Machine Learning, 44, 245–271, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Parameter Estimation in Stochastic Logic Programs

JAMES CUSSENS jc@cs.york.ac.uk
Department of Computer Science, University of York, Heslington, York YO10 5DD, UK

Editors: Peter Flach and Sašo Džeroski

Abstract. Stochastic logic programs (SLPs) are logic programs with parameterised clauses which define a log-linear distribution over refutations of goals. The log-linear distribution provides, by marginalisation, a distribution over variable bindings, allowing SLPs to compactly represent quite complex distributions.

We analyse the fundamental statistical properties of SLPs addressing issues concerning infinite derivations, 'unnormalised' SLPs and impure SLPs. After detailing existing approaches to parameter estimation for log-linear models and their application to SLPs, we present a new algorithm called *failure-adjusted maximisation (FAM)*. FAM is an instance of the EM algorithm that applies specifically to normalised SLPs and provides a closed-form for computing parameter updates within an iterative maximisation approach. We empirically show that FAM works on some small examples and discuss methods for applying it to bigger problems.

Keywords: logic programming, parameter estimation, EM algorithm

1. Introduction

There is currently considerable interest amongst the AI community in developing probabilistic knowledge representations that extend existing approaches to incorporate domain knowledge and/or relational data (Ngo & Haddaway, 1997; Koller & Pfeffer, 1997; Koller & Pfeffer, 1998; Kersting & De Raedt, 2000; Muggleton, 2000; Cussens, 2000).

One approach to this problem is to integrate probabilistic and 'logical' methods: an idea that has a long history dating back to Boole (1854) and which is extensively discussed in Cussens (1999a). A common idea is to use a set of formulae in first-order logic (a *first-order theory*) to encode domain knowledge. Such a theory can describe *relations* between objects. Probability can then be 'added' so that the probability with which an object has some attribute depends on the attributes which related objects have (Friedman et al., 1999) or, more generally, so that probabilities depend on what is entailed by the logically-encoded domain knowledge (Ngo & Haddaway, 1997).

As well as arguing that these more complex models are required to adequately represent complex probabilistic knowledge, workers have addressed the question of algorithms for inference in, and learning of, such statistical-logical models. This paper focuses exclusively on the parameter learning problem for a particular choice of statistical-logical model: stochastic logic programs (SLPs) (Muggleton, 1996; Cussens, 1999b; Muggleton, 2000; Cussens, 2000).

Muggleton (1996) explicitly introduced SLPs as generalisations of hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs). Statistical approaches, often using

HMMs and SCFGs, have revolutionised computational linguistics (Charniak, 1993). More recently, there has been work using Maximum Entropy methods applied to non-context-free models (Abney, 1997; Riezler, 1998). SLPs fall into this non-context-free category, and can be seen as a special case of Riezler's probabilistic constraint logic programs (Riezler, 1998) which Riezler uses to represent constraint grammars.

However, SLPs are applicable to more than computational linguistics. Muggleton (1996) argued that SLPs can be used within inductive logic programming (ILP). ILP is machine learning within a logical framework where logic programs are induced from data and domain knowledge and where there already exists valuable work involving probabilistic methods (Pompe & Kononenko, 1995; Muggleton, 1996; Pompe & Kononenko, 1997; Flach & Lachiche, 1999). Most notable is Dehaspe (1997) where, in work related to the current paper, Dehaspe combines maximum entropy and ILP methods in the MACCENT algorithm. The connections and contrasts between Dehaspe's *log-linear models with clausal constraints* and SLPs can be found in Cussens (1999b). Despite this existing work, it is fair to say that probabilistic methods in ILP require considerable development if ILP is to realise its potential for statistical relational learning. From a statistical point of view most ILP is lop-sided, focussing exclusively on the structure of the model (i.e. the induced logic program) at the expensive of parameter learning. This clearly must be remedied if ILP is to become better at inducing models that represent uncertainty.

This paper is lop-sided in the opposite direction, focusing exclusively on a *parametric statistical* analysis of SLPs, so that a large number of topics examined in related work are expressly left out. Most noticeably, the current paper does not examine structure learning as Dehaspe did. Also there is no attempt to connect the semantics of SLPs to that of logic programs as in Muggleton (2000). There is also little on the connections between SLPs and related approaches, which are discussed in Cussens (1999a, 1999b).

The paper is organised as follows. Section 2 is devoted entirely to defining terminology. Section 3 introduces SLPs and the probability distributions they define. A number of different types of SLPs are examined, and the relationships between SLPs and three existing probabilistic models are given. The core of the paper is Section 4, which concerns parameter estimation for SLPs. Conclusions and pointers to further work are in Section 5.

2. Logic programming definitions

For convenience, here is a brief overview of the more important logic programming concepts. For more details, the reader can consult any standard textbook on logic programming (e.g., Lloyd, 1987). *Definite* (*logic*) *programs* consist of a set of definite clauses, where each *definite clause* is a disjunctive first-order formula such as $p(X, Y) \vee \neg q(X, Z) \vee \neg r(Z)$ which we will write as $p(X, Y) \leftarrow q(X, Z), r(Z)$. All variables are implicitly universally quantified. We will denote variables by names starting with upper-case letters and use lower-case letters for predicate and function symbols. A *literal* is an atomic formula (briefly *atom*) or the negation of an atom. Definite clauses consist of exactly one positive literal ($p(X, Y)$ in our example) and zero or more negative literals (such as $q(X, Z)$ and $r(Z)$). The positive literal is the *head* of the clause and the negative literals are the *body*. If C is a clause, C^+ denotes the head of C and C^- the body.

A *goal* or *query* is of the form $\leftarrow A_1, \ldots, A_M$, where $A_1, \ldots, A_M$ are all atoms. If $M = 0$ then we have the empty goal denoted by $\Box$. It is also convenient to introduce an extra goal `fail`. A *substitution* such as $\theta = \{X/f(a, W), Y/Z\}$ is a mapping from variables to first-order terms. A substitution θ *unifies* two atoms A_1 and A_2 if $A_1\theta$ (θ applied to A_1) is identical to $A_2\theta$. A *computation rule* is a function from goals to atoms such that the value of the function for a goal is an atom, called the *selected* atom, in that goal. (With Prolog the selected atom is always the leftmost atom of a goal.) *Resolution* is an inference rule that takes an atom A selected from a goal $\leftarrow A_1, \ldots, A, \ldots, A_M$, unifies A with the head C^+ of an *input clause* $C = C^+ \leftarrow C^-$ using a substitution θ and returns $(\leftarrow A_1, \ldots, C^-, \ldots, A_M)\theta$ as a new goal. Note that C^- may be empty.

An *SLD-derivation* of a goal G_0 using a logic program $\mathcal{P}$ via a computation rule *Sel* is a (finite or infinite) sequence of goals with the following properties.

- The initial goal in the sequence is G_0
- If $G_j = \Box$ or $G_j =$ `fail` then the sequence terminates at G_j.
- If $G_j \neq \Box$ and $G_j \neq$ `fail`, then G_{j+1} is produced from G_j using *resolution* as follows:
  - Let A_j be the atom of the non-empty goal G_j selected using *Sel*.
  - Let C_j be any clause in $\mathcal{P}$ such that C_j^+ and A_j have the same predicate symbol. (C_j will have its variables renamed so that C_j and G_j have no variables in common.)
  - Let θ_j be the most general unifier of A_j and C_j^+ if one exists, otherwise it is undefined.

  If θ_j is undefined then $G_{j+1} =$ `fail`. Otherwise, G_{j+1} is the result of replacing A_j by C_j^- in G_j and then applying θ_j to the result. Note that this may mean that $G_{j+1} = \Box$ (is empty).

We will abbreviate SLD-derivation to *derivation*. Clearly, all finite derivations consist of a finite number of goals followed by either `fail` or $\Box$. Since the computation rule is fixed and the initial goal is given, an SLD-derivation is completely determined (up to renaming of variables) by the sequence of clauses it uses $(C_0, C_1, C_2, \ldots)$ and where appropriate we will denote a derivation by this clause sequence.

The *SLD-tree* for a goal G_0 is a tree of goals, with G_0 as root node, and such that the children of any goal G are all the goals which can be produced by one resolution step from G and a choice of clause. ($\Box$ and `fail` have no children.) An *SLD-refutation* is a finite SLD-derivation ending in the empty goal $\Box$. We will sometimes refer to an SLD-refutation as a successful derivation and will also abbreviate it to *refutation*. Branches of the SLD-tree ending in the empty goal are *success branches* corresponding to refutations (successful derivations). A *computed answer* for a goal G_0 is a substitution for the variables in G_0 produced by an SLD-refutation of G_0. The substitution is found by composing the θ_j produced during the refutation and then restricting to just those variables in G_0.

We will often use Prolog notation, where $p(X, Y) \leftarrow q(X, Z), r(Z)$ is represented thus `p(X,Y):-q(X,Z), r(Z).` and $\leftarrow q(X, a)$ is represented thus `:- q(X,a).` We will use the symbol x to represent an SLD-derivation and r to represent an SLD-refutation. $R(G)$ denotes the set of all refutations of G; $R(G)$ can be empty, finite or infinite. $D(G)$ denotes the set of all derivations of G.

3. Stochastic logic programs

Kameya and Sato (2000) note that "there are two different basic attitudes towards the use of probability in logic or logic programming". These are the *constraint* approach where the probability that a logical formula is true is constrained to lie in some region and the *distribution* approach where a specific probability distribution is defined which gives the probability that each logical formula is true. However, it is common to both strands that the probability distributions in question are defined over 'possible worlds' (first-order models) which (by marginalisation) give for each closed logical formula the probability that it is true. For example, in Poole (1997) "Possible worlds are built by choosing propositions from sets of independent choice alternatives".

It is important to understand that SLPs do *not* define distributions of this nature. An SLP $\mathcal{S}$ with parameters λ together with a goal G defines up to three different related distributions: $\psi_{(\lambda,\mathcal{S},G)}$, $f_{(\lambda,\mathcal{S},G)}$ and $p_{(\lambda,\mathcal{S},G)}$, defined over derivations, refutations and atoms, respectively. None of these necessarily define a distribution over possible worlds, although it may be possible to encode such a distribution using an SLP. In particular, as shown in Section 3.1.3, $p_{(\lambda,\mathcal{S},G)}$ defines a distribution over atoms, *not* over the truth values of atoms.

Definition 1. A *stochastic logic program* (SLP) $\mathcal{S}$ is a definite logic program where some of the clauses are parameterised with non-negative numbers. A *pure SLP* is an SLP where all clauses have parameters, as opposed to an *impure SLP* where not all clauses have parameters. A *normalised SLP* is one where parameters for clauses whose heads share the same predicate symbol sum to one. If this is not the case, then we have an *unnormalised SLP*.

Figure 1 shows $\mathcal{S}_0$, a very simple example of a pure normalised SLP. Note that the first clause in $\mathcal{S}_0$, although syntactically legal, would not be found in a real logic program, since it is logically equivalent to the simpler clause $s(X) \leftarrow p(X)$. However, the distributions defined by SLPs depend on the syntactic structure of the underlying logic program so that replacing $0.4 : s(X) \leftarrow p(X), p(X)$ by $0.4 : s(X) \leftarrow p(X)$ would change the probability distributions defined by $\mathcal{S}_0$.

This section provides formal definitions for SLPs, starting with pure normalised SLPs in Section 3.1 and then moving on to consider unnormalised and impure SLPs in Sections 3.2 and 3.3 respectively. In Section 3.4, SLPs are related to Bayesian nets, Markov nets and stochastic context-free grammars.

3.1. Pure normalised SLPs

3.1.1. Defining distributions over derivations. We begin by describing pure normalised SLPs since these have a particularly simple characterisation in terms of Markov chains. In

```
0.4:s(X) :- p(X), p(X).      0.3:p(a).     0.2:q(a).
0.6:s(X) :- q(X).            0.7:p(b).     0.8:q(b).
```

Figure 1. $\mathcal{S}_0$: A simple pure, normalised SLP.

the definition of SLD-derivation given in Section 2, C_j the jth clause in a derivation can be any clause such that the predicate symbol of C_j^+ matches that of A_j, the jth selected atom. Recall that, assuming a fixed computation rule and a given initial goal, a derivation is determined by its choice of clauses. It follows that there is only room for probability in the choice of clause. In a pure normalised SLP each choice for C_j 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 $l(C_i)$ denotes the probability that C_i is the next clause used in a derivation given that C_i^+ has the correct predicate symbol. Abbreviate $l(C_i)$ to l_i and write $\lambda_i = \log l_i$ where log denotes natural logarithm. In this paper both λ_i and l_i are used to describe SLP parameters.

The clause parameters thus define transition probabilities between goals. We will now define these transition probabilities formally. To do so consider the set $\{G_\kappa\}_\kappa$ of all possible goals and define $p_{\kappa,\kappa'}$ the probability of moving from G_κ to $G_{\kappa'}$. Together with a_κ the probability of beginning with G_κ, there is enough to define the Markov chain defined by a pure normalised SLP and an initial goal. The set $\{G_\kappa\}_\kappa$ can be all goals in some particular first-order language, but typically only those appearing in the SLD-tree of the initial goal G_0 are of interest. (Readers unfamiliar with Markov chains are recommended (Feller, 1950) as an introduction.)

Definition 2. Let $\mathcal{S}$ be a pure normalised SLP and let G_0 be a goal. $\mathcal{S}$ and G_0 define a Markov chain where the states of the Markov chain are goals G_κ and where $a_\kappa = 1$ if $G_\kappa = G_0$ and $a_\kappa = 0$ otherwise

$$p_{\kappa\kappa'} = \begin{cases} l_i & \text{if } G_{\kappa'} \text{ is a child of } G_\kappa \text{ produced by using clause } C_i \\ 1 & \text{if } G_\kappa = G_{\kappa'} = \Box \\ 1 & \text{if } G_\kappa = G_{\kappa'} = \texttt{fail} \\ 0 & \text{otherwise} \end{cases}$$

$\Box$ and `fail` are *absorbing* states of the Markov chain—once they are reached the chain remains stuck there forever. We are therefore mapping finite derivations to infinite realisations of the Markov chain where only a finite initial sequence of states are not $\Box$ or `fail`. This is common in the application of Markov chains since it means we can conveniently situate all sequences in the single sample space of infinite sequences. (We will nonetheless continue to refer to finite derivations whenever this is more intuitive.)

Since the a_κ and $p_{\kappa\kappa'}$ define a Markov chain it follows immediately that, for any n, we have a probability distribution over all sequences of goals of length n and also a probability distribution over all infinite sequences of goals. However, it is clear from the definitions of a_κ and $p_{\kappa\kappa'}$ that any sequence of goals that does not correspond to an SLD-derivation starting with G_0 has probability zero. It follows that our Markov chain defines a probability distribution over SLD-derivations starting with G_0.

For any SLD-derivation starting with G_0, it is not difficult to see that the probability of that derivation, as defined by the Markov chain, is $\prod_{i=1}^{n} l_i^{\nu_i}$ where ν_i is the number of times the clause C_i is used in that derivation. We now state this formally, by defining a distribution $\psi_{(\lambda,\mathcal{S},G)}(x)$ over $D(G)$, the set of all SLD-derivations which begin with the goal

G and which use a pure normalised SLP $\mathcal{S}$ with parameters λ. $D(G)$ contains (i) infinite derivations, (ii) finite derivations ending in $\square$ (refutations of G) and (iii) finite derivations ending in `fail`. Definition 3 (and later Definition 4) is an adaptation of a definition found in Riezler (1998).

Definition 3. Let G be a goal, and $\mathcal{S}$ be a pure normalised SLP. $\mathcal{S}$ defines a probability distribution $\psi_{(\lambda,\mathcal{S},G)}(x)$ on the set $D(G)$ (the set of derivations starting with G using $\mathcal{S}$) s.t. for all $x \in D(G)$:

$$\psi_{(\lambda,\mathcal{S},G)}(x) = e^{\lambda\cdot\nu(x)} = \prod_{i=1}^{n} l_i^{\nu_i(x)}$$

$\lambda = (\lambda_1, \ldots, \lambda_n) \in I\!R^n$ is a vector of log-parameters where λ_i is the log of l_i, the parameter attached to the ith parameterised clause,
$\nu = (\nu_1, \ldots, \nu_n)$ is a vector of clause counts s.t. for each $\nu_i : D(G) \rightarrow I\!N \cup \{\infty\}$, $\nu_i(x)$ is the number of times the ith parameterised clause is used as an input clause in derivation x,
$\lambda \cdot \nu(x)$ is a weighted count s.t. $\lambda \cdot \nu(x) = \sum_{i=1}^{n} \lambda_i \nu_i(x)$,

Usually it will be clear which SLP and goal is being used to define the distribution, so we will often abbreviate $\psi_{(\lambda,\mathcal{S},G)}(x)$ to $\psi_\lambda(x)$.

3.1.2. Defining distributions over refutations. Our main focus of interest is not $\psi_{(\lambda,\mathcal{S},G)}(x)$ but the conditional distribution $\psi_{(\lambda,\mathcal{S},G)}(x \mid x \in R(G))$, the distribution over derivations given that each is a refutation. We will denote this distribution by $f_{(\lambda,\mathcal{S},G)}$:

$$f_{(\lambda,\mathcal{S},G)}(x) \overset{\text{def}}{=} \psi_{(\lambda,\mathcal{S},G)}(x \mid x \in R(G))$$

Let

$$Z_{(\lambda,\mathcal{S},G)} \overset{\text{def}}{=} \sum_{x\in R(G)} \psi_{(\lambda,\mathcal{S},G)}(x) = \psi_{(\lambda,\mathcal{S},G)}(R(G))$$

then

$$f_{(\lambda,\mathcal{S},G)}(x) = \begin{cases} Z^{-1}_{(\lambda,\mathcal{S},G)}\psi_{(\lambda,\mathcal{S},G)}(x) & \text{if } x \in R(G) \\ 0 & \text{if } x \in D(G)\backslash R(G) \end{cases}$$

$f_{(\lambda,\mathcal{S},G)}(x)$ is a *log-linear model* over refutations, defined for goals where $Z_{(\lambda,\mathcal{S},G)} > 0$. To see this, consider Definition 4, where $f_{(\lambda,\mathcal{S},G)}$ is defined without reference to $\psi_{(\lambda,\mathcal{S},G)}$. The only slight alteration to a standard log-linear model is that it is extended so that derivations which are not refutations have probability zero, rather than being undefined.

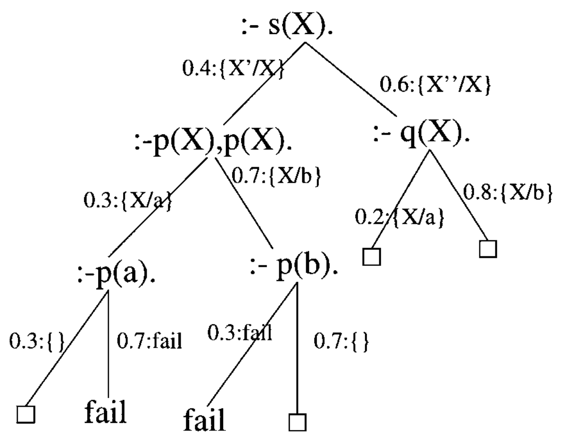


Figure 2. Annotated SLD-tree for $\mathcal{S}_0$.

Definition 4. Let G be a goal such that $Z_{(\lambda,\mathcal{S},G)} > 0$, and $\mathcal{S}$ be a pure normalised SLP. $\mathcal{S}$ defines a log-linear probability distribution $f_{(\lambda,\mathcal{S},G)}(r)$ on the set $R(G)$ (the set of refutations of G using $\mathcal{S}$) s.t. for all $r \in R(G)$:

$$f_{(\lambda,\mathcal{S},G)}(r) = Z^{-1}_{(\lambda,\mathcal{S},G)} e^{\lambda\cdot\nu(r)} = Z^{-1}_{(\lambda,\mathcal{S},G)} \prod_{i=1}^{n} l_i^{\nu_i(r)}$$

$Z_{(\lambda,\mathcal{S},G)} = \sum_{r\in R(G)} e^{\lambda\cdot\nu(r)}$ is a normalizing constant, λ, ν and $\lambda\cdot\nu(r)$ are defined as in Definition 3.

We extend the normal log-linear definition so that $f_{(\lambda,\mathcal{S},G)}(x) = 0$, if $x \in D(G)\backslash R(G)$. $f_{(\lambda,\mathcal{S},G)}(x)$ is thus a distribution over the whole of $D(G)$.

The distributions ψ_λ and f_λ are more easily understood by referring to the SLD-tree which underlies them. By way of example, figure 2 shows an annotated SLD-tree for refutations of the goal $\leftarrow s(X)$ using $\mathcal{S}_0$. There are 6 derivations, of which 4 are successful and 2 are failures. The branches of the tree are labelled with (i) the unification effected by choosing clauses and (ii) the parameters attached to these clauses. Since $\mathcal{S}_0$ is pure and normalised ψ_λ is a probability distribution over derivations, and the tree shows how the probability mass of one is divided up as we move down the tree. To find $\psi_\lambda(x)$ for any derivation x we multiply the parameters on the branches corresponding to that derivation. Both failure derivations have probability 0.084, so $Z_{(\lambda,\mathcal{S}_0,\leftarrow s(X))} = 1 - 2 \times 0.084 = 0.832$. So, for example, if the leftmost refutation is r_1, then $f_{(\lambda,\mathcal{S}_0,\leftarrow s(X))}(r_1) = (0.4 \times 0.3 \times 0.3)/0.832 \approx 0.043$ (The tree assumes that the variable in the two $s/2$ clauses is renamed to X' and X''.)

3.1.3. Defining distributions over atoms. f_λ defines a distribution over atoms via marginalisation. First define the *yield* of a refutation and the *proofs* of an atom.

Definition 5. The *yield* $Y(r)$ of a refutation r of unit goal $G = \leftarrow A$ is $A\theta$ where θ is the computed answer for G using r. The set of proofs for an atom y_k is the set $X(y_k) = \{r|Y(r) = y_k\}$. Note that $X(Y(r))$ is the set of all refutations that yield the same atom as r.

We only define yields with respect to unit goals. This is just a convenience, since given a non-unit goal $\leftarrow G_1, \ldots, G_M$, we can always add the clause $A' \leftarrow G_1, \ldots, G_M$, where A' contains all the variables of $G_1, \ldots, G_M$, and then consider yields of $\leftarrow A'$. Note that from a logical perspective a refutation of $\leftarrow A$ with computed answer θ amounts to a proof of $A\theta$, so this choice of terminology is natural.

We now define a distribution $p_{(\lambda,\mathcal{S},G)}$ over atoms in terms of their proofs.

$$p_{(\lambda,\mathcal{S},G)}(y_k) \stackrel{\text{def}}{=} \sum_{r\in X(y_k)} f_{(\lambda,\mathcal{S},G)}(r) = Z^{-1}_{(\lambda,\mathcal{S},G)} \sum_{r\in X(y_k)} e^{\lambda\cdot\nu(r)} \tag{1}$$

If G has t variables, then $p_{(\lambda,\mathcal{S},G)}(y_k)$ defines a t-dimensional distribution over variable bindings for these t variables. Note that we allow non-ground bindings unlike in (Muggleton, 1996; Cussens, 1999b; Mugleton, 2000). We will see in Section 3.4 how we can use these t-dimensional distributions to encode probabilistic models using other formalisms into SLPs. Returning to our example SLP $\mathcal{S}_0$ we find that it defines a distribution over the sample space $\{s(a), s(b)\}$, where

$$p_{(\lambda,\mathcal{S}_0,\leftarrow s(X))}(s(a)) = (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2)/0.832 = 0.1875$$

and

$$p_{(\lambda,\mathcal{S}_0,\leftarrow s(X))}(s(b)) = (0.4 \times 0.7 \times 0.7 + 0.6 \times 0.8)/0.832 = 0.8125$$

It follows from the definition in (1) that we can express $p_{(\lambda,\mathcal{S},G)}(y_k)$ as a ratio of Z-values:

$$p_{(\lambda,\mathcal{S},G)}(y_k) = \frac{Z_{(\lambda,\mathcal{S},\leftarrow y_k)}}{Z_{(\lambda,\mathcal{S},G)}}$$

Z-values can be expressed recursively. Let $Z_{(\lambda,\mathcal{S},\square)} = 1$, $Z_{(\lambda,\mathcal{S},\texttt{fail})} = 0$ and for all other goals G_κ

$$Z_{(\lambda,\mathcal{S},G_\kappa)} = \sum_{G_{\kappa'}} p_{\kappa\kappa'} Z_{(\lambda,\mathcal{S},G_{\kappa'})}$$

These equations can form the basis of a variable elimination algorithm for computing $p_{(\lambda,\mathcal{S},G)}(y_k)$ as discussed in Cussens (2000).

3.2. *Unnormalised SLP*

It is clear from Definition 4 that the parameters λ of an SLP need not be normalised to define distributions f_λ and p_λ. The only condition that needs to be added to Definition 4 is that Z_λ

```
2 : p(a) :- p(b). %C1
1 : p(b).         %C2
```

Figure 3. $\mathcal{S}_{\text{UNNORM}}$, an unnormalised SLP not equivalent to any normalised SLP.

must be finite. If this condition is met but the parameters are not normalised then we have an *unnormalised* SLP. We will see in Section 3.4 that unnormalised SLPs can conveniently represent Bayesian nets.

If an SLP is normalised then each clause parameter can be interpreted as a probability—the probability with which that clause is chosen when its head has the appropriate predicate symbol. This property of normalised SLPs is exploited in the failure-adjusted maximisation (FAM) algorithm for normalised SLPs presented in Section 4.4. Given that normalised SLPs have this useful feature, it would be nice if any unnormalised SLP could be reparameterised to an equivalent normalised SLP with the same underlying logic program. Unfortunately, this is not the case.

Consider $\mathcal{S}_{\text{UNNORM}}$, the unnormalised SLP in figure 3. Clearly, $p(a)$ gets probability 2/3 and $p(b)$ gets probability $1/3$, according to $p_{(\lambda,\mathcal{S}_{\text{UNNORM}},\leftarrow p(X))}$. Consider parameters l_1 and l_2 for C_1 and C_2 giving the same distribution. It is easy to see that $l_1 = 2$ and l_2 can take any value. In particular, no normalised SLP, with the same clauses as $\mathcal{S}_{\text{UNNORM}}$ can represent $p_{(\lambda,\mathcal{S}_{\text{UNNORM}},\leftarrow p(X))}$.

The point is that because unnormalised SLPs allow clause parameters to exceed one, the derivation (C_1, C_2) can get strictly higher probability than (C_2) despite using extra clauses. This is not possible with normalised SLPs, where a refutation can be seen as a sequence of probabilistic choices between clauses: as more choices are made the probability is non-increasing.

A restriction to normalised SLPs is a real restriction. However, it is only because normalised SLPs can be viewed in terms of probabilistic choices that we can apply the FAM algorithm described in Section 4.4. Another appealing intuitive feature of normalised SLPs is that we can view refutations as *arguments* for a particular yielded atom where the more steps there are in the argument, the weaker it becomes.

3.3. *Impure SLPs*

Consider $\mathcal{S}_{\text{IMPURE}}$, the impure SLP in figure 4. Informally, we want $\mathcal{S}_{\text{IMPURE}}$ to say that James has had papers accepted for both MLJ and UAI, and because of this the atoms `james(vhappy)`, `james(happy)`, `james(ok)` and `james(unhappy)` have the probabilities given by the parameters. So we want $p_{(\lambda,\mathcal{S}_{\text{IMPURE}},\leftarrow james(X))}$ to give these 4 probabilities to the 4 possible yields produced by $\leftarrow james(X)$.

Let us attempt to find a pure SLP which defines such a probability on the four yields of $\leftarrow james(X)$. Suppose the `paper_accepted/1` clauses were parameterised with parameters l'_1 and l'_2, giving the parameter set λ'. Since we want $p_{\lambda'}(james(vhappy)) = 0.5$, and $p_{\lambda'}(james(happy)) = 0.3$, it is easy to see that we must have $l'_1 = l'_2$. We must also have $p_{\lambda'}(james(vhappy))/p_{\lambda'}(james(ok)) = 5$, but clearly since $l'_1 = l'_2$, we have $p_{\lambda'}(james(vhappy))/p_{\lambda'}(james(ok)) = 0.5l'_1/(0.1l'_1 + 0.1l'_1) = 5/2$. Therefore we can not extend $\mathcal{S}_{\text{IMPURE}}$ to a pure SLP defining the same distribution.

```
0.5 : james(vhappy) :- paper_accepted(mlj).
0.3 : james(happy)  :- paper_accepted(uai).
0.1 : james(ok)     :- paper_accepted(X).
0.1 : james(unhappy).

      paper_accepted(mlj).
      paper_accepted(uai).
```

Figure 4. $\mathcal{S}_{\text{IMPURE}}$, an impure SLP.

The desired meaning for unparameterised clauses is to see them as non-probabilistic domain knowledge acting as *constraints*. The ability to combine such domain knowledge with probabilities is the central feature of SLPs (although not unique to SLPs). In pure SLPs only equational constraints between first-order terms are possible. These are not sufficiently expressive in many cases as even the tiny linguistic examples in Cussens (2000) show.

On this view, it is enough to know that the `paper_accepted(X)` constraint in the 3rd `james/1` clause is satisfied—we do not care that is 'satisfied twice' and we do not want the fact that there are two refutations of $\leftarrow paper\_accepted(X)$ to affect the distribution, which must happen if the `paper_accepted/1` clauses are parameterised.

To effect the desired meaning we just alter Definition 4, so that the possibly many ways of satisfying our constraints are collapsed to a single element in the probability space.

Definition 6. Let $\mathcal{S}$ be an SLP (pure or impure). Identify refutations with the sequence of clauses they use. For any refutation, let $\mathsf{red}(r)$ be the clause sequence produced by deleting all unparameterised clauses. Let $\simeq$ be an equivalence relation on $R(G)$ where $r_1 \simeq r_2$ iff $\mathsf{red}(r_1) = \mathsf{red}(r_2)$. Let r be a representative of its equivalence class $\mathrm{Eq}(r)$, then

$$f_{(\lambda,\mathcal{S},G)}(\mathrm{Eq}(r)) \overset{\text{def}}{=} f_{(\lambda,\mathcal{S},G)}(r)$$

$f_{(\lambda,\mathcal{S},G)}(\mathrm{Eq}(r))$ is well-defined since each member of the equivalence class has exactly the same parameterised clauses, so $f_{(\lambda,\mathcal{S},G)}$ is the same for all of them. We write $\mathrm{Eq}(r) \in R(G)$ to mean that every refutation in $\mathrm{Eq}(r)$ is in $R(G)$.

If, for any $r \in R(G)$, all refutations in $\mathrm{Eq}(r)$ yield the same atom

$$p_{(\lambda,\mathcal{S},G)}(y_k) \overset{\text{def}}{=} \sum_{\mathrm{Eq}(r)\text{s.t. } r \in X(y_k)} f_{(\lambda,\mathcal{S},G)}(\mathrm{Eq}(r))$$

Definition 6 ensures that impure SLPs are defined to have the desired behaviour discussed previously. $\mathcal{S}_{\text{IMPURE}}$ defines the right probabilities, since the two refutations yielding *james*(*ok*) have been collapsed into one equivalence class. Definition 6 agrees with Definition 4 when the SLP happens to be pure, since then all the equivalence classes contain a single refutation. Note the restriction concerning yields which means that the SLP $\mathcal{S}_{\text{WRONG}}$ in figure 5 does not define a distribution $p_{(\lambda,\mathcal{S}_{\text{WRONG}},\leftarrow p(X,Y))}$. This is essentially because there

```
p(X,1) :- q(X).      0.6:q(a).
p(X,2) :- q(X).      0.4:q(b).
```

Figure 5. $\mathcal{S}_{\text{WRONG}}$, an SLP that *does not define* a distribution $p_{(\lambda,\mathcal{S}_{\text{WRONG}},\leftarrow p(X,Y))}$.

is an unquantified choice between the $p/2$ clauses, which is a real choice because it affects yields. Note also that Definition 6 ensures that an SLP with no parameterised clauses does not define a distribution for any goal that can yield more than one atom.

3.4. *Relations to some existing probabilistic models*

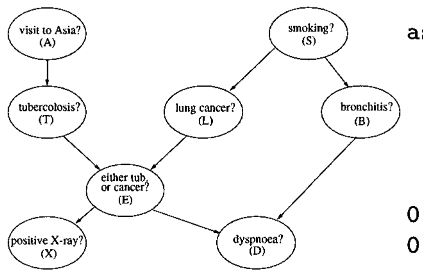
In this section we encode three familiar probabilistic models into SLPs. Considerably more complex SLPs encoding, for example, distributions over a hypothesis space of logic programs are used in Cussens (2000).

Figure 6 shows the *Asia* Bayesian network and an SLP representation of it, where $p_{(\mathcal{S},\lambda,\leftarrow asia(A,T,E,S,L,B,X,D))}$ gives the joint distribution represented by the Bayesian net. The equation

$$P(A,T,E,S,L,B,X,D)$$
$$= P(A)P(S)P(T \mid A)P(L \mid S)P(B \mid S)P(E \mid T,L)P(D \mid E,B)P(X \mid E)$$

is directly encoded using an *impure, unnormalised* SLP, with each of the 8 conditional probability tables defined by a single predicate. Since E is a function of T and L, we only need 4 unparameterised clauses to encode $P(E|T,L)$ as opposed to the 8 that would be required if $P(E|T,L)$ were encoded as the other conditional probability distributions are. It is clear that any Bayesian net with discrete variables can be represented by an SLP in this manner.

The translation from Bayesian net to SLP is problematic in that the directionality of Bayesian nets is obscured. In contrast, the mapping between Markov nets and SLPs is

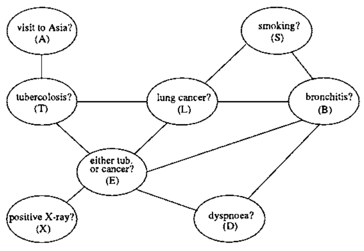


```
asia(A,T,E,S,L,B,X,D) :-
    a(A), s(S), t_a(T,A),
    l_s(L,S), b_s(B,S),
    e_tl(E,T,L), d_eb(D,E,B),
    x_e(X,E).

0.01:a(1).        0.99:a(0).
0.50:s(1).        0.50:s(0).
```

```
0.05:t_a(1,1). 0.95:t_a(0,1). 0.1:t_a(0,0). 0.9:t_a(0,0).
e_tl(0,0,0).   e_tl(1,0,1).   e_tl(1,1,0).   e_tl(1,1,1).
% l_s/2, b_s/2, d_eb/2 and x_e/2 definitions omitted
```

Figure 6. Asia Bayesian net and its encoding as an SLP.



```
asia(A,B,D,E,L,S,T,X) :-
 c6(E,X), c5(E,B,D), c4(L,B,S),
 c3(L,E,B), c2(E,L,T), c1(A,T).

0.0005:c1(1,1). 0.0095:c1(1,0).
0.0099:c1(0,1) 0.9801:c1(0,0).
%Clauses for c2, c3,
%c4, c5, c6  omitted
```

Figure 7. *Asia* Markov net and its encoding as an SLP.

```
0.5:s(A,D) :- A=[a|B], s(B,C), C=[a|D].  0.1:s([a,a|T],T).
0.3:s(A,D) :- A=[b|B], s(B,C), C=[b|D].  0.1:s([b,b|T],T).
```

Figure 8. $\mathcal{S}_{\text{PALINDROME}}$, an SLP representation of an SCFG.

transparent. Figure 7 shows a Markov net derived from the *Asia* Bayesian net and its translation to an impure unnormalised SLP. The structure of the Markov net can be completely described with a single clause, and the 6 clique potentials each get their own predicate symbol.

Since SLPs generalise stochastic context-free grammars (SCFGs) it is easy to encode SCFGs as SLPs. Consider the context-free grammar $S \rightarrow aSa \mid bSb \mid aa \mid bb$ which generates palindromes. By placing a probability distribution over the four productions we have an SCFG which defines a distribution over palindromic strings of as and bs. $\mathcal{S}_{\text{PALINDROME}}$ in figure 8 encodes such an SCFG as an SLP where $p_{(\lambda,\mathcal{S}_{\text{PALINDROME}},\leftarrow s(X,[]))}$ is the distribution over strings. Hidden Markov models, which are essentially stochastic regular grammars, can be dealt with similarly.

4. Parameter estimation in SLPs

Our goal is to estimate λ, the true values of the clause parameters of an SLP $\mathcal{S}$ whose underlying logic program is fixed and where we have decided which clauses are to be parameterised. So the features of the log-linear model are given—we just need to estimate the parameters. We assume that we have a set of atoms $y = (y_1, \ldots, y_N)$ which have been generated by $\mathcal{S}$ according to the (unknown) distribution $p_{(\lambda,\mathcal{S},G)}$, where $\mathcal{S}$ and G are given. The data define $\tilde{p}$, the empirical distribution over atoms. $\tilde{p}(y_k)$ is just the relative frequency with which the atom y_k appears in the data y.

Since SLPs are a special case of loglinear models, we can apply parameter fitting algorithms for general loglinear models to SLPs. Section 4.1 briefly summarises approaches to parameter estimation for general loglinear models following the presentation given in Della Pietra et al. (1997). Section 4.2 shows how these general methods apply to SLPs using an example taken from Abney (1997). Section 4.3 explains how Riezler has extended these approaches to deal with incomplete data. These sections are included for completeness, and can be skimmed by those familiar with this work. Section 4.4 contains the main contribution of the paper: a new method of parameter estimation for pure normalised SLPs.

4.1. Parameter estimation for loglinear models

In this section we give a brief and informal account of maximum likelihood parameter estimation for general loglinear models taken from Della Pietra et al. (1997) where a detailed and formal account of this topic can be found. Della Pietra et al. consider generalised Gibbs distributions defined over a space Ω of the form

$$p(\omega) = [Z_\lambda(q_0)]^{-1} e^{\sum_{i=1}^{n} \lambda_i \nu_i(\omega)} q_0(\omega) = [Z_\lambda(q_0)]^{-1} e^{\lambda.\nu(\omega)} q_0(\omega)$$

where $\nu(\omega) = (\nu_1(\omega), \nu_2(\omega), \dots, \nu_n(\omega))$ is a vector of feature values with $\nu_i : \Omega \to I\!\!R$, $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ is a vector of real-valued parameters and q_0 is some 'initial' distribution over Ω. In Della Pietra et al. (1997) the feature vector is denoted by 'f', but we will use ν, since in this paper f represents a distribution.)

Let the empirical distribution $\tilde{p}(\omega)$ be the relative frequency with which $\omega \in \Omega$ occurs in the training data. Let us introduce the notation *Prob*[*rv*] to mean the expected value of a random variable *rv* according to a distribution *Prob*. $\tilde{p}[\nu]$ is then the vector of empirical mean values of the features. Consider now $\mathcal{P}(\nu, \tilde{p})$, the set of distributions which agree with the data as to the expected values of the features:

$$\mathcal{P}(\nu, \tilde{p}) = \{p : p[\nu] = \tilde{p}[\nu]\} \tag{2}$$

It turns out that the MLE estimate for λ is that value of λ which defines a distribution in $\mathcal{P}(\nu, \tilde{p})$ which maximises the entropy of $p(\omega)$ relative to q_0. It is for this reason that loglinear models are often referred to as maximum entropy (MAXENT) models, particularly in the computational linguistics literature. It follows that we can do MLE by solving the equations in (2) and maximising relative entropy. This approach is applied to a small SLP parameter estimation problem in Section 4.2, but is not feasible for large problems.

As an alternative Della Pietra et al. present an iterative approach called *Improved Iterative Scaling* (IIS) which is guaranteed to converge to the MLE. To describe IIS, we need to introduce some new notation. Let

$$\nu_\#(\omega) \stackrel{\text{def}}{=} \sum_{i}^{n} \nu_i(\omega)$$

$\nu_\#(\omega)$ can be thought of as something like the total value of all features for ω. If the features are binary then $\nu_\#(\omega)$ is the number of features which are 'on' for ω.

We also introduce a notation useful for updating loglinear distributions. Let $\circ$ be defined so that, for any parameter vector γ:

$$(\gamma \circ q)(\omega) = [Z_\gamma(q)]^{-1} e^{\gamma.\nu} q(\omega)$$

It is easy to see that if

$$q(\omega) = [Z_\gamma(q_0)]^{-1} e^{\lambda.\nu} q_0(\omega)$$

then

$$(\gamma \circ q)(\omega) = \left[Z_{(\lambda+\gamma)}(q_0)\right]^{-1} e^{(\lambda+\gamma).\nu} q_0(\omega)$$

So $(\gamma \circ q)(\omega)$ is $q(\omega)$ with γ added to the parameter vector.

Definition 7 (Improved Iterative Scaling (IIS)).

1. Set $q^{(0)} = q_0$
2. For each i, let $\gamma_i^{(h)} \in [-\infty, \infty)$ be the unique solution of

$$q^{(h)}\left[\nu_i e^{\gamma_i^{(h)}\nu_\#}\right] = \tilde{p}[\nu_i] \tag{3}$$

3. Set $q^{(h+1)} = \gamma^{(h)} \circ q^{(h)}$.
4. Set $h \leftarrow h + 1$ and go to 2 unless $q^{(h)}$ has converged.

Crucially, we can solve (3) for each feature parameter in turn. The value for $\gamma_i^{(h)}$ depends only on the data and on $q^{(h)}$, not on the values $\gamma_{i'}^{(h)}$ for $i' \neq i$. If $\nu_\#(\omega) = K$ for all ω, we have a closed form solution for $\gamma_i^{(h)}$:

$$\gamma_i^{(h)} = \frac{1}{K} \log \frac{\tilde{p}[\nu_i]}{q^{(h)}[\nu_i]}$$

Otherwise it is necessary to solve a polynomial equation for each feature (Della Pietra et al. 1997). In either case expectations with respect to $q^{(h)}$ must be calculated as well as the observed frequencies $\tilde{p}[\nu_i]$. In many cases, Ω will be too complex for an exact computation of these expectations, so sampling over Ω will be required to estimate the required expectations.

4.2. *Existing approaches to complete-data parameter estimation for* SLPs

Suppose we have data y in the form of a set of atoms and we wish to perform maximum likelihood estimation (MLE) to estimate the parameters for an SLP with known structure. We will assume for the time being that the data is *complete* (or *unambiguous*) in the sense that, for each atom in the data, there is only one refutation that can yield it. This bijective mapping between refutations and atoms means that $\tilde{p}$, the empirical distribution over atoms, determines $\tilde{f}$, the empirical distribution over refutations. $\tilde{f}(r)$ is the relative frequency with which refutation r must have occurred to produce the data y.

For each refutation r we have $\nu_i(r)$, the frequency with which clause C_i is used in r. So from $\tilde{f}$, a distribution over refutations, we can compute the expected frequency of each clause C_i according to $\tilde{f}$. We will denote this expectation by $\tilde{f}[\nu_i]$. Since the distribution over refutations is a log-linear model we can apply the results for general log-linear models

```
l_1:s(X,p) :- p(X), p(X).       l_3:p(a).     l_5:q(a).
l_2:s(X,q) :- q(X).             l_4:p(b).     l_6:q(b).
```

Figure 9. $\mathcal{S}_1$: A simple SLP, unambiguous for $\leftarrow s(X, Y)$.

and find the MLE estimates for the SLP parameters by equating expectations as in (2). The MLE estimates are $\hat{\lambda}$ where

$$\forall i : \tilde{f}[\nu_i] = f_{\hat{\lambda}}[\nu_i]$$

and $f_{\hat{\lambda}}$ is the distribution with maximum entropy which meets these constraints.

Let us apply this approach to MLE using $\mathcal{S}_1$, the SLP shown in figure 9. $\mathcal{S}_1$ is an odd looking SLP, but we will use it since it has essentially the same structure as the stochastic attribute-value grammar G_2 that is used in Abney (1997).

$\mathcal{S}_1$ is unambiguous for $\leftarrow s(X, Y)$ and the bijection between the four refutations $\leftarrow s(X, Y)$ and the four atoms that they yield is given in detail in Table 1 where each 4-tuple gives the goal, selected atom, input clause and substitution used at each stage in the refutation. We also record the distributions p_λ and f_λ in Table 1.

Table 1. The mapping between refutations and atoms and the distributions p_λ and f_λ.

Atom	$s(a, p)$
Refutation	$(\leftarrow s(X, Y), s(X, Y), C_1, \{Y/p\})$
	$((\leftarrow p(X), p(X)), p(X), C_3, \{X/a\})$
	$(\leftarrow p(a), p(a), C_3, \{\})$
	$\square$
Probability	$p_\lambda(s(a, p)) = f_\lambda(C_1C_3C_3) = Z_\lambda^{-1} l_1 l_3^2$
Atom	$s(b, p)$
Refutation	$(\leftarrow s(X, Y), s(X, Y), C_1, \{Y/p\})$
	$((\leftarrow p(X), p(X)), p(X), C_4, \{X/b\})$
	$(\leftarrow p(b), p(b), C_4, \{\})$
	$\square$
Probability	$p_\lambda(s(b, p)) = f_\lambda(C_1C_4C_4) = Z_\lambda^{-1} l_1 l_4^2$
Atom	$s(a, q)$
Refutation	$(\leftarrow s(X, Y), s(X, Y), C_2, \{Y/q\})$
	$(\leftarrow q(X), q(X), C_5, \{X/a\})$
	$\square$
Probability	$p_\lambda(s(a, q)) = f_\lambda(C_2C_5) = Z_\lambda^{-1} l_2 l_5$
Atom	$s(b, q)$
Refutation	$(\leftarrow s(X, Y), s(X, Y), C_2, \{Y/q\})$
	$(\leftarrow q(X), q(X), C_6, \{X/b\})$
	$\square$
Probability	$p_\lambda(s(b, q)) = f_\lambda(C_2C_6) = Z_\lambda^{-1} l_2 l_6$

Table 2. Empirical distributions of atoms and refutations.

Count	4	2	3	3
Atoms	$s(a, p)$	$s(b, p)$	$s(a, q)$	$s(b, q)$
$\tilde{p} =$	1/3	1/6	1/4	1/4
Refutations	$C_1C_3C_3$	$C_1C_4C_4$	C_2C_5	C_2C_6
$\tilde{f} =$	1/3	1/6	1/4	1/4

Suppose our data comprise 12 atoms. Each atom should be seen as the yield of a refutation sampled from the distribution $f_{(\lambda,\mathcal{S}_1,\leftarrow s(X,Y))}$. Following Abney, we will assume that the proportions of atoms (and consequently refutations) in our data are as in Table 2. Now that we have f_λ and $\tilde{f}$ tabulated in Tables 1 and 2 respectively we can compute the expected values of clause frequencies according to these two distributions. These expectations are tabulated in Table 3.

By equating the expectations in Table 3 and manipulating the resulting set of 6 polynomials, we find that for $\hat{\lambda} = (\hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3, \hat{\lambda}_4, \hat{\lambda}_5, \hat{\lambda}_6) = (\log \hat{l}_1, \log \hat{l}_2, \log \hat{l}_3, \log \hat{l}_4, \log \hat{l}_5, \log \hat{l}_6)$ to be a set of MLE parameters we must have:

$$(\hat{l}_3)^2 = 2(\hat{l}_4)^2 \tag{4}$$
$$\hat{l}_2 = 3\hat{l}_1(\hat{l}_4)^2 \tag{5}$$
$$\hat{l}_5 = \hat{l}_6 = 1/2 \tag{6}$$

It follows that $Z_{\hat{\lambda}} = 2\hat{l}_2$, and by applying (4)–(6) to the expressions in Table 1 we find that $f_{\hat{\lambda}} = \tilde{f}$ and $p_{\hat{\lambda}} = \tilde{p}$. (For example $p_{\hat{\lambda}}(s(a, p)) = Z_{\hat{\lambda}}^{-1}\hat{l}_1(\hat{l}_3)^2 = \hat{l}_1 2(\hat{l}_4)^2/2\hat{l}_2 = \frac{2}{3}\hat{l}_2/2\hat{l}_2 = 1/3$. The other 3 probabilities follow equally easily.)

Since any parameter set satisfying (4)–(6) defines the same distribution $f_{\hat{\lambda}}$ they are obviously all MLE estimates. Our current MLE estimation problem is somewhat atypical in that there are more parameters (six) than elements in the sample space (four). This is why we have a spare two degrees of freedom in choosing MLE parameters. It also explains why we were able to find $\hat{\lambda}$ such that $f_{\hat{\lambda}} = \tilde{f}$. Usually in parameter estimation of loglinear

Table 3. Expected frequency of clauses according to f_λ and $\tilde{f}$, where $Z_\lambda = l_1(l_3^2 + l_4^2) + l_2(l_5 + l_6)$.

Clause	$f_\lambda[\nu_i]$	$\tilde{f}[\nu_i]$
C_1	$Z_\lambda^{-1}l_1(l_3^2 + l_4^2)$	1/2
C_2	$Z_\lambda^{-1}l_2(l_5 + l_6)$	1/2
C_3	$Z_\lambda^{-1}2(l_1l_3^2)$	2/3
C_4	$Z_\lambda^{-1}2(l_1l_4^2)$	1/3
C_5	$Z_\lambda^{-1}l_2l_5$	1/4
C_6	$Z_\lambda^{-1}l_2l_6$	1/4

Table 4. MLE estimates wrt f_λ for a normalised SLP.

$\hat{l}_1$	$\hat{l}_2$	$\hat{l}_3$	$\hat{l}_4$	$\hat{l}_5$	$\hat{l}_6$
$\frac{3+2\sqrt{2}}{6+2\sqrt{2}}$	$\frac{3}{6+2\sqrt{2}}$	$\frac{\sqrt{2}}{1+\sqrt{2}}$	$\frac{1}{1+\sqrt{2}}$	$\frac{1}{2}$	$\frac{1}{2}$

models there are many more points in the sample space than parameters and it is not possible to find $\hat{\lambda}$ such that $f_{\hat{\lambda}} = \tilde{f}$. If we throw in the constraint that we are only interested in $\hat{\lambda}$ which define a normalised SLP we get the unique values given in Table 4 which are also given in Abney (1997).

In summary, our approach to this small problem is (i) enumerate the 4 refutations of $\leftarrow p(X, Y)$ in $\mathcal{S}_1$ (Table 1), (ii) compute their probabilities (Table 1), (iii) count how often the clauses were used in each refutation, (iv) set up the equations (Table 3) and (v) solve for $\hat{\lambda}$. In real problems such an approach will be infeasible; there will be too many refutations to enumerate. An even more basic difficulty is that we have to solve a set of simultaneous polynomial equations—we can not solve for $\hat{\lambda}_i$ point-wise.

A feasible alternative is to use Improved Iterative Scaling (IIS) which was described for general loglinear models in Section 4.1. Applying IIS to the problem of MLE for SLPs we have

$$\nu_\#(r) = \sum_i^n \nu_i(r)$$

So $\nu_\#(r)$ is the total number of parameterised clauses used in a refutation r. If a particular parameterised clause is used more than once, then each occurrence contributes to $\nu_\#(r)$. We can now rewrite the definition of IIS to apply directly to SLPs:

Definition 8 (Improved Iterative Scaling (IIS)). Let $\lambda^{(0)}$ be the initial set of parameter estimates. Set $h = 0$

1. For each parameterised clause C_i, let $\gamma_i^{(h)} \in [-\infty, \infty]$ be the unique solution of

$$f_{\lambda^{(h)}}\left[\nu_i e^{\gamma_i^{(h)}\nu_\#}\right] = \tilde{f}[\nu_i]$$

2. Set $\lambda^{(h+1)} = \gamma^{(h)} + \lambda^{(h)}$.
3. Set $h \leftarrow h + 1$ and go to 2 unless $f_{\lambda^{(h)}}$ has converged.

If $\nu_\#$ is constant we have:

$$\gamma_i = \frac{1}{K} \log \frac{\tilde{f}[\nu_i]}{f_\lambda[\nu_i]}$$

4.3. Riezler's iterative maximization for incomplete data parameter estimation in SLPs

To use IIS directly we must have the values $\tilde{f}[\nu_i]$—where $\tilde{f}(r)$ is the relative frequency with which refutation r has occurred to produce the observed data composed of atoms. But we are not in a position to calculate these values if the atoms in our data have more than one refutation that can yield them. If we have *incomplete* or *ambiguous* data of this sort IIS needs adapting to be usable.

Riezler (1998) shows how the complete data approach can be adapted for incomplete data. Essentially we replace the *unknown actual* relative frequency ($\tilde{f}(r)$) with which refutation r has occurred, with $f_\lambda(r \mid y)$ the *known probability* of r occurring according to our current parameter estimates, conditional on y, the observed data. We have

$$f_\lambda(r \mid y) = \tilde{p}(Y(r)) f_\lambda(r \mid X(Y(r)))$$

$f_\lambda(r|y)$ is the probability that an atom, randomly chosen from the data, was generated with refutation r. In the complete data case, $X(Y(r)) = r$, so that $f_\lambda(r \mid y)$ does not depend on λ and $f_\lambda(r|y) = \tilde{f}(r)$ which is known. With ambiguous data, we have to divide up (using f_λ) the empirical probability for an atom $y_k = Y(r)$ between the various rs that might have generated it. Let us extend our notation for expectations so that $f_\lambda[\nu_i \mid y]$ denotes the expected value of ν_i according to the distribution $f_\lambda(r \mid y)$. $f_\lambda[\nu_i \mid y]$ is hence the expected number of times the parameterised clause C_i was used to generate an atom randomly chosen from the data. Riezler shows that we can iteratively improve parameter estimates by adapting Improved Iterative Scaling to cope with incomplete data. We just replace $\tilde{f}(r)$ in IIS with $f_\lambda(r \mid y)$ giving us the **Iterative Maximization (IM)** algorithm.

Definition 9 (Iterative Maximization (IM)). Let $\lambda^{(0)}$ be our initial set of parameter estimates. Set $h = 0$

1. For each parameterised clause C_i, let $\gamma_i^{(h)} \in [-\infty, \infty]$ be the unique solution of

$$f_{\lambda^{(h)}}\left[\nu_i e^{\gamma_i^{(h)} \nu_\#}\right] = f_{\lambda^{(h)}}[\nu_i \mid y]$$

2. Set $\lambda^{(h+1)} = \gamma^{(h)} + \lambda^{(h)}$.
3. Set $h \leftarrow h + 1$ and go to 1 unless $f_{\lambda^{(h)}}$ has converged.

If $\nu_\#(r) = K$ for all r, we have a closed form solution:

$$\gamma_i^{(h)} = \frac{1}{K} \log \frac{f_{\lambda^{(h)}}[\nu_i \mid y]}{f_{\lambda^{(h)}}[\nu_i]}$$

Let us apply IM to learning the parameters of $\mathcal{S}_2$ in figure 10 using the data set described in Table 5. $\mathcal{S}_2$ is just $\mathcal{S}_0$ with unknown parameters. Note that both $s(a)$ and $s(b)$ are ambiguous—both could have been generated in two ways.

We begin by computing $f_\lambda(r \mid y)$, $f_\lambda(r)$, $f_\lambda[\nu_i \mid y]$ and $f_\lambda[\nu_i]$ as functions of λ; the results are in Tables 6 and 7.

Table 5. Empirical distribution of atoms (incomplete data).

Atom	$s(a)$	$s(b)$
Count	7	5
$\tilde{p} =$	7/12	5/12

Table 6. Probability distributions for iterative maximization, where $Z_\lambda = l_1(l_3^2 + l_4^2) + l_2(l_5 + l_6)$.

r	$f_\lambda(r \mid y)$	$f_\lambda(r)$
(C_1, C_3, C_3)	$\frac{7}{12}\frac{l_1 l_3^2}{l_1 l_3^2 + l_2 l_5}$	$Z_\lambda^{-1} l_1 l_3^2$
(C_2, C_5)	$\frac{7}{12}\frac{l_2 l_5}{l_1 l_3^2 + l_2 l_5}$	$Z_\lambda^{-1} l_2 l_5$
(C_1, C_4, C_4)	$\frac{5}{12}\frac{l_1 l_4^2}{l_1 l_4^2 + l_2 l_6}$	$Z_\lambda^{-1} l_1 l_4^2$
(C_2, C_6)	$\frac{5}{12}\frac{l_2 l_6}{l_1 l_4^2 + l_2 l_6}$	$Z_\lambda^{-1} l_2 l_6$

Table 7. Expectations for iterative maximization.

Clause	$f_\lambda[\nu_i \mid y]$	$f_\lambda[\nu_i]$
C_1	$\frac{7}{12}\frac{l_1 l_3^2}{l_1 l_3^2 + l_2 l_5} + \frac{5}{12}\frac{l_1 l_4^2}{l_1 l_4^2 + l_2 l_6}$	$Z_\lambda^{-1}(l_1 l_3^2 + l_1 l_4^2)$
C_2	$\frac{7}{12}\frac{l_2 l_5}{l_1 l_3^2 + l_2 l_5} + \frac{5}{12}\frac{l_2 l_6}{l_1 l_4^2 + l_2 l_6}$	$Z_\lambda^{-1}(l_2 l_5 + l_2 l_6)$
C_3	$2\frac{7}{12}\frac{l_1 l_3^2}{l_1 l_3^2 + l_2 l_5}$	$2Z_\lambda^{-1} l_1 l_3^2$
C_4	$2\frac{5}{12}\frac{l_1 l_4^2}{l_1 l_4^2 + l_2 l_6}$	$2Z_\lambda^{-1} l_1 l_4^2$
C_5	$\frac{7}{12}\frac{l_2 l_5}{l_1 l_3^2 + l_2 l_5}$	$Z_\lambda^{-1} l_2 l_5$
C_6	$\frac{5}{12}\frac{l_2 l_6}{l_1 l_4^2 + l_2 l_6}$	$Z_\lambda^{-1} l_2 l_6$

Fortunately, all refutations involving C_1 use 3 parameterised clauses, similarly for refutations involving C_3 and C_4. For C_2, C_5 and C_6 the number of parameterised clauses used in refutations is always 2. This allows us to use the closed-form for $\gamma_i^{(h)}$ in Eq. (7).

$$\begin{aligned}\gamma_i^{(h)} &= \frac{1}{3}\log\left(\frac{f_{\lambda^{(h)}}[\nu_i \mid y]}{f_{\lambda^{(h)}}[\nu_i]}\right), \quad i = 1, 3, 4\\ \gamma_i^{(h)} &= \frac{1}{2}\log\left(\frac{f_{\lambda^{(h)}}[\nu_i \mid y]}{f_{\lambda^{(h)}}[\nu_i]}\right), \quad i = 2, 5, 6\end{aligned} \tag{7}$$

```
l1:s(X) :- p(X), p(X).        l3:p(a).      l5:q(a).
l2:s(X) :- q(X).              l4:p(b).      l6:q(b).
```

Figure 10. $\mathcal{S}_2$: A simple SLP.

Table 8. Iterative maximization from two different starting points.

	C_1	C_2	C_3	C_4	C_5	C_6	$\log L_\lambda(y)$
$(\hat{l}_i)^0$	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	−8.3178
			...				...
$(\hat{l}_i)^9$	0.5005	0.4996	0.5470	0.4471	0.5722	0.4227	−8.1503
$(\hat{l}_i)^0$	5.0000	0.5000	0.4350	0.5000	5.0000	25.0000	−12.3703
			...				...
$(\hat{l}_i)^{15}$	6.5340	0.4524	0.7655	0.3215	11.6720	12.8896	−8.1503

We can use the results contained in Table 7 and Eq. (7) to do Iterative Maximization (IM). The results of doing IM, with two different starting parameter sets are shown in Table 8. The final column shows the log-likelihood of the data which is always increasing. We converge to two different parameter sets which both define the same distribution, showing that, in general, the parameters of SLPs, like other loglinear models, are not *identifiable*: different parameters sets can define the same distribution.

4.4. Failure-adjusted maximisation

In previous sections we have applied general purpose algorithms to parameter estimation in SLPs. In this section we show how to use the EM algorithm to do MLE parameter estimation for *pure, normalised* SLPs. (Familiarity with the EM algorithm is assumed.) We will call this application of the EM algorithm *failure-adjusted maximisation* (*FAM*) because the algorithm can be seen as an adjustment of the application of EM to context-free models such as HMMs and SCFGs, where the adjustment is explicitly expressed in terms of failure derivations.

FAM rests on the observation that a data set of *atoms* can be viewed as an incomplete data set derived from a complete data set of *derivations* which has been *truncated* to form a set of *refutations* and then *grouped* to give the observed set of atoms. The basic idea is to suppose that the observed n atoms are generated as follows. An unknown number of derivations are sampled according to ψ_λ, then all the derivations that end in failure are thrown away (the data is truncated) leaving us with a set of refutations. Next all information about the refutations is thrown away, except the atoms that each of them yield. In other words, all refutations yielding the same atom are *grouped* together.

The application of the EM algorithm to grouped truncated data is given in Dempster et al. (1997) and our presentation of the FAM algorithm specialises that given in Dempster et al. (1997) to SLPs. The observed data is $y = (y_1, y_2, \ldots, y_N)$ a set of atoms, which we will reformulate into a more convenient form. Let us assume that N_0 of the atoms are *unambiguous* in the sense that they only have one proof. Let us denote this set of proofs by $\mathbf{x}_0 = x_{01}, x_{02}, \ldots, x_{0N_0}$. As for the other (ambiguous) atoms in the data, let there be $t-1$ different ambiguous atoms and let N_k $(k = 1, \ldots, t-1)$ denote the number of times atom y_k appears in y. Writing $\mathbf{N} = (N_0, N_1, \ldots, N_{t-1})$, we have that the observed data is $y = (\mathbf{N}, \mathbf{x}_0)$.

For each $k = 1, \ldots, t-1$, let $\mathbf{x}_k = (x_{k1}, \ldots, x_{kN_k})$ denote the unknown proofs corresponding to the N_k observations of atom y_k. We could consider $(\mathbf{N}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ the hidden complete data sampled from f_λ and proceed to (attempt to) apply EM. The problem is (as demonstrated in Section 4.2) that MLE for f_λ, whilst easier than for p_λ, is still hard. As Dempster et al. (1997) observe

> The drawback of [considering $(\mathbf{N}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ as the complete data] in many standard examples is that maximum likelihood estimates from a truncated family are not expressible in closed form, so that the M-step of the EM-algorithm itself requires an iterative procedure.
>
> We propose therefore a further extension of the complete data [...] to include truncated sample points. We denote by m the number of truncated sample points.

Following Dempster et al. let us suppose that there were m failure derivations which were truncated. Denote these unknown failure derivations by $\mathbf{x}_t = (x_{t1}, \ldots, x_{tm})$. Now the complete data is $\mathbf{x} = (\mathbf{N}, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, m, \mathbf{x}_t)$. This is a set of $N + m$ independent and identically distributed (iid) derivations. The sampling distribution over such a set of derivations can be compactly defined by defining $\nu(\mathbf{x}) = (\nu_1(\mathbf{x}), \ldots \nu_n(\mathbf{x}))$ where

$$\nu_i(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{k=0}^{t} \sum_{\iota=1}^{N_k} \nu_i(x_{k\iota})$$

So $\nu_i(\mathbf{x})$ is the total number of times clause C_i was used to produce the complete data. Using ψ_λ to represent the distribution over iid sequences of derivations, as well as a the distribution over derivations, we have (as given in Dempster et al. (1997)):

$$\begin{aligned}
\psi_\lambda(\mathbf{x}) &= \frac{N!}{\prod_{k=0}^{t-1} N_k!} \binom{m+N+1}{m} \prod_{k=0}^{t} \prod_{\iota=1}^{N_k} \psi_\lambda(x_{k\iota}) \\
&= \frac{N!}{\prod_{k=0}^{t-1} N_k!} \binom{m+N+1}{m} \prod_{k=0}^{t} \prod_{\iota=1}^{N_k} e^{\lambda \cdot \nu(x_{k\iota})} \\
&= \frac{N!}{\prod_{k=0}^{t-1} N_k!} \binom{m+N+1}{m} e^{\lambda \cdot \nu(\mathbf{x})}
\end{aligned}$$

The motivation for completing the data in this way is that MLE for ψ_λ (as opposed to p_λ or f_λ) is extremely easy. In the case of normalised SLPs each derivation is a walk through the Markov chain, and we can do MLE by just counting how often each clause is used. The MLE estimate for each clause is just that clause's frequency divided by the total frequency for each clause whose head has the same predicate symbol.

By positing lots of missing data, we have made the M-step of EM very easy. We pay a price for this when we come to the E-step. The E-step, in this case, requires us to compute the expected values of the sufficient statistics of $\mathbf{x}$ according to $\psi_{\lambda^{(h)}}(\mathbf{x}|y)$ where $\lambda^{(h)}$ is the current parameter estimate. The sufficient statistics of $\mathbf{x}$ are the $\nu_i(\mathbf{x})$. So, for each i, we need $\psi_{\lambda^{(h)}}[\nu_i|y]$, the expected frequency for clause C_i given the observed data and the current parameter estimate. Any given clause C_i can 'fire' when producing either (i) an unambiguous atom or (ii) an ambiguous atom or (iii) a failure derivation. We will denote the conditional distribution over derivations given that they yield y_k by $\psi_\lambda(x \mid y_k)$ and given that they end with `fail` by $\psi_\lambda(x \mid \texttt{fail})$. The expected frequency of a clause is hence made up of three components. Following Dempster et al. (1997), we have:

$$\psi_{\lambda^{(h)}}[\nu_i \mid y] = \sum_{\iota=1}^{N_0} \nu_i(x_{0\iota}) + \sum_{k=1}^{t-1} N_k \psi_{\lambda^{(h)}}[\nu_i \mid y_k] + N\left(Z_{\lambda^{(h)}}^{-1} - 1\right)\psi_{\lambda^{(h)}}[\nu_i \mid \texttt{fail}] \quad (8)$$

Equation 8 forms the basis of FAM which is defined in Definition 10. Note that as long as our initial parameters $\lambda^{(0)}$ are such that $Z_{\lambda^{(0)}} > 0$ then $Z_{\lambda^{(h)}}$ will remain positive, since a zero value would imply that the data has zero probability and the EM algorithm of which FAM is an instance always increases the likelihood of the data.

Definition 10 (Failure-adjusted maximisation (FAM)). Let $\lambda^{(0)}$ be our initial set of parameter estimates such that $Z_{\lambda^{(0)}} > 0$. Set $h = 0$

1. For each parameterised clause C_i, compute $\psi_{\lambda^{(h)}}[\nu_i \mid y]$ using (8)
2. For each parameterised clause C_i let $S_i^{(h)}$ be the sum of the expected counts $\psi_{\lambda^{(h)}}[\nu_{i'} \mid y]$ for all the clauses $C_{i'}$ such that $C_{i'}^+$ shares the same predicate symbol as C_i^+.
3. For each parameterised clause C_i, if $S_i^{(h)} = 0$ then $l_i^{(h+1)} = l_i^{(h)}$ otherwise

$$l_i^{(h+1)} = \frac{\psi_{\lambda^{(h)}}[\nu_i \mid y]}{S_i^{(h)}}$$

4. Set $h \leftarrow h + 1$ and go to 1 unless $\lambda^{(h+1)}$ has converged.

The decomposition of $\psi_{\lambda^{(h)}}[\nu_i \mid y]$ into three parts facilitates a comparison of various parameter estimation problems of different degrees of difficulty. If there are no ambiguous atoms and zero probability of failure, then the latter two terms in (8) are zero and $\psi_{\lambda^{(h)}}[\nu_i|y]$ can be computed by simple counting. This corresponds to parameter estimation for SCFGs from annotated data. If there are ambiguous atoms but still no possibility of failure then the computation becomes harder since we have to compute the expectations given in the second term. This corresponds to parameter estimation for SCFGs from unannotated data. In the general case we have all three terms to compute. The final term can be thought of as a measure of the degree of 'non-context-freeness'. The lack of context-freeness is here identified with the possibility of failure and is reflected in the name FAM.

4.4.1. Failure-adjusted maximisation for complete data. Consider the data set in Table 2, and set our initial clause parameters as follows: $\forall i : l_i = 1/2$. Since each atom is unambiguous, the second term in Eq. (8) is zero. Also, since our only 2 failure derivations are (C_1, C_3, C_4) and (C_1, C_4, C_3) $\psi_\lambda[\nu_i|\texttt{fail}]$ does not depend on λ. Table 9 tabulates $\sum_{\iota=1}^{12} \nu_i(x_{0\iota})$ and $\psi_\lambda[\nu_i|\texttt{fail}]$, where $x_{0\iota}$ is the unique derivation which yields the ιth atom

Table 9. λ-independent expectations.

	C_1	C_2	C_3	C_4	C_5	C_6
$\sum_{\iota=1}^{12} \nu_i(x_{0\iota})$	6	6	8	4	3	3
$\psi_\lambda[\nu_i \,\vert\, \mathtt{fail}]$	1	0	1	1	0	0

in the data. Table 10 shows that FAM converges to the correct MLE estimates (given by Table 4) in 7 iterations. $\log L_\lambda(y)$ denotes the log-likelihood of the data which increases with every iteration.

4.4.2. Failure-adjusted maximisation for incomplete data. Consider estimating clause parameters from the incomplete data set in Table 5 using FAM. Both atoms are ambiguous so the first term in Eq. (8) is zero. Let $y_1 = p(a)$ and $y_2 = p(b)$, so that $N_1 = 7$ and $N_2 = 5$. First we tabulate the required expectations as functions of λ in Table 11. Note that as for complete data the $\psi_\lambda[\nu_i|\mathtt{fail}]$ happen to be independent of λ. Setting all our initial clause parameters to 0.5, and running FAM, we converge to the MLEs in 3 iterations as shown in Table 12.

4.4.3. Limitations of failure-adjusted maximisation. FAM is *not* a general purpose algorithm for log-linear distributions, it is applicable only to *normalised SLPs*. To see this consider $\mathcal{S}_{\text{UNNORM}}$, the unnormalised SLP in figure 3. Suppose we did not have the

Table 10. Failure-adjusted maximisation for complete data.

	C_1	C_2	C_3	C_4	C_5	C_6	$\log L_\lambda(y)$
l^0	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	−17.3422
			…				…
l^7	0.6602	0.3398	0.5858	0.4142	0.5000	0.5000	−16.2957

Table 11. Expectations for failure-adjusted maximisation (incomplete data).

Clause	$\psi_\lambda[\nu_i\,\vert\, y_1]$	$\psi_\lambda[\nu_i\,\vert\, y_2]$	$\psi_\lambda[\nu_i\,\vert\,\mathtt{fail}]$
C_1	$\frac{l_1 l_3^2}{l_1 l_3^2 + l_2 l_5}$	$\frac{l_1 l_4^2}{l_1 l_4^2 + l_2 l_6}$	1
C_2	$\frac{l_2 l_5}{l_1 l_3^2 + l_2 l_5}$	$\frac{l_2 l_6}{l_1 l_4^2 + l_2 l_6}$	0
C_3	$2\frac{l_1 l_3^2}{l_1 l_3^2 + l_2 l_5}$	0	1
C_4	0	$2\frac{l_1 l_4^2}{l_1 l_4^2 + l_2 l_6}$	1
C_5	$\frac{l_2 l_5}{l_1 l_3^2 + l_2 l_5}$	0	0
C_6	0	$\frac{l_2 l_6}{l_1 l_4^2 + l_2 l_6}$	0

Table 12. Failure-adjusted maximisation for incomplete data.

	C_1	C_2	C_3	C_4	C_5	C_6	$\log L_\lambda(y)$
l^0	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	−8.3178
l^1	0.5000	0.5000	0.5417	0.4583	0.5833	0.4167	−8.1503
l^2	0.5000	0.5000	0.5418	0.4582	0.5835	0.4165	−8.1503

parameters 2 and 1, and had to estimate them using MLE from a dataset consisting of 2 $p(a)$ atoms and 1 $p(b)$ atom. If l_1 is the parameter for the first clause, then we get the following likelihood for the data y:

$$L(y) = \frac{l_1 l_2}{l_1 l_2 + l_2} \times \frac{l_1 l_2}{l_1 l_2 + l_2} \times \frac{l_2}{l_1 l_2 + l_2} = \frac{l_1^2}{(l_1 + 1)^3}$$

which is maximised at $l_1 = 2$ and an arbitrary value of l_2. Running IM and working to 4 decimal places, we do get convergence to $l_1 = 2$, after 89 iterations. The value for l_2 grows with every iteration of IM reaching $\approx 4.5 \times 10^{19}$ by the 89th iteration. Because FAM is constrained to produce normalised SLPs, it can not converge to a value of $l_1 = 2$. The MLE for a normalised SLP is to have $l_1 = 1 - \epsilon$, where $\epsilon > 0$ is as small as possible, so strictly speaking there is no MLE. We see this when running FAM on this data: the value for l_1 crawls towards 1, reaching 0.9990 after 6000 iterations.

4.5. *Using FAM*

The practicality of the FAM algorithm, like other instances of the EM algorithm, depends on the ability to compute or accurately estimate the required expectations. The work in Kameya and Sato (2000) on applying EM to models described in the PRISM (programming in statistical modelling) language addresses the problem by using tabulation to increase efficiency. Tabular approaches can take advantage of the shared common structure of different refutations and avoid repeated computations. The naive approach in this paper of just enumerating refutations does not. PRISM models define distributions where every ground atom represents a binary random variable with possible values true and false. SLPs represent a different type of distribution, so it is difficult to see how the EM approach in Kameya and Sato (2000) could be directly applied to SLP parameter estimation. Nonetheless there are interesting connections and a tabular approach seems the sensible way to compute exact expectations in SLPs. Done correctly, such an approach would become the forward-backward algorithm when the SLP was equivalent to an HMM and the inside-outside algorithm when the SLP was equivalent to a SCFG.

It seems likely that even efficient exact computation of expectations is likely to be infeasible for large SLPs. In such cases, a more realistic approach will be based on sampling. We can easily sample derivations according to ψ_λ. Failure derivations so sampled contribute to estimation of the $\psi_\lambda[\nu_i \mid \texttt{fail}]$ and refutations which yield atoms y_k in the dataset contribute to estimation of the $\psi_\lambda[\nu_i \mid y_k]$.

Since FAM is an instance of the EM algorithm, 'missing' data is easily handled. Suppose we were using FAM to estimate the parameters of $\mathcal{S}_{\text{PALINDROME}}$ in figure 8 and we had the following atom in our data set; $y_k = s([X1, a, b, X2, X2, b, a, X1], [])$ which represents the observation of a string where some of the elements are unknown. We can just treat $s([X1, a, b, X2, X2, b, a, X1], [])$ like any other atom, finding its refutations and computing expectations for FAM, or sampling derivations and counting how many of these happen to be refutations of $s([X1, a, b, X2, X2, b, a, X1], [])$. More interestingly, we can do exactly the same for $s([X1, a, b, X1, X1, b, a, X1], [])$—a string where the two unknown values are somehow known to be the same.

5. Conclusions and future work

There were two central goals of this paper: to detail the fundamental statistical properties of SLPs and to find a parameter estimation algorithm for pure normalised SLPs. In addressing the first issue we looked at the properties of unnormalised SLPs and gave a precise definition of the distribution defined by an impure SLP.

The FAM algorithm is not a general algorithm for log-linear parameter estimation, but an algorithm for normalised SLPs that takes advantage of a crucial property of such SLPs—the parameters can be interpreted as probabilities. Unlike the IIS and IM algorithms, we *always* have a closed form when computing updates. Although solving the polynomial for IIS and IM may not be too arduous in many cases it is always more complex than using a closed form and in some cases will be highly computationally burdensome. IIS and IM of course have the advantage that they are applicable to all log-linear models.

Although this paper has given the mathematical justification for the FAM algorithm and showed that it works for two small problems, it is clearly necessary to apply it to more complex, 'real' problems, where we expect sampling approaches to the estimation of expectations to be important. An analysis of sampling from SLPs, using an example SLP representing a distribution over logic programs, can be found in Cussens (2000).

ILP has traditionally been biassed towards structure learning at the expense of parameters. This paper has the complementary deficiency and future work must focus on intertwining parameter estimation and structure learning. The MACCENT algorithm (Dehaspe, 1997) is currently the only ILP algorithm that does this, so it is likely to be productive to incorporate lessons from MACCENT in future work.

Acknowledgments

Many thanks to Stefan Riezler for information on Iterative Maximization and to Stephen Muggleton for discussions on SLPs. Thanks also to three anonymous referees for many useful criticisms and corrections. Peter Flach and Sašo Džeroski provided input too. This work has benefitted from interaction with Taisuke Sato, Luc De Raedt, Luc Dehaspe and Kristian Kersting. Special thanks to Gill, Jane and Robert for putting up with spoiled weekends.

References

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics, 23:4*, 597–618.

Boole, G. (1854). *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Mineola, NY: Dover.

Charniak, E. (1993). *Statistical Language Learning.* MA: Cambridge. MIT Press.

Cussens, J. (1999a). Integrating probabilistic and logical reasoning. *Electronic Transactions on Artificial Intelligence*, *3*(B), 79–103. Selected articles from the machine intelligence 16 workshop.

Cussens, J. (1999b). Loglinear models for first-order probabilistic reasoning. In K. B. Laskey & H. Prade (Eds.), *Proceedings of the fifteenth annual conference on uncertainty in artificial intelligence (UAI-99).* Stockholm. San Francisco, CA: Morgan Kaufmann.

Cussens, J. (2000). Stochastic logic programs: Sampling, inference and applications. In C. Boutilier & M. Goldszmidt (Eds.), *Proceedings of the sixteenth annual conference on uncertainty in artificial intelligence (UAI-2000).* Stanford, CA:Morgan Kaufmann.

Dehaspe, L. (1997). Maximum entropy modeling with clausal constraints. In N. Lavrač & S. Džeroski (Eds.), *Inductive logic programming: Proceedings of the 7th International Workshop (ILP-97). LNAI 1297*. Berlin: Springer.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:4*, 380–393.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society, Series B, 39:1*, 1–38.

Feller, W. (1950). *An Introduction to probability theory and its applications* (Vol. 1) 3rd ed. New York: John Wiley.

Flach, P., & Lachiche, N. (1999). 1BC: A First-Order Bayesian Classifier. In S. Džeroski & P. Flach (Eds.), *Proceedings of the ninth international workshop on inductive logic programming (ILP-99)* (Vol. 1634). *Lecture notes in artificial intelligence*. Berlin: Springer-Verlag.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In T. Dean (Ed.), *Proceedings of the sixteenth international joint Conference on artificial intelligence (IJCAI'99).* San Francisco, CA: Morgan Kaufmann.

Kameya, Y., & Sato, T. (2000). Efficient EM learning with tabulation for parameterized logic programs. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, & P. J. Stuckey (Eds.), *Proceedings of the first international conference on computational logic (CL 2000)* (Vol. 1861). *LNAI*. London. Berlin: Springer.

Kersting, K., & De Raedt, L. (2000). Bayesian Logic Programs. In J. Cussens & A. Frisch (Eds.), *Proceedings of the work-in-progress Track at the 10th international conference on inductive logic programming*.

Koller, D., & Pfeffer, A. (1997). Learning probabilities for noisy first-order rules. In M. Pollack (Ed.), *Proceedings of the fifteenth international joint conference on artificial intelligence (IJCAI-97).* Nagoya, Japan. San Francisco, CA: Morgan Kaufmann.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. In J. Mostow & C. Rich (Eds.), *Proceedings of the fifteenth national conference on artificial intelligence (AAAI-98).* Madison, Wisconsin. Menlo Park, CA: AAAI Press.

Lloyd, J. (1987). *Foundations of logic programming*. 2nd ed. Berlin: Springer.

Muggleton, S. (1996). Stochastic logic programs. In L. de Raedt (Ed.), *Advances in inductive logic programming*. Amsterdam: IOS Press.

Muggleton, S. (2000). Semantics and derivation for stochastic logic programs. In R. Dybowski (Ed.), *Proceedings of the UAI-2000 workshop on fusion of domain knowledge with data for decision support*.

Ngo, L., & Haddaway, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science, 171*, 147–171.

Pompe, U., & Kononenko, I. (1995). Naive Bayesian classifier within ILP-R. In L. De Raedt (Ed.), *Proceedings of the 5th international workshop on inductive logic programming*. Department of Computer Science, Katholieke. Leuven: Universiteit Leuven.

Pompe, U., & Kononenko, I. (1997). Probabilistic first-order classification. In N. Lavrač & S. Džeroski (Eds.), *Inductive logic programming: Proceedings of the 7th international workshop (ILP-97).* Prague. Berlin: Springer.

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence, 94:1–2*, 5–56.

Riezler, S. (1998). Probabilistic constraint logic programming. Ph.D. Thesis, Universität Tübingen. AIMS Report 5(1), 1999, IMS, Universität Stuttgart.

Received May 26, 2000
Revised August 29, 2000
Accepted December 20, 2000
Final manuscript December 20, 2000