Estimation-Based Search Space Traversal in PILP Environments

Joana Côrte-Real, Inês Dutra, and Ricardo Rocha

CRACS & INESC TEC
Faculty of Sciences, University of Porto
Rua do Campo Alegre, 1021/1055,
4169-007 Porto, Portugal
{jcr,ines,ricroc}@dcc.fc.up.pt

Abstract. Probabilistic Inductive Logic Programming (PILP) systems extend ILP by allowing the world to be represented using probabilistic facts and rules, and by learning probabilistic theories that can be used to make predictions. However, such systems can be inefficient both due to the large search space inherited from the ILP algorithm and to the probabilistic evaluation needed whenever a new candidate theory is generated. To address the latter issue, this work introduces probability estimators aimed at improving the efficiency of PILP systems. An estimator can avoid the computational cost of probabilistic theory evaluation by providing an estimate of the value of the combination of two subtheories. Experiments are performed on three real-world datasets of different areas (biology, web-based and medical) and show that, by reducing the number of theories to be evaluated, the estimators can significantly shorten the execution time without losing accuracy.

1 Introduction

Probabilistic Inductive Logic Programming (PILP) [4] is an extension of the ILP paradigm that can represent knowledge using probabilistic facts and rules and which learns, as a result, probabilistic theories that can be used for prediction. Introducing probabilistic information in ILP to create PILP can be used to (i) create better logical models that can take uncertainty into account; (ii) implicitly reduce the theory search space by transforming numerical arguments in annotated probabilistic data; (iii) compress data by representing it as aggregates; or (iv) add knowledge from the literature in the form of probabilistic information. PILP is a Statistical Relational Learning (SRL) [4] technique and in this setting, both parameter and structure learning are possible; however, it is more common for SRL techniques to learn parameters, and only few SRL methods can learn structure, or both. SRL is particularly relevant to produce and manipulate structured representations of data, and its aim is precisely to capture the logic relations that lie beyond the low-level features and reason about them. Some well-known examples of other SRL methods include Markov

Logic Networks [7], Probabilistic Relational Models [5], and Constraint Logic Programming for Probabilistic Knowledge [8], among others.

PILP suffers from the same search space traversal efficiency issues as ILP because similar algorithms are used to generate the logical part of the theories. Additionally, PILP adds a level of complexity because every new theory generated needs to be probabilistically evaluated in order to be considered. This work presents a strategy aimed at improving the performance of PILP systems through the use of *estimators* that can prune the universe of candidate theories and, thus, minimize the search space. These estimators were integrated in the SkILL system [2], but the concepts are general to any PILP engine.

SkILL is a stochastic inductive logic learner which can generate First-Order Logic (FOL) theories based on a database of probabilistic data. These theories are expressed as Horn clauses (a subset of FOL) and so they can be used to extract relational non-trivial knowledge about the dataset where they are inferred from. SkILL differs from other PILP systems such as ProbFOIL+ [3] or SLIPCOVER [1] because it introduces an algorithm of polynomially bound complexity on user-defined parameters, as well as a number of efficient pruning strategies that can reduce execution time while maintaining prediction quality.

To the best of the authors' knowledge, the notion of an estimator is a novel feature in PILP systems. In this work, five estimators that can be incorporated in the estimation pruning strategy are proposed, namely *minimum*, *maximum*, *center*, *independence* and *exclusion*. To validate this estimation-based search space traversal approach, a thorough experimental analysis of the impact that each estimator has on the execution time and theory quality is presented. Experiments are performed in three probabilistic datasets, and the models are validated using cross-validation techniques. Results show that estimators can significantly prune the search space, and thus, reduce execution time, while maintaining the same accuracy when compared with using no estimation pruning.

2 Background

ILP is a machine learning technique that can learn first order models (theories) given: (i) sets of positive and negative examples; (ii) a first order language; and (iii) a set of constraints on the possible generated theories. ILP finds a first order model that can ideally explain all positive examples and none of the negative examples. PILP extends the ILP setting by introducing Probabilistic Background Knowledge (PBK), where FOL data descriptions can be annotated with a probability value ranging from 0 to 1, and Probabilistic Examples (PE), no longer positive or negative, also with a value ranging between 0 and 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, and so the search space of a problem in PILP has similar efficiency issues to those ILP encounters. Furthermore, PILP adds an extra level of complexity due to the probabilistic evaluation of theories w.r.t. the examples.

Table 1. Summary of differences between ILP and PILP.

	Examples	Background Knowledge (Horn clauses)	Classifier (theory)
ILP	target(e_pos) target(e_neg)	fact1(e_pos,propA) fact2(e_pos,propB) fact1(e_neg,propC) fact2(e_neg,propD) def_cl1(X,Y):- fact1(E,X),fact2(E,Y) def_cl2(Y):- fact2(E,Y),def_cl1(X,Y)	target(E):- fact2(E,V),def_cl1(V,V) ↓ *prediction* ∈ {TRUE,FALSE}
PILP	p_e::target(e)	p_{f1A}::fact1(e,propA) p_{f2B}::fact2(e,propB) p_{f1C}::fact1(e,propC) p_{f2D}::fact2(e,propD) p_{c1}::def_cl1(X,Y):- fact1(E,X),fact2(E,Y) p_{c2}::def_cl2(Y):- fact2(E,Y),def_cl1(X,Y)	target(E):- fact2(E,V),def_cl1(V,V) ↓ *prediction* ∈ [0, 1]

Table 1 summarizes the main syntactic difference between ILP and PILP. In both settings, the background is composed of (Horn) clauses, which can be either *facts* (e.g. `fact1(e_pos,propA)`) or *definite clauses* (e.g. `def_cl1(X,Y):- fact1(E,X),fact2(E,Y)`). Definite clauses are composed of a *head* (e.g. `def_cl1 (X,Y)`) and a *body* (e.g. `fact1(E,X),fact2(E,Y)`), and the body represents the explanation for the head (read 'if `fact1(E,X)` and `fact1(E,Y)` then `def_cl1(X ,Y)`'). Facts and definite clauses' heads (`fact1`, `fact2`, `def_cl1` and `def_cl2` in Table 1) are the *literals* or *building blocks* that ILP and PILP use to build rules such as `target(E):- fact2(E,V),def_cl1(V,V)`, where E represents the examples to be explained by the theory named `target(E)`.

The probabilities in the PILP setting used in this work are annotated according to the ProbLog syntax [6]. Each clause $p_j :: c_j$ in the PBK represents an independent binary random variable in ProbLog, meaning that it can either be true with probability p_j or false with probability $1 - p_j$. Each set of possible choices over all clauses of the PBK represents a *possible world* ω_i, where ω_i^+ is the set of clauses that are true in that particular world, and $\omega_i^- = \omega_i \setminus \omega_i^+$ is the set of clauses that are false. Since these clauses have a probabilistic value, a ProbLog program defining a probabilistic distribution over the possible worlds can be formalized as shown in Equation 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 Equation 2.

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

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

One important difference between ILP and PILP lies in the assessment of the fitness of theories – in PILP the *loss function* must be able to evaluate probabilistic inputs. As such, the aim of PILP systems is to find theories which most closely predict the value of the examples (also ranging between 0 and 1), or rather that minimize the error between predictions and the examples' values.

Theories can be formed either by a single rule (clause), or by a set of disjunctive rules. The length of a theory is equal to the number of rules it contains.

Theories can be combined using either the AND or the OR operation, which correspond to the logical conjunction and disjunction of the rules in the theories, respectively. In the case of the AND operation, only single rules (theories of length one) can be combined, and the result is another theory of length one (e.g. combining theories `t1(X):- q(X)` and `t2(X):- r(X,Y)` using the AND operation would result in theory `t(X):- q(X),r(X,Y)`). Conversely, theories of any length can be combined using the OR operation, and the resulting theory's length is equal to the sum of the lengths of the combined theories (for example, combining theories of length one `t1(X):- q(X)` and `t2(X):- r(X,Y)` using the OR operation would result in theory `t(X):- q(X); r(X,Y)` of length 2).

SkILL's algorithm is composed of two main steps: (i) building theories of length one (single rules) using the AND operation, and (ii) building theories of length greater than one using the OR operation. In step (i), single rules of increasing number of literals are built from the mode declarations using the AND operation. Adding literals to a rule in conjunction makes the resulting rule more specific. Once all possible rules are built and evaluated, the algorithm proceeds to step (ii) using the OR operation to combine single rules (theories of length one) into theories of greater length, up to a maximum length. By adding rules to a theory in disjunction, the resulting theory becomes more general.

In order to assess a theory's fitness, its exact probabilistic value for each example must be computed in a Probabilistic Logic Programming language (ProbLog is th language used in this work), so that the theory is *evaluated exactly*. This process can be very time consuming, since the evaluation process must consider all possible worlds where the theory may be true. For a small number of facts in the PBK this is not a problem, but exact computation grows exponentially as the size of the PBK is increased. Consider the process of evaluating exactly the theory `t(X):- q(X), r(X,Y)`. ProbLog would need to compute all possible worlds for this theory in order to assess the overall error of the theory's predictions against the examples. Whether the theory is stored for further combinations or discarded after the evaluation stage, the system has already spent a considerable amount of time just to evaluate it.

To mitigate this problem, this work introduces the *estimation pruning* strategy, which can discard theories based on their previously evaluated subparts. For instance, suppose that theories `t1(X):- q(X)` and `t2(X):- r(X,Y)` had already been evaluated – in that case, it is possible to make an estimation of the value of `t(X):- q(X), r(X,Y)` based on this information. Thus, estimation pruning consists of ruling out theories that have poor estimations and exactly evaluating theories that have good estimations. In SkILL, the decision on whether a theory is discarded is made based on one of two criteria: *soft pruning* or *hard pruning*. After the initial step of estimating the values for each example, the estimated value's usefulness is assessed according to one of these criteria. Note that the criteria are directly applicable to the estimated probabilistic values in lieu of the exact predictions of a theory. The combination is then pruned away if it is found to be useless. Conversely, if the combination is considered useful, then exact probabilistic evaluation is performed and the theory and its exact evaluation are saved for the next iteration.

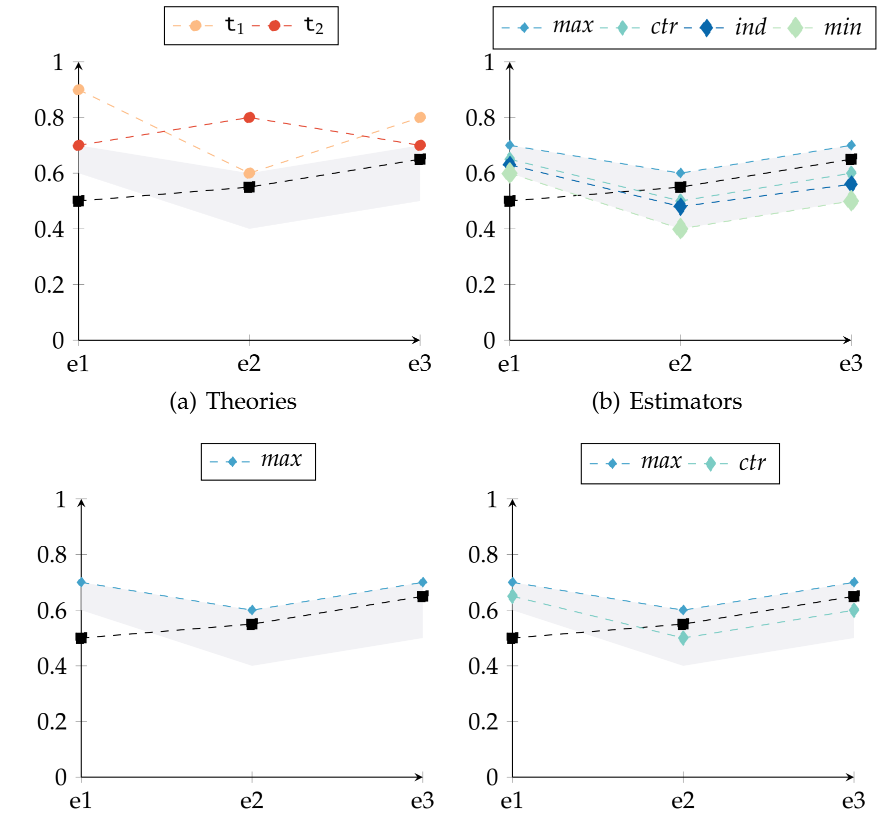
3 Estimation Pruning

Estimation pruning consists of estimating the predictions of two theories combined based on the individual predictions of each theory. Estimation pruning excludes combinations of theories whose *estimated predictions* suggest that the resulting theory will be too specific (for the AND operation) or too general (for the OR operation). This process is somehow similar to the evaluation of theories in ILP. For instance, the more specific theory $\texttt{t}_s$ will not cover more positive examples than a more general theory $\texttt{t}_g$. Therefore, if $\texttt{t}_s$ covers less positive examples than $\texttt{t}_g$ or if it maintains the same coverage but does not reduce the coverage of some negative examples, it will be discarded.

In the PILP setting, the exact probabilistic evaluation of a theory corresponds to the weighted proportion of worlds where the theory is true. The probabilistic value for an example `e` using a theory `t` is given by determining in how many worlds (of all possible worlds in the PBK) `t(e)` is true. The challenge in estimating the value of a probabilistic evaluation knowing the values of the theories being combined lies in the fact that the *amount of overlapping* of the sets of worlds corresponding to those two theories is unknown before evaluation. If two theories are mutually exclusive (or disjoint) w.r.t. the PBK, then their overlap is null. On the other hand, if a theory is more specific than another, the former will cover a subset of the worlds covered by the latter. Theories can also be independent, meaning that the probability that one of the theory is true in a world does not change the probability the other theory is also true in that world.

Despite this uncertainty, it is possible to calculate the interval where the predictions of a combination of two theories will be (this is depicted in Fig. 1 as a shaded area). The lower and upper bounds of the interval are determined by the predictions of the theories that are being combined ($\texttt{t}_1$ and $\texttt{t}_2$ in Fig. 1(a)). Depending on where the resulting theory `t` will lie in the interval, the (vertical) distance between `t`'s values and the example values (squares in Fig. 1) will vary, and as `t` converges to the examples, its prediction quality is improved.

This work presents five estimators that can be used to estimate the value of theories, namely: *minimum*, *maximum*, *center*, *independence* and *exclusion*. These estimators predict different sets of values inside the estimation interval, based on different set theory cases. The *minimum* and *maximum* estimators correspond to the lower and upper boundaries of the estimation interval (*min* and *max* estimators in Fig. 1, respectively). The *center* estimator (*ctr* in Fig. 1) is the center of the estimation interval (halfway between *minimum* and *maximum*). The *independence* estimator (*ind* in Fig. 1) assumes that theories $\texttt{t}_1$ and $\texttt{t}_2$ are independent and calculates the values of their combination accordingly. The *exclusion* estimator (not depicted in Fig. 1) assumes that the theories $\texttt{t}_1$ and $\texttt{t}_2$ are as exclusive as possible. In the AND operation, the *exclusion* estimator is equal to the *minimum* estimator, since when two theories are mutually exclusive,



(a) Theories (b) Estimators

(c) Hard Pruning (d) Soft Pruning

Fig. 1. Estimators in AND operation. The x-axis contains three examples and the y-axis represents probabilistic values. Examples are depicted as squares, theories t_1 and t_2 as circles and estimators *min*, *max*, *ctr*, *ind* as diamonds.

their amount of overlap is minimum. The first row in Table 2 summarizes the expressions used to calculate these estimations.

After calculating an estimation for the combination of theories $\mathtt{t}_1$ and $\mathtt{t}_2$, it is necessary to decide – based on the estimation – whether the combination of theories should be evaluated exactly. To that effect, two pruning criteria can be used: hard pruning (Fig. 1(c)) or soft pruning (Fig. 1(d)). In the case of the AND operation, the hard pruning criterion discards theories that are too specific in

Table 2. Expressions used to calculate estimations

Operation	*minimum*	*maximum*	*center*	*independence*	*exclusion*
AND	$max(0, A+B-1)$	$min(A,B)$	$\frac{1}{2}(min(A,B) + max(0, A+B-1))$	$A \times B$	$max(0, A+B-1)$
OR	$max(A,B)$	$min(A+B, 1)$	$\frac{1}{2}(max(A,B) + min(A+B, 1))$	$A + B - A \times B$	$min(A+B, 1)$

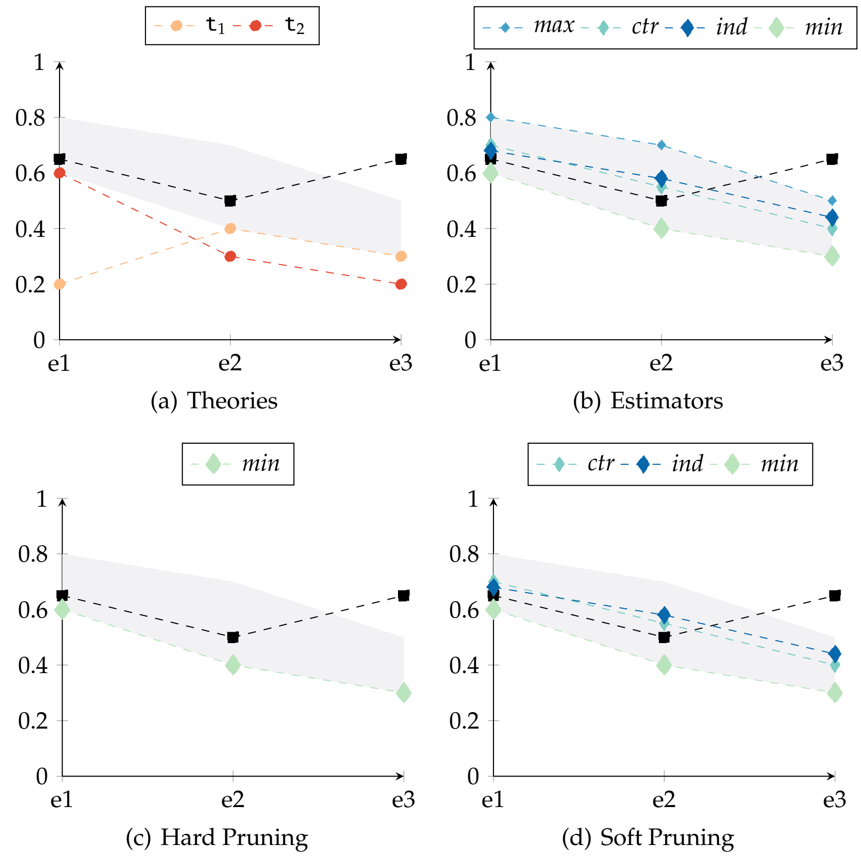


Fig. 2. Estimators in OR operation. The x-axis contains three examples and the y-axis represents probabilistic values. Examples are depicted as squares, theories t_1 and t_2 as circles and estimators *min*, *max*, *ctr*, *ind* as diamonds.

any of their predictions. This means that the estimations must be higher than or equal to the examples' values in every point (in Fig. 1(c) this only happens if the *maximum* estimator is being used to estimate the combination). On the other hand, the soft pruning criterion only prunes the theory away if it is *overall* more specific than the example values. In Fig. 1(d), the estimators that are not discarded are those that are above (*maximum*) or equally above and below (*center*) the examples' values. Estimator *center* is kept because its estimations are just a small distance below two example values but are large distance above the first example value, which balances out.

The concept of pruning combinations of theories can be extended to the OR operation. Similarly to the AND operation, this strategy estimates the value of a combination of two theories. In the OR setting, theories are excluded when they are found to be too general to benefit from further combination. Based on the expressions presented in Table 2 and following a similar reasoning to the AND operation, the same five estimators can be defined. Again, the *minimum* and *maximum* estimators define the estimation interval based on t_1 and t_2 (Fig. 2(a)). The *center* estimator is the value halfway between the lower and upper boundaries of the estimation interval and the *independence* estimator assumes theories are independent. In the OR operation, the *exclusion* estimator is equal to the *maximum* estimator, because when the overlap of two theories is minimum (they are exclusive), the largest area is covered. All estimators are depicted in Fig. 2(b).

The hard and soft pruning criteria can also be extended to the OR operation. The hard pruning criterion (Fig. 2(c)) now excludes estimations that are too general in any point to be of interest. This translates in keeping only estimators whose values are always lower than or equal to the examples' values (only estimator *minimum* passes the hard pruning criterion in Fig. 2). Similarly to the AND operation, the soft pruning criterion only discards estimators whose values are overall more general than the examples' values (in Fig. 2(d), estimators *minimum*, *center* and *independence* pass this test, because they are overall under the examples' values).

4 Experiments

The experiments presented in this section were run on a machine containing 4 AMD Opteron 6300 processors with 16 cores each and a total of 250GB of RAM. Table 3 presents a summary of the dataset characteristics used in our experiments. The **metabolism** dataset is an adaptation of the dataset originally from the 2001 KDD Cup Challenge[1]. The **athletes** dataset consists of a subset of facts regarding athletes and the sports they play collected by the never-ending language learner NELL[2]. 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. For the **metabolism** and **athletes**, a number of n-times holdout sets were made and all measurements were averaged out over the folds. In the **breast cancer** leave-one-out cross-validation was used.

Different combinations of estimation pruning were tested: only pruning the AND operation, only pruning the OR operation, and pruning both operations. The pruning settings are reported as a set of two letters: the first letter is the AND pruning option and the second is the OR pruning option. Pruning options can be soft pruning (S), hard pruning (H) or no pruning (x). For example, using this codification, xS stands for no AND pruning and soft OR pruning. For each configuration, several measurements were recorded for each dataset: execution time, accuracy on the test set, and number of rules and theories pruned.

Tables 4, 6 and 5 present the speedups and ratio of probabilistic accuracy for the **metabolism**, **athletes** and **breast cancer** datasets, respectively. The speedup

[1] `http://www.cs.wisc.edu/~dpage/kddcup2001`
[2] `http://rtw.ml.cmu.edu`

Table 3. Dataset characteristics: no. of examples; no. of facts in the PBK and proportion of probabilistic facts in brackets; no. of folds (n stands for n-times holdout sets and lo for leave-one-out); no. of examples in the train set and proportion of the dataset in brackets; and no. of examples in the test set and proportion of the dataset in brackets.

Dataset	Examples	PBK	Folds	Size train	Size test
metabolism	230	7000 (46%)	30 n	160 (70%)	70 (30%)
athletes	721	4294 (100%)	30 n	505 (70%)	216 (30%)
breast cancer	130	13400 (3%)	130 lo	129 (99%)	1 (1%)

$\frac{B_t}{P_t}$ is calculated w.r.t. the B_t base case time (no pruning) for different P_t pruning options' execution time. If there is a slowdown, the inverse speedup $\frac{P_t}{B_t}$ is presented as a negative number. The ratio of the probabilistic accuracy $\frac{P_a}{B_a}$ is calculated for each probabilistic settings P_a w.r.t the probabilistic accuracy of the B_a base case. Similarly to the speedup, when the accuracy decreases, the inverse of the ratio is given $\frac{B_a}{P_a}$ as a negative number. Figures 3, 5 and 4 depict the variation in execution time in minutes (left y-axis) and the variation in accuracy (right y-axis) for all estimators in the **metabolism**, **athletes** and **breast cancer** datasets, respectively. The estimators analysed were the base case (no estimation pruning performed, or *nop*), *minimum* (*min*), *maximum* (*max*), *center* (*ctr*), *independence* (*ind*), and *exclusion* (*exc*). Each dataset's results will be discussed next.

For the **metabolism** dataset, results in Table 4 show that the greatest reduction in execution time is achieved by all estimators in the HH pruning setting. The xS pruning setting presents the slowest execution times and all estimators, except *center*, cause a slight slowdown. Results also show that there is no significant reduction in accuracy in any setting. Figure 3 shows that overall the accuracy of the theories is unchanged and that the *maximum*, *center* and *exclusion* estimators can all reduce execution time from 40 to less than 25 minutes.

Results in Table 5 show that, in the **breast cancer** dataset, the greatest reduction in execution time can be achieved by using pruning in both the AND and the OR operations (SS and HH settings). The pruning settings that use only OR pruning (xS and xH) present more modest reductions of execution time (about 1.5 times) when compared to the settings that use only AND pruning (about 7 times). The predictive accuracy of the best theory in this dataset never decreases, and in some settings (Sx, Hx, SS and HH in Table 5) even increases slightly. This effect is due to a reduction in overfitting caused by the exclusion of some theories that are better on the training set but perform worse on the test set. Figure 4 shows that, on average, the *maximum*, *center* and *exclusion* datasets can reduce execution time from over 4 minutes to about 1 minute.

In the **athletes** dataset, again the HH pruning setting can reduce most execution time. However, the reduction using estimators *minimum* and *center* is much less than that of estimators *maximum*, *independence* and *exclusion*, where the execution is about 50 times faster (Table 6). Estimators *minimum* and *center* are consistently slower in other pruning settings (xS, xH and SS), and the xS and xH settings present the lowest reduction in execution time in this dataset, of 2

Table 4. Speedup and probabilistic accuracy ratio for **metabolism** dataset

Speedup

Est	Sx	Hx	xS	xH	SS	HH
min	1.45	1.47	-1.03	1.36	1.47	2.52
max	1.56	1.56	-1.09	1.94	1.61	5.66
ctr	1.57	1.58	1.03	1.95	1.61	5.65
ind	1.35	1.33	-1.23	1.64	1.46	5.37
exc	1.57	1.58	-1.04	1.95	1.58	5.69

Probabilistic Accuracy Ratio

Est	Sx	Hx	xS	xH	SS	HH
min	1.00	1.00	1.00	1.01	1.00	1.00
max	1.00	1.00	1.00	1.00	1.00	-1.01
ctr	1.00	1.00	1.00	1.00	-1.01	-1.01
ind	1.00	1.00	1.00	1.00	1.00	-1.01
exc	1.00	1.00	1.00	1.00	1.00	-1.01

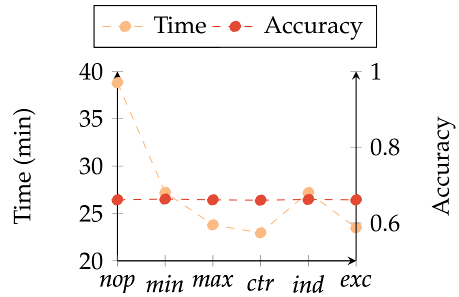


Fig. 3. Average time (in minutes) and accuracy in **metabolism** dataset, for the base case (*nop*) and the five estimators. Values for each estimator are the average of its result over the pruning options.

times on average. Similarly to the other datasets, Table 6 shows that the accuracy in the **athletes** dataset presents no significant reduction and, in particular, in the xS, xH and HH settings it is not reduced at all. Estimators *maximum*, *independence* and *exclusion* present the greatest overall reduction in execution time (Fig. 5), from 20 to about 5 minutes, on average. Finally, for the **athletes** dataset, Table 7 presents the number of probabilistic evaluations performed for each pruning setting and estimator. The first number corresponds to single rules (theories of length one) evaluated, and thus the reduction is caused by AND pruning. Sim-

Table 5. Speedup and probabilistic accuracy ratio for **breast cancer** dataset

Speedup

Est	Sx	Hx	xS	xH	SS	HH
min	7.09	7.18	1.42	1.41	22.44	21.46
max	7.16	7.06	1.65	1.63	25.24	23.49
ctr	7.04	6.98	1.63	1.63	25.25	24.80
ind	7.20	7.02	1.42	1.29	22.62	22.84
exc	7.19	7.19	1.63	1.62	25.00	24.80

Probabilistic Accuracy Ratio

Est	Sx	Hx	xS	xH	SS	HH
min	1.09	1.09	1.00	1.00	1.09	1.09
max	1.09	1.09	1.00	1.00	1.09	1.09
ctr	1.09	1.09	1.00	1.00	1.09	1.09
ind	1.09	1.00	1.00	1.00	1.09	1.09
exc	1.09	1.09	1.00	1.00	1.09	1.09

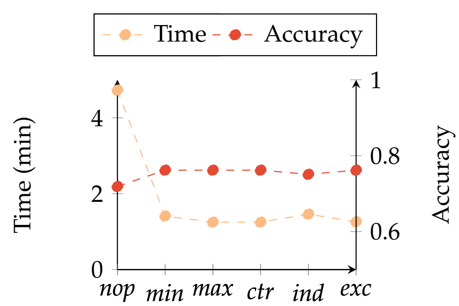


Fig. 4. Average time (in minutes) and accuracy in **breast cancer** dataset, for the base case (*nop*) and the five estimators. Values for each estimator are the average of its result over the pruning options.

Table 6. Speedup and probabilistic accuracy ratio for **athletes** dataset

Speedup

Est	Sx	Hx	xS	xH	SS	HH
min	3.34	3.33	1.01	1.66	3.63	9.48
max	3.35	3.35	2.12	2.19	12.40	49.72
ctr	3.20	3.28	1.00	1.80	3.62	19.82
ind	3.33	3.34	2.10	2.17	12.34	50.36
exc	3.31	3.23	2.02	2.11	11.86	48.01

Probabilistic Accuracy Ratio

Est	Sx	Hx	xS	xH	SS	HH
min	-1.06	-1.06	1.00	1.00	-1.11	1.00
max	-1.06	-1.06	1.00	1.00	-1.15	1.00
ctr	-1.06	-1.06	1.00	1.00	-1.08	1.00
ind	-1.06	-1.06	1.00	1.00	-1.15	1.00
exc	-1.06	-1.06	1.00	1.00	-1.15	1.00

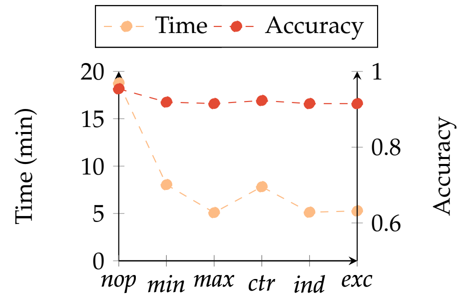


Fig. 5. Average time (in minutes) and accuracy in **athletes** dataset, for the base case (*nop*) and the five estimators. Values for each estimator are the average of its result over the pruning options.

ilarly, the second number in each cell is the number of theories of length greater than one, and its reduction is caused by OR pruning. The greatest reductions correspond to the HH setting (column 8 in Table 7), and are consistent with the setting in Table 6 that presents the greatest speedups. Additionally, from Figure 5, the three fastest estimators (in average) are also the estimators that in Table 7 prune away most theories. In particular, the number of theories pruned away during OR pruning is significantly lower for estimators *max*, *ind* and *exc* when compared to estimators *min* and *ctr*. The same trend can be observed in the other datasets but results were omitted due to lack of space.

Table 7. Number of single rules/theories evaluated for the **athletes** dataset

Est	xx	Sx	Hx	xS	xH	SS	HH
min	2414/1989	164/968	164/968	2414/1981	2414/604	164/913	164/361
max	2414/1989	164/968	164/968	2414/69	2414/0	164/243	164/0
ctr	2414/1989	164/968	164/968	2414/1974	2414/381	164/907	164/128
ind	2414/1989	164/968	164/968	2414/69	2414/0	164/243	164/0
exc	2414/1989	164/968	164/968	2414/69	2414/0	164/243	164/0

5 Conclusion

This work proposed five PILP estimators whose aim is to alleviate the overhead imposed by the exact evaluation of combinations of candidate probabilistic theories. Because PILP theories can be built using both conjunction (AND operation) and disjunction (OR operation), the estimators must be adapted accordingly.

The estimators were implemented in the estimation pruning stage of the SkILL system, but can be generalized to any PILP engine. Experiments using different pruning combinations were performed on three real-world datasets. Results showed that all estimators resulted in faster execution times when coupled with an H pruning setting and, in particular, the HH pruning setting showed the greatest speedups and also the greatest reduction in the number of probabilistic evaluations performed. Even though all estimators maintain predictive quality and reduce execution time, estimators *maximum* and *exclusion* are overall faster, and opting for one of these estimators in lieu of estimators *minimum*, *center* or *independence* can result in an up to 5 times faster runtime for the same pruning setting. Future work includes adding an estimator that divides the estimation interval according to an user-defined distance and dynamically adapting the estimator setting during runtime.

Acknowledgments

Joana Côrte-Real is funded by the FCT grant SFRH/BD/52235/2013. This work is partially funded by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, by National Funds through the FCT as part of project UID/EEA/50014/2013, and by the North Portugal Regional Operational Programme, under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund as part of project NanoSTIMA (NORTE-01-0145-FEDER-000016).

References

1. 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.
2. J. Côrte-Real, T. Mantadelis, I. Dutra, R. Rocha, and E. Burnside. SkILL - a Stochastic Inductive Logic Learner. In *Proceedings of the 14th International Conference on Machine Learning and Applications*, pages 555–558. IEEE, 2015.
3. 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.
4. L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *International Conference on Algorithmic Learning Theory*, pages 19–36. Springer, 2004.
5. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *International Joint Conference on Artificial Intelligence*, volume 99, pages 1300–1309, 1999.
6. 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.
7. M. Richardson and P. Domingos. Markov Logic Networks. *Machine learning*, 62(1-2):107–136, 2006.
8. 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.