Investigating Objectives for Off-policy Value Estimation in Reinforcement Learning

Andrew Patterson AP3@UALBERTA.CA

Sina Ghiassian GHIASSIA@UALBERTA.CA

Adam White AMW8@UALBERTA.CA

Martha White WHITEM@UALBERTA.CA

Department of Computing Science and the Alberta Machine Intelligence Institute (Amii)
University of Alberta
Edmonton, Alberta, Canada

Editor:

Abstract

This paper investigates the problem of online prediction learning, where prediction, action, and learning proceed continuously as the agent interacts with an unknown environment. The predictions made by the agent are contingent on a particular way of behaving—specifying what would happen if the agent behaved in a particular way—represented as a value function. However, the behavior used to select actions and generate the behavior data might be different from the behavior used to define the predictions, and thus the samples are generated off-policy. The ability to learn behavior-contingent predictions online and off-policy has long been advocated as a key capability of predictive-knowledge learning systems, but has remained an open algorithmic challenge for decades. The fundamental issue lies with the temporal difference learning update at the heart of most value-function learning algorithms: combining bootstrapping, off-policy sampling, and fixed-basis function approximation may cause the value estimate to diverge to infinity (e.g., Q-learning with linear function approximation). A major breakthrough came with the development of a new objective function, called the projected Bellman error, that admitted light-weight stochastic gradient descent variants of temporal difference learning. Since then, many sound online off-policy prediction algorithms have been developed, but largely for the linear setting. With this development has come several modifications on the objective itself, and has exposed a fundamental open question in off-policy value estimation: what objective should we use? In this work, we first summarize the large body of literature on off-policy learning, (1) highlighting the similarities in the underlying objectives for algorithms and (2) extracting the key strategies behind many of the algorithms, that can then be used across objectives. We then describe a generalized projected Bellman error, that naturally extends to the nonlinear value estimation setting. We show how this generalized objective unifies previous work, including previous theory. We use this simplified view to derive easy-to-use, but sound, algorithms that we show perform well in both prediction and control.

Contents

Keywords: Off-policy learning, Policy evaluation, Temporal difference learning, Reinforcement learning, Artificial Intelligence.

1. Introduction

To understand the importance of the objective for off-policy value estimation in reinforcement learning, it is useful to first understand the algorithmic development in off-policy learning that led to several of these objectives. We first lay out this development, and then describe the contributions in this work on providing a unified view of the objectives and algorithms, as well as providing theoretical and empirical insights into the properties of these objectives.

1.1 A Short History of Off-policy Temporal Difference Learning

The story of off-policy learning begins with one of the best-known algorithms of reinforcement learning, called Q-learning, and the classic exploration-exploitation tradeoff. Off-policy learning poses an elegant solution to the exploration-exploitation tradeoff: the agent makes use of an independent exploration policy to select actions while learning the value function for the optimal policy. The exploration policy does not maximize reward, but instead selects actions in order to generate data that improves the optimal policy through learning. Ultimately, the full potential of Q-learning—and this ability to learn about one policy from a data generated by a totally different exploration—proved limited. Baird's famous counter-example (Baird, 1995) provided a clear illustration of how, under function approximation, the weights learned by Q-learning can become unstable.[1] Baird's counter-example highlights that divergence can occur when updating off-policy with function approximation and with bootstrapping (as in temporal difference (TD) learning); even when learning the value function of a fixed target policy.

The instability of TD methods is caused by how we correct the updates to the value function to account for the potential mismatch between the target and exploration policies. Off-policy training involves estimating the expected future rewards (the value function) that would be observed while selecting actions according to the target policy with training data (states, actions, and rewards) generated while selecting actions according to an exploration policy. One approach to account for the differences between the data produced by these two policies is based on using importance sampling corrections: scaling the update to the value function based on the agreement between the target and exploration policy at the current state. If the target and exploration policy would select the same action in a state, then they completely agree. Alternatively, if they never take the same action in a state they completely disagree. More generally, there can be degrees of agreement. We call this approach *posterior corrections* because the corrections account for the mismatch between policies ignoring the history of interaction up to the current time step—it does not matter what the exploration policy has done in the past.

Another approach, called *prior corrections*, uses the history of agreement between the exploration and target policy in the update. The likelihood that the trajectory could have occurred under the target policy is used to scale the update. The most extreme version of prior corrections uses the trajectory of experience from the beginning of time, corresponding to what has sometimes been referred to as the *alternative life* framework. Prior and posterior corrections can be combined to achieve stable Off-policy TD updates (Precup et al., 2000), though finite variance of the updates cannot be guaranteed (Precup

1. The action-value star MDP can be found in the errata of Baird's paper (Baird, 1995).

et al., 2001). The perceived variance of these updates, as well as a preference for the excursions framework discussed below, led to a different direction years later for obtaining sound off-policy algorithms (Sutton et al., 2009).

Learning about many different policies in parallel has long been a primary motivation for stable off-policy learning, and this usage suggested that perhaps prior corrections are not essential. Several approaches require learning many value functions or policies in parallel, including approaches based on option models (Sutton et al., 1999), predictive representations of state (Littman and Sutton, 2002; Tanner and Sutton, 2005; Sutton et al., 2011), and auxiliary tasks (Jaderberg et al., 2016). In a parallel learning setting, it is natural to estimate the future reward achieved by following each target policy until termination from the states encountered during training—the value of taking *excursions* from the behavior policy. When value functions or policies estimated off-policy will be used, they will be used starting from states visited by the behavior policy. In such a setting, therefore, it is not necessarily desirable to obtain alternative life solutions.

The first major breakthrough came with the formalization of this excursion model as an objective function, which then enabled development of an online stochastic gradient descent algorithm, called the mean squared projected Bellman error ($\overline{\text{PBE}}$). The resultant family of *Gradient*-TD methods use posterior corrections via importance sampling, and are guaranteed to be stable under function approximation (Sutton et al., 2009). This new excursion objective has the same fixed point as TD, and thus Gradient-TD methods converge to the same solution in the cases for which TD converges. Prior attempts to create an objective function for off-policy learning, namely the mean squared Bellman error due to Baird (1995), resulted in algorithms that converge to different and sometimes less desirable fixed points (see Sutton & Barto, 2018 for an in depth discussion of these issues). The Gradient-TD methods have extensions for incorporating eligibility traces (Maei, 2011), non-linear function approximation such as with a neural network (Maei, 2011), and learning optimal policies (Maei, 2011). Although guaranteed stable, the major critiques of these methods are (1) the additional complexity due to a second set of learned parameters, and (2) the variance due to importance sampling corrections.

The second major family of off-policy methods revisits the idea of using prior corrections. The idea is to incorporate prior corrections, starting only from the beginning of the excursion. In this way, the values of states that are more often visited under the target policy are emphasized, but the high variance of full prior corrections—to the beginning of the episode—is avoided. An incremental algorithm, called *Emphatic* TD(λ), was developed to estimates these emphasis weightings (Sutton et al., 2016), with a later extension to further improve variance of the emphasis weights (Hallak et al., 2016). These Emphatic-TD methods are guaranteed stable under both on-policy and off-policy sampling with linear function approximation (Sutton et al., 2016; Yu, 2015; Hallak et al., 2016).

Since the introduction of these methods, several refinements have been introduced, largely towards improving sample efficiency. These include (1) *Hybrid*-TD methods that behave like TD when sampling is on-policy, (2) *Saddlepoint* methods for facilitating application of improved stochastic approximation algorithms and (3) variance reduction methods for posterior corrections, using different eligibility trace parameters. The Hybrid TD methods can be derived with a simple substitution in the gradient of the excursion objective. The resultant algorithms perform conventional TD updates when data is generated on-policy

(Hackman, 2013; White and White, 2016), and are stable under off-policy sampling. Initial empirical studies suggested that TD achieves better sample efficiency than Gradient-TD methods when the data is sampled on-policy, though later evaluations found little difference (White and White, 2016).

Another potential improvement on Gradient-TD can be derived by reformulating the excursion objective into a saddlepoint problem, resulting in several new methods (Liu et al., 2016; Dai et al., 2017; Du et al., 2017; Touati et al., 2018; Liu et al., 2020). This saddlepoint formulation enables use of optimization strategies for saddlepoint problems, including finite sample analysis (Touati et al., 2018), accelerations (Liu et al., 2020; Du et al., 2017) and even generalizations to use (approximations to) the mean squared Bellman error ($\overline{\text{BE}}$) (Dai et al., 2017; Feng et al., 2019). Though most are applicable to online updating, some acceleration strategies are restricted to offline batch updating (Du et al., 2017). As with the hybrid methods, comparative studies to date remain inconclusive about the advantages of these methods over their vanilla Gradient-TD counterparts (Mahadevan et al., 2014; White and White, 2016).

Finally, several algorithms have been proposed to mitigate variance from importance sampling ratios in the posterior corrections. High magnitude importance sampling corrections introduce variance and slow learning, dramatically reducing the benefits of off-policy learning. In parallel learning frameworks with many target policies, the likelihood of large importance sampling corrections increases as the number of target policies increases. In practice one might use small stepsizes, or avoid eligibility traces to mitigate the variance. The Retrace algorithm solves this issue by truncating the importance sampling ratio and a bias correction, thus avoiding large updates when the exploration and the target policy differ significantly. This approach can diverge with function approximation (Touati et al., 2018). Nevertheless, Retrace has been used in several deep-learning systems with non-linear function approximation (Munos et al., 2016; Wang et al., 2016). The Tree Backup algorithm (Precup et al., 2000) mitigates variance without importance sampling corrections by only using the probability of the selected action under the target policy. Both Retrace and Tree Backup can be viewed as adapting the eligibility trace to reduce variance (see Section **??**). The related ABQ algorithm achieves stable off-policy updates without importance sampling corrections by varying the amount of bootstrapping in an action-dependent manner (Mahmood et al., 2017). These developments are complementary to the GTD and ETD approaches, as the focus is variance and they still require gradient methods or reweighting to be sound.

1.2 The Role of the Objective

Nestled within the development of these algorithms is an important choice: the weighting on states in the objective. The importance of the weighting on states has been well-recognized for many years, and is in fact the reason TD diverges on Baird's counterexample: the weighting on states using the stationary distribution of the behavior policy, rather than the target policy, results in an iterative update that is no longer a contraction. The emphatic algorithms were introduced to adjust this weighting to ensure convergence.

This reweighting, however, is not only about convergence; it also changes the fixed point and the quality of the solution. There has been some work investigating the impact of the state weighting on the optimal solution, not just on the behavior of the updates themselves.

The most stark result is a simple example where using the solution to the $\overline{\text{PBE}}$ can result in an arbitrarily poor mean squared value error ($\overline{\text{VE}}$), under certain state weightings (Kolter, 2011). Several later results extended on-policy bounds on the $\overline{\text{VE}}$, to the off-policy setting, showing that $\overline{\text{VE}}$ could be bounded in the off-policy setting using careful choices on the state weighting—namely using state weightings given by Emphatic TD (Hallak et al., 2016; White, 2017). Despite these insights, the role of the weighting on the quality of the solution in practice remains open. A natural question is how to choose the state weighting in the objective, and how much it matters.

Another important question is the form of the objective itself, and the long-open question about using the $\overline{\text{BE}}$ or the $\overline{\text{PBE}}$. Using the $\overline{\text{BE}}$ avoids the poor counterexamples that exist for the $\overline{\text{PBE}}$ (Scherrer, 2010), but nonetheless in some cases the $\overline{\text{PBE}}$ produces a better solution (Scherrer, 2010; Sutton and Barto, 2018). Further, the $\overline{\text{BE}}$ has been shown to have an identifiability problem (Sutton and Barto, 2018). Though the evidence comparing the $\overline{\text{BE}}$ and $\overline{\text{PBE}}$ is inconclusive, the $\overline{\text{PBE}}$ has been the default choice because we have many algorithms to optimize it. The $\overline{\text{BE}}$, on the other hand, is typically avoided due to the double sampling problem, where it is unclear how to obtain an unbiased sample of the gradient without a simulator.

Recently, however, this technical challenge has been overcome with the introduction of a saddlepoint form for the $\overline{\text{BE}}$ (Dai et al., 2017). The resulting algorithms are similar to the saddlepoint algorithms for the linear $\overline{\text{PBE}}$: a second estimator is used to estimate a part of the gradient. The $\overline{\text{BE}}$ is particularly alluring, as it equally applies to the linear and nonlinear value estimation settings. The $\overline{\text{PBE}}$, on the other hand, was defined for the linear setting, due to the difficulty in computing the projection operator for the nonlinear setting. The one work attempting to extend it to the nonlinear setting used a local linear projection that results in the need to compute gradients through gradients, and so a more complex algorithm (Maei et al., 2009). These potential advantages, as well as a viable strategy for optimizing the $\overline{\text{BE}}$, motivates the utility of answering which of these objectives might be preferable.

1.3 Contributions

In this work, we bring clarity to the question: what objective should we use for off-policy value estimation? We first summarize many existing off-policy algorithms, as optimizing the linear $\overline{\text{PBE}}$ in different ways and in some cases with different state weightings. This summary separates the optimization strategy from the definition of the objective, allowing us to move away from specifically which algorithm is to understanding the differences in fixed points obtained under the different objectives. We then propose a generalized $\overline{\text{PBE}}$, that uses a generalized projection operator that both extends the $\overline{\text{PBE}}$ to the nonlinear setting and unifies the $\overline{\text{BE}}$ and linear $\overline{\text{PBE}}$ under one objective. Using these insights, we provide the following contributions.

1. We show how this helps resolve the non-identifiability of the $\overline{\text{BE}}$, where a particular projection in the generalized $\overline{\text{PBE}}$ provides an Identifiable $\overline{\text{BE}}$.

2. We highlight the role of the state weighting in this generalized objective, both extending theoretical results bounding the the $\overline{\text{VE}}$ and empirically showing that the emphatic

weighting can significantly improve the quality of the solution. We show this for three variants of the generalized $\overline{\text{PBE}}$: the $\overline{\text{BE}}$ (no projection), the linear $\overline{\text{PBE}}$ (projection space equal to the value function space) and a generalized $\overline{\text{PBE}}$ in-between (projection space larger than the value function space).

3. We show that these insights also extend to control, by defining an objective for learning action values with (soft) maximal operators. We use this objective to derive a sound gradient variant of Q-learning.

4. We exploit the connection to the linear $\overline{\text{PBE}}$, to develop a more effective algorithm for the generalized $\overline{\text{PBE}}$ based on using gradient-corrections rather than the saddlepoint update.

5. Finally, we demonstrate the utility of these prediction and control algorithms in several small benchmark problems.

2. Problem Definition and Background

We consider the problem of learning the value function for a given policy under the Markov Decision Process (MDP) formalism. The agent interacts with the environment over a sequence of discrete time steps, $t = 1, 2, 3, \ldots$. On each time step the agent observes a partial summary of the state $S_t \in \mathcal{S}$ and selects an action $A_t \in \mathcal{A}$. In response, the environment transitions to a new state S_{t+1}, according to transition function $P(S_{t+1}|S_t, A_t)$, and emits a scalar reward $R_{t+1} \in \mathcal{R}$. The agent selects actions according to a stochastic, stationary *target policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

We study the problem of *policy evaluation*: the computation or estimation of the expected discounted sum of future rewards for policy π from every state. The *return* at time t, denoted $G_t \in \mathbb{R}$, is defined as the discounted sum of future rewards. The discount factor can be variable, dependent on the entire transition: $\gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, with $\gamma_{t+1} \overset{\text{def}}{=} \gamma(S_t, A_t, S_{t+1})$. The return is defined as

$$\begin{aligned} G_t &\overset{\text{def}}{=} R_{t+1} + \gamma_{t+1} R_{t+2} + \gamma_{t+1}\gamma_{t+2} R_{t+3} + \gamma_{t+1}\gamma_{t+2}\gamma_{t+3} R_{t+4} + \ldots \\ &= R_{t+1} + \gamma_{t+1} G_{t+1}. \end{aligned}$$

When γ_t is constant, we get the familiar return $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$, where we overload γ here to indicate a scalar, constant discount. Otherwise, variable γ_t can discount per transition, including encoding termination when it is set to zero. This generalization ensures we can discuss episodic problems without introducing absorbing states (White, 2017). It also enables the derivations and theory to apply to both the continuing and episodic settings. The *value function* $v : \mathcal{S} \to \mathbb{R}$ maps each state to the expected return under policy π starting from that state

$$v_\pi(s) \overset{\text{def}}{=} \mathbb{E}_\pi[G_t \mid S_t = s] \ \text{, for all } s \in \mathcal{S} \tag{1}$$

where the expectation operator $\mathbb{E}_\pi[\cdot]$ reflects that the distribution over future actions is given by π, to distinguish from a potentially different behavior policy. The distribution over future states and rewards is always given by the transition dynamics of the MDP, and so is not explicitly listed as a subscript.

In this paper, we are interested in problems where the value of each state cannot be stored in a table; instead the agent must approximate the value with a parameterized function. The approximate value function $\hat{v}(s_t, \boldsymbol{w})$ can have arbitrary form, as long as it is everywhere differentiable with respect to the weights $\boldsymbol{w} \in \mathbb{R}^d$. Typically the number of components in $\boldsymbol{w}$ is much less than the number of possible states ($d \ll |\mathcal{S}|$), and thus $\hat{v}$ will generalize values across many states in $\mathcal{S}$. An important special case is when the approximate value function is linear in the parameters and in features of the state. In particular, the current state S_t is converted into feature vector $\boldsymbol{x}_t \in \mathbb{R}^d$ by some fixed mapping $\boldsymbol{x} : \mathcal{S} \to \mathbb{R}^d$. The value of the state can then be approximated with an inner product:

$$\hat{v}(s_t, \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}_t \approx v_\pi(s_t), \qquad \text{for all } s_t \in \mathcal{S}.$$

Another typical parameterization for $\hat{v}(s_t, \boldsymbol{w})$ is a neural network, where $\boldsymbol{w}$ consists of all the weights in the network. Henceforth, we refer to $\boldsymbol{w}$ exclusively as the *weights*, or weight vector, and reserve the word *parameter* for variables like the discount-rate and stepsize parameters.

We first describe how to learn this value function for the on-policy setting, where the behavior policy equals the target policy. Temporal difference learning (Sutton, 1988) is perhaps the best known and most successful approach for estimating $\hat{v}$ directly from samples generated while interacting with the environment. Instead of waiting until the end of a trajectory to update the value of each state, the TD(λ) algorithm adjusts its current estimate of the weights toward the difference between the discounted estimate of the value in the next state and the estimated value of the current state plus the reward along the way:

$$\delta_t \overset{\text{def}}{=} \delta(S_t, A_t, S_{t+1}) \overset{\text{def}}{=} R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w}). \tag{2}$$

We use the value function's own estimate of future reward as a placeholder for the future rewards defining G_t that are not available on time-step $t+1$. In addition, the TD(λ) algorithm also maintains an eligibility trace vector $\boldsymbol{z}_t \in \mathbb{R}^d$ that stores a fading trace of recent feature activations. The components of $\boldsymbol{w}_t$ are updated on each step proportional to the magnitude of the trace vector. This simple scheme allows update information to more quickly propagate in domains when the rewards are often zero, such as a maze with a reward of one upon entering the terminal state and zero otherwise.

The update equations for TD(λ) are straightforward:

$$\begin{aligned} \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha \delta_t z_t \\ \boldsymbol{z}_t &\leftarrow \gamma\lambda \boldsymbol{z}_{t-1} + \nabla \hat{v}(S_t, \boldsymbol{w}), \end{aligned}$$

where $\alpha \in \mathbb{R}$ is the scalar stepsize parameter that controls the speed of learning, and $\lambda \in \mathbb{R}$ controls the length of the eligibility trace. If λ is one, then the above algorithm performs an incremental version of Monte-Carlo policy evaluation. On the other-hand, when λ is zero the TD(λ) algorithm updates the value of each state using only the reward and the estimated value of the next state—often referred to as full one-step bootstrapping. TD(0) is arguably the most common implementation, especially with neural networks, with update

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \delta_t \nabla \hat{v}(S_t, \boldsymbol{w})$$

Under linear function approximation, intermediate values of λ between zero and one often perform best. The TD(λ) algorithm has been shown to converge with probability one to the best linear approximation of the value function under quite general conditions, for the on-policy setting.

TD(λ) is only sound for the linear function approximation setting. The updates for the linear setting use $\nabla\hat{v}(S_t, \boldsymbol{w}) = \boldsymbol{x}_t$, in either the eligibility trace or the TD(0) update. Nonetheless, TD is often used outside the linear setting, and often obtains good performance. Therefore, we consider this more general form for TD(λ), under nonlinear function approximation.

These updates need to be modified for the *off-policy case*, where the agent selects actions according to a *behavior policy* $b : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that is different from the target policy. The value function for target policy π is updated using experience generated from a behavior policy that is *off*, away, or distant from the target policy. For example, consider the most well-known off-policy algorithm, Q-learning. The target policy might be the one that maximizes future discounted reward, while the behavior is nearly identical to the target policy, but instead selects an exploratory action with some small probability. More generally, the target and behavior policies need not be so closely coupled. The target policy might be the shortest path to one or more goal states in a gridworld, and the behavior policy might select actions in each state uniform randomly. The main requirement linking these two policies is that the behavior policy *covers* the actions selected by the target policy in each state visited by b, that is: $b(a|s) > 0$ for all states and actions in which $\pi(a|s) > 0$.

An important difference between these two settings is in the stability and convergence of the algorithms. One of the most distinctive aspects of off-policy learning and function approximation is that it has been shown that Q-learning and TD(λ), appropriately modified for off-policy updates, and even Dynamic Programming can diverge (Sutton and Barto, 2018). In the next two sections, we will discuss different ways to adapt TD-style algorithms to the off-policy setting. We will highlight convergence issues and issues with solution quality, and discuss different ways recent algorithms proposed to address these issues.

3. Off-policy Corrections and the Connection to State Weightings

The key problem in off-policy learning is to estimate the value function for the target policy, conditioned on samples produced by actions selected according to the behavior policy. This is an instance of the problem of estimating an expected value under some target distribution from samples generated by some other behavior distribution. In statistics, we address this problem with importance sampling, and indeed most methods of off-policy reinforcement learning use such corrections.

We can account for the differences between which actions the target policy would choose in each state, and we can account for which states are more likely to be visited under the target policy. More precisely, there are two distributions that we could consider correcting: the distribution over actions, given the state, and the distribution over states. When observing a transition (S, A, S', R) generated by taking the action according to $b(\cdot|S)$, we can correct the update for that transition so that in expectation it is as if actions were taken according to $\pi(\cdot|S)$. However, these updates are still different than if we evaluated π on-policy, because the frequency of visiting state S under b will be different than under π.

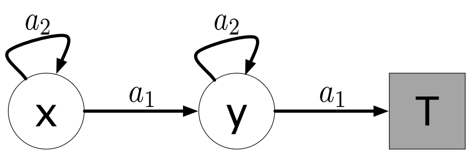

Figure 1: A simple MDP to understand the differences between prior corrections and posterior corrections in Off-policy TD algorithms with importance sampling.

All methods correct for the distribution over actions (posterior corrections), given the state, but several methods correct for the distribution over states (prior corrections) in slightly different ways.

In this section, we first provide an intuitive explanation of the differences between methods that use only posterior correction and those that additionally incorporate prior corrections. We then discuss the optimization objective used by Off-policy TD methods, and highlight how the use of prior corrections corresponds to different weightings in this objective. We discuss the importance of this weighting in the following section. This generic objective will also allow us to easily describe the differences between key off-policy algorithms. We focus first on the linear $\overline{\text{PBE}}$, and extend insights in the next section.

3.1 Posterior Corrections

The most common approach to developing sound Off-policy TD algorithms makes use of posterior corrections based on importance sampling. One of the simplest examples of this approach is *Off-policy TD(λ)*. The procedure is easy to implement and requires constant computation per time step, given knowledge of both the target and behavior policies. On the transition from S_t to S_{t+1} via action A_t, we compute the ratio between π and b:

$$\rho_t \overset{\text{def}}{=} \rho(A_t|S_t) \overset{\text{def}}{=} \frac{\pi(A_t|S_t)}{b(A_t|S_t)}. \tag{3}$$

These importance sampling corrections are then simply added to the eligibility trace update on each time step:

$$\begin{aligned} \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\delta_t \boldsymbol{z}_t^\rho \\ \boldsymbol{z}_t^\rho &\leftarrow \rho_t(\gamma\lambda\boldsymbol{z}_{t-1}^\rho + \boldsymbol{x}_t), \end{aligned} \tag{4}$$

where δ_t is defined in Equation 2. This way of correcting the sample updates ensures that the approximate value function $\hat{v}$ estimates the expected value of the return as if the actions were selected according to π. Posterior correction methods use the target policy probabilities for the selected action to correct the update to the value of state S_t using only the data from time step t onward. Values of π from time steps prior to t have no impact on the correction. Combining importance sampling with eligibility trace updates, as in Off-policy TD(λ), is the most common realization of posterior corrections.

To help understand the implications of posterior corrections, consider the MDP depicted in Figure 1. Each episode starts in the leftmost state denoted ‘x’ and terminates on transition into the terminal state denoted with ‘T’, and each state is represented with a unique tabular state encoding: x: $[1, 0]$, y: $[0, 1]$. In each state there are two possible actions and the behavior policy chooses each action in each state with 0.5 probability. The target policy chooses action a_1 in all states. A posterior correction method like Off-policy TD(λ), will always update the value of a state if action a_1 is taken. For example if the agent experiences the transition $y \to T$, Off-policy TD(λ) will update the value of state y; no matter the history of interaction before entering state y.

Although the importance sampling corrections product in the eligibility trace update, Off-policy TD(λ) does not use importance sampling corrections computed from prior time-steps to update the value of the current state. This is easy to see with an example. For simplicity we assume γ_t is a constant $\gamma \in [0, 1)$. Let's examine the updates and trace contents for a trajectory where b's action choices perfectly agree with π:

$$x \to y \to T.$$

After the transition from $x \to y$, Off-policy TD(λ) will update the value estimate corresponding to x:

$$\begin{bmatrix} \hat{v}_1(x) \\ \hat{v}_1(y) \end{bmatrix} \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \alpha\delta_1 \boldsymbol{z}_1^\rho = \alpha\delta_1 \begin{bmatrix} \frac{\pi(a_1|x)}{b(a_1|x)}\gamma\lambda \\ 0 \end{bmatrix},$$

where $\hat{v}_1(x)$ denotes the estimated value of state x on time step $t = 1$ (after the first transition), and as usual $\boldsymbol{z}_0^\rho$ and $\hat{v}$ are initialized to zero. After the second transition, $y \to T$, the importance sampling corrections will product in the trace, and the value estimates corresponding to both x and y are updated:

$$\begin{bmatrix} \hat{v}_2(x) \\ \hat{v}_2(y) \end{bmatrix} \leftarrow \begin{bmatrix} \hat{v}_1(x) \\ \hat{v}_1(y) \end{bmatrix} + \alpha\delta_2 \begin{bmatrix} \frac{\pi(a_1|y)}{b(a_1|y)}\frac{\pi(a_1|x)}{b(a_1|x)}\gamma^2\lambda^2 \\ \frac{\pi(a_1|y)}{b(a_1|y)}\gamma\lambda \end{bmatrix}.$$

The estimated value of state y is only updated with importance sampling corrections computed from state transitions that occur after the visit to y: using $\frac{\pi(a_1|y)}{b(a_1|y)}$, but not $\frac{\pi(a_1|x)}{b(a_1|x)}$.

Finally, consider another trajectory that deviates from the target policy's choice on the second step of the trajectory:

$$x \to y \to y \to T.$$

On the first transition the value of x is updated as expected, and no update occurs as a result of the second transition. On the third, transition the estimated value of state x is *not* updated; which is easy to see from inspecting the eligibility trace on each time-step:

$$\boldsymbol{z}_1^\rho = \begin{bmatrix} \frac{\pi(a_1|x)}{b(a_1|x)}\gamma\lambda \\ 0 \end{bmatrix};\ \boldsymbol{z}_2^\rho = \boldsymbol{0};\ \boldsymbol{z}_3^\rho = \begin{bmatrix} 0 \\ \frac{\pi(a_1|y)}{b(a_1|y)}\gamma\lambda \end{bmatrix}.$$

The eligibility trace is set to zero on time step two, because the target policy never chooses action a_2 in state y and thus $\frac{\pi(a_2|y)}{b(a_2|y)} = 0$. The value of state S_t is never updated using importance sampling corrections computed on time steps prior to t.

Many modern off-policy prediction methods use some form of posterior corrections including the Gradient-TD methods, Tree Backup(λ), V-trace(λ), and Emphatic TD(λ). In fact, all off-policy prediction methods with stability guarantees make use of posterior corrections via importance sampling. Only correcting the action distribution, however, does not necessarily provide stable updates, and Off-policy TD(λ) is not guaranteed to converge (Baird, 1995). To obtain stable Off-policy TD(λ) updates, we need to consider corrections to the state distribution as well, as we discuss next.

3.2 Prior Corrections

We can also consider correcting for the differences between the target and behavior policy by using the agreement between the two over a trajectory of experience. *Prior correction* methods keep track of the product of either $\prod_{k=1}^{t} \pi(A_k|S_k)$ or $\prod_{k=1}^{t} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$, and correct the update to the value of S_t using the current value of the product. Therefore, the value of S_t is only updated if the product is not zero, meaning that the behavior policy never selected an action for which $\pi(A_k|S_k)$ was zero—the behavior never completely deviated from the target policy.

To appreciate the consequences of incorporating these prior corrections into the TD update consider a state-value variant of Precup et al. (2000) Off-policy TD(λ) algorithm:

$$\begin{aligned} \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha \delta_t \boldsymbol{z}_t^{\rho} \\ \boldsymbol{z}_t^{\rho} &\leftarrow \rho_t \left(\gamma \lambda \boldsymbol{z}_{t-1} + \prod_{k=1}^{t-1} \rho_k \boldsymbol{x}_t \right) \end{aligned} \tag{5}$$

where $\boldsymbol{z}_0^{\rho} = \mathbf{0}$. We will refer to the above algorithm as *Alternative-life TD(λ)*. The product in Equation 5 includes all the ρ_t observed during the current episode. Note that experience from prior episodes does not impact the computation of the eligibility trace, as the trace is always reinitialized at the start of the episode.

Now consider the updates performed by Alternative-life TD(λ) using different trajectories from our simple MDP (Figure 1). If the agent ever selects action a_2, then none of the following transitions will result in further updates to the value function. For example, the trajectory $x \to y \to y \to y \cdots y \to T$ will update $\hat{v}(s)$ corresponding to the first $x \to y$ transition, but $\hat{v}(y)$ would never be updated due to the product in Equation 5. In contrast, the Off-policy TD(λ) algorithm described in Equation 4 would update $\hat{v}(s)$ on the first transition, and also update $\hat{v}(y)$ on the last transition of the trajectory.

The Alternative-life TD(λ) algorithm has been shown to converge under linear function approximation, but in practice exhibits unacceptable variance (Precup et al., 2001). The Emphatic TD(λ) algorithm, on the other hand, provides an alternative form for the prior corrections, that is lower variance but still guarantees convergence. To more clearly explain why, next we will discuss how different prior corrections account for different weightings in optimizing the mean squared Projected Bellman Error ($\overline{\text{PBE}}$).

3.3 The Linear $\overline{\text{PBE}}$ under Posterior and Prior Corrections

In this section, we describe how different prior corrections, or no prior corrections, correspond to optimizing similar objectives, but with different weightings over the state. This section introduces the notation required to explain the many algorithms that optimize the linear $\overline{\text{PBE}}$, and clarifies convergence properties of algorithms, including which algorithms converge and to which fixed point. We start with only the linear $\overline{\text{PBE}}$, because most algorithms have been designed to optimize it and it is worthwhile understanding this special case. In the next section, we extend beyond the linear setting, to discuss the generalized $\overline{\text{PBE}}$.

We begin by considering a simplified setting, with $\lambda = 0$, and a simplified variant of the linear $\overline{\text{PBE}}$, called the NEU (norm of the expected TD update (Sutton et al., 2009))

$$\text{NEU}(\boldsymbol{w}) = \Big\| \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi\big[\delta(S, A, S')\boldsymbol{x}(S) \mid S = s\big] \Big\|_2^2, \tag{6}$$

where $d : \mathcal{S} \to [0, \infty)$ is a positive weighting on the states, and we explicitly write $\delta(S, A, S')$ to emphasize that randomness in the TD-error is due to the underlying randomness in the transition (S, A, S'). Equation 6 does not commit to a particular sampling strategy. If the data is sampled on-policy, then $d = d_\pi$, where $d_\pi : \mathcal{S} \to [0, 1]$ is the stationary distribution for π which represents the state visitation frequency under behavior π in the MDP. If the data is sampled off-policy, then the objective is instead weighted by the state visitation frequency under b, i.e., $d = d_b$.

We first consider how to sample the NEU for a given a state. The behavior selects actions in each state s, so the update $\delta_t \boldsymbol{x}_t$ needs to be corrected for the action selection probabilities of π in state s. Importance sampling is one way to correct these action probabilities from a given state $S_t = s$

$$\begin{aligned}
\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s] &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a)\delta(s, a, s')\boldsymbol{x}(s) \\
&= \sum_{a \in \mathcal{A}} \frac{b(a|s)}{b(a|s)} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a)\delta(s, a, s')\boldsymbol{x}(s) \\
&= \sum_{a \in \mathcal{A}} b(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a) \frac{\pi(a|s)}{b(a|s)} \delta(s, a, s')\boldsymbol{x}(s) \\
&= \mathbb{E}_b[\rho(A_t|S_t)\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s] \, .
\end{aligned} \tag{7}$$

Therefore, the update $\rho_t \delta_t \boldsymbol{x}_t$ provides an unbiased sample of the desired expected update $\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s]$. All off-policy methods use these posterior corrections.

We can also adjust the state probabilities from d_b to d_π, using prior corrections. Alternative-life TD(λ) uses such prior corrections to ask: what would the value be if the data had been generated according to π instead of b. In such a scenario, the state visitation would be according to d_π, and so we need to correct both action probabilities in the updates as well as the distribution from which we update. Prior corrections adjust the likelihood of reaching a state. Consider the expectation using prior corrections, when starting in state s_0 and taking two steps following b:

$$\begin{aligned}
&\mathbb{E}_b[\rho_0\rho_1\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = S_2] \mid S_0 = s_0] \\
&= \mathbb{E}_b\left[\rho_0 \sum_{a_1\in\mathcal{A}} b(a_1|S_1) \sum_{s_2\in\mathcal{S}} P(s_2|S_1, a_1)\rho(a_1|S_1)\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s_2] \mid S_0 = s_0\right] \\
&= \mathbb{E}_b\left[\rho_0 \sum_{a_1\in\mathcal{A}} \pi(a_1|S_1)P(s_1|S_1, a_1)\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s_2] \mid S_0 = s_0\right] \\
&= \mathbb{E}_b[\rho_0\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_{t-1} = S_1] \mid S_0 = s_0] \\
&= \sum_{a_0\in\mathcal{A}} \pi(s_0, a_0) \sum_{s_1\in\mathcal{S}} P(s_1|s_0, a_0)\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_{t-1} = s_1] \\
&= \mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_0 = s_0]\,.
\end{aligned}$$

More generally, we get

$$\mathbb{E}_b\left[\rho_1 \dots \rho_{t-1}\mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_t = s] \,|S_0 = s_0\right] = \mathbb{E}_\pi[\delta(S_t, A_t, S_{t+1})\boldsymbol{x}(S_t) \mid S_0 = s_0]\,.$$

These corrections adjust the probabilities of the sequence from the beginning of the episode to make it as if policy π had taken actions $A_1, \dots, A_{t-1}$ to get to state S_t, from which we do the TD(λ) update.

A natural question is which objective should be preferred: the alternative-life ($d \propto d_\pi$) or the excursions objective ($d \propto d_b$). As with all choices for objectives, there is not an obvious answer. The alternative-life objective is difficult to optimize, because prior corrections can become very large or zero—causing data to be discarded—and is high variance. On the other hand, the fixed-point solution to the excursion objective can be arbitrarily poor compared with the best value function in the function approximation class if there is a significant mismatch between the behavior and target policy (Kolter, 2011). Better solution accuracy can be achieved using an excursion's weighting that includes d_b, but additionally reweights to make the states distribution closer to d_π, as is done with Emphatic TD(λ). We discuss this alternative weighting in the next section.

The above discussion focused on a simplified variant of the $\overline{\text{PBE}}$ with $\lambda = 0$, but the intuition is the same for the $\overline{\text{PBE}}$ and $\lambda > 0$. To simplify notation we introduce a conditional expectation operator:

$$\mathbb{E}_d[Y] = \sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi[Y \mid S = s].$$

We can now define

$$\begin{aligned}
\mathbf{C} &\stackrel{\text{def}}{=} \mathbb{E}_d[\boldsymbol{x}(S)\boldsymbol{x}(S)^\top] \\
\mathbf{A} &\stackrel{\text{def}}{=} -\mathbb{E}_d[(\gamma(S')\boldsymbol{x}(S') - \boldsymbol{x}(S))\boldsymbol{z}(S)^\top] \\
\boldsymbol{b} &\stackrel{\text{def}}{=} \mathbb{E}_d[R(S, A, S')\boldsymbol{z}(S)^\top]
\end{aligned}$$

where the expected eligibility trace $\boldsymbol{z}(S) \in \mathbb{R}^k$ is defined recursively $\boldsymbol{z}(S) \stackrel{\text{def}}{=} \boldsymbol{x}(S) + \gamma(S)\lambda\mathbb{E}_\pi[\boldsymbol{z}(S_{t-1})|S_t = S]$. We can write the TD($\lambda$) fixed point residual as:

$$\mathbb{E}_d[\delta(S, A, S')\boldsymbol{z}(S)] = -\mathbf{A}\boldsymbol{w} + \boldsymbol{b} \tag{8}$$

so called because $\mathbb{E}_{d_\pi}[\delta(S, A, S')\boldsymbol{z}(S)] = \mathbf{0}$ at the fixed point solution for on-policy TD(λ). The linear $\overline{\text{PBE}}$ can be defined simply, given the definition above:

$$\text{linear } \overline{\text{PBE}}(\boldsymbol{w}) \overset{\text{def}}{=} (-\mathbf{A}\boldsymbol{w} + \boldsymbol{b})^\top \mathbf{C}^{-1}(-\mathbf{A}\boldsymbol{w} + \boldsymbol{b}). \tag{9}$$

The only difference compared with the NEU is the weighted ℓ_2 norm, weighted by $\mathbf{C}^{-1}$, instead of simply $\|-\mathbf{A}\boldsymbol{w} + \boldsymbol{b}\|_2^2$. The extension to $\lambda > 0$ requires that posterior corrections also correct future actions from the state S, resulting in a product of importance sampling ratios in the eligibility trace, as described in the previous section. The conclusions about the choice of state probabilities d in defining the objective, however, remain consistent.

3.4 Emphatic Weightings as Prior Corrections

Emphatic Temporal Difference learning, ETD(λ), provides an alternative strategy for obtaining stability under off-policy sampling without computing gradients of the linear $\overline{\text{PBE}}$. The key idea is to incorporate some prior corrections so that the weighting d results in a positive definite matrix $\mathbf{A}$. Given such an $\mathbf{A}$, a TD(λ) algorithm—a semi-gradient algorithm—can be shown to converge. Importantly, this allows for a stable off-policy algorithm with only a single set of weights. Gradient-TD methods, on the other hand, use two stepsize parameters and two weight vectors to achieve stability.

ETD(λ) minimizes a variant of the linear $\overline{\text{PBE}}$ defined in Equation 9, where the weighting d is defined based on the *followon* weighting. The followon reflects (discounted) state visitation under the target policy when doing excursions from the behavior: starting from states sampled according to d_b. The followon is defined as

$$f(s_t) \overset{\text{def}}{=} d_b(s_t) + \gamma(s_t) \sum_{s_{t-1}, a_{t-1}} d_b(s_{t-1})\pi(a_{t-1}|s_{t-1})P(s_t|s_{t-1}, a_{t-1}) + \ldots. \tag{10}$$

The emphatic weighting then corresponds to $m(s_t) = d_b(s_t)\lambda + (1-\lambda)f(s_t)$. This is the weighting used in the linear $\overline{\text{PBE}}$ in Equation 9, setting $d(s) = m(s)$.

The Emphatic TD(λ) algorithm is specified by the following equations:

$$\begin{aligned}
F_t &\leftarrow \rho_{t-1}\gamma_t F_{t-1} + 1\\
M_t &\leftarrow \lambda_t + (1-\lambda_t)F_t\\
\boldsymbol{z}_t^\rho &\leftarrow \rho_t\left(\gamma_t\lambda\boldsymbol{z}_{t-1}^\rho + M_t\boldsymbol{x}_t\right)\\
\boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\delta_t\boldsymbol{z}_t^\rho,
\end{aligned}$$

with $F_0 = 1$ and $\boldsymbol{z}_0^\rho = \mathbf{0}$. The scalar estimate F_t is used to include the weighting defined in Equation 43. To gain some intuition for this weighting, consider a setting where $\gamma_t = \gamma$ is constant and $\lambda = 0$. Then $M_t = F_t = \sum_{j=0}^t \gamma^j \prod_{i=1}^j \rho_{t-i}$, giving trace $\boldsymbol{z}_t^\rho \leftarrow \rho_t\left(\gamma_t\lambda\boldsymbol{z}_{t-1}^\rho + \sum_{j=0}^t \gamma^j \prod_{i=1}^j \rho_{t-i}\boldsymbol{x}_t\right)$.

There are some similarities to the weighting in the Alternative-life TD(λ) trace in Equation 5, where $\boldsymbol{z}_t^\rho \leftarrow \rho_t\left(\gamma_t\lambda\boldsymbol{z}_{t-1}^\rho + \prod_{i=1}^t \rho_i\boldsymbol{x}_t\right)$. Both adjust the weighting on $\boldsymbol{x}_t$ to correct for—or adjust—the state distributions. Alternative-Life TD more aggressively downweights states that would not have been visited under the target policy. ETD, on the other hand, reweights based on how frequently the states would be seen when starting π as an excursion from b.

Emphatic TD(λ) has strong convergence guarantees in the case of linear function approximation. The ETD(λ) under off-policy training has been shown to converge in expectation using the same expected update analysis used to show that TD(λ) converges under on-policy training. Later, Yu (2015) extended this result to show that ETD(λ) converges with probability one. Perhaps more practically relevant, this weighting also resolves the issues raised by Kolter's example (Kolter, 2011). Kolter's example demonstrated that for a particular choice of π and b, the solution to the linear $\overline{\text{PBE}}$ could result in arbitrarily bad error compared with the best possible approximation in the function class. In other words, even if the true value function can be well approximated by the function class, the off-policy fixed point from the linear $\overline{\text{PBE}}$ with weighting $d = d_b$ can result in an arbitrarily poor approximation to the values. In Section 6, we explain that the fixed points of the linear $\overline{\text{PBE}}$ with the emphatic weighting, on the other hand, do not suffer from this problem, expanding on insights from (Hallak et al., 2016, Corollary 1) and (White, 2017, Theorem 1).

3.5 Broadening the Scope of Weightings

The choice of d is generic and we can consider many different weightings. The weightings d_π and d_b were implicitly chosen because they correspond to the data gathering policies. The emphatic weighting m is feasible to compute online, and resolves divergence issues. Other weightings might not be as feasible to compute, but there is a clear opportunity to broaden the scope of weightings further.

To determine which weightings to consider, we need to understand the role of the weighting. There are actually two possible roles. The first is to specify states of interest: determine the relative importance of a state for the accuracy of our value estimates compared to the true values. This corresponds to a weighting in the value error. The second is the choice of weighting in our objective, such as the linear $\overline{\text{PBE}}$, which is a surrogate for the value error.

For the first question, we need to determine the relative importance of states. It is difficult to propose a one-size-fits-all answer to this question, as it is highly goal dependent. For example, if the policy is being evaluated for deployment in an episodic problem, a common choice is putting all weight on the set of start states (**?**). The value in the start states reflects the expected return the agent will receive in each episode, and is sufficient for evaluating its utility for deployment. Alternatively, if many value functions are learned for predictive systems, such as those composed of GVFs (Jaderberg et al., 2016; **?**), it is likely better to ensure every state has a non-zero weighting so that predictions are not nonsensical from those states. It is also possible that value from some states might be queried much more often, or the states might correspond to important catastrophic event from which it is highly important to have accurate predictions to facilitate good decision-making.

Overall, the choice of d is subjective. Rather, we simply need to ensure that we are making the choice of d in a deliberate way in our evaluation objective so that solution performance is appropriately measured. Once we make this choice for our evaluation objective, we can ask the second question: which optimization objectives, with what weightings, are most effective for minimizing the evaluation objective? It is not obvious that, for example, the minimum of the $\overline{\text{PBE}}$ with weighting d provides the best solution to the $\overline{\text{VE}}$ with weighting

	Objective function weight d includes	
	d_π	d_b
Posterior corrections	N/A. Alternative life algorithms cannot only do posterior corrections.	Off-Policy TD(λ) GTD(λ) (Sutton et al., 2010), Hybrid TD(λ) (Maei, 2011; White & White, 2016) Action-dependent bootstrapping, including Tree Backup(λ) (Precup et al., 2000), V-trace(λ) (Espeholt et al., 2018), AB-Trace(λ) (Mahmood, Yu & Sutton, 2017) Saddlepoint methods for GTD2(λ), including GTD2-MP(λ) (Liu et al., 2015), SVRG and SAGA for policy evaluation (Du et al., 2017) and Gradient Tree Backup(λ) (Touati et al., 2018)

Prior + Posterior corrections	Alternative-life TD(λ)
	Alternative-life GTD(λ), HTD(λ), and Saddlepoint methods
		ETD(λ) (Sutton, Mahmood & White, 2016)
		ETD(λ, β) (Hallak et al., 2015)
		Emphatic GTD(λ), HTD(λ), and Saddlepoint methods

Table 1: A summary of off-policy value estimation methods for the linear $\overline{\text{PBE}}$, based on weightings in the objective and whether they incorporate both prior and posterior corrections. The algorithms in grey are hypothetical algorithms that can easily be derived by applying the same derivations as in their original works, but with alternative weightings.

d; potentially the $\overline{\text{PBE}}$ with a different weighting is better. In fact, we know that the linear $\overline{\text{PBE}}$ with $d = d_b$ suffers from a counterexample (Kolter, 2011), whereas using the emphatic weighting in the linear $\overline{\text{PBE}}$ provides an upper bound on the $\overline{\text{VE}}$ under weighting d_b. We discuss the potential utility of using a different weighting for the objective, than the desired weighting in the $\overline{\text{VE}}$, in Section 6.

3.6 Summary of Algorithms for the Linear $\overline{\text{PBE}}$

We conclude this section by summarizing the current algorithms by the objective they optimize. This two axes are whether prior or posterior corrections are used, and whether the weighting includes d_π or d_b. The emphatic weighting adjusts the state weighting with prior corrections, but still includes d_b as part of the weighting in the objective. Alternative-life, on the other hand, removes d_b all together.

The primary differences between the algorithms is in how they optimize their respective objectives. They can use gradient updates or hybrid updates, and can incorporate other additions like action-dependent bootstrapping. For example, we can easily obtain a gradient version of ETD, by incorporating emphatic weights into the GTD algorithm. The GTD(λ) algorithm is

$$\begin{aligned}
\boldsymbol{z}_t^\rho &\leftarrow \rho_t\left(\gamma_t\lambda\boldsymbol{z}_{t-1}^\rho + \boldsymbol{x}_t\right)\\
\boldsymbol{h}_{t+1} &\leftarrow \boldsymbol{h}_t + \alpha_h\left[\delta_t\boldsymbol{z}_t^\rho - (\boldsymbol{h}_t^\top\boldsymbol{x})\boldsymbol{x}_t\right]\\
\boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\delta_t\boldsymbol{z}_t^\rho - \underbrace{\alpha\gamma_{t+1}(1-\lambda)(\boldsymbol{h}_t^\top\boldsymbol{z}_t^\rho)\boldsymbol{x}_{t+1}}_{\text{correction term}}
\end{aligned}$$

The Emphatic GTD(λ) algorithm uses the same updates to the two sets of weights, but with emphatic weights in the trace: $\boldsymbol{z}_t^\rho \leftarrow \rho_t\left(\gamma_t\lambda\boldsymbol{z}_{t-1}^\rho + M_t\boldsymbol{x}_t\right)$. More simply, when considering Emphatic GTD(0), we get

$$\begin{aligned}
F_t &\leftarrow \rho_{t-1}\gamma_t F_{t-1} + 1\\
\boldsymbol{h}_{t+1} &\leftarrow \boldsymbol{h}_t + \alpha_h\rho_t F_t\left[\delta_t - (\boldsymbol{h}_t^\top\boldsymbol{x})\right]\boldsymbol{x}_t\\
\boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\rho_t F_t[\delta_t\boldsymbol{x}_t - \gamma_{t+1}(1-\lambda)(\boldsymbol{h}_t^\top\boldsymbol{x}_t)\boldsymbol{x}_{t+1}]
\end{aligned}$$

where F_t is omitted for standard GTD(0).

Many of the algorithms can incorporate different weightings, by premultiplying the update, but might use different approaches to approximate the gradient. It is important to separate the weighting and algorithmic approach, as otherwise, comparisons between algorithms (e.g., ETD and GTD) could reflect either the difference in objective (weighting by m or d_b) or in the algorithmic approach (TD-style updates or gradient updates). The algorithms, categorized according to weightings, are summarized in Table 1. We include more descriptions about these algorithms in Appendix B.

4. Broadening the Scope of Objectives

The previous section primarily focused on the linear $\overline{\text{PBE}}$ and on weightings. This organization was to provide an intuitive introduction to the importance of weightings, on the most commonly used objective in off-policy prediction. In this section, we discuss how to generalize the linear $\overline{\text{PBE}}$, to obtain the generalized $\overline{\text{PBE}}$, that unifies the $\overline{\text{BE}}$ and linear $\overline{\text{PBE}}$ under one objective.

4.1 An Overview of Existing Objectives

Let us start by discussing the standard evaluation objective used for policy evaluation: the mean squared value error ($\overline{\text{VE}}$):

$$\overline{\text{VE}}(\boldsymbol{w}) \overset{\text{def}}{=} \sum_{s\in\mathcal{S}} d(s)(\hat{v}(s,\boldsymbol{w}) - v_\pi(s))^2. \tag{11}$$

This objective cannot be directly optimized, because it requires access to the true value function $v_\pi(s)$. In experiments, however, it is a common objective for evaluation. The approximation $\hat{v}(s,\boldsymbol{w})$ is penalized more heavily for inaccurate value estimates in highly weighted states s. One way to indirectly optimize the $\overline{\mathrm{VE}}$ is to use the mean squared return error ($\overline{\mathrm{RE}}$):

$$\overline{\mathrm{RE}}(\boldsymbol{w}) \overset{\text{def}}{=} \sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi\Big[(\hat{v}(s,\boldsymbol{w}) - G)^2 \mid S = s\Big]\,. \tag{12}$$

The minima of the $\overline{\mathrm{RE}}(\boldsymbol{w})$ are equivalent to the minima of $\overline{\mathrm{VE}}(\boldsymbol{w})$, which can be seen by looking at the gradients of the $\overline{\mathrm{RE}}$:

$$\begin{aligned}
\nabla\overline{\mathrm{RE}}(\boldsymbol{w}) &= \sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi\Big[\nabla\left(\hat{v}(s,\boldsymbol{w}) - G\right)^2 \mid S = s\Big] \\
&= \sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi[(\hat{v}(s,\boldsymbol{w}) - G)\,\nabla\hat{v}(s,\boldsymbol{w}) \mid S = s] \\
&= \sum_{s\in\mathcal{S}} d(s)\left(\hat{v}(s,\boldsymbol{w}) - \mathbb{E}_\pi[G \mid S = s]\right)\nabla\hat{v}(s,\boldsymbol{w}) \\
&= \sum_{s\in\mathcal{S}} d(s)\left(\hat{v}(s,\boldsymbol{w}) - v_\pi(s)\right)\nabla\hat{v}(s,\boldsymbol{w}) = \nabla\overline{\mathrm{VE}}(\boldsymbol{w}).
\end{aligned}$$

In practice, the $\overline{\mathrm{RE}}$ is rarely used, because it requires obtaining samples of entire returns. Instead, bootstrapping is used and so forms of the Bellman error are used, as in the $\overline{\mathrm{BE}}$ and $\overline{\mathrm{PBE}}$. The $\overline{\mathrm{BE}}$ reflects the goal of approximating the fixed-point formula given by the Bellman operator $\mathcal{T}$, defined as

$$\mathcal{T}\hat{v}(\cdot,\boldsymbol{w})(s) \overset{\text{def}}{=} \mathbb{E}_\pi\big[R + \gamma\hat{v}(S',\boldsymbol{w}) \mid S = s\big] \text{ for all } s. \tag{13}$$

When equality is not possible, the difference is minimized as in the $\overline{\mathrm{BE}}$

$$\begin{aligned}
\overline{\mathrm{BE}}(\boldsymbol{w}) &\overset{\text{def}}{=} \sum_{s\in\mathcal{S}} d(s)\left(\mathcal{T}\hat{v}(\cdot,\boldsymbol{w})(s) - \hat{v}(s,\boldsymbol{w})\right)^2 \\
&= \sum_{s\in\mathcal{S}} d(s)\left(\mathbb{E}_\pi\big[R + \gamma\hat{v}(S',\boldsymbol{w}) \mid S = s\big] - \hat{v}(s,\boldsymbol{w})\right)^2.
\end{aligned} \tag{14}$$

There has been much discussion, formal and informal, about the utility of using the $\overline{\mathrm{BE}}$ versus the $\overline{\mathrm{PBE}}$ (Scherrer, 2010). The $\overline{\mathrm{BE}}$ can be decomposed into the $\overline{\mathrm{PBE}}$ and a projection penalty term (Scherrer, 2010). To understand why, recall the definition of a projection operator. For a given vector space $\mathcal{F}$, the projection of a vector v onto this space is the closest point under a given (weighted) norm $\|\cdot\|_d$: $\min_{u\in\mathcal{F}}\|u - v\|_d$. This definition also applies to function spaces. Let $\Pi_{\mathcal{F},d}$ be the weighted projection on the space of value functions, defined as

$$\Pi_{\mathcal{F},d}u \overset{\text{def}}{=} \underset{u\in\mathcal{F}}{\arg\min}\,\|u - v\|_d \tag{15}$$

For a given vector $v \in \mathbb{R}^|\mathcal{S}|$, composed of value estimates for each state, we get

$$\|v - \mathcal{T}v\|_D^2 = \underbrace{\|v - \Pi_{\mathcal{F},d}\mathcal{T}v\|_d^2}_{\overline{\mathrm{PBE}}} + \underbrace{\|\mathcal{T}V - \Pi_{\mathcal{F},d}\mathcal{T}v\|_d^2}_{\text{Projection Penalty}} \tag{16}$$

This penalty causes the $\overline{\mathrm{BE}}$ to prefer value estimates for which the projection does not have a large impact near the solution. The $\overline{\mathrm{PBE}}$ can find a fixed point where applying the Bellman operator $\mathcal{T}v$ moves far outside the space of representable functions, as long as the projection back into the space stays at V. The projection penalty is sensible, and in fact prevents some of the counterexamples on the solution quality for the $\overline{\text{PBE}}$ discussed in Section 6.

Despite the potential utility of the $\overline{\text{BE}}$, it has not been widely used due to difficulties in optimizing this objective without a model. The $\overline{\text{BE}}$ is difficult to optimize because of the well-known double sampling problem for the gradient. To see why, consider the gradient of the $\overline{\text{BE}}$, where we use the fact that $\mathbb{E}_\pi[\delta \mid S = s] = \mathbb{E}_\pi[R + \gamma\hat{v}(S', \boldsymbol{w}) \mid S = s] - \hat{v}(s, \boldsymbol{w})$ for $\delta = R + \gamma\hat{v}(S', \boldsymbol{w}) - \hat{v}(S, \boldsymbol{w})$

$$\begin{aligned}
\nabla_{\boldsymbol{w}}\overline{\text{BE}}(\boldsymbol{w}) &= \sum_{s\in\mathcal{S}} d(s)\nabla_{\boldsymbol{w}}\mathbb{E}_\pi[\delta \mid S = s]^2 \\
&= 2\sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi[\delta \mid S = s]\, \mathbb{E}_\pi[\nabla_{\boldsymbol{w}}\delta \mid S = s] \\
&= 2\sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi[\delta \mid S = s]\, \mathbb{E}_\pi\big[\gamma\nabla_{\boldsymbol{w}}\hat{v}(S', \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w}) \mid S = s\big]
\end{aligned}$$

To estimate this gradient for a given $S = s$, we need two independent samples of the next state and reward. Then when we use the first to get a sample δ and the second to get a sample of $\gamma\nabla_{\boldsymbol{w}}\hat{v}(S', \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})$. The product of these two samples then gives an unbiased sample of the product of the expectations. If we instead only used one sample, we would erroneously obtain a sample of $\mathbb{E}_\pi[\delta(\gamma\nabla_{\boldsymbol{w}}\hat{v}(S', \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})) \mid S = s]$.

There has been a promising attempt to approximate the $\overline{\text{BE}}$, under restricted conditions, using a non-parametric approach (Feng et al., 2019). The objective, called the $\overline{\text{KBE}}$, takes pairs of samples from a buffer to overcome the double sampling problem. Unfortunately, this cannot overcome the issue of identifiability in the $\overline{\text{BE}}$. There is a simple example where the same data is generated by two different MDPs, with different optima for the corresponding $\overline{\text{BE}}$. The agent cannot hope to use the data to identify which of the two parameters is the optimal solution, because the generated data even in the limit is identical for the two MDPs. The simple conclusion is that we cannot hope to perfectly optimize the $\overline{\text{BE}}$ in all cases, and would instead be optimizing an approximation.

The linear $\overline{\text{PBE}}$, on the other hand, is practical to optimize under linear function approximation, as discussed above: the whole family of (gradient) TD algorithms is designed to optimize the linear $\overline{\text{PBE}}$. Unfortunately, the $\overline{\text{PBE}}$ is hard to optimize for the general nonlinear setting, because the projection is hard to compute. There has been an extension of GTD to the nonlinear $\overline{\text{PBE}}$ (Maei et al., 2009), but the algorithm requires computing Hessian-vector products. In the next section, we discuss how to overcome these issues, with a unified objective, that is a generalization of the $\overline{\text{PBE}}$.

Finally, for completeness, we conclude with a description of the Mean-Squared TD-error ($\overline{\text{TDE}}$), even though it is rarely used. The $\overline{\text{TDE}}$ was introduced to characterize the TD solution as a semi-gradient method. For the objective

$$\overline{\text{TDE}}(\boldsymbol{w}) \overset{\text{def}}{=} \sum_{s\in\mathcal{S}} d(s)\mathbb{E}_\pi\Big[\big(R + \gamma\hat{v}(S', \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w})\big)^2 \mid S = s\Big]. \tag{17}$$

the gradient includes the gradient of $\hat{v}(S', \boldsymbol{w})$. TD omits this term, and so is called a semi-gradient method. One could, however, actually use gradient descent on the $\overline{\text{TDE}}$, though it is not typically done due to commonly held views of poor quality and a counterexample for the residual gradient algorithm which uses the $\overline{\text{TDE}}$ (Sutton and Barto, 2018). We additionally highlight the significant bias due when using the $\overline{\text{TDE}}$, in Appendix A, providing further evidence that it is likely not a useful direction to explore.

4.2 An Identifiable $\overline{\text{BE}}$

Before discussing the generalized $\overline{\text{PBE}}$, we start by showing a conjugate form for the $\overline{\text{BE}}$. This reformulation uses the strategy introduced by Dai et al. (2017), which more generally introduces this conjugate form for several objectives that use conditional expectations. They show how to use it for the $\overline{\text{BE}}$ as an example, but defined it slightly differently because they condition on states and actions. For this reason, and because we will build further, we provide the explicit steps to derive the conjugate form for the $\overline{\text{BE}}$.

Let $\mathcal{F}$ be the space of parameterized value functions and $\mathcal{F}_{\text{all}}$ the space of all functions. Then the $\overline{\text{BE}}$ can be re-expressed as

$$\overline{\text{BE}}(\boldsymbol{w}) = \max_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2 \right)$$

This reformulation comes from the fact that the conjugate of the square function is $y^2 = \max_{h \in \mathbb{R}} 2yh - h^2$ and because the maximum can be brought outside the sum, as long as a different scalar h can be chosen for each state s, as it can be for $\mathcal{F}_{\text{all}}$ the space of all functions. To see the explicit steps,

$$\begin{aligned}
\overline{\text{BE}}(\boldsymbol{w}) &= \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 \\
&= \sum_{s \in \mathcal{S}} d(s) \max_{h \in \mathbb{R}} \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h - h^2 \right) && \triangleright \text{ using the conjugate function} \\
&= \max_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2 \right) && \triangleright \text{ using interchangeability.}
\end{aligned}$$

The optimal $h^*(s) = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$, because

$$\begin{aligned}
&\operatorname*{arg\,max}_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2 \right) \\
&\qquad = \operatorname*{arg\,max}_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \Big(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2 - \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 \Big) \\
&\qquad = \operatorname*{arg\,max}_{h \in \mathcal{F}_{\text{all}}} - \sum_{s \in \mathcal{S}} d(s) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] - h(s) \right)^2 \\
&\qquad = \operatorname*{arg\,min}_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] - h(s) \right)^2 .
\end{aligned}$$

The function $h^*(s) = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$ provides the minimal error of zero. This optimal solution also makes it clear why the above is simply a rewriting of the $\overline{\text{BE}}$ because

$$2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h^*(s) - h^*(s)^2 = 2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 - \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 .$$

More generally, for the continuous state case, interchangeability also holds, as long as the function $b(s) = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$ satisfies $b \in \mathcal{F}_{\text{all}}$. Notice first that we could have more generically expressed the $\overline{\text{BE}}$ using expectations over states: $\overline{\text{BE}}(\boldsymbol{w}) = \mathbb{E}[\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S]^2]$, where the outer expectation is over the random variable $\mathcal{S}$ with distribution d. Let $g(h, s) = (\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] - h)^2$. Then, for the continuous state setting, we have that the $\overline{\text{BE}}$ is

$$\mathbb{E}\left[\min_{h\in\mathbb{R}} g(h, S)\right] = \int_{\mathcal{S}} d(s) \min_{h\in\mathbb{R}} g(h, s) ds = \min_{h\in\mathcal{F}_{\text{all}}} \int_{\mathcal{S}} d(s) g(h(s), s) ds. \tag{18}$$

We use a minimization over h, simply because the resulting g is more intuitive. Because $b(s) = \mathbb{E}_\pi[\delta \mid S = s]$ satisfies $b \in \mathcal{F}_{\text{all}}$, we know that a minimizer exists, as $h^* = b \in \mathcal{F}_{\text{all}}$. Then we can show that $\mathbb{E}[\min_{h\in\mathbb{R}} g(h, S)] = \mathbb{E}[g(b(S), S)] = \mathbb{E}[g(h^*(S), S)] = \min_{h\in\mathcal{F}_{\text{all}}} \mathbb{E}[g(h(S), S)]$.[2]

As highlighted in (Sutton and Barto, 2018, Chapter 8), the $\overline{\text{BE}}$ is not identifiable. In that example, however, the inputs given to the value function learner are partially observable. In terms of the above formulation, this would mean the agent can only see a part of the state for learning $\boldsymbol{w}$ but the whole state to learn h. This is not a realistic setting. Rather, if the agent truly has a partial view of state to learn the values, then the input-space for h should be similarly restricted. The function approximation for h could still be more powerful than for v—the agent can chose to allocate its resources how it wants. This leads us to a new set for h, which includes all functions on the same inputs $\phi(s)$ as given to v, rather than on state:

$$\mathcal{H}_{\text{all}} \overset{\text{def}}{=} \{h = f \circ \phi \mid \text{ where } f \text{ is any function on the space produced by } \phi\}.$$

The resulting h is restricted to functions of the form $h(s) = f(\phi(s))$. We call the resulting $\overline{\text{BE}}$ an *Identifiable* $\overline{\text{BE}}$, written as:

$$\text{Identifiable } \overline{\text{BE}}(\boldsymbol{w}) \overset{\text{def}}{=} \max_{h\in\mathcal{H}_{\text{all}}} \ \mathbb{E}\left[2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S]\, h(S) - h(S)^2\right].$$

Notice that $\mathcal{H}_{\text{all}} \subseteq \mathcal{F}_{\text{all}}$, and so the solution to the Identifiable $\overline{\text{BE}}$ may be different from the solution to the $\overline{\text{BE}}$. In particular, we know Identifiable $\overline{\text{BE}}(\boldsymbol{w}) \le \overline{\text{BE}}(\boldsymbol{w})$, because the inner maximization is more constrained. In fact, restricting h can be seen as a projection on the errors in the objective, as we discuss next, making the Identifiable $\overline{\text{BE}}$ an instance of the generalized $\overline{\text{PBE}}$.

4.3 From the Identifiable Bellman Error back to a Projected Bellman Error

The previous section discussed a conjugate form for the $\overline{\text{BE}}$, which led to an Identifiable $\overline{\text{BE}}$. Even this Identifiable $\overline{\text{BE}}$, however, can be difficult to optimize, as we will not be able to perfectly represent any h in $\mathcal{H}_{\text{all}}$. In this section, we discuss further approximations, with $h \in \mathcal{H} \subseteq \mathcal{H}_{\text{all}}$, leading to a new set of Projected Bellman Errors that encompasses the previously defined linear $\overline{\text{PBE}}$.

2. This argument is similar to (Dai et al., 2017, Lemma 1), except they use a maximization, rather than a minimization and make assumptions about g which for them is generic. Their argument exactly holds for pulling out the minimum as well, but simply modifying the conditions to be lower semi-continuous and convex, rather than upper semi-continuous and concave. We do not need to make these assumptions, as we know the form of our g and can directly assume the existence of a minimizer.

To use minimax formulation for the $\overline{\mathrm{BE}}$, we need to approximate h as an auxiliary estimator. This means h must also be a parameterized function, and we will instead only obtain an approximation to the $\overline{\mathrm{BE}}$. Let $\mathcal{H}$ be the space of parameterized functions for this auxiliary function h. As we show below, this $\mathcal{H}$ defines the projection in the generalized $\overline{\mathrm{PBE}}$.

As we showed above, the maximization over h can be written as a minimization using a weighted squared error, to the function $\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$. In the finite state setting, we simply take the vector $u \in |\mathcal{S}|$ composed of entries $\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$: the vector $u = \mathcal{T}\hat{v}(\cdot, \boldsymbol{w}) - \hat{v}(\cdot, \boldsymbol{w})$. The projection operator is

$$\Pi_{\mathcal{H},d} u \overset{\text{def}}{=} \arg\min_{h \in \mathcal{H}} \|u - h\|_d \tag{19}$$

Notice that $u = \Pi_{\mathcal{H},d} u + \tilde{u} = h + \tilde{u}$, where $h = \Pi_{\mathcal{H},d} u$ and $\tilde{u}$ is the component in u that is orthogonal in the weighted space: $h^\top D \tilde{u} = 0$ for $D \overset{\text{def}}{=} \operatorname{diag}(d)$. Then we get

$$\begin{aligned}
\overline{\mathrm{PBE}}(\boldsymbol{w}) &\overset{\text{def}}{=} \max_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2\right) \\
&= \max_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} d(s) \left(2u(s)h(s) - h(s)^2\right) && \triangleright \text{ rewriting } u(s) = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] \\
&= \sum_{s \in \mathcal{S}} d(s) \left(2u(s)h(s) - h(s)^2\right) && \triangleright \text{ where } h = \Pi_{\mathcal{H},d} u \\
&= \sum_{s \in \mathcal{S}} d(s) \left(2(h(s) + \tilde{u}(s))h(s) - h(s)^2\right) && \triangleright \text{ because } u(s) = h(s) + \tilde{u}(s) \\
&= \sum_{s \in \mathcal{S}} d(s) \left(2h(s)^2 - h(s)^2\right) + 2\sum_{s \in \mathcal{S}} d(s)\tilde{u}(s)h(s) \\
&= \sum_{s \in \mathcal{S}} d(s)h(s)^2 + 2\sum_{s \in \mathcal{S}} d(s)\tilde{u}(s)h(s) \\
&= \sum_{s \in \mathcal{S}} d(s)h(s)^2 && \triangleright \text{ where } \sum_{s \in \mathcal{S}} d(s)\tilde{u}(s)h(s) = 0 \text{ because} \\
&= \|\Pi_{\mathcal{H},d}(\mathcal{T}\hat{v}(\cdot, \boldsymbol{w}) - \hat{v}(\cdot, \boldsymbol{w}))\|_d^2 && h \text{ is orthogonal to } \tilde{u}, \text{ under weighting } d
\end{aligned}$$

Therefore each choice of $\mathcal{H}$ results in different projection operators. This view provides a nice intuition on the role of approximating h. Depending on how errors are projected, the value function approximation will focus more or less on the Bellman errors in particular states. If the Bellman error is high in a state, but those errors are projected to zero, then no further approximation resources will be used for that state. Under no projection—the set for h being the set of all functions—no errors are projected and the values are learned to minimize the Bellman error. If $\mathcal{H} = \mathcal{F}$, the same space is used to represent h and v, then we obtain the projection originally used for the $\overline{\mathrm{PBE}}$.

4.4 Connection to Previous $\overline{\mathrm{PBE}}$ Objectives

These min-max formulations, or saddlepoint formulations, have been used for the (linear) $\overline{\mathrm{PBE}}$ (Mahadevan et al., 2014; Liu et al., 2016; Touati et al., 2018). The goal there was to directly re-express the $\overline{\mathrm{PBE}}$, rather than to re-express the $\overline{\mathrm{BE}}$ or find connections between them. The linear $\overline{\mathrm{PBE}} = \|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}}$ can be rewritten using the conjugate for the two norm. The conjugate for the two-norm is $\frac{1}{2}\|\boldsymbol{y}\|_{\mathbf{C}^{-1}} = \max_{\boldsymbol{h}} \boldsymbol{y}^\top \boldsymbol{h} - \frac{1}{2}\|\boldsymbol{h}\|^2_{\mathbf{C}}$, with optimal $\boldsymbol{h} = \mathbf{C}^{-1}\boldsymbol{y}$. Correspondingly, we get

$$\tfrac{1}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}} = \max_{h \in \mathbb{R}^d} (\boldsymbol{b} - \mathbf{A}\boldsymbol{w})^\top \boldsymbol{h} - \tfrac{1}{2}\|\boldsymbol{h}\|^2_{\mathbf{C}}$$

where the solution for $\boldsymbol{h} = \mathbf{C}^{-1}(\boldsymbol{b} - \mathbf{A}\boldsymbol{w})$. This solution makes the first term equal to $\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}}$ and the second term equal to $-\frac{1}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}}$; adding them together gives $\frac{1}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}}$.

We can obtain the same formulation under the generalized $\overline{\text{PBE}}$, by restricting the sets $\mathcal{F}$ and $\mathcal{H}$ to be the same set of linear functions. Let $\mathcal{L} = \{f : \mathcal{S} \to \mathbb{R} : f(s) = \boldsymbol{x}(s)^\top \boldsymbol{w}, \boldsymbol{w} \in \mathbb{R}^d\}$. For $\mathcal{F} = \mathcal{H} = \mathcal{L}$, we have

$$\begin{aligned} h^* &= \operatorname*{arg\,min}_{h \in \mathcal{L}} \sum_{s \in \mathcal{S}} d(s) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] - h(s)\right)^2 \\ \text{satisfies} \quad h^*(s) &= \boldsymbol{x}(s)^\top \boldsymbol{h}^* \quad \text{where } \boldsymbol{h}^* = \mathbf{C}^{-1}(\boldsymbol{b} - \mathbf{A}\boldsymbol{w}) \end{aligned}$$

where the linear regression solution, for targets $\delta(\boldsymbol{w})$, is $\boldsymbol{h}^* = \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right]^{-1} \mathbb{E}[\boldsymbol{x}\delta(\boldsymbol{w})]$ which equals $\mathbf{C}^{-1}(\boldsymbol{b} - \mathbf{A}\boldsymbol{w})$. Verifying that the resulting $\overline{\text{PBE}}$ matches the linear $\overline{\text{PBE}}$:

$$\begin{aligned} &\max_{h \in \mathcal{L}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2\right) \\ &= \sum_{s \in \mathcal{S}} d(s) 2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, \boldsymbol{x}(s)^\top \boldsymbol{h}^* - \sum_{s \in \mathcal{S}} d(s)(\boldsymbol{x}(s)^\top \boldsymbol{h}^*)^2 \\ &= \left(\sum_{s \in \mathcal{S}} d(s) 2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, \boldsymbol{x}(s)^\top\right) \boldsymbol{h}^* - \sum_{s \in \mathcal{S}} d(s)(\boldsymbol{h}^*)^\top \boldsymbol{x}(s)\boldsymbol{x}(s)^\top \boldsymbol{h}^* \\ &= 2\mathbb{E}\left[\delta(\boldsymbol{w})\boldsymbol{x}(s)\right]^\top \boldsymbol{h}^* - (\boldsymbol{h}^*)^\top \left(\sum_{s \in \mathcal{S}} d(s)\boldsymbol{x}(s)\boldsymbol{x}(s)^\top\right) \boldsymbol{h}^* \\ &= 2(\boldsymbol{b} - \mathbf{A}\boldsymbol{w})\boldsymbol{h}^* - (\boldsymbol{h}^*)^\top \mathbf{C}\boldsymbol{h}^* \\ &= 2\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}} - (\boldsymbol{h}^*)^\top (\boldsymbol{b} - \mathbf{A}\boldsymbol{w}) \\ &= 2\|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}} - \|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}} \\ &= \|\boldsymbol{b} - \mathbf{A}\boldsymbol{w}\|^2_{\mathbf{C}^{-1}} \end{aligned}$$

This result is alluded to in the connection between the NEU and the $\overline{\text{KBE}}$, in (Feng et al., 2019, Corollary 3.5), but not explicitly shown.

This connection also exists with the nonlinear $\overline{\text{PBE}}$, but with a surprising choice for the parameterization of h: using the gradient of the value estimate as the features. The nonlinear $\overline{\text{PBE}}$ was introduced for nonlinear value function approximations that are twice differentiable, and is defined as (Maei et al., 2009)

$$\text{nonlinear } \overline{\text{PBE}}(\boldsymbol{w}) = \mathbb{E}[\delta(\boldsymbol{w})\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})]^\top \mathbb{E}\left[\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})^\top\right]^{-1} \mathbb{E}[\delta(\boldsymbol{w})\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})]$$

This corresponds to the linear $\overline{\text{PBE}}$ when $\mathcal{F} = \mathcal{L}$, because $\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w}) = \boldsymbol{x}(s)$. Define set $\mathcal{G}_{\boldsymbol{w}} = \{f : \mathcal{S} \to \mathbb{R} : f(s) = \boldsymbol{y}(s)^\top \boldsymbol{h}, \boldsymbol{h} \in \mathbb{R}^d \text{ and } \boldsymbol{y}(s) = \nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w})\}$. Notice that this function set for h changes as $\boldsymbol{w}$ changes. Then we get that

$$h^*_{\text{nonlinear}} = \operatorname*{arg\,min}_{h \in \mathcal{G}_{\boldsymbol{w}}} \sum_{s \in \mathcal{S}} d(s) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s] - h(s)\right)^2$$

satisfies $h^*_{\text{nonlinear}}(s) = \nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w})^\top \boldsymbol{h}^*_{\text{nonlinear}}$ where

$$\boldsymbol{h}^*_{\text{nonlinear}} = \mathbb{E}\left[\nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w}) \nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w})^\top\right]^{-1} \mathbb{E}\left[\delta(\boldsymbol{w}) \nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w})\right]$$

Plugging this optimal h back into the formula, and using similar steps to above, we get that

$$\begin{aligned}
&\max_{h \in \mathcal{G}_{\boldsymbol{w}}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2\right) \\
&= \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h^*_{\text{nonlinear}}(s) - h^*_{\text{nonlinear}}(s)^2\right) \\
&= 2\text{nonlinear } \overline{\text{PBE}}(\boldsymbol{w}) - \text{nonlinear } \overline{\text{PBE}}(\boldsymbol{w}) \\
&= \text{nonlinear } \overline{\text{PBE}}(\boldsymbol{w})
\end{aligned}$$

This nonlinear $\overline{\text{PBE}}$ is not an instance of the generalized $\overline{\text{PBE}}$, as we have currently defined it, because the $\mathcal{H}$ changes with $\boldsymbol{w}$. It is possible that such a generalization is worthwhile, as using the gradient of the values as features is intuitively useful. Further, interchangeability should still hold, as the exchange of the maximum was done for a fixed $\boldsymbol{w}$. Therefore, it would be appropriate to assume that $\mathcal{H}$ changes with $\boldsymbol{w}$, and in our experiments we test $\mathcal{H} = \mathcal{G}_{\boldsymbol{w}}$.

4.5 Summary Discussion

The implication of these connections is that we can strictly generalize the $\overline{\text{PBE}}$, by considering different sets $\mathcal{H}$. The most important outcome is that we have natural choices to explore for the nonlinear function approximation setting. As secondary outcomes, we also provide a clear connection between the $\overline{\text{BE}}$ and $\overline{\text{PBE}}$, based on a difference in the choice of projection, and resolve the identifiability issue in the $\overline{\text{BE}}$.

The next question how we should choose $\mathcal{H}$, both ideally and practically. In the next section, we provide some bounds on the $\overline{\text{VE}}$, that include properties of $\mathcal{H}$, as a first step towards guiding our choice of $\mathcal{H}$. These theoretical choices, however, might not always be practical, because $\mathcal{H}$ is naturally restricted by our function approximators. Practically, it is likely the simplest to use the same approximator for h as for the values. However, it is possible that different criteria are used for selecting $\hat{v}$ versus h. For example, we might want $\hat{v}$ to be efficient to query, and so use a compact parametric function approximator. On the other hand, maybe h can be a bit more costly, since it is only used in training, not during deployment.

Further, bias in h may be undesirable, pointing to using less efficient but less biased nonparametric function approximators. For example, a reservoir of samples could be stored, where $\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$ is aproximated using a weighted average over $\delta(\boldsymbol{w})$ in the buffer, where the weighting is proportional to similarity between that state and s. The δ values in similar states to s should be similar to δ in s, and so can provide a reasonable sample average estimator. This is the strategy taken by the Kernel $\overline{\text{BE}}$ (Feng et al., 2019), precisely to reduce bias in h and so better approximate the $\overline{\text{BE}}$. At the same time, its not clear that reducing such bias is key, as solutions under the $\overline{\text{PBE}}$ are usually good despite projection.

The other reason the choice of $\mathcal{H}$ matters is for the optimization path itself, rather than the resulting solution. In the next section, we discuss two strategies to optimize the generalized $\overline{\text{PBE}}$. We highlight that one approach—the saddlepoint update—suffers more when there is bias in h. This saddlepoint update is the one used in previous approximations to the $\overline{\text{BE}}$, potentially explaining why they focused on nonparametric approximations. The gradient correction update, on the other hand, relies much less heavily on accuracy of h during the optimization, and so allows us to focus more on selecting a practical $\mathcal{H}$ based primarily on the fixed point for the generalized $\overline{\text{PBE}}$.

5. Algorithms for the Generalized $\overline{\text{PBE}}$

The $\overline{\text{PBE}}$ is often optimized using gradient-correction algorithms as opposed to saddlepoint methods. The canonical methods are TDC and GTD2, where TDC has been consistently shown to perform better than GTD2 as it relies less on having an accurate estimator for h (White and White, 2016; Ghiassian et al., 2020). The fact that the $\overline{\text{BE}}$ generalizes on the $\overline{\text{PBE}}$ may not mean that the set of algorithms is also a strict generalization. We show in this section that similar gradient-correction algorithms arise for the generalized $\overline{\text{PBE}}$, with some differences in the gradient-correction update for some choices of $\mathcal{H}$. We conclude by extending the algorithms to λ-returns and n-steps returns.

5.1 Estimating the Gradient of the Generalized $\overline{\text{PBE}}$

To see why (at least) two classes of algorithms arise, consider the gradient update for the generalized $\overline{\text{PBE}}$, for a given $h(s) \approx \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]$ with a stochastic sample $\delta(\boldsymbol{w})$ from $S = s$:

$$-\nabla_{\boldsymbol{w}} \delta(\boldsymbol{w}) h(s) = h(s)[\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w})]$$

This is the standard *saddlepoint update*. In GTD2, h is estimated using a linear function approximator. The key issue with this form is that h can be highly inaccurate during learning. Typically, it is initialized to zero, and so it multiples the update to the primary weights by a number near zero, making learning slow. In general, any inaccuracy in h has a big impact on the update of $\boldsymbol{w}$.

The *gradient-correction update* is obtained by assuming the optimal $h^* \in \mathcal{H}$ is used in a part of the gradient. It has only be derived for the linear setting, but here we start with the generic update and see how TDC emerges in this special case. We can rewrite

$$\begin{aligned}
-\nabla_{\boldsymbol{w}} \delta(\boldsymbol{w}) h(s) &= h(s)[\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w})] \\
&= h(s) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s) \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w}) \\
&= (h(s) - \delta(\boldsymbol{w}) + \delta(\boldsymbol{w})) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s) \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w}) \\
&= \delta(\boldsymbol{w}) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) + (h(s) - \delta(\boldsymbol{w})) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s) \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w})
\end{aligned}$$

This resembles the TDC updates, except that it has an extra term $(h(s) - \delta(\boldsymbol{w})) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w})$. In the linear setting, if we have the true linear regression solution for $\boldsymbol{h} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]^{-1} \mathbb{E}[\boldsymbol{x}\delta]$, then this second term is zero in expectation. This is because $\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) = \boldsymbol{x}(s)$ and so

$$\begin{aligned}
\mathbb{E}_\pi[(h(s) - \delta(\boldsymbol{w}))\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) \mid S = s] &= \mathbb{E}_\pi[\boldsymbol{x}(s)(h(s) - \delta(\boldsymbol{w})) \mid S = s] \\
&= \mathbb{E}_\pi\Big[\boldsymbol{x}(s)(\boldsymbol{x}(s)^\top \boldsymbol{h} - \delta(\boldsymbol{w})) \mid S = s\Big] \\
&= \boldsymbol{x}(s)(\boldsymbol{x}(s)^\top \boldsymbol{h} - \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]) \\
&= \boldsymbol{x}(s)\boldsymbol{x}(s)^\top \boldsymbol{h} - \boldsymbol{x}(s)\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s])
\end{aligned}$$

and so in expectation across all states

$$\begin{aligned}
\mathbb{E}[(h(S) - \delta(\boldsymbol{w}))\nabla_{\boldsymbol{w}} v(S, \boldsymbol{w})] &= \mathbb{E}[\boldsymbol{x}(s)\boldsymbol{x}(s)^\top]\boldsymbol{h} - \mathbb{E}[\boldsymbol{x}(S)\delta(\boldsymbol{w})] \\
&= \mathbb{E}[\boldsymbol{x}(s)\boldsymbol{x}(s)^\top]\mathbb{E}[\boldsymbol{x}(S)\boldsymbol{x}(S)^\top]^{-1}\mathbb{E}[\boldsymbol{x}(S)\delta(\boldsymbol{w})] - \mathbb{E}[\boldsymbol{x}(S)\delta(\boldsymbol{w})] \\
&= \mathbb{E}[\boldsymbol{x}(S)\delta(\boldsymbol{w})] - \mathbb{E}[\boldsymbol{x}(S)\delta(\boldsymbol{w})] = 0
\end{aligned}$$

Therefore, this term can be dropped from the full gradient, across all states. The stochastic gradient, then, can also omit this term and still be an unbiased estimate of the full gradient, for the optimal $h \in \mathcal{H}$.

More generally, the same reasoning applies if $h(s)$ can be re-expressed as a linear function of $\nabla_{\boldsymbol{w}} v(S, \boldsymbol{w})$. This provides further motivation for using features produced by the gradient of the values, as in the nonlinear $\overline{\text{PBE}}$, to estimate h. Another appropriate choice is to use the features in the last layer of the neural network used for $v(S, \boldsymbol{w})$. Because the output is a linear weighting of features from the last layer, $\nabla_{\boldsymbol{w}} v(S, \boldsymbol{w})$ includes this last layer as one part of the larger vector. A head for h can be added to the neural network, where h is learned as a linear function of this layer. Its updates do not influence the neural network itself, and gradients are not passed backwards through the network.

Gradient-correction updates are preferable because the gradient estimate relies less on the accuracy of $h(s)$. The first term uses only the sampled TD-error. The update, however, is no longer a straightforward gradient update, complicating analysis. The saddlepoint update is a standard gradient update, and so can rely on many theoretical results. The gradient-correction update assumes we have the optimal h: that we have fully solved for h for a given $\boldsymbol{w}$. However, it is likely that we can characterize the dynamical system induced by the following formulas which omit this second term:

$$\begin{aligned}
\Delta\boldsymbol{w} &\leftarrow \delta(\boldsymbol{w})\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s)\gamma\nabla_{\boldsymbol{w}} v(S', \boldsymbol{w}) \\
\Delta\boldsymbol{h} &\leftarrow (\delta(\boldsymbol{w}) - h(s, \boldsymbol{h}))\nabla_{\boldsymbol{h}} h(s, \boldsymbol{h})
\end{aligned}$$

The asymptotic solution does not require the omitted second term, under certain conditions on h, as discussed above. If the dynamical system itself moves towards this stable solution, then convergence could be shown. The TDC update, which is itself not a gradient update, relies on just such a strategy: the joint update is rewritten as a linear system, that is then shown to be a contraction that iterates towards a stable solution.

Remark: There is one other way to interpret gradient-correction algorithms, as an approximation of the gradient of the Identifiable $\overline{\text{BE}}$. Notice that we can consider two forms for the negative gradient of the $\overline{\text{BE}}$:

$$\mathbb{E}_\pi[\delta \mid S = s]\,\mathbb{E}_\pi\big[\nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w}) - \gamma\nabla_{\boldsymbol{w}} \hat{v}(S', \boldsymbol{w}) \mid S = s\big]$$

or

$$\mathbb{E}_\pi[\delta \mid S=s]\,\nabla_{\boldsymbol{w}}\hat{v}(s,\boldsymbol{w}) - \mathbb{E}_\pi[\delta \mid S=s]\,\mathbb{E}_\pi\big[\gamma\nabla_{\boldsymbol{w}}\hat{v}(S',\boldsymbol{w}) \mid S=s\big]$$

because $\hat{v}(s,\boldsymbol{w})$ is not random. We can estimate the first form of the gradient using an estimate $h(s) \approx \mathbb{E}_\pi[\delta \mid S=s]$:

$$h(s)(\nabla_{\boldsymbol{w}}\hat{v}(s,\boldsymbol{w}) - \gamma\nabla_{\boldsymbol{w}}\hat{v}(S',\boldsymbol{w}))$$

This corresponds to a saddlepoint update. When estimating the second form of the gradient, notice that we do not have a double sampling problem for the first term. This means we do not need to use an estimate for $\mathbb{E}_\pi[\delta \mid S=s]$ and can instead use an unbiased sample.

$$\delta\nabla_{\boldsymbol{w}}\hat{v}(s,\boldsymbol{w}) - h(s)\gamma\nabla_{\boldsymbol{w}}\hat{v}(S',\boldsymbol{w})$$

This strategy corresponds to the gradient correction update.

5.2 Extensions to n-step returns

There are further n-step variants of these objectives, where bootstrapping occurs only after n steps: $(G_{t,n} - \hat{v}(S_t,\boldsymbol{w}))^2$ where $G_{t,n} = R_{t+1} + \gamma_{t+1}R_{t+2} + \ldots + \gamma_{t+1:t+n}R_{t+n} + \gamma_{t+1:t+n+1}\hat{v}(S_{t+n+1},\boldsymbol{w})$ where $\gamma_{t+1:t+n} = \gamma_{t+1}\gamma_{t+2}\ldots\gamma_{t+n}$. The extreme of n-step returns is to use the full return with no bootstrapping, as in Monte Carlo methods, with the objective becoming the $\overline{\text{RE}}$. The conjugate form and derivations above extend to n-step returns, simply by considering the n-step Bellman operator and corresponding n-step TD error:

$$\delta^{(n)}(\boldsymbol{w}) \overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}R_{t+2} + \ldots + \gamma_{t+1:t+n}R_{t+n} + \gamma_{t+1:t+n+1}\hat{v}(S_{t+n+1},\boldsymbol{w}) - v(S_t,\boldsymbol{w})$$

with importance sampling ratios included, in the off-policy setting. The n-step generalized $\overline{\text{PBE}}$ is $\max_{h\in\mathcal{H}} \mathbb{E}_d[2\mathbb{E}_\pi\big[\delta^{(n)}(\boldsymbol{w}) \mid S=s\big]\,h(s) - h(s)^2]$. The function h is trying to estimate the expected n-step return from s: $\mathbb{E}_d[2\mathbb{E}_\pi\big[\delta^{(n)}(\boldsymbol{w}) \mid S=s\big]$. The saddlepoint update for $\boldsymbol{w}$ is

$$\Delta\boldsymbol{w} \leftarrow h(S_t)\,(\nabla_{\boldsymbol{w}}v(S_t,\boldsymbol{w}) - h(S_t)\gamma_{t+1:t+n+1}\nabla_{\boldsymbol{w}}v(S_{t+n+1},\boldsymbol{w}))$$

and the gradient-correction update is

$$\Delta\boldsymbol{w} \leftarrow \delta^{(n)}(\boldsymbol{w})\nabla_{\boldsymbol{w}}v(S_t,\boldsymbol{w}) - h(S_t)\gamma_{t+1:t+n+1}\nabla_{\boldsymbol{w}}v(S_{t+n+1},\boldsymbol{w})$$

where both use the same update for h:

$$\Delta\boldsymbol{h} \leftarrow -(\delta^{(n)}(\boldsymbol{w}) - h(S_t,\boldsymbol{h}))\nabla_{\boldsymbol{h}}h(S_t,\boldsymbol{h})$$

The primary difference when considering n-step returns is that, for large n, it is less necessary to estimate h. For large n, the correlation between $\delta^{(n)}(\boldsymbol{w})$ and $v(S_{t+n+1},\boldsymbol{w})$ becomes smaller. Consequently, it would not be unreasonable to use $\delta^{(n)}(\boldsymbol{w})\nabla_{\boldsymbol{w}}v(S_t,\boldsymbol{w}) - \delta^{(n)}(\boldsymbol{w})\gamma_{t+1:t+n}\nabla_{\boldsymbol{w}}v(S_{t+n+1},\boldsymbol{w})$, as the incurred bias is likely small. Further, if the discount per step is less than 1, then the gradient correction term also diminishes in importance, because it is pre-multiplied by $\gamma_{t+1:t+n+1}$. For example, for a constant $\gamma < 1$, we get $\gamma_{t+1:t+n} = \gamma^n$. One might expect that the gradient-correction update might have an even greater advantage here over the saddlepoint update. It remains an open question as to the relationship between n and some of these choices.

6. Bounding Value Error and the Impact of Weighting on Solution Quality

The desired objective to minimize is the value error. We can ask how minimizing surrogate objectives, like the $\overline{\text{PBE}}$, relates to the value error. Let d_{obj} be the desired weighting on states, and d the weighting used in the surrogate objective. It is possible that $d \neq d_{\text{obj}}$ produces better values estimates for the $\overline{\text{VE}}$ weighted by d_{obj}. In fact, we already know several cases where that is true, where the linear $\overline{\text{PBE}}$ with $d_{\text{obj}} = d = d_b$ can result in arbitrarily bad $\overline{\text{VE}}$—even though it is weighted by the same d_b—but changing d to the emphatic weighting, $d = m$, prevents this bad outcome.

In this section, we characterize the solution quality under the generalized $\overline{\text{PBE}}$, which depends both on $\mathcal{H}$ and d. Let $v_{\boldsymbol{w}_{\mathcal{H},d}}$ be the solution the generalized $\overline{\text{PBE}}$. Our goal is to find bounds of the form

$$\|v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi\|_{d_{\text{obj}}} \leq C(d_{\text{obj}}, d, \mathcal{H})\|\Pi_{\mathcal{F},d} v_\pi - v_\pi\|_d$$

where the constant in the bound depends on the two weightings, and the projection set $\mathcal{H}$. The term $\|\Pi_{\mathcal{F},d} v_\pi - v_\pi\|_d$ represents the best approximation error we could have for our function class $\mathcal{F}$, if we were able to directly minimize the $\overline{\text{VE}}$ under our weighting d. This goal is similar to that of (Yu and Bertsekas, 2010, Equation 5), generalized to the nonlinear setting and where d may not equal d_{obj}. We start by describing such results when $\mathcal{H} = \mathcal{F}$, and then generalize to $\mathcal{H} \supset \mathcal{F}$ in the following subsection.

6.1 Upper Bound on $\overline{\text{VE}}$ when $\mathcal{H} = \mathcal{F}$

Throughout this section we will assume that $\mathcal{H} = \mathcal{F}$, so that the projection operator for both the objective and value function space is the same. This matches the setting original proposed for the $\overline{\text{PBE}}$, though here we explicitly assume that the projection can be nonlinear.

Our goal is to characterize the solution to the generalized $\overline{\text{PBE}}$, $v_{\boldsymbol{w}_{\mathcal{H},d}} = \Pi_{\mathcal{F},d}\mathcal{T} v_{\boldsymbol{w}_{\mathcal{H},d}}$. There are two steps to these results. The first step is to show that $\mathcal{T}$ is a contraction under norm $\|\cdot\|_d$, with contraction constant s_d. If $s_d < 1$, then it immediately follows that

$$\|v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi\|_d \leq (1 - s_d)^{-1}\|\Pi_{\mathcal{F},d} v_\pi - v_\pi\|_d \tag{20}$$

See (Bertsekas and Tsitsiklis, 1996, Lemma 6.9) or (White, 2017, Theorem 1) for this result. For the second step, we can define

$$\kappa(d_{\text{obj}}, d) \overset{\text{def}}{=} \max_{s\in\mathcal{S}} \frac{d_{\text{obj}}(s)}{d(s)} \tag{21}$$

and get that

$$\begin{aligned}
\|v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi\|^2_{d_{\text{obj}}} &= \sum_{s\in\mathcal{S}} d_{\text{obj}}(s)(v_{\boldsymbol{w}_{\mathcal{H},d}}(s) - v_\pi(s))^2 \\
&= \sum_{s\in\mathcal{S}} d_{\text{obj}}(s)\frac{d(s)}{d(s)}(v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi)^2 \\
&\leq \kappa(d_{\text{obj}}, d)\sum_{s\in\mathcal{S}} d(s)(v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi)^2 \\
&\leq \kappa(d_{\text{obj}}, d)\|v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi\|^2_d.
\end{aligned}$$

Combined with the inequality in Equation (20), we obtain

$$\|v_{\boldsymbol{w}_{\mathcal{H},d}} - v_\pi\|_{d_{\text{obj}}} \leq \sqrt{\kappa(d_{\text{obj}}, d)}(1 - s_d)^{-1}\|\Pi_{\mathcal{F},d}v_\pi - v_\pi\|_d$$

with $C(d_{\text{obj}}, d, \mathcal{H}) = \sqrt{\kappa(d_{\text{obj}}, d)}(1 - s_d)^{-1}$. Under $d = d_\pi$ and $d = m$, we know that $s_d < 1$ (White, 2017, Theorem 1).

These bounds identify three key sources of error in our value function approximation: the behavior-target mismatch, the contraction rate and the approximation error of our function class. If our function class contains the true value function v_π—the approximation error $\|\Pi_d v_\pi - v_\pi\|_d = 0$— then the $\overline{\text{VE}}$ is zero regardless of the contraction rate or behavior-target mismatch. If the behavior equals the target policy—the on-policy setting—then $\kappa(d_{\text{obj}}, d) = 1$; otherwise, it strictly increases the bound. The contraction constant is $s_d = \|P_{\pi,\gamma}\|_d$, where $P_{\pi,\gamma}(s, s') = \mathbb{E}_\pi[p(s'|s, A)\gamma(s, A, s')]$. For constant γ, $s_d = \|P_{\pi,\gamma}\|_d \leq \gamma$. More generally, for the episodic setting where γ is zero only for terminal transitions, and 1 otherwise, we do not as yet have a clear characterization of this constant.

6.2 Upper Bound on $\overline{\textbf{VE}}$ when $\mathcal{H} \supseteq \mathcal{F}$

We next extend this result to more general projections, i.e., for any function space $\mathcal{H} \supseteq \mathcal{F}$, that includes the $\overline{\text{BE}}$ and $\overline{\text{PBE}}$ as special cases. We start by re-expressing the generalized $\overline{\text{PBE}}$ as a weighted $\overline{\text{VE}}$, using the same approach as Schoknecht (2003) and Scherrer (2010). Let $v_{\boldsymbol{w}}$ be the vector consisting of value function estimates $v(s, \boldsymbol{w})$. Notice first that $v_\pi = (I - P_{\pi,\gamma})^{-1}r_\pi$. Then the generalized $\overline{\text{PBE}}$, written in projection form, is

$$\begin{aligned}
&\|\Pi_{\mathcal{H},d}(Tv_{\boldsymbol{w}} - v_{\boldsymbol{w}})\|^2_{d_{\text{obj}}} = \|\Pi_{\mathcal{H},d}(r_\pi + P_{\pi,\gamma}v_{\boldsymbol{w}} - v_{\boldsymbol{w}})\|^2_{d_{\text{obj}}} \\
&= \|\Pi_{\mathcal{H},d}(r_\pi - (I - P_{\pi,\gamma})v_{\boldsymbol{w}})\|^2_{d_{\text{obj}}} \\
&= \|\Pi_{\mathcal{H},d}[(I - P_{\pi,\gamma})v_\pi - (I - P_{\pi,\gamma})v_{\boldsymbol{w}}]\|^2_{d_{\text{obj}}} && \triangleright\ r_\pi = (I - P_{\pi,\gamma})v_\pi \\
&= \|\Pi_{\mathcal{H},d}(I - P_{\pi,\gamma})(v_\pi - v_{\boldsymbol{w}})\|^2_{d_{\text{obj}}} \\
&= \|v_\pi - v_{\boldsymbol{w}}\|^2_H && \triangleright H \overset{\text{def}}{=} (I - P_{\pi,\gamma})^\top \Pi^\top_{\mathcal{H},d} D \Pi_{\mathcal{H},d}(I - P_{\pi,\gamma})
\end{aligned}$$

with $v_{\boldsymbol{w}} \in \mathcal{F}$. Minimizing the generalized $\overline{\text{PBE}}$ therefore corresponds to minimizing the $\overline{\text{VE}}$ with a reweighting over states that may no longer be diagonal, as H is not a diagonal matrix. In fact, we can see that the solution to the generalized $\overline{\text{PBE}}$ is a projection of v_π onto set $\mathcal{F}$ under weighting H: $v = \Pi_{\mathcal{F},H}v_\pi$. A projection under such a non-diagonal weighting is called an oblique projection.

Using this form, we can obtain an upper bound using a similar approach to (Scherrer, 2010, Proposition 3).

Theorem 1 *If $\mathcal{H} \supseteq \mathcal{F}$, then the solution $v_{\boldsymbol{w}_{\mathcal{H},d}}$ to the generalized $\overline{PBE}$ satisfies*

$$\|v_\pi - v_{\boldsymbol{w}_{\mathcal{H},d}}\|_d \leq \|\Pi_{\mathcal{F},H}\|_d \|v_\pi - \Pi_{\mathcal{F},d}v_\pi\|_d. \tag{22}$$

Proof If $\Pi_{\mathcal{F},H}$ is the identity—a trivial projection—then the result immediately follows. This is because this implies $v_\pi \in \mathcal{F}$, and so both sides of the equation are zero.

Otherwise, assume $\Pi_{\mathcal{F},H}$ is a non-trivial projection. Notice that $\Pi_{\mathcal{F},H}\Pi_{\mathcal{F},d} = \Pi_{\mathcal{F},d}$, because the first projection already projects to the set $\mathcal{F}$ and so applying $\Pi_{\mathcal{F},H}$ has no effect. Now for any v,

$$\begin{aligned}
(I - \Pi_{\mathcal{F},H})(I - \Pi_{\mathcal{F},d})v &= (I - \Pi_{\mathcal{F},H} - \Pi_{\mathcal{F},d} + \Pi_{\mathcal{F},H}\Pi_{\mathcal{F},d})v \\
&= (I - \Pi_{\mathcal{F},H} - \Pi_{\mathcal{F},d} + \Pi_{\mathcal{F},d})v \\
&= (I - \Pi_{\mathcal{F},H})v
\end{aligned}$$

because by assumption $\Pi_{\mathcal{F},H}\Pi_{\mathcal{F},d} = \Pi_{\mathcal{F},d}$. Therefore we get that

$$\begin{aligned}
\|v_\pi - v\|_d &= \|v_\pi - \Pi_{\mathcal{F},H}v_\pi\|_d \\
&= \|(I - \Pi_{\mathcal{F},H})v_\pi\|_d \\
&= \|(I - \Pi_{\mathcal{F},H})(I - \Pi_{\mathcal{F},d})v_\pi\|_d \\
&\le \|I - \Pi_{\mathcal{F},H}\|_d \|(I - \Pi_{\mathcal{F},d})v_\pi\|_d \\
&= \|I - \Pi_{\mathcal{F},H}\|_d \|v_\pi - \Pi_{\mathcal{F},d}v_\pi\|_d \\
&= \|\Pi_{\mathcal{F},H}\|_d \|v_\pi - \Pi_{\mathcal{F},d}v_\pi\|_d
\end{aligned}$$

where the last step follows from the fact that for a non-trivial projection operator, $\|\Pi_{\mathcal{F},H}\|_d = \|I - \Pi_{\mathcal{F},H}\|_d$ (Szyld, 2006, Theorem 2.3). ■

Corollary 2 *If $\mathcal{H} = \mathcal{F}$ and d is such that $s_d \overset{def}{=} \|P_{\pi,\gamma}\|_d < 1$, then $\|\Pi_{\mathcal{F},H}\|_d \le (1 - s_d)^{-1}$.*

An important insight from the above is that there is a connection between convergence of TD—and generally under iteration of the projected Bellman operator—and the quality of the solution. We only have a bound on the value error when the norm of the projected operator has spectral radius less than 1. This is the same condition required to show that TD converges. This suggests that gradient methods only take us so far, since they prevent divergence but will find a poor solution. Instead, it's likely important to also ensure the weighting and features are controlled.

7. Experiment 1: Emphatic Weightings and the Generalized $\overline{\text{PBE}}$

In this section, we empirically investigate the quality of the solution under the $\overline{\text{PBE}}$ and $\overline{\text{BE}}$, when using emphatic weightings. We show the quality of the solution under three different weightings, d_b, d_π and m, in both the objective that is optimized and the evaluation objective d_{obj}, under different function spaces.

We compute the fixed-point of each objective on a 19-state random walk with randomly chosen target and behavior policies. To isolate the impact of representation on the fixed-points, we investigate several forms of state representation where v_π is outside the representable function class. We include the *Dependent* features from Sutton et al. (2009), randomly initialized sparse ReLu networks, tile-coded features, and state aggregation.

The random-walk has 19 states with the left-most and right-most state being terminal. The reward function is zero everywhere except on transitioning into the right-most terminal state where the agent receives +1 reward, and on the left-most terminal state where the agent receives -1 reward. The discount factor is set to $\gamma = 0.99$.

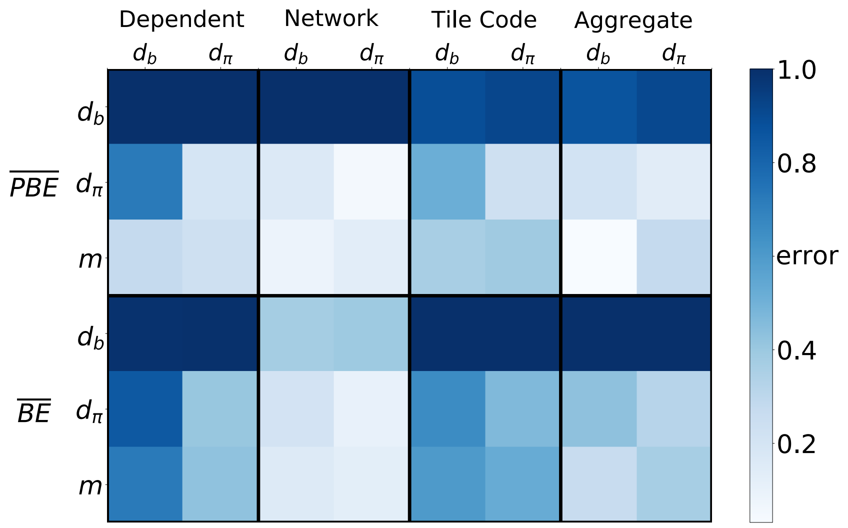

Figure 2: Investigating the fixed-points of $\overline{\text{PBE}}$ and $\overline{\text{BE}}$ under d_b, d_π, and m on a 19-state random walk. The fixed-point of the $\overline{\text{PBE}}$ with emphatic weighting consistently has the lowest error across several different state representations; while the fixed-point of the $\overline{\text{PBE}}$ under d_b has the highest error. Results are averaged over one million randomly generated policies and state representations.

We run each experimental setting one million times with a different randomly initialized neural network, random offset between tilings in the tile-coder, and randomly sampled target and behavior policy. The policies are chosen uniformly randomly on the standard simplex. The neural network is initialized with a Xavier initialization (Glorot and Bengio, 2010), using 76 nodes in the first hidden layer and 9 nodes in the final "feature" layer. Then 25% of the neural network weights are randomly set to zero to encourage sparsity between connections and to increase variance between different randomly generated representations. The tile-coder uses 4 tilings each offset randomly and each containing 4 tiles. The state aggregator aggressively groups the left-most states into one bin and the right-most states into another, creating only two features.

Figure 2 shows the normalized log-error of the fixed-points of $\overline{\text{PBE}}$ and $\overline{\text{BE}}$ under each weighting. The error is computed by subtracting the $\overline{\text{VE}}$ of the best representable value function, then scaling by the maximum $\overline{\text{VE}}$ among the fixed-points for a given feature representation (i.e. by the maximum for each column of Figure 2), thus yielding an error between zero and one for each column. The fixed-points are computed using their least-squares closed form solutions given knowledge of the MDP dynamics. Plotted is the mean error across the one million randomly initialized experimental settings. The standard error between settings is negligibly small.

Interestingly, the fixed-points corresponding to weighting d_b consistently have the highest error across feature representations, even on the excursion $\overline{\text{VE}}$ error metric. One notable exception is the fixed-point of the $\overline{\text{BE}}$ under d_b with the neural network feature representation, where the error is significantly less than under the corresponding $\overline{\text{PBE}}$. The $\overline{\text{PBE}}$ under emphatic weighting, m, consistently has the lowest error across all feature representations and both emphatic and alt-life weightings of the $\overline{\mathrm{VE}}$.

8. Objectives for Control

The previous development was strictly for policy evaluation. The formulation of a sensible generalized $\overline{\mathrm{PBE}}$ for control, however, can be obtained using a similar route. The conjugate form has already been used to develop a novel control algorithm for nonlinear function approximation, called SBEED (Dai et al., 2018). The SBEED algorithm explicitly maintains a value function and policy, to incorporate entropy regularization, and uses the saddlepoint update. We develop an alternative control algorithm, that learns only action-values and the gradient-correction update.

Assume now that we learn parameterized action-values $q(\cdot, \boldsymbol{w})$. Instead of the Bellman operator, we use the Bellman optimality operator or generalizations that use other forms of the max but are still guaranteed to be contractions, like the mellow-max operator (Asadi and Littman, 2017). Let m be the given max operator, that takes action-values and returns a (soft) greedy value. In Q-learning, we use a hard-max $m(q(s,\cdot)) = \max_a q(s,a)$ and in mellow-max, $m(q(s,\cdot)) = \beta^{-1} \log\left(\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \exp(\beta q(s,a))\right)$. As $\beta \to \infty$, the mellow-max operator approaches the hard-max operator.

The Bellman optimality operator $\mathcal{T}_m$ corresponds to

$$(\mathcal{T}_m q)(s,a) \stackrel{\text{def}}{=} \mathbb{E}_\pi\big[R + m(q(S',\cdot)) \mid S = s, A = a\big] \tag{23}$$

We can then define the $\overline{\mathrm{BE}}$ for control as

$$\overline{\mathrm{BE}}(\boldsymbol{w}) \stackrel{\text{def}}{=} \sum_{s,a} d(s,a) \left(\mathbb{E}_\pi\big[R + m(q(S',\cdot;\boldsymbol{w})) \mid S = s, A = a\big] - q(s,a;\boldsymbol{w})\right)^2 \tag{24}$$

for some weighting $d : \mathcal{S}\times\mathcal{A} \to [0,\infty)$. We override the notation for the weighting d, to make the connection to the previous objectives clear. Let $\delta(\boldsymbol{w}) = R + m(q(S',\cdot;\boldsymbol{w}))$. As above, we can use a dual form and get

$$\begin{aligned}
\overline{\mathrm{BE}}(\boldsymbol{w}) &= \sum_{s\in\mathcal{S},a\in\mathcal{A}} d(s,a) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a] - q(s,a;\boldsymbol{w})\right)^2 \\
&= \sum_{s\in\mathcal{S},a\in\mathcal{A}} d(s,a) \max_{h\in\mathbb{R}} \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a]\, h - h^2\right) && \triangleright \text{ conjugate function} \\
&= \max_{h\in\mathcal{F}_{\text{all}}} \sum_{s\in\mathcal{S},a\in\mathcal{A}} d(s,a) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a]\, h(s,a) - h(s,a)^2\right) && \triangleright \text{ interchangeability}
\end{aligned}$$

As before, this maximization can be rewritten as a minimization,

$$\begin{aligned}
&\operatorname*{arg\,max}_{h\in\mathcal{F}_{\text{all}}} \sum_{s\in\mathcal{S},a\in\mathcal{A}} d(s,a) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a]\, h(s,a) - h(s,a)^2\right) \\
&\qquad = \operatorname*{arg\,min}_{h\in\mathcal{F}_{\text{all}}} \sum_{s\in\mathcal{S},a\in\mathcal{A}} d(s,a) \left(\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a]\, h(s,a) - h(s,a)\right)^2
\end{aligned}$$

where the optimal $h^*(s,a) = \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s, A = a]$. Note that this is true, even if we use the hard-max operator rather than the mellow-max, even though the operator is no longer smooth. The mellow-max might still be a preferable choice, for a smoother optimization. Finally, in practice, we will learn an approximate h, from the set $\mathcal{H}$, resulting in a projected Bellman error.

The resulting objective is the first generalized $\overline{\text{PBE}}$ for learning action-values for control, with projection given by the choice of $\mathcal{H}$. The algorithm is a simple modification of the policy evaluation algorithms above. The update to $h(s,a)$ is still a gradient of a squared error to the TD error. Consider the gradient update for the generalized $\overline{\text{PBE}}$, for a given $h(s,a)$ with a stochastic sample $\delta(\boldsymbol{w})$ from $S = s, A = a$:

$$-\nabla_{\boldsymbol{w}}\delta(\boldsymbol{w})h(s,a) = h(s,a)[\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w}) - \gamma\nabla_{\boldsymbol{w}}m(q(S',\cdot;\boldsymbol{w}))]$$

with gradient-correction form

$$\delta(\boldsymbol{w})\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w}) - \gamma h(s,a)\nabla_{\boldsymbol{w}}m(q(S',\cdot;\boldsymbol{w}))$$

The primary difference in the update for $\boldsymbol{w}$ is that the second term—the gradient correction term—involves the gradient through the max operator m. For the hard-max operator, this results in a subgradient. The mellow-max operator, on the other hand, is differentiable with derivative, for $u = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\exp(\beta q(s,a;\boldsymbol{w}))$

$$\begin{aligned}
\frac{\partial}{\partial w_i}m(q(s,\cdot;\boldsymbol{w})) &= \beta^{-1}\frac{1}{u}\frac{\partial}{\partial w_i}u \\
&= \beta^{-1}\frac{1}{u}\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\frac{\partial}{\partial w_i}\exp(\beta q(s,a;\boldsymbol{w})) \\
&= \beta^{-1}\frac{1}{u}\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\beta\exp(\beta q(s,a;\boldsymbol{w}))\frac{\partial}{\partial w_i}q(s,a;\boldsymbol{w}) \\
&= \frac{1}{\sum_{a\in\mathcal{A}}\exp(\beta q(s,a;\boldsymbol{w}))}\sum_{a\in\mathcal{A}}\exp(\beta q(s,a;\boldsymbol{w}))\frac{\partial}{\partial w_i}q(s,a;\boldsymbol{w})
\end{aligned}$$

Now the question is if we still have the same cancellation of the second term, for the gradient-correction approach.

$$\begin{aligned}
-\nabla_{\boldsymbol{w}}\delta(\boldsymbol{w})h(s,a) &= (h(s,a) - \delta(\boldsymbol{w}) + \delta(\boldsymbol{w}))\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w}) - h(s,a)\gamma\nabla_{\boldsymbol{w}}m(q(S',\cdot;\boldsymbol{w})) \\
&= \delta(\boldsymbol{w})\nabla_{\boldsymbol{w}}q(s,a\boldsymbol{w}) + (h(s,a) - \delta(\boldsymbol{w}))\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w}) - h(s,a)\gamma\nabla_{\boldsymbol{w}}m(q(S',\cdot;\boldsymbol{w}))
\end{aligned}$$

Therefore, as before, we can conclude that we can drop this second term, as long as the optimal $h \in \mathcal{H}$ is representable as a linear function of $\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w})$. The fixed point for the gradient-correction updates that drop the term $(h(s,a) - \delta(\boldsymbol{w}))\nabla_{\boldsymbol{w}}q(s,a;\boldsymbol{w})$ will still converge to the same fixed point, if they converge. The key question that remains is, if the dynamical system produced by these equations does in fact converge.

9. Experiment 2: Control with the Generalized $\overline{\text{PBE}}$

In this section, we compare Q-learning, SBEED and QRC—the control algorithm for the generalized $\overline{\text{PBE}}$.

References

Kavosh Asadi and Michael L. Littman. An Alternative Softmax Operator for Reinforcement Learning. *International Conference on Machine Learning*, June 2017.

Leemon Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. *Machine Learning Proceedings 1995*, pages 30–37, 1995. doi: 10.1016/B978-1-55860-377-6.50013-X.

Dimitri P Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from Conditional Distributions via Dual Embeddings. *International Conference on Artificial Intelligence and Statistics*, page 10, 2017.

Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. SBEED: Convergent Reinforcement Learning with Nonlinear Function Approximation. *International Conference on Machine Learning*, page 10, 2018.

Christoph Dann, Gerhard Neumann, and Jan Peters. Policy Evaluation with Temporal Differences: A Survey and Comparison. *Journal of Machine Learning Research*, page 75, 2014.

Simon S. Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic Variance Reduction Methods for Policy Evaluation. *International Conference on Machine Learning*, June 2017.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *International Conference on Machine Learning*, June 2018.

Yihao Feng, Lihong Li, and Qiang Liu. A Kernel Loss for Solving the Bellman Equation. *Advances in Neural Information Processing Systems 32*, pages 15456–15467, 2019.

Sina Ghiassian, Andrew Patterson, Shivam Garg, Dhawal Gupta, Adam White, and Martha White. Gradient Temporal-Difference Learning with Regularized Corrections. *arXiv:2007.00611 [cs, stat]*, July 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

Leah Hackman. *Faster Gradient-TD Algorithms.* PhD thesis, 2013.

Assaf Hallak, Aviv Tamar, Remi Munos, and Shie Mannor. Generalized Emphatic Temporal Difference Learning: Bias-Variance Analysis. *AAAI*, page 7, 2016.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. *International Conference on Learning Representations*, November 2016.

Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

J Z Kolter. The Fixed Points of Off-Policy TD. *Advances in Neural Information Processing Systems*, page 9, 2011.

Michael L. Littman and Richard S Sutton. Predictive Representations of State. *Advances in Neural Information Processing Systems 14*, pages 1555–1561, 2002.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Proximal Gradient Temporal Difference Learning Algorithms. *International Joint Conference on Artificial Intelligence*, page 5, 2016.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-Sample Analysis of Proximal Gradient TD Algorithms. *International Conference on Uncertainty in Artificial Intelligence*, July 2020.

Hamid R. Maei, Csaba Szepesvári, Shalabh Bhatnagar, Doina Precup, David Silver, and Richard S. Sutton. Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation. *Advances in Neural Information Processing Systems 22*, pages 1204–1212, 2009.

Hamid Reza Maei. *Gradient Temporal-Difference Learning Algorithms.* PhD thesis, University of Alberta, Edmonton, 2011.

Sridhar Mahadevan, Bo Liu, Philip Thomas, Will Dabney, Steve Giguere, Nicholas Jacek, Ian Gemp, and Ji Liu. Proximal Reinforcement Learning: A New Theory of Sequential Decision Making in Primal-Dual Spaces. *arXiv:1405.6757 [cs]*, May 2014.

Ashique Rupam Mahmood, Huizhen Yu, and Richard S. Sutton. Multi-step Off-policy Learning Without Importance Sampling Ratios. *arXiv:1702.03006 [cs]*, February 2017.

Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and Efficient Off-Policy Reinforcement Learning. *Advances in Neural Information Processing Systems*, page 9, 2016.

Doina Precup, Richard S Sutton, and Satinder Singh. *Eligibility Traces for Off-Policy Policy Evaluation.* PhD thesis, 2000.

Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-Policy Temporal-Difference Learning with Function Approximation. *International Conference on Machine Learning*, page 9, 2001.

Bruno Scherrer. Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. *arXiv:1011.4362 [cs]*, November 2010.

Ralf Schoknecht. Optimality of Reinforcement Learning Algorithms with Linear Function Approximation. *Advances in Neural Information Processing Systems*, page 8, 2003.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, November 2018. ISBN 978-0-262-35270-3.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112 (1):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1.

Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, 2009. doi: 10.1145/1553374.1553501.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. *International Conference on Autonomous Agents and Multi-Agent Systems*, page 8, 2011.

Richard S Sutton, A Rupam Mahmood, and Martha White. An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning. *Journal of Machine Learning Research*, page 29, 2016.

Daniel B. Szyld. The many proofs of an identity on the norm of oblique projections. *Numerical Algorithms*, 42(3-4):309–323, 2006.

Brian Tanner and Richard S. Sutton. TD(λ) networks: Temporal-difference networks with eligibility traces. *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 888–895, 2005. doi: 10.1145/1102351.1102463.

Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. Convergent Tree Backup and Retrace with Function Approximation. *International Conference on Machine Learning*, page 10, 2018.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *International Conference on Machine Learning*, page 9, 2016.

Adam White. *Developing a Predictive Approach to Knowledge.* PhD thesis, 2015.

Adam White and Martha White. Investigating practical linear temporal difference learning. *International Conference on Autonomous Agents and Multi-Agent Systems*, March 2016.

Martha White. Unifying Task Specification in Reinforcement Learning. *International Conference on Machine Learning*, page 9, 2017.

Huizhen Yu. On Convergence of Emphatic Temporal-Difference Learning. page 28, 2015.

Huizhen Yu and Dimitri P Bertsekas. Error bounds for approximations from projected linear equations. *Mathematics of Operations Research*, 35(2):306–329, 2010.

Huizhen Yu, Ashique Rupam Mahmood, and Richard S. Sutton. On Generalized Bellman Equations and Temporal-Difference Learning. *Advances in Artificial Intelligence*, 10233: 3–14, 2017. doi: 10.1007/978-3-319-57351-9_1.

Appendix A. The Bias of the $\overline{\text{TDE}}$

In this section, we highlight that the $\overline{\text{TDE}}$ can impose significant bias on the value function solution, to reduce variance of the targets in the TD-error. This could have utility for reducing variance of updates in practice. But, when considering the optimal solution for the $\overline{\text{TDE}}$—as opposed to the optimization path to get there—it suggests that the $\overline{\text{TDE}}$ is a poor choice of the objective.

We can similarly write the $\overline{\text{TDE}}$ as a decomposition, in terms of both the $\overline{\text{BE}}$ and the $\overline{\text{PBE}}$. The $\overline{\text{TDE}}$ decomposes into the $\overline{\text{BE}}$ and a bias term due to correlation between samples

$$\begin{aligned}
&\mathbb{E}_\pi\Big[\big(R+\gamma V(S')-V(s)\big)^2 \mid S=s\Big]\\
&=\mathbb{E}_\pi\Big[\big(R+\gamma V(S')-\mathbb{E}_\pi\big[R+\gamma V(S')\mid S=s\big]+\mathbb{E}_\pi\big[R+\gamma V(S')\mid S=s\big]-V(s)\big)^2\mid S=s\Big]\\
&=\mathbb{E}_\pi\Big[\big(R+\gamma V(S')-\mathbb{E}_\pi\big[R+\gamma V(S')\mid S=s\big]\big)^2\mid S=s\Big]+\big(\mathbb{E}_\pi\big[R+\gamma V(S')\mid S=s\big]-V(s)\big)^2
\end{aligned}$$

giving

$$\overline{\text{TDE}}=\|V-\mathcal{T}V\|_D^2+\mathbb{E}\big[\mathbf{Var}[R+\gamma V(S')|S=s]\big] \tag{25}$$

This further yields an equality between the $\overline{\text{PBE}}$ and the $\overline{\text{TDE}}$

$$\overline{\text{TDE}}=\|V-\Pi_D\mathcal{T}V\|_D^2+\|\mathcal{T}V-\Pi_D\mathcal{T}V\|_D^2+\mathbb{E}\big[\mathbf{Var}[R+\gamma V(S')|S=s]\big] \tag{26}$$

This variance penalty encourages finding value functions that minimize the variance of the target. Notice that this connection exists in supervised learning as well, by simply considering the case where $\gamma=0$. The $\overline{\text{BE}}$ include terms $(\mathbb{E}_\pi[R\mid S=s]-V(s))^2$ and the $\overline{\text{TDE}}$ is the standard squared error $\mathbb{E}_\pi\Big[(R-V(s))^2\mid S=s\Big]$. In regression, this additional variance penalty has no impact on the optimal solution, because it does not include V: $\mathbb{E}[\mathbf{Var}[R+0\cdot V(S')|S=s]]=\mathbb{E}[\mathbf{Var}[R|S=s]]$.

Now the question is if it is useful to learn V that results in a lower-variance target. One benefit is that the estimator itself could have lower variance, and so for a smaller number of samples this could warrant the additional bias in V. TODO: think of an actual example. When comparing these objectives in their ideal forms, over all states under function approximation, it is an undesirable penalty on the value estimates. TODO: give example, maybe even with a simple environment where the fixed point is poor.

Appendix B. A Survey of Off-Policy Algorithms for the Linear $\overline{\text{PBE}}$

In this section, we describe the methods used in the empirical study that follows next. In particular, we discuss the optimization objective, and provide detailed update equations highlighting how prior or posterior corrections are used in each method. We begin with the Gradient-TD family of methods that minimize the excursion variant of the linear $\overline{\text{PBE}}$. We then discuss modifications on GTD(λ)—namely the Hybrid methods and the Saddlepoint methods. Then we discuss the second family of off-policy methods, the Emphatic methods. We conclude with a discussion of several methods that reduce variance of posterior corrections, using action-dependent bootstrapping.

B.1 Gradient Temporal Difference Learning

Gradient-TD methods were the first to achieve stability with function approximation using gradient descent (Sutton et al., 2009). This breakthrough was achieved by creating an objective function, the linear $\overline{\text{PBE}}$, and a strategy to sample the gradient of the linear $\overline{\text{PBE}}$. The negative of the gradient of the linear $\overline{\text{PBE}}$, with weighting $d = d_b$, can be written:

$$\begin{aligned}\nabla \text{linear } \overline{\text{PBE}}(\boldsymbol{w}) = \mathbb{E}_{d_b}\big[\delta(S,A,S')\boldsymbol{z}(S)\big] \\ - \mathbb{E}_{d_b}\big[\gamma(S')\boldsymbol{x}(S')\boldsymbol{x}(S)^\top\big]\mathbb{E}_{d_b}\big[\boldsymbol{x}(S)\boldsymbol{x}(S)^\top\big]^{-1}\mathbb{E}_{d_b}\big[\delta(S,A,S')\boldsymbol{z}(S)\big].\end{aligned} \tag{27}$$

Sampling this gradient is not straightforward due to the product of expectations. To resolve this issue, a second weight vector, $\boldsymbol{h}$, can be used to estimate $\mathbb{E}_{d_b}[\boldsymbol{x}_t\boldsymbol{x}_t^\top]^{-1}\mathbb{E}_{d_b}[\delta_t\boldsymbol{z}_t]$ and avoid the need for two independent samples. The resultant method, called GTD(λ), can be thought of as approximate stochastic gradient descent on the linear $\overline{\text{PBE}}$ and is specified by the following updated equations:

$$\begin{aligned}\boldsymbol{h}_{t+1} &\leftarrow \boldsymbol{h}_t + \alpha_h\big[\delta_t\boldsymbol{z}_t^\rho - (\boldsymbol{h}_t^\top\boldsymbol{x})\boldsymbol{x}_{t+1}\big] \\ \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\delta_t\boldsymbol{z}_t^\rho - \underbrace{\alpha\gamma_{t+1}(1-\lambda)(\boldsymbol{h}_t^\top\boldsymbol{z}_t^\rho)\boldsymbol{x}_{t+1}}_{\text{correction term}}\end{aligned} \tag{28}$$

The GTD(λ) algorithm has several important details that merit further discussion. The most notable characteristic is the second weight vector $\boldsymbol{h} \in \mathbb{R}^k$ that forms a quasi-stationary estimate of the last two terms in the gradient of the linear $\overline{\text{PBE}}$. The corresponding two-timescale analysis highlights that the learning rate parameter $\alpha_h \in \mathbb{R}$ should be larger than α, where the weights $\boldsymbol{w}$ change slower to enable $\boldsymbol{h}$ to obtain such a quasi-stationary estimate (Sutton et al., 2009). In practice, the best values of α and α_h are problem dependent, and the practitioner must tune them independently to achieve good performance (White, 2015; White and White, 2016). Another important detail is that the first part of the update to $\boldsymbol{w}$ corresponds to Off-policy TD(λ). When $\lambda = 1$, the second term—the correction term—is removed, making GTD(1) = TD(1). Otherwise, for smaller λ, the correction term plays a bigger role.

The GTD(λ) algorithm has been shown to be stable with linear function approximation. The GTD(λ) with $\lambda = 0$, also known as TDC, has been shown to converge in expectation with i.i.d sampling of states (Sutton et al., 2009). The convergence of Gradient-TD methods with $\lambda > 0$ was later shown in the Markov noise case with constant stepsize and stepsizes that approach zero in the limit (Yu et al., 2017).

The GTD2(λ) algorithm is related to GTD(λ), and can be derived starting from the gradient of the excursion linear $\overline{\text{PBE}}$ in Equation 27. The gradient of the linear $\overline{\text{PBE}}$ given in Equation 27 is an algebraic rearrangement of:

$$\nabla \text{linear } \overline{\text{PBE}}(\boldsymbol{w}) \;=\; \mathbb{E}_{d_b}\big[(\boldsymbol{x}(S)-\gamma(S')\boldsymbol{x}(S'))\boldsymbol{z}(S)^{\top}\big]\mathbb{E}_{d_b}\big[\boldsymbol{x}(S)\boldsymbol{x}(S)^{\top}\big]^{-1}\mathbb{E}_{d_b}\big[\delta(S,A,S')\boldsymbol{z}(S)\big].$$

As before, the last two terms can again be replaced by a secondary weight vector $\boldsymbol{h} \in \mathbb{R}^k$. The resultant expression

$$\mathbb{E}_{d_b}\big[(\boldsymbol{x}(S)-\gamma(S')\boldsymbol{x}(S'))\boldsymbol{z}(S)^{\top}\big]\boldsymbol{h},$$

can be sampled resulting in an algorithm that is similar to GTD(λ), but differs in its update to the primary weights:

$$\boldsymbol{w}_{t+1} \leftarrow \; \boldsymbol{w}_t + \alpha(\boldsymbol{h}_t^{\top}\boldsymbol{x}_t)\boldsymbol{x}_t - \alpha\gamma_{t+1}(1-\lambda)(\boldsymbol{h}_t^{\top}\boldsymbol{z}_t^{\rho})\boldsymbol{x}_{t+1}. \tag{29}$$

This update does not make use of the TD-error δ_t, except through the secondary weights $\boldsymbol{h}$. The GTD2(λ) algorithm performs stochastic gradient descent on the linear $\overline{\text{PBE}}$, unlike GTD(λ), which uses an approximate gradient, as we discuss further in Section B.3 when describing the Saddlepoint methods.

B.2 Hybrid TD methods

The Hybrid TD methods were created to achieve the data efficiency of TD(λ) when data is sampled on-policy, and the stability of Gradient-TD methods when the data is sampled off-policy. Early empirical experience with TD(0) and GTD(0) in on-policy problems suggested that TD(0) might be more sample efficient (Sutton et al., 2009). Later studies highlighted the need for additional empirical comparisons to fully characterize the relative strengths of GTD(λ) compared with TD(λ) (Dann et al., 2014; White and White, 2016).

Hybrid TD methods were first proposed by Maei (2011) and Hackman (2013) and were further developed to make use of eligibility traces by White and White (2016). The derivation of the method starts with the gradient of the excursion linear $\overline{\text{PBE}}$. Recall from Equation (9) that the linear $\overline{\text{PBE}}$ can be written $(\boldsymbol{b}-\mathbf{A}\boldsymbol{w})^{\top}\mathbf{C}^{-1}(\boldsymbol{b}-\mathbf{A}\boldsymbol{w})$. The matrix $\mathbf{C}$ is simply the weighting in the squared error for $\boldsymbol{b}-\mathbf{A}\boldsymbol{w}$. In fact, because we know every solution to the linear $\overline{\text{PBE}}$ satisfies $\boldsymbol{b}-\mathbf{A}\boldsymbol{w} = \mathbf{0}$, the choice of $\mathbf{C}$ asymptotically is not relevant, as long as it is positive definite. The gradient of the linear $\overline{\text{PBE}}$, $-\mathbf{A}^{\top}\mathbf{C}^{-1}(\boldsymbol{b}-\mathbf{A}\boldsymbol{w})$ can therefore be modified to $-\mathbf{A}^{\top}\mathbf{B}(\boldsymbol{b}-\mathbf{A}\boldsymbol{w})$, for any positive definite $\mathbf{B}$, and should still converge to the same solution(s).

In order to achieve a hybrid learning rule, this substitution must result in an update that reduces to the TD(λ) update when $\pi = b$. This can be achieved by setting $\mathbf{B} = \mathbb{E}_{d_b}\Bigg[\Big(\boldsymbol{x}_t - \gamma_{t+1}\boldsymbol{x}_{t+1}\Big)\boldsymbol{z}_t\Bigg]$, which is the $\mathbf{A}$ matrix for the behavior. Because this $\mathbf{B}$ is estimated with on-policy samples—since we are following b—we know $\mathbf{B}$ is positive semi-definite (Sutton, 1988), and positive definite under certain assumptions on the features. Further, when $b = \pi$, we have that $\mathbf{B} = \mathbf{A}^{-\top}$, giving update $-\mathbf{A}^{\top}\mathbf{B}(\boldsymbol{b}-\mathbf{A}\boldsymbol{w}) = \boldsymbol{b}-\mathbf{A}\boldsymbol{w}$. The TD($\lambda$)

update is a stochastic sample of expected update $\boldsymbol{b}-\mathbf{A}\boldsymbol{w}$, and so when HTD($\lambda$) uses a stochastic sample of $-\mathbf{A}^\top\mathbf{B}(\boldsymbol{b}-\mathbf{A}\boldsymbol{w})$ when $b=\pi$, it is in fact using the same update as TD(λ).

The HTD(λ) algorithm is:

$$\begin{aligned}\boldsymbol{h}_{t+1} &\leftarrow \boldsymbol{h}_t + \alpha_h\left[\delta_t \boldsymbol{z}_t^\rho - (\boldsymbol{x}_t - \gamma_{t+1}\boldsymbol{x}_{t+1})(\boldsymbol{h}_t^\top \boldsymbol{z}_t)\right] \\ \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha\left[\delta_t \boldsymbol{z}_t^\rho - (\boldsymbol{x}_t - \gamma_{t+1}\boldsymbol{x}_{t+1})(\boldsymbol{z}_t^\rho - \boldsymbol{z}_t)^\top \boldsymbol{h}_t\right] \end{aligned} \tag{30}$$

HTD(λ) has two eligibility trace vectors, with $\boldsymbol{z}$ being a conventional accumulating eligibility trace for the behavior policy. If $\pi = b$, then all the ρ_t are 1 and $\boldsymbol{z}_t = \boldsymbol{z}_t^\rho$, which causes the last term in the $\boldsymbol{w}$ update to be zero and the overall update reduces to the TD(λ) algorithm. The last term in the $\boldsymbol{w}$ update applies a correction to the usual Off-policy TD(λ).

Like GTD(λ), the HTD(λ) algorithm is a posterior correction method that should converge to the minimum of the excursion variant of the linear $\overline{\text{PBE}}$. No formal stochastic approximation results have been published, though the expected update is clearly convergent because $\mathbf{A}^\top\mathbf{B}\mathbf{A}$ is positive semi-definite. This omission is likely due to the mixed empirical results achieved with Hybrid TD methods Markov chains and random MDPs (Hackman, 2013; White and White, 2016).

B.3 Gradient-TD methods based on a saddlepoint formulation

Optimization of the linear $\overline{\text{PBE}}$ can be reformulated as a saddle point problem, yielding another family of stable Off-policy TD methods based on gradient descent. These include the original Proximal-GTD methods (Liu et al., 2016, 2020) methods, stochastic variance reduction methods for policy evaluation (Du et al., 2017), and gradient formulations of Retrace and Tree Backup (Touati et al., 2018). The linear $\overline{\text{PBE}}$ can be rewritten using convex conjugates:

$$\text{linear } \overline{\text{PBE}}(\boldsymbol{w}) = \min_{\boldsymbol{h}} (\boldsymbol{b} - \mathbf{A}\boldsymbol{w})^\top \boldsymbol{h} - \tfrac{1}{2}\|\boldsymbol{h}\|^2_{\mathbf{C}} \tag{31}$$

where the weighted norm $\|\boldsymbol{h}\|^2_{\mathbf{C}} = \boldsymbol{h}^\top\mathbf{C}\boldsymbol{h}$.

The utility of this saddlepoint formulation is that it removes the product of expectations, with the explicit addition of an auxiliary variable. This avoids the double sampling problem, since for a given $\boldsymbol{h}$, it is straightforward to sample $(\boldsymbol{b} - \mathbf{A}\boldsymbol{w})^\top\boldsymbol{h}$ (see Equation (8)) with sample $\delta_t \boldsymbol{z}_t^{\rho\top}\boldsymbol{h}$. It is similarly straightforward to sample the gradient of this objective for a given $\boldsymbol{h}$. Now this instead requires that this auxiliary variable $\boldsymbol{h}$ be learned. The resulting algorithm is identical to GTD2(0) when using stochastic gradient descent for this saddle point problem. This result is somewhat surprising, because GTD2(0) was derived from the gradient of the linear $\overline{\text{PBE}}$ using a quasi-stationary estimate of a proportion of the gradient.

The saddle point formulation—because it is a clear convex-concave optimization problem—allows for many algorithmic variants. For example, stochastic gradient descent algorithm for this convex-concave problem can incorporate accelerations, such as mirror-prox—as used by Liu et al. (2020) variance reduction approaches—as used (Du et al., 2017). This contrasts the original derivation for GTD2(λ), which used a quasi-stationary estimate and was not obviously a standard gradient descent technique. One such accelerated algorithm is Proximal GTD2(λ), described by the following update equations:

$$\boldsymbol{h}_{t+\frac{1}{2}} \leftarrow \boldsymbol{h}_t + \alpha_{\boldsymbol{h}}\left[\delta_t \boldsymbol{z}_t^{\rho} - (\boldsymbol{h}_t^{\top}\boldsymbol{x}_t)\boldsymbol{x}_t\right] \tag{32}$$

$$\boldsymbol{w}_{t+\frac{1}{2}} \leftarrow \boldsymbol{w}_t + \alpha(\boldsymbol{h}_t^{\top}\boldsymbol{x}_t)\boldsymbol{x}_t - \alpha\gamma_{t+1}(1-\lambda_{t+1})(\boldsymbol{h}_t^{\top}\boldsymbol{z}_t^{\rho})\boldsymbol{x}_{t+1} \tag{33}$$

$$\delta_{t+\frac{1}{2}} \overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}\boldsymbol{w}_{t+\frac{1}{2}}^{\top}\boldsymbol{x}_{t+1} - \boldsymbol{w}_{t+\frac{1}{2}}^{\top}\boldsymbol{x}_t \tag{34}$$

$$\boldsymbol{h}_{t+1} \leftarrow \boldsymbol{h}_t + \alpha_{\boldsymbol{h}}\left[\delta_{t+\frac{1}{2}} \boldsymbol{z}_t^{\rho} - (\boldsymbol{h}_{t+\frac{1}{2}}^{\top}\boldsymbol{x}_t)\boldsymbol{x}_t\right] \tag{35}$$

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha(\boldsymbol{h}_{t+\frac{1}{2}}^{\top}\boldsymbol{x}_t)\boldsymbol{x}_t - \alpha\gamma_{t+1}(1-\lambda_{t+1})(\boldsymbol{h}_{t+\frac{1}{2}}^{\top}\boldsymbol{z}_t^{\rho})\boldsymbol{x}_{t+1} \tag{36}$$

The double update to $\boldsymbol{w}$ and $\boldsymbol{h}$, denoted by subscripts $t + \frac{1}{2}$ and $t + 1$, is produced by applying the Stochastic Mirror-Prox acceleration (Juditsky et al., 2011) to the gradient descent update derived from Equation 31. We will refer to this algorithm by the shorthand name PGTD2 in the figures.

The saddle point formulation cannot be applied to derive an accelerated version of GTD(λ). Recall that GTD(λ) was obtained by reordering expectations in the gradient of the linear $\overline{\text{PBE}}$, and then using quasi-stationary estimates of different expected values. This alternative formulation cannot obviously be written as a saddle point problem—though it has nonetheless been shown to be convergent. Nevertheless, a heuristic approximation of accelerated Proximal GTD(λ) has been proposed (Liu et al., 2020), and its update equations are similar to that of Proximal GTD2(λ) with difference in updating the weight vector $\boldsymbol{w}$:

$$\boldsymbol{w}_{t+\frac{1}{2}} \leftarrow \boldsymbol{w}_t + \alpha\delta_t\boldsymbol{z}_t^{\rho} - \alpha\gamma_{t+1}(1-\lambda_{t+1})(\boldsymbol{h}_t^{\top}\boldsymbol{z}_t^{\rho})\boldsymbol{x}_{t+1} \tag{37}$$

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha\delta_{t+\frac{1}{2}}\boldsymbol{z}_t^{\rho} - \alpha\gamma_{t+1}(1-\lambda_{t+1})(\boldsymbol{h}_{t+\frac{1}{2}}^{\top}\boldsymbol{z}_t^{\rho})\boldsymbol{x}_{t+1} \tag{38}$$

We will refer to this algorithm by the shorthand name PGTD in the figures.

Both Proximal GTD(λ) and Proximal GTD2(λ) minimize the excursion variant of linear $\overline{\text{PBE}}$, as they assume $d = d_b$. The idea of the saddlepoint formulation, however, is more general and alternatives weightings could be considered, such as $d = d_\pi$ (shown in Table 1). The expectations in the linear $\overline{\text{PBE}}$ would simply change, and prior corrections would need to be incorporated to get an unbiased sample of $\boldsymbol{b} - \mathbf{A}\boldsymbol{w}$ weighted by d_π.

The practical utility of these methods for online estimation is still not well understood. Several of the accelerations mentioned above, such as the use of stochastic variance reduction strategies (Du et al., 2017), assume a batch learning setting. The online algorithms, as mentioned, all use variants of GTD2(λ), which seems to perform more poorly than GTD(λ) in practice (Touati et al., 2018). This saddle point formulation, however, does enable continued advances in online convex optimization to be ported to reinforcement learning. Additionally, this formulation allows analysis tools from optimization to be applied to the analysis of TD learning methods. For example, Touati et al. (2018) provided the first finite sample analysis for GTD2(λ), which is not possible with the original GTD2(λ) derivation based on the quasi-stationary secondary weights.

B.4 Off-policy learning with action-dependent boostrapping

A common concern with using importance sampling ratios is the possibility for high variance, due to large ratios.[3] Several methods have been introduced that control this variance, either by explicitly or implicitly avoiding the product of importance sampling ratios in the traces. The Tree Backup(λ) algorithm, which we call TB(λ), was the first off-policy method that did not explicitly use importance sampling ratios (Precup et al., 2000). This method decays traces more, incurring more bias; newer algorithms such as V-trace(λ) and ABQ(ζ) attempt to reduce variance but without decaying traces as much, and improve performance in practice. In this section, we describe the state-value prediction variants of TB(λ), V-trace(λ), and ABQ(ζ) that we investigate in our empirical study.

These three methods can all be seen as Off-policy TD(λ) with λ generalized from a constant to a function of state and action. This unification was highlighted by Mahmood et al. (2017) when they introduced ABQ. This unification makes explanation of the algorithms straightforward: each method simply uses a different action-dependent trace function $\lambda : \mathcal{S} \times \mathcal{A} \to [0, 1]$. All three methods were introduced for learning action-values; we present the natural state-value variants below.

We begin by providing the generic Off-policy TD algorithm with action-dependent traces. The key idea is to set $\lambda_t \overset{\text{def}}{=} \lambda(S_{t-1}, A_{t-1})$ such that $\rho_{t-1}\lambda_t$ is well-behaved. The Off-policy TD(λ) algorithm for this generalized trace function can be written[4]

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \alpha \rho_t \delta_t \boldsymbol{z}_t \\ \boldsymbol{z}_t &= \gamma_t \rho_{t-1} \lambda_t \boldsymbol{z}_{t-1} + \boldsymbol{x}_t, \end{aligned} \tag{39}$$

Now we can specify different algorithms using this generic variant of Off-policy TD(λ), by specifying different implementations of the λ function. Like Off-policy TD(λ), these algorithms all perform only posterior corrections.

TB(λ) is Off-policy TD(λ) with $\lambda_t = b_{t-1}\lambda$, for some tuneable constant $\lambda \in [0, 1]$. Replacing λ_t with $b_{t-1}\lambda$ in the eligibility trace update in Equation 39 simplifies as follows:

$$\begin{aligned} \boldsymbol{z}_t &= \gamma_t \frac{\pi_{t-1}}{b_{t-1}} b_{t-1} \lambda \boldsymbol{z}_{t-1} + \boldsymbol{x}_t \\ &= \gamma_t \pi_{t-1} \lambda \boldsymbol{z}_{t-1} + \boldsymbol{x}_t, \end{aligned} \tag{40}$$

and gives the state-value variant of TB(λ).

A simplified variant of the V-trace(λ) algorithm (Espeholt et al., 2018) can be derived with a similar substitution: $\lambda_t = \min\left(\frac{\bar{c}}{\pi_{t-1}}, \frac{1}{b_{t-1}}\right) \lambda b_{t-1}$, where $\bar{c} \in \mathbb{R}^+$ and $\lambda \in [0, 1]$ are

3. We would like to note that, to the best of our knowledge, variance issues due to importance sampling ratios have not been concretely demonstrated in the literature. This concern, therefore, is based on intuition and should be considered a hypothesis rather than a known phenomenon.

4. This update explicitly uses ρ_t in the update to $\boldsymbol{w}_{t+1}$. This contrasts the earlier Off-policy TD updates in Equation (4), which have ρ_t in the trace. These two forms are actually equivalent, in that the update to $\boldsymbol{w}$ is exactly the same. We show this equivalence in Appendix **??**. We use this other form here, to more clearly highlight the relationship between ρ_{t-1} and λ_t.

both tuneable constants. The eligibility trace update becomes:

$$\begin{aligned}
\boldsymbol{z}_t &= \gamma_t \min\left(\frac{\bar{c}}{\pi_{t-1}}, \frac{1}{b_{t-1}}\right) \lambda b_{t-1} \frac{\pi_{t-1}}{b_{t-1}} \boldsymbol{z}_{t-1} + \boldsymbol{x}_t \\
&= \gamma_t \min\left(\frac{\bar{c}}{\pi_{t-1}}, \frac{1}{b_{t-1}}\right) \lambda \pi_{t-1} \boldsymbol{z}_{t-1} + \boldsymbol{x}_t \\
&= \gamma_t \min\left(\frac{\bar{c}\pi_{t-1}}{\pi_{t-1}}, \frac{\pi_{t-1}}{b_{t-1}}\right) \lambda \boldsymbol{z}_{t-1} + \boldsymbol{x}_t \\
&= \gamma_t \min\left(\bar{c}, \rho_{t-1}\right) \lambda \boldsymbol{z}_{t-1} + \boldsymbol{x}_t, \qquad (41)
\end{aligned}$$

The parameter $\bar{c}$ is used to cap importance sampling ratios in the trace. Note that it is not possible to recover the full V-trace(λ) algorithm in this way. The more general V-trace(λ) algorithm uses an additional parameter, $\bar{\rho} \in \mathbb{R}^+$ that caps the ρ_t in the update to $\boldsymbol{w}_{t+1}$: $\min(\bar{\rho}, \rho_t)\delta_t \boldsymbol{z}_t$. When $\bar{\rho}$ is set to the largest possible importance sampling ratio, it does not affect ρ_t in the update to $\boldsymbol{w}_t$ and so we obtain the equivalence above. For smaller $\bar{\rho}$, however, V-trace(λ) is no longer simply an instance of Off-policy TD(λ). In the experiments that follow, we investigate this simplified variant of V-trace(λ) that does not cap ρ_t and set $\bar{c} = 1$ as done in the original Retrace algorithm.

ABTD(ζ) for $\zeta \in [0, 1]$ uses $\lambda_t = \nu_{t-1} b_{t-1}$, with the following eligibility trace update:

$$\begin{aligned}
\boldsymbol{z}_t &= \gamma_t \frac{\nu_{t-1}}{b_{t-1}} b_{t-1} \lambda \boldsymbol{z}_{t-1} + \boldsymbol{x}_t \\
&= \gamma_t \nu_{t-1} \pi_{t-1} \boldsymbol{z}_{t-1} + \boldsymbol{x}_t. \qquad (42)
\end{aligned}$$

with the following scalar parameters to define ν_t

$$\begin{aligned}
\nu_t &\overset{\text{def}}{=} \nu(\psi(\zeta), s_t, a_t) \overset{\text{def}}{=} \min\left(\psi(\zeta), \frac{1}{\max(b(a_t|s_t), \pi(a_t|s_t))}\right) \\
\psi(\zeta) &\overset{\text{def}}{=} 2\zeta\psi_0 + \max(0, 2\zeta - 1)(\psi_{\max} - 2\psi_0) \\
\psi_0 &\overset{\text{def}}{=} \frac{1}{\max_{s,a} \max(b(a|s), \pi(a|s))} \\
\psi_{\max} &\overset{\text{def}}{=} \frac{1}{\min_{s,a} \max(b(a|s), \pi(a|s))}.
\end{aligned}$$

The convergence properties of all three methods are similar to Off-policy TD(λ). They are not guaranteed to converge under off-policy sampling with weighting d_b and function approximation. With the addition of gradient corrections similar to GTD(λ), these algorithms are convergent. For explicit theoretical results, see Mahmood et al. (2017) for ABQ with gradient correction and Touati et al. (2018) for convergent versions of Retrace and Tree Backup.

B.5 Emphatic-TD learning

Emphatic Temporal Difference learning, ETD(λ), provides an alternative strategy for obtaining stability under off-policy sampling without computing gradients of the linear $\overline{\text{PBE}}$. The key idea is to incorporate some prior corrections so that the weighting d results in a positive definite matrix $\mathbf{A}$. Given such an $\mathbf{A}$, a TD(λ) algorithm—a semi-gradient algorithm—can be shown to converge. Importantly, this allows for a stable off-policy algorithm with only a single set of weights. Gradient-TD methods, on the other hand, use two stepsize parameters and two weight vectors to achieve stability. In this section, we describe two different variants of Emphatic-TD methods: ETD(λ) and ETD(λ, β), which was introduced to reduce the variance of ETD(λ).

ETD(λ) minimizes a variant of the linear $\overline{\mathrm{PBE}}$ defined in Equation 9, where the weighting d is defined based on the *followon* weighting. The followon reflects (discounted) state visitation under the target policy when doing excursions from the behavior: starting from states sampled according to d_b. The followon is defined as

$$f(s_t) \overset{\text{def}}{=} d_b(s_t) + \gamma(s_t) \sum_{s_{t-1}, a_{t-1}} d_b(s_{t-1}) \pi(a_{t-1}|s_{t-1}) P(s_t|s_{t-1}, a_{t-1}) + \dots . \tag{43}$$

The emphatic weighting then corresponds to $m(s_t) = d_b(s_t)\lambda + (1-\lambda) f(s_t)$. This is the weighting used in the linear $\overline{\mathrm{PBE}}$ in Equation 9, setting $d(s) = m(s)$.

The Emphatic TD(λ) algorithm is specified by the following equations:

$$\begin{aligned} F_t &\leftarrow \rho_{t-1}\gamma_t F_{t-1} + 1 \\ M_t &\leftarrow \lambda_t + (1-\lambda_t) F_t \\ \boldsymbol{z}_t^{\rho} &\leftarrow \rho_t \left(\gamma_t \lambda \boldsymbol{z}_{t-1}^{\rho} + M_t \boldsymbol{x}_t\right) \\ \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha \delta_t \boldsymbol{z}_t^{\rho}, \end{aligned}$$

with $F_0 = 1$ and $\boldsymbol{z}_0^{\rho} = \mathbf{0}$. The scalar estimate F_t is used to include the weighting defined in Equation 43. To gain some intuition for this weighting, consider a setting where $\gamma_t = \gamma$ is constant and $\lambda = 0$. Then $M_t = F_t = \sum_{j=0}^{t} \gamma^j \prod_{i=1}^{j} \rho_{t-i}$, giving trace $\boldsymbol{z}_t^{\rho} \leftarrow \rho_t \left(\gamma_t \lambda \boldsymbol{z}_{t-1}^{\rho} + \sum_{j=0}^{t} \gamma^j \prod_{i=1}^{j} \rho_{t-i} \boldsymbol{x}_t\right)$.

There are some similarities to the weighting in the Alternative-life TD(λ) trace in Equation 5, where $\boldsymbol{z}_t^{\rho} \leftarrow \rho_t \left(\gamma_t \lambda \boldsymbol{z}_{t-1}^{\rho} + \prod_{i=1}^{t} \rho_i \boldsymbol{x}_t\right)$. Both adjust the weighting on $\boldsymbol{x}_t$ to correct for—or adjust—the state distributions. Alternative-Life TD more aggressively downweights states that would not have been visited under the target policy. ETD, on the other hand, reweights based on how frequently the states would be seen when starting π as an excursion from b.

Emphatic TD(λ) has strong convergence guarantees in the case of linear function approximation. The ETD(λ) under off-policy training has been shown to converge in expectation using the same expected update analysis used to show that TD(λ) converges under on-policy training. Later, Yu (2015) extended this result to show that ETD(λ) converges with probability one. Perhaps more practically relevant, this weighting also resolves the issues raised by Kolter's example (2011). Kolter's example demonstrated that for a particular choice of π and b, the solution to the linear $\overline{\mathrm{PBE}}$ could result in arbitrarily bad error compared with the best possible approximation in the function class. In other words, even if the true value function can be well approximated by the function class, the off-policy fixed point from the linear $\overline{\mathrm{PBE}}$ with weighting $d = d_b$ can result in an arbitrarily poor approximation to the values. Hallak et al. (2016) showed that the fixed points of the linear $\overline{\mathrm{PBE}}$ with the emphatic weighting, on the other hand, do not suffer from this problem (see their Corollary 1). This result was actually generally shown for an extended ETD(λ) method, called ETD(λ, β), which we describe next.

ETD(λ) was extended to include an additional scalar tuneable parameter β, to further control variance due to prior corrections. The ETD(λ, β) algorithm updates are identical to ETD(λ) except for the update to F_t:

$$F_t \leftarrow \rho_{t-1}\beta F_{t-1} + 1$$

If $\beta = \gamma$, then the update is identical to ETD(λ). If $\beta = 0$, then the update is identical to Off-policy TD(λ), and there are a spectrum of methods in between. This β, then, introduces bias to reduce variance in the followon trace for ETD. Hallak et al. (2016) showed that β can be less than γ, and ETD(λ, β) can still enjoy the convergence properties of ETD(λ), depending on the mismatch between the target and behavior policy. For more similar policies, β can be closer to zero and still converge: it can behave like Off-policy TD(λ) and still converge if the setting is almost on-policy. For a greater mismatch, β must be nearer to γ.