IEICE Proceeding Series

Approximated Probabilistic Inference on a Dynamic Bayesian Network Using a Multistate Neural Network

Makito Oku

Vol. 2 pp. 374-377
Publication Date: 2014/03/18
Online ISSN: 2188-5079

2013 International Symposium on Nonlinear Theory and its Applications
NOLTA2013, Santa Fe, USA, September 8-11, 2013

Approximated Probabilistic Inference on a Dynamic Bayesian Network Using a Multistate Neural Network

Makito Oku†

†Institute of Industrial Science, the University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
Email: oku@sat.t.u-tokyo.ac.jp

Abstract—
Dynamic Bayesian networks (DBNs) are flexible tools for modeling complex relationship among time-evolving random variables. An application of DBNs to computational neuroscience is to represent the internal model, which the brain uses to simulate the environment, as a DBN. The exact inference on the DBN defines the optimal behavior for both sensory and motor signal processing. However, since the exact inference requires huge computational resources, approximation methods are the key to utilize the DBN representation. Here, I propose a new heuristic algorithm for probabilistic inference on the DBN using a multistate neural network. Each random variable of the DBN is replaced by a multistate neuron, and the directional links of the DBN are translated into the nonlinear interactions among the multistate neurons. To approximate backward dependencies among variables in the DBN, the network supports a bottom-up error-reporting mechanism against top-down predictions. The proposed method is tested on a simple partially observable Markov decision process task, and exhibits better performance than ancestral sampling method.

1. Introduction

Graphical models such as Bayesian networks and Markov random fields are useful tools to visualize the relationship among random variables. They are also useful for solving various tasks because efficient algorithms such as belief propagation [1] have been proposed to solve inference problems on graphical models. Dynamic Bayesian network (DBN) model [2] is a graphical model that describes the structure of time-evolving random variables. DBN model is a generalization of classical models such as Kalman filter model and Hidden Markov model, and has much higher representational flexibility.

Recently, DBNs have been applied to planning tasks [3–5]. Planning tasks (for example, robot navigation) are to select future actions to achieve the goal or to maximize the accumulation of reward/utility values. In fact, DBN-based planning shows good performance. Verma and Rao [4] showed that DBN-based planning achieves comparable or even better performance in some benchmark tests of partially observable Markov decision process (POMDP) as compared to the state-of-the-art algorithms.

In the field of neuroscience, the notion of probabilistic inference has been widely accepted in the context of sensory signal processing [6–13]. Many experimental evidences show that the brain behaves almost optimally as Bayesian theory predicts. The theory even predicts apparently irrational behavior such as illusion [9, 12]. Although the probabilistic inference paradigm has succeeded in explaining sensory signal processing, it is still unclear whether the same viewpoint could work well for the motor signal processing in the brain.

To answer this question, Oku and Aihara [14] proposed to adopt DBN representation for the brain's internal model (see Fig. 1). In this internal model, both sensory signal processing (estimation of the hidden state X from noisy and limited observation Y) and motor signal processing (planning of future actions U given a certain goal state or its distribution) are mathematically formulated. This framework differs from reinforcement learning [15] in that we calculate the posterior probability of future actions [3, 4] rather than the optimal policy. This kind of model-based computations may work with model-free computations (for example, habitual behavior [16]) so that they play complementary roles [17].

The previous work [14] separated the sensory and motor signal processing into two different probabilistic inference

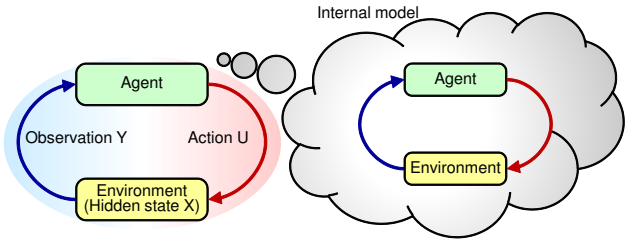

Figure 1: The agent interacting with its environment. The agent repeatedly observes the environment and takes actions based on its motivation. The observation Y is limited so that the true state of the environment X is hidden to the agent. The agent uses its internal model to simulate this loop, both for estimating the hidden state and for planning future actions.

problems. If the two problems are combined to a single problem, the exact inference on the DBN requires much more computational resources. To overcome this, approximation methods are the key to utilize the DBN representation.

Many efficient approximation methods are known (for more information, see Ref. [18]). However, if we consider biologically plausible algorithms, i.e., algorithms that can be performed by parallel processing in neural networks and/or any other biological mechanisms, few studies are found.

In this paper, I propose a new heuristic algorithm for probabilistic inference on the DBN using a multistate neural network. This algorithm is biologically plausible because (1) a multistate neuron represents a neural population with multiple attractor states [19], (2) the forward signal transmissions in the network represent top-down predictions whose nonlinearity can be explained by the existence of intermediate neurons [20], and (3) the backward signal transmissions represent bottom-up error signals whose existence has been predicted both in theory [21] and in experiments [22, 23]. In addition, the proposed algorithm uses asynchronous updating to represent parallelism in neural processing.

2. DBN representation and the exact inference

Figure 2 shows the DBN representation of the internal model [14]. The agent-environment loop is expanded in the time domain. Each node represents a random variable, and the links define dependencies among the variables. A special node labeled as R is added to incorporate the notion of reward or utility. Precisely, any reward value is normalized to $[0, 1]$ and represented as a probability measure.

The DBN consists of 4 mappings: the observation model $P(Y_k \mid X_k)$, the state transition model $P(X_{k+1} \mid X_k, U_{k+1})$, constraints on the actions $P(U_{k+1} \mid X_k)$, and the reward model $P(R \mid X_{t+1:t+T_F})$, where t denotes the current time step and T_F denotes the length of the time window for future planning. Node R takes either 0 or 1, and $P(R = 1 \mid X_{t+1:t+T_F})$ represents the normalized reward values [3–5, 24].

Figure 3 shows the inference problem on the DBN. At any time step, past observations and the previously taken actions are known to the agent, which makes the nodes' values fixed (shaded in Fig. 3). In addition, the reward node's value is fixed to 1, representing an optimistic view of the future. Our task here is to find the best sequence of future actions (marked in red) that maximizes the probability to realize the optimistic assumption.

In our previous work [14], the inference problem was separated into two parts. One is the estimation of the current hidden state X_t. The other is the future planning based on the best estimation $\hat{X}_t$. In this paper, the two inference problems are combined to a single problem. That is, the distribution of X_t rather than the best estimation $\hat{X}_t$ is used for planning.

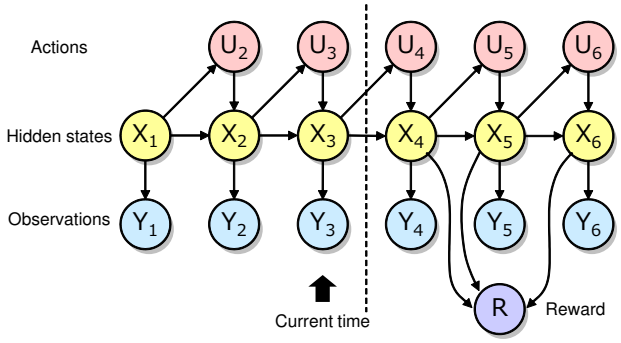

Figure 2: The DBN representation of the internal model.

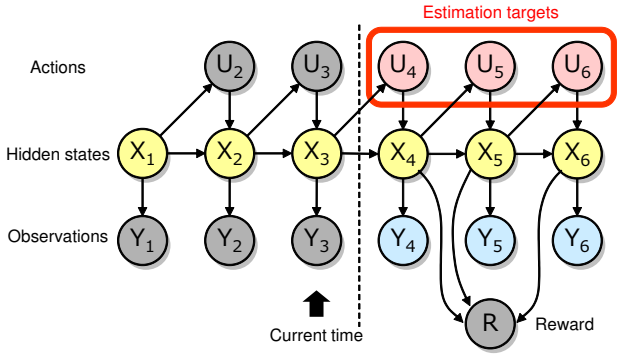

Figure 3: Inference on the DBN model.

For simplicity, we assume that every variable is discrete. Then, the full inference problem is described as follows:

$$P(U_{t+1:t+T_F} \mid Y_{1:t}, U_{2:t}, R) = \propto \sum_{X_{1:T_F}} \left[P(X_1) \left(\prod_{k=2}^{t+T_F} P(X_k \mid U_k, X_{k-1}) P(U_k \mid X_{k-1}) \right) \times \left(\prod_{k=1}^{t} P(Y_k \mid X_k) \right) P(R \mid X_{t+1:t+T_F}) \right]. \quad (1)$$

3. Approximation algorithm

To approximate the posterior distribution in (1), I propose a heuristic algorithm based on a multistate neural network. The network is constructed as follows. First, each node of the DBN is replaced by a multistate neuron. The number of each neuron's states are the same as that of the corresponding variable. Then, the neurons are directionally connected according to the DBN. These connections serve as the forward dependencies among variables of the DBN (for example, the state transition model $P(X_{k+1} \mid X_k, U_{k+1})$). In addition, we also consider backward signal transmissions through the connections, which convey error signals against the forward models' predictions.

The network is updated asynchronously. In each iteration, a neuron is selected. For simplicity, we assume that the order of selection is from upstream to downstream neurons. Let us denote the current state of the selected neuron i as $z_i \in \{1, 2, \ldots, n_i\}$. If the neuron's state is not fixed, it is updated stochastically according to the neuron's internal potential H_i^m $(m = 1, \ldots, n_i)$ as follows:

$$P(z_i' = m) \propto \exp(\beta H_i^m), \tag{2}$$

where β is the inverse temperature that globally modulates the network dynamics. This update strategy is based on the softmax function, which is a natural generalization of the logistic function commonly used for binary neurons.

If the selected neuron's state is fixed, we do not update the neuron's state, but just generate z_i' stochastically in a similar manner. Then, the two states z_i and z_i' are compared. If they are different, the neuron sends an error signal $e_i = -1$ to all of its parent neurons $\mathrm{Pa}(i)$. Otherwise, no error is reported, i.e., $e_i = 0$. The error signal makes the parent nodes avoid their current states in the subsequent iterations. Although the error signal is not very informative immediately due to the stochasticity of z_i', its time-averaged value becomes a rough approximation of the log-likelihood for parent nodes $\ln P(Z_i \mid Z_{\mathrm{Pa}(i)})$.

The internal potential is the sum of the top-down prediction from the parent neurons $F_i^m(Z_{\mathrm{Pa}(i)})$ and the time-averaged error signals from the child neurons h'^m_i as follows:

$$H_i^m = F_i^m(Z_{\mathrm{Pa}(i)}) + h'^m_i, \tag{3}$$

$$F_i^m(Z_{\mathrm{Pa}(i)}) = \ln\left(P(z_i = m \mid Z_{\mathrm{Pa}(i)}) + \epsilon\right), \tag{4}$$

$$h'^m_i = \begin{cases} \gamma h_i^m + (1-\gamma) \sum_{j \in \mathrm{Ch}(i)} e_j & (z_i = m), \\ h_i^m & (\text{otherwise}), \end{cases} \tag{5}$$

where γ denotes a decay factor of the accumulation of error signals and $\mathrm{Ch}(i)$ denotes the indexes of the child neurons of neuron i. A small positive number ϵ is introduced to avoid $\ln 0 = -\infty$. h_i^m is the current value of average error signals, and h'^m_i denotes its new value.

Let *a cycle* denote a set of iterations in which all the neurons are selected once. In the end of each cycle, we take the entire network state as a sample if no error is reported at any neuron. This means that we collect realizations of variables that do not conflict with either past observations or past actions and suffice the given reward criteria.

4. Simulation results

We tested the algorithm on a simple POMDP task as shown in Fig. 4. This task is a simplification of benchmark problems presented in Ref. [25]. Note that the robot's state depends on both its location and direction. Therefore, the total number of states is 48. If we set the time window for future planning T_F to 10 and that for retrospective computation T_B to 5, the total number of possible combinations of variables are $48^{15} \cdot 8^{15} \cdot 3^{14} \cdot 2 \simeq 5.6 \times 10^{45}$. Obviously, any efficient algorithm is needed to solve the inference problem.

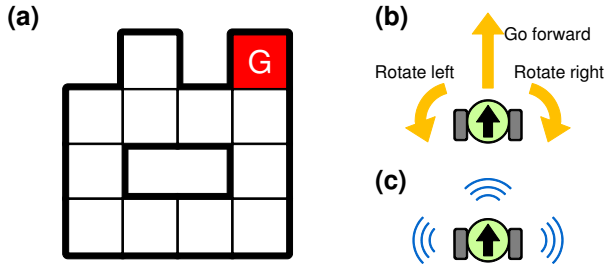

Figure 4: A simple POMDP task. (a) The maze in which the robot moves around. Thin lines define the grid which the robot can pass through, and thick lines are walls. "G" denotes the goal position. (b) The robot takes three types of actions. "Go forward" moves the robot to the next grid in front of it. "Rotate left" and "rotate right" changes the robot's direction 90 degrees. (c) The robot uses sensors to detect walls next to the current grid in three directions.

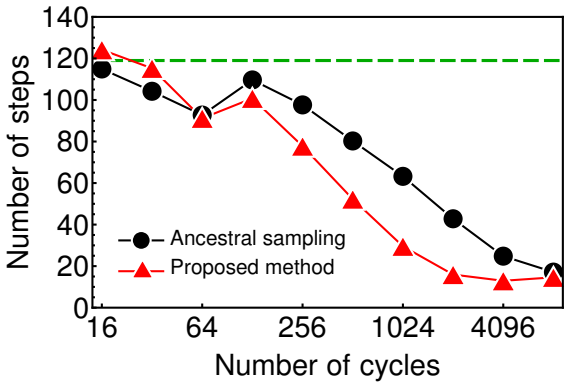

Figure 5: Comparison between the proposed method and ancestral sampling. The average number of steps over 100 trials are shown. The dashed line indicates the average number of steps in the case of random action selections.

The simulation settings were as follows. The start positions and directions of the robot were randomly chosen in each trial. Each trial was terminated when the robot reached the goal or the number of steps exceeded 200. The normalized reward value was 1 if the planned path could visit the goal. Otherwise, it was 0, i.e., such a planned path was just discarded. If no sample was obtained within the specified number of cycles, the next action was randomly chosen. Parameters were: $\beta = 2$, $\gamma = 0.95$, $\epsilon = 0.0001$, $T_\mathrm{F} = 10$, and $T_\mathrm{B} = 5$.

Figure 5 shows the comparison between the proposed algorithm and ancestral sampling. Ancestral sampling is a classical Monte Carlo sampling method that only uses the forward models and does not utilize the backward depen-

dencies among variables. As Fig. 5 shows, both methods were no better than random actions for small number of cycles. However, the average number of steps decreased as the cycle number increased for both methods, and the proposed method exhibited much better performance than ancestral sampling for hundreds of to thousands of cycles. The two curves eventually converged to a small number of steps.

The advantage of the proposed method over ancestral sampling can be explained as follows. The proposed method uses bottom-up error signals so that parent nodes prefer error-free states to error-reported ones. Therefore, the proposed method is more efficient than ancestral sampling to generate samples consistent with the fixed variables. This yields more precise approximation of the posterior probability of future actions, which may lead to smaller number of steps.

5. Conclusions

In this paper, I have proposed a new heuristic algorithm for probabilistic inference on DBN model based on a multistate neural network. The algorithm was developed in consideration of biological plausibility as well as efficiency. The results of numerical simulations revealed that the proposed algorithm works well for a simple POMDP task.

One of the major challenges is to improve the error-reporting mechanism. Its approximation quality of the parent nodes' log-likelihood functions is not high. Another problem is that the proposed method is sensitive to the update order. If we update neurons in a random order, the performance of the algorithm is deteriorated, which should be solved in future works. In addition, we took samples if no neuron reports an error. However, the real brain may not utilize such global information. Therefore, any alternative way that is locally computable is worth considering.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 24800013.

References

[1] J. Pearl, *Artif. Intell.*, 29(3):241–288, 1986.

[2] K. P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[3] H. Attias, in *Proc. AISTATS'03*, 2003.

[4] D. Verma, R. P. N. Rao, in *Proc. IROS'06*, pp. 2382–2387, IEEE, 2006.

[5] M. Botvinick, J. An, *Adv. Neural Info. Proc. Syst.*, 21:169–176, 2009.

[6] K. P. Körding, D. M. Wolpert, *Nature*, 427(6971):244–247, 2004.

[7] D. C. Knill, A. Pouget, *Trends Neurosci.*, 27(12):712–719, 2004.

[8] M. Miyazaki *et al*., *Nat. Neurosci.*, 9(7):875–877, 2006.

[9] Y. Sato, T. Toyoizumi, K. Aihara, *Neural Comput.*, 19(12):3335–3355, 2007.

[10] K. Doya *et al*., eds., *Bayesian brain: Probabilistic approaches to neural coding*. The MIT Press, 2007.

[11] Y. Sato, K. Aihara, *PLoS One*, 6(4):e19377, 2011.

[12] K.-I. Sawai, Y. Sato, K. Aihara, *Front. Psychol.*, 3:524, 2012.

[13] P. Berkes *et al*., *Science*, 331(6013):83–87, 2011.

[14] M. Oku, K. Aihara, *SEISAN KENKYU*, 65(3), 2013. (in Japanese).

[15] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] B. W. Balleine, A. Dickinson, *Neuropharmacology*, 37(4-5):407–419, 1998.

[17] N. D. Daw, Y. Niv, P. Dayan, *Nat. Neurosci.*, 8(12):1704–1711, 2005.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] M. Oku, K. Aihara, in *Proc. ICCN'11*, pp. 213–219, 2011.

[20] M. Oku, K. Aihara, *Phys. Lett. A*, 374(48):4859–4863, 2010.

[21] R. P. N. Rao, D. H. Ballard, *Nat. Neurosci.*, 2(1):79–87, 1999.

[22] S. O. Murray *et al*., *Proc. Natl. Acad. Sci. U.S.A.*, 99(23):15164–15169, 2002.

[23] L. H. Arnal, V. Wyart, A.-L. Giraud, *Nat. Neurosci.*, 14(6):797–801, 2011.

[24] G. F. Cooper, in *Proc. UAI'88*, pp. 55–63, 1988.

[25] M. L. Littman, A. R. Cassandra, L. P. Kaelbling, in *Proc. ICML'95*, pp. 362–370, 1995.