A Revised Comparison of Bayesian Logic Programs and Stochastic Logic Programs *

Jianzhong Chen and Stephen Muggleton

Department of Computing, Imperial College London,
180 Queen's Gate, South Kensington, London SW7 2AZ, UK
{cjz, shm}@doc.ic.ac.uk

Abstract. Probabilistic logic learning integrates three underlying constituents – statistical learning and probabilistic reasoning with logical or relational representations. Probabilistic logic models and representations have varying levels of expressivity. This paper conducts on a behavioural comparison between Bayesian logic programs (BLPs) and stochastic logic programs (SLPs). We first explore their semantical differences and relations in probability definitions by revisting Halpern's first-order probabilistic logics and claim that SLPs define probabilities on atoms through proof trees and therefore are neither domain-frequency nor possible-world approaches. We then develop BLP-SLP translations by revising our previous work to solve a potential 'contradictory refutation' problem. We finally work on the development and comparison of learnabilities of the two frameworks based on computational learning theory and revised translations.

1 Introduction

Probabilistic logic learning (PLL) [1] combines expressive knowledge representation formalisms such as relational and first-order logic with principled probabilistic and statistical approaches to inference and learning. This combination is needed in order to face the challenge of real-world learning and data mining problems in which data are complex and heterogeneous and we are interested in finding useful predictive and/or descriptive patterns. PLL aims at integrating its three underlying constituents: statistical learning and probabilistic reasoning within logical or relational representations [2].

Probabilistic logic models (PLMs) and representations have varying levels of expressivity. As yet more effort has been put into defining new variants of PLMs than into characterising their relationships. Studying the relative expressive power of various probabilistic logic representations is a challenging and interesting task. Halpern's study of identifying two types of first-order probabilistic logics, ie. domain-frequency and possible-world [3], provides some fundamental grounds in the area. Besides, the idea of investigating the relations between

Bayesian logic programs (BLPs) and *stochastic logic programs* (SLPs) in particular has been put forward in [4, 5], which show the possibility of developing translations between PLMs with different semantics.

This paper extends the study in [5] by comparing BLPs vs. SLPs in semantics, translation and learnability. A *behavioural* approach is conducted which aims at developing a framework of comparing different PLMs at different levels. From a semantical perspective, we investigate if the same probabilistic knowledge (logic statements and queris associated with probability distributions) are encoded in different PLMs. More practically, we try to find inter-translations between these PLMs in which probabilistic knowledge encoded in one framework can be transferred into another framework. Furthermore, we work on the development and comparison of sample and time complexity of different PLL frameworks based on computational learning theory and translations.

For the semantics, we argue that BLPs are possible-world approaches, but SLPs are hard to be categorized into Halpern's category of first-order probabilistic logic as the probability defined in SLPs is taken over atoms or Herbrand base through proof trees. For the translation, we demonstrate a revised BLP-SLP translation approach to solve a potential 'contradictory refutation' problem existing in previous work. Our learnability study results in that BLPs and SLPs are polynomial-sample polynomial-time PAC-learnable from positive examples only. However, the ratio of sample complexity of a BLP and its translation SLP shows that SLPs need larger sample space compared to BLPs.

2 BLPs vs. SLPs: Representations

This section briefly reviews the syntax of BLPs and SLPs, and then discusses their semantics of probability definitions in terms of Halpern's category of first-order probabilistic logics.

2.1 BLPs vs. SLPs: Syntax

Bayesian logic programs (BLPs) were first introduced in [4], as a generalization of Bayesian nets (BNs) and first-order logic programs. The basic idea is to view logical (ground) atoms as sets of random variables which are similar to each other. Clauses specify general dependencies among random variables which are quantified by associating conditional probability distributions (CPDs) with clauses. As an example, BNs correspond to propositional clauses, with one clause per node and no combining rules.

In general, a BLP has two components – a logical one, which is a set of *Bayesian clauses* and a quantitative one, which is a set of CPDs and *combining rules* corresponding to the logical structure. A Bayesian (definite) clause c is an expression of the form $\mathtt{A} \mid \mathtt{A}_1, \ldots, \mathtt{A}_n$ where $n \geq 0$, the $\mathtt{A}, \mathtt{A}_1, \ldots, \mathtt{A}_n$ are Bayesian atoms and all Bayesian atoms are (implicitly) universally quantified. It further defines $head(c) = \mathtt{A}$ and $body(c) = \{\mathtt{A}_1, \ldots, \mathtt{A}_n\}$. A Bayesian clause is different from a logical definite clause because the sign '|' is employed instead of ': −' in

order to highlight Bayesian atoms are associated with a set of possible states, not just binary values. As there can be many clauses with same head or non-ground heads that can be unified, a combining rule is a function that maps finite sets of CPDs $\{\mathbf{Pr}(\mathtt{A} \mid \mathtt{A_{i1}}, \ldots, \mathtt{A_{in_i}}) \mid i = 1, \ldots, m\}$ onto one (*combined*) CPD $\mathbf{Pr}(\mathtt{A} \mid \mathtt{B_1}, \ldots, \mathtt{B_k})$ with $\{\mathtt{B_1}, \ldots, \mathtt{B_k}\} \subseteq \bigcup_{i=1}^{m} \{\mathtt{A_{i1}}, \ldots, \mathtt{A_{in_i}}\}$. Common combining rules include the 'noisy-or' rule (for boolean domains) and the 'max' rule (for finite domains). Furthermore, aggregate functions such as 'median' and 'mode' can also be implemented.

Stochastic logic programs (SLPs) [6] were introduced originally as a way of lifting stochastic grammars to the level of first-order logic programs. SLPs were considered as a generalization of hidden Markov models and stochastic context-free grammars. SLPs have later been used to define distributions for sampling within inductive logic programming (ILP). It is clear that SLPs provide a way of probabilistic logic representations and make ILP become better at inducing models that represent uncertainty.

A *pure* SLP, S, consists of a set of labelled clauses $l : C$, where l is a probability label, i.e. a number in the interval [0,1], and C is a first-order range-restricted definite clause [1]. An *impure* SLP is a definite logic program where not all clauses have probability labels. The subset S_p of clauses in S with predicate symbol p in the head is called the definition of p. For each definition S_p the sum of probability labels π_p must be at most 1. S is said to be *nomalised* (or *complete*) if $\pi_p = 1$ for each p and *unnormalised* (or *incomplete*) otherwise. In a pure normalised SLP, each choice for a clause C has a parameter attached and the parameters sum to one, so they can therefore be interpreted as probabilities. Pure normalised SLPs are defined such that each parameter l denotes the probability that C is the next clause used in a derivation given that C^+ has the correct predicate symbol.

2.2 BLPs vs. SLPs: Semantics

BLPs have a close link to BNs, ie., each ground Bayesian atom can be associated to a chance node (a standard random variable), whose set of states is the domain of the Bayesian atom. The links (influence relations) between chance nodes are given by the Bayesian clauses, and the link matrices by the CPDs associated to these Bayesian clauses. The set of ground Bayesian atoms in the least Herbrand model together with the structure defined by the set of ground instances of the Bayesian clauses define a global (possibly infinite) dependency graph. [4] shows that any *well-defined* BLP [2] B defines a unique probability distribution over the possible valuations of a ground query $G \in LH(B)$. Given a G and some evidence, the induced BNs can be constructed using knowledge-based model construction (KBMC), which can be queried using any BN inference algorithm.

[1] a definite logical clause C is range-restricted if every variable in C^+, the head of C, is found in C^-, the body of C.

[2] A range-restricted BLP B is well-defined if its least Herbrand model $LH(B)$ is not empty, its induced dependency graph is acyclic and each random variable is only influenced by finite set of random variables.

An SLP S has a distributional semantics, that is one which assigns a probability distribution to the atoms of each predicate in the Herbrand base of the clauses in S. The probabilities are assigned to atoms in terms of their proofs according to a stochastic SLD-resolution strategy which employs a stochastic selection rule based on the values of the probability labels. [7] introduces the normal semantics for SLPs by defining distributional Herbrand interpretation and distributional Herbrand model. In [8], an SLP S with parameters $\lambda = \log l$ together with a goal G defines up to three related distributions in the SLD-tree of G: $\psi_{\lambda,S,G}(x)$, $f_{\lambda,S,G}(r)$ and $p_{\lambda,S,G}(y)$, defined over derivations $\{x\}$, refutations $\{r\}$ and atoms $\{y\}$, respectively. It is important to understand that SLPs do not define distributions over possible worlds, i.e., in particular, $p_{\lambda,S,G}(y)$ defines a distribution over atoms, not over the truth values of atoms.

In the framework of Halpern [3], two types of first-order probabilistic logic are categorized, ie. probabilities on the domain (or *type 1* probability structure) and probabilities on possible worlds (or *type 2* probability structure). Type 1 probability structure can represent statements like "The probability that a randomly chosen bird will fly is greater than .9". It provides a type of domain-frequency approaches, which semantically illustrates objective and 'sampling' probabilities of domains. More precisely, a type 1 probability structure is a tuple (D, π, μ), where D is a domain, π maps predicate and function symbols in alphabet to predicates and functions of the right arity over D, and μ is a discrete probability function on D. The probability here is taken over the domain D. On the other hand, type 2 probability structure may represent statements like "The probability that Tweety (a particular bird) flies is greater than .9". It is a kind of possible-world approaches and illustrates the subjective and 'degree-of-belief' semantics of the probabilities of domains. Formally, a type 2 probability structure is a tuple (D, W, π, μ), where D is a domain, W is a set of states or possible worlds, for each state $w \in W$, $\pi(w)$ maps predicate and function symbols in alphabet to predicates and functions of the right arity over D, and μ is a discrete probability function on W. The probability here is taken over W, the set of states (or possible worlds or logic models). BNs and related models are type 2 approaches.

In terms of the above categorization, it is straightforward to map BLPs into type 2 approaches, as BLPs are first-order Bayesian networks and define probabilities on possible worlds. In contrast, it is hard to categorize SLPs to either type 1 or type 2 approaches, as probabilities defined in SLPs are taken over atoms in Herbrand base in terms of their proofs (or SLD-trees). However, SLPs have a "close" semantics of probability definition with type 1 approaches, ie., they all take into account 'sampling' probabilities of domains, which are objective and computed from only one world, rather than 'degree-of-belief' probabilities that are subjective and estimated from multiple possible worlds. Compared to standard type 1 probability structures, SLPs define probabilities on atoms through SLD-trees rather than on domains. Formally, a *proof-tree* probability structure is defined in SLPs, which is denoted as a tuple $(D, HB, < G, ST_G >, \pi, \mu)$, where D is a domain, HB is the Herbrand base of clauses over D, G is a definite goal

(or query), ST_G is the SLD-tree for G, π maps predicate and function symbols in alphabet to predicates and functions of the right arity over D, and μ is a discrete probability function on HB in terms of $< G, ST_G >$. Given a goal and corresponding (stochastic) SLD-tree, a probability distribution is thus defined over a sample space of atoms taken from HB.

3 BLPs vs. SLPs: Translation

We have talked about what respective kinds of knowledge can be captured with probabilities in BLPs and SLPs. We now provide methods to find possible links between their semantics. A *behavioral translation* approach is conducted whose aim is to provide ways in which knowledge encoded in one framework can be transferred into another framework in terms of probability distributions. We first define that the semantics are equivalent when equivalent (set of) queries on equivalent programs infer the same (set of) probability distributions. Our goal is eventually to provide inter-translation algorithms that transform a program in one framework (eg. BLPs) into an equivalent program in another framework (eg. SLPs) and to analyze their learnabilities.

A general translation between 'sampling' and possible-world probabilities can be briefly described as follows. Our goal is to map a formulae φ in a probability structure M to a formulae φ' in another probability structure M' such that $M \vDash \varphi$ iff $M' \vDash \varphi'$. Given $M = (D, \pi, \mu)$ over a domain D or $M = (D, HB, < G, ST_G >, \pi, \mu)$ over a Herbrand base HB, we construct a corresponding possible world probability structure $M' = (D, W, \pi', \mu')$ or $M' = (D, HB, \pi', \mu')$ over the same domain or Herbrand base, such that for each $d \in D$ or $d \in HB$, there is a nonempty set of states $W_d = \{w' : \pi(w)(a) = d\}$, where a is a fresh constant symbol, such that $\mu'(W_d) = \mu(d)$. For the translation in the other direction, we replace a predicate $P(x_1, \ldots, x_n)$ in φ' by a predicate $P(x_1, \ldots, x_n, w)$ in φ, where w ranges over states. Loosely speaking, the translation from possible-world structures to 'sampling' structures will 'flatten' multiple worlds into one world, and in contrast the translation in the other direction will 'compress' one world to make multiple worlds.

Details of translation between BLPs and SLPs have been provided in [5], including translations from restricted BLPs to SLPs, from standard BLPs to extented SLPs, and from SLPs to BLPs. However, there is a potential 'contradictory refutation' problem in BLP-SLP translation, which is illustrated in Fig. 2(a) for a BLP example shown in Fig.1. The error lies in the potential inconsistent value settings (or substitutions) between atoms in a clause, eg., in clause **d(tom,y)** ← **b(tom,y),c(tom,y)**, **b(tom,y)** may be set to **a(tom,y)** while **c(tom,y)** might be set to a contradictory value **a(tom,n)** simultaneously. To solve the problem, we introduce an extra data structure of list to 'set and remember' values instead of just setting values. Translations from the BLP to an SLP by applying previous method (in [5]) and a revised method (this paper) are shown in Fig. 3(a) and (b) respectively, and the resolved stochastic SLD-tree can be seen in Fig.2(b). More precisely, a revised BLP-SLP translation algorithm is

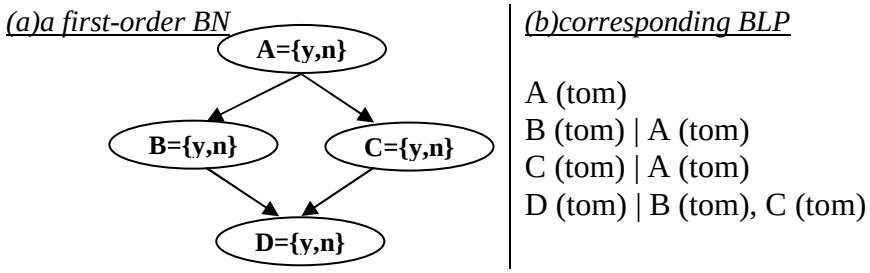

Fig. 1. An example of a first-order BN and corresponding BLP representation

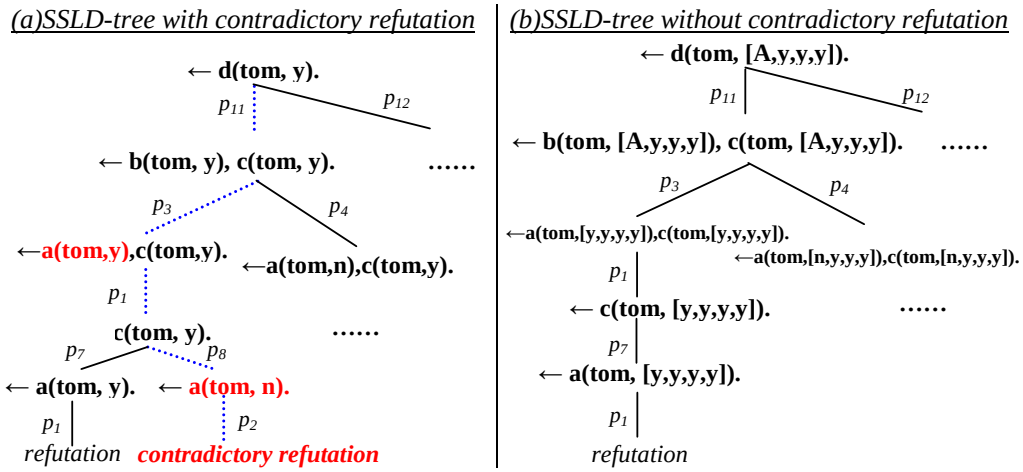

Fig. 2. (a)Stochastic SLD-tree with contradictory refutation (shown in dash lines) and (b)Resolved SSLD-tree without contradictory refutation

(a)translating to SLP S_1	*(b)translating to SLP with revision, S_2*
p_1: a(tom, y) ← .	p_1: a(tom, [y,B,C,D]) ← .
p_2: a(tom, n) ← .	p_2: a(tom, [n,B,C,D]) ← .
p_3: b(tom, y) ← a(tom, y).	p_3: b(tom, [y,y,C,D]) ← a(tom, [y,y,C,D]).
p_4: b(tom, y) ← a(tom, n).	p_4: b(tom, [n,y,C,D]) ← a(tom, [n,y,C,D]).
p_5: b(tom, n) ← a(tom, y).	p_5: b(tom, [y,n,C,D]) ← a(tom, [y,n,C,D]).
p_6: b(tom, n) ← a(tom, n).	p_6: b(tom, [n,n,C,D]) ← a(tom, [n,n,C,D]).
p_7: c(tom, y) ← a(tom, y).	p_7: c(tom, [y,B,y,D]) ← a(tom, [y,B,y,D]).
p_8: c(tom, y) ← a(tom, n).	p_8: c(tom, [n,B,y,D]) ← a(tom, [n,B,y,D]).
p_9: c(tom, n) ← a(tom, y).	p_9: c(tom, [y,B,n,D]) ← a(tom, [y,B,n,D]).
p_{10}: c(tom, n) ← a(tom, n).	p_{10}: c(tom, [n,B,n,D]) ← a(tom, [n,B,n,D]).
p_{11}: d(tom, y) ← b(tom, y), c(tom, y).	p_{11}: d(tom, [A,y,y,y]) ← b(tom, [A,y,y,y]), c(tom, [A,y,y,y]).
p_{12}: d(tom, y) ← b(tom, y), c(tom, n).	p_{12}: d(tom, [A,y,n,y]) ← b(tom, [A,y,n,y]), c(tom, [A,y,n,y]).
p_{13}: d(tom, y) ← b(tom, n), c(tom, y).	p_{13}: d(tom, [A,n,y,y]) ← b(tom, [A,n,y,y]), c(tom, [A,n,y,y]).
p_{14}: d(tom, y) ← b(tom, n), c(tom, n).	p_{14}: d(tom, [A,n,n,y]) ← b(tom, [A,n,n,y]), c(tom, [A,n,n,y]).
p_{15}: d(tom, n) ← b(tom, y), c(tom, y).	p_{15}: d(tom, [A,y,y,n]) ← b(tom, [A,y,y,n]), c(tom, [A,y,y,n]).
p_{16}: d(tom, n) ← b(tom, y), c(tom, n).	p_{16}: d(tom, [A,y,n,n]) ← b(tom, [A,y,n,n]), c(tom, [A,y,n,n]).
p_{17}: d(tom, n) ← b(tom, n), c(tom, y).	p_{17}: d(tom, [A,n,y,n]) ← b(tom, [A,n,y,n]), c(tom, [A,n,y,n]).
p_{18}: d(tom, n) ← b(tom, n), c(tom, n).	p_{18}: d(tom, [A,n,n,n]) ← b(tom, [A,n,n,n]), c(tom, [A,n,n,n]).

Fig. 3. Previous and revised translations from the above BLP to an SLP

shown in the following steps. Let B denote a restricted BLP [3] and S denote its translation SLP.

- Identify each k-ary Bayesian atom b, which appears in B and has the value domain V_b, to the $(k+1)$-ary (logical) atom $b(v_b)$ having the same k first arguments and a value $v_b \in V_b$ as last argument.
- Construct a list lv_b to replace v_b. The length of lv_b is the number of all Bayesian atoms. Each element of lv_b corresponds to an arbitrary Bayesian atom b' and is set to a fresh variable if $b' \neq b$ or a value $v_{b'} \in V_{b'}$ if $b' = b$.
- For each Bayesian clause $head \mid b_1, ..., b_n$ in B, for each value in the associated CPD, which indicates the probability $p_{v_h, v_{b1}, ..., v_{bn}}$ that the Bayesian atom $head$ takes the value v_h given that the $\{b_i : i \in \mathcal{N}_n\}$ take the values $(v_{b1}, ..., v_{bn})$, construct a list lv_h for $head$ as done in step 2, then construct the stochastic clause consisting of the parameter $p_{v_h, v_{b1}, ..., v_{bn}}$, and the definite clause: $head(lv_h) \leftarrow b_1(lv_{b1}), ..., b_n(lv_{bn})$.
- For $lv_h, lv_{b1}, ..., lv_{bn}$, update the value for each element in lists with respect to $v_h, v_{b1}, ..., v_{bn}$ respectively.
- The standard translation of B consists of the n stochastic clauses constructible in that way, n being the sum of the numbers of coefficients in the CPD tables. This SLP is pure and unnormalised (the parameters of the clauses in $S_h \subseteq S$ sum to the product of the domain sizes of the Bayesian atoms in the body of the Bayesian clause with head h).

Note that, in a definite clause (eg. **b(tom,[n,y,C,D])** $\leftarrow$ **a(tom,[n,y,C,D])**), all atoms have the same list values (eg. **[n,y,C,D]**), in which the elements corresponding to the atoms occurred in the clause are value set (eg. **[n,y, ,]** corresponds to atoms **a,b**) and other elements are assigned to be variables (eg. **[, ,C,D]** correspond to atoms **c,d**). The introduction of lists with variables will guarentee atoms to propagate consistent values to their predecessors in stochastic SLD-trees.

4 BLPs vs. SLPs: Learnability

We now discuss on the learnability of SLPs and BLPs based on the above revised translation. The purpose is to investigate the computational properties and sample complexities, therefore, compare their learnability and measure the trade-offs of the two language fragments. This is of particular interest if we hope to predict the number of examples required to reach a particular level of accuracy with a particular representation and learning algorithm. Our study is based on computational learning theory [9] and the probably approximately correct (PAC) learning paradigm [10, 11]. Loosely speaking, given some class C of possible target concepts and a learner L using hypothesis space H, we say that the concept

[3] A restricted BLP is one whose predicate definitions contain a single stochastic clause each. For the sake of simplicity, we discuss BLP without combining rules here, however, our revised algorithm can be applied to translating BLPs with combining rules as well.

class C is PAC-learnable by L using H if, for any target concept c in C, L will with probability $(1-\delta)$ output a hypothesis h with $error(h) < \varepsilon$, after observing a reasonable number of training examples and performing a reasonable amount of computation.

Our first finding is based on the result "*first-order jk-clausal theories are PAC-learnable*" presented in [12]. In the nonmonotonic (or descriptive or learning from interpretations) ILP setting, where hypotheses are learned from positive examples only under closed world assumption, *BLPs and SLPs are polynomial-sample polynomial-time PAC-learnable with one-sided error from positive examples only.* This is obvious as BLPs and SLPs are in the class of jk-clausal theories, which is the class of all theories composed of range-restricted clauses that contain at most k literals per clause with each literal (atom) is of size at most j, and can be learned from positive examples only.

Furthermore, we can estimate the ratio of corresponding sample-complexities of BLP and its translation SLP by computing their Vapnik-Chervonenkis (VC) dimensions, which are known to be useful in quantifying the bias inherent in a restricted hypothesis space [10, 11]. We first define two hypothesis space as

- $\mathcal{H}_B = \{H_B | H_B$ is a BLP hypothesis of jk-clausal theories that is learned by a BLP learner from positive examples.$\}$
- $\mathcal{H}_S = \{H_S | H_S$ is a corresponding SLP hypothesis of jk-clausal theories that is translated from BLP.$\}$

We then assume the VC-dimension of a clausal hypothesis space consists of a parameter (or probability) space and a structure (or clauses) space, ie.

$$VC - dim(\mathcal{H}_B) \leq \log_2 |\mathcal{H}_B| \leq \log_2 |\mathcal{H}_B^P| + \log_2 |\mathcal{H}_B^C| \leq \alpha_B \times c_B + \beta_B \times c_B,$$

$$VC - dim(\mathcal{H}_S) \leq \log_2 |\mathcal{H}_S| \leq \log_2 |\mathcal{H}_S^P| + \log_2 |\mathcal{H}_S^C| \leq \alpha_S \times c_S + \beta_S \times c_S,$$

where c_B and c_S are the number of clauses in $\mathcal{H}_B$ and $\mathcal{H}_S$ respectively with each clause has α_B or α_S bits for encoding parameters [4] and β_B or β_S bits for encoding clause structures. As shown in [12], the corresponding sample complexities can be computed by applying Blumer bound [13], eg.

$$m_B \geq \max(\frac{4}{\varepsilon} \log_2 \frac{2}{\delta}, \frac{8 \times VC - dim(\mathcal{H}_B)}{\varepsilon} \log_2(\frac{13}{\varepsilon})),$$

$$m_S \geq \max(\frac{4}{\varepsilon} \log_2 \frac{2}{\delta}, \frac{8 \times VC - dim(\mathcal{H}_S)}{\varepsilon} \log_2(\frac{13}{\varepsilon})),$$

where $\varepsilon \in (0, 1]$ is the error parameter and $\delta \in (0, 1]$ is the confidence parameter of the learner. Now we may compute the ratio of m_S against m_B as follows,

$$\frac{m_S}{m_B} \geq \frac{VC - dim(\mathcal{H}_S)}{VC - dim(\mathcal{H}_B)} \geq \frac{\alpha_S \times c_S + \beta_S \times c_S}{\alpha_B \times c_B + \beta_B \times c_B},$$

[4] In SLPs, α_S is the encoding length of a real number; while in BLPs, α_B corresponds to an average encoding length of a CPD table, in which a number of probability values are encoded.

Knowing that

$$\alpha_S \times c_S = \alpha_B \times c_B,$$

$$\beta_S \times c_S \geq (\beta_B + c_B^2) r^{c_B},$$

where we assume each predicate has at most r states (or values). Then we have, for large c_B ($c_B \longrightarrow \infty$),

$$\frac{m_S}{m_B} \geq \frac{\alpha_B \times c_B + (\beta_B + c_B^2) r^{c_B}}{\alpha_B \times c_B + \beta_B \times c_B} \geq \frac{1}{2} + \frac{c_B \times r^{c_B}}{2} \geq \frac{c_B \times r^{c_B}}{2}.$$

As an instance, the result for the translation example shown in Figures 1, 2 and 3 is $\frac{m_S}{m_B} \geq c_B \times 2^{c_B - 1} = 32$. The ratio shows that the translated SLPs need much larger sample space (or more training examples) compared to BLPs as a result of our revised translation algorithm, in which extra data structures are introduced and every possible world needs to be encoded.

5 Conclusions

It is a challenging and interesting task to study the relative expressive power of various probabilistic logic representations. The behavioural approach developed in this paper provide a framework for fulfilling the task. We have demonstrated differences and relations between SLPs' and BLPs' semantics in terms of their definitions of probability distributions. This will help to understand different types of first-order probabilistic logics. We have also shown the possibility of developing inter-translations between the two frameworks so that they can encode the same knowledge. It is important to solve the existing 'contradictory refutation' problem, as it is helpful to further study of the expressivities of domain-frequency (type 1), possible-world (type 2) and proof-tree (eg. SLPs) probability approaches. Moreover, we have provided an initial result of analyzing the learnabilities of different PLMs based on the translations and computational learning theory. To our knowledge, this is the first work in the area. It is useful because the computational properties and complexities of PLMs can be measured and compared from machine learning point of view. All the findings in this paper will help us further study and develop an integrated theory of probabilistic logic learning.

Acknowledgements

The authors would like to acknowledge the support of the EC Sixth Framework (Priority 2) IST Specific Targetd Research Project "Application of Probabilistic Inductive Logic Programming II (APrIL II)" (Grant Ref: FP-508861).

References

1. De Raedt, L., Kersting, K.: Probabilistic Logic Learning. ACM-SIGKDD Explorations: Special Issue on Multi-Relational Data Mining. 5(1) (2003) 31–48
2. De Raedt, L., Dietterich, T., Getoor, L., Muggleton, S.: 05051 Executive Summary – Probabilistic, Logical and Relational Learning - Towards a Synthesis. In: Dagstuhl Seminar Proceedings 05051. Dagstuhl, Germany. (2006)
3. Halpern J.Y.: An Analysis of First-Order Logics of Probability. Artificial Intelligence. 46 (1989) 311–350
4. Kersting, K., De Raedt, L.: Bayesian Logic Programs. In: Cussens, J., Frisch, A. (eds.): Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming. (2000) 138–155
5. Puech, A., Muggleton, S.: A Comparison of Stochastic Logic Programs and Bayesian Logic Programs. IJCAI03 Workshop on Learning Statistical Models from Relational Data. (2003)
6. Muggleton, S.: Stochastic logic programs. In: De Raedt, L. (eds.): Advances in Inductive Logic Programming. (1996) 254–264
7. Muggleton, S.: Learning Stochastic Logic Programs. Electronic Transactions in Artificial Intelligence. 5(041) (2000)
8. Cussens, J.: Parameter Estimation in Stochastic Logic Programs. Machine Learning. 44(3) (2001) 245–271
9. Valiant, L.G.: A Theory of the Learnable. Communications of the ACM. 27(11) (1984) 1134–1142
10. Haussler, D.: Probably Approximately Correct Learning. In: National Conference on Artificial Intelligence. (1990) 1101–1108
11. Mitchell, T.M.: Machine Learning. The McGraw-Hill Companies, Inc. (1997)
12. De Raedt, L., Džeroski, S.: First-Order jk-clausal Theories are PAC-learnable. Aritificial Intelligence. 70 (1994) 375–392
13. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.: Learnability and the Vapnic-Chervonenkis Dimension. Journal of the ACM. 36(4) (1989) 929–965