SUPPLEMENTARY: KNOWLEDGE IS REWARD: LEARNING OPTIMAL EXPLORATION BY PREDICTIVE REWARD CASHING

Anonymous authors
Paper under double-blind review

A COMPUTING CROSS-VALUES BY BELLMAN EQUATION AND VALUE ITERATIONS

In a finite horizon problem with known transition and reward distributions, the cross-values can be computed by backward iterations that are a straightforward extension of the bellman equation:

$$\begin{cases} v_t^{e_1}(x_t; e_1) = R\left(r_t \mid x_t, a_t, e_1\right) + \gamma \sum_{x_{t+1}} T\left(x_{t+1} \mid x_t, a_t, e_1\right) v_{t+1}^{e_1}(x_{t+1}; e_1) \\ v_t^{e_1}(x_t; e_2) = R\left(r_t \mid x_t, a_t, e_2\right) + \gamma \sum_{x_{t+1}} T\left(x_{t+1} \mid x_t, a_t, e_2\right) v_{t+1}^{e_1}(x_{t+1}; e_2) \\ a_t = \underset{\alpha}{\operatorname{argmax}} \left(R\left(r_t \mid x_t, a_t, e_1\right) + \gamma \sum_{x_{t+1}} T\left(x_{t+1} \mid x_t, a_t, e_1\right) v_{t+1}^{e_1}(x_{t+1}; e_1) \right) \end{cases} . \quad (1)$$

This algorithm allows to efficiently compute the cross-values for an arbitrary pair of environments e_1 and e_2 in a highly parallelizable manner. As usual, the system of equations can be turned into value iterations in the infinite horizon setting simply by dropping the explicit time dependency from the value function.

B CONVERGENCE OF BELIEF HORIZON SAMPLING

Here we will prove that tabular TD learning training with belief horizon sampling convergences to the true cross-value table under the usual conditions of TD learning. Like the rest of this paper, this section will have a rather informal tone. However, all reasoning can be easily rigorously formalized.

We define the deterministic inference setting as a belief-augmented Markov decision process where the belief state can only transition from its initial prior state b_0 into one of the S possible deterministic states, here denoted as δ_j. These terminal states correspond to posterior distributions with zero entropy. In other words, each of them corresponds to a unique (non-augmented) Markov decision process. After a sequence of observed transitions $\{ x_\tau, a_\tau, r_\tau \}_{\tau=0}^{t-1}$, the belief state b_τ can either be equal to b_0 or to one of the δ_js. In the latter case, at least one of the observed transitions has zero probability under all environments but one. On the other hand, in the former case all transitions have equal probability under all environments (this follows from the fact that the prior belief state has not been updated).

In belief horizon sampling, the ground truth environment e_1 is sampled from the terminal belief state b_T. This environment is then used to select the proper entry of the cross-values table and to perform a TD update. The sampled environment e_1 can either be equal or different from the (inaccessible) true environment e^* that generated the transitions. If it is equal then standard TD convergence results apply directly. Since all the δ_js represent deterministic posterior distributions, e_1 can be different from e^* only when b_T is equal to b_0. In this case, the sequence $\{ x_\tau, a_\tau, r_\tau \}_{\tau=0}^{t-1}$ could be generated by any of the possible environments with equal probability and, consequently, the transitions can be used to update any of the cross-values tables as if they were sampled from their own environments without affecting convergence.

C DETAILS OF THE TREASURE MAP EXPERIMENT

Task details. In each episode, the agent starts in a random location in the grid-world and stays in the environment for $T = 25$ time steps. At each time step, the agent can move to each of the 8

(except at the borders/corners) neighboring cells or stay in place. Reward is collected at each time step and is discounted with $\gamma = 0.96$.

Architecture details. The network output is a 2d array of value of future information, one for each spatial location. The network input was a tensor with two spatial dimensions and two channels, one for each parameter of the beta distributions. The network had two convolutional layers, the first with kernel sizes $(3, 3)$ and 20 output channels and the second with kernel sizes $(1, 1)$ and 1 output channels. Relu activation functions were applied after the first convolutional layer. The output tensor was scaled by 0.01 and then summed to a learnable constant table of values, one for each spatial location.

Policy details. Predictively cashed reward $\tilde{\lambda}_t$ was computed from the cross-values using the approximation in Eq. **??** and $N = 80$ samples. We used an ϵ-greedy training policy with respect to the value function. This does not require the training of a policy network since the transition model is known and deterministic. The TD loss for each transition was $\mathcal{L}(w) = (\tilde{\lambda}_t + \gamma v(x_{t+1}, b_{t+1}; w) - v(x_t, b_t; w))^2$.

Baseline architecture details. The baseline TD algorithm has to learn a substantially more complex value function which, loosely speaking, includes both the exploitation and the exploration part of the value. For these reasons, we thought it more effective to use a fully connected architecture as it is in theory capable of learning arbitrarily complex dependencies between the input variables. We therefore used a two-layers fully connected architecture and $4H^2$ hidden layers, where H is the linear size of the grid-world. To facilitate learning, the output of the network corresponding to the $j, k - th$ grid point was $\rho_{jk} + f_{jk}(b_t)$, where $f_{jk}(b_t)$ denotes the output of the network and ρ_{jk} is the expected reward probability of the cell given the current belief. This significantly improves performance. Since our method uses the solution of the value iterations, we also used an alternative method (VI-TD) with the belief-augmented value given by $v(\rho_{jk}) + f_{jk}(b_t)$, with $v(\rho_{jk})$ being the value obtained from the expected values by value iteration.