
On Generative Parameterisations of Markov logic networks

James Cussens

JC@CS.YORK.AC.UK

York Centre for Complex Systems Analysis & Dept of Computer Science
University of York, York, UK

Keywords: Markov logic, PRISM, parameter estimation

1. Markov logic networks

Given some fixed first-order language, a Markov logic network (MLN) (Domingos et al., 2008) uses weighted formulae to define a probability distribution over Herbrand interpretations of the language. (To save space a Herbrand interpretation will be called a ‘world’ from now on. In this paper all such formulae will be clauses.) “An MLN can be viewed as a *template* for constructing Markov networks.” (Domingos et al., 2008) and many analyses of MLNs has taken Markov networks as their point of departure. Here MLNs are analysed from the more general perspective of exponential-family distributions.

An MLN is a set $L = \{(F_i, w_i)\}_{i=1,\dots,k}$ where each F_i is a first-order clause and w_i is the weight associated with that clause. For any world x , the MLN assigns it the following probability:

$$P(x) = Z_w^{-1} \exp\left(\sum_i w_i n_i(x)\right) \quad (1)$$

where $n_i(x)$ is the number of ground instances of clause F_i which hold in world x and Z_w is a normalising constant. As noted in (Domingos et al., 2008), from (1) it is clear that MLNs define an exponential-family distribution. $\mathbf{N}(x) = n_1(x), \dots, n_k(x)$ is the canonical statistic.

From (1) it is simple to construct the marginal distribution $P(\mathbf{N})$ over possible values of the canonical statistic. Let $\mathbf{n} = n_1, \dots, n_k \in N_0^k$, where $N_0 = \{0, 1, \dots\}$, be some fixed value of \mathbf{N} , then:

$$\begin{aligned} P(\mathbf{n}) &= \sum_{x:\mathbf{N}(x)=\mathbf{n}} Z_w^{-1} \exp\left(\sum_i w_i n_i(x)\right) \quad (2) \\ &= Z_w^{-1} \times |x : \mathbf{N}(x) = \mathbf{n}| \times \exp\left(\sum_i w_i n_i\right) \end{aligned}$$

From (2) it is clear that $P(\mathbf{n})$ is also an exponential-family distribution with $b(\mathbf{n}) = |x : \mathbf{N}(x) = \mathbf{n}|$ as the

base measure.

The distribution $P(\mathbf{n})$ is interesting from the point of view of maximum likelihood estimation (MLE) for MLNs. Let x^1, x^2, \dots, x^m be an observed sample of m worlds. If \hat{w} is the MLE then it is well-known that:

$$E_{\hat{w}}[\mathbf{N}] = \frac{1}{m} \sum_{j=1}^m \mathbf{N}(x^j) \quad (3)$$

From this it follows that MLE for MLNs can be performed via MLE for $P(\mathbf{N})$.

2. Multivariate power series distributions

Since $P(\mathbf{N})$ is defined on N_0^k it is a *power series distribution*, a “discrete multivariate exponential-type distribution” (Patil, 1966), and can be written as:

$$P(\mathbf{n}) = b(\mathbf{n})\lambda^{\mathbf{n}}/Z_\lambda \quad (4)$$

where $\lambda^{\mathbf{n}} = \lambda_1^{n_1} \dots \lambda_k^{n_k}$ and $\lambda_i = e^{w_i}$.

In general, MLE for power series distributions seems no easier than for any other sort of exponential-family distribution. However, in the case that Z_λ only depends on λ through $\lambda_\Sigma = \lambda_1 + \dots + \lambda_k$ the distribution is known as *sum-symmetric* (Barndorff-Nielsen, 1976; Joshi & Patil, 1971) and certain useful properties follow. In particular, the statistic λ_Σ is a *cut* which allows the likelihood function to be factorised: $P(\mathbf{n}|\omega) = P(\lambda_\Sigma|\omega^{(1)})P(\mathbf{n}|\omega^{(2)}, \lambda_\Sigma)$ for some parameterisation $\omega = (\omega^{(1)}, \omega^{(2)})$ (Barndorff-Nielsen, 1976).

3. Generative characterisation of MLNs

It is well known that undirected graphical models can be encoded as directed ones via the introduction of auxiliary *observed* variables (Jensen, 1996; Cox & Wermuth, 1996). This naturally suggests a rejection sampling approach for undirected models: sample from the directed model encoding and throw away those sam-

ples not agreeing with the values of the auxiliary observed variables.

The same basic approach can be applied to MLNs. Assume that there is an (integer) upper bound ν_j for each n_j . Observe that $\lambda_i^{n_i} = (1/\lambda_i)^{-n_i}$. We have:

$$\begin{aligned}
 P(\mathbf{n}) &\propto b(\mathbf{n})\lambda^{\mathbf{n}} \\
 &\propto \left(\frac{\prod_{\lambda_j > 1} (1/\lambda_j)^{\nu_j}}{\sum_{\mathbf{n}'} b(\mathbf{n}')} \right) b(\mathbf{n})\lambda^{\mathbf{n}} \\
 &= \left(\frac{\prod_{\lambda_j > 1} (1/\lambda_j)^{\nu_j}}{\sum_{\mathbf{n}'} b(\mathbf{n}')} \right) b(\mathbf{n}) \prod_{\lambda_i < 1} \lambda_i^{n_i} \prod_{\lambda_j > 1} (1/\lambda_j)^{-n_j} \\
 &= \frac{b(\mathbf{n})}{\sum_{\mathbf{n}'} b(\mathbf{n}')} \prod_{\lambda_i < 1} \lambda_i^{n_i} \prod_{\lambda_j > 1} (1/\lambda_j)^{\nu_j - n_j}
 \end{aligned}
 \tag{5}$$

The RHS of (5) leads to the following sampling scheme for both $P(x)$ and $P(\mathbf{n})$. A world is sampled uniformly (first term of (5)), and the world has canonical statistic \mathbf{n} . For each $\lambda_i < 1$ (corresponding to a negatively weighted clause) n_i Bernoulli trials with probability of ‘success’ λ_i are conducted, the sample being rejected if any trial ‘fails’ (middle term of (5)). Note that n_i corresponds to the number of true groundings of clause F_i in the sampled world. For each $\lambda_j > 1$, $\nu_j - n_j$ Bernoulli trials with probability of success $1/\lambda_j$ are conducted, the sample being rejected if any trial fails (final term of (5)). By choosing ν_j to be the total number of groundings of F_i , $\nu_j - n_j$ becomes the number of false groundings of clause F_i in the sampled world.

4. PRISM encoding of Markov logic networks

An MLN can now be encoded by writing a PRISM program which implements the sampling procedure given in Section 3. The PRISM code for the MLN in Fig 1 (taken from (Domingos et al., 2008)) is given in Fig 2. In this example, it is assumed that there are only three constants a , b and c in the language (and no other function symbols). The ‘observational predicate’ in Fig 2 is `hi/1` representing a distribution over worlds. A world is sampled as follows. Firstly, the call to `atoms/1` generates a world M uniformly choosing truth values for each atom using the switch `atom`. Secondly, a list of all groundings of all clauses is collected by the call to `gcs/1`. Lastly, the `clauses/2` goes through each ground clause C . If M satisfies C then the process simply continues. (See second clause of `clauses/2`.) If M does not satisfy C then the switch must return `f` to avoid a failure (3rd clause). This example is simpler

$$\begin{aligned}
 0.7 &: \neg fr(X, Y) \vee \neg fr(Y, Z) \vee fr(X, Z) \\
 1.5 &: \neg sm(X) \vee ca(X) \\
 1.1 &: \neg fr(X, Y) \vee sm(X) \vee \neg sm(Y) \\
 1.1 &: \neg fr(X, Y) \vee \neg sm(X) \vee sm(Y)
 \end{aligned}$$

Figure 1. Example MLN

```

values(_, [t, f]).
:- set_sw(atom, [0.5, 0.5]).
:- set_sw(c1, [0.497, 0.503]).
:- set_sw(c2, [0.223, 0.777]).
:- set_sw(c3, [0.333, 0.667]).
:- set_sw(c4, [0.333, 0.667]).

hi(M) :- atoms(M), gcs(GCs), clauses(GCs, M).

clauses([], _M).
clauses([id(Id, C) | T], M) :-
    sat(C, M), !, clauses(T, M).
clauses([id(Id, _C) | T], M) :- msw(Id, f), clauses(T, M).
    
```

Figure 2. Part of the PRISM encoding of the MLN in Fig 1 with only 3 constants a , b and c .

than the general case since all clause weights are positive.

This is a simple encoding and could be made more sophisticated by, for example, generating ground clauses ‘on the fly’ rather than pre-computing them. Nonetheless, the program remains usable. Fig 3 shows a short PRISM session using the program to sample worlds and compute their unnormalised probabilities. Note that `repeat` is used to force PRISM to keep trying to sample a world until successful.

However, very little useful work can be done with this program due to the high number of failure derivations and high probability of a failure. Parameter estimation

```

| ?- repeat, sample(hi(X)), prob(hi(X), P).

X = [tv(ca(a), f), ... , tv(sm(c), f)]
P = 0.000030517578125 ?yes

| ?- repeat, sample(hi(X)), prob(hi(X), P).

X = [tv(ca(a), f), ... , tv(sm(c), t)]
P = 0.000006805419922 ?yes
    
```

Figure 3. PRISM session using program in Fig 2.

and Viterbi (MAP) computations are both infeasible.

5. Balanced Markov logic networks

The inefficiency of the ‘generate-and-(probabilistically)-test’ style sampling procedures given in Section 3 is the root cause of the problems of using the given PRISM program. In the general case, there seems little hope of fixing this problem; a reflection of the difficulty of (exact) computation in MLNs. However, there is an interesting class of *balanced* MLNs which are more computationally convenient.

Suppose that instead of sampling truth values for ground atoms and then checking that the interpretation so produced matches the (probabilistically chosen) truth values of the ground clauses, we sampled truth values for *ground clauses* and then sampled an interpretation satisfying these clause truth values, if any.

To see what distribution is so defined, define an equivalence relation between worlds so that $x \sim x'$ iff $\forall F_i, \theta : (x \vdash F_i \theta) \Leftrightarrow (x' \vdash F_i \theta)$. For example, worlds which differ only on the truth values of atoms which appear in no grounding of any clause in an MLN will be equivalent. Evidently, if $x \sim x'$ then $n_i(x) = n_i(x')$ for all i and so $P(x) = P(x')$.

A distribution P' can be defined over equivalence classes. Let $[x]$ be the equivalence class containing world x , then:

$$P'([x]) \propto \prod_i p_i^{n_i(x)} (1 - p_i)^{(\nu_i - n_i(x))} \quad (6)$$

An equivalence class can be sampled from this distribution by setting the truth value of each ground clause $F_i \theta$ to true with probability p_i and false with probability $1 - p_i$. This produces T , a conjunction of clauses and negated clauses. If there is a world such that $x \vdash T$, then $[x]$ is returned. Otherwise, sampling restarts from scratch, repeating until an output is produced. (In the special case where $\forall T \exists x : x \vdash T$ we say the MLN is *failure-free*.) This distribution can then be extended to be over worlds by choosing a world uniformly from the equivalence class.

If $\forall x, x' : |[x]| = |[x']|$ let us say that the MLN is *balanced*. In this case, choosing a world uniformly from equivalence classes defines a distribution over worlds where $P'(x) \propto \prod_i p_i^{n_i(x)} (1 - p_i)^{(\nu_i - n_i(x))}$. By setting $p_i = 1/(1 + e^{-w_i})$ P' becomes the same distribution as that defined by the MLN $\{(F_i, w_i)\}_{i=1, \dots, k}$. If, in addition, the MLN is failure-free, we have $P'(x) = P(x) = |[x]|^{-1} \prod_i p_i^{n_i(x)} (1 - p_i)^{(\nu_i - n_i(x))}$.

$$\begin{aligned} w_1 & : p(X) \vee q(X, Y) \\ w_2 & : p(X) \vee \neg q(X, Y) \\ w_3 & : \neg p(X) \vee q(X, Y) \end{aligned}$$

Figure 4. A balanced MLN (for any choice of weights)

The clauses in an MLN define a function $f : \{0, 1\}^{|HB|} \rightarrow \{0, 1\}^{|CB|}$ mapping truth values of atoms in HB the Herbrand base to truth values of ground clauses CB . The MLN is failure-free iff f is surjective. If $\exists k \forall y \in \{0, 1\}^{|CB|} : |f^{-1}(y)| = k$ then the MLN is balanced. MLNs where all the formulae are just atoms will be both balanced and (barring pathological cases) failure-free.

A more useful example of a balanced MLN is given in Fig 4. It is not difficult to see that the function f is in this case injective so that $\forall x : |[x]| = 1$. The MLN is not failure-free, since, for example, there is no world satisfying $\neg(p(a) \vee q(a, b))$, $\neg(p(a) \vee \neg q(a, b))$, and $\neg(\neg p(a) \vee q(a, b))$.

The challenge is to see whether interesting MLNs can be reformulated (or approximated) by balanced and/or failure-free MLNs. For the balanced and failure-free case, PRISM-style parameter learning, where the underlying generative process samples truth values for (ground) clauses, is an attractive option.

References

- Barndorff-Nielsen, O. (1976). Factorization of likelihood functions for full exponential families. *Journal of the Royal Statistical Society (Series B)*, 38, 37–44.
- Cox, D. R., & Wermuth, N. (1996). *Multivariate dependencies: Models, analysis and interpretation*. London: Chapman & Hall.
- Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., & Singla, P. (2008). Markov logic. In L. De Raedt, P. Frasconi, K. Kersting and S. Muggleton (Eds.), *Probabilistic inductive logic programming*, 92–117. Springer.
- Jensen, F. (1996). *Introduction to Bayesian networks*. UCL Press.
- Joshi, S. W., & Patil, G. P. (1971). Certain structural properties of the sum-symmetric power series distributions. *Sankhyā A*, 33, 175–184.
- Patil, G. P. (1966). On multivariate generalized power series distribution and its application to the multi-

nomial and negative multinomial. *Sankhyā A*, 28,
225–238.