Predictive Knowledge in Reinforcement Learning

A brief survey for Planning Algorithms on Predictive State Representation

Di Wu, 260562997

1 Introduction

"AlphaGo is missing one key thing: the ability to learn how the world works ął such as an understanding of the laws of physics, and the consequences of one′ s actions." This is the comments given by Professor Rich Sutton in 2016 after Alpha Go won the famous Go competition. Professor Sutton has long been paying attention to knowledge prediction and thinking that the ability to predict is an essential component of intelligence.

Predictive knowledge is an important part and has been considered for all the three important components for reinforcement learning: action, state and reward. We can use predictive state representation (PSR) [1] to get a better representation for the state. We can use bootstrap with future prediction information like multi-step TD. Options [2] extended the concepts of actions with considering temporal abstract information. We can also modify the reward to incorporate potential future opportunity loss. We are trying to use predictive knowledge to help us to learn better and to learn fast.

Predictive state representation is proposed in 2004 [1]. Compared with POMDPs, PSRs model the system entirely on the observed variables (observations). PSRs do not need to model the system latent state variables. There are some previous work [3, 4, 5, 6, 7, 8] discussed planning algorithms for PSRs system. In this report, I focus on predictive state representation and try to give a brief survey for the planning algorithms on PSR models.

2 Predictive State Representation

In this section, I will first introduce the PSR with fully observations [1, 9] and then discuss the PSR with mixed observability [10].

2.1 Predictive State Representation for Partial Observable System

PSR is introduced is introduced [1, 9] as a means to model the partial observable system without to model the latent states. There are two approaches to model the partial observable system: generative-model approach, typically known as partially observable Markov Decision Processes (POMDPs) and history-based approach, typically known as k-th order Markov methods. Of these two approaches, POMDP is more general since the model's internal state can give it temporally unlimited memory while the history based approach can only remember as far as its history extends.

However, most POMDPs will assume a perfect dynamic model and attempt only to estimate state. The predictive state representation approach will update the state representation recursively. It looks into the future and represents what will happen. We consider a dynamic system with a discrete action set $\mathcal{A}$ and generate

observations from a discrete set $\mathcal{O}$. History h is denoted as the initial stream of experience. The probability of a test t conditioned on history h is defined in equation 1.

$$P(t|h) = \frac{P(th)}{P(h)} \tag{1}$$

Given a set of q tests $Q = t_i$, we denoted the prediction vector, $p(h) = [P(t_1|h), P(t_2|h), ..P(t_q|h)]$ as a predictive state representation (PSR) if and only if equation 2 holds for any test t and history h with $f_t : [0,1]^q \rightarrow [0,1]$.

$$P(t|h) = f_t(p(h)) = f_t(p(h)) \tag{2}$$

For linear PSR, there is a projection vector m_t, and for every test t, we will have equation 3. Let $p_i(h)$ denote the ith component of the prediction vector we will have equation 4. Compared with POMODPs, in PSRs are modeled completely on the observable variables.

$$P(t|h) = f_t(p(h)) = p(h)m_t^T \tag{3}$$

$$p_i(hao) = P(t_i|hao) = \frac{P(ot_i|ha)}{P(o|ha)} = \frac{f_{aot_i}(p(h))}{f_{ao}(p(h))} = \frac{p(h)m_{aot_i}^T}{p(h)m_{ao}^T} \tag{4}$$

2.2 Predictive State Representation for Mixed Observable System

The mixed observability systems mean that there are some system components are fully observable while other components are partially observable. This means some system observations are noiseless and perfect while others are imperfect. The PSR proposed in is [1] entirely based on observable quantities. The predictive state representation for mixed observability system (MO-PSR) is proposed in [10].

The MO-PSR models a controlled, discrete-time, finite dynamical system which generates observations from a set $\mathcal{O}$ in response to action set $\mathcal{A}$. We have perfect observations o^x, and imperfect observations o^y. We have: $o^x \in \mathcal{O}^x$, $o^y \in \mathcal{O}^y$, and $\mathcal{O} = \mathcal{O}^x \times \mathcal{O}^y$.

In MO-PSR, the set of all histories $\mathcal{H}$ is partitioned into sets $\mathcal{H}_i$, $i = 1, ..., |\mathcal{O}^x|$. $h \in \mathcal{H}_i$ is of the form $a_1[o_1^x, o_1^y]a_2[o_2^x, o_2^y]....a_t[o_t^x, o_t^y]$ with $o_t^x = i$. A set of matrices are then estimated $P_{T,\mathcal{H}_i}$. The MO-PSR learning algorithms are briefly mentioned in following three steps.

- Sample action-observation trajectories from the mixed observability system and compute vectors and matrices for: $P_{\mathcal{H}_i}$, $P_{\mathcal{T},\mathcal{H}_i}$, $P_{\mathcal{T},a\mathbf{o},\mathcal{H}_\mathbf{i}}$, $\mathbf{p}$.
- Use singular vector decomposition (SVD) on empirically estimated $\hat{P}_{\mathcal{T},\mathcal{H}_i}$, and then we can obtain its left sigular vectors.
- Compute model parameters from the above empirical estimates. We can compute $B_{a\mathbf{o}}^i$, b_o^i, b_∞^i

With the learned parameters, we can calculate the prediction for a sequence $a_1\mathbf{o_1}, a_2\mathbf{o_2}, ..., a_t\mathbf{o_t}$. This is shown in equation 5.

$$p(a_1\mathbf{o_1}, a_2\mathbf{o_2}, ..., a_t\mathbf{o_t}) = \sum_{i_0=1}^{|\mathcal{O}^x|} \mathbf{p}_{i_0} \times ((b_\infty^{it})^\mathcal{T} B_{a_t\mathbf{o_t}}^{i_{t-1}} ... B_{a_2\mathbf{o_2}}^{i_1} B_{a_1\mathbf{o_1}}^{i_0} b_0^{i_0}) \tag{5}$$

3 Planning Algorithms for Predictive State Representation

Planning is a central problem for artificial intelligence. In this section, we briefly review some previous work on planning algorithms for PSRs based dynamic models [3, 4, 5, 6, 7]. In total, there are six planning algorithms for PSRs dynamic systems.

3.1 Policy Iteration on PSRs

In [5], the authors propose a policy iteration algorithm with PSRs. Assuming that we are given a policy π $\mathcal{H} \rightarrow \mathcal{A}$, and the staring state is s_0 which is draw from probability I. Ψ_t is the set of possible tests. Then the value for policy π with a given state distribution I is shown in equation 6. $R(q|I,\pi)$ is the expected returen for test q with initial state draw from distribution I and policy π.

$$V^{\pi} = \sum_{q \in \Psi_t} V(q|\pi) = \sum_{q \in \Psi_t} P(q|I,\pi) R(q|I,\pi) \tag{6}$$

U is the matrix formed by concatenating the outcome vectors of all the tests, U^+ is the pseudoinverse. $M^{a,o}$ and $m^{a,o}$ are the projection matrix and projection vector. Then the expected return given a test q, I, π is shown in equation 7.

$$V(q|I,\pi) = \sum_{i=1}^{t} \frac{IU \prod_{j=1}^{i} (M^{a_j,o_j})^T U^+}{IU m^T_{a_1 o_1 \ldots a_i o_i}} * R^a P(a_i|\pi, a_1 o_1, .., a_{i-1} o_{i-1}) \tag{7}$$

The action-value is then defined in equation 8 and we can choose action greedily with equation 9.

$$Q_i^{\pi}(q,a) = R^a u(q) + \sum_{a \in O} \sum_{q\prime \in \Psi_{i-1}} p(q\prime|\pi, u(qao)) \tag{8}$$

$$\pi^*(t) = arg \max_a Q_i^{\pi}(a,t) \tag{9}$$

As it is shown, this planning algorithm will require a lot of pre-computation. The results in this paper shows that this algorithms could provide good results especially for small system.

3.2 Incremental Pruning on PSRs

In [3], the value iteration algorithm for POMDP called incremental pruning (IP) [11] is extended for PSR planning. The IP algorithm can be described as a method to transform the set S_i to S_{i+1} via a series of intermediate sets. Given a set S_i, there are two intermediate sets used to calculate S_{i+1}, which are shown in equation 10 and equation 11.

$$S_o^a = purge(\mathcal{T}(\alpha, a, o) | \forall \alpha \in S_i) \tag{10}$$

$$S^a = purge(\bigoplus_o S_o^a) \tag{11}$$

$$S^{i+1} = purge(\bigcup_o S^a) \tag{12}$$

$\mathcal{T}(\alpha, a, o)$ is the $|S|$ vector given by:

$\mathcal{T}(\alpha, a, o) = \frac{1}{|\mathcal{O}|} r^a(s) + \gamma \sum_{s\prime} a(s\prime) prob(o|s\prime, a) prob(s\prime|s, a)$

For a single stage of value iteration (including IP) algorithms transforms the set S_i to the set S_{i+1}. The difference from IP on POMDP to IP on PSRs is that we need to use equation 13 to calculate the policy tree value functions for PSRs. The basis for calculating the value of policy trees from POMDP nominal-states to PSR core tests. ρ is the policy tree, and w_ρ is prediction vector, M is prediction matrix and h is the history.

$$w_\rho = n - a + \gamma \sum_{o \in \mathcal{O}} M_{a,o} w_{\rho^a} \tag{13}$$

3.3 Q Learning on PSRs

Q learning is a kind of model free reinforcement learning algorithm and can also be extended to PSRs based dynamic system [3]. In this paper, the authors use prediction vectors as the state representation. As the prediction vector is continuous, then function approximation is needed. The CMAC [12] is used as function approximators.

$$Q(p, a) = \sum_g v_{g,a}(i_g(p)) \tag{14}$$

The action-value of a prediction vector p and action a is the sum over all grids of each grid's action-value for the prediction vector as shown in equation 14. $i_g(p)$ returns the index of partition in grid g that p falls into, and $v_{g,a}(i)$ returns the partition in grid g for action a. For a given current prediction vector p, action a, reward r, and next prediction $p\prime$, we use equation 15

$$\delta = r + \gamma arg \max_a Q(p\prime, a\prime) - Q(p, a) \tag{15}$$

We then update the appropriate partition of each grid g by following equation. The actions from this interaction will be chosen by ϵ-greedy policy. Then we can get q learning for PSRs.

$$v_{g,a}(i_g(p)) = v_{g,a}(i_g(p)) + \alpha\delta \tag{16}$$

3.4 Incremental Pruning on mPSRs

In [7], the application of incremental pruning is extended for memory PSR (mPSR) [13] which can preserve the PSR property of composing state with observable values while potentially reveal the structure of dynamic system.

The memory part of mPSR can work in two fields. First is that the memory can be used to decompose the problems. Instead of pruning one large set of policy trees for all memory at once, now it can prune many smaller sets. Second is that we can use memory to construct policy trees. There are mainly three difference for extending incremental-pruning for mPSR compared with using IP for PSRs systems.

- The sets $S_o^{\mu,a}, S^{\mu,a}, S_i^\mu$ must be maintained separately for each memory μ.
- The calculation of $S_o^{\mu,a}$ from $\mathcal{T}(w_\rho, a, o)$ need only consider the $w_\rho \in S_i^o$ for the memory corresponding to observation o.
- Landmarks have to be treated separately.

3.5 Point Based Planning on PSRs

In [6], the point based value iteration (PBVI) [14] algorithm is extended to planning for PSRs. In this paper, it is mentioned that there are very good reasons for using point based methods for PSRs. First point methods work by providing a discretization of continuous space of prediction vectors based on reachable points which avoids to check the validity of linear segment of PSR. Another thing is that, the number of core tests in a linear PSR can be smaller than the number of states in a POMDP which means that the space of PSR prediction vectors can be lower dimensional than belief space of original POMDP.

As mentioned in [8], the value function of PSR can be approximated by a finite set of hyperplanes (α-vectors). The computation of Bellman equation can be performed analogously to using a set of beliefs in the case of POMDPs. The approximation of values function can be represented by a set of α-vectors $\Gamma = \{\alpha_1, ..., \alpha_k\}$, α_i corresponds to the best policy tree for at least one prediction vector $P(Q|h_i) \in D$. The back up operator will then follows equation 17. To point out that the candidate of back up will be exponential if all reachable prediction vectors are to be considered. In the simulation part, the authors show that the point based PSR planning algorithm is much more efficient than POMDP with PBVI.

$$V_{t+1}(\rho) = arg\max_{a \in A} R(\rho, a) + \gamma(\sum_{\rho'} P(\rho\prime|\rho, a)V_t(\rho\prime)) \tag{17}$$

3.6 Policy Search on MO-PSR

In [4], the authors propose a policy-search-based approach coupled with Mixed Observability based PSR to address hydroelectric power plants planning. As the power generation will be influenced by the water inflow, then it is intuitively clear that predicting future inflows could lead to better control policies. MO-PSR is used to find the expected inflow for the following week, and then this value is used ad additional feature for policy. Then we can follow policy search process and learn the policy to optimize the energy production.

To learn the future inflow with MO-PSE, we first need to get inflow trajectories from generative models. Here we consider the index of week as the perfect observation and future inflow as imperfect observation. We then split the history $\mathcal{H}$ into 52 (assuming 52 weeks for one year) subsets. Then estimate a collection of vectors and matrices from data: $\{P_{\mathcal{H}_w}\}_{w \in \mathcal{W}}, \{P_{\mathcal{T},\mathcal{H}_w}\}_{w \in \mathcal{W}}, \{P_{\mathcal{T},o,\mathcal{H}_w}\}_{w \in \mathcal{W}, o \in \mathcal{O}}$. Then SVD is applied for $\{P_{\mathcal{H}_w}\}_{w \in \mathcal{W}}$, and use corresponding low rank matrces of left singular vectors to learn B_o^w, b_o^w and $b_\star^w$. Then we can use the above to learn the probability of any sequence of future observations, given week w in functions 18.

$$P(o_1, ..., o_t) = \mathbf{b_\star^{w+t\mathcal{T}} B_{ot}^{w+t-1} ... B_{o1}^{w} b_0^w} \tag{18}$$

4 Conclusion

In this report, I first briefly discuss some applications of predictive knowledge in reinforcement leaning. Then we introduce PSRs for fully observation and mixed observation system. The major objective for this report is to present a brief survey for planning algorithms on PSRs. We briefly reviewed six planning algorithms for PSRs based dynamic systems.

References

[1] M. L. Littman, R. S. Sutton, S. Singh, et al. Predictive representations of state. *Advances in neural information processing systems*, 2:1555–1562, 2002.

[2] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[3] M. R. James, S. Singh, and M. L. Littman. Planning with predictive state representations. In *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pages 304–311. IEEE, 2004.

[4] Y. Grinberg, D. Precup, and M. Gendreau. Optimizing energy production using policy search and predictive state representations. In *Advances in Neural Information Processing Systems*, pages 2969–2977, 2014.

[5] M. T. Izadi and D. Precup. A planning algorithm for predictive state representations. In *IJCAI*, pages 1520–1521, 2003.

[6] M. T. Izadi and D. Precup. Point-based planning for predictive state representations. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 126–137. Springer, 2008.

[7] M. R. James and S. Singh. Planning in models that combine memory with predictive representations of state. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20, page 987. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[8] B. Boots, S. M. Siddiqi, and G. J. Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

[9] S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 512–519. AUAI Press, 2004.

[10] S. C. Ong, Y. Grinberg, and J. Pineau. Mixed Observability Predictive State Representations. In *AAAI*, 2013.

[11] N. L. Zhang and W. Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical report, Technical Report HKUST-CS96-31, Hong Kong University of Science and Technology, 1996.

[12] J. S. Albus. A theory of cerebellar function. *Mathematical Biosciences*, 10(1-2):25–61, 1971.

[13] M. R. James, B. Wolfe, and S. P. Singh. Combining Memory and Landmarks with Predictive State Representations. In *IJCAI*, pages 734–739, 2005.

[14] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032, 2003.