Improving Candidate Quality of Probabilistic Logic Models

Joana Côrte-Real[1]
CRACS & INESC TEC and Faculty of Sciences, University of Porto
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
jcr@dcc.fc.up.pt
https://orcid.org/0000-0002-1085-3264

Anton Dries
KU Leuven, Department of Computer Science
Celestijnenlaan 200A bus 2402, 3001 Leuven, Belgium
anton.dries@cs.kuleuven.be
https://orcid.org/0000-0003-2944-2067

Inês Dutra
CINTESIS and Faculty of Sciences, University of Porto
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
ines@dcc.fc.up.pt
https://orcid.org/0000-0002-3578-7769

Ricardo Rocha
CRACS & INESC TEC and Faculty of Sciences, University of Porto
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
ricroc@dcc.fc.up.pt
https://orcid.org/0000-0003-4502-8835

Abstract

Many real-world phenomena exhibit both relational structure and uncertainty. Probabilistic Inductive Logic Programming (PILP) uses Inductive Logic Programming (ILP) extended with probabilistic facts to produce meaningful and interpretable models for real-world phenomena. This merge between First Order Logic (FOL) theories and uncertainty makes PILP a very adequate tool for knowledge representation and extraction. However, this flexibility is coupled with a problem (inherited from ILP) of exponential search space growth and so, often, only a subset of all possible models is explored due to limited resources. Furthermore, the probabilistic evaluation of FOL theories, coming from the underlying probabilistic logic language and its solver, is also computationally demanding. This work introduces a *prediction-based pruning strategy*, which can reduce the search space based on the probabilistic evaluation of models, and a *safe pruning criterion*, which guarantees that the optimal model is not pruned away, as well as two alternative more aggressive criteria that do not provide this guarantee. Experiments performed using three benchmarks from different areas show that prediction pruning is effective in (i) maintaining predictive accuracy for all criteria and experimental settings; (ii) reducing the execution time when using some of the more aggressive criteria, compared to using no pruning; and (iii) selecting better candidate models in limited resource settings, also when compared to using no pruning.

2012 ACM Subject Classification Computing methodologies → Probabilistic reasoning

Keywords and phrases Relational Machine Learning, Probabilistic Inductive Logic Programming, Search Space Pruning, Model Quality, Experiments

[1] Funded by the FCT grant SFRH/BD/52235/2013.

Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018).
Editors: Alessandro Dal Palu', Paul Tarau, Neda Saeedloei, and Paul Fodor; Article No. 6; pp. 6:1–6:14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/OASIcs.ICLP.2018.6

Funding Work partially funded by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF) as part of project NanoSTIMA (NORTE-01-0145-FEDER-000016) and through the operation POCI-01-0145-FEDER-007746 funded by COMPETE2020 and by national funds from FCT within CINTESIS, R&D Unit (reference UID/IC/4255/2013).

1 Introduction

The ability to take uncertainty into account when building a declarative model of a real-world phenomena can result in a closer representation of reality. The Probabilistic Logic Programming (PLP) paradigm addresses this issue by encoding knowledge as facts or rules, which are believed to be true to some degree or with a given frequency, instead of using crisp true or false statements. There are several Prolog-based probabilistic logic languages in the literature that can represent and manipulate uncertainty, such as SLP [15], ICL [17], Prism [20], BLP [12], CLP($\mathcal{BN}$) [19], MLN [18], ProbLog [13], among others. Please see the work by [8] for a recent survey of PLP.

Performing structure learning over PLP produces models which are understandable by humans whilst still taking uncertainty into account. Probabilistic Inductive Logic Programming (PILP) is a subset of Statistical Relational Learning (SRL) that uses a probabilistic First Order Logic (FOL) language to represent data and their induced models. PILP differs from traditional Inductive Logic Programming (ILP) in that facts and rules have success probabilities ranging between 0 and 1, as opposed to being either 0 or 1 (false or true, respectively). In this setting, there are no longer positive and negative examples, but only *target probabilities* for each example. The aim of a PILP model is to predict probability values which are as close as possible to the target probabilities of each example. PILP algorithms use (i) a set of Probabilistic Examples (PE), and (ii) logical information pertaining complex relations expressed as logic facts and rules, the Probabilistic Background Knowledge (or PBK), to find a FOL model that explains the PE. PILP focuses on structure learning – the logic rules compose a theory that models the structure of the PE w.r.t PBK – but parameter learning can also be incorporated by tuning the probabilistic output of the rules which are learned [7].

A number of PILP systems exist in the literature: ProbFOIL [9, 7], SLIPCOVER [1, 2], and SkILL [5, 4]. Additionally, there are other ILP-based structure learning methods such as CLP($\mathcal{BN}$) [19] and MLN [14]. One of the limitations of the available PILP systems is that they inherit the exponential search space from ILP, and must in addition evaluate the fitness of each candidate model by computing, for each example, the likelihood of that example given the model. This can be very time consuming, since the evaluation process must consider all possible worlds where the theory in the model may be true. For a small number of facts and rules in the PBK this is not a problem, but computation grows exponentially as the size of the PBK is increased [10].

To address this problem, this work introduces *prediction pruning*. Prediction pruning prunes the PILP search space based on previously evaluated theories by taking into account the logical operation (conjunction or disjunction) that will be performed next. Prediction pruning can be effective in reducing the execution time, compared to using no pruning. Additionally, the quality of the explored candidate models is improved when prediction pruning is used in conjunction with beam search. Unlike other pruning approaches, such as

beam search as used in [7, 2, 5], or estimation pruning as used in [4], prediction pruning can guarantee *safety* such that when the safe criterion is used the optimal model is never pruned away. This work thus also investigates three possible criteria for prediction pruning: a *safe criterion* and two other more aggressive pruning criteria. Experiments using three benchmarks and two PILP systems show that all three criteria are effective in maintaining (or increasing) predictive accuracy for all experimental settings. Furthermore, the more aggressive criteria reduce execution time compared to using no pruning, without loss of predictive accuracy. Finally, in limited resource settings, better candidate models are generated when compared to using no pruning.

This paper is organized as follows. Section 2 briefly introduces the main concepts of PILP. Next, Section 3 presents the proposed pruning strategy and the proposed pruning criteria. Section 4 evaluates the proposed approach and discusses the results. Finally, conclusions and perspectives of future work are put forward in Section 5.

2 Background

Traditional ILP generates sets of FOL rules (or theories) trying to describe a problem, given as a *target predicate*, in terms of the clauses contained in a given background knowledge. The theory's fitness to describe the problem is assessed according to a *loss function*. The aim of ILP is to find a theory that explains all given positive examples and does not explain any of the given negative examples, but in practice it is common to relax these criteria and allow for some noise (misclassified examples). It is also common to define a declarative *language bias* using mode declarations in order to specify which rules are valid within the search space.

PILP extends the ILP setting by introducing a Probabilistic Background Knowledge (or PBK), where FOL data descriptions can be annotated with a probability value ranging from 0 to 1, and by introducing a set of Probabilistic Examples (PE), no longer positive or negative, also with a value ranging between 0 and 1. Facts and rules in the PBK and PE can represent either statistical information or the degree of belief in a statement, using type I or type II probability structures, respectively [11]. Non-annotated data is assumed to have a probabilistic value of 1. Because PILP theories are still generated based on the logical information of the data, the ILP language bias translates directly to PILP. The process of generating theories also mimics ILP, since they are based on the logical clauses in the PBK. Good theories are the ones which most closely predict the values of the PE or rather that minimize the error between predictions and the PE values.

In this work, probabilities are annotated according to ProbLog's syntax, using *possible world semantics* [8]. In ProbLog, each fact $p_j :: c_j$ in the PBK represents an independent binary random variable, meaning that it can either be true with probability p_j or false with probability $1-p_j$. This means that each probabilistic fact introduces a probabilistic choice in the model. Each set of possible choices over all facts of the PBK represents a possible world ω_i, where ω_i^+ is the set of facts that are true in that particular world, and $\omega_i^- = \omega_i \setminus \omega_i^+$ is the set of facts that are false. Since these facts have a probabilistic value, a ProbLog program defining a probabilistic distribution over the possible worlds can be formalized as shown in Eq. 1.

$$P(\omega_i) = \prod_{c_j \in \omega_i^+} p_j \prod_{c_j \in \omega_i^-} (1 - p_j) \tag{1}$$

A ProbLog *query* q is said to be true in all worlds w^q where $w^q \models q$, and false in all other worlds. As such, the *success probability* of a query is given by the sum of the probabilities of

all worlds where it is found to be true, as denoted in Eq. 2.

$$P(q) = \sum_{\omega_i \models q} P(w_i) \tag{2}$$

Even though the prediction (success probability) of a rule changes according to the literals contained in its body, the probabilistic model generated from the PBK is not altered throughout the execution of the program. The search for the best model in PILP thus consists of finding the theory whose success probabilities (for all examples) have the best fitness w.r.t. the PE values (according to some loss function), given a PBK. This allows for defining standard scoring metrics such as *probabilistic accuracy (or PAcc)*, as introduced by De Raedt *et al.* in [9]. PAcc can also be represented in terms of the mean absolute error (MAE) between predictions and example values as used by Chen *et al.* in [3]. These two formulations are equivalent.

3 Prediction Pruning

The PILP search space can be split in two separate dimensions w.r.t. the operation that is being used to traverse it, i.e., there is a dimension for rules (or theories of length one), which uses the AND operation to generate new rules, and a dimension for theories (of length greater than one), which in turn uses the OR operation to generate new theories. Fully exploring the PILP search space is equivalent to evaluating each theory in the theory lattice in order to determine the best theory according to a given metric.

The theories used to explain examples in PILP are built from the literals that are present in the program's PBK. The rule (AND) search space is composed by all rules whose body contains one or more of those literals. Rules can be combined using logical conjunction to form longer, more *specific* rules. The theory (OR) search space can be defined in a similar way. Theories are formed by combining a set of distinct rules using logical disjunction. In the same way that literals are the building blocks of rules, rules are the building blocks of theories. Adding a rule to a theory makes it more *general*.

The procedure to explore the PILP search space can thus be done in two steps: (i) explore the AND search space, and (ii) explore the OR search space. An exhaustive search strategy would be very time-consuming leading to a scenario where good theories might never have a chance to be evaluated due to the complexity of the probabilistic evaluation. When resources are limited, it is thus preferable to focus on good candidate theories and avoid candidate theories which are below a threshold of quality to transition to the next iteration. Prediction pruning is thus applied over previously evaluated theories which are *determined* to be useless for further combination. Prediction pruning excludes theories whose predictions suggest that the theory is already too specific, for the AND operation, or too general, for the OR operation. Algorithm 1 presents this procedure.

Algorithm 1 starts by exploring the AND search space in a direction of increasing specificity. It starts out by generating rules containing only one literal (line 3) and then uses these rules to generate combinations for the next iteration (lines 5–8). In order to prevent rules which are determined to be too specific from being considered for combination in the next iteration, prediction pruning is applied according to a given $CriterionAND$ (procedure $AND\_pred\_pruning$ on line 7). Rules that are pruned by this criterion are still included in R_{all} but they are not further specialized in R_{new} (line 8). The combination process is repeated until it yields no new rules. The set of initial theories T_1 is then populated with all rules in R_{all} (line 9). Similarly to the AND search space, T_1 is used to generate new

Algorithm 1 $PILP\_algorithm(PBK, PE, CriterionAND, CriterionOR)$.

1: $T_{all} = \emptyset$
2: $R_{all} = \emptyset$
3: $R_1 = generate\_rules\_one\_literal(PBK, PE)$
4: $R_{new} = R_1$
5: **while** $R_{new} \neq \emptyset$ **do**
6: $\quad R_{all} = R_{all} \cup R_{new}$
7: $\quad R_{pru} =$ **AND_pred_pruning**$(R_{new}, CriterionAND)$
8: $\quad R_{new} = \{r_1 \wedge r_{pru} \mid (r_1, r_{pru}) \in R_1 \times R_{pru}\}$
9: $T_1 = R_{all}$
10: $T_{new} = T_1$
11: **while** $T_{new} \neq \emptyset$ **do**
12: $\quad T_{all} = T_{all} \cup T_{new}$
13: $\quad T_{pru} =$ **OR_pred_pruning**$(T_{new}, CriterionOR)$
14: $\quad T_{new} = \{t_1 \vee t_{pru} \mid (t_1, t_{pru}) \in T_1 \times T_{pru}\}$
15: **return** T_{all}

theories T_{new} through combination using logical disjunction (lines 11-14). This process is analogous to the exploration of the AND search space, except that the pruning criterion $CriterionOR$, used in procedure $OR\_pred\_pruning$ (line 13), is based on generality as opposed to specificity.

The decision on whether a candidate theory should be further explored is made based on the theory's individual prediction values for each example. Depending on which search space is being explored, the criterion to exclude theories will differ. When two rules r^a and r^b are combined using logical conjunction, a more *specific* rule $r^{a,b} = r^a \wedge r^b$ will result. This is due to the fact that more literals in the body of the rule must succeed simultaneously so that the rule can be verified.

In the probabilistic setting, a rule r is composed of a logical part $l(r)$ and a prediction value $p(r)$ ranging from 0 to 1. The prediction value of rule r for a given example i, $p_i(r)$ is equal to the sum of the probabilities $P(\omega_n)$ of each world ω_n in the program in which $\omega_n \models l_i(r)$ for that same example i. This means that for the more specific rule $r^{a,b}$ to be true, both r^a and r^b must be true simultaneously, i.e. only the worlds where both r^a and r^b are true can be considered. This is equivalent to the intersection of the set of worlds which entail $l(r^a)$ and $l(r^b)$, taking also into account the variable groundings for r^a and r^b. Therefore, the prediction value of a specific rule for an example i can be defined in terms of the prediction values of less specific rules which compose it.

$$p_i(r^{a,b}) = \sum_{\omega_n \models l_i(r^{a,b})} P(\omega_n) = \sum_{\substack{\omega_n \models l_i(r^a) \cap \\ \omega_n \models l_i(r^b)}} P(\omega_n) \tag{3}$$

From Eq. 3, it follows that, for an example i, the prediction value of a more specific rule $p_i(r^{a,b})$ will always be less than or equal to the prediction value of $p_i(r^a)$ and $p_i(r^b)$. Therefore, the prediction value of rule $p_i(r)$ will be monotonically decreasing with the application of the AND operation, since in each iteration the rules become more specific.

Having established this ordering allows prediction pruning to be applied over previously evaluated rules to determine whether they are useless for further combination, given some criterion. For a given example i, if the prediction value of a rule $p_i(r)$ is less than the example value e_i, then continuing to apply the AND operation can only result in distancing $p_i(r)$

Table 1 Expressions for the soft, hard and safe criteria.

Criterion	*Search Space*	
	AND	*OR*
Soft	$\sum_i \big(p_i(t) - e_i\big) < 0$	$\sum_i \big(p_i(t) - e_i\big) > 0$
Hard	$\exists i : p_i(t) < e_i$	$\exists i : p_i(t) > e_i$
Safe	$\forall i : p_i(t) < e_i$	$\forall i : p_i(t) > e_i$

from e_i further, since $p_i(r)$ can only decrease from the application of the AND operation. As such, prediction pruning excludes rules whose prediction values for all examples suggest that the theory is already too specific when compared to the example values. A similar argument can be made for the OR operation and the *generality* of theories.

To determine whether theories will be pruned away or not, several criteria are possible. This work proposes three criteria for deciding if a theory is too specify/general: a *soft criterion*, a *hard criterion* and a *safe criterion*. These three criteria take into account the predictions of a theory $p_i(t)$ for the given examples, as well as the example values e_i themselves. Table 1 presents the expressions for the pruning criteria when applied to the AND and OR search spaces. The soft pruning criterion takes into account the theory's predictions for every example, and only prunes the theory away if it is *overall* more specific (for the AND operation) or more general (for the OR operation) than the values of the examples. The hard pruning criterion prunes a theory away if, in *any* example, the theory made a prediction that was more specific (for the AND operation) or more general (for the OR operation) than the annotated value for that example. The soft criterion differs from the hard criterion in that it takes into account the *aggregate* value of all examples, whilst the hard pruning criterion can discard theories based on one example value only. On the other hand, the safe pruning criterion excludes theories only when *all* of their predictions are found to be too specific (for the AND operation) or too general (for the OR operation), and no prediction can be improved by continuing with the search in that search space. Therefore, it is *safe* to prune away these candidate theories, since they can never perform better with more specialisation/generalisation, respectively.

Figure 1 illustrates these concepts for a PILP setting with three examples and three theories. For each example i, the example value e_i (squares in black) and three predictions of theories $p_i(t^1)$, $p_i(t^2)$ and $p_i(t^3)$ are plotted. The ground truth model would predict exactly e_i for every example. If a prediction value $p_i(t)$ is plotted *below* the example value e_i, then that theory is too specific for that example. Conversely, if $p_i(t)$ is plotted *above* e_i, the theory is more general for that example.

In Fig. 1, for the AND operation, the safe pruning criterion would prune away theory t^1 because, for every example, its prediction values are lower than the example values. The soft pruning criterion would prune away theories t^1 and t^2 because their prediction values are overall lower than the example values. Finally, the hard pruning criterion would prune away all theories. For example, theory t^3 is pruned away because its prediction for $e = 1$ is lower than the example value. An analogous reasoning can be made for the OR operation and higher prediction values. In summary, the theories pruned away by the safe criterion are a subset of the theories pruned away by the soft criterion, and similarly the theories pruned away by the soft criterion are a subset of those pruned away by the hard criterion.

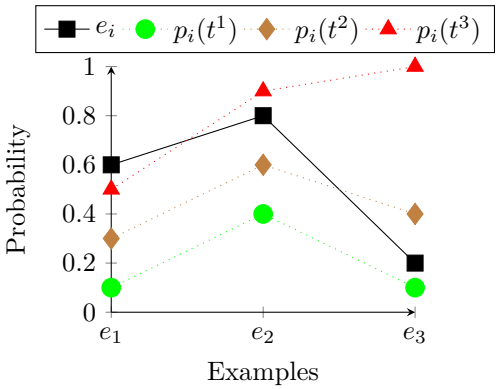


Figure 1 PILP setting with three examples and three theories. For each example, the example values (squares in black) and three predictions of theories (green circles for $p_i(t^1)$, brown diamonds for $p_i(t^2)$ and red triangles for $p_i(t^3)$) are plotted.

4 Experiments

The experiments presented in this section are aimed at answering the following three questions: (i) how much does prediction pruning reduce the exhaustive PILP search space? (ii) can prediction pruning maintain predictive quality of models? (iii) how does prediction pruning impact the quality of the candidate models explored in a limited resource setting?

Prediction pruning was implemented and evaluated in two state-of-the-art PILP systems: SkILL [5] and ProbFOIL+ [7]. SkILL runs on top of the Yap Prolog system [6], uses TopLog [16] as the basis for rule generation and the ProbLog Yap library as its probabilistic inference engine. The experiments using the SkILL system were run on a machine containing 4 AMD Opteron 6300 processors with 16 cores each and a total of 250GB of RAM. ProbFOIL+ is based on Python and it uses the Yap Prolog system for logical inference of theories. In these experiments, ProbFOIL+ uses only the examples provided in the training data (without generation of additional negative examples as used in the original paper) and it uses negated literals in the theories. The experiments using ProbFOIL+ were run on a machine containing an Intel Core i7 processor with 4 cores and a total of 16GB of RAM. All experiments use five-fold stratified cross validation and results presented are the average values for all folds. The evaluation was performed using three different datasets: **metabolism**, **athletes** and **breast cancer**.

The metabolism dataset consists of an adaptation of the dataset originally from the 2001 KDD Cup Challenge[2]. It is composed of 230 examples (half positive and half negative) and approximately 7000 BK facts. To obtain probabilistic facts for the PBK, the predicate `interaction(gene1,gene2,type,strength)` was adapted from the original metabolism dataset. The fourth argument of this predicate indicates the strength of the interaction between a pair of genes. This fact was converted to the probabilistic fact `p_strength::interaction(gene1,gene2,type)`, where `p_strength` was calculated from strength interactions as follows:

$$p\_strength = \frac{strength - min_{strength}}{max_{strength} - min_{strength}}$$

[2] `http://www.cs.wisc.edu/~dpage/kddcup2001`

This resulted in about 3200 probabilistic facts in the PBK. 5 folds were generated from this dataset, and each one of them is composed of 46 test examples selected randomly from the main dataset (but keeping the same positive/negative ratio) and, for each fold, the 184 remaining examples are used for training.

The athletes dataset consists of a subset of facts regarding athletes and the sports they play collected by the never-ending language learner NELL[3]. NELL iteratively reads the web, gathering knowledge, and for each fact that it comes across it assigns a weight that can be used as a probability. As NELL iterates, the weights of the facts in its database are updated, and the dataset used for this experiment contains the facts and weights from iteration 850. The dataset is composed of 720 probabilistic examples of athletes that play for a team, and 4294 probabilistic facts in the PBK pertaining to the origin of the player, his/her gender, the city where a team plays, and so on. 5 folds were generated from this dataset, and each one of them is composed of 144 test examples selected randomly from the main dataset and the 576 remaining examples are used for training. Because in this case examples do not clearly belong to one of two classes, the test examples were randomly selected from the dataset without taking their expected value into account.

The breast cancer dataset contains data from 130 biopsies dating from January 2006 to December 2011, which were prospectively given a non-definitive diagnosis at radiologic-histologic correlation conferences. Twenty-one cases were determined to be malignant after surgery, and the remaining 109 proved to be benign. The probabilities assigned to the examples represent the chance of malignancy for each patient. A high probability indicates the team of physicians thinks the case is most likely malignant, and conversely a low probability indicates the case is most likely benign. Five folds were generated from this dataset, and each one of them is composed of 26 test examples selected randomly from the main dataset (but keeping the same positive/negative ratio) and the 104 remaining examples are used for training.

4.1 Probabilistic Accuracy and Search Space Reduction

Baseline. Because exploring the search space exhaustively is computationally taxing, the quality of candidate theories was assessed in a limited resource setting. Resources can be limited in two ways: either a timeout is imposed or a maximum number of evaluations is defined, which corresponds to using beam search (or the fitness pruning setting in the case of the SkILL system). To this effect, the impact of prediction pruning was assessed by comparing the AND and OR search spaces that are evaluated without pruning with those which are evaluated in a pruning setting, given the same limitation of resources. In these experiments, the default fitness pruning / beam search settings of both systems are used (that is, for SkILL, primary and secondary population sizes of 25/20 for both AND and OR space, and for ProbFOIL+, a beam size of 5 for the AND space and greedy search in the OR space, as ProbFOIL+ only supports greedy search there).

Prediction Pruning. The use of prediction pruning enables PILP systems to focus their (limited) resources on more promising candidates, when traversing the search space. Table 2 presents the results of applying prediction pruning in the AND search space in combination with fitness pruning / beam search. It shows the execution time (in seconds), the number of theories evaluated probabilistically and the probabilistic accuracy of the best theory found for different pruning criteria (Safe, Soft and Hard), using the SkILL and ProbFOIL+ systems. Please note that execution times between systems are not comparable.

[3] `http://rtw.ml.cmu.edu`

Table 2 Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy for datasets metabolism, athletes and breast cancer using the SkILL and ProbFOIL+ systems with prediction pruning for the AND search space. Standard deviation is presented in brackets. Execution times between systems are not comparable.

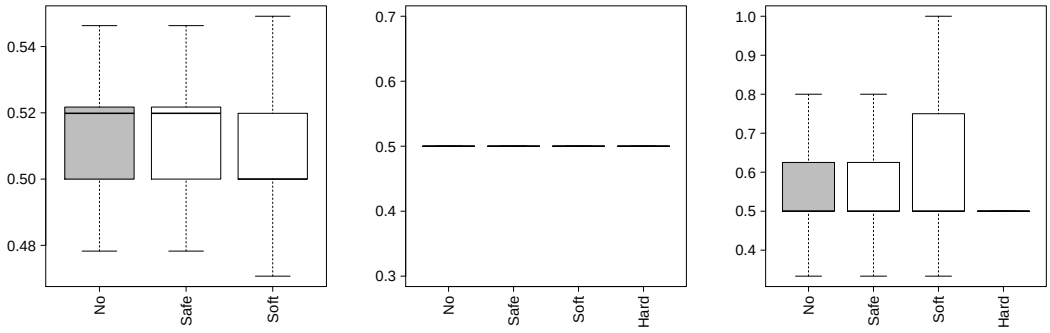
(a) SkILL.

	Baseline	Safe	Soft	Hard
	Execution Time (s)			
metabolism	3353 (204)	2286 (185)	3216 (472)	1791 (37)
athletes	4610 (79)	4230 (582)	2322 (164)	2358 (73)
breast cancer	1449 (63)	616 (50)	636 (26)	353 (42)
	No. Evaluations			
metabolism	2151 (44)	2150 (44)	3234 (90)	2103 (37)
athletes	1852 (25)	1896 (18)	994 (3)	994 (3)
breast cancer	1235 (68)	1234 (67)	1306 (43)	941 (70)
	Probabilistic Accuracy			
metabolism	0.67 (0.05)	0.67 (0.05)	0.67 (0.05)	0.67 (0.05)
athletes	0.95 (0.01)	0.95 (0.01)	0.95 (0.01)	0.95 (0.01)
breast cancer	0.86 (0.04)	0.86 (0.04)	0.84 (0.08)	0.86 (0.03)

(b) ProbFOIL+.

	Baseline	Safe	Soft	Hard
	Execution Time (s)			
metabolism	2008 (2016)	1999 (2019)	752 (215)	464 (71)
athletes	57 (5)	57 (5)	55 (4)	14 (0)
breast cancer	3890 (339)	3828 (302)	8093 (2101)	725 (38)
	No. Evaluations			
metabolism	3734 (2328)	4549 (3734)	4518 (1493)	2452 (492)
athletes	201 (43)	201 (43)	171 (21)	0 (0)
breast cancer	24290 (851)	24267 (828)	26495 (3542)	3532 (231)
	Probabilistic Accuracy			
metabolism	0.51 (0.04)	0.51 (0.03)	0.63 (0.11)	0.58 (0.07)
athletes	0.80 (0.01)	0.80 (0.01)	0.80 (0.01)	0.80 (0.01)
breast cancer	0.85 (0.01)	0.85 (0.01)	0.85 (0.03)	0.87 (0.01)

Probabilistic Accuracy. Prediction pruning results in Table 2 show that applying the Soft or Hard strategies leads to clear improvements in probabilistic accuracy for ProbFOIL+ and does not lead to degradation in SkILL. The effect of prediction pruning is more evident for ProbFOIL+ because it selects fewer candidates in each iteration, when compared to the SkILL's primary and secondary populations. It is therefore more important that bad candidates are pruned such that the limited beam is filled with better candidates. The prediction pruning strategy is thus particularly useful when traversing the search space with a narrow beam, so that the candidates selected to populate it are of greater predictive value when compared to using no prediction pruning. Safe pruning has no effect on these datasets because its pruning power is too limited.



(a) metabolism. **(b)** athletes. **(c)** breast cancer.

Figure 2 Distribution of theories' AUCs for the AND search space for datasets metabolism, athletes and breast cancer using different prediction pruning settings in the SkILL system.

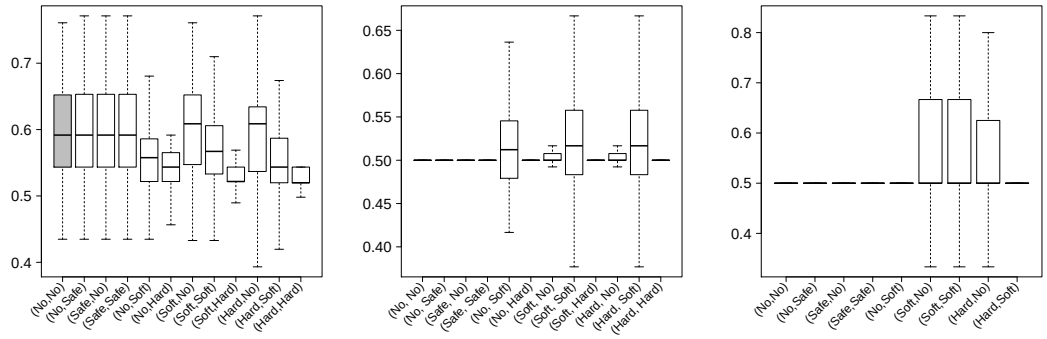
Search Space Reduction. Table 2 also shows that applying prediction pruning does not necessarily reduce the search space. It can actually increase the number of rules evaluated during the execution, and even the execution time in some cases. This happens because prediction pruning provides a type of lookahead, that is, it makes an assessment of the predictive power of a rule in future iterations. When no prediction pruning is used, the algorithms have a strong bias toward rules that show good performance early on and the best rule (in the limited search space) is found after a few iterations. Prediction pruning counteracts this bias, and also allows candidates that only reach their full predictive accuracy after a higher number of iterations to be explored. However, since the algorithm may take more iterations, this can lead to more evaluations and longer rules that are harder to evaluate.

4.2 Search Space Quality

Each theory in the PILP search space can be thought of as a predictor, and for this reason its predictive quality can be assessed using the area under the ROC curve (AUC). Since prediction pruning removes theories from the search space based upon the operation that is being performed (AND or OR), the distribution of the remaining candidate theories can change (there may be cases where no candidate theories are left for the next iteration). As such, comparing the two search spaces using the AUCs of the theories they contain shows how the predictive quality of their candidates compares.

For the SkILL experiments, the AUC of all rules containing more than one literal (AND search space) and all theories (OR search space) was calculated. The AUC of rules composed of only one literal was not considered because prediction pruning has no effect on these rules, which must always be evaluated. Analysing the distribution of the AUC values is relevant because if the upper quartiles of the distribution are improved, this shows that there are better candidate members selected to be explored given limited resources. Lower quartiles will naturally be discarded by the PILP algorithm's metric to select the best final theory. The distribution of these values for each setting and search space are presented in Figures 2 and 3 for the AND and OR search spaces, respectively. Each box depicts percentiles 0 and 100 (the lower and upper whiskers, respectively), percentiles 25 and 75 (lower and upper box boundaries, respectively), and the percentile 50 (median) using a bold line.

In Figs. 2–3, the higher the AUC value (y-axis), the greater the predictive power of the theory. Each boxplot corresponds to a setting. In Fig. 2 (AND search space only), the first boxplot corresponds to the rules generated using no prediction pruning, the second



(a) metabolism. **(b)** athletes. **(c)** breast cancer.

Figure 3 Distribution of theories' AUCs for the OR search space, for datasets metabolism, athletes and breast cancer using different prediction pruning settings in the SkILL system.

Table 3 Number of significant differences (left) for the number of tested folds (right) in the AND and OR AUC distributions for datasets metabolism, athletes and breast cancer using different prediction pruning settings in the SkILL system.

Setting	metabolism		athletes		breast cancer	
(AND,OR)	AND	OR	AND	OR	AND	OR
(No, No)	0/4	0/5	0/5	0/5	0/5	0/5
(No, Safe)		0/5		2/5		0/5
(Safe, No)	0/4	0/5	0/5	2/5	0/5	0/5
(Safe, Safe)		0/5		2/5		0/5
(No, Soft)	0/4	4/4	0/5	4/5	0/5	0/5
(No, Hard)		4/4		4/5		–
(Soft, No)		2/5		3/5		2/5
(Soft, Soft)	4/4	5/5	0/5	3/5	3/5	3/5
(Soft, Hard)		5/5		4/5		–
(Hard, No)		5/5		3/5		5/5
(Hard, Soft)	–	3/4	0/5	4/5	1/4	4/4
(Hard, Hard)		1/1		4/5		–

boxplot to the rules generated using safe prediction pruning, and so on. In Fig. 3 (AND and OR search spaces), the pruning settings are reported as a tuple where the first value is the AND prediction pruning option and the second is the OR prediction pruning option. For example, the tuple (Soft,Hard) stands for soft AND prediction pruning and hard OR prediction pruning, whilst the tuple (No,Safe) stands for no AND pruning and safe OR prediction pruning.

For the AUC distributions, statistical significance is also calculated (using non-paired two-tailed t-test) by comparing the distribution of AUCs fold to fold (e.g. fold 1 using soft OR prediction pruning against fold 1 without pruning). Table 3 reports the number of folds where the results were statistically significant for both the AND and the OR search spaces. In some cases, some folds do not produce an AND or OR search space because all theories are pruned away, and this is the cause for not always reporting five folds in comparison.

In Fig. 3, it is visible that prediction pruning can improve the general quality of the evaluated theories, particularly in the case of the athletes and breast cancer datasets. In the breast cancer dataset, the two upper quartiles of the AUC distribution are clearly improved

in three settings. This trend is also clear in the athletes dataset, where again prediction pruning significantly increases the predictive quality of the evaluated theories in three cases (and slightly in two other settings). On the metabolism dataset, the improvements due to prediction pruning are not as evident, but it is noteworthy that there is in fact a slight increase in the maximum AUC value for the case of hard AND pruning and no OR pruning, as well as in all safe pruning settings. The boxplots with range zero indicate that in those settings the candidates that populate the beam do not have any predictive power in the test set. However, this does not imply a loss in predictive accuracy of the optimal model since rules of only one literal are not included in these boxplots because they are not affected by prediction pruning.

Regarding the quality of the AND search space (Fig. 2), it is only significantly improved in the breast cancer dataset, using soft prediction pruning. However, the candidate rules that are selected for the AND search space impact the OR search space, since candidate theories will be selected from the rules that were previously explored in the AND search space. As such, even though the AND search space only shows direct impact from using prediction pruning in the breast cancer dataset, it indirectly impacts the candidate theories available for the OR search space in all datasets. This is particularly relevant for the athletes dataset, where the quality of the OR search space is affected by soft and hard AND pruning. For instance, setting (Soft, Soft) performs significantly better when compared to setting (No, Soft), and setting (Hard, No)'s 50 and 100 percentiles are higher than its counterpart setting (No, No). This effect is also visible in the breast cancer dataset, where the settings using soft or hard AND prediction pruning present the greatest improvement. In most cases where the quality of the OR search space increased, AND prediction pruning had previously been applied to the AND search space.

Table 3 shows that the safe pruning criterion causes no significant difference in candidate theory predictive quality, both for the AND and the OR operation (lines 2–4). This is due to the fact that the safe pruning criterion is the least aggressive criterion and therefore the proportion of candidates that are pruned in this setting is limited. On the other hand, both soft and hard pruning criteria cause a significant difference in the AUC distribution of candidates, in particular for the OR operation, where most folds present a significant difference (lines 5–12 and columns 2, 4 and 6 in Table 3). However, for the AND operation, aggressive criteria do not cause such a significant difference in the distribution, in particular for the athletes dataset. This happens because the predictive power of rules in this dataset is similar among candidates, and so even though different rules can be selected, this is not reflected in the distribution of AUC values. In cases where aggressive pruning causes the search space to be empty for all folds, there is no boxplot in Figs. 2–3, and no value reported in Table 3.

Prediction pruning thus impacts the quality of the search space positively, allowing for limited resources to be targeted towards better candidate theories. Furthermore, even though in some cases the quality of the search space decreases (for instance the quality of the AND search space using hard prediction pruning in the breast cancer dataset), the accuracy of the best final theory found never decreases significantly, thus showing that prediction pruning can be applied to better select candidate theories without risk of impacting the final test accuracy.

5 Conclusion

This work proposes a novel prediction pruning methodology whose aim is to improve the quality of the explored candidate models in a PILP search space. Unlike previously proposed pruning approaches, such as beam search and estimation pruning, prediction pruning focuses on improving the quality of the search space. In doing so, it can direct the search towards more promising candidates which can lead to a reduction in execution time or an increase in predictive accuracy.

This work also introduces three pruning criteria, with increasing pruning power, which can be used to decide which models should be pruned away during the prediction pruning stage in the PILP algorithm. All pruning criteria are based on the probabilistic information of candidate models and depend on which operation is being performed in the PILP algorithm: logic conjunction (AND search space) or disjunction (OR search space). The safe pruning criterion guarantees the safeness of the prediction pruning strategy, meaning that the optimal model is never pruned away during the search, but experiments show that this criterion is not very successful in pruning the search space significantly. The soft and hard pruning criteria, however, do exhibit pruning power while not suffering from a reduction in predictive performance.

Results also show that prediction pruning maintains the predictive quality of the generated models. Prediction pruning impacts the distribution of the predictive quality of theories and the use of prediction pruning can shift the maximum value and upper quartile of the distribution upwards, thus indicating improved candidate theory quality. Deeper analysis of the AUC of theories shows that all three criteria improve the quality of the OR search space. AND prediction pruning, while not presenting a significant difference in all datasets, can influence the OR search space quality, and so using prediction pruning for both operations can increase the quality of the candidate theories while not sacrificing the final predictive accuracy.

An interesting direction for future work is to study how to automatically adjust the pruning criterion based on data characteristics of the dataset. Further work also includes developing a search space traversal strategy combining several pruning strategies and, in particular, study how prediction pruning interacts with beam search and estimation pruning.

References

1 E. Bellodi and F. Riguzzi. Learning the structure of probabilistic logic programs. In *Inductive Logic Programming*, pages 61–75. Springer, 2012.

2 E. Bellodi and F. Riguzzi. Structure learning of probabilistic logic programs by searching the clause space. *Theory and Practice of Logic Programming*, 15(02):169–212, 2015.

3 J. Chen, S. Muggleton, and J. Santos. Learning Probabilistic Logic Models from Probabilistic Examples. *Machine Learning*, 73(1):55–85, October 2008. doi:10.1007/s10994-008-5076-4.

4 J. Côrte-Real, I. Dutra, and R. Rocha. Estimation-Based Search Space Traversal in PILP Environments. In A. Russo and J. Cussens, editors, *Proceedings of the 26th International Conference on Inductive Logic Programming (ILP 2016)*, LNAI, pages –, London, UK, September 2016. Springer. Published in 2017.

5 J. Côrte-Real, T. Mantadelis, I. Dutra, R. Rocha, and E. Burnside. SkILL - a Stochastic Inductive Logic Learner. In *International Conference on Machine Learning and Applications*, pages –, Miami, Florida, USA, December 2015.

6 V. Santos Costa, R. Rocha, and L. Damas. The YAP Prolog System. *Journal of Theory and Practice of Logic Programming*, 12(1 & 2):5–34, 2012.

7 L. De Raedt, A. Dries, I. Thon, G. Van den Broeck, and M. Verbeke. Inducing Probabilistic Relational Rules from Probabilistic Examples. In *International Joint Conference on Artificial Intelligence*, pages 1835–1843. AAAI Press, 2015.

8 L. De Raedt and A. Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015. doi:10.1007/s10994-015-5494-z.

9 L. De Raedt and I. Thon. Probabilistic Rule Learning. In *Inductive Logic Programming*, pages 47–58. Springer, 2011.

10 Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

11 J. Halpern. An Analysis of First-Order Logics of Probability. *Artificial intelligence*, 46(3):311–350, 1990.

12 K. Kersting, L. De Raedt, and S. Kramer. Interpreting Bayesian Logic Programs. In *AAAI Workshop on Learning Statistical Models from Relational Data*, pages 29–35, 2000.

13 A. Kimmig, B. Demoen, L. De Raedt, V. Santos Costa, and R. Rocha. On the Implementation of the Probabilistic Logic Programming Language ProbLog. *Theory and Practice of Logic Programming*, 11(2 & 3):235–262, 2011.

14 S. Kok and P. Domingos. Learning the Structure of Markov Logic Networks. In *International Conference on Machine learning*, pages 441–448. ACM, 2005.

15 S. Muggleton. Stochastic Logic Programs. *Advances in inductive logic programming*, 32:254–264, 1996.

16 S. Muggleton, J. Santos, C. Almeida, and A. Tamaddoni-Nezhad. TopLog: ILP Using a Logic Program Declarative Bias. In *International Conference on Logic Programming*, pages 687–692. Springer, 2008.

17 D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial intelligence*, 94(1):7–56, 1997.

18 M. Richardson and P. Domingos. Markov Logic Networks. *Machine learning*, 62(1-2):107–136, 2006.

19 V. Santos Costa, D. Page, M. Qazi, and J. Cussens. CLP(BN): Constraint Logic Programming for Probabilistic Knowledge. In *Conference on Uncertainty in Artificial Intelligence*, pages 517–524, 2002.

20 T. Sato and Y. Kameya. PRISM: A language for symbolic-statistical modeling. In *International Joint Conference on Artificial Intelligence*, volume 97, pages 1330–1339. Morgan Kaufmann, 1997. URL: http://ijcai.org/Proceedings/97-2/Papers/078.pdf.