atoms. Diterpenes are organic compounds of low molecular weight with a skeleton of 20 carbon atoms. They are of interest because of their use as lead compounds in the search for new pharmaceutical effectors. The dataset contains information on 1530 diterpenes with known structure. There are in total 23 classes. We use the version where both relational and propositional information about the NMR spectra are available.

6.2 Algorithms and Methodology

We investigate the following learners:

- **NFOIL**
  An implementation of the NFOIL algorithm as outlined in Section 4.1. Instead of a greedy search a beam search with beam size $k = 5$ is performed. During the search for a clause, the algorithm also keeps a set C^* of the k best (partial) clauses found so far. If this set does not change from one refinement level to the next, the search is stopped and the best element $c \in C^*$ is returned. The search for additional clauses is stopped if the change in score between two successive iterations is less than 0.1%. A hypothesis is limited to contain at most 25 clauses and a clause to contain at most 10 literals. As most other ILP systems, NFOIL allows the specification of intensional background knowledge to be used in hypothesis. When classifying unseen examples, the class receiving the highest probability is returned (in particular, the default classification threshold of 0.5 is used in the binary case).

- **TFOIL**
  An implementation of the TFOIL algorithm as outlined in Section 5. The beam search and stopping criterion are implemented as for NFOIL.

- **MFOIL**
  MFOIL (Lavrač and Džeroski, 1994) is a variant of FOIL also employing beam search and different search heuristics. The beam size is set to $k = 5$, otherwise, default parameters are used. Note that MFOIL, unlike the other systems considered, by default allows negated literals in clauses.

- **ALEPH**
  ALEPH is an advanced ILP System developed by Ashvin Srinivasan [1]. It is based on the concept of *bottom clauses*, which are maximally specific clauses covering a certain example. The theory is then built from clauses that contain a subset of the literals found in a bottom clause. We used this standard mode of ALEPH for the binary domains **Mutagenesis**, **Alzheimer** and **DSSTox**. Additionally, ALEPH implements a tree learning algorithm, which we used for the multi-class domain **Diterpene**. The maximum number of literals in a clause was set to 10 instead of the default 4. Otherwise, default settings are used except on **DSSTox** as explained below.

- **1BC2**
  1BC2 is a naïve Bayes classifier for structured data (Flach and Lachiche, 2004). 1BC2 was run with a maximum number of 5 literals per clause, as larger values caused the system to crash. The decision threshold was optimized based on a five fold cross validation.

It would also be interesting to compare against the MACCENT system (Dehaspe, 1997), as maximum entropy models and naïve Bayes are somewhat related. Unfortunately, only an implementation of a propositional version of MACCENT is available, which cannot handle the relational datasets used in this study[2]. We have therefore investigated a static propositionalization approach: frequent clauses were extracted from the relational datasets and then used as features in the propositional MACCENT system. More precisely, we have used a variant of the frequent pattern miner WARMR (Dehaspe et al., 1998), as WARMR patterns have shown to be effective propositionalization techniques on similar benchmarks in inductive logic programming (Srinivasan et al., 1999). The variant used was c-ARMR (De Raedt and Ramon, 2004), which allows to remove redundancies amongst the found patterns by focusing on so-called free patterns. c-ARMR was used to generate free frequent patterns in the data with a frequency of at least 20%. However, results obtained using this technique were on average not competitive with those of the other systems, and we decided not to include them.

To compare the different algorithms, we measure both accuracy and area under the ROC curve (see Fawcett, 2003), denoted as AUC. Accuracy is determined by a 10-fold cross-validation on all datasets except the small **Mutagenesis r.u.**, where a leave-one-out cross validation is used instead. ROC curves and AUC are determined from the cross-validation by pooling the rankings obtained on the different test folds. All algorithms were run on the same splits into training/test set for every fold. To test for significant differences in accuracy, a sampled paired t-test is applied to the results of the different folds.

1. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_toc.html`.
2. Luc Dehaspe, personal communication

Dataset	TFOIL	NFOIL	MFOIL	ALEPH	1BC2
Mutagenesis r.f.	79.7 ± 13.0	75.4 ± 12.3	76.6 ± 6.7	69.7 ± 11.9	$\mathbf{82.4}$
Mutagenesis r.u.	83.3 ± 37.7	78.6 ± 41.5	71.4 ± 45.7	$\mathbf{85.7} \pm 35.4$	76.2
Alzheimer amine	$\mathbf{87.5} \pm 4.4$	86.3 ± 4.3	$74.2 \pm 5.9 \bullet \blacktriangle$	$70.9 \pm 5.8 \bullet \blacktriangle$	72.3
Alzheimer toxic	$\mathbf{92.1} \pm 2.6$	89.2 ± 3.4	$81.9 \pm 2.9 \bullet \blacktriangle$	90.9 ± 1.4	83.4
Alzheimer acetyl	$\mathbf{82.8} \pm 3.8$	81.2 ± 5.2	$75.2 \pm 2.5 \bullet \blacktriangle$	$73.9 \pm 3.4 \bullet \blacktriangle$	73.4
Alzheimer memo.	$\mathbf{80.4} \pm 5.3 \triangle$	$72.9 \pm 4.3 \bullet$	$60.9 \pm 4.6 \bullet \blacktriangle$	$69.2 \pm 5.3 \bullet$	68.8
DSSTox	$\mathbf{78.5} \pm 8.9$	78.0 ± 9.1	70.4 ± 15.4	$52.1 \pm 11.2 \bullet \blacktriangle$	64.7
Diterpene	$\mathbf{90.9} \pm 2.1$	90.8 ± 3.1	–	$85.0 \pm 3.6 \bullet \blacktriangle$	81.9

Table 2: Cross-validated predictive accuracy results on all data sets. •/○ indicates that TFOIL's mean is significantly higher/lower, and ▲/△ that NFOIL's mean is significantly higher/lower (paired sampled t-test, $p = 0.05$). Bold numbers indicate the best result on a dataset. Note that the high variance on **Mutagenesis r.u.** is an artifact of the leave-one-out cross-validation. For **1BC2**, we do not test significance because the results on Mutagenesis and Diterpene are taken from Flach and Lachiche (2004).

6.3 Results

Table 2 shows the accuracy results for TFOIL, NFOIL, MFOIL, ALEPH and 1BC2 on all datasets. There is no result for MFOIL on Diterpene as it cannot handle multi-class problems. Comparing the results for NFOIL/TFOIL and MFOIL, the experiments clearly show a gain for dynamic propositionalization over the baseline ILP algorithm, giving an affirmative answer to Question **Q1**.

The comparison between NFOIL and its tree augmented extension TFOIL shows that—as in the propositional case—relaxing the naïve Bayes assumption can yield more accurate models in some cases. TFOIL always gains some predictive accuracy, with gains ranging from very slight (0.1 percentage points) to substantial (7.5 percentage points). Although only one of these gains is significant according to a paired sampled t-test on the folds, a simple sign test on the results of TFOIL and NFOIL $(8/0)$ shows that TFOIL significantly outperforms NFOIL ($p < 0.01$). This affirmatively answers Question **Q3**: performance can be improved by using a more expressive model than naïve Bayes. Furthermore, TFOIL significantly outperforms ALEPH on five datasets, and both a sign test $(7/1)$ and the average accuracy seem to favor TFOIL and NFOIL over ALEPH. This shows that relatively simple dynamic propositionalization approaches are competitive with more advanced ILP systems, giving an affirmative answer to Question **Q4**. The **DSSTox** domain seems to be particularly hard for the ALEPH system. Using standard settings, ALEPH only accepts rules if they cover no negative examples. In this case, for most folds it does not return any rules on the **DSSTox** dataset, and only reaches 43.1% accuracy. The result reported above was obtained by setting the "noise" parameter to 10 (which means that a rule can cover up to 10 negative examples). The *minacc* parameter was left at its default value of 0.

To complement the study of predictive accuracy presented above, we investigate the performance of the probabilistic classifiers by means of ROC analysis. ROC curves evaluate how well the probability estimates produced by classifiers can discriminate between positive and negative examples (i.e., *rank* examples) without committing to a particular decision threshold. ROC curves provide more detailed information about performance than accuracy estimates, for example with

Dataset	TFOIL	NFOIL	MFOIL	ALEPH	1BC2
Mutagenesis r.f.	**0.817**	0.809	0.791	0.713	0.816
Mutagenesis r.u.	0.753	0.737	0.645	**0.790**	0.729
Alzheimer amine	**0.945**	0.937	0.747	0.708	0.793
Alzheimer toxic	**0.983**	0.965	0.821	0.912	0.925
Alzheimer acetyl	**0.932**	0.916	0.759	0.752	0.815
Alzheimer memory	**0.913**	0.824	0.608	0.696	0.744
DSSTox	**0.789**	0.760	0.668	0.567	0.636

Table 3: Area under the ROC curve for all binary data sets. Bold numbers indicate the best result on a dataset. Results for **1BC2** on Mutagenesis are taken from Flach and Lachiche (2004). Rankings are pooled over the different folds of a 10-fold cross-validation, except for Mutagenesis r.u. where a leave-one-out cross-validation is used instead.

regard to possible error trade-offs under variable misclassification costs (Provost et al., 1998). As ROC curves are only well-defined for binary classification problems, we do not report results for the multi-class dataset **Diterpene**. For the ILP systems, ROC curves were produced by clause voting as introduced by Davis et al. (2004). In clause voting, the threshold that is varied is the number of clauses that have to cover an example before it is classified as positive (the default threshold being one).

AUC results, shown in Table 3, generally confirm the results obtained by accuracy analysis: TFOIL outperforms NFOIL across the board, and TFOIL/NFOIL generally produce better rankings than the ILP methods and 1BC2. Note that in some cases dynamic propositionalization achieves higher AUC scores than ILP systems even though it achieves lower accuracy (e.g., comparing NFOIL and ALEPH on the **Alzheimer toxic** dataset). To investigate this behavior in more detail, ROC curves for the different methods on all binary datasets are shown. Figure 1 shows curves on the relatively large **Alzheimer amine**, **Alzheimer toxic**, **Alzheimer acetyl** and **Alzheimer memory** datasets. Here, ROC curves clearly fall into two groups: for the ILP systems MFOIL and ALEPH, the curve has one sharp angle and is otherwise mostly linear, while the curves for TFOIL, NFOIL and 1BC2 are convex almost across the whole range of the threshold parameter. This means that for the ILP systems there is one (namely, the default) decision threshold which offers a good trade-off between true positives and false positives, but ranking below and above this point is relatively poor. At the level of induced clauses, it indicates that clauses induced by the ILP systems are specific in the sense that positive examples are typically covered by only one clause—if the decision threshold in clause voting is set to more than one, the true positive rate drops rapidly. In contrast, ranking performance for the dynamic propositionalization systems and 1BC2 is more stable, meaning that these systems also offer good classification performance under varying misclassification costs (or, equivalently, class skew). This indicates that dynamic propositionalization approaches can make use of more diverse rule sets, which are helpful in ranking examples by providing additional information but would produce too many false-positive classifications if interpreted as a disjunctive hypothesis. We will provide further evidence for this claim below.

Figure 2 shows ROC curves on the three smaller datasets included in our study. On **Mutagenesis r.f.** and **DSSTox**, similar observations hold as noted above, although class separation is

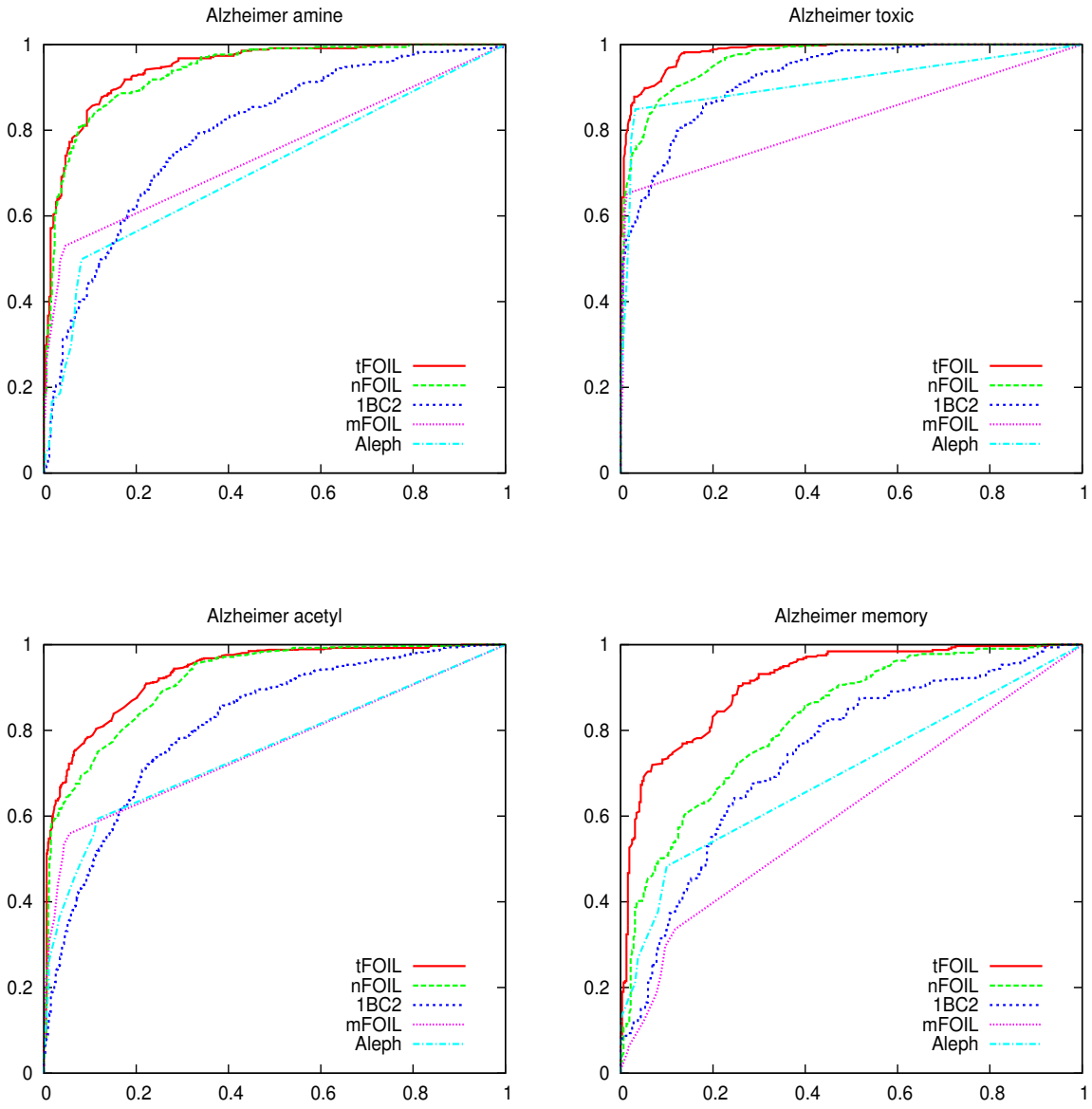

Figure 1: ROC curves on the Alzheimer amine, Alzheimer toxic, Alzheimer acetyl and Alzheimer memory datasets for TFOIL, NFOIL, MFOIL, ALEPH and 1BC2. Rankings are pooled over the different folds of a 10-fold cross-validation.

generally poorer and curves behave less well. On the very small **Mutagenesis r.u.** dataset ranking performance is poor for all methods.

To investigate Question **Q2**, i.e., to compare dynamic and static propositionalization approaches, two additional experiments were performed:

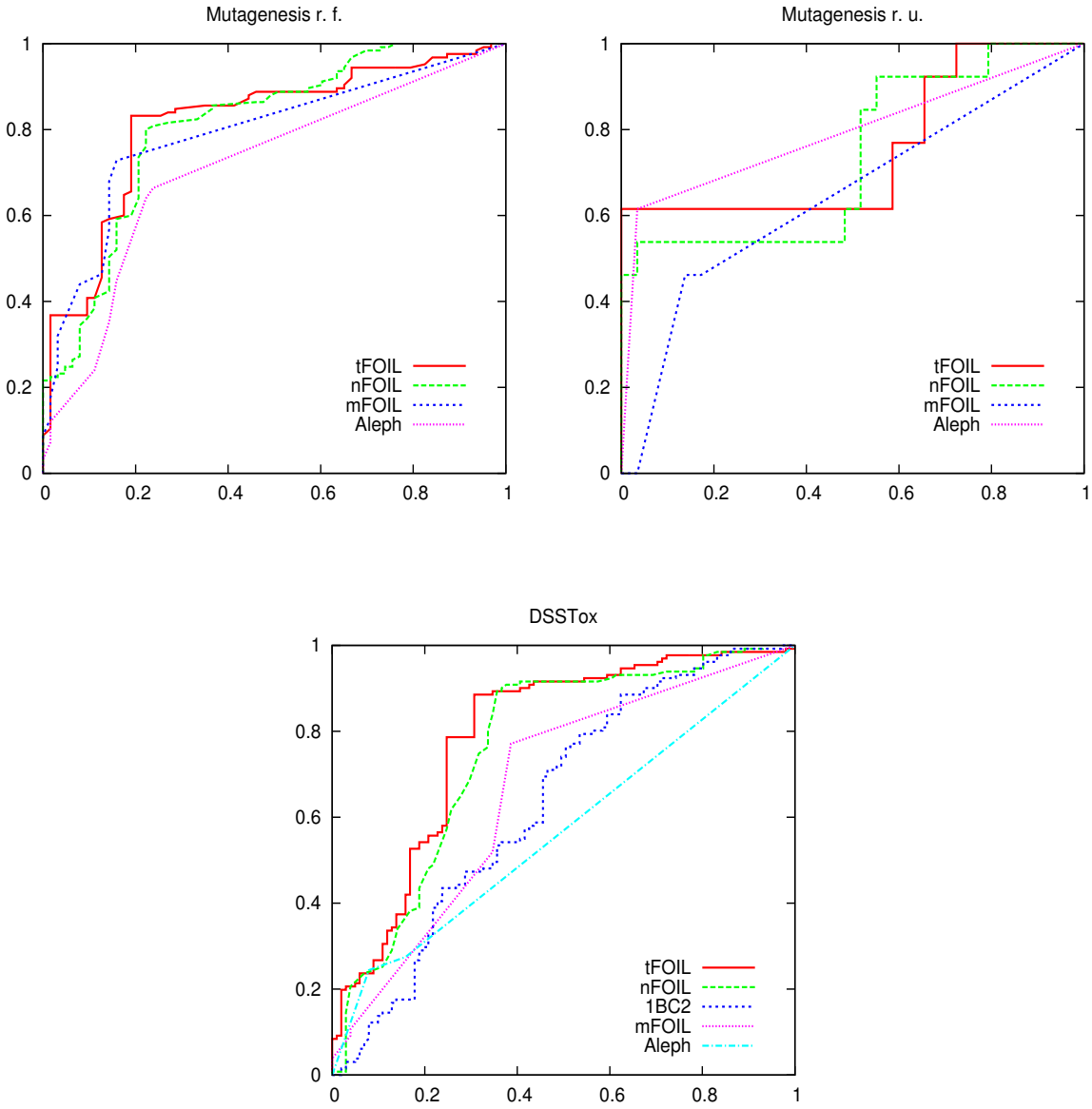

Figure 2: ROC curves on the Mutagenesis regression friendly, Mutagenesis regression unfriendly and DSSTox datasets for TFOIL, NFOIL, MFOIL, ALEPH and 1BC2. There is no curve for 1BC2 on the Mutagenesis datasets because results are taken from Flach and Lachiche (2004). Rankings are pooled over the different folds of a 10-fold cross-validation, except for Mutagenesis r.u. where a leave-one-out cross-validation is used instead.

	MFOIL		ALEPH		
Dataset	**+NB**	**+TAN**	**+NB**	**+TAN**	**NFOIL/disjunctive**
Mutagenesis r.f.	± 0.0 •	± 0.0 •	± 0.0	± 0.0	-8.9 • ▲
Mutagenesis r.u.	± 0.0	$+2.4$	± 0.0	± 0.0	-47.6 • ▲
Alzheimer amine	-0.5 • ▲	± 0.0 • ▲	± 0.0 • ▲	± 0.0 • ▲	-36.3 • ▲
Alzheimer toxic	± 0.0 • ▲	$+0.2$ • ▲	± 0.0	± 0.0	-39.2 • ▲
Alzheimer acetyl	-0.4 • ▲	± 0.0 • ▲	± 0.0 • ▲	$+0.1$ • ▲	-31.2 • ▲
Alzheimer memory	± 0.0 • ▲	± 0.0 • ▲	± 0.0 •	± 0.0 •	-22.9 • ▲
DSSTox	$+2.1$	$+1.7$	$+4.3$ • ▲	$+4.3$ • ▲	-21.5 • ▲
Diterpene	–	–	± 0.0 • ▲	-0.3 • ▲	–

Table 4: Gain/loss in cross-validated predictive accuracy for the two-step methods MFOIL+NB/TAN, ALEPH+NB/TAN and NFOIL/DISJUNCTIVE over their corresponding baselines MFOIL, ALEPH and NFOIL. •/○ indicates that TFOIL's mean accuracy is significantly higher/lower, and ▲/△ that NFOIL's mean accuracy is significantly higher/lower (paired sampled t-test, $p = 0.05$).

1. Learning a naïve Bayes or tree augmented naïve Bayes model over a set of clauses using a two-step approach: First, a set of rules is learned using MFOIL or ALEPH, and afterwards a (tree augmented) naïve Bayes model is built using these rules. This is a static propositionalization approach, where the propositionalized data is used as input for the probabilistic learner. Question **Q2** is whether these rules are less useful when combined with (tree augmented) naïve Bayes than the rules constructed by a dynamic propositionalization approach (NFOIL/TFOIL). For the training of the TAN model we used the local score based TAN implementation in WEKA 3.4.6 (Witten and Frank, 2000), which implements the learning algorithm of Friedman and Goldszmidt (1996).

2. Using the rules learned by NFOIL as a disjunctive hypothesis Q: For this, every rule q learned by NFOIL is evaluated on the training set. If it covers more positive than negative examples, q is added to Q, otherwise $not(q)$. The rule set Q is then evaluated as a disjunctive hypothesis on the test data. This technique can only be used for binary classification problems.

Table 4 shows the result of these experiments. It displays the average gain/loss of the two-step methods MFOIL+NB, ALEPH+NB and NFOIL/disjunctive over their corresponding baselines MFOIL, ALEPH and NFOIL. Significantly higher/lower mean accuracies of a method as compared to TFOIL/NFOIL are also indicated. On most datasets, applying naïve Bayes as a method for post-processing the learned rule set of MFOIL or ALEPH does not yield any improvement, with the exception of small gains on the **DSSTox** and **Mutagenesis r.u.** datasets for MFOIL. ALEPH+NB/TAN only improves on the original result of ALEPH on **DSSTox** by always predicting the majority class. Furthermore, NFOIL/TFOIL significantly outperform MFOIL+NB and MFOIL+TAN on the same datasets on which they significantly outperform MFOIL. In ROC space, post-processing rules with (tree augmented) Naïve Bayes can increase or decrease the performance depending on the dataset (Table 5). However, AUC scores of the static propositionalization approaches are on average much lower than those of dynamic propositionalization approaches.

	MFOIL		ALEPH	
Dataset	**+NB**	**+TAN**	**+NB**	**+TAN**
Mutagenesis r.f.	$+0.045$	$+0.036$	$+0.011$	-0.034
Mutagenesis r.u.	-0.183	-0.183	-0.175	-0.063
Alzheimer amine	$+0.001$	$+0.007$	-0.013	-0.012
Alzheimer toxic	$+0.003$	$+0.011$	$+0.006$	$+0.005$
Alzheimer acetyl	$+0.002$	-0.003	-0.006	-0.010
Alzheimer memory	-0.010	-0.016	-0.010	-0.027
DSSTox	$+0.054$	$+0.048$	-0.095	-0.099

Table 5: Gain/loss in area under ROC curve for the two-step methods MFOIL+NB/TAN and ALEPH+NB/TAN over their corresponding baselines MFOIL and ALEPH. Rankings are pooled over the different folds of a 10-fold cross-validation, except for Mutagenesis r.u. where a leave-one-out cross-validation is used instead.

Using the (possibly negated) rules learned by NFOIL as a disjunctive hypothesis strongly degrades performance. This is because NFOIL can make use of rules which have a very low accuracy individually but still help as additional features in the naïve Bayes. If these rules are used for disjunctive classification, they produce many false-positive classifications. In fact, on some domains (e.g, **Alzheimer**) all examples in the test set are always classified as positive. This shows that NFOIL uses significantly different rule sets to ILP approaches. Thus, we can answer **Q2** as follows:

> A dynamic propositionalization approach that selects rules based on the criterion of the probabilistic model performs better than static propositionalization approaches that post-process a rule set using a probabilistic learner. This is because dynamic propositionalization can make use of different/additional rules that are not considered by traditional ILP systems as they would produce too many false-positive classifications.

It remains to answer Question **Q5**, that is, to compare dynamic propositionalization approaches to relational naïve Bayes methods such as 1BC2 in terms of accuracy and the number of features they employ. With respect to predictive accuracy and AUC score, sign tests prefer TFOIL over 1BC2 (at 7/1 and 8/0, respectively). Theory complexity is hard to compare because of the different representations. For TFOIL/NFOIL and 1BC2, one complexity measure is the number of probability values attached to clauses. There are $\#$*classes* probability values attached to each clause in NFOILand 1BC2, and $2 \cdot \#$*classes* in TFOIL ($\#$*classes* denotes the number of classes). Additionally, we have to specify the prior distribution over the class variable. The overall number of probability values to be specified is thus of the order of $\mathcal{O}(\#$*clauses*$)$, and it is sufficient to compare the number of clauses. In the experiments, 1BC2 uses an order of magnitude more clauses than NFOIL. More precisely, 1BC2 uses more than 400 (in some cases even more than 1000) clauses whereas NFOIL is limited to using 25 clauses. This clearly shows that **Q5** can be answered affirmatively as well.

We furthermore investigated whether the proposed dynamic propositionalization approach sometimes constructs too many clauses and overfits the training data, as the stopping criterion is based on the training set score. We therefore tried post-pruning a learned hypothesis. Post-pruning is more easily realized for NFOIL than for TFOIL, as the additional TAN structure in the TFOIL model prevents removal of rules which are parents of other rules. Rule post-pruning was carried out us-

Algorithm 3 Post-pruning a hypothesis learned by NFOIL.

Initialize $H := \text{NFOIL}(E, B)$
repeat
  for all $c \in H$ **do**
    $s(c) :=$ *cross-validate-accuracy*$(H \setminus \{c\}, E, B)$
  end for
  $c^* := \arg\max_{c \in H} s(c)$
  $H := H \setminus \{c^*\}$
until cross-validated accuracy decreases
output H

	NFOIL		**NFOIL/pruning**	
Dataset	Accuracy	#clauses	Accuracy	#clauses
Mutagenesis r.f.	$\mathbf{75.4} \pm 12.3$	25.0	73.9 ± 12.1	17.7
Mutagenesis r.u.	78.6 ± 41.5	23.1	$\mathbf{85.7} \pm 35.4$	1.4
Alzheimer amine	$\mathbf{86.3} \pm 4.3$	24.7	85.0 ± 4.5	20.1
Alzheimer toxic	$\mathbf{89.2} \pm 3.4$	22.4	87.8 ± 3.6	16.6
Alzheimer acetyl	$\mathbf{81.2} \pm 5.2$	25.0	80.8 ± 4.2	20.5
Alzheimer memory	72.9 ± 4.3	24.9.5	$\mathbf{74.5} \pm 4.3$	20.4
DSSTox	78.0 ± 9.1	15.4	$\mathbf{79.3} \pm 9.7$	4.5
Diterpene	$\mathbf{90.8} \pm 3.1$	25.0	90.7 ± 3.1	23.4

Table 6: Cross-validated predicative accuracy results and average number of clauses in the final model for NFOIL and NFOIL/pruning. Bold numbers indicate the best result on a dataset. The are no significant differences in mean accuracy between the two methods (paired sampled t-test, $p = 0.05$).

ing the greedy algorithm outlined in Algorithm 3. The procedure *cross-validate-accuracy(H,E,B)* cross-validates the naïve Bayes model on the training data for a fixed set H of clauses and returns an accuracy estimate. The algorithm greedily drops clauses from H as long as this does not decrease the cross-validated accuracy estimate. Table 6 lists the accuracies of NFOIL and NFOIL/pruning which incorporates this rule post-pruning algorithm. There is some gain in accuracy on the small **Mutagenesis r.u.** and the **DSSTox** domain, although no differences in accuracy are significant at the $p = 0.05$ level. On these two datasets the number of features is also greatly reduced, while few or no features are pruned for the other datasets. To summarize, a clear overfitting behavior can not be observed except possibly on the very small **Mutagenesis r.u.** dataset.

We conclude that our experimental study affirmatively answers Questions **Q1**–**Q5** posed in the introduction. The dynamic propositionalization approaches NFOIL and TFOIL yield more accurate models than simple ILP rule learning and static propositionalization approaches, and also compare favorably to the first order naïve Bayes system 1BC2 and one of the most advanced ILP systems, namely Aleph.

7. Related Work

The approaches that combine statistical learning with inductive logic programming techniques for addressing classification can be divided into three categories.

A first class of techniques are static propositionalization approaches. They start by generating a set of first order features and then use these features as attributes in a probabilistic model. Probabilistic models of different expressivity have been used, ranging from naïve Bayes (Pompe and Kononenko, 1995; Flach and Lachiche, 2004), to tree augmented naïve Bayes or full Bayesian networks as in (Davis et al., 2004). The set of features is obtained either by taking all features within a pre-defined language bias, as in the 1BC system (Flach and Lachiche, 2004), or by running a traditional ILP algorithm (Pompe and Kononenko, 1995; Davis et al., 2004). Furthermore, aggregation-based feature construction methods such as RELAGGS (Krogel and Wrobel, 2001) and ACORA (Perlich and Provost, 2006) that search a relational feature space using aggregation operators fall into this group. In this class of techniques the feature construction and the statistical learning steps are performed consecutively and independent of one another, whereas in NFOIL and TFOIL they are tightly integrated. However, an initial step beyond static propositionalization has been taken in the work by Pompe and Kononenko (1997), where rules generated by an ILP system are post-processed by splitting and merging clauses in order to find a rule set that satisfies the naïve Bayes assumption.

A second class of techniques employs a rich probabilistic model such as a Probabilistic Relational Model (Getoor et al., 2001) or a higher-order probabilistic logic (Flach and Lachiche, 2004). The logical component (and hence the features) of such a model are fixed, and only the parameters are learned using statistical learning techniques. The work by Taskar et al. (2001) on using Probabilistic Relational Models for clustering and classification of relational data, the Relational Bayesian Classifier (Neville et al., 2003) and the 1BC2 system (Flach and Lachiche, 2004) fall into this category. The difference compared to NFOIL and TFOIL is that this class of techniques does not address structure learning or feature generation.

A third class of techniques (Popescul et al., 2003; Dehaspe, 1997) indeed tightly integrates the inductive logic programming step with the statistical learning step in a dynamic propositionalization approach. However, whereas the dynamic propositionalization methods presented in this paper employ the simplest possible statistical model, namely naïve Bayes, those approaches use more advanced (and hence computationally more expensive) statistical models such as logistic regression and maximum entropy modeling, which does seem to limit the application potential. For instance, Popescul et al. (2003) report that—in their experiments—they had to employ a depth limit of 2 when searching for features. The work on NFOIL and TFOIL is similar in spirit to these two approaches but is much more simple and therefore, we believe, also more appealing for the traditional classification task considered in inductive logic programming.

Probably the most closely related approach to the methods presented in this paper is the SAYU system (Davis et al., 2005). SAYU uses a "wrapper" approach where (partial) clauses generated by the refinement search of an ILP system are proposed as features to a (tree augmented) naïve Bayes, and incorporated if they improve performance. This means that feature learning and naïve Bayes are tightly coupled as in our approach. However, in SAYU the scores for feature and parameter selection are different, and the feature selection is based on a separate tuning set. The probabilistic model is trained to maximize the likelihood on the training data, while clause selection is based on the area under the precision-recall curve of the model on a separate tuning set. This contrasts

with the approach of selecting features and parameters that jointly optimize a probabilistic score on the training data used in NFOIL and TFOIL. Davis et al. (2005) also report that a tree-augmented naïve Bayes model in SAYU does not significantly outperform a naïve Bayes, which contrasts with the results obtained in this study. Furthermore, SAYU cannot handle multi-class problems, while NFOIL and TFOIL handle them as naturally as naïve Bayes does.

Finally, there is also the approach of Craven and Slattery (2001) who combine several naïve Bayes models with FOIL. The decisions of naïve Bayes models are viewed as truth values of literals occurring in clauses. This work can be regarded as the inverse of the approach presented in this paper in that NFOIL/TFOIL employ naïve Bayes on top of logic, whereas Craven and Slattery employ naïve Bayes as a predicate in the logical definitions.

8. Conclusions

We have introduced the NFOIL and TFOIL systems, two dynamic propositionalization approaches that combine the simplest techniques from ILP and probabilistic learning. In an experimental study on several benchmark datasets, the proposed approaches were compared to the ILP systems MFOIL and ALEPH, static propositionalization approaches and the first-order naïve Bayes system 1BC2. Experimental results show that dynamic propositionalization is superior to static propositionalization and simple rule learning. Despite their simplicity, the proposed approaches are also competitive with the more advanced ILP system ALEPH. Moreover, our experiments indicate that the superior performance of dynamic propositionalization approaches is due to the fact that they can make use of more diverse rule sets than ILP systems. NFOIL and TFOIL are also particularly strong at ranking examples (as measured by ROC analysis), a task in which standard ILP systems performed rather poorly in our experiments.

Further exploring dynamic propositionalization as a way of combining (possibly more powerful) statistical learners with ILP search techniques is an interesting direction for future work.

Acknowledgments

We would like to thank the three anonymous reviewers for their helpful comments and suggestions, which have greatly improved this paper. We furthermore thank Romaric Gaudel for fruitful discussions and help with performing the experimental study. We acknowledge support for this work from the European Union under contract number FP6-508861, Applications of Probabilistic Inductive Logic Programming II.

References

Hendrik Blockeel and Luc De Raedt. Lookahead and Discretization in ILP. In N. Lavrač and S. Džeroski, editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming (ILP-1997)*, volume 1297 of *Lecture Notes in Computer Science*, pages 77–84. Springer, 1997.

Marc Craven and Seán Slattery. Relational Learning with Statistical Predicate Invention: Better Models for Hypertext. *Machine Learning*, 43(1–2):97–119, 2001.

Jesse Davis, Irene M. Ong Vítor Santos Costa, David Page, and Inês Dutra. Using Bayesian Classifiers to Combine Rules. In *Working Notes of the Third Workshop on Multi-Relational Data Mining (MRDM-2004) in conjunction with the Tenth ACM SIGKDD International Conference on Knowlege Discovery and Data Mining (KDD-2004)*, Seattle, Washington, USA, 2004.

Jesse Davis, Elizabeth Burnside, Inês de Castro Dutra, David Page, and Vítor Santos Costa. An Integrated Approach to Learning Bayesian Networks of Rules. In Joao Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge, and Luís Torgo, editors, *Proceedings of the Sixteenth European Conference on Machine Learning (ECML-2005)*, volume 3720 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2005.

Luc De Raedt and Kristian Kersting. Probabilistic Logic Learning. *ACM-SIGKDD Explorations*, 5 (1):31–48, 2003.

Luc De Raedt and Kristian Kersting. Probabilistic Inductive Logic Programming. In S. Ben-David, J. Case, and A. Maruoka, editors, *Proceedings of the Fifteenth International Conference on Algorithmic Learning Theory (ALT-2004)*, volume 3244 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2004.

Luc De Raedt and Jan Ramon. Condensed Representations for Inductive Logic Programming. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR-2004)*, Whistler, Canada, 2004. AAAI Press.

Luc Dehaspe. Maximum Entropy Modeling with Clausal Constraints. In N. Lavrač and S. Džeroski, editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming (ILP-1997)*, volume 1297 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 1997.

Luc Dehaspe, Hannu Toivonen, and Ross D. King. Finding Frequent Substructures in Chemical Compounds. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-1998)*, New York City, New York, USA, 1998. AAAI Press.

Saso Džeroski, Steffen Schulze-Kremer, Karsten Heidtke, Karsten Siems, Dietrich Wettschereck, and Hendrik Blockeel. Diterpene Structure Elucidation from ^{13}C NMR Spectra with Inductive Logic Programming. *Applied Artificial Intelligence, Special Issue on First-Order Knowledge Discovery in Databases*, 12:363–383, 1998.

Hong Fang, Weida Tong, Leming M. Shi, Robert Blair, Roger Perkins, William Branham, Bruce S. Hass, Qian Xie, Stacy L. Dial, Carrie L. Moland, and Daniel M. Sheehan. Structure-Activity Relationships for a Large Diverse Set of Natural, Synthetic, and Environmental Estrogens. *Chemical Research in Toxicology*, 14(3):280–294, 2001.

Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003.

Peter Flach and Nicholas Lachiche. Naive Bayesian Classification of Structured Data. *Machine Learning*, 57(3):233–269, 2004.

Nir Friedman and Moises Goldszmidt. Building Classifiers Using Bayesian Networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-1996), Vol. 2*, pages 1277–1284, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.

Lise Getoor and Ben Taskar, editors. *Statistical Relational Learning*. MIT Press, 2007. In press.

Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning Probabilistic Relational Models. In *Relational Data Mining*. Springer, 2001.

Daniel Grossman and Peter Domingos. Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*, pages 361–368, Banff, Canada, 2004. ACM Press.

Ross D. King, Ashvin Srinivasan, and Michael J, E Sternberg. Relating Chemical Activity to Structure: an Examination of ILP Successes. *New Generation Computing*, 13(2,4):411–433, 1995.

Mark A. Krogel and Stefan Wrobel. Transformation-based Learning Using Multirelational Aggregation. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP-2001)*, volume 2157 of *Lecture Notes in Computer Science*. Springer, 2001.

Niels Landwehr, Kristian Kersting, and Luc De Raedt. nFOIL: Integrating Naïve Bayes and FOIL. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*, pages 795–800, Pittsburgh, Pennsylvania, USA, 2005. AAAI Press.

Nada Lavrač and Saso Džeroski. *Inductive Logic Programming*. Ellis Horwood, 1994.

Stephen Muggleton. Inverse Entailment and Progol. *New Generation Computing, Special Issue on Inductive Logic Programming*, 13:245–286, 1995.

Stephen Muggleton. Stochastic Logic Programs. In *Advances in Inductive Logic Programming*. IOS Press, 1996.

Stephen Muggleton and Luc De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

Stephen Muggleton and Cao Feng. Efficient Induction of Logic Programs. In *Proceedings of the First Conference on Algorithmic Learning Theory (ALT-1990)*, pages 368–381, Tokyo, Japan, 1990. Springer.

Jennifer Neville, David Jensen, and Brian Gallagher. Simple Estimators for Relational Bayesian Classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM-2003)*, pages 609–612, Melbourne, Florida, USA, 2003. IEEE Computer Society.

Claudia Perlich and Foster Provost. Distribution-based Aggregation for Relational Learning with Identifier Attributes. *Machine Learning*, 62:65–105, 2006.

Uros Pompe and Igor Kononenko. Naive Bayesian Classifier within ILP-R. In *Proceedings of the Fifth International Workshop on Inductive Logic Programming (ILP-1995)*, pages 417–436, Tokyo, Japan, 1995.

Uros Pompe and Igor Kononenko. Probabilistic First-Order Classification. In N. Lavrač and S. Džeroski, editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming (ILP-1997)*, volume 1297 of *Lecture Notes in Computer Science*, pages 235–242. Springer, 1997.

Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical Relational Learning for Document Mining. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM-2003)*, pages 275–282, Melbourne, Florida, USA, 2003. IEEE Computer Society.

Foster J. Provost, Tom Fawcett, and Ron Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *Proceeding of the Fifteenth International Conference on Machine Learning (ICML-1998)*, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

J. Ross Quinlan. Learning Logical Definitions from Relations. *Machine Learning*, pages 239–266, 1990.

Jorma Rissanen. Modeling by Shortest Data Description. *Automatica*, 14:465–471, 1978.

Ashvin Srinivasan, Stephen Muggleton, Ross D. King, and Michael J. E. Sternberg. Theories for Mutagenicity: a Study of First-order and Feature Based Induction. *Artificial Intelligence*, 85: 277–299, 1996.

Ashwin Srinivasan, Ross D. King, and Douglas W. Bristol. An Assessment of ILP-Assisted Models for Toxicology and the PTE-3 Experiment. In Saso Dzeroski and Peter A. Flach, editors, *Proceedings of the Ninth Internatinal Workshop on Inductive Logic Programming (ILP-1999)*, volume 1634 of *Lecture Notes in Computer Science*. Springer, 1999.

Ben Taskar, Eran Segal, and Daphne Koller. Probabilistic Clustering in Relational Data. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 870–878, Seattle, Washington, USA, 2001. Morgan Kaufmann.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implemenations*. Morgan Kaufmann, 2000.