Neural belief states for partially observed domains

Pol Moreno* **Jan Humplik*** **George Papamakarios*** **Bernardo Ávila Pires**
Lars Buesing **Nicolas Heess** **Théophane Weber**
DeepMind

1 Learning neural belief states

An important challenge in reinforcement learning arises in domains where the agent's observations are partial or noisy measurements of the state of the environment. In such domains, a policy that depends only on the current observation x_t is generally suboptimal; an optimal policy must in principle depend on the entire history of observations and actions $h_t = (x_1, a_1, \ldots, a_{t-1}, x_t)$.

Alternatively, an optimal policy can depend on a statistic b_t of the history h_t, as long as b_t is sufficient for predicting future observations; in a POMDP, b_t is known as a *belief state* [1, 2, 3]. Ideally, a rich belief state should capture the agent's memory of the past (e.g. where the agent has been) as well as represent the agent's remaining uncertainty about the world (e.g. what the agent has not yet seen but may be able to infer). The most commonly used solution for tackling POMDPs in deep RL is to endow agents with memory (e.g. LSTMs), which could in principle learn such a representation implicitly through model-free reinforcement learning. However, the memory formed is often limited, and the reward signal alone may be too weak to form a good approximation to the belief state. Enriching the learning signal with auxiliary losses (see e.g. [4, 5]) often increases performance, but does not capture a clear or interpretable notion of uncertainty. Similar observations were made in [6], where the belief state is represented by a collection of particles. [7] also investigates agents with predictive modeling of the environment, but filtering state-space models do not provide access to a full belief state, single samples only. In contrast, our approach is to learn a *neural* belief state, i.e. a representation that fully parametrizes the state distribution.

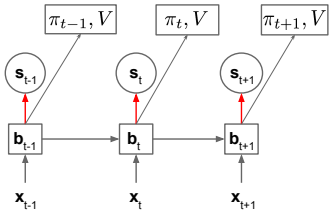
Our design of agent architectures is guided by the fact that the posterior $p(s_t \mid h_t)$ over the state of the environment s_t given the history h_t is itself a belief state, and so is any statistic $b_t = \phi(h_t)$ for which $p(s_t \mid b_t) = p(s_t \mid h_t)$. Our proposed agent architectures (Fig. 1a and 1b) consist of a recurrent model $b_t = f(b_{t-1}, a_{t-1}, x_t)$ that aggregates the history and computes the belief state b_t, and a conditional generative model $p(s_t \mid b_t)$ that predicts a distribution of the state given b_t. In addition to maximizing reward, we train the agent with an auxiliary loss that encourages $p(s_t \mid b_t)$ to become the posterior over the true state of the environment. We investigate two training schemes for achieving this:

(i) *Supervised with privileged information*, where the true current state of the environment is provided as a target to the generative model at training time (but *not* at test time).

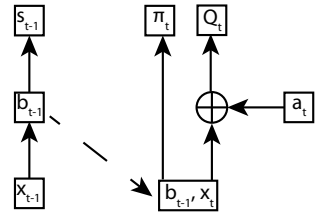
(ii) *Unsupervised*, where the state is defined implicitly by reconstructing future observations (no privileged information).

2 Experiments and discussion

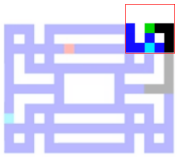
Our first environment is MiniPacman [8], a 15×19 grid world where the agent (Pacman) navigates a maze and tries to eat all the food while being chased by enemies. The observation is restricted to a 5×5 window around Pacman (Fig. 2a). The second is Numpad, an environment with continuous states and actions. A torque-actuated spherical robot (6 DoF, 3 action dimensions) has to visit a sequence of numbered tiles on a platform which light up if they were stepped on in the right order and turn off when stepped on out of sequence. The robot only observes its location on a platform, an indicator of which tiles are currently active, and a *partial* specification of the sequence of tiles it has to visit (Fig. 2b and 2c, also appendix A). This task is difficult because the agent needs to search for the correct sequence and memorize previous unsuccessful attempts, while also learning to actuate the rolling ball so as to produce temporally coherent behavior which includes turning and speeding up.



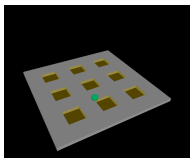
(a) Actor-critic architecture with shared belief and policy networks.



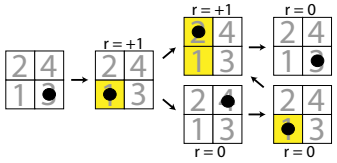
(b) Actor-critic architecture with separate belief and policy networks. The policy network takes the belief from the previous step as an additional input.



(a) MiniPacman.



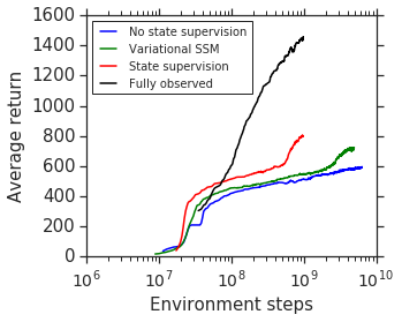
(b) Numpad.



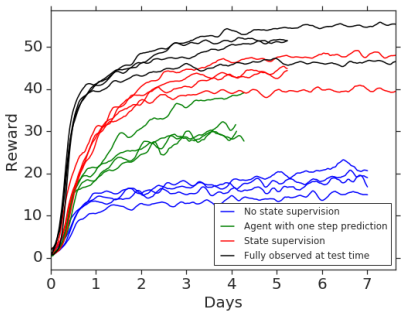
(c) Logic of Numpad: the sequence to be visited is [1; 2] but the agent only observes the first digit. The agent doesn't know how to make the right move in the 2nd step, so it may visit 4 before trying again (appendix A).

Fig. 3a and 3b show average reward vs training time. In MiniPacman we used the shared architecture (Fig. 1a), whereas in Numpad we used the separate architecture (Fig. 1b). The supervised agent (red) is trained with the full state as privileged information. In MiniPacman, the unsupervised agent (green) is a variational state-space model trained to reconstruct the sequence of observations, whereas in Numpad it is a next-step predictor trained to predict the state of the tiles in the next step. For comparison, we also trained a baseline agent which did not utilize any belief-related auxiliary losses (blue), as well as an agent which had access to the full state of the environment (black). For a fair comparison, all agents follow the same architecture (see appendix B for details).

Discussion. We propose to train generative models of the true state given the internal state of an agent, in order to endow agents with a notion of a belief state. There are two key benefits. First, such models enable us to inspect and interpret the agent's belief about the state of the world (appendix C, see also [9]). We find that model-free agents often capture rather limited notions of uncertainty about the world. Although unsupervised prediction helps, current techniques may still provide too weak a signal to form a strong belief state / memory. Second, our experiments show that learning belief states can lead to significant improvements in RL performance without any changes to the agent architecture. If privileged information is not available, we show that modest improvements in RL performance can still be achieved by unsupervised modeling of the state. As such, state supervision provides an 'upper bound' to unsupervised modeling. In future work, we will scale up our approach to more complex environments, and investigate whether other unsupervised prediction models and algorithms may help close the gap between state supervision and predictive modeling.



(a) Average reward on MiniPacman.



(b) Average reward on Numpad.

References

[1] Karl J Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

[2] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[3] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[4] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429, 2018.

[5] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[6] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for POMDPs. *arXiv preprint arXiv:1806.02426*, 2018.

[7] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[8] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5690–5701, 2017.

[9] Anonymous. Neural predictive belief representations. *Submitted to International Conference on Learning Representations*, 2019. Under review.

[10] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pages 3549–3557, 2016.

[11] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.

[13] Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28*, pages 2944–2952. 2015.

[14] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 1054–1062. 2016.

A Numpad environment details

The agent is supposed to visit a sequence of up to four tiles specified as four numbers (if the sequence is shorter, then the specification is padded with -1). The specification which the agent has access to is a masked version of the full sequence, i.e. some digits are masked out with 0s. The reward structure is as follows (see Fig. 2c for an illustration). When the agent visits a correct tile for the first time, it gets a reward of $+1$, and the tile activates (i.e. it changes color). When it steps on a tile which breaks

the correct order, then all the tiles are deactivated and the agent does not receive any reward. At this point, the agent needs to reactivate all the tiles which it already got right before but now it does not get any reward for activating them. Therefore, it can collect at most four $+1$ rewards before it finishes the sequence for the first time. Once that happens, it is allowed to reset the rewards by stepping on an arbitrary tile not on the sequence, and start re-executing the sequence. The agent can keep repeating the sequence and collecting rewards until it runs out of time (which is 500 control steps).

The agent observes its position and orientation on the platform, an indicator of which tiles are currently active, partial specification of the sequence to be executed, and several proprioceptive features necessary for motion control. The sequences are uniformly sampled from continuous paths, i.e. we only allow sequences such that each tile is a neighbor of the previous tile. We sample the masks over sequences by first uniformly sampling how many digits will be hidden, and then uniformly sampling from binary masks with that many zeros.

B Architecture and training details

B.1 MiniPacman

Fig. 4a shows the architecture chosen to train the MiniPacman agents, with the supervised component marked in red. The belief state transition function is a convolutional LSTM, and the belief state loss uses the variational model DRAW [10]. We represent the full state s_t as a one-hot encoding of the agent per cell in the MiniPacman grid. The belief state b_t is further encoded via a ConvNet and fed to an LSTM network that computes the policy logits π_t and value function V_t.

For the unsupervised experiment, we used a variational state-space model that encodes the sequence of observations into a sequence of latent states, and then decodes them back to reconstructed observations. The encoder $q(s_t \mid b_t)$ has the same architecture as in Fig. 4a. The decoder is a state-space model, where the state transition model $p(s_t \mid s_{t-1})$ is an MLP that outputs a diagonal Gaussian, and the observation model $p(x_t \mid s_t)$ is a transpose ConvNet that outputs a discrete distribution. The auxiliary loss is the following variational lower bound:

$$\log p(x_1, \ldots, x_T) \geq \sum_t \mathbb{E}_{s_t \sim q(s_t \mid b_t)} \left[\log p(x_t \mid s_t) + \log p(s_t \mid s_{t-1}) - \log q(s_t \mid b_t) \right]. \quad (1)$$

Both the supervised and the unsupervised agent were trained using the distributed actor-critic set-up of the Importance Weighted Actor-Learner Architecture [11].

B.2 Numpad

The agents consist of networks which process the observations with a MLP before passing them into a LSTM. The output of the LSTM is then passed through a single linear layer to output the relevant targets. For the actor, the targets are the parameters of a diagonal Gaussian distribution which models the policy. For the belief network, the targets are the parameters of the posterior distribution which we model as four independent categorical distributions (one for each number in a sequence of four). For the critic, the target is the action-value. The critic also processes an action by concatenating it with the outputs of the MLP.

The agents for the Numpad environment are trained using a distributed, asynchronous version of the soft actor-critic algorithm [12, 13] in which the critic is learned using the Retrace algorithm [14].

C Visualizing and interpreting the belief state

In addition to learning a neural belief state, the posterior $p(s_t \mid b_t)$ can be used to interpret the belief state at test time, and visualize the agent's uncertainty about the state of the world. This information enables us to diagnose the agent's behavior, and potentially improve on its weaknesses. We test the ability of the agents in Sec. 2 to understand the environment dynamics by training a new DRAW model conditioned on their belief states. In Fig. 4b we show the negative ELBO loss on a validation set of MiniPacman trajectories. We see how the agent with privileged supervision results in a significantly better predictive performance, and how the variational SSM agent has captured the uncertainty of the environment better than the one without auxiliary losses. Fig. 5 and 6 show different snapshots of the