A Particle Filter for Probabilistic Dynamic Relational Domains

Extended Abstract

Davide Nitti, Tinne De Laet, McElory Hoffmann, Ingo Thon, Guy Van den Broeck, Luc De Raedt
Department of Computer Science, KU Leuven, Belgium
davide.nitti@cs.kuleuven.be

Abstract

We propose a probabilistic logic programming framework for the state estimation problem in dynamic relational domains such as Russell and Norvig's wumpus world and its probabilistic variants. The framework is based on the recently introduced notion of distributional clauses, an extension of Sato's distribution semantics with continuous distributions. The key contribution of this paper is that we introduce a particle filter for use with distributional clauses in dynamic relational domains and an unknown number of objects. The particles represent (partial) interpretations or possible worlds (with discrete and/or continuous variables) and the filter recursively updates its beliefs about the current state. Probabilistic background knowledge can be used to determine which variables must be included in the partial interpretations and magic sets or backward reasoning are employed to compute these.

1 INTRODUCTION

The problem of filtering is concerned with the estimation of the current state of an agent in a dynamic environment where the world is not directly observable but only through observations obtained from sensors. Particle filtering is an approximation that represents the posterior density function using a set of random samples (particles) with associated weights. This paper combines probabilistic logic programming techniques with particle filtering for use in filtering in dynamic relational environments involving both discrete and continuous random variables.

1.1 DISTRIBUTIONAL CLAUSES

We shall now develop a logical representation for specifying the distributions of the particle filter in a statistical relational framework; it is based on the notion of distributional clauses (Gutmann et al., 2011), an extension of Sato's distribution semantics (Sato, 1995). To illustrate our representation, we consider a probabilistic version of Russell and Norvig's wumpus world. This is a discrete world with a two-dimensional grid of cells, that can be either free, a wall or a pit. In one of the cells the horrible wumpus lives and each cell can contain gold. Each pit produces a breeze in the neighboring cells and the wumpus produces a stench in the neighboring cells. The agent has to estimate the hidden state consisting of the location of the wumpus, the state of each cell as well as its own position in the maze. The agent has four stochastic 'move' actions: up, down, left, right and noisy sensors to observe whether there is a breeze, a stench, or gold in the cell, and if there are walls or not in the neighboring cells. For (part of) the state distribution at time 0, we use the following distributional clauses (DCs):

$\mathtt{maze(0,0)}_0 \sim \mathtt{free}. \quad ; \quad \mathtt{position}_0 \sim \mathtt{(0,0)}.$

$\mathtt{maze(X,Y)}_0 \sim \mathtt{finite([0.3\!:\!wall, 0.5\!:\!free, 0.2\!:\!pit])}$:-
$\qquad \mathtt{int(X), int(Y), (X,Y)\backslash\!=(0,0)}.$

$\mathtt{gold(X,Y)}_0 \sim \mathtt{finite([0.1\!:\!true, 0.9\!:\!false])}.$

$\mathtt{wumpus}_0 \sim \mathtt{uniform(D)}$:-
$\quad \mathtt{findall((X,Y), (between(0,4,X), between(0,4,Y)), D)}.$

$\mathtt{maze(0,0)}_0 \sim \mathtt{free}$ denotes that the cell at position (0,0) is free at time $t = 0$ with probability 1, while $\mathtt{maze(X,Y)}_0$ denotes the random variable for the cell at position (X, Y) with for X and Y integers. This produces an infinite state space, we will show later how to deal and exploit this representation. $\mathtt{position}_0$ is a two-dimensional random variable that has value (0,0) with probability 1 for $t = 0$. The wumpus is at a uniformly distributed position in a finite grid.

Part of the state transition model can be described as:

$$\begin{aligned}&\mathtt{position_t} \sim \mathtt{finite([0.1:\simeq(position_{t-1})};\\ &\qquad\qquad \mathtt{0.9:(\simeq(position_{t-1})+(1,0))])} \text{ :-}\\ &\quad \mathtt{action(right)_t},\\ &\quad \mathtt{equal(maze(\simeq(position_{t-1})+(1,0)))_{t-1}, free)}.\end{aligned}$$

This DC says that if the agent performs the action right and the maze cell on the right is free, then the position of the agent moves to the right of 1 cell with probability 0.9, otherwise it remains in its original position. We also assume that the cell state, the gold and the wumpus position do not change over time, that is:

$$\begin{aligned}\mathtt{maze(X,Y)_t} &\sim (\simeq \mathtt{maze(X,Y)_{t-1}}).\\ \mathtt{wumpus_t} &\sim (\simeq \mathtt{wumpus_{t-1}}).\\ \mathtt{gold(X,Y)_t} &\sim (\simeq \mathtt{gold(X,Y)_{t-1}}).\end{aligned}$$

Part of the measurement probability for the observation gold can be represented with:

$$\begin{aligned}&\mathtt{obs\_gold_t} \sim \mathtt{finite([0.8:true, 0.2:false])} \text{ :-}\\ &\qquad \mathtt{equal(gold(\simeq position_t), true)}.\end{aligned}$$

The clause states that the probability of observing $\mathtt{obs\_gold_t} = \mathtt{true}$ is 0.8 if the cell where the agent is contains gold, 0.2 otherwise. We can define any custom discrete or continuous distribution, e.g.,

$$\mathtt{indicator_t} \sim \mathtt{gaussian(\simeq(energy_t), 0.1)}.$$

that specifies the measurement probability of an energy indicator distributed as a Gaussian with variance 0.1 around the continuous state *energy*. The other DCs in the model are omitted for lack of space.

2 A PARTICLE FILTER FOR DISTRIBUTIONAL CLAUSES

In this section, we introduce our logical particle filter for dynamic relational clauses called DCPF. We shall assume that the DCs define the required pdfs for a particle filter, that is, the prior distribution, the proposal distribution and the weighting function. Furthermore, the particles will be interpretations, that is, sets of ground facts for the predicates and random variables that hold at timepoint t. These interpretations can be *complete* or *partial*. In the former case, the particle will consist of all ground atoms that hold at time t, in the later case, this will only be a subset. The basic particle filter works with complete interpretations. To deal with infinite state spaces and to perform efficient inference, the particles should be partial interpretations and should specify the relevant variables only. For instance, in the wumpus world we want to represent the particle at time 0 with the partial interpretation

$$(\mathtt{maze(0,0), free}),\ (\mathtt{position, (0,0)}). \qquad (1)$$

This partial interpretation is essentially a conjunctive formula that represents all possible states of the agent where the cell at (0,0) is free and the agent is at (0,0). No assumptions are made about other cells. Thus multiple states are represented through a logical formula (a partial interpretation), which is somewhat similar to (Zettlemoyer et al., 2007).

To motivate our solution, reconsider the wumpus world example. The agent's belief of $\mathtt{maze(2,0)_t}$ does not change over time as long as it is not in the cell's neighborhood. Therefore, as long as the observations do not provide any information about that variable, the agent's belief of that cell will remain unchanged. Hence, we do not need to sample this variable and implicitly marginalize it. However, when the agent moves two steps to the right, or would observe a breeze on a neighboring position of $(2,0)$, the variable $\mathtt{maze(2,0)}$ would need to be included in the particles.

We need to include only those variables in the particles whose belief changes over time, which leads to gradually expanding the partial interpretations. Furthermore, this contrasts with the classical particle filter which always needs to keep track of all variables; cf. also Figure 1 later. This is an example of probabilistic background knowledge: the DC defines the maze cells' belief distribution not only for step 0 but for each step and marginalizes over each cell maze(X,Y) that has not been sampled. Indeed if a random variable maze(x,y) is not sampled it is not involved in the belief update, and then the belief is equal to the prior distribution. A further advantage is that it is no longer necessary to restrict the domains of X and Y to finite ones. For example, in the wumpus world the particles are initialized with (1). To generate such partial interpretations we use goal-directed forward reasoning (magic set transformation) or backward reasoning (SLDNF resolution) together with Monte Carlo methods.

3 EXPERIMENTS

Our experiments use different variations of the wumpus world. The results show that our DCPF produces lower errors and is faster as compared to the classical particle filter (both implemented in YAP Prolog) for the same number of particles. This is due to the fact that the DCPF reduces the number of tracked random variables and therefore reduces both the variance in the sampling process and the execution time. Finally, in more complex environments, the DCPF produces stable results compatible with the real world state. On the contrary, the classical particle filter produces unstable results for big mazes, as shown in Figure 1. It has to sample the entire real world state in the beginning, which is unlikely.

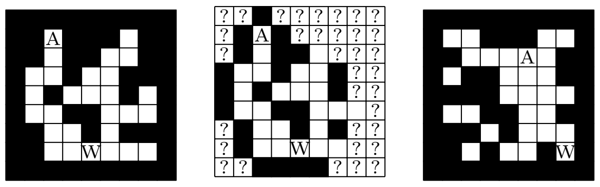

Figure 1: The symbols "A" and "W" denote the agent and wumpus. The left figure shows a randomly generated world of size 9x9 after 100 steps; the center figure shows a particle of the DCPF, where "?" represents variables that are not sampled. Note that for the DCPF the size of the maze is not fixed and hence implicitly surrounded by "?". The right figure illustrates the best sample of the classical particle filter.

4 RELATED WORK

Our particle filter is related to some recent work in the SRL and probabilistic programming tradition.

Firstly, it is related to probabilistic programming languages such as BLOG (Milch et al., 2005), Church (Goodman et al., 2008), ProbLog (Kimmig et al., 2008), and the Distributional Clauses of (Gutmann et al., 2011). While these languages are expressive enough to be used for modeling dynamic relational domains, there are—to the best of our knowledge—no special mechanisms for filtering, which means that inference would be prohibitively slow.

Secondly, there exist SRL approaches such as logical particle filtering (Zettlemoyer et al., 2007), the logical HMMs (Kersting et al., 2006), CPT-L (Thon et al., 2011), the relational particle filter of (Manfredotti et al., 2010), Lifted relational Kalman filter (Choi et al., 2011), the work of (Hajishirzi and Amir, 2008) that have been dealing with dynamics explicitly. Logical HMMs employ logical atoms as observations and states and hence, their expressivity is lower. The lifted relational Kalman performs efficient lifted exact inference for continuous dynamic domains but assumes Gaussian models. The relational particle filters CPT-L and (Manfredotti et al., 2010) employ a dynamic Probabilistic Relational Model and introduces a particle filter for this domain, however, they do not deal with partial particles. Finally, the approaches that are most similar to ours are those of (Zettlemoyer et al., 2007; Hajishirzi and Amir, 2008). They employ more expressive first order formulae for representing the particles than the partial interpretations that we employ. However, a partial interpretation is also a logical formula, and the key point that their technique and ours share is that this allows one to keep track of the relevant part of the state only. On the other hand, these techniques do not cope with continuous variables and furthermore, the inference technique is — due to the increased expressivity — much more complex. The use of magic sets and backward reasoning allows inference to be simplified a lot while realizing a similar effect.

Acknowledgements

Davide Nitti is supported by the IWT (agentschap voor Innovatie door Wetenschap en Technologie). Guy Van den Broeck is supported by the Research Foundation-Flanders (FWO-Vlaanderen). This work is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258.

References

J. Choi, A. Guzman-Rivera, and E. Amir. Lifted relational kalman filtering. In *IJCAI*, 2011.

N. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: A language for generative models. In *UAI*, 2008.

B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 2011.

H. Hajishirzi and E. Amir. Sampling first order logical particles. In *UAI*, pages 248–255. AUAI Press, 2008.

K. Kersting, L. De Raedt, and T. Raiko. Logical hidden markov models. *JAIR*, 2006.

A. Kimmig, V. Santos Costa, R. Rocha, B. Demoen, and L. De Raedt. On the efficient execution of ProbLog programs. In *Logic Programming*, volume 5366, pages 175–189. Springer, 2008.

C. E. Manfredotti, D. J. Fleet, H. J. Hamilton, and S. Zilles. Relational particle filtering. Monte Carlo Methods for Modern Applications, 2010 NIPS Workshop, Whistler, B.C., December 2010.

B. Milch, B. Marthi, S. Russell, D. Sontag, D. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. In *IJCAI*, pages 1352–1359, 2005.

T. Sato. A statistical learning method for logic programs with distribution semantics. In *Proceedings of ICLP*. MIT Press, 1995.

I. Thon, N. Landwehr, and L. De Raedt. Stochastic relational processes: Efficient inference and applications. *Machine Learning*, 82, 2011.

L. S. Zettlemoyer, H. M. Pasula, and L. P. Kaelbling. Logical particle filtering. In *In Proceedings of the Dagstuhl Seminar on Probabilistic, Logical, and Relational Learning*, 2007.