Don't Fear Optimality: Sampling for Probabilistic-Logic Sequence Models

Ingo Thon

Katholieke Universiteit Leuven {firstname.lastname}@cs.kuleuven.be

Abstract. One of the current challenges in artificial intelligence is modeling *dynamic environments* that change due to the actions or activities undertaken by people or agents. The task of inferring hidden states, e.g. the activities or intentions of people, based on observations is called filtering. Standard probabilistic models such as Dynamic Bayesian Networks are able to solve this task efficiently using approximative methods such as particle filters. However, these models do not support logical or relational representations. The key contribution of this paper is the upgrade of a *particle filter* algorithm for use with a *probabilistic logical* representation through the definition of a proposal distribution. The performance of the algorithm depends largely on how well this distribution fits the target distribution. We adopt the idea of logical compilation into Binary Decision Diagrams for sampling. This allows us to use the optimal proposal distribution which is normally prohibitively slow.

1 Introduction

One of the current challenges in artificial intelligence is modeling dynamic environments that are influenced by actions and activities undertaken by people or agents. Consider modeling the activities of a cognitively impaired person [1]. Such a model can be employed to assist people, using common patterns to generate reminders or detect potentially dangerous situations, and thus can help to improve living conditions. To realize this, the system has to infer the intention or the activities of a person from features derived from sensory information. The typical model used in such processes are Hidden Markov Models (HMM) and their generalizations like factorial HMMs, coupled HMMs or Dynamic Bayesian Networks (DBN).

Algorithms that perform efficient and exact inference in single state models like HMMs are well known. Also, for factorial HMMs and coupled HMMs efficient approximative algorithms exist that exploit structural properties [2] and for DBNs particle filters [3] present a good alternative.

However, recent research has shown that in many activity modeling domains, relational modeling is not only useful [4] [5] but also required [6]. Here, transitions between states are factored into sets of probabilistic logical conjectures that allow a dynamic number of random variables, which makes the translation into a standard Dynamic Bayesian Network impossible.

Our **contributions** are: First we show how hidden state inference problems can be formulated through Causal Probabilistic Time Logic (CPT-L). CPT-L was introduced previously, together with a learning algorithm for the fully observable case [7]. We

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

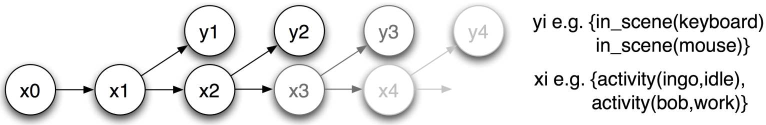

Fig. 1. Graphical representation of an HMM. States and observations are in our case herbrand interpretations.

use a logical compilation approach to implement efficient sampling from the *optimal proposal distribution*. The proposal distribution is a key component of the particle filter algorithm *Sequential Importance Resampling* (SIR). Logical compilation has gained lot of interest in the past few years for speeding up inference and learning in probabilistic logic, especially compilation into Binary Decision Diagrams (BDD) [8] [9] [7] annotated with probabilities and its variants [10] [11]. In this work we show how as second contribution how the generated BDDs can be used to sample models of the represented formula according to the underlying distribution.

Related Work: Most sequential SRL models restrict themselves to a single atom per time point [12] or one probabilistic choice, e.g. outcome of an action. We are only aware of the following three exceptions: The Simple Transition Cost Models [13] proposed by Alan Fern. These models allow the specification of costs for transitions, which can be used to represent probabilities, but they have to be equal over all transitions. In Biswas et al.[4] the authors learn a Dynamic Markov Logic Network, but translate to DBNs for inference. Even though this is not a problem in general, it requires a state space with a fixed size that is known in advance. In Zettlemoyer et. al [14], the hidden state is defined by means of one weighted FO-Formula. This approach requires mutually exclusive hypotheses, which are hard to construct and it is unclear whether they can deal with functors.

We proceed by introducing the CPT-L model. Afterwards, we specify the components of the algorithm that samples from the filtering distribution. Finally we discuss experimental result and a conclusion. Due to a lack of space, we assume basic knowledge of First Order Logic and Binary Decision Diagrams.

2 Model

The model considered is basically a HMM where states and observations are Herbrand interpretations and transition- and observation-probabilities are defined in terms of a probabilistic logic (cf. Fig 1). More formally:

Definition 1. *A* **CPT-L model** *consists of a set of rules of the form*

$$r = (h_1 : p_1) \vee \ldots \vee (h_n : p_n) \leftarrow b_1, \ldots, b_m$$

where the p_i *form a probability distribution such that* $\sum_i p_i = 1$*,* h_i *are literals,* b_i *are atoms forming the body and the* $h_i : p_i$*s are called the head.*

The interpretation of such a rule is: whenever the body is true at time-point k the rule will cause one of the head elements to be true at time-point $k + 1$.

Consider the following example of a rule that models the current activity:

$$r(P, X) = a(P, X) : 0.8 \vee a(P, drink) : 0.1 \vee a(P, work) : 0.1 \leftarrow a(P, X).$$

This rule states that person P will continue its current activity with probability 0.8 or switch to one of the activities $work$, $drink$ with probability 0.1. A successor state is generated by first determining the set of applicable rules, and then, for each applicable rule, selecting one head element. The probability of a selection is defined by the product of all selected head elements probabilities. The probability of a successor state is the sum of of the probabilities of all selections generating this state.

In this work, we additionally assume that a subset of the predicates are declared as observable whereas the others are unobservable. In the previous example the predicate $a/2$ would typically be unobservable, whereas predicates like $pose/2$, $movement/2$, $object\_in\_scene/1$ would be observable. We assume that all predicates in the head of one rule are either observable or unobservable. A rule is called un-/observable according to the observability of the predicates in the head.

In the rest of the text, x_k denotes the set of all unobservable facts and y_k the observable facts true at time point k. The probability of a hidden state sequence together with a sequence of observations follows directly from the semantics of CPT-L. Note: From the viewpoint of CPT-L the observation y_k of time-point k belongs to the successor state as both are caused by the current state. For readability purposes, we keep indexes of hidden states and observations together. With $x_{l:k}$ we denote the sequence of all states between l and k.

3 Algorithm

Our goal is to estimate the filtering distribution $p(x_k|y_{1:k})$. This distribution can be used to answer queries like $a(P1, Act), a(P2, Act), P1 \neq P2$ or for parameter learning with (Stochastic) Expectation Maximization. Exact calculation of this distribution is typically prohibitively slow due to the large structured state space [15]. Therefor we use Sampling Importance Resampling (SIR) [3]: we sample from a proposal distribution and compute importance weights that make up for the difference. In this section we first briefly discuss the mechanics of SIR. Afterwards we alter the original CPT-L algorithm [7] using a BDD to represent the distributions required by SIR.

3.1 Sampling Importance Resampling (SIR)

The filtering distribution can be approximated by a set of particles (w_k^i, x_k^i) consisting of a weight w_k^i and a state x_k^i. The weights are an approximation of the relative posterior distribution. The SIR algorithm calculates the samples recursively. A single step is described in Algorithm 1. In each step, the particles are drawn from a sampling distribution $\pi(x_k|x_{0:k-1}^i, y_{0:k})$ and each particle's weight is calculated. Typical sampling distributions are the transition prior $p(x_k|x_{k-1}^i)$, a fixed importance function $\pi(x_k|x_{0:k-1}^i, y_{0:k}) = \pi(x_k)$, or the transition posterior $p(x_k|x_{0:k-1}^i, y_{0:k})$. For reasonable distributions, not the correctness, but the sample variance, and thus the required number of particles, largely depends on the choice.

Algorithm 1 number particles N, time step k, proposal distribution π

function SAMPLE((π, p_k, p_o, P))
  for $i = 1, \ldots, N$ **do**
    $x_k^i \sim \pi(x_k | x_{0:k-1}^i, y_{1,k})$
    $\hat{w}_k^i := w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{0:k-1}^i, y_{0:k})}$
  Normalize weights $w_k^i = \hat{w}_k^i / \sum_j \hat{w}_k^j$
  Sample N particles of x_k^i according to w_k^i with weight $1/P$ if $\hat{N}_{thresh} > \overbrace{\left(\sum_i (w_k^i)^2\right)^{-1}}^{\text{effective \# particles}}$

3.2 Optimal Proposal distribution

The transition prior $p(x_k|x_{k-1})$ is often used as proposal distribution for SIR as it allows for efficient sampling. Using the transition prior, the state space is explored without any knowledge of the observations which makes the algorithm sensitive to outliers. While this nontheless works well in many cases, it is problematic in discrete, high dimensional state spaces when combined with spiked observation distributions. High dimension state spaces are common in relational domains. It can be shown that the proposal distribution $p(x_k^i|x_{k-1}^i, y_k)^\star$ together with weight update $w_k^i := w_{k-1}^i P(y_k|x_{k-1}^i)$ is optimal [3] and does not suffer from this problem.

Algorithm 2 Generate formula/BDD representing $\sum_{x_k^i} P(y_k, x_k^i | x_{k-1}^i)$

Initialize $f := \top$, $I_{max} = \emptyset$ the "maximal" successor state
Compute applicable ground rules $\mathbf{R}_k = \{r\theta | body(r\theta)$ is true in x_{k-1}^j, r unobservable$\}$
for all rules $(r = (p_1 : h_1, ..., p_n : h_n) \leftarrow b_1, ..., b_m)$ in $\mathbf{R}_k$ **do**
  $f := f \wedge (r.h_1 \vee ... \vee r.h_n)$, where $r.h$ denotes the proposition obtained by concatenating the name of the rule r with the ground literal h resulting in a new propositional variable $r.h$ (if not $h_i = nil$).
  $f := f \wedge (\neg r.h_i \vee \neg r.h_j)$ for all $i \neq j$
  $h_i \leftarrow r.h_i$; $I_{max} = I_{max} \cup h_i$
Compute applicable ground observation $\mathbf{S}_k = \{r\theta | body(r\theta)$ is true in I_{max}, r observable$\}$
for all observations $(r = (p_1 : h_1, ..., p_n : h_n) \leftarrow b_1, ..., b_m)$ in $\mathbf{S}_k$ **do**
  $f := f \wedge ((r.h_1 \vee ... \vee r.h_n) \leftrightarrow (b_1, \ldots, b_n))$, for $h_i \in I_{y_{t+1}}$
  $f := f \wedge (\neg r.h_i \vee \neg r.h_j)$ for all $i \neq j$
for all facts $l \in I_{y_{t+1}}$ **do**
  Initialize $g := false$
  for all $r \in \mathbf{S}_k$ with $p : l \in head(r)$ do $g := g \vee r.l$
  $f := f \wedge g$

BDD construction: To sample from the proposal distribution and update the weight efficiently we build a BDD that represents $P(y_k|x_{k-1}^i)$. The algorithm (shown as Algorithm 2) is a modification of the algorithm presented in previous work [7]. The algorithm builds a BDD representation of a formula which computes the joint probability of

* Here it is crucial, that the observation y_k is the observation generated by the state x_k.

all possible selections that result in a transition for which the following four conditions hold. The transition (a) starts at x^i_{t-1} (line 2) and (b) goes over to a valid successor state x_t. In x_t it (c) generates the observation y_t (line 8-14) using (d) the observable rule applicable in x_t. Each node of the generated BDD $r : h_i$ corresponds to selecting (for one rule) the head h_i or not, as dictated by the probability p_i.

BDD sampling Sampling a path according to the p_i from the root of this BDD to the terminal node with label 1 corresponds to sampling a value from $p(x^i_k|x^i_{k-1}, y_k)$. However, in most cases, sampling paths naively according to the p_i's will yield a path ending in the 0-terminal, that will then have to be rejected. There for at every node, when choosing the corresponding subtree, we base our choice not only on its probability, but also on the probability of reaching the 1-terminal through this subtree. This corresponds to conditioning the paths such that we get a valid successor state and together with the observation y_k. We call this probability upward probability because of the way it is computed. The upward probability of the root node corresponds to the probability of reaching the 1-terminal, i.e., $P(y_k|x^i_{k-1})$. The computation starts by initializing the values in the leafs with the label of the leaf. Afterwards, the probabilities are propagated upward as illustrated in Fig. 2 **. The subtree is then chosen in every node according to

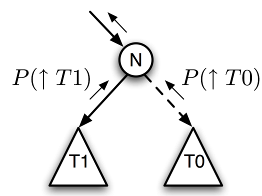

Fig. 2. Calculation of upward probability.

$$\frac{p_N \cdot P(\uparrow T1_N)}{P(\uparrow T0_N)+p_N \cdot P(\uparrow T1_N)} = \frac{p_N \cdot P(\uparrow T1_N)}{P(\uparrow N)} \qquad \text{respectively: } \frac{P(\uparrow T0_N)}{P(\uparrow N)}.$$

4 Experiments

To evaluate our algorithm, we recreated the model of Biswas et al [4], according to the parameters specified in their work. There a person is observed during writing, typing, mousing, eating, and so on. The computer has multiple cues to classify the activities. Two examples are the pose of the person or whether an apple is observed in the scene. As the observation distribution is fairly smooth and had nowhere zero mass the transition-prior is expected to perform well as proposal distribution.

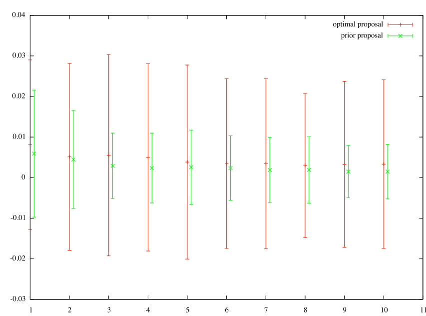

Fig. 3. Effective number of particles divided by runtime in dependence on sequence length

For the experiments we sampled 5 sequences of length 10. Afterwards we run the particle filter algorithm with the exact proposal distribution and the transition prior using 100 particles. For the optimal prior each run took less then a minute on a MacBook

** For the reader familiar with [8] [9] the use here and in [7] is a bit different. The former is the choice of a literal being true or false, whereas the latter represents whether one of the head gets chosen. Using the backward probability of [9] instead of the upward, the sampling generalizes.

Pro 2.16 Ghz. The transition prior was approximately 5 times faster. In Fig 3 the effective number of particles (cf. Alg 1) divided by the runtime in ms is plotted. The horizontal axis is the sequence length. Even though not significant the optimal performed on average better. In toy example with spiked observation distribution the transition prior typically lost all particles in a few steps.

5 Conclusions and Future work

We propose a novel way of sampling from a joint distribution in the presence of evidence by the means of BDDs. We show that the final system allows more efficient filtering than using the transition prior in relational domains. An advantage of our complete system is that the final algorithms are very intuitive as it builds on well established algorithm like SIR. We plan to extend our filtering algorithm towards more elaborate techniques like for example Rao-Blackwellized Particle Filters, and Online Stochastic EM. Finally, we will investigate the use of our technique of sampling from a BDD also for non-sequential probabilistic logics, as well as for standard DBNs

Acknowledgments: For discussion we are grateful to (in temporal order) K. Kersting, N. Landwehr, L. De Raedt, and B. Gutmann. Special thanks to K. Driessens for approximating a native speaker.

References

1. Pollack, M.E.: Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. AI Magazine **26**(2) (2005) 9–24
2. Landwehr, N.: Modeling interleaved hidden processes. In: ICML 2008
3. Doucet, Freitas, D., Gordon, eds.: Sequential Monte Carlo methods in practice. Springer
4. Biswas, R., Thrun, S., Fujimura, K.: Recognizing activities with multiple cues. Lecture Notes in Computer Science **4814** (2007) 255
5. Natarajan, S., Bui, H., Tadepalli, P., Kersting, K., Wong, W.K.: Logical hierarchical hidden markov models for modeling user activities. In: ILP-08
6. Sridhar, M., Cohn, A.G., Hogg, D.C.: Learning functional object-categories from a relational spatio-temporal representation. In: ECAI 2008. (2008)
7. Thon, I., Landwehr, N., De Raedt, L.: A simple model for sequences of relational state descriptions. In: ECML08
8. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic Prolog and its application in link discovery. In: ICJAI-07. (2007) 2462–2467
9. Ishihata, M., Kameya, Y., Sato, T., ichi Minato, S.: Propositionalizing the em algorithm by bdds. In: ILP. (2008)
10. ichi Minato, S.: Compiling bayesian networks by symbolic probability calculation based on zero-suppressed bdds. In: In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, AAAI Press (2007) 2550–2555
11. Chavira, M., Darwiche, A., Jaeger, M.: Compiling relational bayesian networks for exact inference. Int. J. Approx. Reasoning **42**(1-2) (2006) 4–20
12. Kersting, K., De Raedt, L., Gutmann, B., Karwath, A., Landwehr, N.: Relational sequence learning. In: Probabilistic Inductive Logic Programming. Volume 4911 of Lecture Notes in Computer Science. Springer (2008) 28–55
13. Fern, A.: A simple-transition model for relational sequences. In: IJCAI-05
14. Zettlemoyer, L.S., Pasula, H.M., Kaelbling, L.P.: Logical particle filtering. (2008)
15. Boyen, X., Koller, D.: Tractable inference for complex stochastic processes. In: UAI 1998