Deep Predictive State Representation With K-Clustering

By Group C

28 April 2020

1 Introduction

In the Markov decision process (MDP) framework [4], the state of the system is assumed to be completely observable by decision maker with the help of perfect observations (and hence known with certainty by the decision maker).
Relaxation of this assumption yields a Partially observable Markov decision process(POMDP) [6]. It is a generalization of MDPs in which there is incomplete information and uncertainty regarding the current state of the Markov process. But the state of a POMDP is artificial and requires a prior understanding of the world but if important aspects of the dynamics are genuinely unknown, like assumption of parameters or initial belief, then these methods are rarely effective.
Predictive State Representations (PSRs) [3] [5] is based entirely on predicting the conditional probability of sequences of future observations, conditioned on future sequences of actions and on the past history. Because there are no hidden states in the model, such a representation should be easier to learn from data.
In this paper, a new method is proposed. Deep Predictive State Representation With K-Clustering adds a carefully designed value function into the original model. Besides, Deep Neural Network (DNN) is applied to construct the history probe.

2 Preliminaries

2.1 Partial Observable Markov Decision Process

A POMDP is completely characterized by $\langle S, A, Z, P, Q, R\rangle$: the core state space, the action space, the set of observations, the transition probability matrix, the information matrix, and the reward function. We'd like to give more details in the following part.

Core sate

For simplicity in presentation, $\{s(t) : t = 0, 1, 2, ...\}$ is assumed to be a finite state discrete-time Markov chain with stationary transition probability matrix

$P = [p_{ij}], \forall i, j \in S$ where S is called core state space. Decision maker cannot directly observe this core process at time t.

Observations

To learn about the true value of state at time t, observations $\{z(t) : t = 0, 1, 2, ...\}$ are collected. The observations can be regarded as functions of the core state and here we use the information matrix $Q = [q_{ik}], \forall i \in S, k \in Z$ where Z is called the observation space.

Action

At each time t, the decision maker chooses an action $\{a(t) : t = 0, 1, 2, ...\}$ from the set of available actions from the action space A. The state transition matrix P and the information matrix Q could be functions of the action a(t) in common.

Data Process and Belief States

The data process is the collection of all past and present observations and actions taken prior to time t including the initial distribution of core states, $\pi(0)$. It can be set as $d(t) = (\pi(0), z(t), a(t-1), ..., z(1), a(0))$ where $\pi(0) = (\pi_i(0), i \in S)$ is a S-dimensional vector. For the belief states, $\pi_i(t)$ can be defined the probability of being in the core state at time t, $i \in S$, by given the data process available to the decision maker.

Updating Belief States

There are two models that differ with respect to the update function for the belief state. For model 1,

$$\pi_j(t+1) = \frac{\sum_{i \in S} q_{jk}(a) p_{ij}(a) \pi_j(t)}{\sum_{i \in S} \sum_{l \in S} q_{lk}(a) p_{il}(a) \pi_j(t)}, \tag{1}$$

POMDP for model 1, that is, assuming that the state transition occurs prior to the observation.

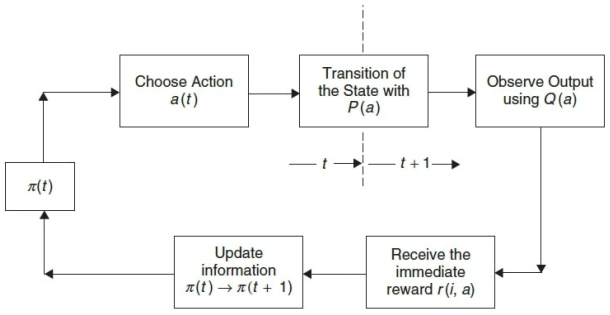


Figure 1: Model 1

For model 2, the sequence is action first, then observation, and finally the transition of the state.

$$\pi_j(t+1) = \frac{\sum_{i \in S} q_{ik}(a) p_{ij}(a) \pi_i(t)}{\sum_{i \in S} q_{ik}(a) \pi_i(t)}, \tag{2}$$

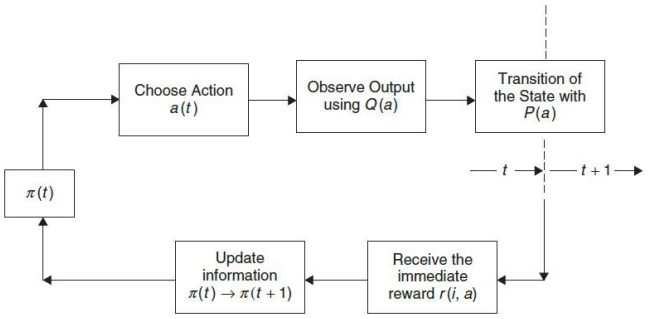


Figure 2: Model 2

Optimal Value Function

We define an optimal value function, $V_\lambda(\pi)$, to determine the policy that maximizes the total expected discounted reward.

$$V_\lambda^{T+1}(\pi) = max_{a \in A}\Big\{\sum\nolimits_{i \in S} \pi_i r^{T+1}(i,a)\Big\} \quad \text{for } \pi \in \Pi(S),$$

$$V_\lambda^{T}(\pi) = max_{a \in A}\Big\{\sum\nolimits_{i \in S} \pi_i r(i,a) + \lambda \sum\nolimits_{k \in Z} V_\lambda^{T+1}(\pi(t+1))\sigma(k|\pi,a)\Big\},$$
$$\text{for } \pi \in \Pi(S), \quad t = 1, 2, ..., T.$$

2.2 Predictive State Representation

We'd like to introduce some definitions before turning to the PSR.

Definition 1 *A history, denoted by h_τ, is an action-observation sequence received up to the time step τ: $h_\tau = (a_0 o_0 a_1 o_1 ... a_\tau o_\tau)$.*

Definition 2 *An action-observation sequence starting at time $\tau + 1$ is called a test, $t_\tau = (a_{\tau+1} o_{\tau+1} ... a_{\tau+k} o_{\tau+k})$*

Definition 3 *The prediction for test t given history h, $p(t|h)$, is defined as the conditional probability that $\omega(t)$ occurs, if $\sigma(t)$ is executed.*

$$p(t|h) = P((\omega(t)|h, \sigma(t))$$

where $\omega(t_\tau)$ is the sequence of observations of the test $(o_{\tau+1}, ..., o_{\tau+k})$ and $\sigma(t_\tau)$ is the sequence of actions of the test $(a_{\tau+1}, ..., a_{\tau+k})$.

2.2.1 System-Dynamic Matrix

System-dynamic matrix is a conceptual representation of a dynamical system. Let T be the set of all tests and H the set of all histories observed from the system. Given an ordering over H and T, the matrix D can be defined, and its entry in the matrix is the prediction of a specific test, given a history, which is shown as following figure:

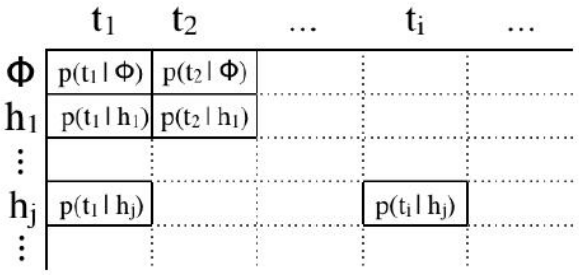


Figure 3: System-dynamic Matrix

Singh et al.(2004) have shown that, even though this matrix is infinite, if histories and tests are generated from a POMDP model, the matrix has finite rank. If we assume that D has rank n, then there exist n linearly independent columns and rows. We define that the n tests associated with the linearly independent columns are called the core tests. Analogously, corresponding n histories can be the core histories.

2.2.2 Linear PSR

Given a set of core tests Q, we define their $(1\times n)$ *predictive vector*, $P(Q|h) = [P(t_1|h), P(t_2|h), ..., P(t_n|h)]$, which can be viewed as a predictive state representation if and only if it forms a sufficient statistic for the environment, i.e., if and only if

$$P(t|h) = f_t(P(Q|h))$$

for any test t and history h, and for some *projection function* $f_t : [0,1]^n \longrightarrow [0,1]$ Focusing on Linear PSR, because of the independence of the core test, there exists a weight vector, which is defined as *projection vector*, m_t, for every test t and all history h, such that

$$P(t|h) = P(Q|h)^T m_t$$

Let $p_i(h)$ denote the i-th component of the predictive vector for some PSR. This can be update recursively, given a new action-observation pair a, o, by

$$p_i(hao) = P(t_i|hao) = \frac{P(aot_i|h)}{P(ao|h)} = \frac{f_{aot_i}(P(Q|h))}{f_{ao}(P(Q|h))} = \frac{P(Q|h)^T m_{aot_i}}{P(Q|h)^T m_{ao}}$$

Thus we can combine these updates into a single update for Q by defining the *projection matrix* M_{ao}, where the i-th column is m_{aot_i}.

$$P(Q|hao) = \frac{P(Q|h)^T m_{ao}}{P(Q|h)^T m_{ao}}$$

More general way is that the prediction for an arbitrary test, given a history, can be represented as

$$P(t|h) = \frac{P(Q|\phi)^T m_{ht}}{P(Q|\phi)^T m_h}$$

3 Model and Algorithm

3.1 Probe g

Given history h, we can already predict t using $p(t|h)$. However, in our tasks, set size of possible h is much more larger than set size of possible t. This inspired the idea of clustering all possible h so as to obtain a more concise set of history. In [2] and [1], the author first use a probe function g to map the history to a real number, and then cluster $g(h)$ instead of cluster h. The probe [2] and [1] used is either given by human or selected among a finite set. We disapprove this approach for two reasons: (1) Mapping the history to a single number may drop a lot of useful information; (2) Hand-crafted probe g may not well extract preferable features. To solve these problems, we use a DNN to learn the probe. And our probe maps the history to a same length vector, not a single number.

3.2 Value function

Denote

$$h_g = g(h)$$

We designed a value function to better learn the probe g, this value function contains three part.
The first part is the intra-class distance of all classes. This part is added to let g study the pattern of cluster and it is considered better to be small.

$$\mathrm{d}_1 = \frac{\sum_{k=0}^{k0} \sum_{i=1}^{|h_g^k|} (h_g^{k,i} - C_k)^2}{k_0 \cdot |h_g^k|} \tag{3}$$

in which C_k is the center of the k-th cluster. $h_g^{k,i}$ is $g(h)$ with $h = h^{k,i}$, and $h^{k,i}$ is the i-th history in the k-th cluster. k_0 is the cluster number.
The second part is the distance between classes, also designed to study the pattern of cluster, and we want to maximize it.

$$\mathrm{d}_2 = \frac{\sum_{i=0}^{k0} \sum_{j=i+1}^{k0} (C_i - C_{i+1})}{\frac{(1+k_0)k_0}{2}} \tag{4}$$

The third part is the most important part. It tries to minimize the distance between $g(h)$ with the same prediction t. In other word, we want those histories with the same predictions being as close as possible after being mapped by probe g.

$$\mathrm{d}_3 = \sum_{t=1}^{|T|}(h_g^1 - h_g^2)^2, \tag{5}$$

h_g^1, h_g^2 are randomly chosen with the same t.

And the final value function is

$$V = d_1 - d_2 + d_3 \tag{6}$$

3.3 Main Algorithm

Our main algorithm looks as follows:

Algorithm 1

Input: A set of data $D = (h, t)$, $\forall h \in H$, $\forall t \in T$, k_0, number of iteration n.
Output: g, history cluster A, probed predictions $p_f(T|H)$
Using K-means, initialize H_k, k_{labels}
for i in range(n) **do**
  for minibatch (H, T) in D **do**
    Calculate representation of H

$$h_g = g \cdot H$$

    Calculate new centers

$$C_k = \frac{\sum_{i=0}^{|h_g^k|} h_g^{k,i}}{|h_g^k|}$$

    Calculate intra-class distance

$$d_1 = \frac{\sum_{k=0}^{k0}\sum_{i=1}^{|h_g^k|}(h_g^{k,i} - C_k)^2}{k_0 \cdot |h_g^k|}$$

    Calculate distance between classes

$$d_2 = \frac{\sum_{i=0}^{k0}\sum_{j=i+1}^{k0}(C_i - C_{i+1})}{\frac{(1+k_0)k_0}{2}}$$

    Calculate distance of h_g^k with the same t

$$d_3 = \sum_{t=1}^{|T|}(h_g^1 - h_g^2)^2,$$

    h_g^1,h_g^2 are randomly chosen with the same t.
    Calculate the loss function

$$V = d_1 - d_2 + d_3$$

    Update g to minimize V.
  end for
end for

4 Experiment

In this section, we apply for three kinds of experiments with both deep neural network and our own algorithm, and then we show the comparative result to measure the availability of our algorithm.

4.1 Tunnel World

The first environment we will consider is a small probability domain.

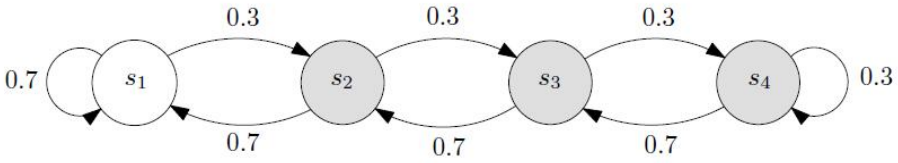


Figure 4: Tunnel World

The agent transitions from state i to state $i-1$ with probability 0.7, and to state $i+1$ with probability 0.3 (like the picture above). There are 4 states, which are not observable, and 2 deterministic observations: dark(D) and light(L). What we want to do is predicting the observations by given a number of history and test.

4.2 Half Moon World

In this domain, each of the observation, black and white, are temporally coherent. There are 12 states and the agent transitions from state i to either $i+1$ or $i-1$ with the same probability 0.5.
The agent should choose 2 actions, forward or backward. Forward means transition from state i to state $i+1$ while backward means transition from state i to state $i-1$.

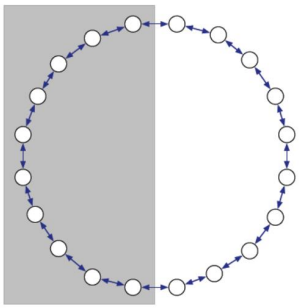


Figure 5: Half Moon World

4.3 Grid World

Then we consider a larger domain. There are 6×6 grid cells and each cell has 4 different orientations, thus there are $6 \times 6 \times 4$ states. There are 4 walls next to each boundary with different colors like the picture above. Initial position and orientation are randomly generated with equal probability to each grid and orientation.

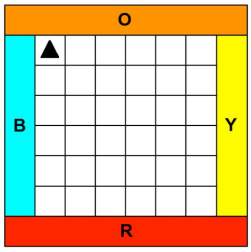
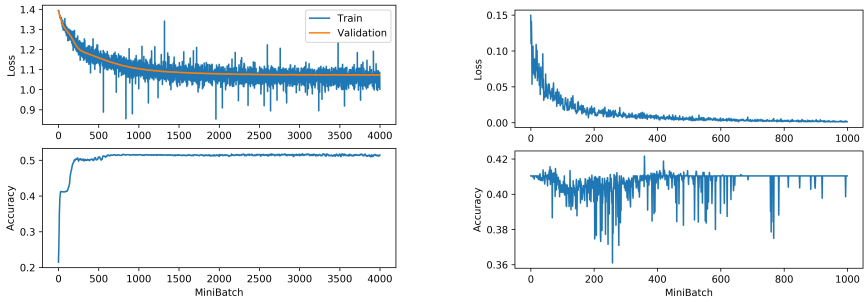


Figure 6: Grid World

There are five colours including blue, orange,yellow, red and white, which are our observations. If the agent is next to a wall and facing it ,it will observe the wall's colour, otherwise it will observe white.

4.4 Result

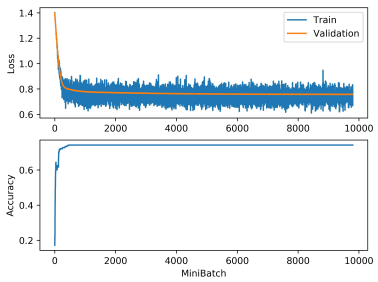


(a) DNN for Tunnel World

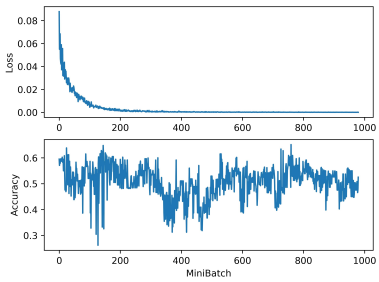
(b) DPSR-8 for Tunnel World

Figure 7: Experiment results of Tunnel World

From Fig. 7-9, it is observable that the DNN method has more stable performance and accuracy than DPSR-K in tunnel world and half moon world

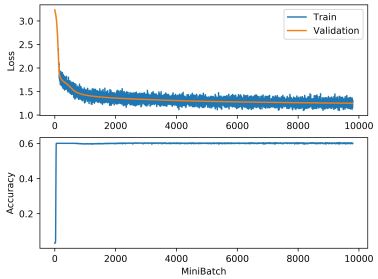


(a) DNN for Half Moon World

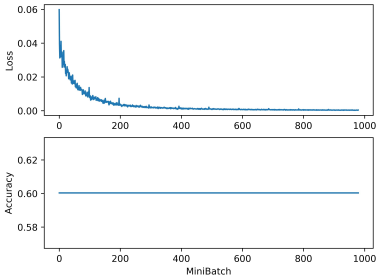


(b) DPSR-8 for Half Moon World

Figure 8: Experiment results of Half Moon World



(a) DNN for Grid World



(b) DPSR-8 for Grid World

Figure 9: Experiment results of Grid World

Method	Grid world	Half Moon World	Tunnel World
DNN	0.601	0.740	0.517
DPSR-K	0.600	0.654	0.423

Table 1: Accuracy

experiments, while in gridworld experiment, DPSR-K receives more robustness and converges faster than DNN. Moreover, based on Table 1, although DPSR-K loose some accuracy on tunnel world and half moon world experiments, it receives no worse performance than DNN in gridworld experiment, which is the hardest task among three experiments. We hypothesis that our proposed DPSR-K seems to performance better and faster in more sophisticated task, such as grid world but due to the limitation of the number of state and observation, DPSR-K does not receive ideal performance and despite all this it is still better than random guess.

5 Conclusion

In this research, we propose a deep structure based method called DPSR-K to solve the PSR problem. We combine the unsupervised clustering method and the DNN based history probe representation method to predict the upcoming observation based on a period of history observation and action. Our experiment results show that in two simple task DPSR-K does not receive ideal results, however, for more sophisticated task, our proposed DPSR-K performances better than DNN method. We hypothesis this phenomenon is caused by the lack of state in sample tasks. We will do further research about that in future work.

References

[1] M. Dinculescu. Learning approximate representations of partially observable systems. 04 2020.

[2] M. Dinculescu and D. Precup. Approximate predictive representations of partially observable systems. pages 895–902, 08 2010.

[3] M. L. Littman and R. S. Sutton. Predictive representations of state. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1555–1561. MIT Press, 2002.

[4] A. G. B. Richard S. Sutton. The MIT Press, Cambridge, MA, 2018.

[5] S. Singh, M. James, and M. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *arXiv preprint arXiv:1207.4167*, 2012.

[6] M. T. J. Spaan. *Partially Observable Markov Decision Processes*, pages 387–414. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.