# Learning structure and parameters of Stochastic Logic Programs

S. Muggleton

Department of Computing,
Imperial College,
London,
United Kingdom.

**Abstract.** Previous papers have studied learning of Stochastic Logic Programs (SLPs) either as a purely parametric estimation problem or separated structure learning and parameter estimation into separate phases. In this paper we consider ways in which both the structure and the parameters of an SLP can be learned simultaneously. The paper assumes an ILP algorithm, such as Progol or FOIL, in which clauses are constructed independently. We derive analytical and numerical methods for efficient computation of the optimal probability parameters for a single clause choice within such a search.

Keywords: Stochastic logic programs, generalisation, analytical methods, numerical methods.

## 1 Introduction

Stochastic Logic Programs (SLPs) [11] were introduced originally as a way of lifting stochastic grammars to the level of first-order Logic Programs (LPs). Later Cussens [1] showed that SLPs can be used to represent undirected Bayes' nets. SLPs have been used [9] to define distributions for sampling within Inductive Logic Programming (ILP) [7].

Previous papers have studied learning of Stochastic Logic Programs (SLPs) either as a purely parametric estimation problem [2] or separated structure learning and parameter estimation into separate phases [12]. In this paper we consider ways in which both the structure and the parameters of an SLP can be learned simultaneously. We assume an ILP algorithm, such as Progol [6] or FOIL [14], in which clauses are constructed independently. Analytical and numerical methods are derived for efficient computation of the optimal probability label for a single clause choice within such a search.

The paper is arranged as follows. Section 2 gives a general introduction to SLPs. The generalisation model for SLPs described in [12] is reviewed in Section 3. Section 4 describes the problem of choosing the probability label for a single new clause, assuming all other clauses and probability labels in the SLP remain fixed. Equations for the general case are derived using calculus in Section 4.1.

A closed form solution for the case in which exactly two examples are present is described in Section 4.2. Iterative numerical methods for solving the general case are described in Section 4.3. Related work is then discussed in Section 5. Section 6 describes provides conclusions and a description of further work.

## 2 Stochastic logic programs

### 2.1 Syntax of SLPs

An SLP $S$ is a set of labelled clauses $p{:}C$ where $p$ is a probability (ie. a number in the range $[0,1]$) and $C$ is a first-order range-restricted definite clause[1]. 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 $\pi_p$ must be at most 1. $S$ is said to be complete if $\pi_p = 1$ for each $p$ and incomplete otherwise. $P(S)$ represents the definite program consisting of all the clauses in $S$, with labels removed.

*Example 1.* **Unbiased coin.** The following SLP is complete and represents a coin which comes up either heads or tails with probability 0.5.

$$S_1 = \left\{ \begin{array}{l} 0.5 : coin(head) \leftarrow \\ 0.5 : coin(tail) \leftarrow \end{array} \right\}$$

$S_1$ is a simple example of a sampling distribution

*Example 2.* **Pet example.** The following SLP is incomplete.

$$S_2 = \left\{ \begin{array}{r} 0.3 : likes(X,Y) \leftarrow pet(Y,X), pet(Z,X), \\ cat(Y), mouse(Z) \end{array} \right\}$$

$S_2$ shows how statements of the form $\Pr(P(\boldsymbol{x})|Q(\boldsymbol{y})) = p$ can be encoded within an SLP, in this case $\Pr(\text{likes(X,Y)}|\ldots)) = 0.3$.

### 2.2 Proof for SLPs

A Stochastic SLD (SSLD) refutation is a sequence $D_{S,G} = \langle 1{:}G, p_1{:}C_1, .., p_n{:}C_n\rangle$ in which $G$ is a goal, each $p_i{:}C_i \in S$ and $D_{P(S),G} = \langle G, C_1, .., C_n\rangle$ is an SLD refutation from $P(S)$. SSLD refutation represents the repeated application of the SSLD inference rule. This takes a goal $p{:}G$ and a labelled clause $q{:}C$ and produces the labelled goal $pq{:}R$, where $R$ is the SLD resolvent of $G$ and $C$. The answer probability of $D_{S,G}$ is $Q(D_{S,G}) = \prod_{i=1}^{n} p_i$. The incomplete probability of any ground atom $a$ with respect to $S$ is $Q(a|S) = \sum_{D_{S,(\leftarrow a)}} Q(D_{S,(\leftarrow a)})$. We can state this as $S \vdash_{SSLD} Q(a|S) \leq Pr(a|S) \leq 1$, where $Pr(a|S)$ represents the conditional probability of $a$ given $S$.

[1] Cussens [1] considers a less restricted definition of SLPs.

*Remark 1.* **Incomplete probabilities.** If $a$ is a ground atom with predicate symbol $p$ and the definition $S_p$ in SLP $S$ is incomplete then $Q(a|S) \leq \pi_p$.
**Proof.** Suppose the probability labels on clauses in $S_p$ are $p_1, .., p_n$ then $Q(a|S) = p_1 q_1 + .. + p_n q_n$ where each $q_i$ is a sum of products for which $0 \leq q_i \leq 1$. Thus $Q(a|S) \leq p_1 + .. + p_n = \pi_p$.

## 3 Generalisation model

We assume an ILP framework in which we are provided with a background SLP $S$ which corresponds to the underlying logic program $B = P(S)$. It is further assumed that $S$ is complete. In addition we are given a set of ground unit positive examples $E$. The aim is to construct a labelled definite clause $x : H$ which when added to $S$ gives the SLP $S'$. The clause $H$ must be such that

$$B \wedge H \models E.$$

Suppose $x : H$ is placed within definition $S'_q$. By replacing all other labels $y_i$ in $S_q$ by $y_i(1-x)$ in $S'_q$ we can ensure that $\pi_q$ remains 1. The label $x$ is chosen to maximise the likelihood $p(E|S')$ where

$$p(E|S') = \prod_{e \in E} \sum_{p \in SS(e,S')} \prod_{l \text{ in } p} l \tag{1}$$

and $SS(e, S')$ represents the set of SSLD derivations of e from $S'$.

## 4 Optimal parameter choice

*Remark 2. Assuming that S is non-recursive, the term $\prod_{l \in p} l$ either has the form*

**c** *(where c is a constant) if proof p does not involve a q clause,*
**cx** *if proof p involves $x : H$ or*
**c(1-x)** *if proof p involves a clause other than $x : H$ in q.*

If we vary the probability label $x$ then $p(E|S')$ is maximal when

$$\frac{\mathrm{d}\ p(E|S')}{dx} = 0 \tag{2}$$

Unfortunately the term-size of a differential of a product of sums of products increases rapidly in the number of summed terms due to the form of the product rule. We take the alternative route of differentiating ln $p(E|S')$. The following theorem allows us to identify the solution of Equation (2) with the solution for the maximum of ln $p(E|S')$.

**Theorem 1.** *Every differentiable function $f(x)$ is maximal when $\frac{d\ ln\ f(x)}{dx} = 0$.*
**Proof.** *Let $g(x) = \frac{d\ f(x)}{dx}$ and $h(x) = \frac{d\ ln\ f(x)}{dx} = \frac{1}{f(x)} \frac{d\ f(x)}{dx} = \frac{g(x)}{f(x)}$. From calculus $f(x)$ is maximal when $g(x) = 0$. In this case $h(x) = \frac{0}{f(x)} = 0$ when $f(x) \neq 0$.*

### 4.1 General case

Given the assurances of Theorem 1 we now log transform Equation (1) as follows.

$$\begin{aligned}
\ln \mathrm{p}(E|S') &= \sum_{e\in E} \ln \left[ \sum_{p\in SS(e,S')} \prod_{l\in p} l \right] \\
&= \sum_{e\in E} \ln U(e,x) \\
\frac{\mathrm{d}\ln \mathrm{p}(E|S')}{dx} &= \sum_{e\in E} \frac{\mathrm{d}\ln U(e,x)}{dx} \\
&= \sum_{e\in E} \frac{1}{U(e,x)} \frac{\mathrm{d}\,\mathrm{U}(e,x)}{dx} \qquad (3)
\end{aligned}$$

From Remark 2 above we get the following.

$$U(e,x) = x(c_1 + c_2 + \ldots) + (1-x)(d1 + d2 + \ldots) + c(e) \qquad (4)$$

where $c_i, d_j$ are products of probability labels from $S$ and $c(e)$ is a sum of products of labels from $S$. We can now simplify Equation (4) as follows.

$$U(e,x) = xk_1(e) + k_2(e) \qquad (5)$$

where $c = c_1 + c_2 \ldots$, $d = d_1 + d_2 \ldots$, $k_1(e) = (c-d)$, $k_2(e) = d + c(e)$. Combining Equation (4) and (5) gives the following.

$$\begin{aligned}
\frac{\mathrm{d}\ln \mathrm{p}(E|S')}{dx} &= \sum_{e\in E} \frac{k_1(e)}{xk_1(e) + k_2(e)} \\
&= \sum_{e\in E} \frac{1}{x + \frac{k_2(e)}{k_1(e)}} \\
&= \sum_{e\in E} \frac{1}{x + k(e)} \qquad (6)
\end{aligned}$$

where $k(e) = \frac{k_2(e)}{k_1(e)}$.

The following theorem defines the general case for finding the value of $x$ which maximises $p(E|S')$.

**Theorem 2.** *If $p(E|S) \neq 0$ then $p(E|S')$ is maximal when $x$ takes a value defined by $\sum_{e\in E} \frac{1}{x+k(e)} = 0$.*
**Proof.** *Follows trivially from Theorem 1 and Equation (6).*

### 4.2 Analytical solution for two example case

We now demonstrate how the general equation from Theorem 2 can be solved analytically for the case in which $E$ contains only two examples, $e_1$ and $e_2$. In terms of ILP this corresponds to the case of finding the probability label for the least general generalisation of two examples. In this case

$$\begin{aligned}\sum_{e\in E}\frac{1}{x+k(e)} &= 0\\ \Rightarrow (x+k(e_1))+(x+k(e_2)) &= 0\\ \Rightarrow x &= -\left[\frac{k(e_1)+k(e_2)}{2}\right]. \end{aligned} \tag{7}$$

We demonstrate this analytical solution below by way of an example.

*Example 3.* Let $S'$ be

$$S' = \begin{cases} x: & p(X,Y) \leftarrow q(X,Z), r(Z,Y) & [A] \\ 1-x: & p(X,Y) \leftarrow r(X,Z), s(Y,Z) & [B] \\ & & \\ 0.3: & q(a,b) \leftarrow & [C] \\ 0.4: & q(b,b) \leftarrow & [D] \\ 0.3: & q(c,e) \leftarrow & [E] \\ & & \\ 0.4: & r(b,d) \leftarrow & [F] \\ 0.6: & r(e,d) \leftarrow & [G] \\ & & \\ 0.9: & s(d,d) \leftarrow & [H] \\ 0.1: & s(e,d) \leftarrow & [I] \end{cases}$$

and $E$ be

$$E = \begin{cases} p(b,d) \leftarrow [e_1] \\ p(c,d) \leftarrow [e_2] \end{cases}$$

Now the proofs for $e_1$ and $e_2$ are as follows.

$$\mathrm{SSLD}(e_1,S) = \{\langle A,D,F\rangle, \langle B,F,H\rangle\}$$
$$\mathrm{SSLD}(e_2,S) = \{\langle A,E,G\rangle, \langle B,G,I\rangle\}$$

Given these proofs the likelihood function is as follows.

$$p(E|S) = [x(0.4)(0.4) + (1-x)(0.4)(0.9)] * [x(0.3)(0.6) + (1-x)(0.6)(0.1)]$$

This function is plotted in Figure 1. Now for $e_1$ we have the following.

$$\begin{aligned} U(e_1,x) &= x(c_1+c_2+\ldots) + (1-x)(d_1+d_2+\ldots) + c(e_1)\\ c = c_1 &= (0.4)(0.4) = 0.16\\ d = d_1 &= (0.4)(0.9) = 0.36\\ c(e_1) &= 0\\ k_1(e_1) &= (c-d) = -0.20\\ k_2(e_1) &= (d + c(e_1)) = 0.36\\ k(e_1) &= \frac{k_2(e_1)}{k_1(e_1)} = \frac{0.36}{-0.20} = -1.8 \end{aligned}$$

By similar reasoning we have the following for $e_2$.

$$k(e_2) = \frac{k_2(e_2)}{k_1(e_2)} = \frac{0.06}{0.12} = 0.5$$

Thus according to Equation (7) the optimal value for $x$ is as follows.

$$\begin{aligned} x &= -\frac{k(e_1) + k(e_2)}{2} \\ &= 0.65 \end{aligned}$$

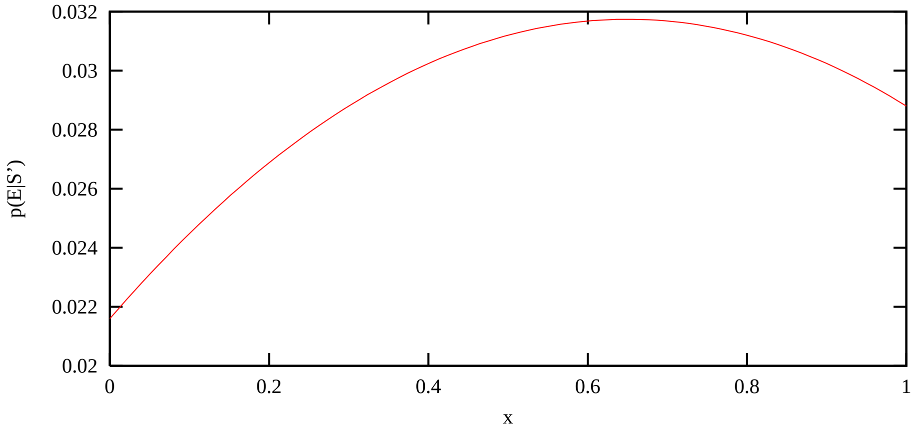


**Fig. 1.** Likelihood function for Example 3

### 4.3 Iterative numerical method for $n$ example case

It is in fact possible to find analytical solutions for numbers of examples $n = 3, 4, \ldots$. This is done by solving quadratic, cubic and higher-order equations. However, the size of the functions to be solved grows exponentially in $n$, and some of the roots are imaginary. As an alternative to this approach we can use numerical techniques to solve roots of the general equation found in Theorem 2. Standard methods here include Newton's method and the iteration method[2]. In practice the author has found that the following variant of the iteration method appears to converge rapidly.

**Variant of iteration method** Suppose that you can put an equation g(x)=0 into the form x = f(x). Start with an approximation $x_0$ of the root. Calculate

[2] See http://www.ping.be/math/root.htm

$x_0, x_1, \ldots, x_n, \ldots$ such that

$$x_{i+1} = \frac{x_i + f(x_i)}{2}$$

The sequence converges to the root as long as the initial approximation is sufficiently close. The following is a derivation from the general equation of Theorem 2 of a suitable function $f(x)$.

$$\begin{aligned}
\sum_{e \in E} \frac{1}{x + k(e)} &= 0 \\
\sum_{e \in E \setminus e_1} \frac{1}{x + k(e)} &= -\left[\frac{1}{x + k(e1)}\right] \\
x &= -\left[k(e_1) + \frac{1}{\sum_{e \in E \setminus e_1} \frac{1}{x+k(e)}}\right] \\
&= f(x)
\end{aligned}$$

*Example 4.* We revisit example 3.

$$\begin{aligned}
f(x) &= -\left[-1.8 + \frac{1}{\sum_{einE \setminus e_1} \frac{1}{x+k(e)}}\right] \\
&= -[-1.8 + (x + 0.5)] \\
&= 1.8 - 0.5 - x \\
&= 1.3 - x
\end{aligned}$$

Choosing $x_0 = 0.5$ as the initial guess the sequence converges as follows.

$$\begin{aligned}
x_0 &= 0.5 \\
x_1 &= 0.65 \\
x_2 &= 0.65
\end{aligned}$$

## 5 Discussion of related work

This section describes some of the related approaches which have been taken to learning probabilistic logic representations.

### 5.1 Learning PRMS

PRMs share the underlying probabilistic semantics and local independence assumptions of Bayesian Networks. This has allowed many of the Bayesian net learning techniques to be extended to PRMs. For instance, Koller and Pfeffer [5] have used EM (Expectation Maximisation [3]) to estimate the parameters $\theta_S$ of a PRM for which the dependency structure is known. Unlike the algorithms described in the present paper, EM is not guaranteed to converge to optimal values. However, the multi-variate parameter estimation problem being attacked by Koller and Pfeffer using EM is considerably harder than the univariate problem considered here. More recently Friedman et al. [4] have also attacked the more difficult problem of learning the dependency structure $S$ directly from data.

### 5.2 Learning SLPs

The task of learning SLPs, like that of learning PRMs, has previously been divided into that of parameter estimation and structure learning. Cussens [2] presents an algorithm called Failure-Adjusted Maximisation (FAM) which estimates from data the parameters of an SLP for which the underlying logic program is given. FAM is an instance of the EM algorithm that applies specifically to normalised SLPs. Recently the author [12] presented a two-phase algorithm for learning both the parameters of an SLP and the underlying logic program from data. The algorithm is based on maximising Bayes' posterior probability, and has been demonstrated on problems involving learning an animal taxonomy and a simple English grammar.

## 6 Conclusions and further work

In this paper we have considered ways in which both the structure and the parameters of an SLP can be learned simultaneously. The paper assumes an ILP algorithm, such as Progol or FOIL, in which clauses are constructed independently. We have derive analytical and numerical methods for efficient computation of the optimal probability parameters for a single clause choice within such a search.

Further analysis is required for some of the approaches described in this paper. For instance, the simplifying assumption made in Remark 2 that $S$ be non-recursive may be overly-restrictive in some cases, though it is almost identical to the $C$-derivation assumption used in relative least generalisation [13, 10]. Also the convergence rate of the iteration method described in Section 4.3 needs to be analysed, and compared to alternatives approaches. In particular, empirical comparisons should be made against Cussen's FAM algorithm [2] when the latter is restricted to the case of single label estimation.

Further work is required to implement and test the approach described in this paper. The intention is to incorporate the iterative numerical method described in Section 4.3 within a version of Progol. The performance of this new version of Progol can then be compared with the two stage implementation described in [8]. The author believes that the new approach has the potential to improve on that described in [8] in terms of both efficiency and accuracy of the generated result.

## Acknowledgements

Many thanks are due to my wife, Thirza and daughter Clare for the support and happiness they give me. This work was supported partly by the ESPRIT IST project "Application of Probabilistic Inductive Logic Programming (APRIL)", the EPSRC grant "Closed Loop Machine Learning" and the BBSRC/EPSRC Bio-informatics and E-Science Programme, "Studying Biochemical networks using probabilistic knowledge discovery".

## References

1. J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 126–133, San Francisco, 1999. Kaufmann.
2. J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 2000. In press.
3. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
4. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI-99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1309, San Mateo, CA:, 1999. Morgan-Kaufmann.
5. D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *IJCAI-97: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1316–1321, San Mateo, CA:, 1997. Morgan-Kaufmann.
6. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
7. S. Muggleton. Inductive logic programming: issues, results and the LLL challenge. *Artificial Intelligence*, 114(1–2):283–296, December 1999.
8. S. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 5(041), 2000.
9. S. Muggleton. Learning from positive data. *Machine Learning*, 2001. Accepted subject to revision.
10. S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, pages 368–381, Tokyo, 1990. Ohmsha.
11. S.H. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
12. S.H. Muggleton. Learning stochastic logic programs. In Lise Getoor and David Jensen, editors, *Proceedings of the AAAI2000 workshop on Learning Statistical Models from Relational Data*. AAAI, 2000.
13. G. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6. Edinburgh University Press, 1971.
14. J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.