Preferential Temporal Difference Learning

Nishanth Anand [1 2] **Doina Precup** [1 2 3]

Abstract

Temporal-Difference (TD) learning is a general and very useful tool for estimating the value function of a given policy, which in turn is required to find good policies. Generally speaking, TD learning updates states whenever they are visited. When the agent lands in a state, its value can be used to compute the TD-error, which is then propagated to other states. However, it may be interesting, when computing updates, to take into account other information than whether a state is visited or not. For example, some states might be more important than others (such as states which are frequently seen in a successful trajectory). Or, some states might have unreliable value estimates (for example, due to partial observability or lack of data), making their values less desirable as targets. We propose an approach to re-weighting states used in TD updates, both when they are the input and when they provide the target for the update. We prove that our approach converges with linear function approximation and illustrate its desirable empirical behaviour compared to other TD-style methods.

1. Introduction

The value function is a crucial quantity in reinforcement learning (RL), summarizing the expected long-term return from a state or a state-action pair. The agent uses this knowledge to make informed action decisions. Temporal Difference (TD) learning methods (Sutton, 1988) enable updating the value function before the end of an agent's trajectory by contrasting its return predictions over consecutive time steps, i.e., computing the *temporal difference error* (TD-error). State-of-the-art RL algorithms, e.g. Mnih et al. (2015); Schulman et al. (2017) use this idea coupled with function approximation.

[1]Mila (Quebec Artificial Intelligence Institute), Montreal, Canada [2]School of Computer Science, McGill University, Montreal, Canada [3]Deepmind, Montreal, Canada. Correspondence to: Nishanth Anand <nishanth.anand@mail.mcgill.ca>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

TD-learning can be viewed as a way to approximate dynamic programming algorithms in Markovian environments (Barnard, 1993). But, if the Markovian assumption does not hold (as is the case when function approximation is used to estimate the value function), its use can create problems (Gordon, 1996; Sutton & Barto, 2018). To see this, consider the situation depicted in Figure 1a, where an agent starts in a fully observable state and chooses one of two available actions. Each action leads to a different long-term outcome, but the agent navigates through aliased states that are distinct but have the same representation before observing the outcome. This setting poses two challenges:

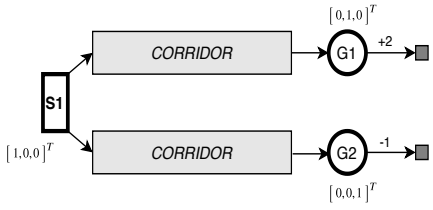
1. Temporal credit assignment: The starting state and the outcome state are temporally distant. Therefore, an efficient mechanism is required to propagate the credit (or blame) between them.

2. Partial observability: With function approximation, updating one state affects the value prediction at other states. If the generalization is poor, TD-updates at partially observable states can introduce errors, which propagate to estimates at fully observable states.

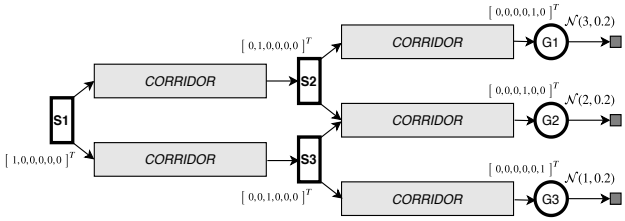
TD(λ) is a well known class of *span independent algorithm* (van Hasselt & Sutton, 2015) for temporal credit assignment introduced by Sutton (1988) and further developed in many subsequent works, e.g. Singh & Sutton (1996); Seijen & Sutton (2014), which uses a *recency* heuristic: any TD-error is attributed to preceding states with an exponentially decaying weight. However, recency can lead to inefficient propagation of credit (Aberdeen, 2004; Harutyunyan et al., 2019). Generalized TD(λ) with a state-dependent λ can tackle this problem (Sutton, 1995; Sutton et al., 1999; 2016; Yu et al., 2018). This approach mainly modifies the target used in the update, not the extent to which any state is updated. Besides, the update sequence may not converge. Emphatic TD (Sutton et al., 2016) uses an independent interest function, not connected to the eligibility parameter, in order to modulate how much states on a trajectory are updated.

In this paper, we introduce and analyze a new algorithm: *Preferential Temporal Difference (Preferential TD or PTD) learning*, which uses a single state-dependent *preference function* to emphasize the importance of a state both as an input for the update, as well as a participant in the target. If

a state has a low preference, it will not be updated much, and its value will also not be used much as a bootstrapping target. This mechanism allows, for example, *skipping* states that are partially observable and propagating the information instead between fully observable states. Older POMDP literature (Loch & Singh, 1998; Theocharous & Kaelbling, 2004) has demonstrated the utility of similar ideas empirically. We provide a convergence proof for the expected updates of Preferential TD in the linear function approximation setting and illustrate its behaviour in partially observable environments.



(a) Task 1



(b) Task 2

Figure 1. Delayed effect MDPs: decision states are shown as boxes, goal states are shown in circles. Feature vectors are specified next to the states. The corridor represents a chain of partially observable states.

2. Preliminaries and Notation

A Markov Decision Process (MDP) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the finite set of states, $\mathcal{A}$ is the finite set of actions, $\mathcal{P}(s'|s,a) = \mathbb{P}\{s_{t+1} = s' | s_t = s, a_t = a\}$ is the transition model, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. The agent interacts with the environment in state $s_t \in \mathcal{S}$ by selecting an action $a_t \in \mathcal{A}$ according to its policy, $\pi(a|s) = \mathbb{P}\{a_t = a | s_t = s\}$. As a consequence, the agent transitions to a new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and receives reward $r_{t+1} = r(s_t, a_t)$. We consider the policy evaluation setting, where the agent's goal is to estimate the value function:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s], \tag{1}$$

where $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}$ is the discounted return obtained by following the policy π from state s. If $|\mathcal{S}|$ is very big, v_π must be approximated by using a function approximator. We consider linear approximations:

$$v_{\mathbf{w}}(s) = \mathbf{w}^T \phi(s), \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^k$ is the parameter vector and $\phi(s)$ is the feature vector for state s. Note that the linear case encompasses both the tabular case as well as the use of fixed non-linear features, as detailed in Sutton & Barto (2018). By defining a matrix Φ whose rows are $\phi^T(i)$, $\forall i \in \mathcal{S}$, this approximation can be written as a vector in $\mathbb{R}^{|\mathcal{S}|}$ as $\mathbf{v}_\mathbf{w} = \Phi\mathbf{w}$.

TD(λ) (Sutton, 1984; 1988) is an online algorithm for updating $\mathbf{w}$, which performs the following update after every time step:

$$\begin{aligned} \mathbf{e}_t &= \gamma\lambda \mathbf{e}_{t-1} + \phi(s_t) \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha_t (r_{t+1} + \gamma \mathbf{w}^T \phi(s_{t+1}) - \mathbf{w}^T \phi(s_t)) \mathbf{e}_t, \end{aligned} \tag{3}$$

where α_t is the learning rate parameter, $\lambda \in [0,1]$ is the eligibility trace parameter, used to propagate TD-errors with exponential decay to states that are further back in time, and $\mathbf{e}_t$ is the eligibility trace.

3. Preferential Temporal Difference Learning

Let $\beta : \mathcal{S} \to [0,1]$ be a preference function, which assigns a certain importance to each state. A preference of $\beta(s) = 0$ means that the value function will not be updated at all when s is visited, while $\beta(s) = 1$ means s will receive a full update. Note, however, that if s is not updated, its value will be completely inaccurate, and hence it should not be used as a target for predecessor states because it would lead to a biased update. To prevent this, we can modify the return G_t to a form that is similar to λ-returns (Watkins, 1989; Sutton & Barto, 2018), by bootstrapping according to the preference:

$$\begin{aligned} G_t^\beta &= r_{t+1} + \\ &\quad \gamma[\beta(s_{t+1}) \mathbf{w}^T \phi(s_{t+1}) + (1 - \beta(s_{t+1})) G_{t+1}^\beta]. \end{aligned} \tag{4}$$

The update corresponding to this return leads to the offline Preferential TD algorithm:

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\beta(s_t)(G_t^\beta - \mathbf{w}^T\phi(s_t))\phi(s_t), \\ &= \mathbf{w}_t + \\ &\alpha_t \Big(\underbrace{\beta(s_t) G_t^\beta + (1 - \beta(s_t)) \mathbf{w}^T \phi(s_t)}_{target} - \mathbf{w}^T \phi(s_t) \Big) \phi(s_t). \end{aligned} \tag{5}$$

The expected target of offline PTD (cf. equation 5) can be written as a Bellman operator:

Theorem 1. *The expected target in the forward view can be summarized using the operator*

$$\mathcal{T}^\beta \mathbf{v} = B(I - \gamma \mathcal{P}_\pi (I - B))^{-1} (r_\pi + \gamma \mathcal{P}_\pi B \mathbf{v}) + (I - B)\mathbf{v},$$

where B is the $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix with $\beta(s)$ on its diagonal and r_π and $\mathcal{P}_\pi$ are the state reward vector and state-to-state transition matrix for policy π.

We obtain the desired result by considering expected updates in vector form. The complete proof is provided in Appendix A.1.

Using equation 4, one would need to wait until the end of the episode to compute G_t^β. We can turn this into an online update rule using the eligibility trace mechanism (Sutton, 1984; 1988):

$$\mathbf{e}_t = \gamma(1-\beta(s_t))\mathbf{e}_{t-1} + \beta(s_t)\phi(s_t), \tag{6}$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t(r_{t+1} + \gamma \mathbf{w}^T\phi(s_{t+1}) - \mathbf{w}^T\phi(s_t))\mathbf{e}_t,$$

where $\mathbf{e}_t$ is the eligibility trace as before. The equivalence between the offline and online algorithm can be obtained following Sutton (1988); Sutton & Barto (2018). Theorem 2 in Section 4 formalizes this equivalence. We can turn the update equations into an algorithm as shown in algorithm 1.

Algorithm 1 Preferential TD: Linear FA

Input: π,γ,β, ϕ
Initialize: $\mathbf{w}_{-1} = 0, e_{-1} = 0$
Output: $\mathbf{w}_T$
for $t : 0 \to T$ **do**
  Take action $a \sim \pi(s_t)$, observe r_{t+1}, s_{t+1}
  $v(s_t) = \mathbf{w}_t^T\phi(s_t), v(s_{t+1}) = \mathbf{w}_t^T\phi(s_{t+1})$
  $\delta_t = r_{t+1} + \gamma v(s_{t+1}) - v(s_t)$
  $e_t = \beta(s_t)\phi(s_t) + \gamma(1-\beta(s_t))e_{t-1}$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t\delta_t e_t$
end for

4. Convergence of PTD

We consider a finite, irreducible, aperiodic Markov chain. Let $\{s_t | t = 0, 1, 2, \dots\}$ be the sequence of states visited by the Markov chain and let $d_\pi(s)$ denote the steady-state probability of $s \in \mathcal{S}$. We assume that $d_\pi(s) > 0$, $\forall s \in \mathcal{S}$. Let D^π be the $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix with $D^\pi_{i,i} = d_\pi(i)$. We denote the Euclidean norm on vectors or Euclidean-induced norm on matrices by $\|\cdot\|$: $\|A\| = \max_{\|x\|=1} \|Ax\|$. We make the following assumptions to establish the convergence result [1].

Assumption 1. *The feature matrix, $\Phi \in \mathbb{R}^{|\mathcal{S}|} \times \mathbb{R}^k$ is a full column rank matrix, i.e., the column vectors $\{\phi_i | i = 1 \dots k\}$ are linearly independent. Also, $\|\Phi\| \leq M$, where M is a constant.*

Assumption 2. *The Markov chain is rapidly mixing:*

$$|\mathcal{P}_\pi(s_t = s | s_0) - d_\pi(s)| \leq C\rho^t, \ \forall s_0 \in \mathcal{S}, \rho < 1,$$

where C is a constant.

Assumption 3. *The sequence of step sizes satisfies the Robbins-Monro conditions:*

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \textit{and} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty.$$

The update equations of Preferential TD (cf. equation 6) can be written as:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t(b(X_t) - A(X_t)\mathbf{w}_t),$$

where $b(X_t) = \mathbf{e}_t r_{t+1}$, $A(X_t) = \mathbf{e}_t(\phi(s_t) - \gamma\phi(s_{t+1}))^T$, $X_t = (s_t, s_{t+1}, e_t)$ and $\mathbf{e}_t$ is the eligibility trace of PTD. Let $\mathbf{A} = \mathbb{E}_{d_\pi}[A(X_t)]$ and $\mathbf{b} = \mathbb{E}_{d_\pi}[b(X_t)]$. Let $\mathcal{P}_\pi^\beta$ be a new transition matrix that accounts for the termination due to bootstrapping and discounting [2], defined as: $\mathcal{P}_\pi^\beta = \gamma\Big(\sum_{k=0}^{\infty}(\gamma\mathcal{P}_\pi(I-B))^k\Big)\mathcal{P}_\pi B$. This is a sub-stochastic matrix for $\gamma < 1$ and a stochastic matrix when $\gamma = 1$.

Lemma 1. *The expected quantities $\mathbf{A}$ and $\mathbf{b}$ are given by $\mathbf{A} = \Phi^T D^\pi B(I - \mathcal{P}_\pi^\beta)$ and $\mathbf{b} = \Phi^T D^\pi B(I - \gamma\mathcal{P}_\pi)^{-1} r_\pi$.*

We use the proof template from Emphatic TD (Sutton et al., 2016) to get the desired result. The proof is provided in Appendix A.1.

We are now equipped to establish the forward-backward equivalence.

Theorem 2. *The forward and the backward views of PTD are equivalent in expectation:*

$$\mathbf{b} - \mathbf{A}\mathbf{w} = \Phi^T D\Big(\mathcal{T}^\beta(\Phi\mathbf{w}) - \Phi\mathbf{w}\Big).$$

The proof is provided in Appendix A.2.

The next two lemmas verify certain conditions on $\mathbf{A}$ that are required to show that it is positive definite.

Lemma 2. $\mathbf{A}$ *has positive diagonal elements with positive row sums.*

Proof. Consider the term $(I - \mathcal{P}_\pi^\beta)$. $\mathcal{P}_\pi^\beta$ is a sub-stochastic matrix for $\gamma \in [0, 1)$, hence, $\sum_j [\mathcal{P}_\pi^\beta]_{i,j} < 1$. Therefore, $\sum_j [I - \mathcal{P}_\pi^\beta]_{i,j} = 1 - \sum_j [\mathcal{P}_\pi^\beta]_{i,j} > 0$. Additionally, $[I - \mathcal{P}_\pi^\beta]_{i,i} > 0$ since $[\mathcal{P}_\pi^\beta]_{i,i} < 1$. □

Lemma 3. *The column sums of $\mathbf{A}$ are positive.*

The idea of the proof is to show that $d_\pi B\mathcal{P}_\pi^\beta = d_\pi B$ for $\gamma = 1$. This implies $d_\pi B(I - \mathcal{P}_\pi^\beta) = 0$ for $\gamma = 1$ and $d_\pi B(I - \mathcal{P}_\pi^\beta) > 0$ for $\gamma \in [0, 1)$ as $d_\pi B\mathcal{P}_\pi^\beta < d_\pi B$. Therefore, column sums are positive. The complete proof is provided in Appendix A.2. Note that $d_\pi B$ is not the stationary distribution of $\mathcal{P}_\pi^\beta$, as it is unnormalized.

[1] Similar assumptions are made to establish the convergence proof of TD(λ) in the linear function approximation setting for a constant λ (Tsitsiklis & Van Roy, 1997).

[2] $\mathcal{P}_\pi^\beta$ is similar to ETD's $\mathcal{P}_\pi^\lambda$ (Sutton et al., 2016), $\beta(s)$ takes the role of $1 - \lambda(s)$ and we consider a constant discount setting.

Lemma 4. $\mathbf{A}$ *is positive definite.*

Proof. From lemmas 2 and 3, the row sums and the column sums of $\mathbf{A}$ are positive. Also, the diagonal elements of $\mathbf{A}$ are positive. Therefore, $\mathbf{A} + \mathbf{A}^T$ is a strictly diagonally dominant matrix with positive diagonal elements. We can use corollary 1.22 from Varga (1999) (provided in Appendix A.1 for completeness) to conclude that $\mathbf{A} + \mathbf{A}^T$ is positive definite. Hence, $\mathbf{A}$ is also positive definite. □

The next lemma shows that the bias resulting from initial updates, i.e., when the chain has not yet reached the stationarity, is controlled.

Lemma 5. *We have,* $\sum_{t=0}^{\infty} \|\mathbb{E}_\pi[A(X_t)|X_0] - \mathbf{A}\| \leq C_1$ *and* $\sum_{t=0}^{\infty} \|\mathbb{E}_\pi[b(X_t)|X_0] - \mathbf{b}\| \leq C_2$ *under assumption 2, where* C_1 *and* C_2 *are constants.*

The proof is provided in Appendix A.3.

We now present the main result of the paper.

Theorem 3. *The sequence of expected updates computed by Preferential TD converges to a unique fixed point.*

Proof. We can establish convergence by satisfying all the conditions of a standard result from the stochastic approximation literature, such as theorem 2 in Tsitsiklis & Van Roy (1997) or proposition 4.8 in Bertsekas & Tsitsiklis (1996) (provided in Appendix A.3 for completeness). Assumption 3 satisfies the step-size requirement. Lemmas 1 and 4 meet conditions 3 and 4. Lemma 5 controls the noise from sampling, hence satisfying the final requirement. Therefore, the expected update sequence of Preferential TD converges. The fixed point lies in the span of the feature matrix and this point has zero projected Bellman error. That is, $\Pi\Big(\mathcal{T}^\beta(\Phi\mathbf{w}_\pi) - \Phi\mathbf{w}_\pi\Big) = 0$, where $\Pi = \Phi(\Phi^T D^\pi \Phi)^{-1}\Phi^T D^\pi$ is the projection matrix. □

Consequences: It is well-known that using state-dependent bootstrapping or re-weighing updates in TD(λ) will result in convergence issues even in the on-policy setting. This was demonstrated using counterexamples (Mahmood, 2017; Ghiassian et al., 2017). Our result establishes convergence when a state-dependent bootstrapping parameter is used *in addition to* re-weighing the updates. We analyze these examples below and show that they are invalid for Preferential TD. This makes Preferential TD one of only two algorithms to converge with standard assumptions in the function approximation setting, Emphatic TD being the other (Yu, 2015).

Counterexample 1 (Mahmood, 2017): Two state MDP with $\mathcal{P}_\pi = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$, $\Phi = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$, $\lambda = \begin{bmatrix} 0.99 & 0.8 \end{bmatrix}$, $d_\pi = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$, $\gamma = 0.99$. The key matrix of TD(λ) is given by, $\mathbf{A} = \begin{bmatrix} -0.0429 \end{bmatrix}$, which is not positive definite. Therefore, the updates will not converge. The key matrix of Preferential TD for the same setup is $\mathbf{A} = \begin{bmatrix} 0.009 \end{bmatrix}$ which is positive definite [3]. The second counterexample is analyzed in Appendix A.3.

5. Related work

A natural generalization of TD(λ) is to make λ a state-dependent function (Sutton, 1995; Sutton et al., 1999; 2016). This setting produces significant improvements in many cases (Sutton, 1995; Sutton et al., 2016; White & White, 2016; Xu et al., 2018). However, it only modifies the target during bootstrapping, not the extent to which states are updated. The closest approach in spirit to ours is Emphatic TD(λ) (Emphatic TD or ETD), which introduces a framework to reweigh the updates using the interest function ($i(s) \in \mathbb{R}^+, \forall s \in \mathcal{S}$) (Sutton et al., 2016). Emphatic TD uses the interest of past states and the state-dependent bootstrap parameter λ of the current state to construct the emphasis, $M_t \in \mathbb{R}^+$, at time t. The updates are then weighted using M_t. By this construction, the emphasis is a trajectory dependent quantity. Two different trajectories leading to a particular state have a different emphasis on the updates. This can be beneficial in the off-policy case, where one may want to carry along importance sampling ratios from the past. However, a trajectory-based quantity can result in high variance, as the same state could get different updates depending upon its past.

Emphatic TD and Preferential TD share the idea of reweighing the updates based on the agent's preference for states. Emphatic TD uses a trajectory-based quantity, whereas Preferential TD uses a state-dependent parameter to reweigh the updates. Furthermore, Preferential TD uses a single parameter to update and bootstrap, instead of two in Emphatic TD. Our analysis in Section 4 suggests that the two algorithms' fixed points are also different. However, we can set up Emphatic TD to achieve a similar effect to PTD (though not identical) when $\beta \in \{0, 1\}$, by setting $\lambda = 1 - \beta$ and the interest of a state proportional to the preference for that state. In our experiments, we use this setting as well, which will be denoted as ETD-variable algorithm.

The fact that preference is a state-dependent quantity fits well with the intuition that if a state is partially observable, we may always want to avoid involving it in bootstrapping, regardless of the trajectory on which it occurs. Preferential TD lies in between TD(λ) and Emphatic TD. Like TD(λ), it uses a single state-dependent bootstrap function. Like Emphatic TD, it reweighs the updates but using only the preference.

[3] We set $\lambda(s_0) = 0.99$ instead of 1, a small change to the original example, because $\beta(s_0) = 0$ when $\lambda(s_0) = 1$ and such states can be excluded from the analysis.

Other works share the idea of ignoring updates on partially observable states, e.g. Xu et al. 2017; Thodoroff et al. 2019. They use a trajectory-based value as a substitute to the value of a partially observable state. Temporal value transport (Hung et al., 2019) uses an attention mechanism to pick past states to update, bypassing partially observable states for credit assignment. However, the theoretical properties of trajectory-dependent values or attention-based credit assignment are poorly understood at the moment. Our method is unbiased, and, as seen from Section 4, it can be understood well from a theoretical standpoint.

Our ideas are also related to the idea of a backward model Chelu et al. (2020), in which an explicit model of precursor states is constructed and used to propagate TD-errors in a planning-style algorithm, with a similar motivation of avoiding partially observable states and improving credit assignment. However, our approach is model-free.

6. Illustrations

In this section, we test Preferential TD on policy evaluation tasks in four different settings[4]: tabular, linear, semi-linear (linear predictor with non-linear features), and non-linear (end-to-end training). Note that our theoretical results do not cover the last setup; however, it is quite easy to implement the algorithm in this setup.

6.1. Tabular

In this setting, the value function is represented as a look-up table. This experiment is provided to assist in understanding PTD as a whole in the limited sample setting for various choices of constant preference function.

Task description: We consider the 19-state random walk problem from Sutton 1988; Sutton & Barto 2018. In this MDP, there are 19 states connected sequentially, with two terminal states on either side. From each state, the agent can choose to go left or right. The agent gets a reward of $+1$ when it transitions to the rightmost terminal state and -1 in the leftmost state; otherwise, rewards are 0. We set $\gamma = 1$ and the policy to uniformly random. We compare Emphatic TD, TD(λ), and PTD on this task. We consider fixed parameters for these methods, i.e., interest in ETD is fixed, λ in ETD and TD(λ) are fixed, and β in PTD is fixed. For ETD, we selected a constant interest of 0.01 on all states (selected from $\{0.01, 0.05, 0.1, 0.25\}$ based on hyperparameter search) for each λ-α pair. We consider fixed parameters to understand the differences between the three algorithms.

Observations: The U-curves obtained are presented in Figure 2, which depicts Root Mean Square Error (RMSE) on all states, obtained in the first 10 training episodes, as a function of α. Various curves correspond to different choices of fixed λ (or β). In TD(λ), when λ is close to 0, the updates are close to TD(0), and hence there is high bias. When λ is close to 1, the updates are close to Monte-Carlo, and there is a high variance. A good bias-variance trade-off is obtained for intermediate values of λ, resulting in low RMSE. A similar trade-off can be observed in PTD; however, the trade-off is different from TD(λ). A β value close to 1 results in a TD(0)-style update (similar to $\lambda = 0$), but $\beta \approx 0$ results in negligible updates (as opposed to a Monte-Carlo update for similar values of λ). The resulting bias-variance trade-off is better.

TD(λ) is very sensitive to the value of the learning rate. When λ gets close to 1, the RMSE is high even for a low learning rate, resulting in sharp U-curves. This is because of the high variance in the updates. However, this is not the case for PTD, which is relatively stable for all values of β on a wide range of learning rates. PTD exhibits the best behaviour when using a learning rate greater than 1 for some values of β. This behaviour can be attributed to negligible updates performed when a return close to Monte-Carlo (β is close to 0) is used. Due to negligible updating, the variance in the updates is contained.

The performance of ETD was slightly worse than the other two algorithms. ETD is very sensitive to the value of the interest function. We observed a large drop in performance for very small changes in the interest value. ETD also performs poorly with low learning rates when the interest is set high. This behaviour can be attributed to the high variance caused by the emphasis.

6.2. Linear setting

Task description: We consider two toy tasks to evaluate PTD with linear function approximation.

In the first task, depicted in Figure 1a, the agent starts in $S1$ and can pick one of the two available actions, $\{up, down\}$, transitioning onto the corresponding chain. A goal state is attached at the end of each chain. Rewards of $+2$ and -1 are obtained upon reaching the goal state on the upper and lower chains, respectively. The chain has a sequence of partially observable states (corridor). In these states, both actions move the agent towards the goal by one state.

For the second task, we attach two MDPs of the same kind as used in task 1, as shown in Figure 1b. The actions in the connecting states $S1, S2, S3$ transition the agent to the upper and lower chains, but both actions have the same effect in the partially observable states (corridor). A goal state is attached at the end of each chain. To make the task challenging, a stochastic reward (as shown in Figure 1b) is given to the agent. In both tasks, the connecting states and the goal

[4]Code to reproduce the results can be found here.

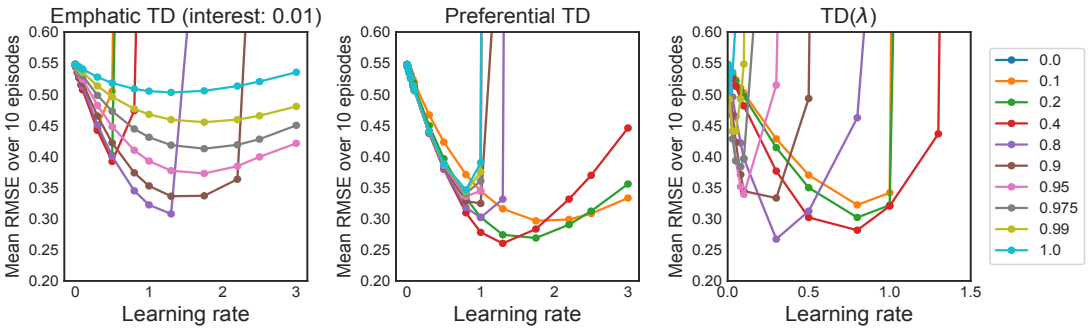
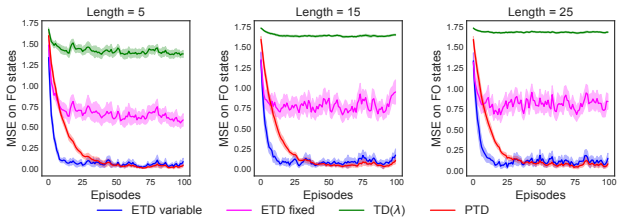
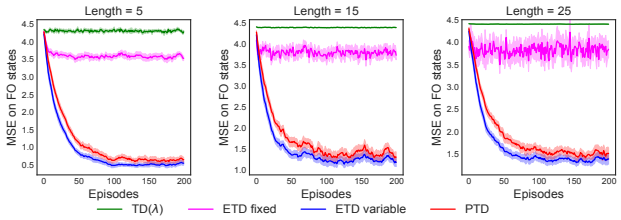


Figure 2. Root Mean Square Error (RMSE) of Emphatic TD, TD(λ), and Preferential TD as a function of learning rate α. Different curves in the plot correspond to different values of λ or β (depending upon the algorithm).

states are fully observable and have a unique representation, while the states in the corridor are represented by Gaussian noise, $\mathcal{N}(0.5, 1)$.



(a) Task 1 results



(b) Task 2 results

Figure 3. The mean squared error of fully observable states' values plotted as a function of episodes for various algorithms. Different plots correspond to different lengths of the corridor.

Setup: We tested 4 algorithms: TD(λ), PTD, and two versions of ETD. The first ETD version has a fixed interest on all the states (referred to as ETD-fixed), and the second variant has a state-dependent interest (referred to as ETD-variable). We set the preference on fully observable states to 1 and partially observable states to 0. For a fair comparison, we make λ a state-dependent quantity by setting the values of the fully observable states to 0 and 1 on partially observable states for TD(λ) and ETD. In the case of ETD-fixed, we set the interest of all states to 0.01 (selected from $\{0.01, 0.05, 0.1\}$ based on the hyperparameter search, see Appendix A.4.2 for details). For ETD-variable, we set the interest to 0.5 for fully observable states and 0 on all other states. This choice results in an emphasis of 1 on fully observable states. ETD-variable yielded very similar performance with various interest values but with different learning rates. We varied the corridor length ($\{5, 10, 15, 20, 25\}$) in our experiments. For each length and algorithm, we chose an optimal learning rate from 20 different values. We used $\gamma = 1$, a uniformly random policy, and the value function is predicted only at fully observable states. The results are averaged across 25 seeds, and a confidence interval of 95% is used in the plots[5].

Observations: A state-dependent λ (or β) results in bootstrapping from fully observable states only. Additionally, PTD weighs the updates on various states according to β. The learning curves on both tasks are reported in Figures 3 and A.2. TD(λ) performs poorly, and the errors increase with the length of the chain on both tasks. Also, the values of the fully observable states stop improving even as the number of training episodes increases. This is to be expected, as TD(λ) updates the values of all states. Thus, errors in estimating the partially observable states affect the predictions at the fully observable states due to the generalization in the function approximator.

ETD-fixed performs a very small update on partially observable states due to the presence of a small interest. Nevertheless, this is sufficient to affect the predictions on fully observable states. The error in ETD-fixed increases with a small increase in the interest. ETD-fixed is also highly sensitive to the learning rate. However, ETD-variable performs the best. With the choice of λ and interest function described previously, ETD-variable ignores the updates on corridor states and, at the same time, does not bootstrap from these states, producing a very low error in both tasks. This effect is similar to PTD. However, bootstrapping and updating are controlled by a single parameter (β) in PTD, while ETD-variable has two parameters, λ and the interest function, and we have to optimize both. Hence, while it

[5]We use the t-distribution to compute the confidence intervals in all the settings.

converges slightly faster than PTD, this is at the cost of increased tuning complexity.

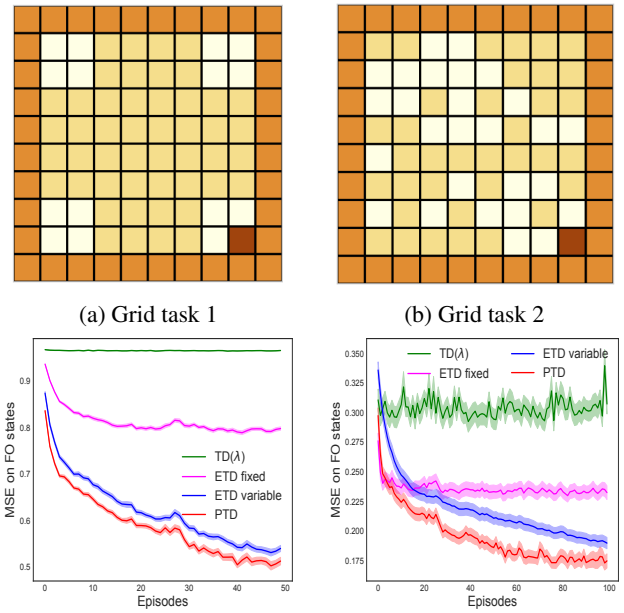
6.3. Semi-linear setting

Task description: We consider two grid navigation tasks shown in figures 4a and 4b. In the first task, the states forming a cross at the center are partially observable. In the second task, 50% of the states are partially observable (sampled randomly for each seed). The agent starts randomly in one of the states in the top left quadrant, and it can choose from four actions $(up, down, left, right)$, moving in the corresponding direction by one state. A reward of $+5$ is obtained when the agent transitions to the state at the bottom right corner. The transitions are deterministic and $\gamma = 0.99$. To generate the features for these tasks, we first train a single-layer neural network to predict the value function using Monte Carlo targets on the fully observable grid. We use the output of the penultimate layer of the trained network to get the features for the states. We then train a linear function approximator on top of these features to predict the value function of a uniformly random policy. We use three grid sizes ($\{8 \times 8, 12 \times 12, 16 \times 16\}$) for experimentation.

Setup: We consider the same four algorithms in this task. We set $\lambda = 0$ and $\beta = 1$ on fully observable states, and set $\lambda = 1$ and $\beta = 0$ on other states. We set the interest to 0.01 for ETD-fixed (selected from $\{0.01, 0.05, 0.1\}$ based on the hyperparameter search, see Appendix A.4.3). The interest is set to 0.5 on fully observable states and 0 on other states for ETD-variable. The results are averaged across 50 seeds, and a confidence interval of 50% is used in the plots[6].

Observations: We report the mean squared error of the value function at fully observable states as the performance measure. The learning curves on both tasks, as a function of number of episodes are reported in figures 4c, 4d, and A.8. TD(λ) performed the worst because of the reasons mentioned earlier. Interestingly, ETD-fixed performed much better in this setting compared to the previous one. Its performance dropped significantly as the grid size increased in task 1, but the drop was marginal for task 2. ETD-variable performed much better than ETD-fixed. The performance on both tasks for grid size 8×8 and 12×12 was the same as PTD. However, there was a slight drop in performance for grid size 16×16. This is because, in the larger tasks, the trajectories can vary widely. As a result, the update on a particular state is reweighed differently depending upon the trajectory, causing variance. The hyperparameter tuning experiments also showed that ETD-variable is sensitive to small changes in learning rate. PTD performs best on all grid sizes and both tasks.

[6]Picked for clarity of the plotting



(a) Grid task 1 (b) Grid task 2

(c) Grid task 1 results (d) Grid task 2 results

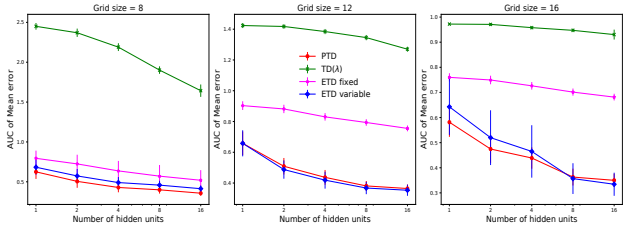
Figure 4. (*Top*) **Grid tasks:** The dark brown square is the terminal state. The light squares are fully observable and the darker squares are partially observable. The walls are indicated in orange. (*Bottom*) The mean squared error of fully observable states' values as a function of the number of episodes for various algorithms on 16×16 grid.

6.4. Non-linear setting

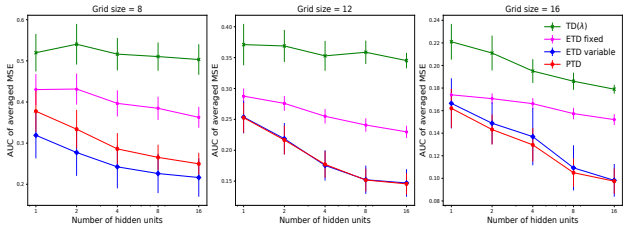
Task description: The aim of this experiment is two-fold: (1) to verify if PTD still learns better predictions when the capacity of the function approximator is decreased, and (2) to verify if PTD is compatible with end-to-end training of a non-linear function approximator. We use the same grid tasks described in section 6.3 for experimentation.

6.4.1. Forward view (Returns)

Setup: We consider the same four algorithms and set λ, β and interest values as described in the previous section. We use a single-layered neural network whose input is a one-hot vector (size $n \times n$) indicating the agent's location in the grid, if the state is fully observable, and Gaussian noise otherwise. We used ReLU non-linearity in the hidden layer. We experimented with $\{1, 2, 4, 8, 16\}$ hidden units to check if PTD can estimate the values when the approximation capacity is limited. We searched for the best learning rate for each algorithm and network configuration from a range of values. We picked the best learning rate based on the area under the error curve (AUC) of the MSE averaged across episodes and 25 seeds. We used 250 episodes for all the tasks. We trained the networks using the forward-view returns for all algorithms.

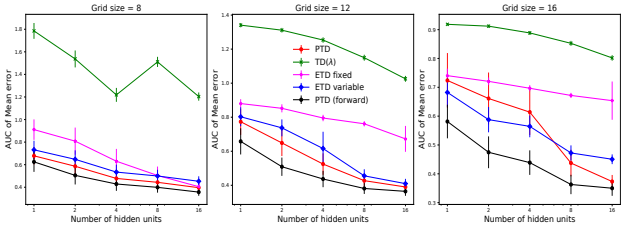


(a) Grid task 1 results

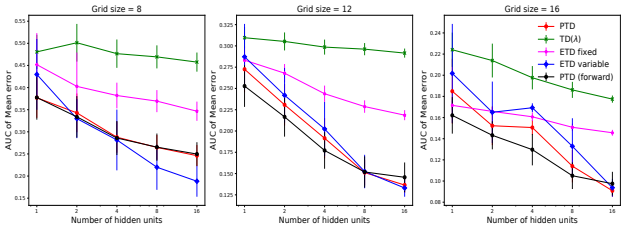


(b) Grid task 2 results

Figure 5. The area under the curve (AUC) of mean squared error of the values of the fully observable states, averaged across seeds and episodes as a function of the number of hidden units for various algorithms. The error bars indicate the confidence interval of 90% across 50 seeds. Different plots corresponds to different sizes of the grid task.



(a) Grid task 1 results



(b) Grid task 2 results

Figure 6. The area under the curve (AUC) of mean squared error of the values of the fully observable states, averaged across seeds and episodes as a function of the number of hidden units for various algorithms. The error bars indicate the confidence interval of 90% across 50 seeds. Different plots corresponds to different sizes of the grid task.

Observations: The results are presented in Figure 5. Each point in the plot is the AUC of the MSE for an algorithm and neural network configuration pair averaged across episodes and seeds. The error bars indicate the confidence interval of 90% across seeds. TD(λ) performs poorly because of the reasons stated before. ETD-fixed performs slightly better than TD(λ), but the error is still high. PTD performs better than the other algorithms on both tasks and across all network sizes. The error is high only in the 16×16 grid for really small networks, with four or fewer hidden units. This is because the number of fully observable states is significantly larger for this grid size, and the network is simply not large enough to provide a good approximation for all these states. Nevertheless, PTD's predictions are significantly better than other algorithms in this setting. The behavior of ETD-variable is similar, but the variance is much higher. As before, hyperparameter tuning experiments indicate that ETD is also much more sensitive to learning rates.

6.4.2. Backward view (Eligibility Traces)

In this experiment, we are interested in finding out how PTD's eligibility traces would fare in the non-linear setting.

Setup: The set of algorithms tested, network architecture, and the task details are same as Section 6.4.1. We also followed the same procedure as the previous section to select the learning rate. We trained the networks in an online fashion using eligibility traces for all algorithms.

Observations: Eligibility traces facilitate online learning, and they provide a way to assign credit to the past states, making it a valuable tool in theory. However, the prediction errors of all the algorithms are higher than their forward view counterparts. The drop in performance is because eligibility traces remember the gradient that is computed with respect to the past parameters, which introduces significant bias. Besides, the bias in the eligibility traces caused instabilities in the training procedure of PTD and ETD-variable for high learning rates. The results are presented in Figure 6. As shown, the performances of TD(λ) and ETD-fixed performs are still poor compared to PTD and ETD-variable. Backward views of PTD and ETD-variable have slightly more error and variance across the seeds than their forward view equivalents when the capacity of the function approximator is small. However, they match the forward view performance when the capacity is increased. PTD still performs slightly better than ETD-variable in the 16×16 grids, indicating the usefulness of its eligibility traces with neural networks.

7. Conclusions and Future Work

We introduced Preferential TD learning, a novel TD learning method that updates the value function and bootstraps in the presence of state-dependent preferences, which allow these operations to be prioritized differently in different states. Our experiments show that PTD compares favourably to other TD variants, especially on tasks with partial observability. However, partial observability is not a requirement

to use our method. PTD can be useful in other settings where updates can benefit from re-weighting. For instance, in an environment with *bottleneck* states, having a high preference on such states could propagate credit faster. Preliminary experiments in Appendix A.4.5 corroborate this claim, and further analysis could be interesting.

We set the preference function from the beginning to a fixed value in our experiments, but an important direction for future work is to learn or adapt it based on data. The preference function plays a dual role: its presence in the targets controls the amount of bootstrapping from future state values, and its presence in the updates determines the amount of re-weighting. Both can inspire learning approaches. The bootstrapping view opens the door to existing techniques such as meta-learning (White & White, 2016; Xu et al., 2018; Zhao et al., 2020), variance reduction (Kearns & Singh, 2000; Downey et al., 2010), and gradient interference (Bengio et al., 2020). Alternatively, we can leverage methods that identify partially observable states or important states (e.g. bottleneck states) (McGovern & Barto, 2001; Stolle & Precup, 2002; Bacon et al., 2017; Harutyunyan et al., 2019) to learn the preference function. We also believe that PTD would be useful in transfer learning, where one could learn parameterized preferences in a source task and use them in the updates on a target task to achieve faster training.

We demonstrated the utility of preferential updating in on-policy policy evaluation. The idea of preferential updating could also be exploited in other RL settings, such as off-policy learning, control, or policy gradients, to achieve faster learning. Our algorithm could also be applied to learning General Value Functions (Comanici et al., 2018; Sutton et al., 2011), an exciting future direction.

As discussed earlier, our method bridges the gap between Emphatic TD and TD(λ). Eligibility traces in PTD propagate the credit to the current state based on its preference, and the remaining credit goes to the past states. This idea is similar to the gradient updating scheme in the backpropagation through time algorithm, where the *gates* in recurrent neural networks control the flow of credit to past states. We suspect an interesting connection between the eligibility trace mechanism in PTD and backpropagation through time. In the binary preference setting ($\beta \in \{0, 1\}$), our method completely discards zero preference states in the MDP from bootstrapping and updating. The remaining states, whose preference is non-zero, participate in both. This creates a level of *state abstraction* in the given problem.

Our ultimate goal is to train a small agent capable of learning in a large environment, where the agent can't represent values correctly everywhere since the state space is enormous (Silver et al., 2021). A logical step towards this goal is to find ways to effectively use function approximators with limited capacity compared to the environment. Therefore, it should prefer to update and bootstrap from a few states well rather than poorly estimate the values for all states. PTD helps achieve this goal and opens up avenues for developing more algorithms of this flavour.

Acknowledgements

This research was funded through grants from NSERC and the CIFAR Learning in Machines and Brains Program. We thank Ahmed Touati for the helpful discussions on theory; and the anonymous reviewers and several colleagues at Mila for the constructive feedback on the draft.

References

Aberdeen, D. Filtered reinforcement learning. In *European Conference on Machine Learning*, pp. 27–38. Springer, 2004.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31/1, 2017.

Barnard, E. Temporal-difference methods and markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):357–365, 1993.

Bengio, E., Pineau, J., and Precup, D. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pp. 767–777. PMLR, 2020.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-dynamic programming*. Athena Scientific, 1996.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chelu, V., Precup, D., and van Hasselt, H. Forethought and hindsight in credit assignment. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Comanici, G., Precup, D., Barreto, A., Toyama, D. K., Aygün, E., Hamel, P., Vezhnevets, S., Hou, S., and Mourad, S. Knowledge representation for reinforcement learning using general value functions. *Open review*, 2018.

Downey, C., Sanner, S., et al. Temporal difference bayesian model averaging: A bayesian perspective on adapting lambda. In *ICML*, pp. 311–318. Citeseer, 2010.

Ghiassian, S., Rafiee, B., and Sutton, R. S. A first empirical study of emphatic temporal difference learning. *arXiv preprint arXiv:1705.04185*, 2017.

Gordon, G. J. Chattering in SARSA(λ). A CMU learning lab internal report. *Citeseer*, 1996.

Harutyunyan, A., Dabney, W., Mesnard, T., Azar, M. G., Piot, B., Heess, N., van Hasselt, H. P., Wayne, G., Singh, S., Precup, D., et al. Hindsight credit assignment. In *Advances in neural information processing systems*, pp. 12467–12476, 2019.

Hung, C.-C., Lillicrap, T., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):1–12, 2019.

Kearns, M. J. and Singh, S. P. Bias-variance error bounds for temporal difference updates. In *COLT*, pp. 142–147. Citeseer, 2000.

Loch, J. and Singh, S. P. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pp. 323–331, 1998.

Mahmood, A. Incremental off-policy reinforcement learning algorithms. *Ph.D. thesis, U of Alberta*, 2017.

McGovern, A. and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Seijen, H. and Sutton, R. True online td (lambda). In *International Conference on Machine Learning*, pp. 692–700, 2014.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. Reward is enough. *Artificial Intelligence*, pp. 103535, 2021.

Singh, S. P. and Sutton, R. S. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3): 123–158, 1996.

Stolle, M. and Precup, D. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pp. 212–223. Springer, 2002.

Sutton, R. S. Temporal credit assignment in reinforcement learning, phd thesis. *University of Massachusetts, Department of Computer and Information Science*, 1984.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. Td models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*, pp. 531–539. Elsevier, 1995.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

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

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pp. 761–768, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0982657161.

Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17 (1):2603–2631, 2016.

Theocharous, G. and Kaelbling, L. P. Approximate planning in pomdps with macro-actions. In *Advances in Neural Information Processing Systems*, pp. 775–782, 2004.

Thodoroff, P., Anand, N., Caccia, L., Precup, D., and Pineau, J. Recurrent value functions. *4th Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2019.

Tsitsiklis, J. N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

van Hasselt, H. and Sutton, R. S. Learning to predict independent of span. *arXiv preprint arXiv:1508.04582*, 2015.

Varga, R. S. *Matrix iterative analysis*, volume 27. Springer Science & Business Media, 1999.

Watkins, C. J. C. H. Learning from delayed rewards. *King's College, Cambridge*, 1989.

White, M. and White, A. A greedy approach to adapting the trace parameter for temporal difference learning. In

Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '16, pp. 557–565, Richland, SC, 2016. InternationalFoundation for Autonomous Agents and Multiagent Systems. ISBN 9781450342391.

Xu, Z., Modayil, J., van Hasselt, H. P., Barreto, A., Silver, D., and Schaul, T. Natural value approximators: Learning when to trust past estimates. In *Advances in Neural Information processing systems*, pp. 2120–2128, 2017.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In *Advances in neural information processing systems*, pp. 2396–2407, 2018.

Yu, H. On convergence of emphatic temporal-difference learning. In *Conference on Learning Theory*, pp. 1724–1751, 2015.

Yu, H., Mahmood, A. R., and Sutton, R. S. On generalized bellman equations and temporal-difference learning. *The Journal of Machine Learning Research*, 19(1):1864–1912, 2018.

Zhao, M., Luan, S., Porada, I., Chang, X.-W., and Precup, D. Meta-learning state-based eligibility traces for more sample-efficient policy evaluation. In *International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '20. International Foundation for Autonomous Agents and Multiagent Systems, 2020. URL http://www.ifaamas.org/Proceedings/aamas2020/pdfs/p1647.pdf.