Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

Pruning Strategies for the Efficient Traversal of the Search Space in PILP Environments

Joana Côrte-Real[1,2] · **Inês Dutra**[1,3] · **Ricardo Rocha**[1,2]

Received: 23 Jul 2020 / Revised: 23 Jul 2021 / Accepted: 11 Sep 2021

Abstract Probabilistic Inductive Logic Programming (PILP) is a Statistical Relational Learning technique which extends Inductive Logic Programming by considering probabilistic data. The ability to use probabilities to represent uncertainty comes at the cost of an exponential evaluation time when composing theories to model the given problem.For this reason, PILP systems rely on various pruning strategies in order to reduce the search space. However, to the best of the authors' knowledge, there has been no systematic analysis of the different pruning strategies, how they impact the search space, and how they interact with one another.

This work presents a unified representation for PILP pruning strategies which enables end-users to understand how these strategies work both individually and combined, and to make an informed decision on which pruning strategies to select so as to best achieve their goals. The performance of pruning strategies is evaluated both time and quality-wise in two state-of-the-art PILP systems with datasets from three different domains. Besides analysing the performance of the pruning strategies, we also illustrate the utility of PILP in one of the application domains, which is a real world application.

Keywords Statistical Relational Learning · Probabilistic Inductive Logic Programming · Search Space Evaluation · Implementation · Performance

1 Introduction

Relational Learning is the set of Data Mining techniques used to discover non-trivial knowledge in contexts where data may have complex relationships. Some examples of such techniques are Inductive Logic Programming (Muggleton and Raedt 1994), Graph Mining (Cook and Holder 2006), Relational Data Mining (Džeroski 2010) and Relational Reinforcement Learning (Džeroski et al. 2001). Statistical Relational

[1]Department of Computer Science, Faculty of Sciences, University of Porto
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
[2]CRACS & INESC TEC, [3]CINTESIS
E-mail: {jcr,ines,ricroc}@dcc.fc.up.pt

Learning (SRL) (De Raedt and Kersting 2004) extends Relational Learning by: (i) allowing the training examples to be annotated with probabilistic information known a priori, and (ii) producing classifiers whose prediction is a probability (as opposed to a class). 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.

Probabilistic Inductive Logic Programming (PILP) is a subset of SRL which extends Inductive Logic Programming (ILP) to compose theories as understandable First Order Logic (FOL) sentences based on data annotated with probabilistic information. PILP algorithms use a set of Probabilistic Examples (PE) and logical information pertaining complex relations expressed as logic facts, the Probabilistic Background Knowledge (PBK), to find the FOL model that best 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 in this technique by tuning the probabilistic output of the theories which are learnt (De Raedt et al. 2015). PILP differs from other SRL techniques in (i) the data being represented as logical predicates and the model being a logical theory itself; and (ii) the focus on learning the (logical) structure of the data inductively, by using ILP algorithms to find the logical model which best explains the data.

In order to learn (probabilistic) theories from probabilistic data, PILP systems need to rely on a specific language that can code probabilities. There are many probabilistic inference systems that can represent and manipulate probabilities, such as SLP (Muggleton 1996), ICL (Poole 1997), Prism (Sato and Kameya 1997), BLP (Kersting et al. 2000), CLP($\mathcal{BN}$) (Santos Costa et al. 2002), MLN (Richardson and Domingos 2006), ProbLog (Kimmig et al. 2011), among others. Whilst there are a number of probabilistic logic languages in the literature, there are few works dedicated to performing structure learning over one of these languages, and thus representing uncertain knowledge in a human-readable form. PILP approaches such as ProbFOIL (De Raedt and Thon 2011; De Raedt et al. 2015), SLIPCOVER (Bellodi and Riguzzi 2012, 2015), and SkILL (Côrte-Real et al. 2015) can perform this task, and there are ILP-based structure learning methods in the CLP($\mathcal{BN}$) (Santos Costa et al. 2002) and MLN (Kok and Domingos 2005) languages as well. The ProbFOIL+ and SkILL systems are both based on the ProbLog language, whilst SLIPCOVER is based on LPADs (Vennekens et al. 2004).

PILP systems evaluate the fitness of each candidate theory by computing the probability of the theory entailing each example. 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 and relations in the PBK this may not be a problem, but exact computation grows exponentially as the size of the PBK is increased. In order to make the search space traversal less computationally taxing, a *pruning strategy* aimed at searching for the most relevant theories can be used. In previous work, we have already proposed different pruning strategies for PILP environments, namely *fitness pruning* (Côrte-Real et al. 2015), *prediction pruning* (Côrte-Real et al. 2018) and *estimation pruning* (Côrte-Real et al. 2016). This work presents a unified representation of these pruning strategies and how they can be applied in distinct parts of PILP algorithms, thus enabling end-users to better understand how they work not only individually but also combined. These pruning strategies can follow alternative criteria, with varying degrees of strictness to prune the PILP search space. Because those criteria are based on the data's probabilistic characteristics, the pruning strategies result in minimum information loss, which in turn maintains the quality of the generated theories while significantly reducing average execution time.

The main contributions of this work are: (i) a unified representation and description of the three pruning strategies in the context of a generic algorithm for traversing the search space in PILP environments; (ii) the implementation of these pruning strategies within the SkILL and ProbFOIL+ systems (chosen because both are ProbLog based); and (iii) an experimental evaluation of how these pruning strategies can be used in conjunction with each other. Experiments are performed on three datasets: a dataset regarding metabolism interactions, a knowledge base extracted from the web, and one real-world medical dataset. In general, the pruning strategies are shown to shorten execution time while maintaining the quality of the generated theories. To the best of the authors' knowledge, the effect of applying several pruning strategies in conjunction in a PILP system has never been studied before. The fact that PILP systems may be able to prune out a significant part of the search space is a relevant improvement to the exhaustive PILP algorithm, where the search space grows exponentially. Besides evaluating the effect of applying several pruning strategies, we also explore the utility of PILP in one of the domains, the medical dataset. We show how to represent the probabilistic medical knowledge, including probabilities to the examples, and how the SkILL models compare with the medical expert mental model.

The remainder of the paper is organized as follows. First, Section 2 presents the main background concepts of ILP and PILP. Next, Section 3 presents the PILP pruning strategies and how they fit together in the PILP search space exploration algorithm. In Section 4, the experimental methodology and the datasets used in this work are described, followed by results and discussion. Before ending, in Section 5, we discuss about how PILP can be used to represent knowledge in the medical domain using one of our datasets and compare the performance of SkILL with expert medical doctors. At the end, in Section 6, final remarks and perspectives of future work are put forward.

2 Background

2.1 ILP

ILP is a machine learning method which stands out due to its suitability for relational data analysis. ILP's main goal is to construct a *theory* which can explain a set of observations - known as *examples* - or that can be used to build a predictive model (Muggleton and Raedt 1994). In this work, an ILP theory refers to a set of Horn clauses combined (disjunctively), and each individual clause is termed a *rule*. A rule is composed of a head and a body, and follows the structure presented in Eq. 1 (using Prolog's syntax).

$$head :- body\_literal_1, body\_literal_2, ..., body\_literal_N. \qquad (1)$$

The body of a rule is composed by a sequence of *literals*, or *goals*, interacting with one another through *connectives*, in this case the AND connective (represented by `,/2` in Prolog's syntax). Each goal represents a call to a predicate, which is then determined to be true or false. The number of literals in the body of a rule is termed the rule *length*. The more literals are contained in the body of a rule (the greater its length), the more specific the rule is. All rules of various lengths which can be formed from the existing literals in a program form an AND search space.[1] More formally, let *Literals* be the set of distinct literals in the program. The AND search space *Rules* is then the power set of *Literals*.

$$Rules = \mathcal{P}(Literals) \tag{2}$$

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. The number of rules in a theory corresponds to its *length*. All rules are thus theories of length one. Because theories are combined using logical disjunction (as opposed to logical conjunction for rules), adding a rule to a theory makes it more general. The OR search space *Theories* is the set of all theories such that:

$$Theories = \mathcal{P}(Rules) \tag{3}$$

The theories used to explain examples in ILP are built from the literals that are present in the program's Background Knowledge (BK). The BK of an ILP program consists of the support knowledge that one would have about the problem at hand, and is often of a multi-relational nature. It is therefore crucial to have structured and well-suited BK for learning. Fully exploring the ILP search space is equivalent to evaluating each theory in the OR search space in order to determine the best theory w.r.t. the Examples (E) and according to a given metric.

2.2 PILP

One of the limitations of (deterministic) ILP is that, whilst it can be easily used to express relational data, it does not allow for any measure of uncertainty. The ability to take uncertainty into account when building a declarative model of a real-world phenomenon 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. One way to incorporate uncertainty into PLP consists of using Sato's distribution semantics (Sato 1995), where a program is generalised to a distribution over a set of logic programs that share the original definite clauses, but differ in the set of facts. There are several Prolog-based PLP languages in the literature, but only ProbLog will be discussed here since a comparison of PLP languages is out of the scope of this work. For such a comparison see De Raedt and Kimmig (2015).

ProbLog is based on the *possible world semantics* (Kimmig et al. 2011). A ProbLog program consists of a set of probabilistic facts F and a set of deterministic rules R.

[1] Depending on the *language bias* (mechanism employed to constrain the search space), it might also be the case that the same literal with different arguments can occur multiple times in the rule. However, for the description of the search space, repeated literals in rules can be mimicked by introducing auxiliary predicates (for instance). Therefore, for the sake of simplicity, this case is not considered in what follows.

Each fact $p_j :: f_j$ in the program represents an independent binary random variable, 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 facts of the program 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. 4.

$$P(\omega_i) = \prod_{f_j \in \omega_i^+} p_j \prod_{f_j \in \omega_i^-} (1 - p_j) \tag{4}$$

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. 5, with R being a set of deterministic rules as previously mentioned.

$$P(q) = \sum_{\omega_i^+ \cup R \models q} P(\omega_i) \tag{5}$$

2.2.1 From Rules to Probabilities

PILP differs from ILP in that the *Probabilistic* Examples (PE) have success probabilities ranging between 0 and 1, as opposed to being either false or true. Facts and rules in the *Probabilistic* Background Knowledge (PBK) can also be annotated with a probabilistic value ranging from 0 to 1, which can represent either statistical information or the degree of belief in a statement, using type I or type II probability structures, respectively (Halpern 1990). In this setting, there are no longer positive and negative examples, but only target probabilities for each example. The aim of a PILP theory with respect to probabilistic examples is thus to produce probability values for each one of them which are as close as possible to the target probability for that example. Despite the similarities between ILP and PILP, there are several syntactic differences between them. Table 1 summarises these differences using Prolog syntax for ILP and ProbLog syntax for PILP.

Table 1 Main syntactic 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).`
		`clause1(X,Y):- fact1(E,X),fact2(E,Y).`
		`clause2(Y):- fact2(E,Y),clause1(X,Y).`
			`target(E):-`
			`fact2(E,V),clause1(V,V).`
			$\downarrow$
			truth value $\in$ {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}`::clause1(X,Y):- fact1(E,X),fact2(E,Y).`
		p_{c2}`::clause2(Y):- fact2(E,Y),clause1(X,Y).`
			`target(E):-`
			`fact2(E,V),clause1(V,V).`
			$\downarrow$
			truth value $\in [0, 1]$

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. Therefore, the search space algorithm of PILP has the same efficiency issues of ILP's. Furthermore, PILP adds an extra level of complexity due to the probabilistic evaluation of theories w.r.t. the PE.

The fact that the PBK is now composed of probabilistic facts and relations alters the semantics of the logical part of the predicates as well, when compared to the deterministic version. When a rule is generated in PILP, its success probability (i.e., the success probability of the body) is calculated for each example. This value is called the *prediction* of the rule for an example. As such, different literals in the body of a rule will generate different success probabilities for an example, depending on the probabilities of the probabilistic facts in the underlying model.

Another 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.

2.2.2 Search Space Traversal

Similarly to ILP, fully exploring the PILP search space is equivalent to evaluating each theory in order to determine the best theory according to a given metric. The exhaustive procedure to explore the search space, starting by exploring the rule space (AND search) and only then proceeding to the theory space (OR search), is given in Alg. 1 and Alg. 2, respectively.

Algorithm 1 *and_search_space*(PBK, PE)

```
1: $R_1 = generate\_rules\_of\_length\_one(PBK, PE)$
2: $T_1 = R_1 = R_N = prob\_evaluation(R_1, PBK, PE)$
3: $R_{N+1} = combine\_rules(R_1, R_N)$
4: while $R_{N+1} \neq \emptyset$ do
5:     $R_{N+1} = prob\_evaluation(R_{N+1}, PBK, PE)$
6:     $T_1 = T_1 \cup R_{N+1}$
7:     $R_N = R_{N+1}$
8:     $R_{N+1} = combine\_rules(R_1, R_N)$
9: return $T_1$
```

Algorithm 2 *or_search_space*(T_1, PBK, PE)

```
1: $T_{All} = T_N = T_1$
2: $T_{N+1} = combine\_theories(T_1, T_N)$
3: while $T_{N+1} \neq \emptyset$ do
4:     $T_{N+1} = prob\_evaluation(T_{N+1}, PBK, PE)$
5:     $T_{All} = T_{All} \cup T_{N+1}$
6:     $T_N = T_{N+1}$
7:     $T_{N+1} = combine\_theories(T_1, T_N)$
8: return $T_{All}$
```

Algorithms 1 and 2 are similar to the ILP algorithm in that they start out with rules (or theories) of shorter length and proceed to specify (or generalise) them. The main difference relies on the evaluation procedure. In the case of ILP, it is a discrete evaluation which verifies the truth value of a theory for all the examples. In the case of PILP, the probability of success is computed for each example (lines 2/5 in Alg. 1 and line 4 in Alg. 2). This is also the reason for the greater execution time and complexity of PILP when compared to ILP – the cost of a probabilistic evaluation is much greater than that of computing the truth value of a theory.

2.2.3 Loss Function and Theory Evaluation

Predictions of an ILP theory are either 1 or 0 (true or false, respectively) and, for the purpose of this work, they will be interpreted as numeric values. This interpretation is not strictly necessary to compute these metrics in ILP, but this distance based approach extends directly to PILP and predictions ranging between 0 and 1. A theory can thus be used to compute a *prediction* for each example. For some theory t, an example i and its corresponding value e_i, the theory's prediction is defined as $p_i(t)$. The closer the prediction $p_i(t)$ is to the original example value e_i, the better that theory is, for that example i.

The assessment of the fitness of a theory to describe the problem is measured according to some loss function. Loss functions can also be defined in terms of the distance $d_i(t)$ between a prediction $p_i(t)$ and an example value e_i. For ILP, the distance $d_i(t)$ will be either -1, 0 or 1. For PILP, the distance $d_i(t)$ can be any real number ranging from -1 to 1. If the distance is zero, then the prediction of the model is correct. Otherwise, the prediction includes some error. Several metrics considering distance $d_i(t)$ over all examples are possible, namely Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). These metrics calculate the fitness of a theory t based on the linear or quadratic average distance between its predictions and the example values, respectively Eq. 6 and Eq. 7, over all examples $i \in E$.

$$MAE(t) = \frac{1}{|E|} \sum_{i \in E} |d_i(t)| \tag{6}$$

$$RMSE(t) = \sqrt{\frac{1}{|E|} \sum_{i \in E} d_i(t)^2} \tag{7}$$

Alternatively, loss functions can be defined in terms of true positive ($TP(t)$), true negative ($TN(t)$), false positive ($FP(t)$) and false negative ($FN(t)$) parts of examples, as shown in Eq. 8.

$$\begin{aligned} TP(t) &= \sum_{i \in E} min(e_i, p_i(t)) \\ TN(t) &= \sum_{i \in E} min(1 - e_i, 1 - p_i(t)) \\ FP(t) &= \sum_{i \in E} max(0, 1 - e_i - min(1 - e_i, 1 - p_i(t))) \\ FN(t) &= \sum_{i \in E} max(0, e_i - min(e_i, p_i(t))) \end{aligned} \tag{8}$$

Accuracy is used in classification problems to assess the number of correct predictions in proportion to all predictions made. Extending these concepts to the probabilistic setting allows for defining standard scoring metrics, such as Probabilistic Accuracy (PAcc) (De Raedt and Thon 2011). The PAcc metric can be defined in terms of either MAE or TP, TN, FP, FN metrics. The two formulations are equivalent, as shown in Eq. 9.

$$\begin{aligned}
PAcc(t) &= \frac{TP(t) + TN(t)}{TP(t) + TN(t) + FP(t) + FN(t)} \\
&= \frac{1}{|E|} \sum_{i \in E} \Big(min(e_i, p_i(t)) + min(1 - e_i, 1 - p_i(t)) \Big) \\
&= \frac{1}{|E|} \sum_{i \in E} \Big(1 - |e_i - p_i(t)| \Big) \\
&= \frac{1}{|E|} \sum_{i \in E} \Big(1 - |d_i(t)| \Big) \\
&= 1 - MAE(t)
\end{aligned} \tag{9}$$

3 Pruning Strategies

The PILP search space can be split in two separate dimensions w.r.t. the operation that is being used to traverse it. This means that there is a search space for rules (theories of length one), which uses the AND operation to navigate between them, and a search space for theories (of length greater than one), which in turn uses the OR operation to generate new theories. In what follows, these two dimensions will be referred as the *AND search space* and the *OR search space*, respectively.

Traversing the PILP search spaces exhaustively is of exponential complexity with the size of the PBK (both literals and facts), which requires significant computational resources as datasets grow larger. To address this issue, three distinct ways to prune the PILP search space can be used: *fitness pruning*, *prediction pruning* and *estimation pruning*. These pruning strategies are based on the probabilistic information of candidate theories and can be applied in different parts of the PILP search space traversal process. Fitness pruning implements a polynomial bound complexity on the PILP search space by selecting a subset of theories to be combined (as opposed to calculating all possible combinations). Prediction pruning can safely reduce the search space based on the probabilistic evaluation of candidate theories at each step in the rule and theory combination procedure. Estimation pruning is designed to save computational time by avoiding unnecessary exact probabilistic evaluation of both rules and theories which will most likely be useless.

The three pruning strategies can be used individually or in combination with each other. Algorithms 3 and 4 show a simplified version of how the three pruning strategies can be applied to the AND and OR search space in PILP. Prediction pruning is applied over previously evaluated rules and theories to remove candidates that can not be improved on during the next iteration (lines 3 and 9 in Alg. 3 and lines 2 and 8 in Alg. 4, respectively). Rules or theories are then selected from the set of candidates after prediction pruning to make combinations of length $N + 1$.

This combination process can now use fitness pruning to limit the number of combinations (lines 4 and 10 in Alg. 3 and lines 3 and 9 in Alg. 4, respectively). Before candidates of length $N + 1$ are evaluated probabilistically, estimation pruning can be applied to determine whether some combinations produced are not interesting for exact evaluation (line 6 in Alg. 3 and line 5 in Alg. 4, respectively). Each pruning strategy is presented next, in more detail.

Algorithm 3 *and_search_space_with_pruning*(PBK, PE)

```
1: $R_1 = generate\_rules\_of\_length\_one(PBK, PE)$
2: $T_1 = R_1 = prob\_evaluation(R_1, PBK, PE)$
3: $R_1 = R_N = AND\_prediction\_pruning(R_1)$
4: $R_{N+1} = combine\_rules\_with\_AND\_fitness\_pruning(R_1, R_N)$
5: while $R_{N+1} \neq \emptyset$ do
6:     $R_{N+1} = AND\_estimation\_pruning(R_{N+1})$
7:     $R_{N+1} = prob\_evaluation(R_{N+1}, PBK, PE)$
8:     $T_1 = T_1 \cup R_{N+1}$
9:     $R_N = AND\_prediction\_pruning(R_{N+1})$
10:    $R_{N+1} = combine\_rules\_with\_AND\_fitness\_pruning(R_1, R_N)$
11: return $T_1$
```

Algorithm 4 *or_search_space_with_pruning*(T_1, PBK, PE)

```
1: $T_{All} = T_1$
2: $T_1 = T_N = OR\_prediction\_pruning(T_1)$
3: $T_{N+1} = combine\_theories\_with\_OR\_fitness\_pruning(T_1, T_N)$
4: while $T_{N+1} \neq \emptyset$ do
5:     $T_{N+1} = OR\_estimation\_pruning(T_{N+1})$
6:     $T_{N+1} = prob\_evaluation(T_{N+1}, PBK, PE)$
7:     $T_{All} = T_{All} \cup T_{N+1}$
8:     $T_N = OR\_prediction\_pruning(T_{N+1})$
9:     $T_{N+1} = combine\_theories\_with\_OR\_fitness\_pruning(T_1, T_N)$
10: return $T_{All}$
```

3.1 Fitness Pruning

Fitness pruning consists of limiting the number of candidate theories in each iteration to a maximum size. The advantage of the fitness approach is that it imposes a polynomial limit on the exponential complexity of the search space, and therefore makes the runtime of the program shorter (Côrte-Real et al. 2015). Before starting a new iteration in the PILP algorithm, fitness pruning will select a limited number of candidate theories to be used, from all the possible combinations for that iteration.

This is done by defining two sets – *Primary* and *Secondary* – and, for each stage of the AND and OR search spaces, fitness pruning then selects a subset of theories for the two sets, and then new candidate theories are generated by combining all members from each set using the respective AND or OR operation. By default (i.e., with no pruning), when traversing the AND search space, the *Primary* set contains all rules of length one (with one literal) and the *Secondary* set is filled, in each iteration, with the set of rules generated in the previous iteration (initially the rules of length one, then the rules of length two, three and so on). Similarly, for the OR search space, the *Primary* set contains all rules generated in the AND search process (theories of length one) and the *Secondary* set is filled, in each iteration, with the set of theories generated in the previous iteration (initially the theories of length one, then the theories of length two, three and so on). With fitness pruning, the user defines a maximum size *PrimarySize* for the *Primary* set and another independent maximum size *SecondarySize* for the *Secondary* set, such that the total number of candidates *MaxSize* in each iteration is calculated by multiplying the maximum length of both sets as follows:

$$MaxSize = PrimarySize \times SecondarySize \tag{10}$$

Fully traversing the search space and evaluating all possible combinations of theories for each iteration corresponds to the particular case where $PrimarySize = SecondarySize = \infty$. Furthermore, in the case where $SecondarySize = \infty$, fitness pruning is equivalent to a beam search with beam width of *PrimarySize*.

The selection of the theories that will be part of the *Primary* and *Secondary* sets used for combination can be done according to different metrics, which are equivalent to the loss functions introduced in Section 2.2.3 – for instance, RMSE or PAcc. The selection is performed by ordering all theories according to the chosen metric and populating the sets with the top theories. In addition to the loss functions, a random metric, named RAND, can also be used. The RAND metric introduces a stochastic selection of theories in this process. Using deterministic ranks based on the quality of theories may in some cases result in sets of very similar theories due to the restricted nature of the selection. The RAND metric solves this issue by combining stochastically selected theories, as opposed to only *good ones* based on a deterministic metric. There may be cases where using a *weaker* theory in a combination may actually result in a final candidate theory of *better* quality.

Depending on the given metrics, fitness pruning can generate theories in a fully stochastic way, use best theories or create a heterogeneous mix. If the RAND metric is chosen, the selection process is distinct for each iteration.

Once the *Primary* and *Secondary* sets are populated, the theories for the next iteration will be generated from the members which are present in these sets only. Generated theories can still be excluded from the result if they do not comply with the language bias of the problem.

The fact that fitness pruning uses ranking metrics and fixed-size sets makes its complexity polynomially bound on user-defined parameters. As the datasets grow larger than a few examples, the probabilistic evaluation of theories represents the largest proportion of execution time, making the theory-generation procedure and other control operations negligible. If fitness pruning was not used to select sets, the overall number of probabilistic evaluations would be $\sum_{l=1}^{TMaxLength} \binom{|T_1|}{l}$, where *TMaxLength* is the maximum theory length and T_1 is the set of theories of length one.[2] It is easy to see that using sets of fixed size for each iteration, the number of probabilistic evaluations will be at most $|T_1| + (TMaxLength - 1) \times PrimarySize \times SecondarySize$, which is much less than all possible combinations of theories in the search space.

[2] Remember that a theory is evaluated probabilistically against the set of all examples.

3.2 Prediction Pruning

The aim of prediction pruning is to guide the search focusing on good candidate theories, and not allowing candidates which are below a threshold of quality to transition to the next iteration, particularly in scenarios where there are limited resources (Côrte-Real et al. 2018). Unlike fitness pruning, the decision on whether a candidate theory should be further explored is made based on the theory's individual prediction values for each example. The criterion to exclude theories depends on the search space being explored. For the AND search space, the combination of two rules r^a and r^b using logical conjunction results in a more specific rule $r^{a,b} = r^a \wedge r^b$. In the probabilistic setting, for each rule r and example i, the prediction value $p_i(r)$ is equal to the sum of the probabilities $P(\omega_n)$ of each world ω_n in the program in which $\omega_n \models r(i)$. 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 r^a and 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.

$$p_i(r^{a,b}) = \sum_{\omega_n \models r^{a,b}(i)} P(\omega_n) = \sum_{\omega_n \models r^a(i) \,\cap\, \omega_n \models r^b(i)} P(\omega_n) \tag{11}$$

In Eq. 12, set theory principles state that the intersection of the worlds which entail $r^a(i)$ and the worlds which entail $r^b(i)$ will always be equal or smaller than either set.

$$\begin{aligned} |\{\omega_n : \omega_n \models r^{a,b}(i)\}| &\le |\{\omega_n : \omega_n \models r^a(i)\}| \\ |\{\omega_n : \omega_n \models r^{a,b}(i)\}| &\le |\{\omega_n : \omega_n \models r^b(i)\}| \end{aligned} \tag{12}$$

Therefore, the prediction value $p_i(r)$ of a rule 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 and rule r, if the prediction value $p_i(r)$ is less than the example value e_i, then continuing to apply the AND operation can only result in distancing $p_i(r)$ 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. When a rule is excluded in the prediction pruning stage, the rule is still considered as a candidate for the best model (since it was already probabilistically evaluated), but it does not transition to the next iteration of the algorithm as a candidate.

A similar argument can be made for the OR operation and the generality of theories. The disjunctive combination $t^{a;b} = t^a \vee t^b$ of two theories t^a and t^b is true when either t^a, t^b or both t^a and t^b are true. In the probabilistic setting, the prediction value $p_i(t^{a;b})$ will be equal to the sum of the probabilities $P(\omega_n)$ of each world ω_n in the program in which $\omega_n \models t^a(i)$ or $\omega_n \models t^b(i)$, meaning the union of these sets of worlds. Similarly to rules, the prediction value of a more general theory for an example i can be defined in terms of the prediction values of more specific theories which compose it.

$$p_i(t^{a;b}) = \sum_{\omega_n \models t^{a;b}(i)} P(\omega_n) = \sum_{\omega_n \models t^a(i) \,\cup\, \omega_n \models t^b(i)} P(\omega_n) \tag{13}$$

In Eq. 14, set theory principles state that the union of the worlds which entail t^a and the worlds which entail t^b will always be equal or greater than either set.

$$\begin{aligned} |\{\omega_n : \omega_n \models t^{a;b}(i)\}| \geq |\{\omega_n : \omega_n \models t^a(i)\}| \\ |\{\omega_n : \omega_n \models t^{a;b}(i)\}| \geq |\{\omega_n : \omega_n \models t^b(i)\}| \end{aligned} \tag{14}$$

Therefore, the prediction value $p_i(t)$ of a theory t will be monotonically increasing with the application of the OR operation, since in each iteration in the OR search space the theories become more general. The monotonic increasing prediction values for each example of a given theory allows prediction pruning to be applied over previously evaluated theories to determine whether they are useless for further combination, given some criterion. Prediction pruning excludes theories whose prediction values for all examples suggest that the theory is already too general when compared to the example values. When a theory is excluded in the prediction pruning stage, the theory is still considered as a candidate for the best model, but it does not transition to the next iteration of the algorithm as a candidate.

3.3 Estimation Pruning

The aim of estimation pruning is to reduce execution time by preventing the probabilistic evaluation of theories which are not good candidates and would thus be pruned away at a later stage according to some criterion (Côrte-Real et al. 2016). Estimation pruning follows a similar approach to prediction pruning but instead of excluding theories based on their (previously calculated) probabilistic evaluation, estimation pruning excludes theories whose *estimation* is too specific (for the AND operation) or too general (for the OR operation) to be an interesting candidate according to a given loss function. The advantage of this approach lies in the fact that, by avoiding probabilistic evaluation on theories, the computational cost of traversing the search space is significantly reduced, since the main time component of this task is precisely the probabilistic evaluation of theories. However, the fact that estimations are used (instead of prediction values obtained from probabilistic evaluation) introduces a degree of uncertainty when applying the pruning criterion, since estimations are not necessarily always a good approximation for theory prediction values.

Estimation pruning estimates the prediction values of a theory based on theories of smaller length whose probabilistic evaluation has already been calculated at a previous iteration of the algorithm. For instance, in the case of rules, this can be achieved by decomposing a rule into two other rules of smaller length which are in themselves valid rules in the AND search space. If the probabilistic evaluations for the rules of smaller length are known, it is possible to estimate the range in which the prediction values of the composed rule will lie. In the OR search space, theories can also be decomposed in theories of shorter length, and estimations made based on the probabilistic evaluation of the composing theories. Similarly to prediction pruning, estimation pruning excludes combinations of theories whose estimated prediction values suggest that the resulting theory will be too specific (for the AND operation) or too general (for the OR operation).

Thus, estimation pruning aims to reduce execution time by ruling out theories that have poor estimations and exactly evaluating theories that have good estimations. The decision on whether a theory is discarded is made based on some criterion which is now directly applicable to the estimated probabilistic values in lieu of the exact prediction values of a theory[3]. 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. Since this pruning method is based on the estimation of a combination of theories, it can be considered *not safe* in the sense that it is possible to discard theories which might be better candidates than their estimations suggest.

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 without 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 theory is true in a world does not change the probability that another 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 t^a and t^b will lie. The bounds of that interval are determined by (i) the prediction values of the theories that are being combined, and (ii) the operation being used to combine the theories. The minimum and maximum boundaries of an estimation interval can be calculated by considering the theories' prediction values point wise, i.e., determine the minimum and maximum possible values for the combination of $p_i(t^a)$ and $p_i(t^b)$, for all examples. For each pair of prediction values, the possible resulting prediction value for the combination of t^a and t^b will vary monotonically from the minimum possible amount of overlap of the world sets (mutual exclusivity, corresponding to disjoint sets) to the maximum amount of overlap in the world sets (inclusiveness, corresponding to at least one of the world sets being a subset of the other).

In the case of logical conjunction, the minimum possible value for a combination of two predictions occurs if the sets of worlds for those predictions are as mutually exclusive as possible, i.e., when the amount of overlap is minimum. This occurs because the AND operation requires theories to be true in both sets of worlds. Therefore, the maximum boundary for the case of the AND operation happens if one of the sets of worlds is a subset of the other, meaning that one of the theories is included by the other. Equation 15 shows the expressions for the minimum and maximum boundaries when using the AND operation.

$$\begin{aligned} min\_bound_{AND} &= max(0, p_i(t^a) + p_i(t^b) - 1) \\ max\_bound_{AND} &= min(p_i(t^a), p_i(t^b)) \end{aligned} \tag{15}$$

[3] Another option would be to use a probabilistic inference method which determines the prediction values of a theory using an approximation, but this is outside of the scope of this work.

Conversely, for logical disjunction, the minimum and maximum boundaries correspond to the inclusive and mutually exclusive case, respectively. This is due to the fact that, for the combination of two theories to be true using the OR operation, only one of them needs to be true. Equation 16 shows the expressions used to calculate the minimum and maximum boundaries for the OR operation.

$$\begin{aligned} min\_bound_{OR} &= max(p_i(t^a), p_i(t^b)) \\ max\_bound_{OR} &= min(p_i(t^a) + p_i(t^b), 1) \end{aligned} \tag{16}$$

Based on the boundaries of the estimation interval, estimation pruning defines 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. The *center* estimator is the center of the estimation interval (halfway between *minimum* and *maximum*). The *independence* estimator assumes that theories t^a and t^b are independent and calculates the values of their combination accordingly. The *exclusion* estimator assumes that the theories t^a and t^b are as exclusive as possible (in the AND operation, the *exclusion* estimator is equal to the *minimum* estimator, and in the OR operation, it is equal to the *maximum* estimator). Equation 17 shows the expressions used to calculate the *center*, *independence* and *exclusion* estimator for the AND and OR operations.

$$\begin{aligned} estimator\_center_{AND} &= \frac{1}{2}(max(0, p_i(t^a) + p_i(t^b) - 1) + min(p_i(t^a), p_i(t^b))) \\ estimator\_center_{OR} &= \frac{1}{2}(max(p_i(t^a), p_i(t^b)) + min(p_i(t^a) + p_i(t^b), 1)) \\ estimator\_independence_{AND} &= p_i(t^a) \times p_i(t^b) \\ estimator\_independence_{OR} &= p_i(t^a) + p_i(t^b) - p_i(t^a) \times p_i(t^b) \\ estimator\_exclusion_{AND} &= max(0, p_i(t^a) + p_i(t^b) - 1) \\ estimator\_exclusion_{OR} &= min(p_i(t^a) + p_i(t^b), 1) \end{aligned} \tag{17}$$

3.4 Pruning Criteria

For both prediction and estimation pruning, a criterion is necessary to decide whether theories will be pruned away or not. Three criteria for deciding if a theory is too specify/general are described next (though different ones may be defined): a *hard criterion*, a *soft criterion* and a *safe criterion*. All criteria take into account the predictions (or estimations) of a theory $p_i(t)$ for the available examples, as well as the example values e_i themselves. Furthermore, the operation (AND/OR) under which the criterion is being applied must be taken into account.

The hard pruning criterion prunes a theory away if, in any example, the theory made a prediction (or has an estimation) that was more specific (for the AND operation) or more general (for the OR operation) than the annotated value for that example. Given a theory t, Eq. 18 and Eq. 19 present the expressions for the hard pruning criterion for the AND and OR search space, respectively.

$$\exists i : p_i(t) < e_i \tag{18}$$

$$\exists i : p_i(t) > e_i \tag{19}$$

The soft pruning criterion takes into account the theory's predictions (or estimations) 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 annotated values of the examples. 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. Given a theory t, Eq. 20 and Eq. 21 present the expressions for the soft pruning criterion for the AND and OR search space, respectively.

$$\sum_i \big(p_i(t) - e_i\big) < 0 \tag{20}$$

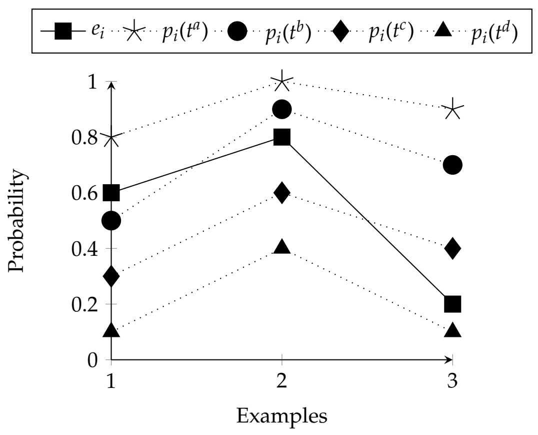
$$\sum_i \big(p_i(t) - e_i\big) > 0 \tag{21}$$

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. Figure 1 illustrates these concepts for a PILP setting with three examples and four theories. For each example i, the example value e_i (squares in black) and four predictions (or estimations) of theories t^a, t^b, t^c and t^d are plotted.

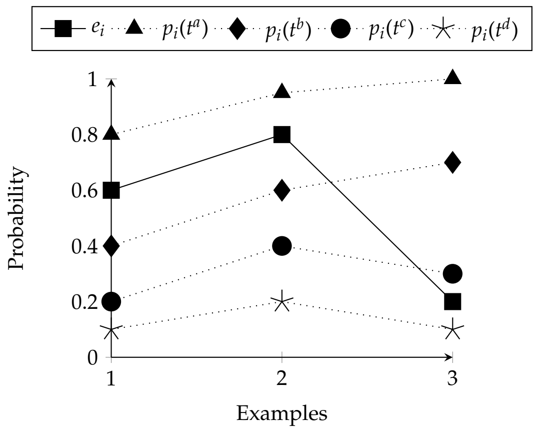
For the AND operation in Fig. 1(a), the safe pruning criterion would prune away theory t^d (red triangles) because, for every example, its prediction values are lower than the example values. The soft criterion prunes t^c (diamonds) and t^d (triangles) because their prediction values are overall lower than the example values. Finally, the hard pruning criterion would prune away all theories except t^a (for example, t^b is pruned away because its prediction for e_1 is lower than the example value). All pruning criteria would keep t_a because all its prediction (or estimation) values are higher than their respective example value.

An analogous reasoning can be made for the OR operation in Fig. 1(b). The safe pruning criterion would prune t^a (red triangles) because all its prediction values are higher than the example values. The soft pruning criterion would prune both t^a (triangles) and t^b (diamonds) because their prediction values are overall higher than the example values. The hard pruning criterion would prune away t^a (triangles), t^b (diamonds) and t^c (circles). No pruning criterion would prune away t^d because all its prediction (or estimation) values are lower than the example values.

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.



(a) Pruning criteria for the AND search space



(b) Pruning criteria for the OR search space

Fig. 1 Pruning criteria for AND and OR search spaces. The x-axis contains three examples $i = \{1, 2, 3\}$ and the y-axis represents probabilistic values ranging from 0 to 1. For each example i, example values e_i are depicted as black squares and four theories' prediction values $p_i(t^a)$, $p_i(t^b)$, $p_i(t^c)$ and $p_i(t^d)$ are depicted in different markers.

3.5 Safeness

One concern of using pruning strategies is whether the best theories can be lost due to the use of pruning. A *safe* pruning strategy can thus be defined as a strategy which only prunes away candidate theories which are assuredly not the best theory. In order to guarantee safeness, the pruning strategy must use the same evaluation ranking that is used during search space traversal to determine the best theory (Côrte-Real et al. 2018). This happens because the search space traversal evaluation metric defines a ranking for the utility of candidate theories and therefore also defines the first theory of that ranking, i.e., the best theory.

In general, fitness pruning can not be guaranteed to be a safe pruning strategy. This is because the candidate theories of iteration $j + 1$ are combinations of the most promising candidates of iteration j, according to some ranking. Even if this ranking uses the same metric as the evaluation function, there is no guarantee that a combination of two candidate theories which were not included in the populations cannot be a better theory than all the candidate theories generated from the population members. This can happen when two theories whose rankings are low combine to form a theory with much higher ranking. One way to ensure that fitness pruning is a safe pruning strategy would be to include all possible theories in the populations, which is equivalent to exhaustively traversing the search space by definition.

Estimation pruning is also not a safe pruning strategy in general. In the case of this pruning strategy, theories are pruned away based on an estimation of their prediction values (i.e., before evaluation). If the estimate does not correspond exactly to the predictions of the theory for every example, the decision on whether to prune or keep the theory is made with inaccurate information. This therefore makes it impossible to guarantee that the best theory will not be pruned away in a setting where its estimated predictions are not as good as the actual prediction values.

Prediction pruning, on the other hand, can be a safe pruning strategy given the right conditions. This pruning strategy decides, for each evaluated theory, whether candidate theories should be generated from it. In the case of AND operation, candidate theories generated from another theory can only be more specific than the original theory. Similarly, candidate theories generated from a theory using the OR operation can only be more general than the original theory. Therefore, if the original theory is already too specific (for the AND operation) or too general (for the OR operation), candidate theories generated from it can only be even more specific (AND operation) or general (OR operation). This fact guarantees that under certain conditions of specificity/generality of the original theory, prediction pruning can be a safe pruning strategy. In addition, prediction pruning can only guarantee safeness if it is not used in combination with another (unsafe) pruning strategy.

3.6 Related Work

The most common pruning strategy used during the traversal of PILP spaces is fitness pruning (or beam search). Beam search is a particular case of fitness pruning, and it is used by the PILP systems SLIPCOVER and ProbFOIL+. The SkILL PILP system uses fitness pruning for both the AND and the OR operations, allowing for a greater control of the beam.

The prediction pruning strategy aims at focusing the search on good candidate theories, and not allowing candidates which are below a threshold of quality to transition to the next iteration. The SkILL system supports this pruning strategy for both the AND and the OR operation, using safe, soft and hard criteria. ProbFOIL+ also supports these criteria, but only for the AND operation. Additionally, ProbFOIL+ includes a refinement rejection criterion which is a combination of three metrics that are calculated for each possible refinement: local score, global score and significance. This rejection criterion can also be seen as a form of prediction pruning for the AND operation, since once a specification of a theory is calculated, further refinements of it will not be pursued unless it passes the rejection criterion. Furthermore, the learning process of SLIPCOVER also calculates possible refinements and their respective log-likelihood for each rule, which again can be seen as a form of prediction pruning with a different criterion.

Estimation pruning avoids the costly operation of exact probabilistic evaluation of theories by estimating the value of a combination of theories based on its sub-theories. This strategy is supported by both the SkILL and the ProbFOIL+ systems, even though ProbFOIL+ only supports the case of the AND operation.

4 Experiments

This experimental section analyses the effect of using a combination of pruning strategies in the SkILL (Côrte-Real et al. 2015) and ProbFOIL+ (De Raedt et al. 2015) PILP systems, which are both based on the ProbLog language. The SkILL system runs on top of the Yap Prolog system (Costa et al. 2012), uses TopLog (Muggleton et al. 2008) as the basis rules generator 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, unless otherwise indicated. The ProbFOIL+ system 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.

The performance of the pruning strategies was analysed using three different datasets: **metabolism**, **athletes** and **breast cancer**. All experiments use five-fold cross validation and results presented are the average values for all folds, unless otherwise indicated. Table 2 summarises the characteristics of the datasets.

Table 2 Dataset characteristics: number of probabilistic examples; number of facts in the PBK and proportion of probabilistic facts in brackets; number of examples in the train set and proportion of the dataset in brackets; and number of examples in the test set and proportion of the dataset in brackets.

Dataset	Examples	PBK		Train Size		Test Size	
metabolism	230	7000	(46%)	184	(70%)	46	(30%)
athletes	721	4294	(100%)	576	(70%)	144	(30%)
breast cancer	130	13400	(3%)	104	(80%)	26	(20%)

The **metabolism** dataset consists of an adaptation of the dataset originally from the 2001 KDD Cup Challenge[4]. 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 $\mathtt{p}_{strength}$`::interaction(gene1, gene2, type)`, where $\mathtt{p}_{strength}$ was calculated from strength interactions as follows:

$$p_{strength} = \frac{strength - min_{strength}}{max_{strength} - min_{strength}} \tag{22}$$

[4] `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[5]. 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. 21 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 Fitness Pruning

This section analyses the effect of fitness pruning on probabilistic accuracy (PAcc) and execution time using three different configurations for the primary/secondary populations: 25/5, 25/10 and 25/20. The same configuration was used for both the AND and OR search spaces and the PAcc metric was used for ranking both populations. For example, the size 5 in 25/5 corresponds to the size of the secondary population, i.e., in each iteration *j*, five theories (or rules) of length *j* are chosen to populate this set according to the ranking metric PAcc. The first size (25 in all cases) is the fixed number of theories (or rules) of length one that are combined against the secondary set. This corresponds to using beam sizes of 125, 250 and 500 candidates, respectively. These sizes were chosen based on populations that would make it feasible to complete executions in the systems we tested.

Table 3 shows the running time in seconds, total number of evaluations performed and probabilistic accuracy for datasets **metabolism**, **athletes** and **breast cancer** using SkILL with the three population sizes described above and ProbFOIL+ using a combination of beam search in the AND-space (with the default beam size: 5) and greedy search in the OR-space.

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

Table 3 Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy on the test set using SkILL with varying population sizes (same sizes used for AND and OR operation) and ProbFOIL+ with greedy beam search, for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets. Execution times between systems are not comparable.

	SkILL 25/5	SkILL 25/10	SkILL 25/20	ProbFOIL+
	Execution Time			
metabolism	2065 (111)	2552 (85)	3353 (204)	2008 (2016)
athletes	1715 (25)	3413 (469)	4610 (79)	57 (5)
breast cancer	779 (10)	1102 (76)	1449 (63)	3890 (339)
	# Evaluations			
metabolism	1450 (43)	1683 (45)	2151 (44)	3734 (2328)
athletes	679 (6)	1142 (16)	1852 (25)	201 (43)
breast cancer	326 (38)	647 (66)	1235 (68)	24290 (851)
	Probabilistic Accuracy			
metabolism	0.67 (0.06)	0.67 (0.06)	0.67 (0.05)	0.51 (0.04)
athletes	0.95 (0.01)	0.95 (0.01)	0.95 (0.01)	0.80 (0.01)
breast cancer	0.85 (0.03)	0.85 (0.03)	0.86 (0.04)	0.85 (0.01)

Fitness pruning establishes a bound on the number of AND and OR evaluations performed during PILP search space traversal, reducing them from an exponential number to a polynomially bound one. However, there is always a constant component that is related to the number of rules of length one, which must be evaluated regardless of the pruning settings (1181 rules for **metabolism**, 53 rules for **athletes** and 25 rules for **breast cancer**). In Table 3, the effect of this component in SkILL is evident on the execution time and evaluations presented, since they do not reduce linearly with the number of combinations. Furthermore, as the number of combinations grows larger, there may be cases where there are not enough rules or theories of a given length to fill the population sets. This is more likely to happen during AND search since the language bias may restrict how the rule search space can grow.

Even though fitness pruning only traverses part of the search space, the results on Table 3 show that it does not sacrifice probabilistic accuracy. This is due to the fact that rules and theories are ranked by ausing the same performance-related metric (PAcc), and so combinations are made from the more accurate members generated in each iteration (even though other ranking metrics can be used to select theories, e.g., random selection to include weaker candidates). Nonetheless, as the probabilistic accuracy does not decrease when the population sizes are reduced, it can be concluded that PAcc is effective in keeping a good population of candidate theories.

This procedure is different in the ProbFOIL+ system, where a greedy beam search is used. For that reason, there are cases in which ProbFOIL+ performs significantly more/less evaluations than SkILL (more evaluations for **metabolism** and **breast cancer**, less for **athletes**). However, ProbFOIL+'s probabilistic accuracy is significantly less in two of the datasets (**metabolism** and **athletes**), as is to be expected from the reduced number of evaluations, which in turn results in shorter execution times.

4.2 Estimation Pruning

This section analyses the performance of estimation pruning in the SkILL and ProbFOIL+ systems using the independence estimator as it is was shown to have good overall performance. For a detailed comparison of different estimators please refer to Côrte-Real et al. (2016). In the ProbFOIL+ system, estimation pruning with the independence estimator was added only for the AND operation, since in ProbFOIL+ the OR search space forms a theory by adding the best rule found in the AND search space and then adjusting the examples accordingly. As baseline for the SkILL experiments, setting 25/20 fitness pruning is taken as exhaustive experiments without fitness pruning were computationally infeasible. Likewise, for ProbFOIL+, a beam of size 5 is used.

Table 4 shows the running time in seconds, total number of evaluations performed and probabilistic accuracy for datasets **metabolism**, **athletes** and **breast cancer** using varying estimation pruning criteria (but always the independence estimator) for SkILL and ProbFOIL+. The pruning criteria are reported as a tuple where the first value 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, Sx stands for soft AND pruning and no OR pruning.

Results in Table 4 indicate that, in most cases, the use of estimation pruning can reduce the runtime of the experiments when compared to using no estimation pruning (baseline columns). In the cases where this does not happen (for instance, in the **breast cancer** dataset using ProbFOIL+), the number of probabilistic evaluations is still reduced, but the theories that are being evaluated using estimation pruning are more complex, and so it takes much longer to evaluate them. Furthermore, there is no significant variation in accuracy when applying estimation pruning using the independence estimator, for any of the settings presented in Table 4.

4.3 Prediction Pruning

This section assesses the quality of candidate theories 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 fitness pruning in SkILL or beam search in ProbFOIL+. Each theory in the PILP search space can be thought of as a predictor. Since prediction pruning removes theories from the search space, the distribution of the remaining candidate theories can change. For a detailed analysis of this effect refer to Côrte-Real et al. (2018). This experiment uses the same baseline as the previous experiment (SkILL with fitness pruning 25/20 and ProbFOIL+ with a beam of size 5). Table 5 shows the running time in seconds, total number of evaluations performed and probabilistic accuracy for datasets **metabolism**, **athletes** and **breast cancer** using prediction pruning for the AND search space, given the same limitation of resources as the baseline case.

Results in Table 5 show that applying the soft or hard pruning criteria (in the AND search space) leads to clear improvements in probabilistic accuracy for ProbFOIL+ and does not lead to degradation in SkILL, when compared to the baseline results. Safe pruning has no effect on these datasets because its pruning power is too limited. The effect of prediction pruning is more evident for ProbFOIL+ because it selects fewer candidates in each iteration, when compared to SkILL's primary

Table 4 Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy on the test set using varying pruning criteria and independence estimator for SkILL and ProbFOIL+ (only for the AND operation), for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets. Execution times between systems are not comparable. The baselines are taken from Table 3 with setting 25/20 for SkILL.

(a) SkILL

	Baseline	**Sx**	**Hx**	**xS**	**xH**
	Execution Time (s)				
metabolism	3353 (204)	1719 (328)	1753 (319)	2873 (828)	1422 (194)
athletes	4610 (79)	339 (28)	337 (27)	536 (73)	520 (27)
breast cancer	1449 (63)	53 (8)	220 (20)	199 (8)	220 (20)
	# Evaluations				
metabolism	2151 (44)	3312 (76)	3312 (76)	7053 (262)	5260 (280)
athletes	1852 (25)	1133 (90)	1133 (90)	2483 (271)	2414 (94)
breast cancer	1235 (68)	731 (0)	1919 (0)	1919 (0)	1919 (0)
	Probabilistic Accuracy				
metabolism	0.67 (0.05)	0.66 (0.05)	0.66 (0.05)	0.66 (0.05)	0.66 (0.05)
athletes	0.95 (0.01)	0.90 (0.12)	0.90 (0.12)	0.95 (0.00)	0.95 (0.00)
breast cancer	0.86 (0.04)	0.78 (0.31)	0.72 (0.36)	0.72 (0.36)	0.72 (0.36)

(b) ProbFOIL+

	Baseline	**Sx**	**Hx**
	Execution Time (s)		
metabolism	2008 (2016)	1336 (798)	313 (83)
athletes	57 (5)	22 (18)	18 (1)
breast cancer	3890 (339)	6576 (1177)	4515 (431)
	# Evaluations		
metabolism	3734 (2328)	3531 (2959)	304 (76)
athletes	201 (43)	29 (3)	20 (2)
breast cancer	24290 (851)	8325 (286)	699 (30)
	Probabilistic Accuracy		
metabolism	0.51 (0.04)	0.51 (0.04)	0.51 (0.01)
athletes	0.80 (0.01)	0.80 (0.01)	0.80 (0.01)
breast cancer	0.85 (0.01)	0.87 (0.01)	0.87 (0.02)

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.

Table 5 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.

Table 5 Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy on the test set using SkILL and ProbFOIL+ with prediction pruning for the AND search space, for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets. Execution times between systems are not comparable. The baselines are taken from Table 3 with setting 25/20 for SkILL.

(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)
		# 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)
		# 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)

4.4 Combining Pruning Strategies

Fitness pruning focuses on limiting the number of candidates. Estimation pruning is aimed at eliminating probabilistic evaluations, therefore reducing the execution time. Conversely, prediction pruning focuses on ensuring that all rules or theories that proceed to the next iteration are viable candidates for combination, thus focusing on the quality of the theories combined.

Pruning options for prediction and estimation pruning allow parameters to be set differently for the AND and OR operations, which results in 81 possible combinations for those settings. Since the aim of this experimental section is to showcase the combined effect of the pruning operations, results are only presented for the cases where the same pruning setting is used for the AND and Or search spaces. This results in 9 combinations for a fixed fitness pruning size selection. In what follows, only the 25/5 fitness pruning setting is reported.

Table 6 presents the pruning configurations for each setting, as well as the predictive accuracy on the test set for SkILL. Table 7 presents the number of probabilistic evaluations performed and the execution time in seconds on the test set for SkILL. All settings were tested for datasets **metabolism**, **athletes** and **breast cancer**, using 25/5 fitness pruning for both the AND and OR operation.

Table 6 Pruning configuration for each setting (EA, EO, PA and PO stand for AND estimation pruning, OR estimation pruning, AND prediction pruning and OR prediction pruning, respectively) and probabilistic accuracy on the test set using SkILL, for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets.

	Configuration				**Probabilistic Accuracy**		
Setting	**EA**	**EO**	**PA**	**PO**	**metabolism**	**athletes**	**breast cancer**
Baseline	x	x	x	x	0.67 (0.06)	0.95 (0.01)	0.85 (0.03)
(E=x, P=S)	x	x	S	S	0.67 (0.07)	0.95 (0.01)	0.84 (0.03)
(E=x, P=H)	x	x	H	H	0.67 (0.06)	0.95 (0.01)	0.86 (0.04)
(E=S, P=x)	S	S	x	x	0.67 (0.07)	0.95 (0.01)	0.86 (0.03)
(E=S, P=S)	S	S	S	S	0.67 (0.07)	0.95 (0.01)	0.86 (0.03)
(E=S, P=H)	S	S	H	H	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=x)	H	H	x	x	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=S)	H	H	S	S	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=H)	H	H	H	H	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)

Table 7 Number of probabilistic evaluations and execution time in seconds on the test set using SkILL, for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets.

Setting	**# Evaluations**			**Execution Time**		
	metabolism	**athletes**	**breast cancer**	**metabolism**	**athletes**	**breast cancer**
Baseline	1450 (43)	679 (6)	326 (38)	2065 (111)	1715 (25)	779 (10)
(E=x, P=S)	1321 (107)	309 (3)	333 (26)	7867 (590)	794 (10)	516 (173)
(E=x, P=H)	1227 (142)	292 (24)	36 (7)	2518 (817)	749 (57)	86 (3)
(E=S, P=x)	1412 (45)	287 (3)	84 (38)	1917 (57)	753 (10)	170 (109)
(E=S, P=S)	1358 (96)	288 (3)	67 (35)	2012 (37)	741 (8)	159 (111)
(E=S, P=H)	1227 (142)	247 (32)	25 (4)	2172 (489)	639 (75)	82 (2)
(E=H, P=x)	1181 (47)	92 (7)	26 (6)	1476 (61)	281 (16)	95 (4)
(E=H, P=S)	1181 (47)	105 (7)	26 (6)	1490 (90)	299 (16)	92 (4)
(E=H, P=H)	1228 (142)	194 (23)	25 (5)	1955 (77)	510 (53)	82 (1)

Results in Table 6 show that the proposed pruning strategies can maintain (and in the case of the medical dataset increase) the probabilistic accuracy of the best theory found, on the test set, even though the number of evaluations performed (columns 2–4 on Table 7) is greatly reduced, especially in the case of the **athletes** and **breast cancer** datasets. Despite the fact that probabilistic accuracies in the test set are similar in most cases, the best theory may differ, and thus the information content of different pruning settings w.r.t. the logic literals is different for each scenario.

Similarly to fitness pruning, the reduction in the number of evaluations correlates with the shorter execution time for all settings where only estimation pruning is used (lines 4 and 7 in Table 7). This is to be expected, since estimation pruning only prevents probabilistic evaluations from being performed and has no effect on the theories that are combined to traverse the search space. Introducing prediction pruning, however, can have a negative effect both on the execution time and on the number of evaluations performed. This is particularly evident for the **metabolism** dataset in the soft prediction pruning setting (line 2 and column 5 in Table 7).

The fact that prediction pruning is introduced alters the candidate theories for the next iteration, meaning that when fitness pruning selects populations, the selected theories are a different set than they would be if there was no prediction pruning. This selection leads to an earlier collection of more complex to compute theories and less likely to be discarded from estimation pruning. This effect is also visible in the **metabolism** and **athletes** dataset, when there is a transition from soft to hard prediction pruning while using hard estimation pruning (lines 8 and 9 in Table 7).

Table 8 presents the number of rules and theories pruned during the estimation and evaluation pruning for the 9 combinations being considered for the case of the **athletes** dataset and 25/5 fitness pruning. Other datasets and fitness pruning options present a similar relation between pruned rules and theories.

Table 8 Number of rules and theories pruned during estimation and prediction pruning for the **athletes** dataset. Standard deviation is presented in brackets.

Setting	# Pruned Rules		# Pruned Theories	
	Estimation	Prediction	Estimation	Prediction
Baseline	–	–	–	–
(E=x, P=S)	–	73 (3)	–	32 (4)
(E=x, P=H)	–	73 (3)	–	43 (3)
(E=S, P=x)	597 (9)	–	0 (0)	–
(E=S, P=S)	24 (0)	52 (3)	0 (0)	14 (2)
(E=S, P=H)	24 (0)	52 (3)	0 (0)	29 (2)
(E=H, P=x)	598 (8)	–	194 (7)	–
(E=H, P=S)	24 (0)	52 (3)	181 (7)	3 (0)
(E=H, P=H)	24 (0)	52 (3)	54 (27)	29 (2)

Results in Table 8 show that there is a trade-off between estimation and prediction pruning: as the amount of prediction pruning increases, estimation pruning is not able to prune away as many combinations, since the candidate theories composing those populations are better suited for combination (this is visible in lines 5, 6, 8 and 9 in Table 8). Because fitness pruning limits the amount of theories which are selected for combination at each iteration, the fact that better candidates are selected each time (caused by prediction pruning) has an impact on the estimation pruning's ability to avoid probabilistic evaluations.

The dependency between estimation and prediction pruning does not impact the quality of the final theory obtained, and so the decision about the best estimation configuration will depend on the execution time alone. Results in Table 7 show that the two fastest pruning settings are obtained when hard estimation pruning is coupled with off or soft prediction pruning. Settings with only prediction pruning enabled underperform timewise when compared to other settings. The second fastest pruning configuration is to use hard estimation and prediction pruning, which is even slightly faster in **breast cancer**. These results indicate that the best compromise regarding estimation and prediction pruning is to use a combination of both in order to obtain a faster execution time and the maximum potential benefit out of combining relatively small fitness pruning populations.

5 PILP as an Explainable Classifier

In this section, we explore the usefulness of PILP representations and PILP classifiers through the **breast cancer** dataset. Breast cancer is one of the most common forms of cancer. Mammograms are the most commonly used technique to detect patients at risk. Image-guided core needle biopsy of the breast is then performed to decide on surgery. Biopsy is a necessary, but also aggressive, high-stakes procedure. The assessment of malignancy risk following breast core biopsy is imperfect and biopsies can be *non-definitive* in 5-15% of cases Berg et al. (1996); Brancato et al. (2012); Burbank (1997); Gonçalves et al. (2011); Liberman (2000); Liberman et al. (2000). A non-definitive result means that the chance of malignancy remains high due to possible sampling error (i.e., the obtained biopsy is not representative of the suspicious finding), for which surgical excisional biopsy or aggressive radiologic follow-up is proposed. Non-definitive biopsies may therefore result in missed breast cancers (false negatives) and unnecessary interventions (false positives).

The dataset used in this Section is the same used to test our pruning strategies. They contain anonymised data from 130 biopsies dating from January 2006 to December 2011, collected from the School of Medicine and Public Health of the University of Wisconsin-Madison. For all cases, several sources of variables were systematically collected including variables related to demographic and historical patient information (age, personal history, family history, etc.), mammographic BI-RADS descriptors (like mass shape, mass margins or calcifications), pathological information after biopsy (type of disease, if it is incidental or not, number of foci, and so on), biopsy procedure information (such as needle gauge, type of procedure), and other relevant facts about the patient.

Probabilistic data was then added to (i) the Probabilistic Examples (PE) and (ii) the Probabilistic Background Knowledge (PBK). The confidence in malignancy for each case (before excision) is associated with the target predicate `is_malignant/1`. The chance of malignancy is an empirical confidence value assigned by a multidisciplinary group of physicians. A high probability indicates the team of physicians thinks the case is most likely malignant. A low probability indicates the case is most likely benign. A sample of the PE is presented in Fig. 2.

```
example(is_malignant(case1), 0.10).
example(is_malignant(case2), 0.15).
example(is_malignant(case3), 0.01).
```

Fig. 2 Sample of the target predicate `is_malignant/1`

The domain knowledge incorporated in the PBK was taken from the literature on breast cancer. For example, it is well known among radiology experts in mammography that if a mass has a spiculated margin, the probability that the associated finding is malignant is around 90%. The same kind of information is available in the literature for mass shape or mass density (all part of the BIRADS terms). Figures 3, 4, and 5 show how these variables are encoded in the PBK.

Figure 3 encodes the probabilistic information regarding mass shape obtained from the literature. There are three possible independent rules, each one applicable to

```
0.05::feature_shape(Case) :-
   mass(Case, Mass),
   mass_shape(Mass, oval).

0.50::feature_shape(Case) :-
   mass(Case, Mass),
   mass_shape(Mass, round).

0.50::feature_shape(Case) :-
   mass(Case, Mass),
   mass_shape(Mass, irregular).
```

Fig. 3 Probabilistic information from the literature regarding mass shape

```
0.02::feature_margin(Case) :-
   mass(Case, Mass),
   mass_margin(Mass, circumscribed).

0.20::feature_margin(Case) :-
   mass(Case, Mass),
   mass_margin(Mass, indistinct).

0.70::feature_margin(Case) :-
   mass(Case, Mass),
   mass_margin(Mass, microlobulated).

0.90::feature_margin(Case) :-
   mass(Case, Mass),
   mass_margin(Mass, spiculated).
```

Fig. 4 Probabilistic information from the literature regarding mass margin

```
0.05::density(low);
0.10::density(equal);
0.50::density(high).

feature_density(Case) :-
   mass(Case, Mass),
   mass_density(Mass, MassDensity),
   density(MassDensity).
```

Fig. 5 Probabilistic information from the literature regarding mass density

a particular kind of shape (oval, round, or irregular). The probability value annotated in each rule is the frequency with which a mass whose shape is of that type is malignant. This means that a finding may have simultaneously an oval and round mass shape, for instance. Given that possible world semantics is used to encode these rules, the probability of two rules occurring simultaneously is given by the product of their probabilities. For instance, the probability that a mass has both an oval and round shape is equal to $0.05 \times 0.50 = 0.025$. Similarly, Fig. 4 also encodes independent rules, each for a characteristic of the mass margin. Figure 5 differs in that it encodes three mutually exclusive possibilities for the mass density: low, equal, or high (`density/1` fact), defined in the top three lines (";" is used for disjunction).

The density rule is then constructed based on the mutual exclusivity introduced by the `density/1` fact.

```
is_malignant(Case) :-
  feature_margin(Case).
is_malignant(Case) :-
  feature_shape(Case),
  feature_density(Case).
```

Fig. 6 A PILP model for the target predicate `is_malignant/1`

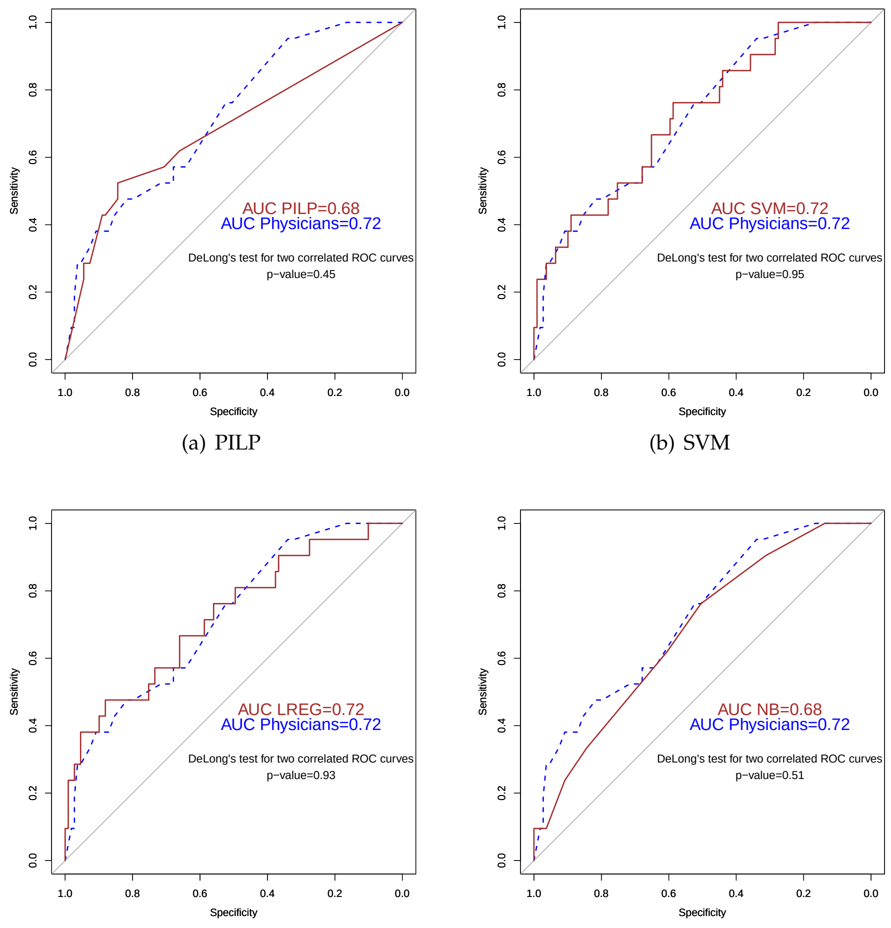
Figure 6 presents an example of a rule for the target predicate `is_malignant/1`, which explains malignancy in terms of margin OR mass shape and density. Since the rules in this explanation are composed of probabilistic literals (`feature_margin/1`, `feature_shape/1`, and `feature_density/1`), the target predicate `is_malignant/1` will also predict a probabilistic value ranging from 0 to 1, even though this is not made explicit in the PILP model. This probability output is computed using the *possible world semantics* (Kimmig et al. 2011), and it takes into account the mutual dependency between all the probabilistic literals in the model.

Next, we show performance results for the PILP models produced by SkILL on this dataset. Exhaustive search was used. 130 train and tune sets were used to perform leave-one-out cross validation on the dataset, and the predicted values for the test examples were recorded.

In addition to the PILP model described earlier, three other methods were used to compare against PILP in terms of predictive accuracy, using default parameters: a Support Vector Machine (SVM), a Linear Regression (LREG), and a Naive Bayes classifier (NB). The *scikit-learn* python library (Pedregosa et al. 2011) was used. Since these data contain several categorical features, it was necessary to transform them into numerical features to be able to apply these methods. As such, each possible label was first encoded as an integer. Once this was done, each feature was transformed in several auxiliary features, each one of them binary and regarding only one of the labels. This methodology was used to prevent the integer values corresponding to the labels of a feature from being interpreted as being ordered, which would not represent the independence between the labels accurately. Once these operations were performed over all categorical features, a scaler (standardization) was applied so as to reduce all features to mean 0 and unit variance. The predictions for each method were then obtained.

Figure 5 presents the ROC curves for the malignant class and four methods tested: PILP, SVM, LREG and NB. Each subfigure shows the ROC of the physicians' predictions (blue dashed line) and the ROC of a method (brown solid line), both against the ground truth (confirmed malignancy or benignity of a tumour after excision). Each figure also presents the respective AUCs and the p-value found using DeLong's test for comparing both curves plotted.

The difference between the curves was found to be statistically not significant, thus implying that all methods are statistically indistinguishable from a physician when predicting the degree of malignancy of a patient in this dataset. Additionally, PILP reduces more the false negative rate than the other methods, which is relevant to the breast cancer domain.



(a) PILP (b) SVM

(c) LREG (d) NB

Fig. 7 ROC curves, AUCs and p-values for PILP, SVM, LREG and NB methods

Learned rules are shown in Fig. 8. The first one contains a probabilistic fact related to one mammography descriptor: the shape of a mass. In medical literature, irregular shapes or spiculated margins indicate higher risk of malignancy. This is captured by the system, as well as other features such as no observed increase in mass size and an ultrasound core needle biopsy type. Similarly, the other two rules present features that are evidence of higher risk of malignancy, such as asymmetry, the gauge of the needle and a possible displacement of the needle (offset) during biopsy which can contribute as a confounding factor.

```
is_malignant(Case):-
   biopsyProcedure(Case,usCore),
   changes_Sizeinc(Case,missing),
   feature_shape(Case).
is_malignant(Case):-
   assoFinding(Case,asymmetry),
   breastDensity(Case,scatteredFDensities),
   vacuumAssisted(Case,yes).
is_malignant(Case):-
   needleGauge(Case,9),
   offset(Case,14),
   vacuumAssisted(Case,yes).
```

Fig. 8 Theory extracted for physician's mental models

6 Conclusion

PILP has generated a significant interest from the research community due to its remarkable suitability to model both complex relationships between data and uncertainty. Unfortunately, this expressiveness comes at the cost of an exponentially large search space and an exponential theory evaluation time. Therefore, PILP systems rely on the ability to prune the search space using pruning strategies.

This work categorizes existing PILP pruning strategies in three different types: fitness, estimation and prediction pruning. Fitness pruning bounds the maximum number of probabilistic evaluations according to user-defined parameters, which allows for a polynomially bound algorithm w.r.t. probabilistic evaluations as opposed to an exponential one. Estimation pruning estimates the prediction of a combination of theories and, based on that estimation, it prunes away combinations which would not result in an accurate theory. This method can reduce the execution time of the PILP algorithm since it prevents unnecessary probabilistic evaluations, which represent the core of the runtime. Prediction pruning ensures that the candidate theories for the next iteration can all benefit from being combined using a given operation (AND or OR). The aim of this pruning is to guarantee that the quality of the candidate theories available for combination is increased w.r.t the operation that will be performed on them.

These pruning strategies were tested in the SkILL and ProbFOIL+ systems, using three different datasets: an adaptation of the classic ILP dataset metabolism; a knowledge base gathered from the web; and a breast cancer medical dataset. From the results of those experiments, it can be concluded that the pruning strategies maintain or increase probabilistic accuracy in all cases, while generally achieving a reduction in execution time. Further experimental analysis shows that there is a dependency between estimation and prediction pruning, which results in the best pruning settings enabling both of these strategies so as to maintain quality of theories while reducing execution time to a minimum. It was also shown that fitness pruning can maintain predictive quality even for relatively small populations, and so the best overall pruning strategy consists of enabling all three pruning strategies simultaneously.

We also explored the power and explainability of PILP on our medical dataset. We have shown how to represent data in this domain using a probabilistic logic programming language notation and demonstrated that PILP can learn interpretable rules with the same prediction power of an expert in the domain of breast cancer.

Further work on this topic includes testing these pruning strategies also in a deterministic setting, adding a parameter-learning stage to the presented AND and OR pipelines, using the probabilistic information to select one path through the search space, as opposed to pruning away theories as is shown in this work, and to autonomously select the best parameters for pruning.

Acknowledgments

This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020. Joana Côrte-Real is financed by the FCT grant SFRH/BD/52235/2013.

References

Bellodi, E. and Riguzzi, F. (2012), Learning the structure of probabilistic logic programs, *in* 'Inductive Logic Programming', Springer, pp. 61–75.

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

Berg, W. A., Hruban, R. H., Kumar, D., Singh, H. R., Brem, R. F. and Gatewood, O. M. (1996), 'Lessons from mammographic histopathologic correlation of large-core needle breast biopsy', *Radiographics* **16**(5), 1111–1130.

Brancato, B., Crocetti, E., Bianchi, S., Catarzi, S., Risso, G. G., Bulgaresi, P., Piscioli, F., Scialpi, M., Ciatto, S. and Houssami, N. (2012), 'Accuracy of needle biopsy of breast lesions visible on ultrasound: audit of fine needle versus core needle biopsy in 3233 consecutive samplings with ascertained outcomes', *Breast* **21**(4), 449–454.

Burbank, F. (1997), 'Stereotactic breast biopsy: comparison of 14- and 11-gauge mammotome probe performance and complication rates', *Am Surg.* **63**(11), 988–995.

Cook, D. and Holder, L. (2006), *Mining Graph Data*, John Wiley & Sons.

Côrte-Real, J., Dries, A., Dutra, I. and Rocha, R. (2018), Improving Candidate Quality of Probabilistic Logic Models, *in* A. dal Palù and P. Tarau, eds, 'Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018)', Oxford, UK.

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

Côrte-Real, J., Mantadelis, T., Dutra, I., Rocha, R. and Burnside, E. (2015), SkILL - a Stochastic Inductive Logic Learner, *in* 'International Conference on Machine Learning and Applications', Miami, Florida, USA, pp. –.

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

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

De Raedt, L. and Kersting, K. (2004), Probabilistic inductive logic programming, *in* 'International Conference on Algorithmic Learning Theory', Springer, pp. 19–36.

De Raedt, L. and Kimmig, A. (2015), 'Probabilistic (logic) programming concepts', *Machine Learning* **100**(1), 5–47.

De Raedt, L. and Thon, I. (2011), Probabilistic Rule Learning, *in* 'Inductive Logic Programming', Springer, pp. 47–58.

Džeroski, S. (2010), *Relational Data Mining*, Springer.

Džeroski, S., De Raedt, L. and Driessens, K. (2001), 'Relational Reinforcement Learning', *Machine learning* **43**(1-2), 7–52.

Gonçalves, A. V., Thuler, L. C., Kestelman, F. P., Carmo, P. A., Lima, C. F. and Cipolotti, R. (2011), 'Underestimation of malignancy of core needle biopsy for nonpalpable breast lesions', *Rev Bras Ginecol Obstet.* **33**(7), 123–131.

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

Kersting, K., De Raedt, L. and Kramer, S. (2000), Interpreting Bayesian Logic Programs, *in* 'AAAI Workshop on Learning Statistical Models from Relational Data', pp. 29–35.

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

Kok, S. and Domingos, P. (2005), Learning the Structure of Markov Logic Networks, *in* 'International Conference on Machine learning', ACM, pp. 441–448.

Liberman, L. (2000), 'Percutaneous imaging-guided core breast biopsy: state of the art at the millennium', *Am J Roentgenol.* **174**(5), 1191–1199.

Liberman, L., Drotman, M., Morris, E. A. et al. (2000), 'Imaging-histologic discordance at percutaneous breast biopsy', *Cancer* **89**(12), 2538–2546.

Muggleton, S. (1996), 'Stochastic Logic Programs', *Advances in inductive logic programming* **32**, 254–264.

Muggleton, S. and Raedt, L. D. (1994), 'Inductive Logic Programming: Theory and Methods', *Journal of Logic Programming* **19/20**, 629–679.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

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

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

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

Sato, T. (1995), A Statistical Learning Method for Logic Programs with Distribution Semantics, *in* 'Proceedings of the 12th International Conference on Logic Programming (ICLP 95', Citeseer.

Sato, T. and Kameya, Y. (1997), PRISM: A language for symbolic-statistical modeling, *in* 'International Joint Conference on Artificial Intelligence', Vol. 97, Morgan Kaufmann, pp. 1330–1339.

Vennekens, J., Verbaeten, S. and Bruynooghe, M. (2004), Logic Programs with Annotated Disjunctions, *in* 'Logic Programming', Springer, pp. 431–445.

Author Biographies

Joana Côrte-Real received her joint Ph.D. degree in Computer Science from the Universities of Minho, Aveiro and Porto in 2018 on the topic of Probabilistic Inductive Logic Programming. After her Ph.D. she continued her career in industry.

Inês Dutra is an Assistant Professor at the Department of Computer Science, Faculty of Sciences of the University of Porto, Portugal and a researcher at the CINTESIS research center. Her main research interests are in (probabilistic) logic programming, inductive logic programming, data science and big data. She has been involved in organization of various international conferences and has served as a reviewer to several conferences and journals in the areas of artificial intelligence in general, machine learning and logic programming.

Ricardo Rocha is an Associate Professor at the Department of Computer Science, Faculty of Sciences, University of Porto, Portugal and a researcher at the CRACS & INESC TEC research unit. He received his PhD degree in Computer Science from the University of Porto in 2001 and his main research topics are the Design and Implementation of Logic Programming Systems, Tabling in Logic Programming and Parallel and Distributed Computing. Other areas of interest include Lock-Free Data Structures, Inductive Logic Programming, Probabilistic Logic Programming and Deductive Databases. He is also one of the main developers of Yap Prolog system, and in particular of the execution models that support tabling and parallel evaluation. He has published more than 100 refereed papers in international journals, conferences and workshops, and has served more than 50 events as PC chair or PC member.