Unsupervised learning of probabilistic programs with latent predicate networks

Eyal Dechter, Joshua Rule, Joshua B. Tenenbaum*

Department of Brain and Cognitive Sciences, MIT

Abstract

Probabilistic programs successfully capture many aspects of the complex symbolic and statistical knowledge that human beings possess. Children are not born with this knowledge and must learn it, often in an unsupervised fashion. How can we flexibly and tractably learn probabilistic programs? We present early work on learning probabilistic programs with latent predicate networks (LPNs), a framework for unsupervised learning in a restricted space of probabilistic context-sensitive grammars. Specifically, we encode LPNs in the probabilistic logic programming language PRISM and use stochastic variational EM to infer its parameters. Using a small fragment of English sentences about number, we show that LPNs can recover much of the relevant conceptual structure necessary to complete novel sentences in this domain with high probability.

1 Introduction

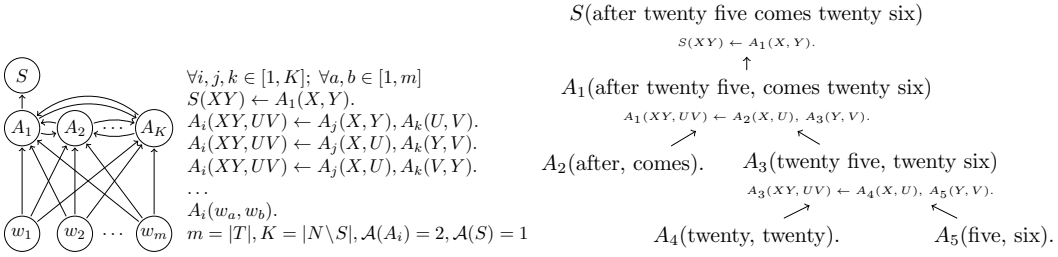
Probabilistic programs combine symbolic and statistical knowledge in a way that is natural for humans to communicate and reason about. In fact, probabilistic programs have been proposed as models of human cognition in a variety of domains [14, 7]. As a suitable representation for learning, however, the flexibility of probabilistic programs poses a challenge. At one extreme, we can approach learning as fixing the symbolic structure of a probabilistic program and fitting some of its parameters; at the other extreme, the structure is unspecified and learning is performed by searching over valid syntactic forms. The former sacrifices the symbolic flexibility that probabilistic programs have to offer, and the latter is computationally infeasible. It is natural to suppose, then, that human learning charts a middle course between these extremes, restricting the space of probabilistic programs – perhaps in a domain-specific way – to accomodate both flexibility and tractability.

Here, we propose *latent predicate networks* (LPNs) as such a middle course. LPNs maintain tractability by using Range Concatenation Grammars – a context-sensitive grammar formalism originally developed in linguistics – as the core knowledge representation. They maintain symbolic flexibility by building in latent predicates, which, through learning, come to represent underlying concepts in the learning domain.

Learning problems at the intersection of concept learning and language acquisiton are good testcases for LPNs. Such problems are important because people learn many abstract concepts primarily through language, and understanding language depends on understanding the underlying concepts. Number concepts are a good example this: children do not learn about the meaning of "seventy five" by seeing examples of seventy five things; they do not know that "seventy five" is more than "twenty five" because of their perceptual experiences of these quantities. Rather, children learn the meaning of "seventy five" (or "a billion and five") by noticing how number words are used in language, in counting sequences, in arithmetic exercises, *etc*. Other good examples of such abstract concepts are kinship and social relations (*e.g.* "my father in law's grandmother"), temporal relations ("the day after last Thanksgiving"), and spatial relations ("above and just to the left of").

The rest of this paper describes our early work on LPNs. First, we present LPNs. Then, we describe our approach to hierarchical Bayesian inference for LPNs and its implementation in PRISM, a probabilistic logic programmming language [12]. Finally, we present preliminary experimental results in the domain of number concept learning.

*{edechter, rule, jbt} @mit.edu



(a) (b)

Figure 1: (a) The architecture and rules of a schematic single-layer LPN. (b) A possible parse of the sentence "after twenty five comes twenty six" using a 5-predicate LPN.

2 Latent predicate networks

An LPN is a hierarchical Bayesian model of strings extending the Hierarchical Dirichlet PCFG model [6] to Probabilistic Range Concatenation Grammars (PRCGs).

2.1 Probabilistic Range Concatenation Grammars

Range Concatenation Grammars (RCGs) are a class of string grammars that represent all and only those languages that can be parsed in time polynomial in the length of the target string [1]. An RCG $G = (N, T, V, P, S)$ is a 5-tuple where N is a finite set of predicate symbols, T is a set of terminal symbols, V is a set of variable symbols, P is a finite set of $M \geq 0$ clauses of the form $\psi_0 \rightarrow \psi_1 \ldots \psi_M$, and $S \in N$ is the *axiom*. Each ψ_m is a term of the form $A(\alpha_1, \ldots, \alpha_{\mathcal{A}(A)})$, where $A \in N$, $\mathcal{A}(A)$ is the arity of A, and each $\alpha_i \in (T \cup V)^*$ is an argument of ψ_m. We call the left hand side term of any clause the *head* of that clause and its predicate symbol is the *head predicate*.

A string x is in the language defined by an RCG if one can *derive* $S(x)$. A derivation is a sequence of rewrite steps in which substrings of the left hand side argument string are bound to the variables of the head of some clause, thus determining the arguments in the clause body. If a clause has no body terms, then its head is derived; otherwise, its head is derived if its body clauses are derived.[1]

We extend RCGs to PRCGs by annotating each clause $C_k \in P$ with probabilities p_k such that for all predicates $A \in N$, $\sum_{k:head(C_k)=A} p_k = 1$. A PRCG defines a distribution over strings x by sampling from derivations of $S(x)$ according to the product of probabilities of clauses used in that derivation. This is a well defined distribution as long as no probability mass is placed on derivations of infinite length; here, we only consider PRCGs with derivations of finite length.

2.2 Learning Model

$$\vec{w}_{A_k} \sim Dir(\vec{\alpha}_{A_k}) \qquad x_j \underset{iid}{\sim} p_{\text{PRCG}}(S(x_j) \mid \{w_{A_k}\})$$

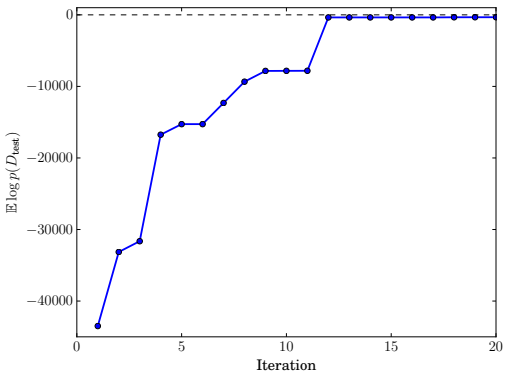
$$p(\{\vec{w}_{A_k}\} \mid \vec{x}, \{\vec{\alpha}_{A_k}\}) \propto \prod_j p_{\text{PRCG}}(x_j | \{\vec{w}_{A_k}\}) \prod_{A_k} p_{\text{DIR}}(\vec{w}_{A_k} \mid \vec{\alpha}_{A_k})$$

Given a collection of predicates, $\{A_k\}_{k=1}^K$, and a distribution over clauses, $\{\vec{w}_{A_k}\}$, the learning task is to model a set of outputs, $\{x_j\}_{j=1}^J$, as being generated according to the above distribution. The weights of clauses with head predicate A_k are drawn from a Dirichlet distribution defined by $\vec{\alpha}_{A_k}$. Each datum x_j is then drawn from the resulting PRCG. The probability of a set of weights given our data and Dirichlet prior is determined with Bayes' rule.

[1] This description of the language of an RCG technically only holds for *non-combinatory* RCGs, in which the arguments of body terms can only contain single variables. Since any *combinatory* RCG can be converted into a non-combinatory RCG and we only consider non-combinatory RCGs here, this description suffices.

Question	$K = 4$	
after twenty comes ...	twenty one	✓
after forty five comes ...	forty six	✓
after forty nine comes ...	forty ten	×
after fifty nine comes ...	fifty ten	×
after sixty one comes ...	sixty two	✓
before twenty three comes ...	twenty two	✓
before thirty comes ...	thirty eighty	×
before thirty eight comes ...	thirty seven	✓
before forty one comes ...	forty	✓
before fifty three comes ...	fifty two	✓
before eighty five comes ...	eighty four	✓

(a)



(b)

Figure 2: (a) Viterbi completions of held-out sentences for a 4-predicate 1-layer LPN. (b) expected log likelihood on a set of held-out test data in a 5-layer LPN with 5 predicates per layer.

3 Inference

LPNs can be encoded as a restricted subclass of PRISM programs in a very similar way to how PCFGs are encoded in PRISM [9]. Bayesian inference over stochastic grammars and stochastic logic programs has been an active area of research in recent decades [10, 2, 8, 3, 5]. Variational inference is a popular approach in this domain and the one we adopt here by translating LPNs into PRISM programs and using its built-in Variational Bayes Expectation-Maximization (VBEM) algorithm [13].

For large data sets, VBEM is slow and, as implemented in PRISM, too memory intensive to scale. This is because the parse forests of every dataset must either be evaluated on each iteration or they must be stored in memory between iterations. Therefore, we also implemented a stochastic VBEM algorithm adapted for PRISM programs, following [11, 4].

4 Preliminary Experiments

To evaluate LPNs as a probabilistic model of concept acquisition, we trained a single layer LPN with the architecture in Figure 1a and $K = 4$ latent predicates on a set of sentences expressing successor and predecessor relations in numbers between one and ninety-nine. The training set was the collection of sentences $X = \{\text{after } \langle n \rangle \text{ comes } \langle n+1 \rangle \mid n \in 1, \ldots, 98\} \cup \{\text{before } \langle n \rangle \text{ comes } \langle n-1 \rangle \mid n \in 2, \ldots, 99\}$, where $\langle n \rangle$ is the number words corresponding to n. The lexicon was the set of number word corresponding to 1 through 19, the decades $20, \ldots, 90$, the empty string, and the words "before", "after", and "comes". See Figure 1b for a possible LPN derivation of an example sentence. We trained the LPN using 2000 examples, holding out a subset of sentences – including those in Table 2a – for evaluation.

We evaluated the learned model by asking for Viterbi (*i.e.* maximum *a posteriori*) completions of the last words of each held out test sentence. Table 2a shows some of these completions. The grammar correctly learns much of the structure of these sentences, including the difference between sentences starting with "before" and "after" and the edge cases that relate decade words like "twenty" to non-decade words like "twenty one."

Separately, we trained an LPN with five layers of five predicates each on a larger dataset including multiple forms of each sentence (*e.g.* "twenty comes before twenty one" and "before twenty one comes twenty"). Figure 2b shows the the expected log likelihood of held out data as a function of algorithm iteration. The expected log likelihood is a lower bound on the log expected probability. Its convergence to nearly zero suggests that the trained LPN discovers the underlying structure of the sentences.

5 Conclusion

LPNs allow for tractable learning over a restricted subset of probabilistic programs. Our preliminary results suggest that LPNs may enable learning at the intersection of concept learning and language acquisition, a domain for which an integration of statistical and symbolic knowledge is particularly promising.

References

[1] Pierre Boullier. Range concatenation grammars. In *New developments in parsing technology*, pages 269–289. Springer, 2005.

[2] James Cussens. Parameter estimation in Stochastic Logic Programs. *Machine Learning*, 44(3):245–271, 2001.

[3] Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics, 2006.

[4] Matthew D. Hoffman, David M. Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[5] Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems*, pages 641–648, 2006.

[6] Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. Bayesian inference for pcfgs via markov chain monte carlo. In *Proceedings of NAACL HLT*, pages 139–146, 2007.

[7] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Concept learning as motor program induction: A large-scale empirical study. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, pages 659–664, 2012.

[8] Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP-CoNLL*, pages 688–697, 2007.

[9] John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors. *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, volume 1861 of *Lecture Notes in Computer Science*. Springer, 2000.

[10] Stephen Muggleton. Learning Stochastic Logic Programs. *Electron. Trans. Artif. Intell.*, 4(B):141–153, 2000.

[11] Rajesh Ranganath, Chong Wang, Blei David, and Eric Xing. An adaptive learning rate for stochastic variational inference. In *Proceedings of The 30th International Conference on Machine Learning*, pages 298–306, 2013.

[12] Taisuke Sato and Yoshitaka Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *J. Artif. Intell. Res. (JAIR)*, 15:391–454, 2001.

[13] Taisuke Sato, Yoshitaka Kameya, and Kenichi Kurihara. Variational Bayes via propositionalized probability computation in PRISM. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):135–158, 2008.

[14] Andreas Stuhlmüller and Noah D. Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research*, 28:80–99, 2014.