nFOIL: Integrating Naïve Bayes and FOIL

Niels Landwehr and **Kristian Kersting** and **Luc De Raedt**
University of Freiburg, Machine Learning Lab
Georges-Koehler-Allee, Building 079, 79110 Freiburg, Germany
{landwehr,kersting,deraedt}@informatik.uni-freiburg.de

Abstract

We present the system nFOIL. It tightly integrates the naïve Bayes learning scheme with the inductive logic programming rule-learner FOIL. In contrast to previous combinations, which have employed naïve Bayes only for post-processing the rule sets, nFOIL employs the naïve Bayes criterion to directly guide its search. Experimental evidence shows that nFOIL performs better than both its base line algorithm FOIL or the post-processing approach, and is at the same time competitive with more sophisticated approaches.

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 & Kersting 2003; Dietterich, Getoor, & Murphy 2004). Whereas the typical approaches upgrade existing probabilistic learning schemes to deal with relational or logical data (such as e.g. PRMs (Getoor *et al.* 2001) or SLPs (Muggleton 1996)), we start from an inductive logic programming system and extend it with a probabilistic model. More specifically, we have selected the simplest approaches from both domains, i.e. the inductive logic programming system FOIL (Quinlan 1990) and naïve Bayes, and integrated them in the nFOIL system. The advantage of combining such simple learning schemes is that the resulting probabilistic logical or relational model is easy to understand and interpret.

In relational learning or inductive logic programming, one typically induces a set of rules (or clauses). The resulting rule-set then defines a disjunctive hypothesis, since 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 "attributes" 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*".

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

This idea is not new. It has been pursued by Pompe and Kononenko (1995) and Davis *et al.* (2004). In the existing approaches for combining ILP and naïve Bayes, however, one learns the model in two *separate* steps. First, the features or clauses have been generated (e.g. using an existing inductive logic programming system such as ILP-R (Pompe & Kononenko 1995)) and then the probabilities for the naïve Bayes estimates are determined. This actually corresponds to a *propositionalization* approach, where the propositionalized problem is learned using naïve Bayes. We propose a *different* and *novel* approach, in which the learning of the features is tightly integrated with naïve Bayes. The advantage is that the criterion according to which the features are generated is that of naïve Bayes. The idea of guiding the structure search by the probabilistic score contrasts with the propositionalization approaches, in which the criteria employed for feature generation and classification are different, and has also been suggested by Popescul *et al.* (2003) for learning relational logistic regression models.

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

We will proceed as follows. After reviewing FOIL and its learning setting in the next section, we lift FOIL's learning setting to the probabilistic case and introduce the nFOIL system. In a subsequent section, we experimentally evaluate nFOIL on ILP benchmark datasets. Finally, we conclude and touch upon related work.

FOIL's Problem Specification

The problem that we tackle in this paper is a (probabilistic) inductive logic programming problem, which can be formalized as follows:

Given

- a background theory B, in the form of a set of definite clauses, i.e., clauses of the form $h \leftarrow b_1, \cdots, b_n$ where h and the b_i are logical atoms, and $b_1, \cdots, b_n$ is called *body* of the clause;
- a set of classified examples E, in the form of ground facts;

- 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, with respect to a hypothesis H, and the background theory B;
- a $score(E, H, B)$ function, which specifies the quality of the hypothesis H with respect to the data E and the background theory;

Find

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

The traditional approaches to inductive logic programming 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.

This setting is incorporated in many well known inductive logic programming systems such as FOIL (Quinlan 1990), GOLEM (Muggleton 1990), PROGOL (Muggleton 1995) and TILDE (Blockeel & De Raedt 1997).

Example 1 *To illustrate this setting, consider the following background theory B (inspired on the well known mutagenicity benchmark (Srinivasan* et al. *1996)):*

$$\begin{array}{ll} atom(mol_1, a_0, c) & bond(mol_1, a_0, a_{10}, 7) \\ atom(mol_1, a_1, n) & bond(mol_1, a_1, a_3, 7) \\ atom(mol_1, a_2, o) & bond(mol_1, a_3, a_4, 2) \\ \cdots & \cdots \end{array}$$

and the example $muta(mol_1)$. It is covered by the following hypothesis:

$$muta(X) \leftarrow$$
$$atom(X, A, c), atom(X, B, o), bond(X, A, B, 7)$$
$$muta(X) \leftarrow atom(X, A, fl), bond(X, A, B, 2)$$

FOIL, like many inductive logic programming systems, follows a *separate-and-conquer* approach to induce a hypothesis. Such an algorithm is described in Algorithm 1. The algorithm repeatedly searches for clauses that score well with respect to the data set and the current hypothesis and adds them to the current hypothesis. An *update* function removes examples from E that are covered by the current hypothesis H, i.e.,

$$update(E, H) = E \setminus covered(H) \ .$$

The inner loop 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 $c_1\theta \subseteq c_2$. The most general clause is

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

Algorithm 1 Summarizing the 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
```

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. Many different scoring functions and stopping criteria have been employed.

nFOIL's Problem Specification

We now show 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 choices, such as the form of examples and hypotheses, will be untouched.

A probabilistic $covers$ function

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

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

$$covers(e, H \cup B) = P(e \mid H, B) \ .$$

where H and B are as before, i.e. a set of clauses defining the target predicate p and a background theory. 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 examples 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 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 $c \in H$ of the form $p(X_1, \cdots, X_n) \leftarrow b_1, \cdots, b_n$ we view the query q_c, i.e. $\leftarrow b_1, \cdots, b_n$, as a boolean feature or attribute. Applied to an example θ these queries become

instantiated, e.g. $\leftarrow b_1\theta, \cdots, b_n\theta$, and either succeed or fail in the background theory B. We will use the notation $q_c\theta$ to refer to such instantiated queries.

The probabilistic model specifies a distribution over the random variables $\mathbf{p}$ and $\mathbf{q_c}$, where $\mathbf{q_c}$ represents whether the query corresponding to clause c succeeds or fails. The observed (boolean) value of $\mathbf{q_c}$ is $q_c\theta$. We now define

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

where the queries q_i correspond to the clauses C in the hypothesis $H = (C, \Phi)$ and Φ denotes the parameters of the probabilistic model H. Thus, a hypothesis H actually consists of the clauses C and the parameters Φ.

Now it becomes possible to state the naïve Bayes assumption $\mathbf{P}_\Phi(\mathbf{q_1}, ..., \mathbf{q_k}|\mathbf{p}) = \prod_i \mathbf{P}_\Phi(\mathbf{q_i}|\mathbf{p})$ and to apply it to our *covers* function:

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

At the same time, this equation specifies the parameters Φ of the probabilistic logical model H, i.e. the distributions $\mathbf{P}_\Phi(\mathbf{q_i}|\mathbf{p})$ and $\mathbf{P}_\Phi(\mathbf{p})$ (note that $\mathbf{P}_\Phi(\mathbf{q_1}, ..., \mathbf{q_k})$ can be computed by summing out).

Example 2 *Reconsider the mutagenicity example, and assume that the hypothesis is as sketched before. Then the queries q_1 and q_2 are*

$$\leftarrow \quad atom(X, A, c), atom(X, B, o), bond(X, A, B, 7)$$

and $\quad \leftarrow \quad atom(X, A, fl), bond(X, A, B, 2)$.

and the target predicate p is "muta(X)". Now assume that the probability distributions $\mathbf{P}_\Phi(\mathbf{q_i}|\mathbf{p})$ encoded in the model are

$$P_\Phi(\mathbf{p} = t) = 0.6$$
$$P_\Phi(\mathbf{q_1} = t|\mathbf{p} = t) = 0.7 \quad P_\Phi(\mathbf{q_1} = t|\mathbf{p} = f) = 0.4$$
$$P_\Phi(\mathbf{q_2} = t|\mathbf{p} = t) = 0.5 \quad P_\Phi(\mathbf{q_2} = t|\mathbf{p} = f) = 0.1$$

Summing out yields

$$P_\Phi(\mathbf{q_1} = t, \mathbf{q_2} = t) = 0.226 \quad P_\Phi(\mathbf{q_1} = t, \mathbf{q_2} = f) = 0.354$$
$$P_\Phi(\mathbf{q_1} = f, \mathbf{q_2} = t) = 0.114 \quad P_\Phi(\mathbf{q_1} = f, \mathbf{q_2} = f) = 0.306$$

where t (f) denotes $true$ ($false$). For the positively labeled example $\theta = \{X/mol_1\}$, we have that q_1 succeeds and q_2 fails: $p\theta = true$, $q_1\theta = true$, $q_2\theta = false$. Thus,

$$P(e \mid H, B) = \frac{P_\Phi(q_1\theta | p\theta) \cdot P_\Phi(q_2\theta | p\theta) \cdot P_\Phi(p\theta)}{P_\Phi(q_1\theta, q_2\theta)} = \frac{0.7 \cdot 0.5 \cdot 0.6}{0.354} \approx 0.59$$

The $score$ function

As scoring function, we employ the likelihood 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.). Therefore, we want to find the hypothesis H that maximizes

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

However, in contrast to the traditional naïve Bayes approach, the model consists of two components: a set of clauses and the corresponding probabilistic parameters.

The nFOIL Algorithm

Formally, the model space under consideration is

$$\mathcal{H} = \{H = (C, \Phi_C) | C \subseteq \mathcal{L}, \Phi_C \in \mathbb{R}^{2k+1}, |C| = k)$$

and the goal of learning is to identify

$$H^* = \arg\max_H l(E, H, B) = \arg\max_C \arg\max_{\Phi_C} l(E, (C, \Phi_C), B) .$$

Thus, there are two nested optimization tasks: Finding a globally optimal "structure" of the model involves the task of augmenting a given model structure C with its optimal parameters Φ_C.

Roughly speaking, the existing approaches pursued by Pompe and Kononenko (1995) and Davis *et al.* (2004) solve these nested optimization tasks one after the other: First, C is found (using a standard ILP system, and thus some different score) and fixed; second, the parameters for the fixed structure are optimized using a probabilistic score (usually, maximum likelihood). Therefore, it is unclear which global score is being maximized.

In the nFOIL system, we follow the more principled approach of guiding the search for the structure directly by the probabilistic objective function. We will show how this can be achieved by modifying the original search technique used in FOIL.

From FOIL to nFOIL

The main difference between FOIL and nFOIL is that nFOIL does not follow a separate-and-conquer approach. In FOIL, this is possible because the final model is taken as the disjunction of the clauses (every clause covering a certain subset of examples), which has two consequences:

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: $score(E, H \cup \{c'\}, B) = accuracy(c')$

These properties do not hold in nFOIL because every clause can affect the likelihood of all examples. Consequently, the nFOIL algorithm is obtained from FOIL by changing two components in Algorithm 1:

1. The set of examples is not changed: $update(E, H) = E$.
2. The score of a clause c' is the conditional likelihood of data assuming model $C \cup \{c'\}$ with optimal parameters:

$$score(E, H \cup \{c'\}, B) = \max_{\Phi_{C \cup \{c'\}}} l(E, (C \cup \{c'\}, \Phi_{C \cup \{c'\}}), B)$$

where C is the clause set of H.

We also have to modify the stopping criterion. The basic FOIL algorithm stops if all positive examples are covered. Instead, nFOIL stops adding new clauses when the improvement in score is below a certain threshold. In general, this simple criterion might lead to overfitting. Standard techniques to avoid overfitting such as rule post-pruning could be applied. In the experiments, however, our basic stopping criterion worked surprisingly well.

Parameter Estimation: An Approximation

To compute $score(E, H \cup \{c'\}, B)$, one needs to solve the "inner" optimization problem of finding the parameters Φ_C for a clause set C with bodies $\{q_1, ..., q_k\}$. From the naïve Bayes point of view, this amounts to finding the *maximum conditional likelihood* (MCL) parameters:

$$\begin{aligned}\Phi^*_C &= \arg\max_{\Phi_C} \prod_{\theta \in E} P_{\Phi_C}(p\theta | q_1\theta, \cdots, q_k\theta) \\ &= \arg\max_{\Phi_C} \prod_{\theta \in E} \frac{\prod_j P_{\Phi_C}(q_j\theta \mid p\theta) \cdot P_{\Phi_C}(p\theta)}{P_{\Phi_C}(q_1\theta, ..., q_k\theta)} \quad (1)\end{aligned}$$

The usual way of estimating parameters for naïve Bayes is

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

where $n(X)$ are the *counts*, i.e. the number of examples for which the query X succeeds. However, these are the *maximum likelihood* parameters, maximizing

$$\begin{aligned}&\textstyle\prod_{p\theta \in E} P_{\Phi_C}(p\theta, q_1\theta, ..., q_k\theta) = \\ &\qquad = \prod_{p\theta \in E} \prod_j P_{\Phi_C}(q_j\theta \mid p\theta) \cdot P_{\Phi_C}(p\theta) \quad (2)\end{aligned}$$

Could we use the likelihood as defined by Equation (2) as the score? The problem is that this term is dominated by the "feature likelihood" $P_{\Phi_C}(q_1\theta, ..., q_k\theta)$, which is maximized for constant features q_i (which are completely uninformative). In fact, in this case no feature could achieve a higher likelihood than the feature that always succeeds, and no refinements would ever be considered. In contrast, in Equation (1), the likelihood is corrected by the term $P_{\Phi_C}(q_1\theta, ..., q_k\theta)$; in this way informative features are selected. On the other hand, solving for the MCL parameters is computationally expensive.

A similar problem also arises in Bayesian Network structure learning (Grossman & Domingos 2004). Here, likelihood maximization leads to over-connected structures, a problem which is also solved by maximizing conditional likelihood. Because finding MCL parameters is computationally too expensive, Grossman and Domingos propose a "mixed" approach: using conditional likelihood as the score, but setting parameters to their ML values (seen as an approximation to their MCL values).

For nFOIL, we follow the same approach. Parameters are estimated to maximize the likelihood, i.e., Equation (2), while the conditional likelihood, see Equation (1), is retained as the score for selecting the features. The computational costs for evaluating a hypothesis in nFOIL are dominated by computing for each query the examples on which it succeeds. This involves basically the same computational steps as scoring in FOIL. FOIL, however, profits from its separate-and-conquer approach; the number of examples is reduced after each iteration. Thus, even though nFOIL needs to consider more examples when evaluating a hypothesis, nFOIL's complexity is very similar to FOIL's.

Finally, we note that unlike for ILP (and the propositionalization approaches relying on ILP systems), the extension of the nFOIL model and algorithm for multi-class problems is straightforward: replace the binary class variable by a multi-valued class variable.

Experiments

Our intention here is to investigate to which extent nFOIL is competitive with related approaches on typical ILP benchmark problems. More precisely, we will investigate:

(Q1) Is there a gain in predictive accuracy of nFOIL over its baseline, FOIL?

(Q2) If so, is the gain of an integrated approach (such as nFOIL) over its baseline larger than the gain of propositionalization approaches?

(Q3) Relational naïve Bayes approaches such as 1BC2 essentially follow a propositionalization approach, employing all features within the bias. Does nFOIL employ less features and perform well compared to these approaches?

(Q4) Is nFOIL competitive with advanced ILP approaches?

In the following section, we will describe the datasets and algorithms used to experimentally investigate **Q1**–**Q4**.

Datasets and Algorithms

In order to investigate **Q1**–**Q4**, we conduct experiments on three ILP benchmark datasets. On **Mutagenesis** (Srinivasan *et al.* 1996) the problem is to predict the mutagenicity of a set of compounds. In our experiments, we use the atom and bond structure information only. 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, Srinivasan, & Sternberg 1995), the aim is to compare four desirable properties of drugs against Alzheimer's disease. In each of the four subtasks, the aim is to predict whether a molecule is better or worse than another molecule with respect to the considered property: inhibit **amine** reuptake (686 examples), low **toxicity** (886 examples), high **acetyl** cholinesterase inhibition (1326 examples), and good **reversal** of scopolamine-induced memory deficiency (642 examples). For **Diterpene** (Džeroski *et al.* 1998), the task is to identify the skeleton of diterpenoid compounds, given their C-NMR-Spectra which include the multiplicities and the frequencies of the skeleton atoms. The dataset contains information on 1530 diterpenes with known structure. There is a total of 23 classes. We used the version with both relational and propositional information.

We investigate the following learners. **mFOIL** (Lavrač & Džeroski 1994) is a variant of FOIL employing beam search and different search heuristics. The beam size is set to $k = 5$, and the confidence parameter to prune clauses

Table 1: Cross-validated accuracy results on ILP benchmark data sets. For **Mutagenesis r.u.**, leave-one-out cross-validated accuracies are reported because of the small size of the data set. For all other domains, 10-fold cross-validated results are given. •/◦ indicates that nFOIL's mean is significantly higher/lower (paired sampled t-test, $p = 0.05$). For **1BC2**, we do not test significance because the results on **Mutagenesis** are taken from (Flach and Lachiche 2004).

Dataset	nFOIL	mFOIL	mFOIL+NB	Aleph	Aleph+NB	1BC2
Mutagenesis r.f.	78.3 ± 12.0	68.6 ± 8.7	68.6 ± 8.7	72.8 ± 11.7	72.8 ± 11.7	79.3
Mutagenesis r.u.	78.6 ± 41.5	78.6 ± 41.5	78.6 ± 41.5	88.1 ± 32.8	88.1 ± 32.8	73.8
Alzheimer amine	83.1 ± 6.5	70.4 ± 6.6•	69.9 ± 6.9•	70.1 ± 7.6•	70.1 ± 7.6•	70.6
Alzheimer toxic	90.0 ± 2.9	81.4 ± 4.3•	81.4 ± 4.3•	90.8 ± 4.9	90.8 ± 4.9	80.0
Alzheimer acetyl	78.3 ± 4.3	73.5 ± 2.4•	73.1 ± 2.6•	69.2 ± 4.1•	69.2 ± 4.1•	74.7
Alzheimer memory	66.7 ± 5.3	60.6 ± 7.8	60.6 ± 7.8	66.7 ± 5.2	66.7 ± 5.2	66.8
Average over Muta. + Alz.	**79.2**	**72.2**	**72.0**	**76.3**	**76.3**	**74.2**
Diterpene	84.2 ± 4.1	–	–	85.0 ± 3.6	85.0 ± 3.6	81.9
Overall Average	**79.9**	–	–	**77.5**	**77.5**	**75.3**

is set to 1 as this yields better results than the default setting. Our implementation of **nFOIL** also uses – as mFOIL – beam search with beam size $k = 5$. To consider the same space of clauses, we do not allow for negative literals in mFOIL and nFOIL. **Aleph** (Srinivasan 2004) is an advanced ILP system. For the two-class datasets **Mutagenesis** and **Alzheimer**, we apply Aleph's standard rule learner. To handle the multi-class problem **Diterpene**, we use Aleph's tree learner. In both cases, the standard settings are used. **1BC2** is a naïve Bayes classifier for structured data (Flach & Lachiche 2004). Finally, we consider propositionalization approaches, denoted by ILP+**NB**, along the lines of (Davis *et al.* 2004). They employ clauses learned by the ILP systems **mFOIL** and **ALEPH** as features of a **n**aïve **B**ayes.

Results

The cross-validated experimental results are summarized in Table 1. Comparing the results for **nFOIL** and **mFOIL**, the experiments clearly affirmatively answer **Q1**. On average, **nFOIL**'s gain in predictive accuracy is 7.0. In contrast, the propositionalization approaches **Aleph+NB** and **mFOIL+NB** show no gains in predictive performance over their corresponding baselines. This indicates that the answer to **Q2** is yes. Moreover, in 3 out of 6 cases, **nFOIL**'s mean is significantly higher than that of **mFOIL+NB**. A simple sign test $(5/0)$ shows that **mFOIL+NB**'s mean was never higher than that of **nFOIL**. A sign test between **nFOIL** and **Aleph** yields a draw while the average predictive accuracy (79.9/77.5) slightly favors **nFOIL**. Only **nFOIL** shows significantly higher means. This is a positive answer to question **Q4**. Question **Q3** seems to have an affirmative answer because both the average predictive accuracy (79.9/75.3) and a sign test (5/2) prefer **nFOIL** over **1BC2**.

The complexities of the learned models are hard to compare because of the different formats of the complexities. For **nFOIL** and **1BC2**, they are the number of probability values attached to clauses. More precisely, there are $\#$*classes* many probability values attached to each clause where $\#$*classes* is the number of classes. Additionally, we have to specify the prior distribution over the class variable. Thus, there are $\#\textit{classes} \cdot \#\textit{clauses} + \#\textit{classes} - 1$ many probability values. Because this is of the order of $\mathcal{O}(\#\textit{clauses})$, 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** selects fewer than 23 clauses on average. Indeed, **1BC2** actually does not select clauses but rather uses all clauses within a given language bias. This clearly shows that **Q3** can be answered affirmatively as well.

Furthermore, on average, **nFOIL** selected roughly as many clauses as **mFOIL** and **Aleph**. The number of clauses varied between 8 and 25. The problem when comparing the **nFOIL**/**1BC2** results with the **mFOIL** and **Aleph** results is that the latter do not attach probability values to clauses. Still, the model complexity – as argued above – linearly scales with the number of clauses for a fixed application domain, and, more importantly, the selected clauses give the expert some indication of which logical knowledge discriminates well between the classes. This supports an affirmative answer to **Q4**.

To summarize, the experiments show that **nFOIL** is competitive with state-of-the-art machine learning approaches.

Related Work

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

The first class of techniques starts by generating a set of first-order features (using either a kind of propositionalization approach, as in the 1BC system (Flach & Lachiche 2004), or by running a traditional ILP algorithm (Pompe & Kononenko 1995; Davis *et al.* 2004) and then using the generated clauses as attributes in a probabilistic model (such as naïve Bayes (Pompe & Kononenko 1995), or tree-augmented naïve Bayes or Bayesian networks (Davis *et al.* 2004)). In this class of techniques, the feature construction and the statistical learning steps are performed consecutively and independently of one another, whereas in nFOIL they are tightly integrated.

The second class of techniques ((Taskar, Segal, & Koller 2001), the 1BC2 system (Flach & Lachiche 2004), (Neville,

Jensen, & Gallagher 2003)) employs a relational or a higher-order logical probabilistic model, whose logical component (and hence its features) are fixed, and then learns the parameters of such a model using statistical learning techniques. The difference with nFOIL is that this class of techniques does not address structure learning or feature generation.

The third class of techniques (Popescul *et al.* 2003; Dehaspe 1997) indeed tightly integrates the inductive logic programming step with the statistical learning one. However, whereas nFOIL employs the simplest possible statistical model, i.e. naïve Bayes, these approaches employ much more advanced (and hence computationally much 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 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.

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 nFOIL in that nFOIL employs naïve Bayes on top of logic, whereas Craven and Slattery employ naïve Bayes as a predicate in the logical definitions.

Conclusions

We have introduced the nFOIL system. It combines the simplest approaches from ILP and probabilistic learning. Despite its simplicity, it was shown to be competitive with more advanced systems, such as Aleph, and to have advantages over baseline approaches (such as 1BC2 and mFOIL).

In further work, we want to investigate whether one can also integrate more advanced ILP systems (such as Aleph) with more advanced probabilistic models (such as tree-augmented naïve Bayes).

Acknowledgements The authors would like to thank the anonymous reviewers for valuable comments. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic Inductive Logic Programming II*.

References

Blockeel, H., and De Raedt, L. 1997. Lookahead and discretization in ILP. In *Proceedings of ILP-97*, 77–85.

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

Davis, J.; V. Santos Costa, I. O.; Page, D.; and Dutra, I. 2004. Using Bayesian Classifiers to Combine Rules. In *Working Notes of MRDM-04*.

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

De Raedt, L., and Kersting, K. 2004. Probabilistic Inductive Logic Programming. In *Proceedings of ALT-04*, 19–36.

Dehaspe, L. 1997. Maximum entropy modeling with clausal constraints. In *Proceedings of ILP-97*, 109–124.

Dietterich, T.; Getoor, L.; and Murphy, K., eds. 2004. *Working Notes of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields (SRL-04)*.

Džeroski, S.; Schulze-Kremer, S.; Heidtke, K.; Siems, K.; Wettschereck, D.; and Blockeel, H. 1998. Diterpene Structure Elucidation from ^{13}C NMR Spectra with Inductive Logic Programming. *Applied Artificial Intelligence* 12:363–383.

Flach, P., and Lachiche, N. 2004. Naive Bayesian classification of structured data. *Machine Learning* 57(3):233–269.

Getoor, L.; Friedman, N.; Koller, D.; and Pfeffer, A. 2001. Learning probabilistic relational models. In Džeroski, S., and Lavrač, N., eds., *Relational Data Mining*. Springer.

Grossman, D., and Domingos, P. 2004. Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood. In *Proceedings of ICML-04*, 361–368.

King, R.; Srinivasan, A.; and Sternberg, M. 1995. Relating chemical activity to structure: an examination of ILP successes. *New Gen. Comput.* 13(2,4):411–433.

Lavrač, N., and Džeroski, S. 1994. *Inductive Logic Programming*. Ellis Horwood.

Muggleton, S. 1990. Efficient induction of logic programs. In *Proceedings of ALT-90*, 368–381.

Muggleton, S. 1995. Inverse Entailment and Progol. *New Gen. Comp.* 13:245–286.

Muggleton, S. 1996. Stochastic logic programs. In *Advances in Inductive Logic Programming*. IOS Press.

Neville, J.; Jensen, D.; and Gallagher, B. 2003. Simple Estimators for Relational Bayesian Classifiers. In *Proceedings of ICDM-03*, 609–612.

Pompe, U., and Kononenko, I. 1995. Naive Bayesian classifier within ILP-R. In *Proceedings of ILP-95*, 417–436.

Popescul, A.; Ungar, L.; Lawrence, S.; and Pennock, D. 2003. Statistical Relational Learning for Document Mining. In *Proceedings of ICDM-03*, 275–282.

Quinlan, J. 1990. Learning logical definitions from relations. *Machine Learning* 239–266.

Srinivasan, A.; Muggleton, S.; King, R.; and Sternberg, M. 1996. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence* 85:277–299.

Srinivasan, A. 2004. The Aleph Manual. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_toc.html. Last update June 30, 2004.

Taskar, B.; Segal, E.; and Koller, D. 2001. Probabilistic Clustering in Relational Data. In *Proceedings of IJCAI-01*, 870–878.