Towards Efficient and Scalable Probabilistic Inductive Logic Programming

Joana Côrte-Real
Programa Doutoral em Informática das Universidades do Minho, Aveiro e Porto
Departamento de Ciência de Computadores
2018

Orientador
Inês Dutra, Professor Auxiliar, Faculdade de Ciências (UP)

Coorientador
Ricardo Rocha, Professor Associado, Faculdade de Ciências (UP)

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2018

To Anton.

Acknowledgements

Firstly and foremost, I would like to thank my advisors Inês Dutra and Ricardo Rocha.

Meeting Professor Inês broadened my horizons, both professionally and personally. Throughout my PhD, she not only encouraged me to seek collaborations in new places, but also tirelessly helped me with setting up visits which allowed me to experience research from different perspectives. Knowing Inês has also enriched me emotionally, and I will always remember her unfaltering support during my darkest days.

I have always thought that Professor Ricardo is one of the most reliable people I know. His work is faultless and his attention to detail is impressive. I am honoured for having had the opportunity to learn from his example how to better myself, and this has enabled me to produce the best work I am capable of, and for this I will always be grateful.

During my PhD, I visited different places. I would like to thank my hosts Keshav Pingali (University of Texas at Austin), Hitoshi Oi (University of Aizu), David Page (University of Wisconsin-Madison) and Hendrik Blockeel (KU Leuven) for hosting and nurturing me. In each of these visits, I felt truly welcome in the research group and have had the pleasure of meeting wonderful new people. From a professional point of view, I feel I have gained incredibly valuable experience, and also a deeper insight into my field and research in general. I am very grateful you enabled me to have this opportunity, it has changed my outlook on life and made me the person I am today.

I would also like to thank everyone at DCC. In particular, I've had the pleasure of collaborating with Professor Vítor Santos Costa, and his brilliancy and creativity is truly inspiring to me. I would also like to thank my colleagues over the years for all the moments well spent and long-lasting friendship: Miguel Areias, João Santos, Rui Vieira, Flávio Cruz, Pedro Ferreira, Edgard Neto, Mário Pereira, Koko, Theo Mantadelis, Margarida Carvalho, Ivone Amorim, Eva Maia, João Vinagre, Amir Hossein, David Aparício, Miguel Araújo, Joaquim Silva, João Silva, Nuno Moniz, Paula Branco, Mariana Oliveira, António Gonçalves, Vítor Cerqueira, Tiago Santos, Diogo Machado, Christopher Harrison and Ângela Cardoso. Finally, it was a privilege meeting all my colleagues from the PhD Programme MAP-i, from various backgrounds and cultures. Thank you for making my PhD a fascinating and unique experience.

I am grateful for the FCT grant SFRH/BD/52235/2013, from which I have profited during this PhD work. I am also grateful to project Advice-Based Learning for Health Care (PTDC/ EEI-SII/ 2094/ 2012) and to project NanoSTIMA (NORTE-01-0145-FEDER-000016).

Finally, I am grateful to my family and friends, for constant and unfaltering support. In particular, my friends Paulo, Joana and Margarida have always been there for me when I needed them, and that made all the difference in the world. Finally, I would like to thank my husband Anton for being my guiding light and always bringing me safely home.

I am deeply indebted to you all, thank you again.

Joana Côrte-Real

Abstract

Many real-world processes exhibit both relational structure and uncertainty, and Machine Learning approaches should be able to deal with these aspects. Probabilistic Inductive Logic Programming (PILP), a subset of Statistical Relational Learning, uses Inductive Logic Programming (ILP) extended with probabilistic facts to produce meaningful and interpretable models. This merger between First Order Logic (FOL) theories and uncertainty makes PILP an adequate tool for knowledge representation and extraction. However, this flexibility in PILP systems is coupled with an exponential search space growth when looking for good models (inherited from ILP), and so often only a subset of all possible models is explored due to limited resources. Furthermore, the probabilistic evaluation of FOL theories, from the underlying probabilistic logic language and its solver, is also computationally demanding. In order to mitigate this problem, this thesis introduces novel PILP pruning strategies that can help reduce the time taken to generate good probabilistic theories, and shows how they can be applied in several distinct parts of the PILP search space. It also describes a safe pruning criterion which guarantees that the optimal model is not pruned away when used in combination with one of the pruning strategies, as well as two alternative more aggressive criteria that do not provide this guarantee. Because these pruning criteria are based on the data's probabilistic characteristics, the use of pruning strategies results in minimum information loss, which in turn maintains the quality of the generated theories while causing a significant reduction in average execution time. Experiments performed using benchmarks from different areas and two PILP systems (one developed during this thesis and another from the literature) show that the pruning strategies are effective in maintaining predictive accuracy for all criteria and experimental settings, and reducing the execution time when using some of the more aggressive strategies and criteria, compared to using no pruning. Finally, a real world study using medical data was conducted and the resulting PILP models were more accurate in identifying false negatives when compared to other techniques.

Resumo

Existe um interesse crescente em técnicas de *Machine Learning* que sejam capazes de lidar com processos que exibem simultaneamente uma estrutura relacional nos seus dados e um grau de incerteza associado a eles. A Programação Lógica Indutiva Probabilística (PILP) – um método de *Statistical Relational Learning* – é uma extensão de Programação Lógica Indutiva (ILP) que utiliza factos e regras com um valor probabilístico para gerar modelos relevantes e facilmente interpretáveis por humanos. Esta combinação entre teorias lógicas e incerteza faz com que PILP seja uma técnica interessante para a representação e extração de conhecimento. No entanto, a flexibilidade inerente aos sistemas PILP está também associada a um problema de crescimento exponencial do espaço de procura dos modelos (semelhante àquele encontrado em ILP). Esta é a razão pela qual é frequente avaliar apenas alguns dos modelos possíveis, quando existem limitações relativas aos recursos computacionais disponíveis. Para além disso, a complexidade de avaliação de teorias probabilísticas, devida à linguagem lógica probabilística e ao seu método de inferência, é igualmente elevada. Com o objetivo de mitigar este problema, esta tese propõe um conjunto de estratégias de corte do espaço de procura de PILP que ajudam a reduzir o tempo necessário para gerar teorias probabilísticas de boa qualidade. Estas estratégias podem ser usadas com vários critérios de corte, um dos quais garante que o modelo ótimo é sempre avaliado, enquanto que os outros são mais agressivos mas não oferecem essa garantia. Dado que as estratégias de corte são baseadas nas características probabilísticas dos dados, o seu uso resulta numa perda mínima de informação e portanto a qualidade dos modelos gerados é geralmente mantida, enquanto que o tempo de execução do programa é significativamente reduzido. Vários conjuntos de dados e dois sistemas PILP (um deles desenvolvido durante este trabalho de tese) foram utilizados para validar as estratégias e critérios de corte propostos. Os resultados experimentais mostram que (i) as estratégias de corte não reduzem significativamente a qualidade dos modelos gerados em nenhum caso, e (ii) os critérios mais agressivos de corte reduzem o tempo de execução do programa. Finalmente, um estudo utilizando dados médicos reais mostrou que os modelos PILP gerados para estes dados são competitivos a identificar os falsos negativos, quando comparados com outras técnicas de *Machine Learning*.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

FOL	First Order Logic
ILP	Inductive Logic Programming
PBK	Probabilistic Background Knowledge
PE	Probabilistic Examples
PILP	Probabilistic Inductive Logic Programming
PLP	Probabilistic Logic Programming
SRL	Statistical Relational Learning

Chapter 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 (ILP) [40, 36], Graph Mining [11], Relational Data Mining [25] and Relational Reinforcement Learning [26]. Whilst Relational Learning techniques have been applied to a fair amount of problems, their discrete nature hinders them being used for solving complex problems involving uncertainty. Many real-world processes exhibit relational structure and uncertainty, and Machine Learning approaches should be able to deal with both these aspects. Statistical Relational Learning (SRL) [19] extends Relational Learning by: (i) allowing the training examples to be annotated with probabilistic information known a priori, and (ii) producing models whose prediction is a probability (as opposed to a class). 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.

SRL techniques need an underlying format to represent data. This work uses Probabilistic Logic Programming (PLP) languages as their knowledge representation framework. PLP languages are an extension of First Order Logic (FOL) where facts and rules can have a probabilistic value. The expressive power of PLP allows for coupling relations and uncertainty in the same knowledge representation framework. There are many inference systems for PLP that can represent and manipulate uncertainty, such as SLP [39], ICL [43], Prism [53], BLP [34], CLP($\mathcal{BN}$) [50], MLN [48], ProbLog [35], among others.

By applying ILP techniques to a *probabilistic logic setting* (as opposed to deterministic) it is possible to perform relational learning over uncertain data, and to produce interpretable models composed of logical theories which have an inherent probabilistic value associated. This process is thus called Probabilistic Inductive Logic Programming (PILP), and it consists of performing structure learning over PLP in order to produce models which are understandable by humans whilst still taking uncertainty into account. PILP is thus based on a merge of ILP and the SRL fields.

1.1 Thesis Purpose

The PILP methodology learns a predictive model from a set of probabilistic logic facts and rules. A PILP model (or theory) corresponds to a set of FOL rules and predicts probabilities using the logical rules in the model to perform probabilistic inference over the probabilities of the facts. PILP algorithms use a set of Probabilistic Examples (PE) and additional probabilistic logical information about the domain, the Probabilistic Background Knowledge (PBK), to find a model that explains the PE. PILP can perform structure learning – the logic rules compose a theory that models the structure of the PE w.r.t PBK – but also parameter learning – which can find probabilities for the background knowledge or for the rules learnt [18, 3]. 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.

One of the limitations of PILP systems is that they inherit the exponential search space from ILP, and must in addition evaluate the fitness of each candidate model by computing, for each example, the likelihood of that example given the model. Because PILP is based on probabilistic logic data, this can be very time consuming since the evaluation process must consider all *possible worlds* where the theory in the model may be true. For a small number of facts and rules in the PBK this is not a problem, but computation grows exponentially as the size of the PBK is increased [27].

The purpose of this thesis is to improve the scalability of PILP by investigating pruning strategies for PILP search space traversal. This work introduces three distinct pruning strategies that can be applied to the PILP algorithm and are aimed at making the best possible use of available resources. *Fitness pruning* is a type of beam search which implements a polynomially bound complexity on the PILP search space by selecting a subset of candidate theories to be combined, as opposed to calculating all possible combinations. *Estimation pruning* is a strategy aimed at improving the performance of PILP systems through the use of *estimators* that can replace the probabilistic evaluation of theories and thus prevent unnecessary theory evaluation. This strategy estimates the utility of candidate theories based on information previously available so as to prune some candidates away before they undergo exact probabilistic evaluation. Finally, *prediction pruning* prunes the PILP search space based on evaluated theories by taking into account the logical operation that will be performed next (conjunction or disjunction). Unlike the other two pruning strategies, prediction pruning can guarantee *safety*, meaning that the optimal model is never pruned away (depending on pruning setting).

Furthermore, this work investigates three possible criteria for pruning: *safe*, *soft* and *hard*. The safe criterion can only be applied in the prediction pruning strategy and it guarantees that the optimal model is never pruned away. Soft and hard pruning criteria are applicable to both estimation and prediction pruning and they perform a more aggressive prune of the search space. Because these 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 causing a significant reduction in average execution time.

In order to assess the impact of these pruning strategies, the PILP system SkILL was created. Unlike other systems in the literature, SkILL supports the exhaustive search of the PILP search space, and does so by first traversing the AND search space, and only then traversing the OR search space. This configuration is the most adequate to individually assess the impact of each pruning strategy, for both these logical operations, and varying criteria. Even though the full capabilities of the pruning strategies introduced here can be better showcased using the SkILL system, the concepts are general to any PILP engine.

An experimental analysis performed using three PILP benchmarks from different areas (biology, web and medical) shows that all three criteria maintain predictive accuracy for all experimental settings. Furthermore, the more aggressive criteria reduce execution time compared to using no pruning, without loss of predictive accuracy. Finally, in limited resource settings better candidate models are explored when compared to using no pruning. The proposed pruning strategies were also implemented in another PILP system, ProbFOIL+ [18], and results show that they are effective in increasing accuracy when using a beam search strategy.

1.2 Main Contributions

This works consists of the design, implementation and evaluation of a set of pruning strategies for PILP. To the best of the author's knowledge, these pruning strategies are the first designed specifically for the PILP probabilistic logic search space, and they make use of the probabilistic information of candidate theories. Because the probabilistic information of candidate PILP theories can be used in different ways, three pruning criteria to be used in conjunction with the pruning strategies are also proposed. Furthermore, the SkILL PILP system, also a part of this work, focuses on providing tools for the exhaustive traversal of the PILP search space and for assessing the impact of pruning strategies in PILP candidate theories evaluated. In more detail, the main contributions of this work are as follows.

PILP Search Space Description The PILP search space can be divided in the AND search space (which explores rules) and the OR search space (which explores theories). The relation between the logical operations of conjunction and disjunction, the predictions of theories, and their specificity/generality make pruning the PILP search space possible.

Pruning Strategies Based on a thorough PILP search space understanding, three pruning strategies are proposed in this work: fitness pruning, estimation pruning and prediction pruning. Different pruning strategies target different characteristics of the PILP search space traversal, but their main purpose is to make the exploration of the PILP search space as efficient as possible. The pruning strategies can be used separately or in combination with each other, and they can each use different pruning criteria, for either search space.

Pruning Criteria In estimation and prediction pruning, the decision to prune a candidate theory away can be made based on different pruning criteria. This work proposes three pruning criteria with varying degrees of aggression: safe, soft and hard pruning criteria. All of the pruning criteria take into account the predictions of a candidate theory, and how they are positioned w.r.t. the example (target) values. In the case of estimation pruning, the estimates of a candidate theory are used in lieu of the actual predictions computed through exact probabilistic evaluation.

The SkILL System Specifically for this work, the SkILL system was designed for the purpose of allowing for a thorough study of the impact of the pruning strategies, both for the AND and the OR search space. Unlike other systems in the literature, it allows for exhaustive traversal of the search space, and the search of the AND and the OR sarch spaces are conducted in separate stages so as to make the individual impact of pruning strategies more evident.

Performance Evaluation This work conducts a performance study for all pruning strategies and criteria (both individually and combined with each other). The performance of the pruning strategies proposed in this work takes into account the execution time of the program, as well as the predictive accuracy of the final optimal model chosen.

Real World Application A PILP breast cancer benchmark using data from non-definitive biopsies is constructed and an exploratory study of these data is proposed. This study builds a PILP model for the breast cancer data which predicts malignancy of the tumours and is statistically indistinguishable from the medical doctor's predictions. The PILP model was shown to predict consistently higher values for malignancy of tumours when compared to other techniques, which is a desirable feature in medical decision support systems.

1.3 Thesis Overview

This thesis document is composed of eight chapters, organised as follows.

1 **Introduction** presents the main context of this thesis, introducing Relational Learning and PILP in particular. This chapter also enumerates the main contributions of this work, namely the pruning strategies, pruning criteria, the SkILL system and experimental evaluation developed. Finally, it presents an overview of the thesis contents.

2 **Deterministic ILP** focuses on explaining ILP and its search space. This chapter contains mostly background and it starts out with a short overview of Logic Programming and how the ILP learning methodology can be implemented on this framework. It then moves on to detailing the AND and OR ILP search spaces and how they can be traversed. Then, ILP mode declarations and loss functions are briefly described, followed by some ILP related work regarding the search space traversal by greedy algorithms.

3 **Probabilistic ILP** describes how ILP can be extended to use Probabilistic Logic Languages as its FOL knowledge representation. This chapter starts out by explaining possible world semantics and how they map to logic FOL rules. It then moves on to describe PILP and the main differences between this approach and deterministic ILP, focusing again on the AND and OR search space traversal. Then, a short analysis of the size of the PILP search space is presented, and finally some related work approaches are contextualised and described.

4 **Pruning Strategies** introduces the three pruning strategies developed during this thesis work in detail. This chapter starts out by giving an overview of the pruning strategies and how they fit within PILP search space traversal. Then, the three pruning strategies are described in detail: fitness pruning, estimation pruning and prediction pruning. Pruning strategies can be used with varying pruning criteria, and so this chapter also describes the pruning criteria developed during this work: the hard, soft and safe pruning criteria. The latter is described in more detail since it allows for pruning theories and still guaranteeing that the optimal model is kept.

5 **The SkILL System** presents the PILP system that was developed during this dissertation, and whose focus is to enable the exhaustive traversal of the PILP search space. This chapter presents SkILL's general architecture, as well as its main level-based algorithm for the AND and OR search spaces, along with its parameters. All pruning strategies and criteria developed during this work are implemented in this system, and can be used either individually or in combination with each other.

6 **Experiments** presents the PILP benchmarks used for the experimental evaluation of the pruning strategies developed during this dissertation. Each pruning strategy is evaluated independently, and then the experimental evaluation proceeds to assess the effect of combining fitness pruning and prediction pruning, and fitness, estimation and prediction pruning. Finally, experiments using the PILP system ProbFOIL+ and using the pruning strategies in this system are presented.

7 **Real World Applications** describe non-trivial knowledge extraction experiments performed on one of the PILP benchmarks. This benchmark is composed of breast cancer data and it was annotated with physician input in several ways, described in detail in this chapter. Then, the PILP approach was compared

with other classifier methods and it was found to have a higher recall that both the expert predictions and other classifiers' prediction. Furthermore, two models using PILP predictions and medical doctors' predictions were developed and assessed in this chapter.

8 **Conclusion** describes the main contributions of this work and how they were attained, as well as presenting some future work directions.

Chapter 2

Deterministic ILP

This chapter presents the background of Logic Programming and Inductive Logic Programming (ILP). In particular, the search space of ILP and the basic algorithm to traverse it are described, as well as the language bias and the loss functions for ILP model evaluation.

2.1 Logic Programming

Since the mid-1900's until the present time numerous programming languages have been developed. As such, a need arose to identify common features amongst the programming languages so as to classify them accordingly. Therefore, four main paradigms have emerged from this process, matching every programming language to one of these categories: imperative programming, functional programming, logic programming or object-oriented programming.

Imperative programming semantic is composed of strict translations from machine language to a set of user commands, whilst object oriented languages are the most recent paradigm and focus on modular code where methods can easily be reused. Logic and functional languages can be grouped under the declarative programming paradigm, and they aim at creating a detachment between a program's goal and its execution details by enhancing the *declarative* characteristics of the language in preference to its control. This allows the programmer to focus on defining the solution to a problem (declarative semantics), rather than how that solution will be executed (procedural semantics) [37].

However, whilst functional programming is concerned with features such as recursion or pattern matching, logic programming languages are focused on the process of automatically reasoning about knowledge contained in the program. There is also a theoretical difference between functional programming and logic programming: the former uses lambda calculus and the latter predicate calculus. Since the latter are a subset of declarative languages, they maintain the characteristic that the programmer is only required to specify what a program should do, and the execution environment is responsible for executing the specification in a fairly efficient way. Because the programmer does not focus on the execution details of the algorithm, this paradigm results in a detachment between the *logic goals* of the program and its *execution goals*, which can be explored towards greater efficiency.

There are various languages in the logic programming category, such as the Datalog [8] or Godel [32] languages, but only the Prolog family will be discussed here since the remaining languages are out of the scope of this work. Prolog first appeared in 1972 [10], in result of extensive research on an experiment whose aim was to develop a strategy for computers to interpret natural language. Since then, it has evolved and branched out into a number of distributions such as SWI-Prolog [57], SICStus Prolog [7] or Yap Prolog [51].

In 1969, Cordell Green adapted Alan Robinson's resolution algorithm to create an automatic theorem proving procedure [30] applicable to first-order logic systems, from where the numerous declarative programming languages in existence today stem. In particular, Prolog's syntax is composed of clauses that can be expressed as a conjunction of literals, also known as Horn clauses. This type of logical construction is a subgroup of first-order logic where the implications can contain at most one positive literal as a consequent, and as such it is not only resoluble and complete given a set of axioms but it is also closed – the resolvent of two Horn clauses is also a Horn clause. This fact makes it possible and convenient to recursively solve these clauses using inference rules based on SLD resolution [30].

In 1983, David Warren introduced a memory architecture and an instruction set, later named the Warren Abstract Machine (or WAM) [56], meant to efficiently translate Prolog code to lower level instructions, then to be resolved. The WAM still presently sets a relevant standard amongst Prolog compilers [1]. It is important to note that whilst the order of the terms in a clause is mathematically indifferent, it can be computationally taxing.

Prolog is then a language composed of *rules* and *terms*, and their mutual interaction. It has been argued that the logic programming paradigm should have been named the relational programming paradigm [37] since that terminology better describes the nature of the language.

A term is the basic Prolog language entity, and it can be an *atom* (starts with lower-case letters or is enclosed in single quotation marks), a number (float, integer), or a compound term (also named a functor). A term can also be a free variable (its name starts with an upper-case letter or an underscore) which is type-less until it is *bound*, meaning that a value is then assigned to the variable. Since Prolog has no destructive assignment of variables, unification for each variable can occur only once. However, *backtracking* allows for unbinding already unified variables, since Prolog stores *choice points* and can restore a previous program state so as to explore alternative rules.

A rule in Prolog is necessarily a Horn clause, composed by a head and a body, and follows the structure presented in Eq. 2.1.

$$head : -body\_literal_1, body\_literal_2, ...body\_literal_N. \tag{2.1}$$

A rule's head and body are related by the operator `:-`, which is an implication: for the head to be true, the body must also be true. A rule can have no body – the equivalent to 2.2.

$$head : -true. \tag{2.2}$$

Or simply:

$$head. \tag{2.3}$$

In these cases the rule is named a *fact* and represents a statement that is always true in the program's scope. The set of rules and facts of a Prolog program is called its *clauses*. The rule/fact names in a program are also called *predicates*, and a predicate can have several clauses with the same *arity* (number of predicate arguments).

The body of a rule is composed of a sequence of *literals*, or *goals*, interacting with one another through *connectives*, or operators; in this case `,/2` corresponds to the AND connective. Each goal represents a call to a predicate, which is then determined to be true or fail. It is thus evident that the execution of a Prolog program requires both a *goal selection rule* to determine which goal is to be called next, and a *search rule* to choose which alternative of a goal to explore, if several exist. Prolog's resolution employs left-to-right goal selection and a depth-first search strategy, and each resolution step taken is called *reduction* or *logical inference*. Consider the following example of a basic Prolog program illustrating most of these concepts.

```
student(joana).
student(miguel).
professor(ines).
professor(ricardo).

advised_by(joana, ines).
advised_by(joana, ricardo).
advised_by(miguel, ricardo).
```

This program introduces four entities: Joana, Miguel, Inês and Ricardo, which in this example are people. There are four facts of arity one, which are encoded by two predicates: `student/1` and `professor/1`. These facts encode the knowledge that Joana and Miguel are students, and that Inês and Ricardo are professors. Note that logical facts are a particular type of clause which is composed of only a head, i.e. the body (antecedents) of facts is always true and when this is not the case (when the head of the clause has antecedents), the clause is a rule.

The `advised_by/2` clauses differ from the `student/1` and `professor/1` facts because they encode a relation between people. In this case, they state that Joana is advised by Inês and also by Ricardo, and that Miguel is advised by Ricardo.

Consider now that the program also contains a predicate `co_authors/2` which expresses a conditional rule between its head and body literals: if `P` is a professor, `S` is a student and `S` is advised by `P`, then `P` and `S` are co-authors.

```
co_authors(P, S) :-
  professor(P),
  student(S),
  advised_by(S, P).
```

Because `co_authors/2` also expresses a relation between two variables (`S` and `P`), it is said to have arity 2. The difference between the relation expressed by `co_authors /2` and `advised_by/2` is that the former is a conditional relation, whilst the latter is always true. The body of a rule such as `co_authors/2` is composed by a set of literals (again, ordering of these literals only impacts the order in which solutions for the rule are computed, it does not change the set of possible solutions). The meaning of this set of literals corresponds to the logical conjunction of all of them.

For instance, the rule for `co_authors/2` stated that both `professor(P)`, `student(S)` and `advised_by(S, P)` must be true so that the rule is verified.

It is also possible to combine literals disjunctively by combining several rules (clauses) with the same head.

```
co_authors(P, S) :-
  professor(P),
  student(S),
  advised_by(S, P).
co_authors(P, S) :-
  professor(P),
  professor(S),
  advised_by(X, P),
  advised_by(X, S),
```

Adding a second rule for `co_authors/2` results in a new *theory* for `co_authors/2`. The meaning of this theory is that `P` and `S` are co-authors if either `P` is a professor, `S` is a student and `P` is an advisor of `S`, or if `P` and `S` are both professors and both advisors of the same student `X`. Note that the rules in a theory are disjunctive w.r.t. each other – the theory is verified if any of its rules are true. Conversely, rules are only true if all of their literals (sub-goals) are true, since the literals are mutually conjunctive.

Theories such as `co_authors/2` can be queried to produce *positive* and *negative* examples, meaning sets of values for `P` and `S` that verify the theory, and sets of values which do not, respectively. The next example contains some queries one could now pose regarding the program shown earlier.

```
?- student(ricardo).
  no
?- student(Someone).
  Someone=joana?
  ;
  Someone=miguel?
  ;
  no
?- co_authors(ines, Someone).
  Someone=joana?
  ;
```

```
Someone=ricardo?
;
no
```

For instance, `{P = ines, S = joana}` is a positive example of `co_authors/2` and `{P = miguel, S = joana}` is a negative example of `co_authors/2`.

The aim of ILP is to reverse engineer this process: given positive and negative examples of some phenomenon (for instance `co_authors/2`), find the theory which best explains the positive and negative examples. In the deterministic case, this means that the theory should succeed for every positive example, and fail for every negative example.

2.2 Inductive Logic Programming

Inductive Logic Programming (ILP) [36] is a machine learning branch 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* - and that can then be used for assessing the quality of the produced predictive model [40]. In fact, these two approaches are also known as *predictive* and *descriptive* ILP, respectively, and their nature is often related to the way explanations are computed for a given problem; in the most frequent case – predictive ILP – learning from entailment is most commonly used. Learning from interpretations is also a possible setting for ILP theory computation, but it has been shown that it can be reduced to a learning from entailment scenario [47, 46].

ILP has a strong theoretical background, inherited from logic programming, and a good experimental approach due to its machine learning basis. Its strong theoretical nature allows for representing problems in a formal way while also offering great expressiveness that other machine learning approaches cannot match. Its experimental approach provides the capability to solve practical applications for inductive learning which can be targeted to solve specific problems.

The main advantage of ILP is, however, its capability to provide classifiers that are interpretable by humans, which makes it popular in different scientific domains. This research area is quite relevant in several field such as Knowledge Acquisition, Inductive Program Synthesis, Inductive Data Engineering, and Knowledge Discovery in Databases [36].

Even though many other classic Data Mining methods have been shown to be effective and are very widely used in numerous fields of knowledge, a shift in paradigm may be in order since the amount of data produced grows daily, making human validation of conclusions intractable. Ideally, one would like knowledge discovery methods to reach the same conclusion as an expert in the area would, in a much shorter time frame, and this can be emulated through ILP models.

The ILP learning algorithm requires that part of that data consists of *background knowledge* (BK), which are often of a relational nature. The BK can be thought of as the support knowledge that one would have when making the observations on the system. It is therefore crucial to have structured and well-suited BK for learning.

```
%% Background Knowledge (BK)
student(joana).
student(miguel).
professor(ines).
professor(ricardo).

advised_by(joana, ines).
advised_by(joana, ricardo).

%% Positive Examples (E+)
co_authors(joana, ines).
co_authors(miguel, ricardo).
co_authors(ricardo, ines).

%% Negative Examples (E-)
co_authors(joana, joana).
co_authors(joana, miguel).
co_authors(ines, miguel).
```

The observations (or examples) given to the system can be named *positive* (E^+) or *negative* (E^-), though negative examples are not required in descriptive settings. Examples are also facts, but this time regarding the relation whose pattern the system is attempting to determine; positive examples are cases in which the relation is verified to be true whilst negative examples are cases in which the relation is false.

From this example, an ILP system could learn this model (theory) for the data.

```
co_authors(P, S) :-
```

```
  professor(P),
  student(S),
  advised_by(S, P).
co_authors(P, S) :-
  professor(P),
  professor(S),
  advised_by(X, P),
  advised_by(X, S),
```

This theory for `co_auhtors/2` has *coverage* of 100% of positive examples and 0% of negative examples, making it an optimal choice for a theory to explain this dataset.

The process of theory (or hypothesis) induction in ILP was first formalized by Muggleton and de Raedt's in 1995 [40]. Its normal semantics are composed of four statements which must be met by the models generated by the system. The symbol $\models$ stands for logical entailment.

Prior Necessity $BK \not\models E^+$, meaning that the theories to be found must be different from the trivial true hypothesis.

Prior Satisfability $BK \wedge E^- \not\models \square$, which guarantees that the problem can be modelled by at least one theory.

Posterior Sufficiency $BK \wedge H \models E^+$, which means that the generated theories explains the positive examples.

Posterior Satisfability $BK \wedge E^- \wedge H \models \square$, so that the generated theories do not explain the negative examples.

The two latter statements (posterior sufficiency and posterior satisfability) are often relaxed in practice; this can be done in several ways like attempting to minimize the number of misclassified positive and/or negative examples.

One would thus expect that, for all theories, there would be a form of assessment of their explanatory value, or *justification*. Since there will often be more than one hypothesis fitting the four semantic statements for a given problem, the system must order them according to a preference, thus introducing a *preference* bias in the system.

In addition to the justification of a theory, the process by which it is formed - or *abduction* - must also be considered. The naïve approach of theory abduction is to just simply form all correct models under the *syntactic* bias of the language and compare their justifications in order to choose the best. However, this leads to an enormous search space and is computationally infeasible in many cases. As such, the notion of *generalization* and *specification* have arisen as ways to prune the search space.

Generalization A theory G is said to be more general than a theory S iff $G \models S$.

Specification A theory S is said to be more specific than a theory G iff $G \models S$.

These two concepts (generalization and specification) are the basis for theoretical search space pruning, since better results cannot be obtained from generalization of a theory that is inconsistent with negative examples or from specialization of a rule which is inconsistent with positive examples.

2.2.1 Search Space Description and Traversal

The theories used to explain examples in ILP are built from the predicates that are present in the program's BK. In the example mentioned before, there are three distinct predicates in the BK: `student/1`, `professor/1` and `advised_by/2`. The rule (AND) search space is composed by all rules whose body contains one or more literals generated from those predicates and BK. The number of literals in the body of a rule is termed the rule *length*. An example of a rule of length one is `co_authors (P, S):- professor(P)` and an example of a rule of length two is `co_authors(P , S):- professor(P), student(S)`. Depending on the mode declarations (to be presented next), it might also be the case that the same predicate can be repeated in a rule with different arguments. However, for the description of the search space, repeated predicates in rules can be mimicked by introducing auxiliary predicates (for instance). Therefore, for the sake of simplicity, this case is not considered in the explanation below.

Rules can be combined using logical conjunction to form longer more *specific* rules. The more literals are contained in the body of a rule (the greater its length), the more specific the rule is. The AND search space is then composed of all rules of various lengths which can be formed from the existing literals. More formally, let *Predicates* be the set of distinct predicates in the BK. For each predicate, a set of literals can be generated by instantiating each argument with either a constant from the BK or a variable. This set of literals for predicate p can be denoted by $\mathcal{L}(p)$.

Thus, the set of literals in the search space can be defined as shown below.

$$Literals = \bigcup_{p \,\in\, Predicates} \mathcal{L}(p) \tag{2.4}$$

Therefore, the AND search space *Rules* can now be defined as the power set of *Literals*, assuming that the mode declarations allow all possible combinations of distinct literals.

$$Rules = \mathcal{P}(Literals) \tag{2.5}$$

Given the three predicates introduced in the example

$$Predicates_{ex} = \left\{\texttt{student/1, professor/1, advised\_by/2}\right\} \tag{2.6}$$

the set of possible literals *Literals* is equal to

$$\begin{aligned} Literals_{ex} = \Big\{ & \texttt{student(X), student(joana), student(ines), student(ricardo),} \\ & \texttt{student(miguel), professor(X), professor(joana),} \\ & \texttt{professor(ines), professor(ricardo), professor(miguel),} \\ & \texttt{advised\_by(X,X), advised\_by(X,Y), advised\_by(joana,X),} \\ & \texttt{advised\_by(joana,joana), advised\_by(joana,ines),} \\ & \texttt{advised\_by(joana,ricardo), advised\_by(joana,miguel),} \\ & \texttt{...,} \\ & \texttt{advised\_by(miguel,ricardo), advised\_by(miguel,miguel)} \Big\} \end{aligned} \tag{2.7}$$

The set of rules *Rules* contains every possible combination of literals from *Literals* and it can also be represented as a lattice where the bottom element is the empty rule `co_authors(P, S):- true`, which corresponds to the ∅ set of literals and is trivially true. The top of the rule lattice is the most specific element, or the element of $\mathcal{P}(Literals)$ whose length is equal to $|Literals|$. Each level of the rule lattice increases in specificity (from bottom to top) and increases in generality (from top to bottom). Elements in the same level of the lattice are rules of the same length.

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. An example of a theory of length two is

```
co_authors(S, P):-
        professor(P).
co_authors(S, P):-
        student(S).
```

Adding a rule to a theory makes it more general. The OR search space is the set of all theories *Theories* such that:

$$Theories = \mathcal{P}(Rules) \tag{2.8}$$

Similarly to rules, theories can be represented in a theory lattice. The bottom element of the theory lattice is the empty set of rules, which is trivially false. The top element of the theory lattice is the most general theory which can be formed from the *Rules* set, and it contains all its elements, thus making its length equal to $|Rules|$. Because theories are combined using logical disjunction (as opposed to logical conjunction for rules), the generality ordering in the lattice is in the opposite direction of that of the rule lattice. In the theory lattice, the bottom element is the most specific element, and the first level of the lattice (starting from bottom) is composed of theories of length one, the second level of theories of length two and so on, until the longest most general element is reached. Therefore, theories increase in generality (and decrease in specificity) from bottom to top.

Fully exploring the ILP search space is equivalent to evaluating each theory in the theory lattice in order to determine the best theory according to a given metric. Algorithm 2.1 shows a complete general-to-specific procedure to exhaustively traverse the rule lattice (AND search space) based on [17]. Because this algorithm shows the exhaustive traversal of the search space, options to stop early or to order or prune the candidate set (beam search) are not included. A similar procedure can also be applied to the theory lattice (OR search space), using specific to general ordering instead.

Algorithm 2.1 starts out by initialising a set of candidate rules with the most general element in the rule lattice (line 1 in Alg. 2.1), as well as the set of all rules, or theories of length 1, T_1 as an empty set (line 2 in Alg. 2.1). The algorithm then traverses the search space exhaustively by popping an element r from the candidate set and evaluating it (lines 4 and 5 in Alg. 2.1). Evaluation of a rule (or theory)

Algorithm 2.1 $generic\_general\_to\_specific\_and(BK, E)$

```
1: $Candidates = \{true\}$
2: $T_1 = \emptyset$
3: while $Candidates \neq \emptyset$ do
4:     $r = pop(Candidates)$
5:     $T_1 = T_1 \cup evaluation(r, BK, E)$
6:     $Candidates = Candidates \cup refinements(r, BK)$
7: return $T_1$
```

can be performed using different loss functions against examples and allows for determining the best rule in the search space. The evaluation results are appended to the rule as meta-data. The evaluated rule is then added to the set of all rules T_1 and the rules refinements are generated and added to the candidate set (lines 5 and 6 in Alg. 2.1). The refinements of a rule r are obtained by specifying this rule, i.e. by generating rules of greater length which contain all the literals present in r. A way in which refinements of r may be generated is by combining rules of length 1 with r, resulting in rules of greater length. This procedure is then repeated until the candidate set is empty and there are no more rules to be evaluated (lines 3–7 in Alg. 2.1).

For the purpose of this work it is important to respect the partial ordering of the search space lattice, since this allows for a clearer assessment of the effect of pruning the search spaces, at each level of the lattice. Therefore, the procedures used to traverse the AND and OR search spaces used in this work (Alg. 2.2 and 2.3) use an explicit level-wise exploration approach. Even though the order of search space traversal can be the same in Alg. 2.1 and Alg. 2.2/2.3 (Alg. 2.1 can be configured to traverse the search space in a breadth-first fashion), a level-based algorithm ensures that, for each level of the lattice, there are separate candidate generation and evaluation stages. In Alg. 2.1, a theory is evaluated and its refinements are added to the queue, whilst in Alg. 2.2 and 2.3, all evaluations for one level of the lattice are performed in one step, and only then are all refinements for the theories in that level generated. This difference in approach allows for a level-based assessment of the effect of pruning strategies.

Algorithm 2.2 presents a procedure to explore the rule lattice. Since the first level of the theory lattice is composed of all theories of length one, i.e. all rules, the AND search space can be explored first (which corresponds to exploring the first level of the theory lattice).

Algorithm 2.2 *and_search_space*(BK, E)

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

Algorithm 2.2 explores the rule lattice in a direction of increasing specificity. The algorithm starts out by generating and evaluating rules of length one from the BK and E (i.e. rules containing only one literal), and then by using these rules to generate combinations R_{N+1} for the next iteration (lines 1–3). Rules R_{N+1} are then evaluated and stored in the set of theories of length one T_1 (lines 5 and 6), and this process is repeated until the combination of R_1 and R_N yields no valid rules, i.e., is empty (lines 4–8). Once this procedure is completed, the algorithm has exhaustively explored the AND search space and returns the set of all rules, which is also the set of all theories of length one (line 9). The set of theories of length one T_1 is then used to start searching the OR search space as shown in Algorithm 2.3.

Algorithm 2.3 *or_search_space*(T_1)

```
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:     $T_{N+1} = evaluation(T_{N+1}, BK, E)$
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 $T_{All}$
```

Algorithm 2.3 is similar to the one for the AND search space. It starts out by combining theories of length one to generate the theories T_{N+1} for the next iteration (line 2). Then it proceeds to evaluating these theories, storing them in T_{All} and generating the theories for the next iteration (lines 3–7). This process is repeated until the set of theories T_{N+1} for the next iteration is empty, meaning that no more valid candidates can be generated and the theory lattice is fully explored (line 3). The variable T_{All} now contains the set of all theories in the OR search space.

2.2.2 Language bias

The search space of ILP is the space of all theories which can be formed using the literals in the BK. However, not all possible rules in this space are relevant to the problem, and so it is common to define a declarative *language bias* using mode declarations in order to specify which rules are valid within the AND search space.

The language bias defines a number of *mode declarations* for specifying the way in which the literals in the BK can fit together, using types. An example of the mode declarations for the co_authors/2 example is presented next.

```
:- modeh(2, co_authors(+professor, -student)).

:- modeb(1, student(+student)).
:- modeb(1, professor(+professor)).
:- modeb(2, advised_by(+professor, -student)).
```

Mode declarations can be either head declarations modeh (only one for the target predicate co_authors/2 being learnt) or body declarations modeb. Each predicate in either modeh or modeb has its arguments annotated with types (in this case arguments are of type student and professor). Arguments also have associated prefixes, which can be + or -. These prefixes denote whether the argument is instantiated (or not) before the predicate is called. If the argument's prefix is a +, then it must be instantiated before the predicate is called. This argument is called an input argument and an example of this type of argument is the first argument in the declaration of professor/1. Conversely, if the argument's prefix is a -, the argument does not need to be instantiated before a call to the predicate is made, and is therefore called an output argument. An example the second argument in the declaration of advised_by/2.

This information can greatly reduce the AND search space, as it specifies a language bias that the final theory must be in accordance with. Rules which are in violation of the type system defined above are never generated. For instance, the rule co_authors(S, P):- professor(S), professor(P) will never be considered because the second argument of co_authors/2 must be of type student (line 1 of mode declarations), whilst the argument of `professor/1` must be of type `professor` instead. Note that this language bias example is meant to illustrate the usage of the argument types, but was not used for the illustrative example presented before, since it would reduce the number of generated rules, and exclude the theory of interest. In particular, the second rule `co_authors/2` in that theory applies to cases where both `P` and `S` are professors, and the language bias presented here prevents that case. In order to generate that theory for the example above, the types `student` and `professor` would have to be merged into only one type (e.g. `person`). Because the problem presented here is meant as a toy example, the usefulness of the mode declarations is not fully showcased. However, in real-world problems, a language bias is commonly used to reduce the AND search space. Furthermore, there is a semantic motivation to use this type system, since ILP is usually applied in the supervised learning setting, in particular for tasks where the aim is to extract non trivial knowledge about a phenomenon from a dataset in the form of logical rules. To this end, rules which are known to be semantically not valid are not of interest as a logical model of the problem.

2.2.3 Loss Function and Theory Evaluation

ILP theories describe a given *target predicate* in terms of the clauses contained in the BK and evaluate that description based on positive and negative examples, E^+ and E^-, of the target predicate. For this purpose, different theories from the ILP search space are evaluated w.r.t. the set of examples $E = E^+ \cup E^-$ and their fitness to describe the problem is assessed according to some *loss function*.

Predictions of an ILP model (i.e. a 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.

For each example, the ILP theory can be used to compute a *prediction*. For some theory t, an example i (positive or negative) and its corresponding value e_i (1 for positive, 0 for negative), the theory's prediction p_i is defined as follows.

$$p_i = t(i) \tag{2.9}$$

The closer this prediction is to the original example value e_i, the better that model is, for that example i. The best possible ILP model for a problem would predict the example value e_i, for each example, as shown in Eq. 2.10.

$$\forall i : e_i - p_i = 0 \tag{2.10}$$

ILP loss functions can also be defined in terms of the distance d_i between a prediction p_i and an example value e_i.

$$d_i = e_i - p_i \tag{2.11}$$

For ILP (in the deterministic case), this distance d_i will be either -1, 0 or 1. If the distance is zero, then the prediction of the model is correct. Otherwise, the prediction is incorrect.

Several metrics considering distance d_i over all points 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, over all examples $i \in E = E^+ \cup E^-$. These metrics are naturally also applicable to the deterministic case.

$$MAE(t) = \frac{1}{|E|} \sum_{i \in E} |d_i| \tag{2.12}$$

$$RMSE(t) = \sqrt{\frac{1}{|E|} \sum_{i \in E} d_i^2} \tag{2.13}$$

Alternatively, loss functions can be defined in terms of true positive (TP_i), true negative (TN_i), false positive (FP_i) and false negative (FN_i) parts of examples.

$$\begin{aligned} TP_i &= min(e_i, p_i) \\ TN_i &= min(1 - e_i, 1 - p_i) \\ FP_i &= max(0, 1 - e_i - TN_i) \\ FN_i &= max(0, e_i - TP_i) \end{aligned} \tag{2.14}$$

The overall TP, TN, FP and FN values for a theory t are calculated by summing over all examples.

$$\begin{aligned} TP(t) &= \sum_{i \in E} TP_i \qquad & TN(t) &= \sum_{i \in E} TN_i \\ FP(t) &= \sum_{i \in E} FP_i \qquad & FN(t) &= \sum_{i \in E} FN_i \end{aligned} \tag{2.15}$$

Accuracy is used in classification problems to assess the number of correct predictions in proportion to all predictions made. Accuracy *Acc* can be defined in terms of either MAE or TP, TN, FP, FN metrics. The two formulations are equivalent, as shown in Eq. 2.16.

$$\begin{aligned} Acc(t) &= \frac{TP(t) + TN(t)}{TP(t) + TN(t) + FP(t) + FN(t)} \\ &= \frac{1}{|E|} \sum_{i \in E} \Big(TP_i + TN_i \Big) \\ &= \frac{1}{|E|} \sum_{i \in E} \Big(min(e_i, p_i) + min(1 - e_i, 1 - p_i) \Big) \\ &= \frac{1}{|E|} \sum_{i \in E} \Big(1 - |e_i - p_i| \Big) \\ &= \frac{1}{|E|} \sum_{i \in E} \Big(1 - |d_i| \Big) \\ &= 1 - MAE(t) \end{aligned} \tag{2.16}$$

In a probabilistic setting, *Acc* as defined in Eq. 2.16 is commonly referred to as *PAcc*.

2.3 Related Work

Most ILP systems in the literature do not traverse the ILP search space exhaustively, since that process would be computationally taxing. Instead, they are based on a greedy algorithm such as FOIL [45], and traverse the ILP search space in a greedy way. A simple greedy procedure is presented in Alg. 2.4, where *BK* is the Background Knowledge, *E* is the set of positive and negative examples (E^+ and E^-, respectively), and L is the set of all literals in the BK. The lower case letters *r* and *t* denote one rule and one theory (respectively), as opposed to a set of rules or theories (as was the case in Algs. 2.2 and 2.3).

Algorithm 2.4 *FOIL_algorithm*(BK, E, L)

```
1: $t_N = \{false\}$
2: $E^+ = positive\_examples(E)$
3: while $E^+ \neq \emptyset$ do
4:     $r_N = \{true\}$
5:     $E^- = negative\_examples(E)$
6:     while $E^- \neq \emptyset$ do
7:         $r_1 = choose\_literal(L, E)$
8:         $r_N = r_N \wedge r_1$
9:         $E^- = E^- \setminus \{e \in E^- : r_N \not\models e\}$
10:    $t_N = t_N \vee r_N$
11:    $E^+ = E^+ \setminus \{e \in E^+ : t_N \models e\}$
12: return $t_N$
```

In Alg. 2.4, the inner loop (lines 6–9) occurs in AND search space, and the outer loop (lines 3–11) in OR search space. The outer loop builds a theory t_N composed of rules r_N built in the inner loop. In order to build a rule, a single literal is selected greedily from the BK taking into account the examples (line 7 in Alg. 2.4). This requires some evaluation to be performed at this point, since the literal is chosen based on a ranking given by a scoring metric. Once this literal is selected, it is added to the rule r_N (line 8 in Alg. 2.4), and only the negative examples that are still entailed by this new rule are kept for the next iteration of the inner loop (line 9 in Alg. 2.4). The decision on which examples to keep for the next iteration also requires evaluation of the rule against the examples.

The outer loop performs a similar operation to the inner loop. The rule r_N produced by the inner loop is added to the current theory t_N (line 10 in Alg. 2.4), and this time positive examples which entail the new theory are removed from the set of positive examples to cover (line 11 in Alg. 2.4). Again, removal of positive examples includes a form of evaluation of the current theory against them. Finally, the algorithm returns the final theory found, when there are no more positive examples to cover.

Several modifications can be done to Algorithm 2.4, such as to relax the stopping conditions in the loops, add other stopping conditions or use a beam instead of only one rule or theory [24]. Nevertheless, this algorithm is not suitable to thoroughly assess pruning strategies in the ILP search space for two reasons: (i) it does not fully traverse the search space, and (ii) it does not respect the partial ordering of theories (i.e. traverse the search space lattice in order), therefore making it difficult to assess whether the pruning strategy is impacting the AND or the OR search space. Furthermore, this search strategy does not guarantees that no element of greater length (more specific/general) will be evaluated before all elements of shorter length (less specific/general) are processed, therefore not respecting the partial ordering of the lattice.

The fact that this algorithm is not suitable for testing pruning strategies does not mean that they can not be applied to such an algorithm. Pruning strategies are applicable to greedy algorithms and have been shown to have an impact in performance (see Section 6). Still, respecting the partial ordering or theories makes for a clearer evaluation and testing procedure for pruning strategies, which is the goal of this thesis work.

Chapter 3

Probabilistic ILP

This chapter introduces Probabilistic Logic Programming and explains the way in which uncertainty is used to extend Logic Programming. Then, the concept of Probabilistic ILP is introduced, and the connection to ILP and Logic Programming established. Finally, a complexity analysis of PILP and a description of PILP systems in the literature are given.

3.1 Probabilistic Logic Programming

One of the limitations of Logic Programming 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 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 Logic Programming consists of using Sato's distribution semantics [52], where a program in Logic Programming 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 Probabilistic Logic Languages in the literature, such as SLP [39], ICL [43, 44], PRISM [53, 54], BLP [34, 33], CLP($\mathcal{BN}$) [50, 49], ProbLog [21, 35], among others, but only ProbLog will be discussed here since a comparison of Probabilistic Logic Languages is out of the scope of this work (for such a comparison see [20]).

ProbLog is based on the *possible world semantics* [35]. Each clause $p_j :: c_j$ 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 clauses of the program represents a possible world ω_i, where ω_i^+ is the set of clauses that are true in that particular world, and $\omega_i^- = \omega_i \setminus \omega_i^+$ is the set of clauses that are false. Since these clauses have a probabilistic value, a ProbLog program defining a probabilistic distribution over the possible worlds can be formalized as shown in Equation 3.1.

$$P(\omega_i) = \prod_{c_j \in \omega_i^+} p_j \prod_{c_j \in \omega_i^-} (1 - p_j) \tag{3.1}$$

A ProbLog *query* q is said to be true in all worlds w^q where $w^q \models q$, and false in all other worlds. As such, the *success probability* of a query is given by the sum of the probabilities of all worlds where it is found to be true, as denoted in Equation 3.2.

$$P(q) = \sum_{\omega_i \models q} P(\omega_i) \tag{3.2}$$

Using ProbLog's notation, the deterministic `advised_by/2` facts can thus be extended with uncertainty as follows:

```
student(joana).
professor(ines).
professor(ricardo).

0.7 :: advised_by(joana, ines).
0.9 :: advised_by(joana, ricardo).
```

In this case, the probabilities annotated above can be interpreted as frequencies. The new probabilistic facts now state that Joana is advised by Inês 70% of the time, and advised by Ricardo 90% of the time. This program is composed of four mutually exclusive worlds defined by the two `advised_by/2` facts which have probabilistic annotations. The worlds and their respective probabilities are shown below (assuming mutual independence between the `advised_by/2` facts). Note that the success probabilities for all four worlds sum to one, as expected.

Joana is advised by Inês and Ricardo simultaneously This possible world describes the situation in which Inês and Ricardo are both performing advisor duties for their student Joana, meaning that both `advised_by/2` facts are true. The success probability of this world can be calculated by taking the product of the truth probability of both facts. As such, the probability of success of this world is $0.7 \times 0.9 = 0.63$. This is the most likely of all possible worlds for this program.

Only Inês is advising Joana but not Ricardo. This corresponds to the setting where the first `advised_by/2` fact is true but the second one is false. The success probability of this world is equal to $0.7 \times (1 - 0.9) = 0.07$.

Only Ricardo is advising Joana but not Inês. This case is similar to the previous point, only this time the first fact is false and the second true. The probability of success for this world is $(1 - 0.7) \times 0.9 = 0.27$.

Neither Inês nor Ricardo are advising Joana In this case, both `advised_by/2` facts are false. This world is the less likely of all possible worlds and its success probability is equal to $(1 - 0.7) \times (1 - 0.9) = 0.03$.

The probabilistic information above is taken into account by ProbLog when queries are posed, in addition to the logic semantics of the program. Queries can still be posed about these facts, and in addition to the logical part of the program, a probability of success is returned. This probability is computed using probabilistic inference over the logical facts and rules of the program.

```
?- advised_by(ines,Someone).
  no (P = 0.0)
?- advised_by(joana,ricardo).
  yes (P = 0.9)
?- advised_by(joana,Someone).
  Someone=ines? (P = 0.7)
  ;
  Someone=ricardo? (P = 0.9)
  ;
  no (P = 0.0)
```

3.2 From Deterministic to Probabilistic ILP

Probabilistic Inductive Logic Programming (PILP) differs from deterministic ILP in that facts have success probabilities ranging between 0 and 1, as opposed to being either 0 or 1 (false or true, respectively). In this setting, there are no longer positive and negative examples, but only target probabilities for each example. The aim of a PILP theory is to produce probability values for each example which are as close as possible to the target probability for that example.

Probabilistic ILP is a subset of Statistical Relational Learning (SRL) that handles statistical relational learning by using a probabilistic First Order Logic (FOL) language to represent data and their induced models. This technique merges technologies from the SRL and ILP [40] fields in order to automatically compose theories as understandable FOL sentences based on data annotated with probabilistic information. Similarly to ILP, PILP algorithms use a set of Probabilistic Examples (PE) and logical information pertaining complex relations expressed as logic facts and rules, Probabilistic Background Knowledge (or PBK), to find a FOL model or explanation that explains the PE. PILP focuses on structure learning – the logic rules compose a theory that models the structure of the PE w.r.t PBK – but parameter learning can also be incorporated in this technique by tuning the probabilistic output of the rules which are learnt [18]. 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.

Introducing probabilistic information in ILP to create PILP allows for modelling uncertain or noisy data, which in turn results in a closer representation of reality – instead of using crisp true or false statements, PILP can model facts or rules which are believed to be true to some degree or with a given frequency. Probabilities in a logic setting can also be used to transform rules that use numerical arguments as weights in annotated probabilistic data. One of ILP's shortcomings is dealing with numerical arguments, which are usually binned. Converting these numerical arguments to probabilities in the PILP setting solves this problem, with the added benefit that it implicitly reduces the overall theory search space, since the predicates have now one less (numerical) argument to be taken into account when generating theories. Furthermore, in cases where the full conditional probability table is not known, the PBK can still be annotated with marginal probabilities (e.g., if available from other sources) and these annotations will still be taken into account when building the final theory model. Additionally, in cases where there are privacy concerns, a similar approach can be used to avoid using the patient instances explicitly, while still considering some of the information contained in the original data, by aggregating data from multiple patients.

As explained in Chapter 2, traditional ILP FOL rules describe a given *target predicate* in terms of the clauses contained in the Background Knowledge (BK) and evaluate that description based on positive and negative examples (E) of the target predicate. For this purpose, different rules are generated from the BK and E and their fitness to describe the problem is assessed according to a *loss function*. It is also common to define a declarative *language bias* using mode declarations in order to specify which rules are valid within the search space.

The aim of ILP is to find a set of rules (referred to as theory in this work) that entails all positive examples and does not entail any of the negative examples, but in practice it is common to relax these criteria and allow for some noise (misclassified examples). PILP extends the ILP setting by introducing PBK, where FOL data descriptions can be annotated with a probability value ranging from 0 to 1, and PE, no longer positive or negative, also with a value ranging between 0 and 1.

Because PILP theories are still generated based on the logical structure 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 examples.

Despite the similarities between ILP and PILP, there are several syntactic differences between them. Table 3.1 summarises these differences using Prolog syntax for ILP and ProbLog syntax for PILP.

Table 3.1: 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).` `def_cl1(X,Y):- fact1(E,X),fact2(E,Y).` `def_cl2(Y):- fact2(E,Y),def_cl1(X,Y).`	`target(E):-` `fact2(E,V),def_cl1(V,V).` ↓ *truth value* ∈ {TRUE,FALSE}
PILP	p_e`::target(e).`	p_{f1A}`::fact1(e,propA).` p_{f2B}`::fact2(e,propB).` p_{f1C}`::fact1(e,propC).` p_{f2D}`::fact2(e,propD).` p_{c1}`::def_cl1(X,Y):- fact1(E,X),fact2(E,Y).` p_{c2}`::def_cl2(Y):- fact2(E,Y),def_cl1(X,Y).`	`target(E):-` `fact2(E,V),def_cl1(V,V).` ↓ *truth value* ∈ $[0,1]$

3.2.1 From Rules to Probabilities

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. However, facts and rules in the PBK can now 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) [31].

As such, the previous `co_authors/2` and `advised_by/2` predicate can now be annotated with a probabilistic value. The fact that the PBK is now composed of probabilistic facts and rules alters the semantics of the logical part of the predicates as well, when compared to the deterministic version. For instance, in this case, the probabilistic value annotated in `advised_by/2` can refer to the proportion of time that the student's advisor tasks are performed by a professor. If a professor is present in every meeting, reviews all the reports and papers, and deals with all the administrative tasks of a student, then this value is set to one. If, conversely, the professor does not advise the student at all, the value is set to 0. This semantics is very different from the deterministic version, where the information represented by the `advised_by/2` facts is just whether the professor is an advisor or co-advisor of a student. In the probabilistic version of this example, non-official advisor can be listed as performing advisor tasks for a student, as shown by the two new `advised_by/2` facts presented next.

```
0.70 :: advised_by(joana, ines).
0.90 :: advised_by(joana, ricardo).
0.10 :: advised_by(miguel, ines).
0.05 :: advised_by(pedro, ricardo).
```

Similarly, the `co_authors/2` predicate can now refer to the proportion of papers in the last year where the second researcher also participated. If the first researcher participated in every paper that the first researcher published, then that fact is annotated with one. If the second researcher did not participate in any of the publications of the first researcher, then the predicate will be annotated with zero. This semantics is very different from the deterministic version, where the information represented by the `co_authors/2` facts is just whether the researchers publish together (at some point in time, always, in the last year, . . .). In particular, note that the probabilistic semantics of the `co_authors/2` predicate makes it non-symmetric, contrary to the deterministic version – the second researcher may have

Table 3.2: Probabilistic value for every possible set of choices for the binary random variables introduced in the PBK (shaded cells contain the probabilistic value for when the fact is false). The last column presents the world probability for the set of choices in that row. The sum of the world probabilities is one.

	(joana, ines)	(joana, ricardo)	(miguel, ines)	(pedro, ricardo)	World probability
1	0.7	0.9	0.15	0.05	0.0047
2	0.7	0.9	0.15	0.95	0.0898
3	0.7	0.9	0.85	0.05	0.0268
4	0.7	0.9	0.85	0.95	0.5087
5	0.7	0.1	0.15	0.05	0.0005
6	0.7	0.1	0.15	0.95	0.0100
7	0.7	0.1	0.85	0.05	0.0030
8	0.7	0.1	0.85	0.95	0.0565
9	0.3	0.9	0.15	0.05	0.0020
10	0.3	0.9	0.15	0.95	0.0385
11	0.3	0.9	0.85	0.05	0.0115
12	0.3	0.9	0.85	0.95	0.2180
13	0.3	0.1	0.15	0.05	0.0002
14	0.3	0.1	0.15	0.95	0.0043
15	0.3	0.1	0.85	0.05	0.0013
16	0.3	0.1	0.85	0.95	0.0242
					1.0000

been an author in all the publications of the first researcher in the last year (a ratio of 1), but the first researcher may only have participated in 10% of the second researcher's papers.

As mentioned before, in this work, probabilities are annotated according to ProbLog's syntax, using *possible world semantics* [20]. Each fact $p_j :: c_j$ in the PBK represents an independent binary random variable in ProbLog, meaning that it can either be true with probability p_j or false with probability $1 - p_j$. This means that each probabilistic fact introduces a probabilistic choice in the model. For instance, the second fact in the PBK `0.70 :: advised_by(ines, joana)` can be chosen to be true with probability 70% and false with probability 30%. For the probabilistic facts presented above, the choices and world probabilities are presented in Table 3.2.

From the table it can be seen that the success probability of (for instance) `advised_by (joana, Someone)` is the sum of all the world probabilities where either `advised_by` (joana, ines) or advised_by(joana, ricardo) is true (rows 1–12 in Table 3.2), totalling 0.97.

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 probabilistic facts in the underlying model.

Suppose the following values were given for PE:

```
1.00 :: co_authors(joana, ines).
1.00 :: co_authors(joana, ricardo).
0.20 :: co_authors(ines, ricardo).
0.10 :: co_authors(ricardo, ines).
```

Based on the PBK, different PILP rules will predict different values for each of the examples. These predictions are shown in Table 3.3.

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

Even though the prediction (success probability) of a rule changes according to the literals contained in its body, the probabilistic model generated from the PBK is not altered during the learning and evaluation stages of the learning algorithm. The search for the best theory in PILP thus consists of finding the theory whose success probabilities (for all examples) have the best fitness w.r.t. the probabilistic example values (according to some loss function) given a probabilistic model specified in the PBK. In the case of the three rules presented in Table 3.3, the third rule obtains the best result using both the MAE and the RMSE metric.

Table 3.3: Rule predictions for the four examples presented above and metrics MAE and RMSE (ca stands for co_authors/2)

Rules	$e_1 = 1.00$	$e_2 = 1.00$	$e_3 = 0.2$	$e_4 = 0.1$	MAE	RMSE
ca(X, Y):- professor(X)	0.0	0.0	1.0	1.0	0.93	0.86
ca(X, Y):- student(Y)	0.0	0.0	0.0	0.0	0.58	0.51
ca(X, Y):- advised_by(X, Y)	0.7	0.9	0.0	0.0	0.18	0.04

Extending these concepts to the probabilistic setting allow for defining standard scoring metrics such as *probabilistic accuracy (or PAcc)*, as introduced by De Raedt *et al.* in [22]. PAcc can also be represented in terms of the mean absolute error (MAE) between predictions and example values as used by Chen *et al.* in [9]. These two formulations are equivalent.

3.2.2 PILP Search Space Traversal

Similarly to ILP, fully exploring the PILP search space is equivalent to evaluating each theory in the theory lattice in order to determine the best theory according to a given metric. The 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 Algorithms 3.1 and 3.2.

Algorithm 3.1 *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 3.2 *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 3.1 and 3.2 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.

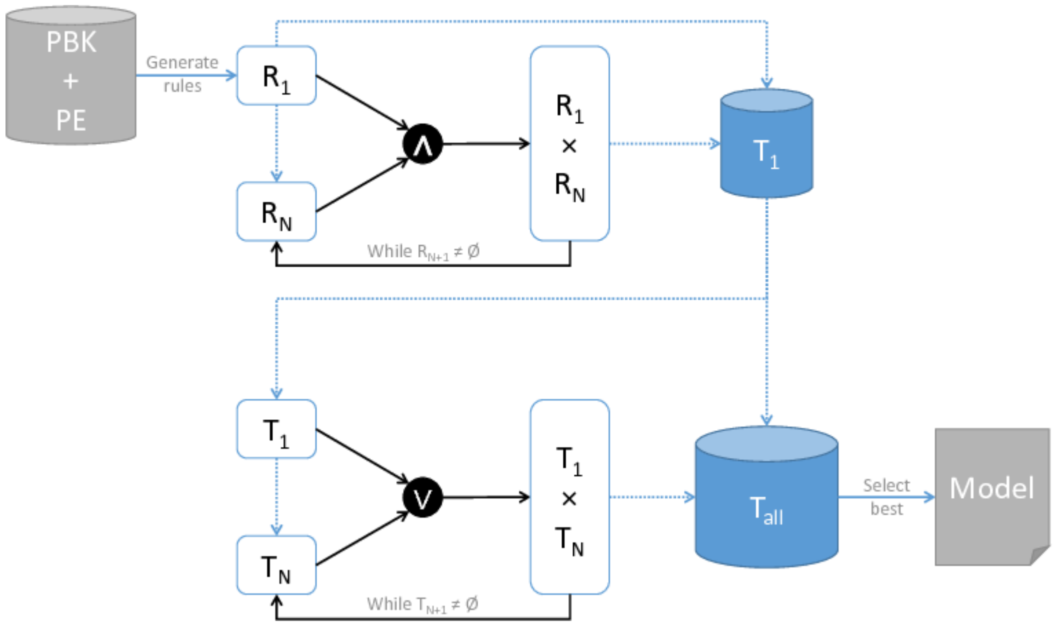


Figure 3.1: Overview of naive PILP search space traversal

The main difference between ILP and PILP algorithms consists of the evaluation of theories. 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 and 4 in Algorithms 3.1 and 3.2, respectively). 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.

Figure 3.1 shows a depiction of the PILP algorithms. On the top left rules of length one R_1 are generated from the probabilistic BK and examples, and they are combined through logical conjunction (increasing specificity) into longer rules. The result of this combination is stored in theories T_1 to the top right of the figure, and this process is repeated until no more combinations of rules can be generated. A similar process occurs for the OR search space (lower half of the figure), where theories of length one T_1 are used to generate combinations of longer more general theories for the next iteration, again until no more valid candidates are possible. All theories in the search space are then stored in T_{All}, and from those the best model can be selected, according to some metric.

3.2.3 Size of the PILP Search Space

The size of PILP search space depends on several distinct factors, namely the number of constants and probabilistic facts in the PBK, $|Constants|$ and $|Facts|$, the number of examples in PE, $|Examples|$, and the number of distinct predicates $|Predicates|$ in the PBK. The set of literals that can be generated for a given predicate p is defined as $\mathcal{L}(p)$. The cardinality of this set is bound by the number of constants in the PBK as shown in Equation 3.3.

$$|\mathcal{L}(p)| \leq |Constants|^{arity(p)} + V \tag{3.3}$$

Predicates may have arguments that can be bound in different ways when building rules (free variables or constants), as described by the mode declarations of the program, but this factor is dependent on each PILP program and cannot be described in general. This is denoted in Equation 3.3 as the term V.

Thus, the set of literals in the search space can be defined as shown below.

$$Literals = \bigcup_{p \in Predicates} \mathcal{L}(p) \tag{3.4}$$

Assuming that the mode declarations allow all possible combinations of distinct literals, the number of rules and theories in the search space can now be defined in terms of the power sets of literals as follows:

$$\begin{aligned} |Rules| &= |\mathcal{P}(Literals)| = 2^{|Literals|} \\ |Theories| &= |\mathcal{P}(Rules)| = 2^{|Rules|} = 2^{2^{|Literals|}} = 4^{|Literals|} \end{aligned} \tag{3.5}$$

Equation 3.5 shows that the size of the theory search space is exponential on the number of distinct literals generated from the PBK.

The probabilistic evaluation of each theory generated during the search space traversal requires a probabilistic query per example in the PE. This is necessary since the evaluation metrics require a prediction p_i for each example i so that the theory can be evaluated. Therefore, the number of probabilistic evaluations of a theory is equal to the number of examples $|Examples|$.

Furthermore, for each theory and each query to that theory, the probabilistic evaluation of that query becomes more taxing with a growing number of probabilistic facts in the PBK, since exact probabilistic evaluation is of exponential complexity on the tree width of the underlying graph represented by the Horn clauses in the PBK [27]. Equation 3.6 shows how the number of worlds in a probabilistic logic program relates to the number of probabilistic facts in the PBK.

$$|Worlds| = 2^{|Facts|} \tag{3.6}$$

Equation 3.6 shows that each new fact added to the underlying probabilistic model results in an exponential increase in the number of worlds, since it corresponds to introducing a new random binary variable. This means that the complexity of obtaining one prediction for a theory in the PILP search space is exponential on the number of probabilistic facts in the PBK.

Therefore, the time complexity of the PILP search space traversal and theory evaluation is

$$O(4^{|Literals|} \cdot |Examples| \cdot 2^{|Facts|}) \tag{3.7}$$

3.3 Related Work

Probabilistic Inductive Logic Programming is a Statistical Relational Learning (SRL) [29] method which can perform structure learning over probabilistic examples using a database of FOL facts and rules, also probabilistic. SRL is a field of research which encompasses many different methods such as Probabilistic Relational Models (PRM) [28], Markov Logic Networks (MLN) [48], Bayesian Logic Programs (BLP) [34] or Stochastic Logic Programs (SLP) [39], but the focus of this work is in PILP methods.

Figure 3.2 shows an (incomplete) overview of the SRL field focusing on the PILP method and systems (highlighted in blue).

There are currently three PILP systems in the literature: ProbFOIL, SLIPCOVER and SkILL.

In 2011, Raedt and Thon presented the first ProbFOIL version [22], based on the probabilistic inference language ProbLog [35]. ProbFOIL extends the standard ILP setting by being able to induce FOL theories from a BK annotated with probabilistic information. ProbFOIL uses a sequential covering algorithm which greedily adds

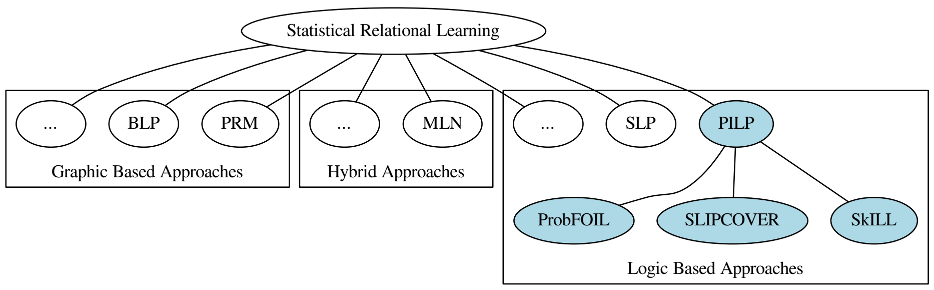


Figure 3.2: Overview of SRL field showing PILP systems in more detail

rules to the theory based on a local scoring metric, and similarly, to build each rule it adds literals taking into consideration the mode declarations. This process stops when adding another rule to the theory does not decrease the value of the loss function. ProbFOIL uses probabilistic accuracy as a global score and experimental results show that ProbFOIL can find the expected theories in two toy datasets, but also a few unwanted rules that are due to noise in the data [22]. Another contribution of this work was to extend standard machine learning metrics such as precision, accuracy and m-estimate to the probabilistic setting. This makes it possible to define the contingency tables of the examples, and interpret the probabilistic predictions in terms of their true positive and negative parts.

ProbFOIL+ [18] extends ProbFOIL by being able to tune the prediction of a theory by finding a weight for each rule in that theory. Because there is a priori probabilistic information annotated in the BK, every theory outputs a probabilistic value for a given example that is calculated based on that information. ProbFOIL+ improves the precision of that prediction by finding the weight (between 0 and 1) that maximizes precision of a rule for all examples. The ProbFOIL+ algorithm is similar to that of ProbFOIL, and so every time a rule is being added to the theory, the best weight is computed and added to that rule, which is then integrated in the theory. This can be seen as a form of *boosting*, since the importance of each rule in the theory is being adjusted, even though the possibility for adjustment is limited (the weight must be between 0 and 1). ProbFOIL+ was tested against regression learners in a propositional setting and was found to be comparable; experiments showed that ProbFOIL+ achieves higher precision results when compared to the first ProbFOIL [18].

SLIPCASE and SLIPCOVER are based on the probabilistic logic language of LPAD [55], which has the same representative capacity of ProbLog. SLIPCASE [2] can generate theories by performing refinements over an user given theory. These refinements are based on user-defined mode declarations, and for each possible refinement a log-likelihood metric is used to assess it. That value is then compared to the log-likelihood of the original theory, without the refinement, and a decision is made on whether to continue refining that rule or to stop. SLIPCASE performs a beam search, and so in each step it produces and evaluates every possible refinement for that theory. Refinements on a given theory consist of performing standard ILP operations such as adding or removing a literal from a rule, adding a rule with an empty body or removing a rule.

SLIPCOVER is an extension of SLIPCASE which introduces the new ability to perform generative learning in the search space. SLIPCOVER still requires a target predicate, but it also gathers a set of good theories which can explain predicates from the BK other than the target predicate – this process can be viewed as a form of deep learning [4], since these intermediate theories will be used to explain the target predicate. The learning process of SLIPCOVER is similar to the SLIPCASE strategy, since it calculates, for each rule, possible refinements and their respective log-likelihood using a beam search strategy. Once this operation is completed, the algorithm's second stage consists of learning the parameters for the rules generated earlier. This is done by adding the non-target theories to the BK and learning a theory for the target w.r.t to them, which allows for more information to be incorporated in the final target theory, since it is using another level of abstraction due to the non-target theories that are now being used.

The SKILL system is a result of this work and it implements an exhaustive search algorithm based on traversing the AND search space, followed by traversing the OR search space. Since PILP's search space is exponential and traversing it is a computationally taxing task, SkILL supports several pruning strategies which can reduce the execution time and maintain quality of the final model, namely fitness pruning [13], estimation pruning [12] and prediction pruning. Fitness pruning selects populations of candidate rules and theories so that the exhaustive algorithm is reduced to polynomially bound complexity on user-defined parameters [13]. 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 [12]. Estimation pruning is aimed at reducing the execution time since it avoids probabilistic evaluation, but unlike the prediction pruning strategy introduced here, it is not safe – estimation pruning may discard the best theory, whilst this never happens when using prediction pruning. These three pruning strategies implemented in SkILL are however orthogonal to the system used and can be applied to any PILP system. Prediction pruning proposes a methodology whose aim is to improve the quality of the explored candidate models in a PILP search space. Unlike fitness and estimation pruning approaches, prediction pruning focuses on improving the quality of the search space. In doing so, it can direct the search towards more promising candidates which can lead to a reduction in execution time or an increase in predictive accuracy.

SkILL differs from SLIPCOVER because it does not introduce new probabilistic theories in the BK whose values can also be adjusted. However, SkILL's algorithm can take advantage of the probabilistic information of rules in order to avoid performing inference for every refinement that is generated using the language bias.

ProbFOIL+'s algorithm 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 is similar to SkILL's 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. An important difference between SkILL and ProbFOIL+ is that ProbFOIL+ introduces new negative examples from the constants and predicates appearing in the positive examples by taking advantage of the closed world assumption. Whilst this method can induce theories that are more specific (and therefore more descriptive), it also results in a combinatorial increase in the number of examples, as well as in an unbalanced prediction problem. This in turn can cause a large overhead in the execution time and possible issues finding informative theories if the classes become very unbalanced due to the added examples. Another relevant difference between the systems is that SkILL supports annotated disjunctions [55] as part of the PBK (neither SkILL nor ProbFOIL+ can induce annotated disjunctions). Annotated disjunctions can express mutually exclusive blocks of probabilistic clauses, as opposed to mutually independent probabilistic clauses, which is the standard representation in ProbLog.

Chapter 4

Pruning Strategies

This chapter introduces the three pruning strategies developed for PILP algorithms, starting out with an overview of how they can be used in the PILP search space traversal algorithms, and then proceeding to present each pruning strategy individually in detail.

4.1 Overview

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. The PILP search space can only build theories based on rules that were previously generated taking into account the language bias of the problem.

Traversing the PILP search spaces exhaustively is of exponential complexity with the size of the PBK (both literals and facts), as shown in Section 3.2.3, which requires significant computational resources as datasets grow larger. To address this issue, this thesis proposes three distinct ways to prune the PILP search space: *fitness pruning*, *prediction pruning* and *estimation pruning*. These pruning strategies are based on the probabilistic information of candidate theories and are 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, as shown in Algorithms 4.1 and 4.2.

Algorithm 4.1 $and\_search\_space\_with\_simplified\_pruning(PBK, PE)$

```
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 4.2 $or\_search\_space\_with\_simplified\_pruning(T_1, PBK, PE)$

```
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** $T_{All}$
```

Algorithms 4.1 and 4.2 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. 4.1 and lines 2 and 8 in Alg. 4.2, 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. 4.1 and lines 3 and 9 in Alg. 4.2, 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. 4.1 and line 5 in Alg. 4.2, respectively).

Each pruning strategy is presented next, in more detail.

4.2 Fitness Pruning

Fitness pruning consists of limiting the number of candidate theories in each iteration to a maximum *MaxLen*. The advantage of this 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. At the end of each iteration in the PILP algorithm, fitness pruning will select a limited number of candidate theories to be used for the next iteration, from all the theories that were evaluated at that stage. This is done by defining two sets – *Primary* and *Secondary*. The number of elements of these sets is such that:

$$|Primary| \times |Secondary| = MaxLen \tag{4.1}$$

For each stage of the AND and OR search spaces, fitness pruning first selects candidate theories for these sets up to their cardinality, and then new theories are generated by combining all members from each set using the respective AND or OR operation.

By default, 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 length for the *Primary* set *MaxLenP* and another independent maximum length for the *Secondary* set *MaxLenS*, such that the total number of candidates is calculated by multiplying the maximum length or both sets as follows:

$$MaxLen = MaxLenP \times MaxLenS \tag{4.2}$$

In the particular case where $|Primary| = |Secondary| = \infty$, fitness pruning corresponds to fully traversing the search space, and evaluating all possible combinations of theories for each iteration. Furthermore, in the case where $|Secondary| = \infty$, fitness pruning is equivalent to a beam search with beam width of $|Primary|$.

The selection of the rules/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 (root mean squared error)* or *PAcc (probabilistic accuracy)*. The selection is performed by ordering all candidate theories according to the chosen metric and populating the sets with the top theories. The *Primary* and *Secondary* sets can use different metrics, which gives the user more control over which candidates to favour when building the *Primary* and *Secondary* sets, when compared to beam search.

In addition to the loss functions, a *Random (stochastic selection)* metric can be used. The random metric introduces a stochastic selection of candidate 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 random metric solves this issue by combining stochastically selected candidates, as opposed to only *good ones* based on a deterministic metric. There may be cases when using a weaker candidate in a combination may actually result in a final theory of *better* quality. Depending on the given ranking metrics, fitness pruning can generate theories in a fully stochastic way, use best theories or create a heterogeneous mix. If a stochastic metric is chosen, the selection process is distinct for each iteration.

This procedure can be exemplified using the `advised_by` toy example presented in Chapter 3. For that example, rules of length one which can be generated (and their MAE and RMSE) are presented in Table 4.1.

If the defined maximum sizes for *Primary* and *Secondary* are then 2 and 3 respectively, and the chosen metrics MAE and RMSE respectively, the following sets would be generated:

$$\begin{aligned} Primary = \Big\{ &\Big(\texttt{co\_author(X, Y)}, 0.15\Big), \\ &\Big(\texttt{student(X)}, 0.28\Big)\Big\} \end{aligned} \tag{4.3}$$

Table 4.1: Rules of length one and their respective MAE and RMSE

$Rules_1$	MAE	RMSE
`ab(X, Y):- student(X)`	0.28	0.21
`ab(X, Y):- student(Y)`	0.58	0.51
`ab(X, Y):- professor(X)`	0.73	0.66
`ab(X, Y):- professor(Y)`	0.43	0.36
`ab(X, Y):- co_author(X, X)`	0.58	0.51
`ab(X, Y):- co_author(X, Y)`	0.15	0.03
`ab(X, Y):- co_author(Y, X)`	0.58	0.51

$$\begin{aligned} Secondary_1 = \Big\{ & \Big(\texttt{co\_author(X, Y)}, 0.03\Big), \\ & \Big(\texttt{student(X)}, 0.21\Big), \\ & \Big(\texttt{professor(Y)}, 0.36\Big)\Big\} \end{aligned} \tag{4.4}$$

Once the *Primary* and *Secondary*1 sets are populated, the rules for the next iteration will be generated from the members which are present in these sets only, resulting in the set of $Rules_2$ containing only three members.

$$\begin{aligned} Rules_2 = \Big\{ & \texttt{co\_author(X, Y), student(X)}, \\ & \texttt{co\_author(X, Y), professor(Y)}, \\ & \texttt{student(X), professor(Y)}\Big\} \end{aligned} \tag{4.5}$$

In this case, the $Rules_2$ set does not contain six members because there are repeated elements in the *Primary* and *Sedondary* sets. In general, candidate rules can also be excluded from the sets because they do not comply with the language bias of the problem, though this does not happen in the case of this example.

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[1] would be $\sum_{l=1}^{TMaxLength} \binom{|T_1|}{l}$, where

[1]A theory is evaluated probabilistically against the set of all probabilistic examples.

TMaxLength is the maximum theory length and T_1 is the set of theories of length one. It is easy to see that using sets of fixed size for each iteration, the number of probabilistic evaluations will be at most $|T_1| + (\mathit{TMaxLength} - 1) \times \mathit{PrimarySize} \times \mathit{SecondarySize}$, which is much less than all possible combinations of theories in the search space.

4.3 Prediction Pruning

The aim of prediction pruning is to allow the search to focus 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.

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. Depending on which search space is being explored, the criterion to exclude theories will differ. When two rules r^a and r^b are combined using logical conjunction (the AND operation), a more *specific* rule $r^{a,b} = r^a \wedge r^b$ will result. This is due to the fact that more literals in the body of the rule must succeed simultaneously so that the rule can be verified.

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

$$\begin{aligned} p_i(r^{a,b}) = \sum_{\omega_n \models l_i(r^{a,b})} P(\omega_n) = \\ \sum_{\substack{\omega_n \models l_i(r^a) \cap \\ \omega_n \models l_i(r^b)}} P(\omega_n) \end{aligned} \tag{4.6}$$

From Equation 4.6, it follows that, for an example i, the prediction value of a more specific rule $p_i(r^{a,b})$ will always be less than or equal to the prediction value of $p_i(r^a)$ and $p_i(r^b)$.

$$\begin{aligned} |\{\omega_n : \omega_n \models l_i(r^{a,b})\}| &\leq |\{\omega_n : \omega_n \models l_i(r^a)\}| \\ |\{\omega_n : \omega_n \models l_i(r^{a,b})\}| &\leq |\{\omega_n : \omega_n \models l_i(r^b)\}| \end{aligned} \tag{4.7}$$

In Equation 4.7, set theory principles state that the intersection of the worlds which entail $l_i(r^a)$ and the worlds which entail $l_i(r^b)$ will always be smaller than either set (in the case where $\{\omega_n : \omega_n \models l_i(r^a)\}$ and $\{\omega_n : \omega_n \models l_i(r^b)\}$ are the same, the intersection is equal to them). Therefore, the prediction value of rule $p_i(r)$ will be monotonically decreasing with the application of the AND operation, since in each iteration the rules become more specific (longer).

Having established this ordering allows prediction pruning to be applied over previously evaluated rules to determine whether they are useless for further combination, given some criterion. For a given example i, if the prediction value of a rule $p_i(r)$ is less than the example value e_i, then continuing to apply the AND operation can only result in distancing $p_i(r)$ 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 $p_i(t^{a;b})$ of $t^{a;b}$ for a given example i will be equal to the sum of the probabilities of each world $P(\omega_n)$ in the program in which $\omega_n \models l_i(t^a)$ or $\omega_n \models l_i(t^b)$, meaning the union of these sets of worlds. Similarly to rules, the prediction value of a more general theory $p_i(t^{a;b})$ can be defined in terms of the prediction values of more specific theories $p_i(t^a)$

and $p_i(t_b)$.

$$p_i(t^{a;b}) = \sum_{\omega_n \models l_i(t^{a;b})} P(\omega_n) = \sum_{\substack{\omega_n \models l_i(t^a) \cup \\ \omega_n \models l_i(t^b)}} P(\omega_n) \tag{4.8}$$

From Equation 4.8, it follows that the prediction values for a more general theory $t^{a;b}$ will always be greater than or equal to the prediction value of t^a and t^b.

$$\begin{aligned} |\{\omega_n : \omega_n \models l_i(t^{a;b})\}| &\geq |\{\omega_n : \omega_n \models l_i(t^a)\}| \\ |\{\omega_n : \omega_n \models l_i(t^{a;b})\}| &\geq |\{\omega_n : \omega_n \models l_i(t^b)\}| \end{aligned} \tag{4.9}$$

In Equation 4.9, set theory principles state that the union of the worlds which entail t^a and the worlds which entail t^b will always be greater than either set (in the case where $\{\omega_n : \omega_n \models t^a\}$ and $\{\omega_n : \omega_n \models t^b\}$ are the same, the union is equal to them). 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 (longer).

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.

4.4 Estimation Pruning

The aim of estimation pruning is to reduce execution time by preventing the probabilistic evaluation of theories which are not good models for the system and would thus be pruned away at a later stage according to some criterion. This pruning strategy 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 model according to the 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 stage of the algorithm. In the case of rules, this can be achieved for instance by decomposing a rule into two other rules of smaller length which are in themselves valid rules in the AND search space. For example the rule

```
co_author(X,Y):- advised_by(X, Y), student(X).
```

can be decomposed in

```
co_author(X, Y):- advised_by(X, Y).
co_author(X, Y):- student(X).
```

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[2]. 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 prediction value for an example i using a theory t is given by determining in how many worlds (of all possible worlds in the PBK) $l_i(t)$ is true. The challenge in estimating the value of a probabilistic evaluation knowing the values of the theories being combined lies in the fact that the *amount of overlapping* of the sets of worlds corresponding to those two theories is unknown before evaluation. If two theories are mutually exclusive (or disjoint) w.r.t. the PBK, then their overlap is null. On the other hand, if a theory is more specific than another, the former will cover a subset of the worlds covered by the latter. Theories can also be independent, meaning that the probability that one 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. Equation 4.10 shows the expression for the minimum boundary when using the

[2]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.

AND operation.

$$min\_bound_{AND} = max(0, p_i(t^a) + p_i(t^b) - 1) \tag{4.10}$$

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. The expression for the maximum boundary using the AND operation is given in Equation 4.11.

$$max\_bound_{AND} = min(p_i(t^a), p_i(t^b)) \tag{4.11}$$

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. Equations 4.12 and 4.13 show the expressions used to calculate the minimum and maximum boundaries for the OR operation, respectively.

$$min\_bound_{OR} = max(p_i(t^a), p_i(t^b)) \tag{4.12}$$

$$max\_bound_{OR} = min(p_i(t^a) + p_i(t^b), 1) \tag{4.13}$$

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 (*min* and *max* estimators in Fig. 4.1, respectively). The *center* estimator (*ctr* in Fig. 4.1) is the center of the estimation interval (halfway between *minimum* and *maximum*). Equations 4.14 and 4.15 show the expressions to calculate the *center* estimator for the AND and OR operations, respectively.

$$estimator\_center_{AND} = \frac{1}{2}(max(0, p_i(t^a) + p_i(t^b) - 1) + min(p_i(t^a), p_i(t^b))) \tag{4.14}$$

$$estimator\_center_{OR} = \frac{1}{2}(max(p_i(t^a), t_2(e_i)) + min(p_i(t^a) + p_i(t^b), 1)) \tag{4.15}$$

The *independence* estimator (*ind* in Fig. 4.1) assumes that theories t^a and t^b are independent and calculates the values of their combination accordingly. Equations 4.16 and 4.17 presents the expressions used to calculate the *independence* estimator for the AND and the OR operations, respectively.

$$estimator\_independence_{AND} = p_i(t^a) \times p_i(t^b) \tag{4.16}$$

$$estimator\_independence_{OR} = p_i(t^a) + t_2(e_i) - p_i(t^a) \times p_i(t^b) \tag{4.17}$$

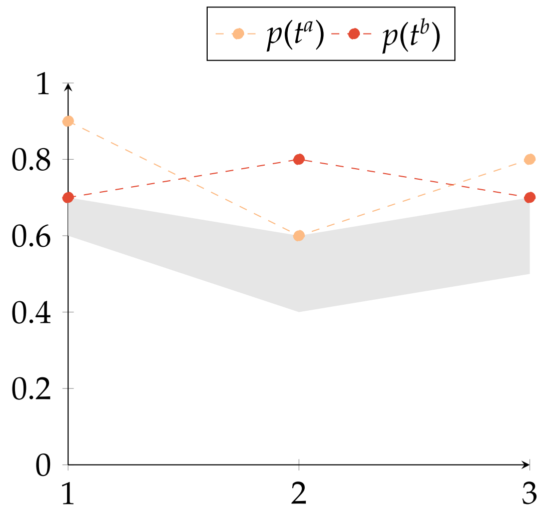
The *exclusion* estimator (not depicted in Fig. 4.1) assumes that the theories t^a and t^b are as exclusive as possible. In the AND operation, the *exclusion* estimator is equal to the *minimum* estimator, and in the OR operation, it is equal to the *maximum* estimator. Equations 4.18 and 4.19 show the expressions used to calculate the *exclusion* estimator for the AND and OR operation, respectively.

$$estimator\_exclusion_{AND} = max(0, p_i(t^a) + p_i(t^b) - 1) \tag{4.18}$$

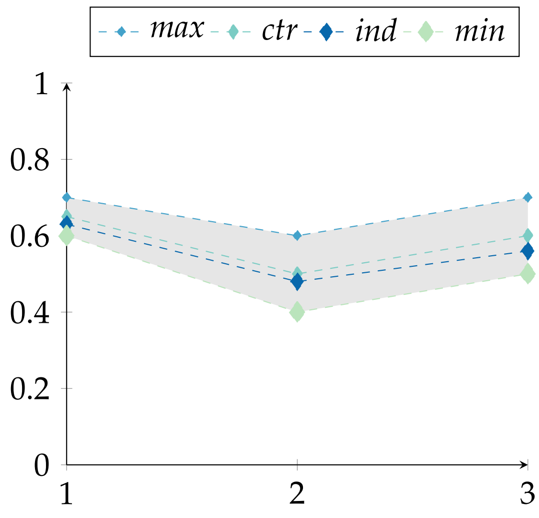
$$estimator\_exclusion_{OR} = min(p_i(t^a) + p_i(t^b), 1) \tag{4.19}$$

The performance of an estimator may vary according to the type of data contained in the PBK (independent data or mutually exclusive data, for instance).

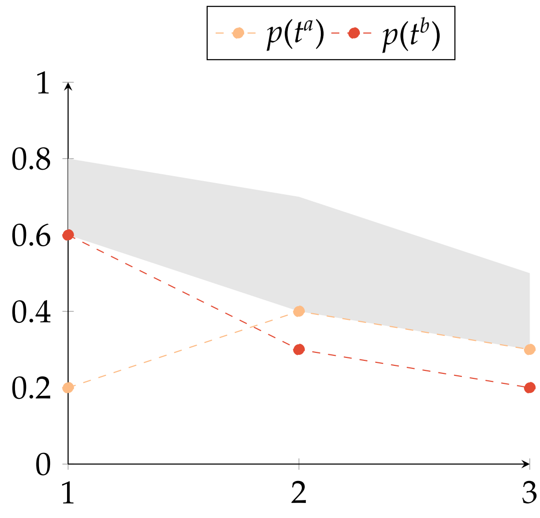
Figure 4.1(a) depicts the prediction values of two theories t^a and t^b (in the y-axis, ranging from 0 to 1), for three examples $i = \{1, 2, 3\}$ (x-axis). The shaded area represents the possible range for the prediction values of theories t^a and t^b when using the AND operation to combine them. The same shaded area is visible in Figure 4.1(b), where the estimation values are also plotted. The *max* and *min* estimation values correspond to the upper and lower bounds of the shaded area, meaning that they represent the extreme values of the possible prediction values for the combination of theories t^a and t^b. The estimator *ctr* lies halfway between the *max* and *min* prediction values, for each example, and the independent estimator *ind* assumes theories t^a and t^b are independent for all examples and estimates the value of the combination accordingly. This process is similar for the OR operation (Figures 4.1(c) and 4.1(d)) and only the location of the shaded area varies, now allowing for prediction values which are more general (higher in the y-axis) than theories t^a and t^b, contrary to the case in the AND operation, where the prediction values of the combination of theories t^a and t^b will be more specific than the theories.



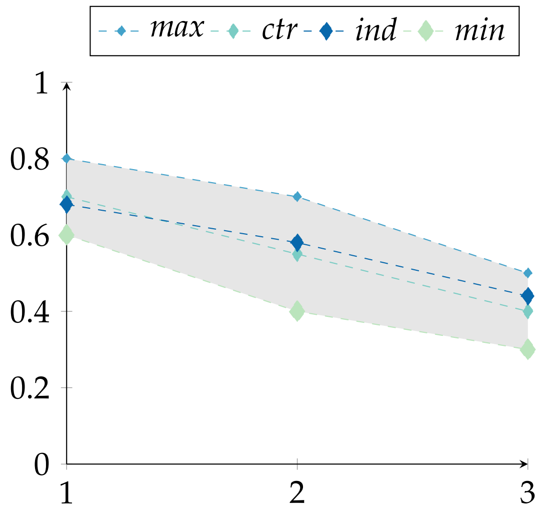
(a) Theories in AND operation



(b) Estimators in AND operation



(c) Theories in OR operation



(d) Estimators in OR operation

Figure 4.1: The x-axis contains three examples $i = \{1, 2, 3\}$ and the y-axis represents probabilistic values ranging from 0 to 1. Theories prediction values $p(t^a)$ and $p(t^b)$ are depicted as circles and estimators *min*, *max*, *ctr*, *ind* as diamonds.

4.5 Pruning Criteria

For both prediction and estimation pruning, a criterion is necessary to decide whether theories will be pruned away or not. Several criteria are possible, and this work proposes three criteria for deciding if a theory is too specify/general: 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 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. Equations 4.20 and 4.21 present the expressions for the hard pruning criterion for the AND and OR search space, respectively.

$$\exists i : p_i(t) < e_i \tag{4.20}$$

$$\exists i : p_i(t) > e_i \tag{4.21}$$

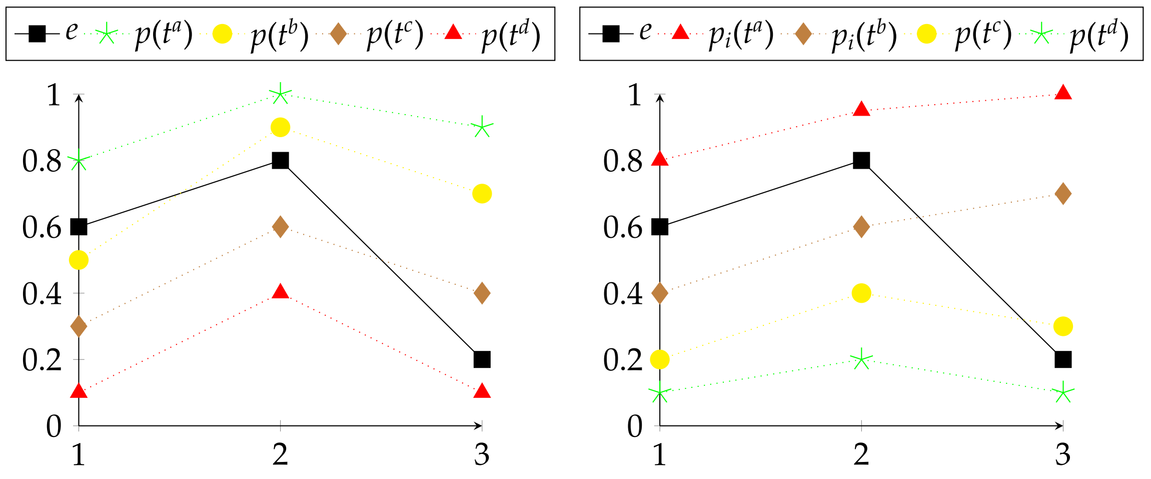
On the other hand, the soft pruning criterion takes into account the theory's predictions (or estimations) for every example, and only prunes the theory away if it is *overall* more specific (for the AND operation) or more general (for the OR operation) than the annotated values of the examples. Equations 4.22 and 4.23 present the expressions for the soft pruning criterion for the AND and OR search space, respectively.

$$\sum_i \left(p_i(t) - e_i\right) < 0 \tag{4.22}$$

$$\sum_i \left(p_i(t) - e_i\right) > 0 \tag{4.23}$$

The soft criterion differs from the hard criterion in that it takes into account the aggregate value of all examples, whilst the hard pruning criterion can discard theories based on one example value only. On the other hand, the safe pruning criterion excludes theories only when all of their predictions are found to be too specific (for the AND operation) or too general (for the OR operation), and no prediction can be improved by continuing with the search in that search space.

Figure 4.2 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. The best possible theory would predict exactly the same values as the annotated values, for every example.



(a) Pruning criteria for the AND search space (b) Pruning criteria for the OR search space

Figure 4.2: 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. Four theories' prediction values $p(t^a)$, $p(t^b)$, $p(t^c)$ and $p(t^d)$ are depicted in different colours and markers. Example values are depicted as black squares

In Fig. 4.2(a), for the AND operation, the safe pruning criterion would prune away t^d (red triangles) because, for every example, its prediction values are lower than the example values. The soft criterion prunes t^c (brown diamonds) and t^d (red triangles) away because their prediction values are overall lower than the example values. Finally, the hard pruning criterion would prune away all theories except t^a in Fig. 4.2(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 and higher prediction values. 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 (red triangles) and t^b (brown diamonds) away because their prediction values are overall higher than the example values. The hard pruning criterion would prune away t^a (red triangles), t^b (brown diamonds) and t^c (yellow 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.

4.6 Safeness

One concern of using pruning strategies is whether theories whose utility is high can be pruned away during the pruning process, and therefore the best theory 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. The safeness of a pruning strategy is defined in terms of the same evaluation metric used for the traversal of the search space. 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 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.

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 raking is 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.

Evaluation metrics introduced in Chapter 2 do not directly take into account whether the predictions of a theory are more or less specific than the example values, they just use a loss function to determine the utility of a theory. However, the degree of specificity of a theory can easily be assessed by considering the distance d_i between the example value e_i and the prediction of a theory for that example p_i. If this distance is a positive number, it means that the prediction of the theory is more specific (less general) than the example value. Conversely, if d_i is a negative number, the prediction of the theory is less specific (more general) than the example value. Therefore, specificity of a theory can be calculated by taking into account the positive and negative terms for the given evaluation metric.

$$\begin{aligned} MAE_{+}(theory) &= \sum_{i=1}^{\#E} \delta_{d_i>0}\, d_i \\ MAE_{-}(theory) &= \sum_{i=1}^{\#E} \delta_{d_i<0}\, d_i \\ MAE(theory) &= MAE_{+}(theory) + MAE_{-}(theory) \end{aligned} \tag{4.24}$$

$$\begin{aligned} RMSE_{+}(theory) &= \sum_{i=1}^{\#E} \delta_{d_i>0}\, d_i^2 \\ RMSE_{-}(theory) &= -\sum_{i=1}^{\#E} \delta_{d_i<0}\, d_i^2 \\ RMSE(theory) &= RMSE_{+}(theory) + RMSE_{-}(theory) \end{aligned} \tag{4.25}$$

Equations 4.24 and 4.25 show how to calculate the positive and negative terms for metrics MAE and RMSE, respectively (δ refers to the Dirac function used to select only the examples where the condition is verified). Determining whether a theory's predictions are more or less specific than the example values can be done by examining the positive and negative terms for the given evaluation metric. When the value of the positive term is equal to zero, it means that every prediction of the theory is equal to or more general than the example values. When the value of the negative term is equal to zero, then every prediction of the theory is equal to or more specific than the example values. If both terms are equal to zero, then the theory is predicting exactly the example values and it is the best possible theory in the search space.

Based on this information, and knowing that the AND operation only produces candidate theories which are increasingly more specific, when a theory is found to be more specific than the example values (according to the applicable Equation 4.24 or 4.25), then candidate theories generated from it will be necessarily equal or more specific. Therefore, it is safe to prune away these candidate theories, since they can never perform better according to the chosen evaluation metric. This is because the value of the negative term is equal to zero and cannot be reduced further, and the value of the positive term can only increase. Similarly for the OR operation, when the theory is found to be more general than the examples, it is safe to prune away candidate theories that can be generated from it, since they will only be even more general.

Therefore, a third safe pruning criterion can be defined for prediction pruning and operations AND (Eq. 4.26) and OR (Eq. 4.27) as follows:

$$\forall i : eval\_metric_{-}(p_i) = 0 \tag{4.26}$$

$$\forall i : eval\_metric_{+}(p_i) = 0 \tag{4.27}$$

The safe pruning criterion presented in Equations 4.26 and 4.27 defines a pruning condition which keeps all theories where at least one of its predictions has the potential to be improved, according to the given evaluation metric. This safe pruning criterion (for prediction pruning only) guarantees that no potentially better theories will be pruned away during the pruning process.

Chapter 5

The SkILL System

This chapter describes the SkILL PILP system. This system is publicly available at*bitbucket.org/joanacortereal/skill*. The general architecture of the system is explained, followed by a description of the algorithms for traversing the AND and OR PILP search spaces.

5.1 General Architecture

SkILL is a stochastic inductive logic learner which can extract readable relational rules from a database of probabilistic data. SkILL's rules are expressed as Horn clauses (a subset of FOL) and so they can be used to extract non-trivial knowledge about the dataset where they are inferred from. Additionally, these rules can be used for prediction since they output a value ranging from 0 to 1, which can be interpreted as a probability. Probabilistic values in SkILL can represent either statistical information or the degree of belief in a statement (using type I or type II probability structures [31], respectively). In more detail, SkILL's inputs are:

PBK Probabilistic background knowledge representing the basic information known about the problem. It can be composed of Horn clauses that can be annotated with probabilistic information known a priori. Clauses can be facts, rules or annotated disjunctions. If not annotated, it is assumed that their probabilistic value is 1. By default, PBK clauses follow the ProbLog semantics.

PE Probabilistic examples represent the observations that the system is attempting to explain. They also have probabilistic values a priori (1 if no probabilistic value is annotated). The best available theory will be the one that approximates the examples values with minimum error.

LB language bias describing the target predicate and the semantically valid combinations of literals. SkILL uses the TopLog engine [41] from the GILPS ILP system to generate all candidate ILP rules based on Mode-Directed Inverse Entailment [38].

Parameters which define the sizes of the Primary and Secondary populations for the AND and OR operations, which pruning strategies should be active and which criterion should be used for each one of them. If these parameters are not provided, the system uses default values.

The rule derivation process in TopLog is composed of three steps. In the first step, an example is chosen as such that it can be proven from the PBK and the language bias. For instance, `co_authors(joana,ines)` might be chosen as an example from which to derive rules. Each successful derivation yields a ground clause entailed by the language bias, which in this case would be

```
co_authors(joana, ines).
co_authors(joana, ines) :- student(joana).
co_authors(joana, ines) :- professor(ines).
co_authors(joana, ines) :- advised_by(joana, ines).
```

for rules containing at most one literal. In this way, both the derivation and the proof are kept. In the second step, the proof is re-ordered (the rules may not be generated in the correct order for variabilization) and finally a least general variabilisation is performed in the third step. This would result in

```
co_authors(X, Y).
co_authors(X, Y) :- student(X).
co_authors(X, Y) :- professor(Y).
co_authors(X, Y) :- advised_by(X, Y).
```

for the rules shown above. Applying this process to all examples in the dataset results in the set of all candidate rules that mirror patterns contained in the examples w.r.t. the PBK. As such, all rules which are considered will always entail at least one of the probabilistic examples in the dataset. SkILL improves on this approach by removing rules which are permutations of each other, for instance

```
co_authors(X, Y) :- student(X), professor(Y).
co_authors(X, Y) :- professor(Y), student(X).
```

are redundant and only one of these rules needs to be kept.

The aim of the SkILL tool is thus to find a theory in the valid search space which minimizes a *loss function* w.r.t. the given PE. There is no parameter learning performed during the construction of this theory and it cannot have disjunctive literals in the head. Because the examples in this setting are probabilistic values ranging from 0 to 1, the loss function can be defined in terms of the *distance* between the predictions of a theory (also ranging between 0 and 1) and the example values, as detailed in Chapter 3. SkILL supports both Mean Average Error (MAE) (or Probabilistic Accuracy, PAcc) and Root Mean Square Error (RMSE), as detailed in Chapter 2. The RMSE is the square root of the sum of the square differences between a theory's predictions and example values, thus penalising theories whose predictions are further from example values, when compared to PAcc. SkILL's outputs the FOL theory that minimizes the error between its predictions and the training examples. This theory can be seen as a:

Model the human-readable FOL theory which describes the contributions and dependencies between the most relevant literals for the given problem.

Classifier the theory can also be used for predicting probabilistic values for new instances of the target predicate using probabilistic inference.

The SkILL system runs on top of the Yap Prolog system [14], uses TopLog [41] as the basis rules generator and the ProbLog Yap library as its probabilistic inference engine. It differs from other PILP systems because it implements three pruning strategies which can be used separately or in combination with each other: fitness pruning, estimation pruning and prediction pruning. The SKILL system will explore the search space exhaustively when no pruning strategy is used.

5.2 Main Algorithm and Parameters

Fully exploring the PILP search space is equivalent to evaluating each theory in order to determine the best theory according to a given metric. This can be done in two steps: (i) exploring the AND search space, and (ii) exploring the OR search space.

Rules in the PILP setting are composed of conjunctive logical literals, and the length of a rule is equal to the number of literals present in its body. As such, a rule becomes more specific as its length increases. Longer rules that contain the literals of shorter rules cover equal or less probabilistic worlds. Theories in PILP can be formed either by a single rule or by a set of disjunctive rules, and their length is equal to the number of rules they contain. As such, all rules (of any length) are also theories of length one. Unlike rules, the longer a theory is (containing more disjunctive rules), the more general it is. Theories can thus be combined using either an AND or an OR operation, which correspond to the logical conjunction and disjunction of the rules in the theories, respectively. In the case of the AND operation, only single rules (theories of length one) can be combined, and the result is another theory of length one (e.g. combining theories *t(X):– p(X)* and *t(X):– q(X,Y)* using the AND operation would result in theory *t(X):– p(X), q(X,Y)*). The unification of variables between the literals is obtained from the specified language bias – and the length of the resulting rule is equal to the number of unique literals it contains (if the same literal is present in both of the combining rules, it only appears once in the resulting rule). Conversely, theories of any length can be combined using the OR operation, and the resulting theory's length is equal to the sum of the lengths of the combined theories (e.g. combining theories of length one *t(X):–r(X)* and *t(X):– s(X,Y)* using the OR operation would result in theory *t(X):– r(X) ; s(X,Y)* of length 2).

Thus, SkILL's algorithm is composed of two main steps: (i) building theories of length one (single rules) using the AND operation to traverse the AND search space, and (ii) building theories of length greater than one using the OR operation to traverse the OR search space. In step (i), single rules of increasing number of literals are built from the mode declarations using the AND operation. Once all possible rules are built and evaluated, the algorithm proceeds to step (ii) using the OR operation to combine single rules (theories of length one) into theories of greater length, up to a maximum length.

Figure 5.1 depicts the search space traversal procedure in SkILL, and Algorithms 5.1 and 5.2 detail the procedure for traversing the AND and OR search spaces, respectively. The *ParameterList* input contains a list of which pruning strategies are enabled. For prediction and estimation pruning strategies, a *PruneCriteria* can be given (can be hard, soft or safe), and for estimation pruning an *Estimator* can be provided. For fitness pruning, it is possible to select the *PrimarySize* and *SecondarySize*

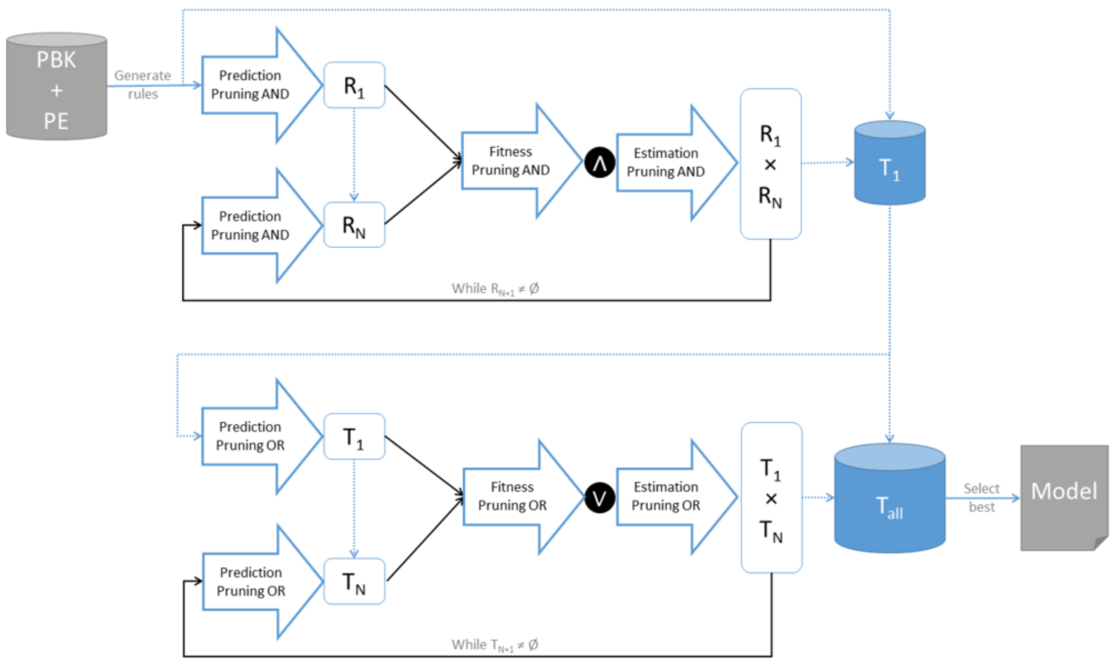


Figure 5.1: Search space traversal of SkILL algorithm

for the AND and OR operations separately, as well as the *RankingMetric* for each set of candidates. If a pruning strategy is enabled but no options are specified, SkILL uses soft as the pruning criterion, independence as the estimator, and 10/100 for primary and secondary set sizes and PAcc as the ranking metric, by default. The *ParameterList* variable also contains the maximum length of rules and theories *RMaxLength* and *TMaxLength*, which are 3 by default.

In Fig. 5.1, the set of rules of length one is depicted on the top left (corresponding to line 1 in Alg. 5.1). This set is comprised of all rules of only one literal obtained from TopLog and whose probabilistic evaluation will always be computed for efficiency reasons (lines 3–4 in Alg. 5.1). The AND search space algorithm receives as input PBK and PE to generate ILP rules. It is optional to also provide a list of pruning criteria and primary and secondary set sizes for fitness pruning. The first pruning operation to be applied (if enabled) is prediction pruning, since this method only requires the probabilistic evaluation of rules to be applied (lines 5–6 in Alg. 5.1). Once some of the initial rules are pruned away, fitness pruning can then be applied to the remaining set (lines 7–9 in Alg. 5.1). When fitness pruning is used, it determines the complexity of the execution, since it limits the amount of rules that will be combined to generate new candidate rules, which results in a polynomially bound number of probabilistic evaluations for every iteration. In the first iteration, rules of length one R_1 are combined to generate rules of length 2 (depicted in the top center

Algorithm 5.1 $and\_search\_space(PBK, PE, ParameterList)$

```
1: $R_{ILP} = generate\_ilp\_rules\_in\_TopLog(PBK, PE)$
2: $R_{Length} = 1$
3: $R_1 = select\_ilp\_rules\_of\_length(R_{Length}, R_{ILP})$
4: $T_1 = R_1 = R_N = prob\_evaluation(R_1, PBK, PE)$
5: if $enable(AND\_prediction\_pruning)$ then
6:    $R_1 = R_N = AND\_prediction\_pruning(R_1, PruneCriteria)$
7: if $enable(AND\_fitness\_pruning)$ then
8:    $R_1 = AND\_fitness\_pruning(R_1, PrimarySize)$
9:    $R_N = AND\_fitness\_pruning(R_1, SecondarySize)$
10: $R_{Length} = R_{Length} + 1$
11: $R_{N+1} = select\_ilp\_rules\_of\_length(R_{Length}, R_{ILP}) \cap combine\_rules(R_1, R_N)$
12: while $R_{N+1} \neq \emptyset$ do
13:    if $enable(AND\_estimation\_pruning)$ then
14:       $R_{N+1} = AND\_estimation\_pruning(R_{N+1}, R_1, R_N, PruneCriteria)$
15:    $R_{N+1} = prob\_evaluation(R_{N+1}, PBK, PE)$
16:    $T_1 = T_1 \cup R_{N+1}$
17:    if $enable(AND\_prediction\_pruning)$ then
18:       $R_{N+1} = AND\_prediction\_pruning(R_{N+1}, PruneCriteria)$
19:    if $enable(AND\_fitness\_pruning)$ then
20:       $R_{N+1} = AND\_fitness\_pruning(R_{N+1}, SecondarySize)$
21:    $R_N = R_{N+1}$
22:    $R_{Length} = R_{Length} + 1$
23:    $R_{N+1} = select\_ilp\_rules\_of\_length(R_{Length}, R_{ILP}) \cap combine\_rules(R_1, R_N)$
24: return $T_1$
```

of Fig. 5.1 and corresponding to lines 10–23 in Alg. 5.1). If estimation pruning is enabled, candidate rules can be pruned away based on the probabilistic values of the rules that were used to generate them (lines 13–14 in Alg. 5.1). After this point, rules of length two R_{N+1} are evaluated exactly, and prediction and fitness pruning are again applied to this set (arrows on top Fig. 5.1 and lines 15–20 in Alg. 5.1). If there are still rules to be evaluated, then R_{N+1} becomes R_N and a new iteration is performed (loop on the top in Fig. 5.1 and lines 21–23 in Alg. 5.1). Only those rules which are present in R_{ILP} and the combination of R_1 and R_N are kept for the next iteration (line 23 in Alg. 5.1). The combination of R_1 and R_N is the product of the sets, i.e. all members or R_1 are combined with all members of R_N. At the end, the AND search space procedure returns the set of evaluated rules of all lengths, which is also the set of all theories of length one (line 24 in Alg. 5.1).

Algorithm 5.2 *or_search_space*$(T_1, PBK, PE, ParameterList)$

```
1: $T_{All} = T_N = T_1$
2: **if** $enable(OR\_prediction\_pruning)$ **then**
3:     $T_1 = T_N = OR\_prediction\_pruning(T_1, PruneCriteria)$
4: **if** $enable(OR\_fitness\_pruning)$ **then**
5:     $T_1 = OR\_fitness\_pruning(T_1, PrimarySize)$
6:     $T_N = OR\_fitness\_pruning(T_1, SecondarySize)$
7: $T_{Length} = 1$
8: $T_{N+1} = combine\_theories(T_1, T_N)$
9: **while** $T_{Length} < TMaxLength$ **do**
10:     **if** $enable(OR\_estimation\_pruning)$ **then**
11:         $T_{N+1} = OR\_estimation\_pruning(T_{N+1}, T_1, T_N, PruneCriteria)$
12:     $T_{N+1} = prob\_evaluation(T_{N+1}, PBK, PE)$
13:     $T_{All} = T_{All} \cup T_{N+1}$
14:     **if** $enable(OR\_prediction\_pruning)$ **then**
15:         $T_{N+1} = OR\_prediction\_pruning(T_{N+1}, PruneCriteria)$
16:     **if** $enable(OR\_fitness\_pruning)$ **then**
17:         $T_{N+1} = OR\_fitness\_pruning(T_{N+1}, SecondarySize)$
18:     $T_N = T_{N+1}$
19:     $T_{Length} = T_{Length} + 1$
20:     $T_{N+1} = combine\_theories(T_1, T_N)$
21: **return** $T_{All}$
```

The OR search space traversal procedure shown in Fig. 5.1 is similar to the AND procedure except for the fact that it receives as input a set of theories of size one (input parameter in Alg. 5.2). It also receives as input the maximum theory length *TMaxLength* which is used in the stopping criterion condition for OR search space exploration. The set of theories of length one T_1 has already been computed during the AND stage of the algorithm and is comprised of all theories of only one clause (line 1 in Alg. 5.2). Similarly to the AND search space, prediction pruning and fitness pruning are applied first to ensure that the search space is limited to user-defined parameters (lines 2–6 in Alg. 5.2). In the first iteration of the OR stage, theories of length one T_1 are combined to generate theories of length 2. If estimation pruning is enabled, candidate theories can be pruned away based on the probabilistic values of the theories that were used to generate them. After this stage, theories of length two T_{N+1} are evaluated exactly, and prediction and fitness pruning is again applied to this set (arrows in bottom Fig. 5.1 and lines 12–17 in Alg. 5.2). If the maximum theory length is greater than two, then T_{N+1} becomes T_N and a new iteration is performed (lines 9–20 in in Algorithm 5.2). Once the stop criterion is met, the best generated theory for all different lengths is then returned (line 21 in Alg. 5.2).

At the end of execution, the SkILL algorithm provides (i) the best theory; (ii) the N best theories, with N selected from the user; and (iii) a list with the best theory of each iteration.

Chapter 6

Experiments

This chapter contains the experimental assessment of the pruning strategies and criteria developed during this thesis work. First, the PILP benchmarks used for assessment are described. Then, for each pruning strategy, experiments are performed using two PILP systems, SkILL and ProbFOIL+ [18], and several configurations are tried. Finally, a study of the three pruning strategies combined is presented, and the results are discussed.

6.1 Methodology and Benchmarks

The PILP SkILL system was used for most of the experiments presented in this chapter, and it runs on top of the Yap Prolog system [14], uses TopLog [41] 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 [18] 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 a 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 5-fold cross

Table 6.1: Dataset characteristics: number of 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		Size train		Size test	
metabolism(met)	230	7000	(46%)	184	(70%)	46	(30%)
athletes (ath)	721	4294	(100%)	576	(70%)	144	(30%)
breast cancer (bc)	130	13400	(3%)	104	(80%)	26	(20%)

validation unless otherwise indicated.

Table 6.1 summarises the characteristics of these benchmarks.

The **metabolism** dataset consists of an adaptation of the dataset originally from the 2001 KDD Cup Challenge[1]. 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:

$$p_{strength} = \frac{strength - min_{strength}}{max_{strength} - min_{strength}} \tag{6.1}$$

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[2]. 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

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

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 but instead have probabilistic mass in both, 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. This dataset is described in more detail in Chapter 7. 5 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.

6.2 Fitness Pruning

This section analyses the effect of fitness pruning on probabilistic accuracy and execution time using three different sizes for primary and secondary populations, for both AND and OR search spaces (the same sizes are used for both search spaces): 25/5, 25/10 and 25/20. The second number (5 in 25/5) corresponds to the secondary population, i.e. in each iteration five theories (or rules) of length N are chosen to populate this set according to a ranking metric based on probabilistic accuracy. The first number (25 in all cases) is the fixed number of rules and theories of length one that are combined against the secondary set. This corresponds to using beam sizes of 125, 250 and 500 candidates, respectively. The number of candidates for the populations was arbitrarily selected.

Table 6.2 shows the results running time in seconds, total number of evaluations performed and probabilistic accuracy for datasets **metabolism**(met), **athletes** (ath) and **breast cancer** (bc) and a varying secondary population size (5, 10 or 20) for both AND and OR operations. It also shows ProbFOIL+ results, which uses beam

Table 6.2: Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy on the test set for varying population sizes (same sizes used for AND and OR operation) and ProbFOIL+ results, for datasets **metabolism**, **athletes** and **breast cancer**. Standard deviation is presented in brackets.

	25/5	**25/10**	**25/20**	**ProbFOIL+**
	Execution Time (s)			
metabolism	2065 (111)	2552 (85)	3353 (228)	2008 (2254)
athletes	1715 (25)	3413 (469)	4610 (88)	57 (6)
breast cancer	779 (10)	1102 (76)	1449 (70)	3890 (379)
	No. Evaluations			
metabolism	1450 (43)	1683 (45)	2151 (49)	3734 (2603)
athletes	679 (6)	1142 (16)	1852 (28)	201 (48)
breast cancer	326 (38)	647 (66)	1235 (76)	24290 (951)
	Probabilistic Accuracy			
metabolism	0.67 (0.06)	0.67 (0.06)	0.67 (0.06)	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)

search to find the best theory, as a baseline. Each value in Table 6.2 is the average of the values for all five folds, and the standard deviation across folds is presented in brackets.

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. There is always a constant component for varying population sizes that is related to the number of rules of length one, which must be evaluated regardless of any pruning settings (1181 rules for **metabolism**, 53 rules for **athletes** and 25 rules for **breast cancer**). The effect of this component is evident on the execution time and evaluation number presented in Table 6.2, 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 the AND search since the language bias may restrict the rule search space from growing exponentially.

Even though fitness pruning only traverses part of the search space as shown by both the execution time and the number of evaluations in Table 6.2, it does not sacrifice probabilistic accuracy. This is due to the fact that rules and theories are ranked by a performance-related metric (probabilistic accuracy), 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). This is not the same process as testing these members for suitability for being combined using a given operation (as is done in prediction pruning).

Nonetheless, as the probabilistic accuracy does not decrease when the population sizes are reduced, it can be concluded that the ranking metric is effective in keeping a good population of candidate theories, and so the 25/5 pruning setting should be used, since it presents a shorter execution time for every case.

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

6.3 Estimation Pruning

This section presents two sets of experiments concerning estimation pruning which are aimed at assessing different aspects. Section 6.3.1 analyses the effect of applying estimation pruning to the benchmarks, using both SkILL and ProbFOIL+, for a given estimator (*independence*). The *independence* estimator was used in this experiment because it assumes there is no dependence between the probabilistic facts in the program, which can often be the case. Next, Section 6.3.2 studies the effect of using different estimators for estimation pruning (only in the SkILL system). The average time and probabilistic accuracy for different estimators, as well as for different pruning criteria, are evaluated in detail.

6.3.1 Estimation Pruning Performance

This experiment compares the performance of the independence estimator in the SkILL and ProbFOIL+ systems. For this experiment, five fold cross validation is used for all datasets. The estimation pruning strategy with an independence estimator was added to the ProbFOIL+ system, but 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.

Table 6.3 shows the running time in seconds, total number of evaluations performed and probabilistic accuracy (standard deviation is presented in brackets) for datasets **metabolism**(met), **athletes** (ath) and **breast cancer** (bc) and varying estimation pruning criteria (but always using independence estimator), using both SkILL (using 25/5 fitness pruning) and ProbFOIL+ (beam size of 5).

Results in Table 6.3 indicate that the use of estimation pruning can reduce the runtime of the experiments when compared to the results using no estimation pruning (see Table 6.2), in most cases. In the cases where this does not happen (for instance in the **breast cancer** benchmark using the ProbFOIL+ system), the number of probabilistic evaluations is still reduced, only 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 for any of the settings presented in Table 6.3, using the independence estimator.

6.3.2 Comparing Estimators

Different combinations of estimation pruning were tested in this section: only pruning the AND operation, only pruning the OR operation, and pruning both operations. The pruning settings are reported as a tuple where the first value is the AND pruning option and the second is the OR pruning option. For estimation pruning, pruning options can be soft pruning (S), hard pruning (H) or no pruning (x). For example, using this codification, xS stands for no AND pruning and soft OR pruning. For each configuration, several measurements were recorded for each dataset: execution time, probabilistic accuracy on the test set, and number of rules and theories pruned. Values reported are the average value across all folds. Next, experiments with different estimators were performed in the SkILL system. Unlike previous experiments, for the **metabolism** and **athletes** datasets, a number of n-times hold-out sets were made and all measurements were averaged out over the folds. In the **breast cancer** dataset, leave-one-out cross-validation was used.

Table 6.4 presents the speedups and ratio of probabilistic accuracy for the **metabolism**, **athletes** and **breast cancer** datasets, respectively, for the experiments performed using the SkILL system. All experiments in this section were performed using a

Table 6.3: Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy on the test set for varying pruning criteria and independence estimator, for datasets **metabolism**, **athletes** and **breast cancer**, with standard deviation in brackets. Execution times between systems are not comparable. The baselines are taken from Table 6.2, 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)
	No. 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)
	No. 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)

Table 6.4: Speedup and probabilistic accuracy ratio for three datasets

(a) metabolism

	Speedup						Probabilistic Accuracy Ratio					
Est	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**
min	1.45	1.47	-1.03	1.36	1.47	2.52	1.00	1.00	1.00	1.01	1.00	1.00
max	1.56	1.56	-1.09	1.94	1.61	5.66	1.00	1.00	1.00	1.00	1.00	-1.01
ctr	1.57	1.58	1.03	1.95	1.61	5.65	1.00	1.00	1.00	1.00	-1.01	-1.01
ind	1.35	1.33	-1.23	1.64	1.46	5.37	1.00	1.00	1.00	1.00	1.00	-1.01
exc	1.57	1.58	-1.04	1.95	1.58	5.69	1.00	1.00	1.00	1.00	1.00	-1.01

(b) athletes

	Speedup						Probabilistic Accuracy Ratio					
Est	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**
min	3.34	3.33	1.01	1.66	3.63	9.48	-1.06	-1.06	1.00	1.00	-1.11	1.00
max	3.35	3.35	2.12	2.19	12.40	49.72	-1.06	-1.06	1.00	1.00	-1.15	1.00
ctr	3.20	3.28	1.00	1.80	3.62	19.82	-1.06	-1.06	1.00	1.00	-1.08	1.00
ind	3.33	3.34	2.10	2.17	12.34	50.36	-1.06	-1.06	1.00	1.00	-1.15	1.00
exc	3.31	3.23	2.02	2.11	11.86	48.01	-1.06	-1.06	1.00	1.00	-1.15	1.00

(c) breast cancer

	Speedup						Probabilistic Accuracy Ratio					
Est	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**	**Sx**	**Hx**	**xS**	**xH**	**SS**	**HH**
min	7.09	7.18	1.42	1.41	22.44	21.46	1.09	1.09	1.00	1.00	1.09	1.09
max	7.16	7.06	1.65	1.63	25.24	23.49	1.09	1.09	1.00	1.00	1.09	1.09
ctr	7.04	6.98	1.63	1.63	25.25	24.80	1.09	1.09	1.00	1.00	1.09	1.09
ind	7.20	7.02	1.42	1.29	22.62	22.84	1.09	1.00	1.00	1.00	1.09	1.09
exc	7.19	7.19	1.63	1.62	25.00	24.80	1.09	1.09	1.00	1.00	1.09	1.09

25/5 fitness pruning setting as the base case setting. The speedup $\frac{B_t}{P_t}$ is calculated w.r.t. the B_t base case time (no pruning) for different P_t pruning options' execution time. If there is a slowdown, the *inverse* speedup $\frac{P_t}{B_t}$ is presented as a negative number. The ratio of the probabilistic accuracy $\frac{P_a}{B_a}$ is calculated for each probabilistic settings P_a w.r.t the probabilistic accuracy of the B_a base case. Similarly to the speedup, when the probabilistic accuracy decreases, the inverse of the ratio is given $\frac{B_a}{P_a}$ as a negative number. The absolute values for the independence estimator were presented above in Table 6.3.

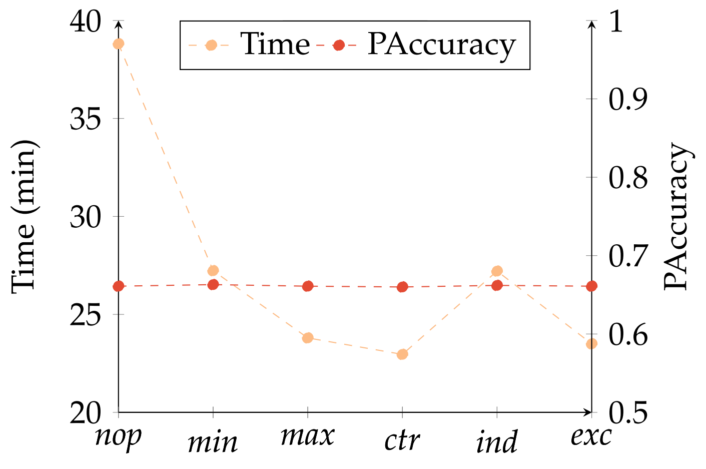
Figure 6.1 depicts the variation in execution time in minutes (left y-axis) and the variation in probabilistic accuracy (right y-axis) for all estimators in the **metabolism**, **athletes** and **breast cancer** datasets, respectively. The estimators analysed were the base case (no estimation pruning performed, or *nop*), *minimum* (*min*), *maximum* (*max*), *center* (*ctr*), *independence* (*ind*), and *exclusion* (*exc*). Each dataset's results will be discussed next.

For the **metabolism** dataset, results in Table 6.4(a) show that the greatest reduction in execution time is achieved by all estimators in the HH pruning setting. The xS pruning setting shows the slowest execution times with all estimators, except *center*, causing a slowdown. There is no significant reduction in probabilistic accuracy in any setting. Figure 6.1(a) shows that, overall, the probabilistic accuracy of the theories is unchanged and that the *maximum*, *center* and *exclusion* estimators can all reduce execution time from 40 to less than 25 minutes.

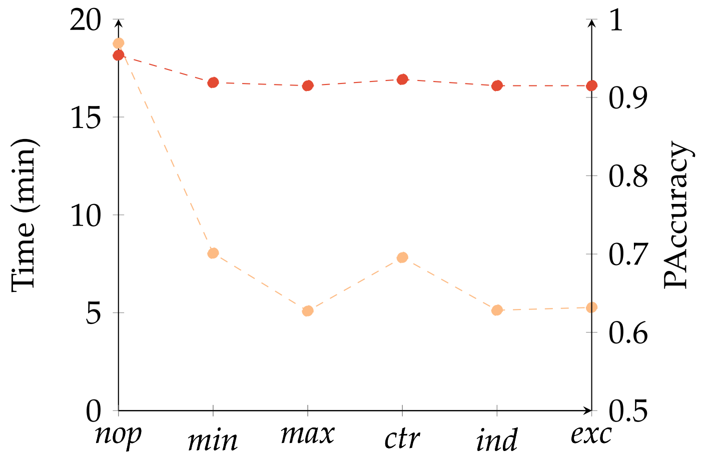
In the **athletes** dataset, again the HH pruning setting can reduce most execution time. However, the reduction using estimators *minimum* and *center* is much less than that of estimators *maximum*, *independence* and *exclusion*, where the execution is about 50 times faster (Table 6.4(b)). Estimators *minimum* and *center* are consistently slower in other pruning settings (xS, xH and SS), and the xS and xH settings present the lowest reduction in execution time in this dataset, of 2 times on average. Similarly to the other datasets, Table 6.4(b) shows that the probabilistic accuracy in the **athletes** dataset presents no significant reduction and, in particular, in the xS, xH and HH settings it is not reduced at all. Estimators *maximum*, *independence* and *exclusion* present the greatest overall reduction in execution time (Fig. 6.1(b)), from 20 to about 5 minutes, on average.

Results in Table 6.4(c) show that, in the **breast cancer** dataset, the greatest reduction in execution time can be achieved by using pruning in both the AND and the OR operations (SS and HH settings). The pruning settings that use only OR pruning (xS and xH) present more modest reductions of execution time (about 1.5 times) when compared to the settings that use only AND pruning (about 7 times). Although the OR operation has the potential to increase the probabilistic accuracy of true positives, it may also increase the accuracy of false positives. On the other hand, the AND operation, for this domain, works better, since it maintains the accuracy of true positives while decreasing the accuracy of false positives, when combining literals in a theory. The predictive accuracy of the best theory in this dataset never decreases, and in some settings (Sx, Hx, SS and HH in Table 6.4(c)) even increases slightly. This effect is due to a reduction in overfitting caused by the exclusion of some theories that are better on the training set but perform worse on the test set. Figure 6.1(c) shows that, on average, the *maximum*, *center* and *exclusion* datasets can reduce execution time from over 4 minutes to about 1 minute.

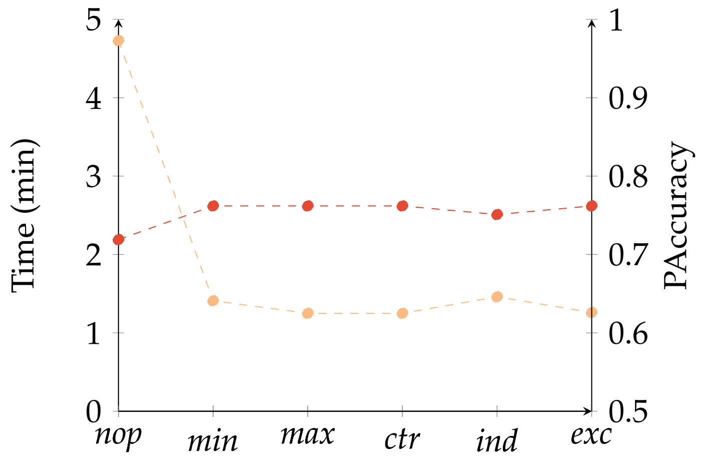
Finally, Table 6.5 presents the number of probabilistic evaluations performed for



(a) metabolism



(b) athletes



(c) breast cancer

Figure 6.1: Average time (in minutes) and probabilistic accuracy in all three datasets, for the base case (*nop*) and the five estimators. Values for each estimator are the average of its result over the pruning options.

Table 6.5: Number of single rules/theories evaluated for all datasets

(a) metabolism

Est	xx	Sx	Hx	xS	xH	SS	HH
min	5262/1982	1327/1986	1327/1986	5261/1983	5262/1186	1327/1985	1327/1229
max	5262/1982	1327/1986	1327/1986	5262/1758	5262/0	1327/1865	1327/104
ctr	5262/1982	1327/1986	1327/1986	5261/1957	5262/0	1327/1965	1327/104
ind	5262/1982	1327/1985	1327/1985	5261/1792	5261/0	1327/1866	1327/106
exc	5262/1982	1327/1985	1327/1985	5261/1759	5261/0	1327/1865	1327/106

(b) athletes

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

(c) breast cancer

Est	xx	Sx	Hx	xS	xH	SS	HH
min	1919/1988	181/550	181/550	1919/0	1919/0	181/0	181/0
max	1919/1988	181/550	181/550	1919/0	1919/0	181/0	181/0
ctr	1919/1988	181/550	181/550	1919/0	1919/0	181/0	181/0
ind	1919/1988	181/550	1919/0	1919/0	1919/0	181/0	181/0
exc	1919/1988	181/550	181/550	1919/0	1919/0	181/0	181/0

each pruning setting and estimator. The first number corresponds to single rules (theories of length one) evaluated, and thus the reduction is caused by AND pruning. Similarly, the second number in each cell is the number of theories of length greater than one, and its reduction is caused by OR pruning. The greatest reductions correspond to the HH setting, and are consistent with the settings in Table 6.4 that presents the greatest speedups. For the **athletes**, the three fastest estimators (in average) from Fig. 6.1(b) are also the estimators that in Table 6.5(b) prune away most theories. In particular, the number of theories pruned away during OR pruning is significantly lower for estimators *max*, *ind* and *exc* when compared to estimators *min* and *ctr*. The same trend can be observed in the **metabolism** and **breast cancer** whose results are presented in Tables 6.5(a) and 6.5(c), respectively.

6.4 Prediction Pruning

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

Because exploring the search space exhaustively is computationally taxing, the quality of candidate theories was assessed in a limited resource setting. Resources can be limited in two ways: either a timeout is imposed or a maximum number of evaluations is defined, which corresponds to using beam search (fitness pruning setting in the SkILL PILP system). To this effect, the impact of prediction pruning was assessed by comparing the AND and OR search spaces that are evaluated without pruning with those which are evaluated in a pruning setting, given the same limitation of resources. All experiments use five-fold stratified cross validation and results presented are the average values for all folds (Sa stands for safe pruning criterion).

Prediction pruning results in Table 6.6 show that applying the Soft or Hard strategies (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 fitness pruning results in Table 6.2. The effect of prediction pruning is more evident for ProbFOIL+ because it selects fewer candidates in each iteration, when compared to the SkILL's primary and secondary populations. It is therefore more important that bad candidates are pruned such that the limited beam is filled with better candidates. The prediction pruning strategy is thus particularly useful when traversing the search space with a narrow beam, so that the candidates selected to populate it are of greater predictive value when compared to using no prediction pruning. Safe pruning has no effect on these datasets because it's pruning power is too limited.

Table 6.6 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

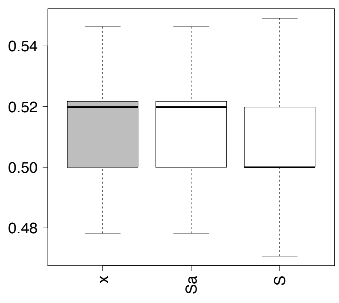
Table 6.6: Execution time in seconds, number of probabilistic evaluations performed and probabilistic accuracy for datasets **metabolism**, **athletes** and **breast cancer** using the SkILL and ProbFOIL+ systems with prediction pruning for the AND search space. Standard deviation is presented in brackets. Execution times between systems are not comparable. The baselines are taken from Table 6.2, setting 25/20 for SkILL.

(a) SkILL

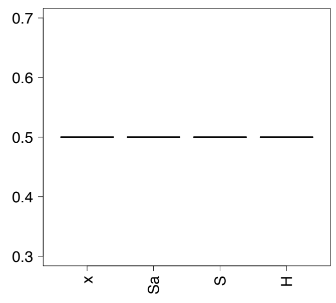
	Baseline	**Safe**	**Soft**	**Hard**
	Execution Time (s)			
metabolism	3353 (204)	2286 (185)	3216 (472)	1791 (37)
athletes	4610 (79)	4230 (582)	2322 (164)	2358 (73)
breast cancer	1449 (63)	616 (50)	636 (26)	353 (42)
	No. Evaluations			
metabolism	2151 (44)	2150 (44)	3234 (90)	2103 (37)
athletes	1852 (25)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

(b) ProbFOIL+

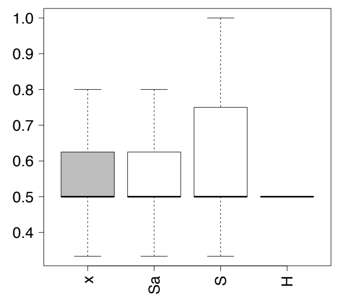
	Baseline	**Safe**	**Soft**	**Hard**
	Execution Time (s)			
metabolism	2008 (2016)	1999 (2019)	752 (215)	464 (71)
athletes	57 (5)	57 (5)	55 (4)	14 (0)
breast cancer	3890 (339)	3828 (302)	8093 (2101)	725 (38)
	No. Evaluations			
metabolism	3734 (2328)	4549 (3734)	4518 (1493)	2452 (492)
athletes	201 (43)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



(a) metabolism

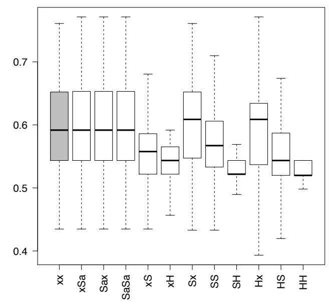


(b) athletes

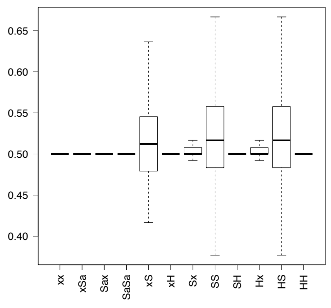


(c) breast cancer

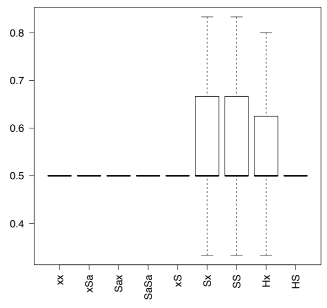
Figure 6.2: Boxplots showing the distribution of theories' AUCs for the AND search space for three datasets (baseline in grey).



(a) metabolism



(b) athletes



(c) breast cancer

Figure 6.3: Boxplots showing the distribution of theories' AUCs for the OR search space, for three datasets (baseline in grey).

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.

For the SkILL experiments, the AUC of all rules containing more than one literal (AND search space) and all theories (OR search space) was calculated. The AUC of rules composed of only one literal was not considered because prediction pruning has no effect on these rules, which must always be explored. Analysing the distribution of the AUC values is relevant because if the upper quartiles of the distribution are improved, this shows that there are better candidate members selected to be explored given limited resources. Lower quartiles will naturally be discarded by the PILP algorithm's metric to select the best final theory. The distribution of these values for each setting and search space are presented in Figs. 6.2 and 6.3 for the AND and OR search spaces, respectively. Each box depicts percentiles 0 and 100 (the lower and upper whiskers, respectively), percentiles 25 and 75 (lower and upper box boundaries, respectively), and the percentile 50 (median) using a bold line. In cases where a search space is not generated for any fold there is no boxplot in Figs. 6.2 and 6.3.

Figures 6.2 and 6.3 present the combined distribution of the AUCs of theories produced in all folds for the AND and OR search spaces, respectively. The higher the AUC value (y-axis), the greater the predictive power of a theory. Each boxplot corresponds to a pruning setting in the AND or OR search space: the boxes labelled with only one pruning criterion refer to the AND search space, whilst the boxes labelled with two pruning criteria refer to the OR search space.

For the AUC distributions, statistical significance is also calculated (using non-paired two-tailed t-test where the null hypothesis is that the distributions are the same) by comparing the distribution of AUCs fold to fold (e.g. fold 1 using soft OR prediction pruning against fold 1 without pruning). An upaired test was used because, even though the comparison is made fold to fold, the same fold can have a different number of candidate theories in different pruning settings.

Table 6.7 reports the number of folds where the results were statistically significant for both the AND and the OR search spaces, and all benchmarks. In some cases, some folds do not produce an AND or OR search space because all theories are pruned away, and this is the cause for not always reporting five folds in comparison. Table 6.7 shows that the safe pruning criterion causes no significant difference is candidate theory predictive quality, both for the AND and the OR operation (lines 1–2 in Table 6.7(a) and lines 1–4 in Table 6.7(b)). This is due to the fact that the safe pruning criterion is the least aggressive criterion and therefore the proportion of candidates that are pruned in this setting is limited. On the other hand, both soft and hard pruning criteria cause a significant difference in the AUC distribution of candidates, in particular for the OR operation, where most folds present a significant difference (lines 5–12 in Table 6.7(b)). Aggressive criteria coupled with the AND pruning operation do not cause such a significant difference in the distribution, in particular for the **athletes** dataset. This happens because the predictive power of rules in this benchmark is similar among candidates, and so even though different rules can be selected, this is not reflected in the distribution of AUC values. In cases

Table 6.7: Number of significant differences (left) for the number of tested folds (right) in the AND and OR AUC distributions for different prediction pruning settings and benchmarks **metabolism**, **athletes** and **breast cancer**

(a) AND search space

Setting	**metabolism**	**athletes**	**breast cancer**
x	0/4	0/5	0/5
Sa	0/4	0/5	0/5
S	4/4	0/5	3/5
H	–	0/5	1/4

(b) OR search space

Setting	**metabolism**	**athletes**	**breast cancer**
xx	0/5	0/5	0/5
xSa	0/5	2/5	0/5
Sax	0/5	2/5	0/5
SaSa	0/5	2/5	0/5
xS	4/4	4/5	0/5
xH	4/4	4/5	–
Sx	2/5	3/5	2/5
SS	5/5	3/5	3/5
SH	5/5	4/5	–
Hx	5/5	3/5	5/5
HS	3/4	4/5	4/4
HH	1/1	4/5	–

where a search space is not generated for any fold there is no boxplot in Figs. 6.2 and 6.3, and no value reported in Table 6.7.

Quality of the AND search space (Fig. 6.2) is only significantly improved in the **breast cancer** benchmark, using soft prediction pruning. However, the candidate rules that are selected for the AND search space impact the OR search space, since candidate theories will be selected from the rules that were previously explored. As such, even though the AND search space only shows direct impact from using prediction pruning in the **breast cancer** benchmark, it indirectly impacts the candidate theories available for the OR search space in the other benchmarks. This is particularly relevant for the **athletes** dataset, where the quality of the OR search space is affected by soft and hard AND pruning. For instance, setting SS performs significantly better when compared to setting xS, and setting Hx's 50 and 100 percentiles are higher than its counterpart setting xx. This effect is also visible in the **breast cancer** benchmark, where the settings using soft or hard AND prediction pruning present the greatest improvement. In most cases where the quality of the OR search space increased, AND prediction pruning had previously been applied to the AND search space.

In Fig. 6.3 it is visible that prediction pruning can improve the quality of the OR search space, particularly in the case of the **breast cancer** and the **athletes** benchmarks. In the **breast cancer** benchmark the two upper quartiles of the AUC distribution are clearly improved in three settings. This trend is also clear in the **athletes** dataset, where again prediction pruning significantly increases the predictive quality of the evaluated theories in three cases (and slightly in two other settings). On the **metabolism**dataset, the improvement due to prediction pruning is not as evident, but it is noteworthy that there is in fact a slight increase in the maximum AUC value for the case of no OR pruning and hard AND pruning, as well as in all safe pruning settings. The boxplots with range zero indicate that in those settings the candidates that populate the beam do not have any predictive power in the test set. However, this does not imply a loss in predictive accuracy of the optimal model since rules of only one literal are not included in these boxplots because they are not affected by prediction pruning.

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

These results show that prediction pruning maintains the predictive quality of the generated models. Prediction pruning impacts the distribution of the predictive quality of theories and that the use of prediction pruning can shift the maximum value and upper quartile of the distribution upwards, thus indicating improved candidate theory quality.

6.5 Combining Pruning Strategies

Fitness pruning focuses on limiting the number of candidates. Estimation pruning is aimed at reducing the execution time since it eliminates probabilistic evaluations, which is where the greatest amount of time during execution is spent. Conversely, prediction pruning focuses on ensuring that all rules or theories that proceed to the next iteration are viable candidates for combination using that operation, thus focusing on the quality of the theories combined.

Pruning options for prediction and estimation pruning allow parameters to be set for one operation only, 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 pruning is set for at least two operations (e.g. soft prediction pruning in both AND and OR operations), and cases where both estimation and prediction pruning are used (e.g. soft prediction pruning and hard estimation pruning using the independence estimator). This results in 9 combinations for each fitness pruning size selection, where several measurements were recorded for each of the folds: number of rules pruned, number of theories pruned, accuracy on the test set, and execution time. For each dataset, all measurements were averaged out over the folds and standard deviation is presented in brackets. As before, in what follows, only the 25/5 fitness pruning setting is reported.

Table 6.8 presents the pruning configurations for each setting, as well as the average predictive accuracy on the test set. Table 6.9 presents the average number of probabilistic evaluations performed and the execution time (in seconds), followed by the standard deviation in brackets. All settings were tested for datasets **metabolism**(met), **athletes** (ath) and **breast cancer** (bc), using 25/5 fitness pruning for both the AND and OR operation.

Results in Table 6.8 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 6.9) is greatly reduced, especially in the case of the **athletes** and medical benchmarks. 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

Table 6.8: Pruning configuration for each setting (EA, PA, EO and PO stand for AND estimation pruning, AND prediction pruning, OR estimation pruning and OR prediction pruning, respectively) and average probabilistic accuracy on test set for datasets **metabolism**(met), **athletes** (ath) and **breast cancer** (bc), using 25/5 fitness pruning.

Configuration					**Accuracy**		
Setting	**EA**	**PA**	**EO**	**PO**	**met**	**ath**	**bc**
base case (xx)	o	o	o	o	0.67 (0.06)	0.95 (0.01)	0.85 (0.03)
(E=x, P=S)	x	S	x	S	0.67 (0.07)	0.95 (0.01)	0.84 (0.03)
(E=x, P=H)	x	H	x	H	0.67 (0.06)	0.95 (0.01)	0.86 (0.04)
(E=S, P=x)	S	x	S	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	S	S	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=x)	H	x	H	x	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=S)	S	S	S	S	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)
(E=H, P=H)	S	H	H	H	0.67 (0.07)	0.95 (0.01)	0.86 (0.04)

Table 6.9: Average number of evaluations and average execution time (in seconds) for datasets **metabolism**(met), **athletes** (ath) and **breast cancer** (bc), using 25/5 fitness pruning.

Setting	**No. Evaluations**			**Execution Time**		
	met	**ath**	**bc**	**met**	**ath**	**bc**
base case (xx)	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)

with the shorter execution time for all settings where only estimation pruning is used (lines 4 and 7 in Table 6.9). 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**benchmark in the soft prediction pruning setting (second line and fourth column in Table 6.9).

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** benchmark, when there is a transition from soft to hard prediction pruning while using hard estimation pruning (lines 8 and 9 in Table 6.9).

Table 6.10 presents the number of rules and theories pruned during the estimation and evaluation pruning for varying pruning settings. This table presents only results for the athletes benchmark and 25/5 fitness pruning. Other benchmarks and fitness pruning options present a similar relation between pruning values.

Table 6.10: Number of rules and theories pruned during estimation and prediction pruning for each pruning setting in **athletes** (ath) dataset.

Setting	**No. Pruned Rules**		**No. Pruned Theories**	
	Estimation	**Prediction**	**Estimation**	**Prediction**
base case (xx)	–	–	–	–
(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 6.10 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 6.10). 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. Figure 6.4 shows average execution time (speedup) compared to the base case (of using no pruning), in logarithmic scale for the three benchmarks using 25/5 fitness pruning.

Results in Fig. 6.4 show that the two fastest pruning settings are obtained when hard estimation pruning is coupled with off or soft prediction pruning (blue -△- and yellow -□- lines). Settings with only prediction pruning enabled underperform timewise when compared to other settings (grey -▲- and pink -■- lines). The second fastest pruning configuration is to use hard estimation and prediction pruning (violet -◇- line), which is even slightly faster in **breast cancer** and a close second in **metabolism**. 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.

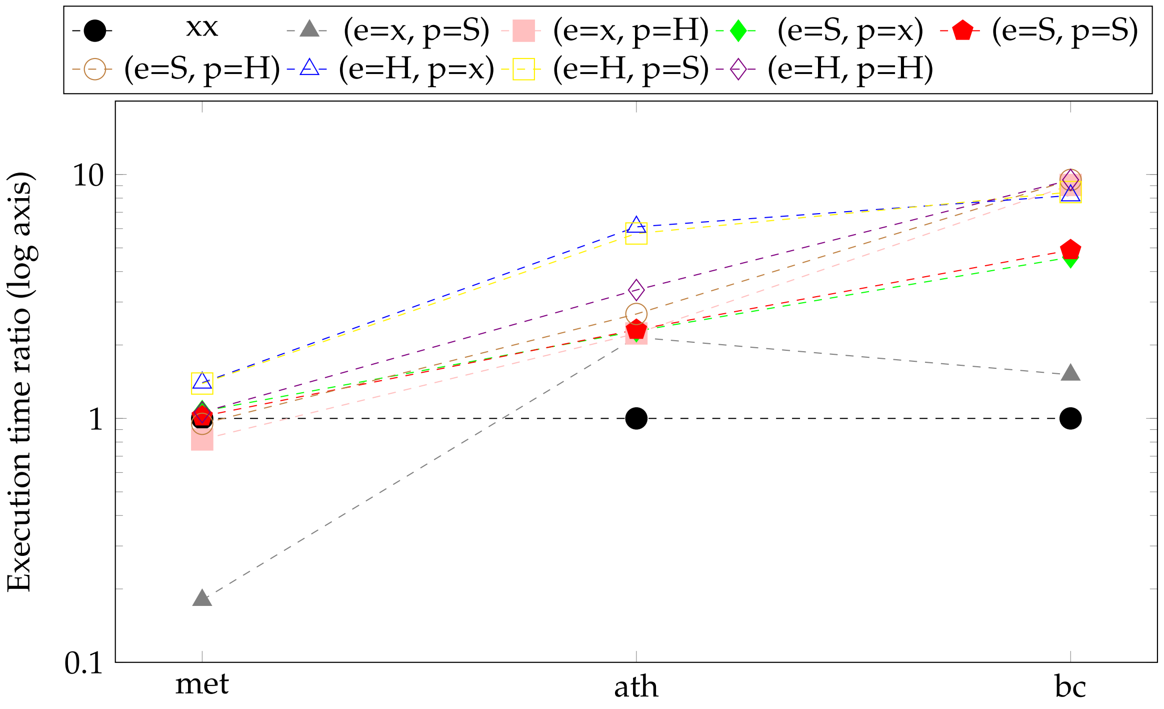


Figure 6.4: Average execution time ratio w.r.t. the base case off for datasets **metabolism** (met), **athletes** (ath), and **breast cancer** (bc), using 25/5 fitness pruning, grouped by setting (off in black, other settings in colour). Execution time ratio is plotted using a logarithmic y axis.

Chapter 7

Real World Application

Breast cancer is one of the most common forms of cancer and 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 [5]. In particular, the dataset used in this work consists of demographic-related variables and information about the biopsy procedure and BI-RADS (Breast Imaging Reporting and Data System) [6] annotations, as well as domain knowledge annotated both prospectively and retrospectively by experts of three different areas: mammography, biopsy surgery and biopsy pathology. Using an automated decision support system is conducive to rigorous and accurate risk estimation of rare events and has the potential to enhance clinician decision-making and provide the opportunity for shared decision making with patients in order to personalize and strategically target health care interventions.

Relational learning in the form of ILP (without probabilities) has been successfully used in the field of breast cancer. Burnside et al. [16] uncovered rules that showed high breast mass density as an important adjunct predictor of malignancy in mammograms. Later, using a similar dataset, Woods et al. [58] validated these findings performing cross-validation. In another work, Davis et al. [15] used SAYU, an ILP system that could evaluate rules according to their score in a Bayesian network, in order to classify new cases as benign or malignant. Results for a dataset of around 65,000 mammograms consisting of malignant and benign cases showed ROC areas slightly above 70% for recall values greater than 50%. Dutra et al. [23] showed that the integration of physician's knowledge in the ILP learning process yielded better results than building models using only raw data.

This chapter extends this work by applying PILP to create a decision support system targeted to this breast cancer setting. Contrary to other decision support systems well-known in the literature (for example, Bayesian-based or SVM-based), PILP combines probabilistic data with first order logic in order to produce both probabilistic outputs and human interpretable rules. The proposed setting includes experts' domain knowledge as (i) probabilistic rules in the background and (ii) probabilistic target values for examples.

7.1 Methodology

The dataset used for this experiment contains 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. The data was 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. For all of these 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). In the first instance, 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 who meet to discuss and reach an agreement about each case. Thus, the target probabilities of examples represent the perceived 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. This probabilistic value was then added to the probabilistic examples and a sample of the PE is presented next:

Each example is a patient case and the three examples in Fig. 7.1 are part of the PE

```
0.10::is_malignant(case1).
0.15::is_malignant(case2).
0.01::is_malignant(case3).
```

Figure 7.1: Probability annotations for Probabilistic Examples

used in this experiment (one per line). The argument for each example corresponds to a particular case (`case1`, `case2`, or `case3`) and the probability annotated in the beginning of the line represents the chance of malignancy for this case (10% for `case1`, 15% for `case2`, and 1% for `case3`).

Regarding the domain knowledge incorporated in the PBK, breast cancer literature values were used to complement the information on the characteristics of masses, since physicians rely on these values to perform a diagnosis. 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 [6]). Figures 7.2, 7.3, and 7.4 show how these variables are encoded in the PBK, (the notation is `probability_value::relation(...)...`). Figure 7.2 encodes the probabilistic information regarding mass shape obtained from the literature. There are three possible rules, each one applicable to a particular kind of shape (oval, round, or irregular). A rule of this type can be read as *IF this Case has a Mass AND the Mass is of type Shape THEN this feature exists with probability P*. The probability value annotated in each rule is the frequency with which a mass whose shape is of that type is malignant. Independent rules such as the ones presented in Fig. 7.2 are not mutually exclusive. 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. 7.3 also encodes independent rules, each for a characteristic of the mass margin. In this case it becomes obvious that both the microlobulated and spiculated margins have a high correlation with malignancy in the literature, given their high probability of malignancy (70% and 90% respectively).

Figure 7.4 differs from Fig. 7.2 and Fig. 7.3 in that it encodes three mutually exclusive possibilities for the mass density: low, equal, or high (note the new operator ";" for

```
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).
```

Figure 7.2: Probabilistic information from the literature regarding mass shape

disjunction). The probability of malignancy from the literature is encoded in the top three lines, which can be read as *IF the density of Mass is low, the probability of malignancy is 5%; ELSE IF the density of the Mass is equal, the probability of malignancy is 10%; ELSE IF the density of the Mass is high, the probability of malignancy is 50%*. The density rule is then constructed based on the mutual exclusivity introduced by the `density/1` fact above.

PILP models produce classifiers which are composed by a set of FOL rules, learnt automatically from the data, that represent a disjunctive explanation to the target predicate being learned. Figure 7.5 presents an example of a PILP model 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* [35], and it takes into account the mutual dependency between all the probabilistic literals in the model.

The experiment presented in this work aims at demonstrating that it is possible to use the probabilistic data to build a model that not only obtains good predictive accuracy, but also presents a human-interpretable explanation of the factors that affect the system in study. This model is learnt automatically from the data. In the medical domain it is crucial to represent data in a way that experts can understand

```
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).
```

Figure 7.3: Probabilistic information from the literature regarding mass margin

and reason about, and as such ILP can successfully be used to produce such models. Furthermore, PILP allows for incorporating in the PBK the confidence of physicians in observations and known values from the literature.

7.2 Experiments

The PILP SkILL system [13] was used for these experiments. It runs on top of the Yap Prolog system [14] and uses TopLog [41] as the basis rules generator and the ProbLog Yap library as its probabilistic inference engine. This system was selected because it can perform exhaustive search over the theory search space. Since this is a small dataset, exhaustive search is possible. However, if the dataset were larger there might be scalability issues in using exhaustive search, and so either SkILL with pruning strategies [12] or another PILP system whose search engine is greedy could be used instead (such as ProbFOIL+ [18] or SLIPCOVER [3]). In this experiment, 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

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

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

Figure 7.4: Probabilistic information from the literature regarding mass density

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

Figure 7.5: A PILP model for the target predicate `is_malignant/1`

Support Vector Machine (SVM), a Linear Regression (LREG), and a Naive Bayes classifier (NB). The *scikit-learn* python library [42] was used to perform the pre-processing of these experiments for the three non-relational methods. 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.

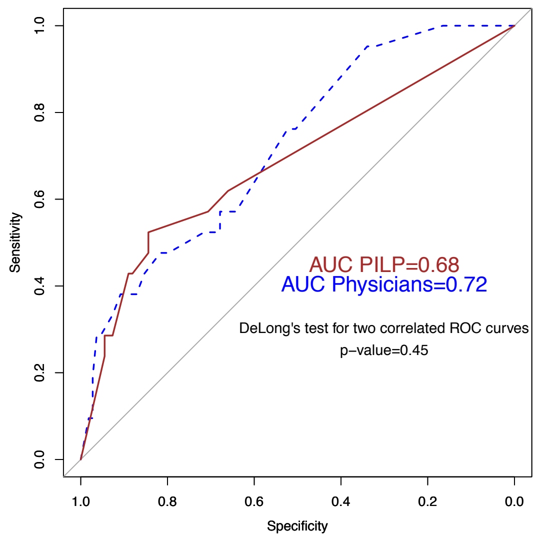
Figure 7.6 presents the ROC curves for the malignant class and four methods tested: PILP, SVM, LREG and NB. Each sub-figure shows the ROC of the physicians' predictions (blue dashed line) and the ROC of a method (brown solid line), both against the ground truth (confirmed malignancy or benignity of a 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 ROC curves presented in Fig. 7.6 were compared using DeLong's test for two correlated ROC curves and the difference between them 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. This experiment established that both PILP and other non-relational methods can successfully mimic the mental model of physicians in what concerns the probabilities of each case in this dataset.

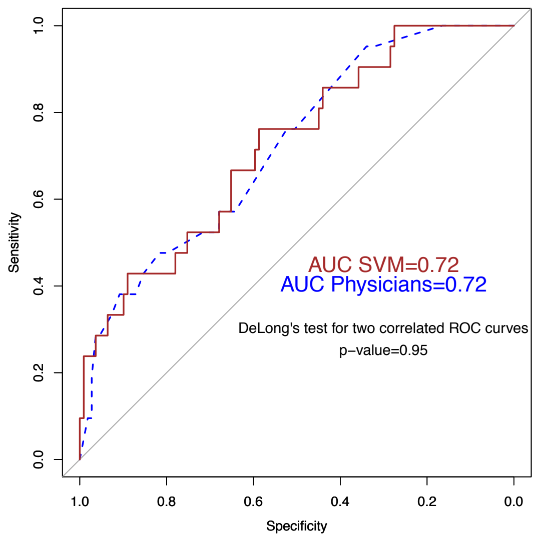
Next, the absolute error of the predictions was analysed. The absolute error is calculated by finding the absolute value of the difference between the prediction and the physicians' score, for a given case. It is relevant to consider the absolute error of predictions because these are the points where the classifiers' predictions disagree with the physicians' mental model, and more information about the performance of the classifier can be obtained from them. Figure 7.7 shows a plot of the classifiers prediction values (x-axis) against the physicians' prediction values (y-axis), for points where the absolute error was greater than 10%. Points in green (round markers) are cases where the tumour was found to be benign after excision, and conversely points in red (square markers) are cases where the tumour was found to be malignant.

Ideally, malignant prediction by both physician and the classifier should agree and appear on the top right of the plot. Conversely, benign predictions would appear on the bottom left. Points that are plotted below the diagonal line have higher classifier scores than physician scores, and conversely points which are plotted above the diagonal line have higher physician scores than classifier scores.

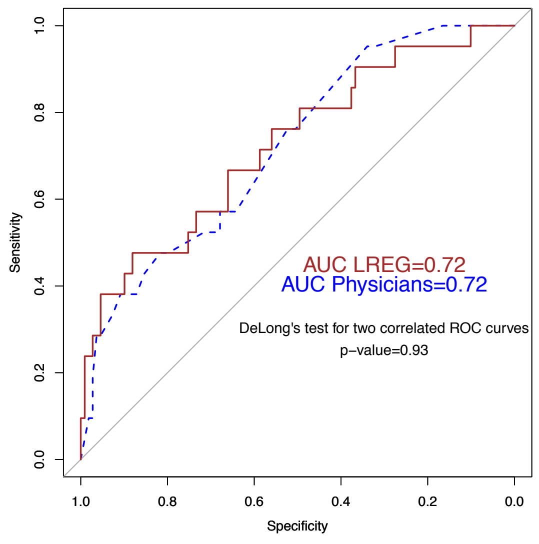
From the plots in Fig. 7.7, it is clear to see that the PILP classifier assigns higher malignancy values than physicians do to the confirmed malignancy cases (red points under the diagonal line). This is the case for 8 of the 9 malignant cases, and in the single case where this does not happen, PILP still predicts a reasonably high probability of malignancy (60%). Furthermore, for a malignancy threshold of 0.8, PILP still classifies five malignant cases correctly, whilst this only happens for one case using the physicians' scores. When PILP is compared to the other methods tested, it becomes clear that, in most cases, the other methods do not assign higher scores to malignant points than physicians do (few red points beneath the diagonal line), therefore not being of as much use to physicians as PILP, to aid in the diagnosis of malignant tumours. The ability to identify malignant cases is desirable in medical data since a false negative corresponds to assigning a benign label to a patient who



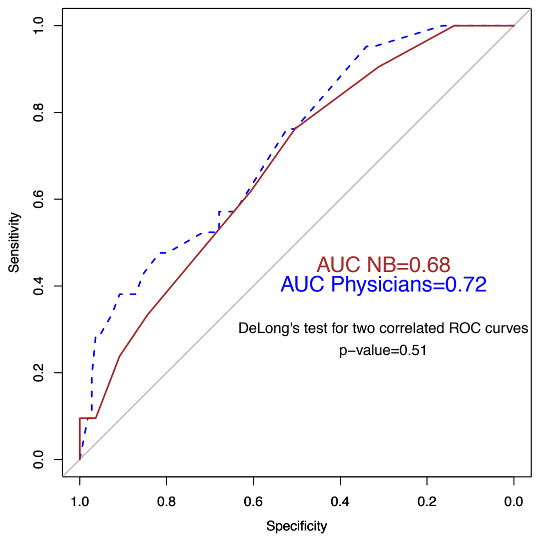
(a) PILP



(b) SVM

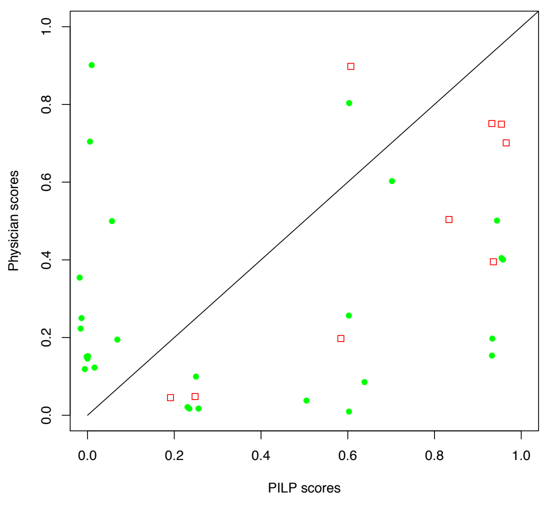


(c) LREG

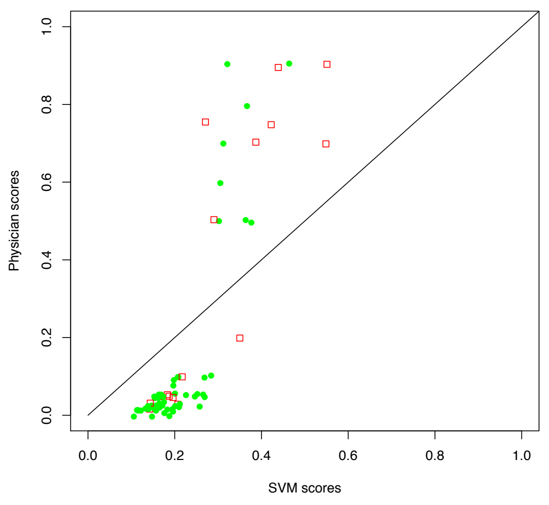


(d) NB

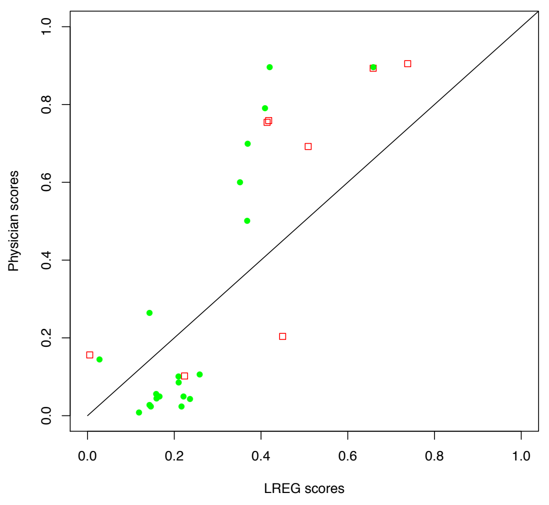
Figure 7.6: ROC curves, AUCs and p-values for PILP, SVM, LREG and NB methods



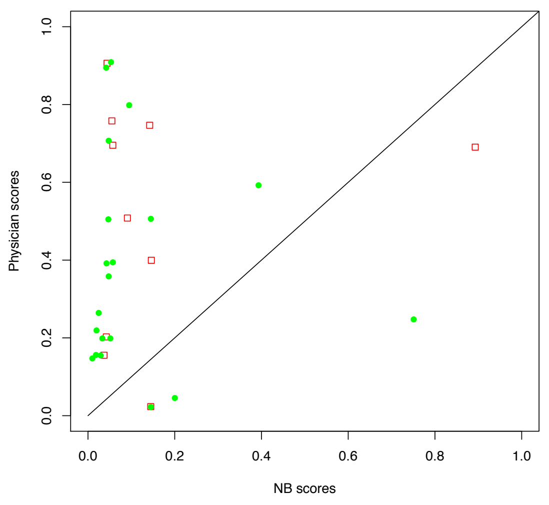
(a) PILP



(b) SVM



(c) LREG

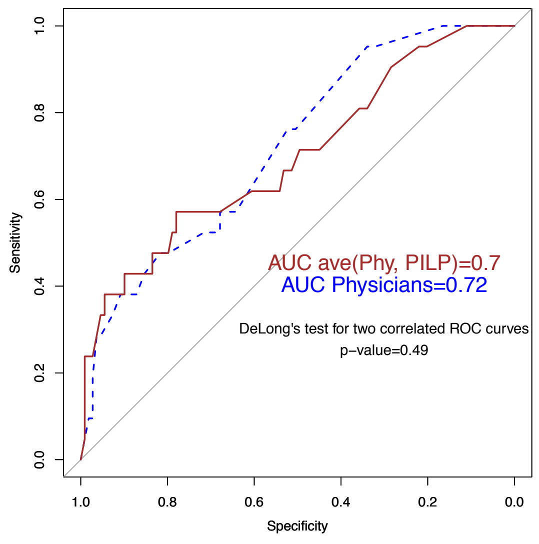


(d) NB

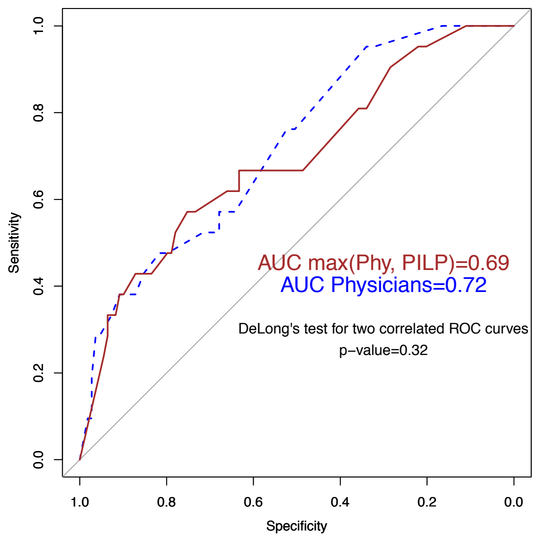
Figure 7.7: Plot of benign (green) and malignant (red) cases for the PILP, SVM, LREG and NB methods, for errors greater than 0.1, using a negligible amount of jittering

in fact has a malignant tumour.

Since the aim of decision support systems is to aid the process of medical diagnoses, two more models were built based on the results obtained previously. These two models are combined human and machine models, meaning that they take into account both the physicians' and the classifiers scores. The PILP classifier was selected as the machine model since it proved to be best at identifying malignant cases that the physicians had difficulty with. For this reason, two models were analysed: calculating the average of physician and the PILP scores, and calculating the maximum of the physician and the PILP scores. Figure 7.8 presents the ROCs, AUCs and p-values using DeLong's test for both these models.



(a) Average model

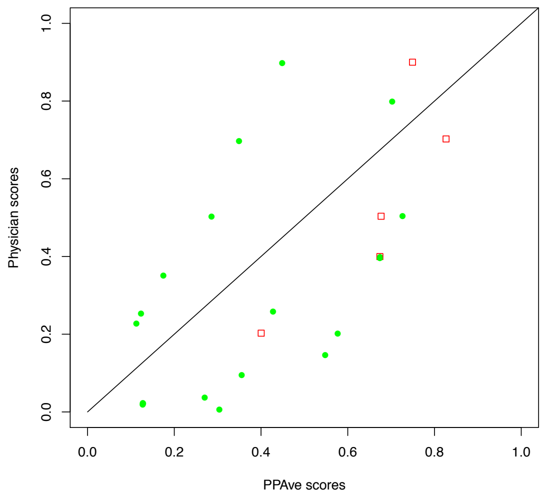


(b) Maximum model

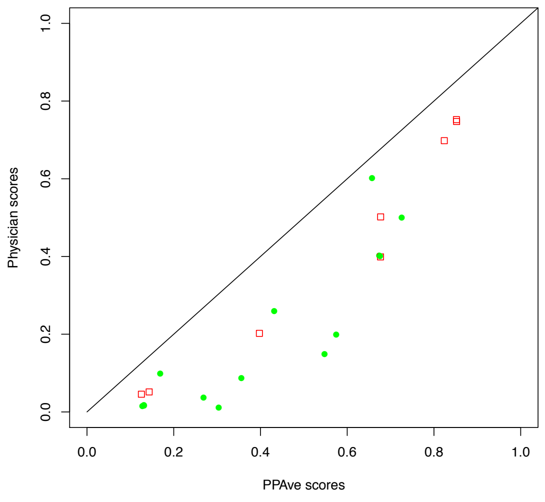
Figure 7.8: ROC curves, AUCs and p-values for the average of physician and PILP scores and for the maximum of physicians and PILP scores

The ROC curves plotted in Fig. 7.8 show no significant difference to the physicians predictive power, similarly to all other classifiers tested. Figure 7.9 performs the absolute error analysis, plotting the points where these models' predictions and physician's predictions differ by a value greater than 10%.

The scatter plots in Fig. 7.9 show that the maximum model can now predict higher scores for all malignant points (all red points below the diagonal line). This is to be expected since the model's scores are in effect the maximum score of the PILP and the Physician's model. However, both these models predict higher values for the benign cases as well, which is particularly evident in the case of the maximum



(a) Average model



(b) Maximum model

Figure 7.9: Plot of benign and malignant cases for the average and maximum of physician and PILP models, for errors greater than 0.1, using a negligible amount of jittering

model, where there are no points above the diagonal line. Whilst a high recall is a desirable feature in a medical decision support system, the ability to discriminate between malignant and benign cases is also important. The PILP model performs better in this area (Figure 7.7), since there is a vertical cluster of benign points which are clearly identified by the PILP model as being benign (score of 0.1 or less), and which are no longer present in the combined models analysed here.

Next, the full dataset was used to extract non-trivial knowledge regarding the physician's mental model that is being mimicked and the final theories found are reported in Fig. 7.10.

From the rules shown in Fig. 7.10, 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).
```

Figure 7.10: Theory extracted for physician's mental models.

7.3 Related Work

Relational learning in the form of ILP (without probabilities) has been successfully used in the field of breast cancer. Burnside et al. [16] uncovered rules that showed high breast mass density as an important adjunct predictor of malignancy in mammograms. Later, using a similar dataset, Woods et al. validated these findings [58] performing cross-validation. In another work, Davis et al. used SAYU, an ILP system that could evaluate rules according to their score in a Bayesian network, in order to classify new cases as benign or malignant. Results for a dataset of around 65,000 mammograms consisting of malignant and benign cases showed ROC areas slightly above 70% for Recall values greater than 50% [15]. Dutra et al. showed that the integration of physician's knowledge in the ILP learning process yielded better results than building models using only raw data [23]. The model we use in this paper was presented in more detail in [13] and [12]. One of the datasets used in those works is the same used in this paper, but only for comparing system's execution times. To the best of our knowledge, this is the first work that applies PILP to the area of breast cancer, and illustrates how a probabilistic knowledge representation can be linked with a logic representation to learn more expressive data models.

Chapter 8

Conclusion

Probabilistic Inductive Logic Programming is an important and hot research topic which has been attracting many communities and fostering events and journals special issues. We believe we contributed to the advance to the state-of-the-art in the field by allowing the efficient modelling and processing of probabilistic knowledge as well as extraction of relevant new (probabilistic) knowledge from the data without compromising (probabilistic) accuracy. This chapter concludes the work by summarising the main contributions it presents, as well as by discussing some directions of future work.

8.1 Main Contributions

The work described in this thesis consisted of the design, implementation and evaluation of a set of three pruning strategies for the PILP search space. To the best of the author's knowledge, the pruning strategies presented in this work are the first pruning strategies designed specifically for the PILP probabilistic logic search space, and which make use of the probabilistic information of candidate theories. Because the probabilistic information of candidate PILP theories can be used in different ways, three pruning criteria were studied, for each pruning strategy. Furthermore, the SkILL PILP system, also implemented as part of this work, allows for the exhaustive traversal of the PILP search space (when using no pruning), which in turn makes it possible to thoroughly assess the impact of pruning strategies in PILP candidate theories evaluated. Next, the main contributions of this work are described in more detail.

PILP Search Space Description An important contribution of this work in the systematic description of the PILP search space. The PILP search space can be divided in the AND search space (which explores rules) and the OR search space (which explores theories). In the AND search space, rules are combined using logic conjunction, and therefore they become more specific the greater the number of literals in their body. Specifying a rule has an impact in the probabilistic mass is covers (i.e. the proportion of worlds in which it is true), and this relation is what makes pruning the search space possible, using probabilistic information. This is a characteristic which is unique to PILP search spaces, since it does not occur in the deterministic version of the ILP algorithm. The behaviour of the OR search space is analogous to that of the AND search space, but in this case logical disjunction is used. This means that, unlike rules, theories which are composed of more rules are more general. The joint exploration of both the AND and the OR search spaces represents the full PILP search space.

Pruning Strategies Based on a thorough PILP search space understanding, three pruning strategies were developed in this work. Different pruning strategies target different characteristics of the PILP search space traversal, but their main purpose is to make the exploration of the PILP search space as efficient as possible. The pruning strategies can be used separately or in combination with each other, and they can each use different pruning criteria, for either search space. The proposed pruning strategies are:

- Fitness pruning: this pruning strategy aims at making the exponential explosion of the PILP search space polynomially bound. This is akin to using beam search, only the candidate members which transition to the following iteration are generated from two separate populations. These populations are termed the Primary and Secondary populations. The number of members of these populations are user specified (though a default is build in the SkILL system) and they are what determines the polynomial bound of explored candidates (the beam size). The members of these populations can be selected using different ranking metrics for each population and each search space this pruning strategy is applied to. The SkILL system supports the use of MAE and RMSE as ranking metrics, as well as an option for random candidates which aims at diversifying the search space. The effect of this pruning strategy is especially visible in the OR search space, since the AND search space is limited by the number of literals present in the PBK.

- Estimation pruning: unlike the fitness pruning strategy, estimation pruning prunes candidate theories one by one, and so there is no clear boundary on the number of explored theories during an execution. The aim of this pruning strategy is to avoid the computationally taxing exact probabilistic evaluation of candidate rules and theories. To achieve this, estimation pruning generates an estimate of the predictions of a candidate theory which is based on previously gathered probabilistic information. This work proposed five possible estimators for this pruning strategy: maximum, minimum, center, independence and mutual exclusion. Each of these estimators produces different estimations for candidate theories, and their behaviour also depends on which search space is being traversed. Once the estimation process is complete, the decision to prune away a candidate can be made based on one of the three pruning criteria. This pruning strategy can greatly reduce execution time since estimated bad candidates are never evaluated, and therefore they do not transition to the next iteration of the algorithm, in particular the independence estimator. However, this pruning strategy cannot guarantee that the optimal candidate theory is not pruned away due to an inaccurate estimation of its predictions.

- Prediction pruning: this pruning strategy differs from the two previous pruning strategies in that, under certain conditions, it is a safe pruning strategy, meaning that it can guarantee that the optimal candidate model is never pruned away. Unlike estimation pruning, prediction pruning prunes candidates after the exact probabilistic evaluation process takes place, and so this only impacts the candidates which transition to the next iteration of the algorithm. As such, the aim of prediction pruning is to allow only good candidates to transition to the next iteration. This is particularly important when there are limited resources available, for instance a maximum amount of running time, as is the case in most real world applications of PILP. The use of prediction pruning increases the quality of the explored fraction of the search space by effectively removing bad candidates from transitioning to the next iteration.

Pruning Criteria In estimation and prediction pruning, the decision to prune a candidate theory away can be made based on different pruning criteria. This work presents three pruning criteria with varying degrees of aggression: hard, soft and safe pruning criteria. All of the pruning criteria take into account the predictions of a candidate theory, and how they are positioned w.r.t. the example (target) values. In the case of estimation pruning, the estimates of a candidate theory are used in lieu of the actual predictions computed through exact probabilistic evaluation. The three pruning criteria are described next.

- Safe pruning criterion: this pruning criterion is the less aggressive pruning criterion presented in this work, and it only prunes away candidates which present a bad predictive performance for every example. When this pruning criterion is coupled with the prediction pruning strategy, the resulting pruning operation is guaranteed to never prune away the optimal model.
- Soft pruning criterion: this criterion is more aggressive than the safe pruning criterion but less so than the hard pruning criterion. The soft pruning criterion takes into account the overall positioning of the predictions w.r.t. the examples. If the candidate theory is performing badly on most examples, it will be pruned away; otherwise, it is kept. This is computed by taking into account the absolute error between the predictions and the examples, calculated for every example point.
- Hard pruning criterion: this is the most aggressive pruning criterion introduced in this work, and it pruned away candidate theories which perform worse in at least one example point (performance is again assessed based on average error). Even though this is an aggressive approach, this criterion can reduce execution time while maintaining predictive quality of the final model is most cases.

The SkILL System Unlike other systems in the literature, the SkILL system allows for exhaustive traversal of the search space (when using no pruning), conducted in separate stages so as to make the individual impact of pruning strategies more evident. SkILL runs on the Yap Prolog system [14], uses TopLog [41] as the basis rules generator and the ProbLog Yap library as its probabilistic inference engine. It is publicly available at *bitbucket.org/joanacortereal/skill/*. The SkILL system supports all the combinations of pruning strategies and criteria presented in this work. Furthermore, due to its modular code, user defined criteria can easily be added, as well as different ranking metrics and evaluation metrics.

Performance Evaluation This work presents a performance study for all pruning strategies and criteria, assessed both individually and combined with each other. The performance of the pruning strategies introduced in this work takes into account the execution time of the program, as well as the predictive accuracy of the final optimal model chosen, always calculated in a test set. This performance is also assessed using two PILP systems: the SkILL system, and another system from the literature (ProbFOIL+).

Real World Application this contribution consisted of building a PILP breast cancer benchmark using data from non-definitive biopsies and performing an exploratory study of these data. This study built a PILP model for the breast cancer data which predicts malignancy of the tumours and is statistically indistinguishable from the medical doctor's predictions. The cases where the model and the medical doctors' predictions disagree (i.e. present an error greater than 10%) were analysed in more detail. In these cases, the PILP model was shown to predict consistently higher values for malignancy of tumours, which is a desirable feature in medical decision support systems.

8.2 Future Work

The author hopes that the work resulting from this thesis can be a basis for further research in this area. Even though the goal of designing, implementing and testing pruning strategies for the increased efficiency of PILP engines was achieved, there are several future directions which can be explored. The pruning strategies introduced in this work weigh equally all misclassification costs, which may not be desirable in many cases. Furthermore, the parameters for the search must be set by the user (the SkILL system implements a default value), which is not ideal if the user is not familiarise with the behaviour of the pruning strategies and pruning criteria. Finally, the pruning strategies introduced here consider a specific flow for search space traversal: first traverse the AND search space and then the OR search space. Whilst this strategy better showcases the effects of pruning, it might be interesting to develop a search strategy which can, at every point in the algorithm, decide whether to specify or to generalise a theory, based on some of the criteria developed in this thesis (however, this was out of the scope of this work). These directions for future research are detailed below.

Different Misclassification Costs By using distance (or square distance) as a metric in the PILP evaluation of theories, it is not possible to differentiate whether a prediction is higher or lower than an example value. In real world applications (such as the medical field, for instance), a false positive (prediction greater than example value) may be more important than a false negative (example value greater than prediction value), and so the PILP model should reflect this preference. This could be done in two ways:

- Pruning criterion: develop a pruning criterion which weights either positive or negative distance more heavily than the opposite distance.
- Evaluation metric: chose the best model based on a metric which weighs positive and negative distance different.

Automatic Selection of Parameters this work introduced three different pruning strategies, each with a set of parameters which can be tuned. Fitness pruning parameters are two population sizes and two population ranking metrics, estimation pruning parameters are an estimator and a pruning criterion and prediction pruning only requires the choice of a pruning criterion, all of these for the AND and OR operations. A future research direction could be to develop a heuristic on how to select these parameters automatically and/or dynamically during the evaluation process, instead of having them set by the user. There are three different types of parameters that could be automatically selected:

- Pruning criterion: for both AND and OR operation, and for both estimation and prediction pruning.
- Estimator: if estimation pruning is used, for both operations.
- Population size: if fitness pruning is used, for both operations. Also, this size could adjust for each iteration, for instance allowing for more less complex (shorter) candidates, but less more complex (longer) candidates.

Search Space Traversal the pruning criterion presented here are aimed at pruning away parts of the PILP search space lattice so that it does not have to be explored. However, it might be possible to adapt them so that the search algorithm can be guided by them, i.e. develop a greedy search algorithm instead of pruning away parts of the search space. This could be done by using the specificity/generality of theories, possibly combined in sequential steps. A naive approach to this would be: (i) select a (good) theory; (ii) specify the theory according to some criterion until it is too specific; (iii) generalise the theory according to some criterion until it is to general (iv) repeat steps (ii) and (iii) until the final theory can no longer be improved by performing either operation.

References

[1] H. Aït-Kaci. *Warren's Abstract Machine – A Tutorial Reconstruction*. The MIT Press, 1991.

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

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

[4] Y. Bengio. Learning Deep Architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

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

[6] C. D'Orsi, L. Bassett, W. Berg, et al. *BI-RADS®: Mammography*. American College of Radiology, Inc., 4th edition, 2003. Reston, VA.

[7] M. Carlsson and P. Mildner. SICStus Prolog - the first 25 years. *Theory and Practice of Logic Programming*, 12(1-2):35–66, 2012.

[8] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What You Always Wanted to Know About Datalog (And Never Dared to Ask). *Knowledge and Data Engineering, IEEE Transactions on*, 1(1):146–166, 1989.

[9] J. Chen, S. Muggleton, and J. Santos. Learning Probabilistic Logic Models from Probabilistic Examples. *Machine Learning*, 73(1):55–85, Oct 2008.

[10] Alain Colmerauer and Philippe Roussel. The birth of Prolog. In *History of programming languages—II*, pages 331–367. ACM, 1996.

[11] D. Cook and L. Holder. *Mining Graph Data*. John Wiley & Sons, 2006.

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

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

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

[15] J. Davis, E. Burnside, I. Dutra, D. Page, R. Ramakrishnan, V. Santos Costa, and J. Shavlik. View Learning for Statistical Relational Learning: With an Application to Mammography. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 677–683. Professional Book Center, 2005.

[16] J. Davis, E. S. Burnside, I. Dutra, D. Page, and V. Santos Costa. Knowledge Discovery from Structured Mammography Reports Using Inductive Logic Programming. In *American Medical Informatics Association 2005 Annual Symposium*, pages 86–100, 2005.

[17] L. De Raedt. *Logical and Relational Learning*. Springer Science & Business Media, 2008.

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

[19] L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *International Conference on Algorithmic Learning Theory*, pages 19–36. Springer, 2004.

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

[21] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.

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

[23] I. Dutra, H. Nassif, D. Page, J. Shavlik, R. M. Strigel, Y. Wu, M. E. Elezaby, and E. Burnside. Integrating machine learning and physician knowledge to improve the accuracy of breast biopsy. In *AMIA Annual Symposium Proceedings*, pages 349–355, Washington, DC, 2011.

[24] S. Džeroski. Handling Imperfect Data in Inductive Logic Programming. In *SCAI*, pages 111–125, 1993.

[25] S. Džeroski. *Relational Data Mining*. Springer, 2010.

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

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

[28] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *IJCAI*, volume 99, pages 1300–1309, 1999.

[29] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT press, 2007.

[30] C. Green. Application of theorem proving to problem solving. Technical report, Defense Technical Information Center Document, Stanford University, 1969.

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

[32] Patricia Hill and John Wylie Lloyd. *The Gödel programming language*. The MIT Press, 1994.

[33] K. Kersting and L. De Raedt. Basic Principles of Learning Bayesian Logic Programs. In *Probabilistic Inductive Logic Programming - Theory and Applications*, pages 189–221, 2008.

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

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

[36] N. Lavrac and S. Dzeroski. *Relational Data Mining*. Springer, September 2001.

[37] J. W. Lloyd. Practical Advantages of Declarative Programming. In *Joint Conference on Declarative Programming, GULP-PRODE*, volume 1, pages 18–30, 1994.

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

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

[40] S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

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

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

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

[44] D. Poole. The Independent Choice Logic and Beyond. In *Probabilistic inductive logic programming*, pages 222–243. Springer, 2008.

[45] J. Quinlan. Learning Logical Definitions from Relations. *Machine learning*, 5(3):239–266, 1990.

[46] L. De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201, 1997.

[47] L. De Raedt and L. Dehaspe. Clausal Discovery. *Machine Learning*, 26:99–146, 1997.

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

[49] V. Santos Costa, D. Page, and J. Cussens. CLP(BN): Constraint Logic Programming for Probabilistic Knowledge. In *Probabilistic inductive logic programming*, pages 156–188. Springer, 2008.

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

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

[52] T. Sato. A Statistical Learning Method for Logic Programs with Distribution Semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP 95*. Citeseer, 1995.

[53] T. Sato and Y. Kameya. PRISM: A language for symbolic-statistical modeling. In *International Joint Conference on Artificial Intelligence*, volume 97, pages 1330–1339. Morgan Kaufmann, 1997.

[54] T. Sato and Y. Kameya. Parameter Learning of Logic Programs for Symbolic-statistical Modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[55] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic Programs with Annotated Disjunctions. In *Logic Programming*, pages 431–445. Springer, 2004.

[56] D. H. D. Warren. An Abstract Prolog Instruction Set. Technical Note 309, SRI International, 1983.

[57] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager. SWI-Prolog. *Journal of Theory and Practice of Logic Programming*, 12(1 & 2):67–96, 2012.

[58] R. Woods, L. Oliphant, K. Shinki, D. Page, J. Shavlik, and E. Burnside. Validation of Results from Knowledge Discovery: Mass Density as a Predictor of Breast Cancer. *J Digit Imaging*, pages 418–419, 2009.