Taylor TD-learning

Michele Garibbo [1] **Maxime Robeyns** [1] **Laurence Aitchison** [1]

Abstract

Many reinforcement learning approaches rely on temporal-difference (TD) learning to learn a critic. However, TD-learning updates can be high variance due to their sole reliance on Monte Carlo estimates of the updates. Here, we introduce a model-based RL framework, Taylor TD, which reduces this variance. Taylor TD uses a first-order Taylor series expansion of TD updates. This expansion allows to analytically integrate over stochasticity in the action-choice, and some stochasticity in the state distribution for the initial state and action of each TD update. We include theoretical and empirical evidence of Taylor TD updates being lower variance than (standard) TD updates. Additionally, we show that Taylor TD has the same stable learning guarantees as (standard) TD-learning under linear function approximation. Next, we combine Taylor TD with the TD3 algorithm (Fujimoto et al., 2018), into TaTD3. We show TaTD3 performs as well, if not better, than several state-of-the art model-free and model-based baseline algorithms on a set of standard benchmark tasks. Finally, we include further analysis of the settings in which Taylor TD may be most beneficial to performance relative to standard TD-learning.

1. Introduction

Actor-critic algorithms underlie many of the recent successes of deep RL (Fujimoto et al., 2018; Haarnoja et al., 2018; Lillicrap et al., 2015; Schulman et al., 2015b;a; 2017; Silver et al., 2014; Voelcker et al., 2022). In these algorithms, the actor provides the control policy while the critic provides estimates of the policy's expected long-term returns (i.e. a value function; Barto et al., 1983; Konda & Tsitsiklis, 1999). The critic is typically trained using some form of temporal-difference (TD) update (e.g. Lillicrap et al., 2015; Silver et al., 2014; Fujimoto et al., 2018; Haarnoja et al., 2018; Schulman et al., 2017). These TD updates need to be computed in expectation over a large distribution of visited states and actions, induced by the policy and the environment dynamics (Sutton, 1988; Sutton & Barto, 2018). Since this expectation is analytically intractable, TD updates are typically performed based on individually sampled state-action pairs from real environmental transitions (i.e. Monte Carlo (MC) estimates). However, the variance of (MC) TD updates can be quite large, meaning that we need to average over many TD updates for different initial states and actions to get a good estimate of the expected updates (Fairbank & Alonso, 2011).

Model-based strategies provide a promising candidate to tackle this high variance (Kaelbling et al., 1996). For instance, Dyna methods, among the most popular model-based strategies, use a learned model of the environment transitions to generate additional imaginary transitions. These imaginary transitions can be used as extra training samples for TD methods (e.g. Sutton, 1990; Gu et al., 2016; Feinberg et al., 2018; Janner et al., 2019; D'Oro & Jaśkowski, 2020; Buckman et al., 2018). Although the additional (imaginary) transitions help in reducing the variance in the expected TD updates, Dyna methods still rely on the same, potentially high-variance (MC) TD-updates as standard TD-learning.

We address the issue of high-variance TD-updates by formulating an expected TD-update over a small distribution of state-action pairs. We show this expected update can be analytically estimated with a first-order Taylor expansion, in an approach we call *Taylor TD*. By analytically estimating this expected update, rather than exclusively relying on MC estimates (as in e.g. Dyna), we show both theoretically and empirically to achieve lower variance TD updates. Additionally, we show Taylor TD does not affect the stable learning guarantees of TD-learning under linear function approximation (for a fixed policy as shown by Tsitsiklis & Van Roy, 1996). Next, we propose a model-based off-policy algorithm, Taylor TD3 (TaTD3), which uses Taylor TD in combination with the TD3 algorithm (Fujimoto et al., 2018). We show TaTD3 performs as well as if not better than several state-of-the art model-free and model-based baseline algorithms on a set of standard benchmark tasks. Finally, we compare TaTD3 to its "Dyna" equivalent, which exclusively

[1]Department of Engineering Mathematics, University of Bristol, Bristol, United Kingdom. Correspondence to: Michele Garibbo <michele.garibbo@bristol.ac.uk>.

Accepted at the ICML Frontiers4LCD workshop, Honolulu, Hawaii, USA. 2023.

relies on MC TD-updates. We found the largest benefits of Taylor TD may appear in high dimensional state-action spaces.

2. Related work

Model-based strategies provide a promising solution to improving the sample complexity of RL algorithms (Kaelbling et al., 1996). In Dyna methods, a model of the environment transitions is learned through interactions with the environment and then employed to generate additional imaginary transitions, for instance in the form of model roll-outs (Sutton, 1990). These imaginary transitions, can be used to enhance existing model-free algorithms, leading to improved sample complexity. For example, within TD-learning, imaginary transitions can be used to train a Q-function by providing additional training examples (e.g. Sutton, 1990; Gu et al., 2016; D'Oro & Jaśkowski, 2020). Alternatively, imaginary transitions can be used to provide better TD targets for existing data points (e.g. Feinberg et al., 2018) or to train the actor and/or critic by generating short-horizon trajectories starting at existing state-action pairs (e.g. Janner et al., 2019; Clavera et al., 2020; Buckman et al., 2018). These (Dyna) approaches have a clear relation to our approach (Taylor TD), as they attempt to estimate the same expected TD-update in Eq. (7). However, Dyna approaches only use potentially high-variance MC estimates, while Taylor TD exploits analytic results to reduce that variance.

Conceptually, our approach may resemble previous methods that also rely on analytical computations of expected updates to achieve lower-variance critic or policy updates (e.g. Ciosek & Whiteson, 2018; Whiteson, 2020; Van Seijen et al., 2009; Sutton & Barto, 2018; Asadi et al., 2017). The most well-known example of this is Expected-SARSA. Expected-SARSA achieves a lower variance TD-update (relative to SARSA), by analytically computing the expectation over the distribution of target actions in the TD-update (i.e. assuming a stochastic target policy) (Van Seijen et al., 2009; Sutton & Barto, 2018);

$$\delta_\theta(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\, \mathrm{E}_{a'\sim\pi}\left[Q_\theta(\mathbf{s}',\mathbf{a}')\right] - Q_\theta(\mathbf{s},\mathbf{a}) \quad (1)$$

This approach can only reduce variance of TD-updates at the level of the target actions, a', induced by a stochastic target policy. In the case of a deterministic target policy, Expected-SARSA does not provide any benefit. Conversely, our approach attempts to reduce the variance at the level of the initial state-action pairs, (s,a) at which TD-updates are evaluated. That is; we take the expectation over (s,a) instead of a' (see Eq. 7 and 19) which yields benefits with both stochastic and deterministic target policies. Other RL approaches exploiting analytical computations of expected updates are Expected Policy Gradients (Ciosek & Whiteson, 2018; Whiteson, 2020) and Mean Actor Critic (Asadi et al., 2017). Both methods attempt to reduce the variance of the stochastic policy gradient update by integrating over the action distribution. Although similar in principle to our approach, these two methods focus on the policy update instead of the critic update and, similarly to Expected-SARSA only apply to stochastic target policies.

In practice, our approach may relate to value gradient methods, as it explicitly incorporates the gradient of the value function into the update (e.g. Heess et al., 2015; Clavera et al., 2020; Amos et al., 2021; D'Oro & Jaśkowski, 2020; Balduzzi & Ghifary, 2015; Fairbank & Alonso, 2012). To our knowledge, the value gradient approach that most relates to our work is MAGE (D'Oro & Jaśkowski, 2020), which, nonetheless, has a radically different motivation from Taylor TD. MAGE is motivated by noting that the action-gradients of Q drive deterministic policy updates (Silver et al., 2014), so getting the action-gradients right is critical for policy learning. In order to encourage the action-gradients of Q to be correct, MAGE explicitly adds a term to the objective that consists of the norm of the action-gradient of the TD-error, which takes it outside of the standard TD-framework. In contrast, our motivation is to reduce the minibatch gradient variance of standard TD updates by performing some analytic integration. We do this through a first-order Taylor expansion of the TD update. This difference in motivation leads to numerous differences in the method and analysis, the least of which is that MAGE uses only the action-gradients, while Taylor TD additionally suggests using the state-gradients, as both the state and action gradients can be used to reduce the variance in the minibatch updates.

3. Background

Reinforcement learning aims to learn reward-maximising behaviour by interacting with the surrounding environment. At each discrete time step, t, the agent in state $\mathbf{s} \in \mathcal{S}$, chooses an action $\mathbf{a} \in \mathcal{A}$ based on a policy $\pi : \mathcal{S} \to \mathcal{A}$, and observes a scalar reward, r and a new state $\mathbf{s}' \in \mathcal{S}$ from the environment. The agent's goal is to find the policy that maximises the expected sum of rewards (i.e. the expected return), from a distribution of initial states (or state-action pairs). As such, it is usually necessary to compute the expected return for a state-action pair $(\mathbf{s},\mathbf{a})$ and a policy π; which we can do with a value function. Given a policy π and an initial state-action pair $(\mathbf{s},\mathbf{a})$, we define the value function $Q^\pi(\mathbf{s},\mathbf{a}) = \mathrm{E}\left[R_t \mid S_t = \mathbf{s}, A_t = \mathbf{a}\right]$, where $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ is the discounted sum of future rewards from the current time step t until termination T, with discount factor $\gamma \in [0,1]$. The value function or critic, Q^π, quantifies how good the policy, π, is in terms of its expected return when taking action $\mathbf{a}$ in state $\mathbf{s}$ and following the policy π thereafter.

To estimate the value function for a policy π, we must

usually interact with the environment. The most popular approach to do so is temporal difference (TD) learning (Sutton, 1988), which is based on the Bellman equation (Bellman, 1966). Assuming the (true) underlying critic Q^π is approximated by a differentiable function approximation Q_θ, we can write the overall TD-update over the entire distribution of visited state-action pairs as:

$$\mathrm{E}\left[\Delta\theta\right] = \mathrm{E}_{\mathbf{s}\sim d^\pi;\mathbf{a}\sim\pi}\left[\delta_\theta(\mathbf{s},\mathbf{a})\nabla_\theta Q_\theta(\mathbf{s},\mathbf{a})\right] \tag{2}$$

where d^π is the state distribution induced by policy π, and

$$\delta_\theta(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma Q_\theta(\mathbf{s}',\mathbf{a}') - Q_\theta(\mathbf{s},\mathbf{a}) \tag{3}$$

Here, $\delta_\theta(\mathbf{s},\mathbf{a})$ represents the TD-error for TD(0), although the same expected update applies for any TD method by adjusting the TD-error term (e.g. TD(n), TD(λ) Sutton, 1988; van Seijen et al., 2015). Note that in off-policy learning, this expectation is computed over the off-policy state distribution d^{π_b}, and the behavioural policy π_b while $\mathbf{a}'$ still comes from the target policy π

$$E\left[\Delta\theta\right] = \mathrm{E}_{\mathbf{s}\sim d^{\pi_\mathrm{b}};\mathbf{a}\sim\pi_\mathrm{b}}\left[\delta_\theta(\mathbf{s},\mathbf{a})\nabla_\theta Q_\theta(\mathbf{s},\mathbf{a})\right] \tag{4}$$

where Q_θ still approximates the target (policy) Q-function Q^π. Conversely, in model-based off-policy learning, the initial action, $\mathbf{a}$, in the TD update may be sampled from a third policy which does not correspond to either the behavioral π_b or the target policy π. This enables us to explore the value of additional actions independently from the behavioural policy, thanks to the model generating transitions for these additional actions. In practice, analytically computing the expected updates in Eq. (2) and (4) is intractable, due to the complex underlying state-action pair distributions (beyond requiring access to the environment dynamics). Hence, TD-learning methods typically employ a Monte Carlo (MC) (i.e. sampled-based) estimate of these expected updates. For instance, at each time step t, a TD-update is computed based on state-action pairs that are sampled from the environment, a replay buffer (i.e. off-policy) or a model (or any combination of those three).

4. Taylor TD-learning

As mentioned above, we can not analytically compute the (expected) TD update over the entire distribution of state-action pairs (e.g. Eq. 2). However, we can reduce the variance by combining MC estimates with some analytic integration. For instance, at time step t, we can consider an expected TD-update over a distribution over actions. We could compute this expected update for continuous actions being drawn from a Gaussian distribution with mean $\mathbf{a}_t$ and covariance $\mathbf{\Sigma}_\mathrm{a}$. In particular, we can do this by re-parametrizing the action, $\mathbf{a}$, at which we evaluate the TD-update, in terms of a (deterministic) action at time t (i.e. $\mathbf{a}_t$), plus some zero-mean Gaussian random noise, $\boldsymbol{\xi}_\mathrm{a}$, with covariance $\mathbf{\Sigma}_\mathrm{a}$.

$$\mathbf{a} = \mathbf{a}_t + \boldsymbol{\xi}_\mathrm{a} \tag{5}$$

$$\mathrm{E}_{\boldsymbol{\xi}_\mathrm{a}}\left[\boldsymbol{\xi}_\mathrm{a}\right] = \mathbf{0} \qquad \mathrm{E}_{\boldsymbol{\xi}_\mathrm{a}}\left[\boldsymbol{\xi}_\mathrm{a}\boldsymbol{\xi}_\mathrm{a}^T\right] = \mathbf{\Sigma}_\mathrm{a} \tag{6}$$

The expected TD update, averaging over actions from the Gaussian distributed policy is,

$$\mathrm{E}_{\boldsymbol{\xi}_\mathrm{a}}[\Delta\theta_t] = \eta\,\mathrm{E}_{\boldsymbol{\xi}_\mathrm{a}}[\delta_\theta(\mathbf{s}_t,\mathbf{a}_t+\boldsymbol{\xi}_\mathrm{a})\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t+\boldsymbol{\xi}_\mathrm{a})] \tag{7}$$

Standard TD-learning updates, which sample actions from a (Gaussian) policy, can be understood as computing MC estimates of this expected update. However, these MC estimates would likely be high variance (e.g. Ciosek & Whiteson, 2018), leading to slow learning.

We should stress that $\mathbf{\Sigma}_\mathrm{a}$, the covariance of initial actions in this expected TD-update does not necessarily need to match the covariance of the behavioural policy. For instance, $\mathbf{\Sigma}_\mathrm{a}$ could be larger than the behavioural policy covariance, enabling to learn the value of broader actions that are not taken in the environment (i.e. assuming knowledge of δ_θ and Q_θ is available for those actions).

4.1. Action expansion

Here, we show we can analytically approximate the expected-TD update in Eq. (7), using a first-order Taylor expansion. The full expectation is taken over $\boldsymbol{\xi}_a$,

$$\mathrm{E}\left[\Delta\theta_t\right] = \eta\,\mathrm{E}\Big[\delta_\theta(\mathbf{s}_t,\mathbf{a}_t+\boldsymbol{\xi}_\mathrm{a})\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t+\boldsymbol{\xi}_\mathrm{a})\Big] \tag{8}$$

Applying the first-order Taylor expansion,

$$\begin{aligned}\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\,\mathrm{E}\Big[&\Big(\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{a}^T\nabla_\mathbf{a}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big) \\ &\nabla_\theta\Big(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{a}^T\nabla_\mathbf{a}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big)\Big]\end{aligned} \tag{9}$$

As ∇_θ is a linear operator, and $\boldsymbol{\xi}_\mathrm{a}$ does not depend on θ,

$$\begin{aligned}\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\,\mathrm{E}\Big[&\Big(\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{a}^T\nabla_\mathbf{a}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big) \\ &\Big(\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{a}^T\nabla^2_{\mathbf{a},\theta}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big)\Big]\end{aligned} \tag{10}$$

where $\nabla^2_{\mathbf{a},\theta}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)$ is a matrix of second derivatives. The expectation of $\boldsymbol{\xi}_\mathrm{a}$ is zero, so the terms linear in $\boldsymbol{\xi}_\mathrm{a}$ are zero, leading to,

$$\begin{aligned}\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = &\eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) \\ &+ \eta\,\mathrm{E}\Big[\Big(\boldsymbol{\xi}_\mathrm{a}^T\nabla_\mathbf{a}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big)\Big(\boldsymbol{\xi}_\mathrm{a}^T\nabla^2_{\mathbf{a},\theta}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\Big)\Big]\end{aligned} \tag{11}$$

Swapping the order of the terms in the expectation,

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\,\mathrm{E}\left[\left(\boldsymbol{\xi}_{\mathrm{a}}^T\nabla^2_{\mathbf{a},\theta}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\right)\left(\boldsymbol{\xi}_{\mathrm{a}}^T\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \tag{12}$$

And transposing the first term in the expectation (which we can do as it is overall a scalar),

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\,\mathrm{E}\left[\left(\nabla^2_{\theta,\mathbf{a}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\boldsymbol{\xi}_{\mathrm{a}}\right)\left(\boldsymbol{\xi}_{\mathrm{a}}^T\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \tag{13}$$

We can then move the terms independent of $\boldsymbol{\xi}_{\mathrm{a}}$ out of the expectation:

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\nabla^2_{\theta,\mathbf{a}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\,\mathrm{E}\left[\boldsymbol{\xi}_{\mathrm{a}}\boldsymbol{\xi}_{\mathrm{a}}^T\right]\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) \tag{14}$$

Finally, we know $\mathrm{E}\left[\boldsymbol{\xi}_{\mathrm{a}}\boldsymbol{\xi}_{\mathrm{a}}^T\right] = \boldsymbol{\Sigma}_{\mathrm{a}}$,

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\nabla^2_{\theta,\mathbf{a}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\boldsymbol{\Sigma}_{\mathrm{a}}\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) \tag{15}$$

If we assume the action covariance is isotropic, $\boldsymbol{\Sigma}_{\mathrm{a}} = \lambda_{\mathrm{a}}\mathbf{I}$, we get the following (1st-order) Taylor TD-update estimating the expected TD-update formulated in Eq. 7:

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\lambda_{\mathrm{a}}\nabla^2_{\theta,\mathbf{a}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) \tag{16}$$

The first term is the standard TD update with state $\mathbf{s}_t$ and action $\mathbf{a}_t$. The new second term tries to align the action gradient of the critic (Q-function) with the action gradient of the TD target. Conceptually, this gradient matching should help reduce the variance across TD-updates since it provides a way to estimate the expected update in Eq. (7). In the appendix, we include a proof that at least under linear function approximation, these extra Taylor gradient terms do not affect the stability of TD-learning, assuming λ_a and η are chosen in a certain way (see Appendix B). Critically, even with linear function approximation, there are errors in the first-order Taylor expansion, as we use a nonlinear function to transform actions into a feature vector, while the Taylor expansion is taken with respect to the underlying action. Nevertheless, we provide theoretical and empirical evidence that the first-order Taylor expansion reduces the variance of standard TD-updates and support efficient learning, even under non-linear function approximation (see sections, 4.4, 5.1 and 5.2.4).

4.2. State expansion

We are not limited to formulating an expected TD-update over a distribution of actions, but we can expand this to a distribution of states too. Namely, instead of performing a TD-update at the single state location, $\mathbf{s}_t$, we perform this update over a distribution of states. We take this distribution to be Gaussian with mean at $\mathbf{s}_t$ and covariance $\boldsymbol{\Sigma}_{\mathrm{s}}$. To do so, we can re-write the state at time t as:

$$\mathbf{s} = \mathbf{s}_t + \boldsymbol{\xi}_{\mathrm{s}} \tag{17}$$

where $\boldsymbol{\xi}_{\mathrm{s}}$ is a Gaussian random variable with mean zero and covariance $\boldsymbol{\Sigma}_{\mathrm{s}}$,

$$\mathrm{E}_{\boldsymbol{\xi}_{\mathrm{s}}}\left[\boldsymbol{\xi}_{\mathrm{s}}\right] = \mathbf{0} \qquad \mathrm{E}_{\boldsymbol{\xi}_{\mathrm{s}}}\left[\boldsymbol{\xi}_{\mathrm{s}}\boldsymbol{\xi}_{\mathrm{s}}^T\right] = \boldsymbol{\Sigma}_{\mathrm{s}} \tag{18}$$

Based on this, we can formulate an expected TD-update, averaging over this Gaussian distribution of states.

$$\mathrm{E}_{\boldsymbol{\xi}_{\mathrm{s}}}[\Delta\theta_t] = \eta\,\mathrm{E}_{\boldsymbol{\xi}_{\mathrm{s}}}[\delta_\theta(\mathbf{s}_t+\boldsymbol{\xi}_{\mathrm{s}},\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t+\boldsymbol{\xi}_{\mathrm{s}},\mathbf{a}_t)] \tag{19}$$

Again, we can approximate this expected update with a first-order Taylor approximation, but this time, expanding it around $\mathbf{s}_t$. Based on a similar derivation to the action expansion, we get the following update (see Appendix B.1.1 for the full derivation):

$$E_{\boldsymbol{\xi}_{\mathrm{s}}}[\Delta_{\mathrm{Ta}}\theta_t] = \eta\delta_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t,\mathbf{a}_t) + \eta\lambda_s\nabla^2_{\theta,\mathbf{s}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\nabla_{\mathbf{s}}\delta_\theta(\mathbf{s}_t,\mathbf{a}_t) \tag{20}$$

The rational behind this update is trying to tackle some of the TD-update variance induced by the (visited) state distribution, although we expect this only to work for states close-by to the visited ones (i.e. for small values of λ_{s})

4.3. State-Action expansion

Finally, we can combine the two Taylor expansions into a single TD-update involving both state and action expansions. Nevertheless, computing the dot products between $\nabla\delta_\theta$ and ∇Q_θ terms for both state and action terms may not be optimal. One reason for this is dot products are unbounded, increasing the risk of high variance (TD) updates (e.g. Luo et al., 2018). To tackle this issue, we use cosine distances between the gradient terms instead of dot products (see Appendix G.2 for the benefits of this). The cosine distance has the advantage of being bounded. By putting everything together, we propose a novel TD update, which we express below in terms of a loss:

$$\begin{aligned}\mathcal{L}_\theta = \;& \eta\delta(\mathbf{s}_t,\mathbf{a}_t)Q_\theta(\mathbf{s}_t,\mathbf{a}_t) \\ & + \eta\lambda_{\mathrm{a}}\,\mathrm{CosineSimilarity}(\nabla_{\mathbf{a}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t),\;\nabla_{\mathbf{a}}\delta(\mathbf{s}_t,\mathbf{a}_t)) \\ & + \eta\lambda_{\mathrm{s}}\,\mathrm{CosineSimilarity}(\nabla_{\mathbf{s}}Q_\theta(\mathbf{s}_t,\mathbf{a}_t),\;\nabla_{\mathbf{s}}\delta(\mathbf{s}_t,\mathbf{a}_t))\end{aligned} \tag{21}$$

Note we used the notation δ instead of δ_θ to indicate we are treating $\delta(\mathbf{s}_t,\mathbf{a}_t)$ as a fixed variable independent of θ.

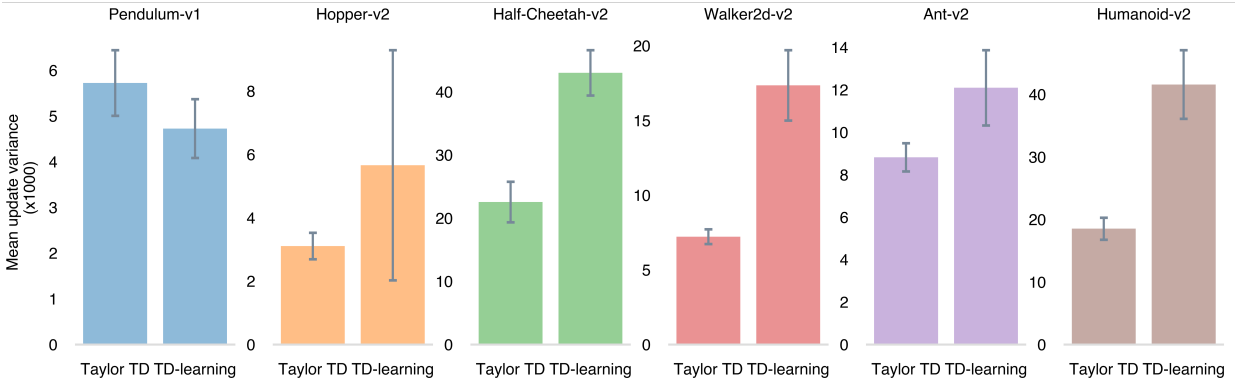

Figure 1. Mean update variance between Taylor TD and standard (MC) TD-learning (batch) updates, based on several sampled states and the distribution of actions for those states (i.e. the policy). All results are based on 10 runs with corresponding error bars.

Algorithm 1 Taylor TD

Initialise reply buffer $\mathcal{B}$
Initialise model, critic, target policy parameters $\{w, \theta, \phi\}$
Initialise $\zeta_\text{a}, \zeta_\text{s} = 0$
for each training step **do**
  Collect transition $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ according to π_b
  $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$
  for each model update step **do**
    $w \leftarrow w - \eta_m \nabla_w \mathcal{L}(\mathbf{s}, \mathbf{a}, r, \mathbf{s}'), \quad (\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim \mathcal{B}$
  end for
  for each policy update step **do**
    $(\mathbf{s}, \cdot, \cdot, \cdot) \sim \mathcal{B}$
    $\mathbf{a} = \pi_\phi(\mathbf{s})$
    $\hat{r}, \hat{\mathbf{s}}' \sim p_w(\cdot \mid \mathbf{s}, \mathbf{a})$
    $\delta = \hat{r} + \gamma Q_\theta(\hat{\mathbf{s}}', \pi_\phi(\hat{\mathbf{s}}')) - Q_\theta(\mathbf{s}, \mathbf{a})$
    if Action expansion **then**
      $\zeta_\text{a} = \text{CosineSimilarity}(\nabla_\mathbf{a} Q_\theta(\mathbf{s}, \mathbf{a}),\ \nabla_\mathbf{a} \delta(\mathbf{s}, \mathbf{a}))$
    end if
    if State expansion **then**
      $\zeta_\text{s} = \text{CosineSimilarity}(\nabla_\mathbf{s} Q_\theta(\mathbf{s}, \mathbf{a}),\ \nabla_\mathbf{s} \delta(\mathbf{s}, \mathbf{a}))$
    end if
    $\theta \leftarrow \theta + \eta_c \delta\ \nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a}) + \eta_c \lambda_\text{a} \nabla_\theta \zeta_\text{a} + \eta_c \lambda_\text{s} \nabla_\theta \zeta_\text{s}$
    $\phi \leftarrow \phi + \eta_p \nabla_\phi Q_\theta(\mathbf{s}, \mathbf{a})$
  end for
end for

This ensures when we take the gradient of this loss relative to θ, we do not differentiate through any δ terms (following the standard implementation of TD-updates in autodiff frameworks such as PyTorch, see Appendix D). It should be noted Taylor TD requires a differentiable model of the environment transitions as well as reward function in order to compute $\nabla_\mathbf{a} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)$ and $\nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)$ (see Appendix C). In principle, Taylor TD can be used with any actor-critic approach that relies on TD-learning, and even be extended to Monte Carlo returns. However, in practice, computing $\nabla \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)$ over long horizons of states and actions will suffer from the same exploding/vanishing gradient problem as backpropagating rewards through several transitions (e.g. Clavera et al., 2020; Xu et al., 2022). Therefore, we implement Taylor TD within a TD(0) set-up and expect it to work best with short-horizon TD updates. We provide a Taylor TD algorithm implementation with this set-up in the Algorithm box 1.

4.4. Variance analysis

Here, we show that the Taylor TD update in Eq. (16) has lower variance than standard (MC) TD-updates over the same distribution of actions. We only provide this variance analysis for the distribution over actions, because analogous theorems can be derived for the distribution over states (i.e. Eq. 20). To begin, we apply the law of total variance, to standard TD-updates,

$$\text{Var}\left[\Delta\theta_t\right] = \text{E}_{\mathbf{s}_t}\left[\text{Var}_\pi\left[\Delta\theta_t | \mathbf{s}_t\right]\right] + \text{Var}_{\mathbf{s}_t}\left[\text{E}_\pi\left[\Delta\theta_t | \mathbf{s}_t\right]\right] \tag{22}$$

Recall that the updates, $\Delta\theta_t = \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$, depend on the starting state, s_t, and action, $a_t \sim \pi(\cdot|\mathbf{s}_t)$. The inner expectation and inner variance sample actions from the policy, π, while the outer expectation and outer variance sample states from the distribution of visited states. To relate this expression to Taylor TD, recall that Taylor TD updates are motivated as performing analytic integration over actions from the policy, π, using a first-order Taylor expansion based approximation (i.e. assuming π corresponds to the re-parameterize actions in Eq. 5),

$$\Delta_\text{Ta}\theta_t \approx \text{E}_\pi\left[\Delta\theta_t | \mathbf{s}_t\right] = \Delta_\text{Exp}\theta_t \tag{23}$$

Here, we defined $\Delta_{\text{Exp}}\theta_t$ as the exact expected update, averaging over actions. Thus, the variance of standard (MC) TD-updates is exactly the variance of $\Delta_{\text{Exp}}\theta_t$, plus an additional term to account for variance induced by sampling actions,

$$\text{Var}\left[\Delta\theta_t\right] = \text{E}_{\mathbf{s}_t}\left[\text{Var}_{\pi}\left[\Delta\theta_t|\mathbf{s}_t\right]\right] + \text{Var}_{\mathbf{s}_t}\left[\Delta_{\text{Exp}}\theta_t\right] \quad (24)$$

This directly gives a theorem,

Theorem 4.1. *The variance for standard (MC) TD-updates is larger than the variance of $\Delta_{Exp}\theta_t$ that arises from exact integration over actions (Eq. 7),*

$$\text{Var}\left[\Delta\theta_t\right] \geq \text{Var}_{\mathbf{s}_t}\left[\Delta_{\text{Exp}}\theta_t\right] \quad (25)$$

Of course, we ultimately seek to relate the variance of the standard (MC) TD updates, $\Delta\theta_t$ to the Taylor-TD updates, $\Delta_{\text{Ta}}\theta_t$ which involve some degree of approximation from the first order Taylor expansion. While at first, it might seem that we are only able to get an approximate relationship,

$$\text{Var}\left[\Delta\theta_t\right] \approx \text{E}_{\mathbf{s}_t}\left[\text{Var}_{\pi}\left[\Delta\theta_t|\mathbf{s}_t\right]\right] + \text{Var}_{\mathbf{s}_t}\left[\Delta_{\text{Ta}}\theta_t\right] \quad (26)$$

we can in actuality obtain a formal relationship by considering differentiable $\Delta\theta_t = \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$. If $\Delta\theta_t$ is differentiable then the Taylor series expansion becomes increasingly accurate as we consider smaller regions around the mean action, which correspond to smaller variances, λ_{a}, in the distribution over actions.

Theorem 4.2. *If $\Delta\theta_t(\mathbf{s}_t, \mathbf{a}_t) = \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ is a continuous and differentiable function of $\mathbf{a}_t$, and if for all $s_t \in S$ $\text{Var}_{\pi}\left[\Delta\theta_t|\mathbf{s}_t\right] > \epsilon$ for some $\epsilon > 0$, and if we truncate the distribution over actions at some multiple of the standard deviation (e.g. sampled actions cannot be more than 10 standard deviations from the mean) then there exists $\lambda_{\text{a}} > 0$ for which*

$$\text{Var}\left[\Delta\theta_t\right] > \text{Var}_{\mathbf{s}_t}\left[\Delta_{\text{Ta}}\theta_t\right] \quad (27)$$

5. Experiments

5.1. Variance reduction

In this section we empirically test the claim that Taylor TD updates are lower variance than standard (MC) TD-learning updates. To do so, we compute "batch updates" (Sutton & Barto, 2018), where given an approximate value function Q_θ and a policy π, several Q_θ updates are computed across several sampled states and actions, updating Q_θ only once, based on the sum of all updates. Batch updates ensure the variance of the updates is estimated based on the same underlying value function. We compute batch updates for both Taylor TD and standard (MC) TD updates, comparing the variance of the updates between the two approaches (see Appendix E for more details).

Fig. (1) shows Taylor TD provides reliably lower variance updates compared to standard TD-learning across all tested tasks, but the Pendulum environment. This finding is consistent with the idea that Taylor TD updates may be most beneficial in higher dimensional state-action spaces, while the Pendulum environment represents the lowest dimensional environment with only four dimensions This is because MC sampling (i.e., standard TD-learning) may become less efficient in higher dimensional spaces. We further explore this proposal in a toy example comparing MC estimates and Taylor expansions of expected updates. We perform this comparison across data points of different dimensions, and find the benefits of the Taylor expansion (over MC) seem to increase with the dimension of the data points (see Appendix F).

5.2. Comparison with baselines

5.2.1. Algorithm

We implement Taylor TD (i.e. Algorithm 1) with the TD3 algorithm (Fujimoto et al., 2018) in a model-based off-policy algorithm we denote as Taylor TD3 (TaTD3). TaTD3 aims to provide a state-of-the-art implementation of Taylor TD for comparison with baseline algorithms. At each iteration, TaTD3 uses a learned model of the transitions and learned reward function to generate several differentiable (imaginary) 1-step transitions, starting from real states (sampled from a reply buffer). These differentiable 1-step transitions are used to train two critics (i.e. TD3) with several Taylor TD updates in a hybrid value gradient and Dyna approach. The model of the transitions consists of an ensemble of 8 Gaussian models trained by maximum likelihood on the observed environment transitions. This model ensemble aims to reduce over-fitting and model biases (Deisenroth & Rasmussen, 2011). The reward function is a Neural Network trained with mean-square error on the observed environment rewards. Hence, TaTD3 does not require any a priori knowledge of the environment transitions or reward function. Crucially, we found we could get good performance for TaTD3 across all tested environments without needing to fine tune the value of λ_{a} and λ_{s} to each environment (see Appendix I). Finally, the actor is trained with the deterministic policy gradient (Silver et al., 2014) on real states as in standard TD3 (Fujimoto et al., 2018).

5.2.2. Environments

The first environment consists of a classic problem in control theory used to evaluate RL algorithms (i.e. Pendulum, Brockman et al., 2016). The other 5 environments are stanard MuJoCo continous control tasks (i.e. Hopper, HalfCheetah, Walker2d, Ant and Humanoid, Todorov et al., 2012). All results are reported in terms of average performance across 5 runs, each with a different random seed

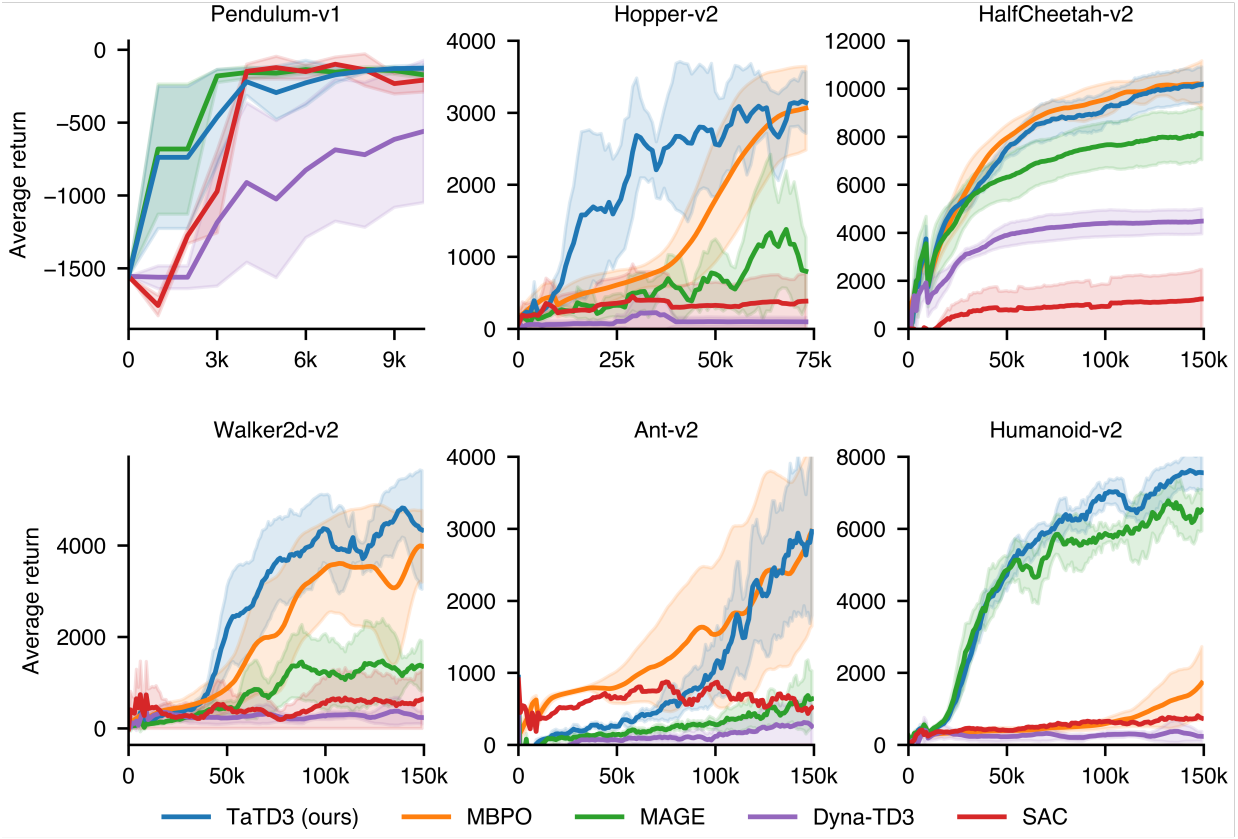

Figure 2. Performance in terms of average returns for TaTD3 and four state-of-the-art baseline algorithms on six benchmark continuous control tasks. TaTD3 performs as well, if not better, than the four baseline algorithms on all six tasks. All performance are based on 5 runs, with shade representing 95% c.i.

(shade represents 95% CI).

5.2.3. Code

All the code is available at `https://anonymous.4open.science/r/TaylorTD-7E61`

5.2.4. Results

Here, we report the comparison of TaTD3 with some state-of-the art model -free and -based baselines on the four benchmark environments. These baselines include 3 model-based algorithms and one model-free algorithm. The first model-based algorithm is Model-based Policy Optimization (MBPO) (Janner et al., 2019), which employs the soft actor-critic algorithm (SAC) (Haarnoja et al., 2018) within a model-based Dyna setting. Plotted performance of MBPO was directly taken from the official algorithm repository on GitHub. The second model-based algorithm is Model-based Action-Gradient-Estimator Policy Optimization (MAGE) (D'Oro & Jaśkowski, 2020), which uses a differentiable model of the environment transitions to train the critic by minimising the norm of the action-gradient of the TD-error. The third model-based algorithm is TD3 combined with a model-based Dyna approach (i.e. Dyna-TD3). This algorithm was proposed by D'Oro & Jaśkowski (2020) and was shown to outperform its model-free counterpart, TD3 (Fujimoto et al., 2018) on most benchmark tasks. Dyna-TD3 is conceptually similar to MBPO, with the main difference of MBPO relying on SAC instead of TD3. Plotted performances of both MAGE and Dyna-TD3 were obtained by re-running these algorithms on the benchmark environments, taking the implementations from the official algorithms' repository. Finally, we included SAC (Haarnoja et al., 2018) as a model-free baseline. Plotted performance of SAC was obtained by running the Stable Baselines implementation of this algorithm on the six benchmark environments (Hill et al., 2018).

Fig. (2) shows TaTD3 performs at least as well, if not better, than the baseline algorithms in all six benchmark tasks: note the much poorer performance of MAGE on Hopper-v2, Walker2d-v2 and Ant-v2, of MBPO on Humanoid-v2 relative to TaTD3.

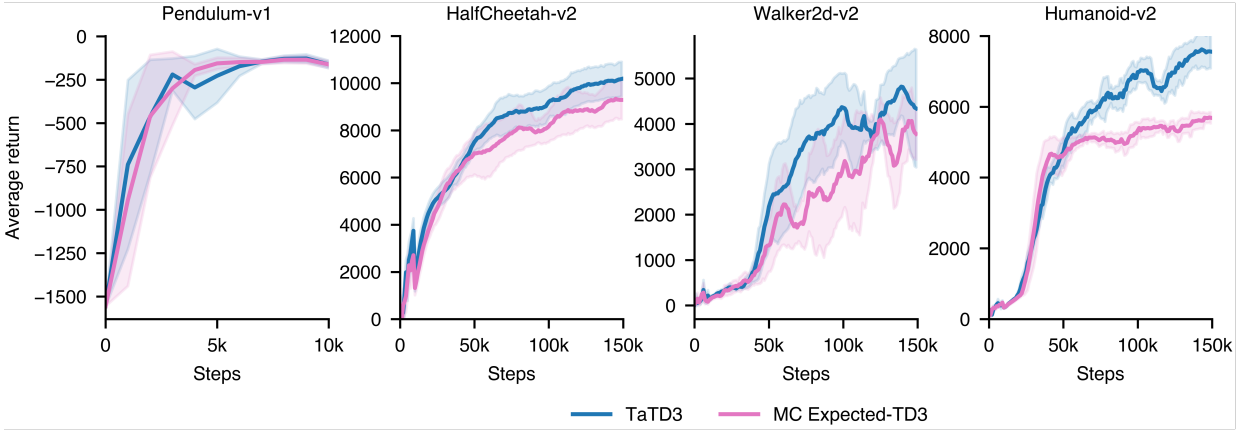

Figure 3. Performance comparison of TaTD3 with its Monte Carlo equivalent, MC Expected-TD3. All performance are based on 5 runs, with shade representing 95% c.i.

5.3. Taylor vs MC-sampling TD-learning

Next, we ask whether Taylor TD provides any performance benefit in computing the expected TD updates in Eq. (7) and (19) over standard MC estimates. To do so, we implement a model-based TD3 algorithm analogous to TaTD3, but where the expected TD updates, Eq. (7) and (19), are estimated by sampling several state and action perturbations at each time step (i.e. instead of being analytically computed through the Taylor expansions). We denote this algorithm MC Expected-TD3 (available at `https://anonymous.4open.science/r/TaylorTD-7E61`). In practice, at each time step, MC Expected-TD3 uses a (learned) model of the transitions to compute multiple TD-updates by sampling several state perturbations of visited states and action perturbations of the current policy (i.e. estimating Eq. (7) and (19) through MC estimates). Crucially, we ensure the variance of the state and action perturbations (i.e. λ_a and λ_s) is matched between TaTD3 and MC Expected-TD3. In Fig. (3), we can see TaTD3 provides performance benefits over MC Expected-TD3 across the three most difficult environments. Interestingly, the benefit of the Taylor expansion (i.e TaTD3) over MC sampling (i.e MC Expected-TD3) may be more evident in high dimensional state-action spaces. Indeed, the largest performance advantage of TaTD3 is seen in Humanoid-v2, which has the highest dimensional state-action space by a large margin. Conversely, the least advantage of TaTD3 over MC Expected-TD3 is seen in Pendulum-v1, which is the task with smallest dimensional state-action space. We further explore this proposal in a toy example comparing MC estimates and Taylor expansions of expected updates. We perform this comparison across data points of different dimensions, and find the benefits of the Taylor expansion (over MC) seem to increase with the dimension of the data points (see Appendix F).

Finally, we should point out MC Expected-TD3 is different from Dyna-TD3, as the latter does not implement any action or state perturbation in the TD-updates. Hence, unlike MC Expected-TD3, Dyna-TD3 does not compute the expected updates in Eq. (7) and (19), but relies on standard TD-learning (This is also evident in the massive performance difference between Dyna-TD3 and MC Expected-TD3 - i.e. see Fig. 2).

6. Conclusion and Limitations

In this article, we introduce a model-based RL framework, Taylor TD, to help reduce the variance of standard TD-learning updates and, speed-up learning of critics. We theoretically and empirically show Taylor TD updates are lower variance than standard (MC) TD-learning updates. We show the extra gradient terms used by Taylor TD do not affect the stable learning guarantees of TD-learning with linear function approximation under a reasonable assumption. Next, we combine Taylor-TD with the TD3 algorithm (Fujimoto et al., 2018) into a model-based off-policy algorithm we denote as TaTD3. We show TaTD3 performs as well, if not better, than several state-of-the art model-free and model-based baseline algorithms on a set of standard benchmark tasks.

Taylor TD has the limitation that it requires a differentiable model of transitions to calculate the additional (TD) gradient terms, i.e. it must be in the model-based rather than model-free setting (see Appendix C). Additionally, the gradient terms in Taylor TD imply additional computational cost; in the Appendix H, we show this cost is not that large in terms of training times and we expect it to reduce as faster automatic differentiation tools are developed (Baydin et al., 2018).

Acknowledgements

This work made use of the HPC system Blue Pebble at the University of Bristol, UK.

References

Amos, B., Stanton, S., Yarats, D., and Wilson, A. G. On the model-based stochastic value gradient for continuous reinforcement learning. In *Learning for Dynamics and Control*, pp. 6–20. PMLR, 2021.

Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., Littman, M., and Amazon, B. U. Mean actor critic. *stat*, 1050:1, 2017.

Balduzzi, D. and Ghifary, M. Compatible value gradients for reinforcement learning of continuous deep policies. *arXiv preprint arXiv:1509.03005*, 2015.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, pp. 834–846, 1983.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Bellman, R. Dynamic programming. *Science*, 153(3731): 34–37, 1966.

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

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

Ciosek, K. and Whiteson, S. Expected policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Clavera, I., Fu, V., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. *arXiv preprint arXiv:2005.08068*, 2020.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

D'Oro, P. and Jaśkowski, W. How to learn a useful critic? model-based action-gradient-estimator policy optimization. *Advances in Neural Information Processing Systems*, 33:313–324, 2020.

Drucker, H. and Le Cun, Y. Improving generalization performance using double backpropagation. *IEEE transactions on neural networks*, 3(6):991–997, 1992.

Fairbank, M. and Alonso, E. The local optimality of reinforcement learning by value gradients, and its relationship to policy gradient learning. *arXiv preprint arXiv:1101.0428*, 2011.

Fairbank, M. and Alonso, E. Value-gradient learning. In *The 2012 international joint conference on neural networks (ijcnn)*, pp. 1–8. IEEE, 2012.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pp. 2829–2838. PMLR, 2016.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Luo, C., Zhan, J., Xue, X., Wang, L., Ren, R., and Yang, Q. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, pp. 382–391. Springer, 2018.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

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

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Tsitsiklis, J. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pp. 177–184. IEEE, 2009.

van Seijen, H., Mahmood, A. R., Pilarski, P. M., and Sutton, R. S. An empirical evaluation of true online td ({\lambda}). *arXiv preprint arXiv:1507.00353*, 2015.

Voelcker, C., Liao, V., Garg, A., and Farahmand, A.-m. Value gradient weighted model-based reinforcement learning. *arXiv preprint arXiv:2204.01464*, 2022.

Whiteson, S. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21, 2020.

Xu, J., Makoviychuk, V., Narang, Y., Ramos, F., Matusik, W., Garg, A., and Macklin, M. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*, 2022.

A. Appendix

B. Proof of stable learning for Taylor TD with linear function approximation (with a fixed policy)

The Taylor TD update for the action expansion can be written as (an equivalent proof can be derived for the state expansion):

$$\Delta_{\text{Ta}}\theta(\mathbf{s},\mathbf{a}) = \delta_\theta(\mathbf{s},\mathbf{a})\nabla_\theta Q_\theta(\mathbf{s},\mathbf{a}) + \lambda_{\text{a}}\nabla^2_{\theta,\mathbf{a}} Q_\theta(\mathbf{s},\mathbf{a})\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s},\mathbf{a}) \tag{28}$$

This update is composed of two quantities, the standard TD update (i.e., first term in the sum) plus the extra term induced by the Taylor expansion (i.e., second term in the sum). For the purposes of this proof, we consider linear function approximation,

$$Q_\theta(\mathbf{s},\mathbf{a}) = \theta^T\mathbf{x} \qquad\qquad Q_\theta(\mathbf{s}',\mathbf{a}') = \theta^T\mathbf{x}' \tag{29}$$

where,

$$\mathbf{x} = \phi(\mathbf{s},\mathbf{a}) \in \mathbb{R}^N \qquad\qquad \mathbf{x}' = \phi(\mathbf{s}',\mathbf{a}') \in \mathbb{R}^N \tag{30}$$

and where $\mathbf{x}$ and $\mathbf{x}'$ are feature-vectors of length N. We can re-write each of the two terms in the Taylor TD update in terms of this linear function approximation. The first term, corresponding to a standard TD-update can be written,

$$\begin{aligned}\delta_\theta(\mathbf{s},\mathbf{a})\nabla_\theta Q_\theta(\mathbf{s},\mathbf{a}) &= (r + \gamma\theta^T\mathbf{x}' - \theta^T\mathbf{x})\mathbf{x} \\ &= r\mathbf{x} - \mathbf{x}(\mathbf{x} - \gamma\mathbf{x}')^T\theta.\end{aligned} \tag{31}$$

The second term, which is the new term introduced by Taylor-TD methods, can be written,

$$\begin{aligned}\nabla^2_{\theta,\mathbf{a}} Q_\theta(\mathbf{s},\mathbf{a})\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s},\mathbf{a}) &= \nabla_{\theta,\mathbf{a}}\left(\mathbf{x}^T\theta\right)\left(\nabla_{\mathbf{a}} r + \gamma\nabla_{\mathbf{a}}(\mathbf{x}'^T\theta) - \nabla_{\mathbf{a}}(\mathbf{x}^T\theta)\right) \\ &= (\nabla_{\mathbf{a}}\mathbf{x})^T\left(\nabla_{\mathbf{a}} r + \gamma(\nabla_{\mathbf{a}}\mathbf{x}')\theta - (\nabla_{\mathbf{a}}\mathbf{x})\theta\right) \\ &= (\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_{\mathbf{a}} r + \gamma\left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\nabla_{\mathbf{a}}\mathbf{x}'\theta - \left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\nabla_{\mathbf{a}}\mathbf{x}\theta \\ &= \left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\nabla_{\mathbf{a}} r - \left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\left(\nabla_{\mathbf{a}}\mathbf{x} - \gamma\nabla_{\mathbf{a}}\mathbf{x}'\right)\theta\end{aligned} \tag{32}$$

Here, $\nabla_{\mathbf{a}}\mathbf{x} \in \mathbb{R}^{A\times N}$, is a matrix, while $\nabla_{\mathbf{a}} r \in \mathbb{R}^A$ is a vector.

Putting the two terms together Eq. 31 & 32 and factorising the terms multiplying θ_t, we can write the expected next weight vector as:

$$\mathrm{E}\left[\theta_{t+1} \mid \theta_t\right] = \theta_t + \eta\Delta\theta = (\mathbf{I} - \eta(\mathbf{A} + \lambda_{\text{a}}\tilde{\mathbf{A}}))\theta_t + \eta\mathbf{u} \tag{33}$$

where:

$$\mathbf{u} = \mathrm{E}\left[r\mathbf{x} + \left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\nabla_a r\right] \tag{34}$$

$$\mathbf{A} = \mathrm{E}\left[\mathbf{x}(\mathbf{x} - \gamma\mathbf{x}')^T\right] \in \mathbb{R}^{N\times N} \tag{35}$$

$$\tilde{\mathbf{A}} = \mathrm{E}\left[\left(\nabla_{\mathbf{a}}\mathbf{x}\right)^T\left(\nabla_{\mathbf{a}}\mathbf{x} - \gamma\nabla_{\mathbf{a}}\mathbf{x}'\right)\right] \in \mathbb{R}^{N\times N} \tag{36}$$

Since only $\mathbf{A}$ and $\tilde{\mathbf{A}}$ multiplies θ, these two quantities exclusively are important for guaranteeing stable learning. If $\mathbf{A}$ is positive definite and $\tilde{\mathbf{A}}$ is positive-semi-definite, then $(\mathbf{A} + \lambda_{\text{a}}\tilde{\mathbf{A}})$ is positive-definite, and for sufficiently small η, the magnitude of the eigenvalues of $\mathbf{I} - \eta(\mathbf{A} + \lambda_{\text{a}}\tilde{\mathbf{A}})$ are all smaller than 1, in which case the system is stable. Crucially, the term $\mathbf{A}$ is the same as traditional TD-learning so (Sutton, 1988) provides a proof that $\mathbf{A}$ is always positive definite (see also Sutton & Barto, 2018). Thus, all we have to prove is that $\tilde{\mathbf{A}}$ is positive semi-definite. In order to prove that $\tilde{\mathbf{A}}$ is positive semi-definite, we require an (very reasonable) assumption, that the timestep Δt is small. Specifically, if we take,

$$\mathbf{s}' = \Delta t\, f(\mathbf{s},\mathbf{a}) + \mathbf{s} \tag{37}$$

Then the $\nabla_{\mathbf{a}}\mathbf{x}'$ terms must be small. Specifically, applying the chain rule, the full derivative is (expressed here for 1d case, $\nabla_{\mathbf{a}}\mathbf{x}' = \frac{\partial x'}{\partial a}$, for simplicity),

$$\frac{\partial x'}{\partial a} = \frac{\partial x'}{\partial a'}\frac{\partial a'}{\partial s'}\frac{\partial s'}{\partial a} + \frac{\partial x'}{\partial s'}\frac{\partial s'}{\partial a}\frac{\partial x'}{\partial a} = \left(\frac{\partial x'}{\partial a'}\frac{\partial a'}{\partial s'} + \frac{\partial x'}{\partial s'}\right)\frac{\partial s'}{\partial a} \tag{38}$$

This is proportional to Δt, as

$$\frac{\partial s'}{\partial a} = \Delta t \frac{\partial f(s,a)}{\partial a} \tag{39}$$

The key test for positive semi-definiteness is that for all vectors $\mathbf{b}$,

$$0 \leq \mathbf{b}^T \tilde{\mathbf{A}} \mathbf{b}. \tag{40}$$

Substituting the definition of $\tilde{\mathbf{A}}$ from Eq. (36),

$$0 \leq \mathbf{b}^T E[(\nabla_{\mathbf{a}} \mathbf{x})^T \nabla_{\mathbf{a}} \mathbf{x}] \mathbf{b} - \gamma \mathbf{b}^T E[(\nabla_{\mathbf{a}} \mathbf{x})^T \nabla_{\mathbf{a}} \mathbf{x}'] \mathbf{b}. \tag{41}$$

Putting the $\mathbf{b}$'s inside the expectation,

$$0 \leq E[(\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})^T (\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})^T (\nabla_{\mathbf{a}} \mathbf{x}'\mathbf{b})] \tag{42}$$

Now, $(\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b}) \in \mathbb{R}^{1 \times A}$ is a length-A row-vector, and the terms inside the expectations are inner products of two length A vectors. We can write these vector inner products as sums,

$$0 \leq \sum_{i=1}^{A} E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{43}$$

where a_i is a particular element of the action-vector, $\mathbf{a}$, so $\nabla_{a_i} \mathbf{x} \in \mathbb{R}^N$ is a length-N vector, and

$$\nabla_{\mathbf{a}} \mathbf{x} = \begin{pmatrix} \nabla_{a_1} \mathbf{x} \\ \nabla_{a_2} \mathbf{x} \\ \vdots \\ \nabla_{a_A} \mathbf{x} \end{pmatrix} \tag{44}$$

Critically, the overall inequality in Eq. (42) holds if a similar inequality holds for every term in the sum in Eq. (43),

$$0 \leq E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{45}$$

As $\nabla_{a_i} \mathbf{x}\mathbf{b}$ and $\nabla_{a_i} \mathbf{x}'\mathbf{b}$ are scalars, we can write,

$$0 \leq E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^2] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})(\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{46}$$

There are now two cases. If $0 = E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^2]$, we must have that $0 = (\nabla_{a_i} \mathbf{x}\mathbf{b})$ always (except at a set of measure zero). Thus, the second term must also be zero (except at a set of measure zero), i.e. $0 = E[(\nabla_{a_i} \mathbf{x}\mathbf{b})(\nabla_{a_i} \mathbf{x}'\mathbf{b})]$ and the inequality holds. Alternatively, if $E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^2]$ is non-zero, it must be positive, in that case, the second term can also be non-zero, but it scales with Δt, so we can always choose a Δt, small enough to ensure that the Eq. (46) holds, in which case $\tilde{\mathbf{A}}$ is positive semi-definite.

B.1. First-order Taylor approximation

B.1.1. State expansion proof

This proof is analogous to the action expansion proof (i.e. section 4.1) and, is included for completeness. The expectation is taken over $\boldsymbol{\xi}_s$,

$$\mathrm{E}\left[\Delta\theta_t\right] = \eta\, \mathrm{E}\left[\delta_\theta(\mathbf{s}_t + \boldsymbol{\xi}_\mathrm{s}, \mathbf{a}_t) \nabla_\theta Q(\mathbf{s}_t + \boldsymbol{\xi}_\mathrm{s}, \mathbf{a}_t)\right] \tag{47}$$

Applying the first-order Taylor expansion

$$\mathrm{E}\left[\Delta_{\mathrm{Ta}}\theta_t\right] = \eta\, \mathrm{E}\left[\left(\delta_\theta(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{s}^T \nabla_{\mathbf{s}} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\right) \nabla_\theta \left(Q(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\xi}_\mathrm{s}^T \nabla_{\mathbf{s}} Q(\mathbf{s}_t, \mathbf{a}_t)\right)\right] \tag{48}$$

As ∇_θ is a linear operator, and $\boldsymbol{\xi}_\text{s}$ does not depend on θ,

$$= \eta \, \text{E}\left[\left(\delta_\theta(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)\left(\nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\xi}_\text{s}^T \nabla^2_{\mathbf{s},\theta} Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)\right] \tag{49}$$

where $\nabla^2_{\mathbf{s},\theta} Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ is a matrix of second derivatives. The expectation of $\boldsymbol{\xi}_\text{s}$ is zero, so the terms linear in $\boldsymbol{\xi}_\text{s}$ are zero,

$$= \eta \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \eta \, \text{E}\left[\left(\boldsymbol{\xi}_\text{s}^T \nabla^2_{\mathbf{s},\theta} Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)\left(\boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)\right] \tag{50}$$

transposing the first term in the expectation (which we can do as it is overall a scalar)

$$= \eta \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \eta \, \text{E}\left[\left(\nabla^2_{\theta,\mathbf{s}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \boldsymbol{\xi}_\text{s}\right)\left(\boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)\right] \tag{51}$$

we can then move the terms independent of $\boldsymbol{\xi}_\text{a}$ out of the expectation:

$$= \eta \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \eta \nabla^2_{\theta,\mathbf{s}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \, \text{E}\left[\boldsymbol{\xi}_\text{s} \boldsymbol{\xi}_\text{s}^T\right] \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \tag{52}$$

we defined $\text{E}\left[\boldsymbol{\xi}_\text{s} \boldsymbol{\xi}_\text{s}^T\right] = \boldsymbol{\Sigma}_\text{s}$,

$$= \eta \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \eta \nabla^2_{\theta,\mathbf{s}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \boldsymbol{\Sigma}_\text{s} \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \tag{53}$$

Finally, if we assume the state covariance is isotropic, $\boldsymbol{\Sigma}_\text{s} = \lambda_\text{s} \mathbf{I}$, we get the following (1st-order) Taylor TD-update for the state expansion

$$= \eta \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\theta Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \ \eta \lambda_s \nabla^2_{\theta,\mathbf{s}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\mathbf{s} \delta_\theta(\mathbf{s}_t, \mathbf{a}_t) \tag{54}$$

C. Using the chain rule to expand the gradient of Taylor TD

A (learned) differentiable model of the transitions and rewards is needed to compute the gradient terms $\nabla_\mathbf{a} \delta_\theta(\mathbf{s}, \mathbf{a})$ and $\nabla_\mathbf{s} \delta_\theta(\mathbf{s}, \mathbf{a})$ in Taylor TD. This is because the TD target, $\delta_\theta = r(\mathbf{s}, \mathbf{a}) + \gamma Q_\theta(\mathbf{s}', \mathbf{a}')$, comprises the reward and the Q-value at the next time step, both of which depend on $\mathbf{s}$ and $\mathbf{a}$. The full gradients for the action expansion can be written:

$$\nabla_\mathbf{a} \delta_\theta(\mathbf{s}, \mathbf{a}) = \frac{\partial \hat{r}(\mathbf{s}, \mathbf{a})}{\partial \mathbf{a}} - \frac{\partial Q_\theta(\mathbf{s}, \mathbf{a}_t)}{\partial \mathbf{a}} + \gamma \frac{\partial \hat{\mathbf{s}}'}{\partial \mathbf{a}} \left(\frac{\partial Q_\theta(\hat{\mathbf{s}}', \mathbf{a}')}{\partial \hat{\mathbf{s}}'} + \frac{\partial \mathbf{a}'}{\partial \hat{\mathbf{s}}'} \frac{\partial Q_\theta(\hat{\mathbf{s}}', \mathbf{a}')}{\partial \mathbf{a}'} \right) \tag{55}$$

where $\hat{r}$ denotes a (differentiable) reward function, $\hat{s}$ denotes the predicted next state, while $\frac{\partial \hat{\mathbf{s}}'}{\partial \mathbf{a}}$ denotes the gradient term computed by differentiating through a differentiable model of the transitions. An analogous gradient can be written for $\nabla_\mathbf{s} \delta_\theta(\mathbf{s}, \mathbf{a})$ (i.e., state expansion). However, we do not have to explicitly implement these expressions, as we use autodiff in PyTorch to find gradients of $\delta_\theta(\mathbf{s}, \mathbf{a})$ wrt $\mathbf{a}$ and $\mathbf{s}$ directly, by re-writing the Taylor TD updates in terms of a simple loss (see Appendix D).

D. Taylor TD-update as a loss

Here we report how to easily implement the Taylor TD-update (i.e. Eq. 21) as a loss to be passed to an optimizer (e.g. PyTorch optimizer).

$$\begin{aligned} \mathcal{L}_\theta = & \\ & - \ \text{stopgrad}_\theta\{\delta_\theta(s_t, a_t)\} \, Q_\theta(s_t, a_t) \\ & - \lambda_\text{a} \frac{\text{stopgrad}_\theta\{\nabla_a \delta_\theta(s_t, a_t)\} \cdot \nabla_a Q_\theta(s_t, a_t)}{\text{stopgrad}_\theta\{\| \left(\nabla_a \delta_\theta(s_t, a_t)\right) \| \|\nabla_a Q_\theta(s_t, a_t)\|\}} \\ & - \lambda_\text{s} \frac{\text{stopgrad}_\theta\{(\nabla_s \delta_\theta(s_t, a_t))\} \cdot \nabla_s Q_\theta(s_t, a_t)}{\text{stopgrad}_\theta\{\| \left(\nabla_s \delta_\theta(s_t, a_t)\right) \| \|\nabla_s Q_\theta(s_t, a_t)\|\}} \end{aligned} \tag{56}$$

"stopgrad$_\theta\{\}$" denotes the optimizer should not differentiate the quantity inside the curly brackets relative to the parameters θ (i.e. equivalent to ".detach()" in PyTorch).

E. Variance reduction

Here we describe in more details how we computed the variance of Taylor TD and standard TD-learning updates on the continuous control task Half-Cheetah-v2 (i.e. Section 5.1). Based on a policy, a value function and a set of randomly sampled states from a memory buffer, we computed the Taylor and the standard (MC) TD batch updates across all sampled states (under the policy). The update for each state takes the form of a gradient evaluation of the value function relative to the TD objective. Next, we computed the variance of these gradient terms across all states and summed them up since the variance of each parameter update contributes to the variance of the value function relative to the TD objective. We repeated this process for different seeds and plotted the mean update variance for both Taylor and the standard TD updates (i.e. Fig.1).

F. Toy example

We provide a toy example to investigate the settings in which using a Taylor expansion of an expected update may provide the largest benefits relative to MC estimates of the expected update. To do so, we train a function approximation parameterised by θ (e.g. a critic) to approximate the expected outputs of an underlying target function (e.g. TD targets). Starting from a single expected target value, we can formulate the following objective (for multiple targets, we can just sum the objectives):

$$J(\theta) = \frac{1}{2}\,\mathrm{E}_{\boldsymbol{\xi}_\mathrm{x}}\left[(y(x+\boldsymbol{\xi}_\mathrm{x}) - \hat{y}_\theta(x+\boldsymbol{\xi}_\mathrm{x}))^2\right] \tag{57}$$

where $\boldsymbol{\xi}_\mathrm{x}$ is sampled form a Gaussian distribution with $\mathrm{E}\left[\boldsymbol{\xi}_\mathrm{x}\right] = \mathbf{0}$, $\mathrm{E}\left[\boldsymbol{\xi}_\mathrm{x}\boldsymbol{\xi}_\mathrm{x}^T\right] = \lambda_\mathrm{x}\mathbf{I}$. Hence, the task requires the function approximation $\hat{y}_\theta()$ to approximate the expected outputs of the target function $y()$, based on a set of randomly perturbed inputs (i.e. $x + \boldsymbol{\xi}_\mathrm{x}$). We do this by comparing two different approaches. The first approach involves sampling different outputs of y (based on different input perturbations $\boldsymbol{\xi}_\mathrm{x}$) and training $\hat{y}_\theta()$ to match those outputs. We denote this approach as "MC targets". This approach aims to mimic standard TD-learning, where TD-targets are sampled at each time step to train the critic (i.e. MC estimates). The second approach aims to mimic Taylor TD, applying a first-order Taylor expansion to the objective in Eq 57,

$$J_\mathrm{Ta}(\theta) = \frac{1}{2}\,\mathrm{E}_{\boldsymbol{\xi}_\mathrm{x}}\left[(y + \boldsymbol{\xi}_\mathrm{x}^T\nabla_\mathrm{x} y\ - \hat{y}_\theta - \boldsymbol{\xi}_\mathrm{x}^T\nabla_\mathrm{x}\hat{y}_\theta)^2\right] \tag{58}$$

For clarity we used the notation $\hat{y}_\theta = \hat{y}_\theta(x)$ and $y = y(x)$. The terms that are linear in $\boldsymbol{\xi}_\mathrm{x}$ cancels and after summing up equal terms, we get:

$$J_\mathrm{Ta}(\theta) = \frac{1}{2}(y - \hat{y}_\theta)^2 + \tfrac{1}{2}\lambda_\mathrm{x}(\nabla_x y - \nabla_x\hat{y}_\theta)^T(\nabla_x y - \nabla_x\hat{y}_\theta) \tag{59}$$

We can then take the gradient of this approximation relative to the function approximation weights to get the weight update for the Taylor approach:

$$\begin{aligned}\nabla_\theta J(\theta) \sim\ & \hat{y}_\theta\nabla_\theta\hat{y}_\theta \\ & + \lambda_\mathrm{x}\nabla_\theta(\nabla_x\hat{y}_\theta^T\nabla_x\hat{y}_\theta) \\ & - y\ \nabla_\theta\hat{y}_\theta \\ & - \lambda_\mathrm{x}\nabla_\theta(\nabla_x y^T\nabla_x\hat{y}_\theta^T)\end{aligned} \tag{60}$$

Note, this update is reminiscent of of double backpropagation settings in the supervised learning literature (Czarnecki et al., 2017; Drucker & Le Cun, 1992). The toy example allows us to compare the Taylor expansion with the MC estimation under different conditions (i.e. dimension size of the data points as well as number of data points). In particular, we compare the two approaches for inputs of different dimensions (from 1 to 100 dimensional inputs) and across two data regimes, a low data regime (i.e. only 15 training samples are used to train the function approximation) and a high data regime (i.e. over 100 samples are used to train the function approximation). Performance are then assessed based on a novel set of inputs (i.e. 50), sampled from the same underlying distribution as the training data. Fig. 4 show that the benefits of the Taylor approach over MC estimates increase as the dimension of the data points grows in size (e.g. RL tasks involving high dimensional action and state spaces). In particular, these benefits are even larger in the presence of a low data regime, such as RL settings in which for any given state-action pair we can only sample a few transitions from the environment.

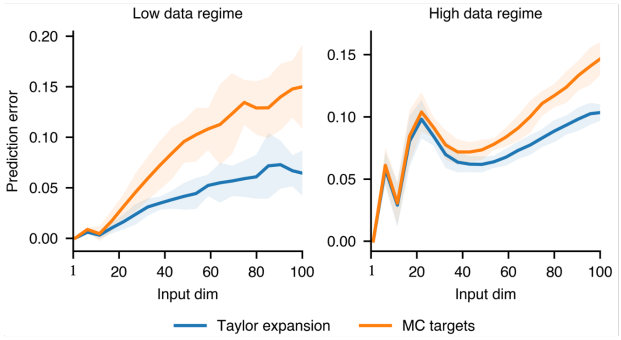

Figure 4. Average performance of the Taylor expansion approach relative to MC estimates on unseen input examples across several input dimensions (x-axis) and two data regimes (5 runs, 95% c.i.).

G. Ablations

G.1. State expansion ablation

Here, we ask whether the Taylor state expansion brings any benefit to performance, on top of the Taylor action expansion. To do so, we compare the TaTD3 algorithms with and without state expansion on two standard benchmark tasks (i.e. analogous to setting $\lambda_{\text{s}} = 0$ in the update Eq. 21). Fig. (5) shows that including the state expansion is beneficial to both environments.

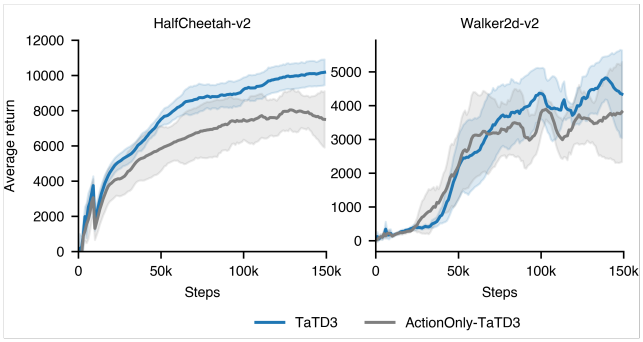

Figure 5. Performance in terms of average returns for TaTD3 (with both state and action expansions) compared to a version of TaTD3 that uses the action expansion only, on two benchmark continuous control tasks. Including the state expansion in TaTD3 seem to improve performance on both tasks (5 runs, 95% c.i.).

G.2. Cosine similarity ablation

Here, we ask whether taking the cosine similarity of state and action gradient terms benefit the performance of TaTD3. To do so, we compare the standard TaTD3 algorithms (trained with the loss in Eq. 21) with a version of TaTD3 trained with a loss without cosine similarity,

$$\mathcal{L}_\theta = \eta\delta(\mathbf{s}_t, \mathbf{a}_t)Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \eta\lambda_{\text{a}}\, \nabla_{\mathbf{a}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \cdot \nabla_{\mathbf{a}}\delta(\mathbf{s}_t, \mathbf{a}_t) + \eta\lambda_{\text{s}}\, \nabla_{\mathbf{s}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \cdot \nabla_{\mathbf{s}}\delta(\mathbf{s}_t, \mathbf{a}_t) \tag{61}$$

Namely, this loss optimizes the dot-product for state and action gradient terms instead of the cosine similarity. We assess this comparison on two standard benchmark tasks (i.e. see Fig. 6). In Fig. 6, we can see the cosine similarity does improve performance on both tasks.

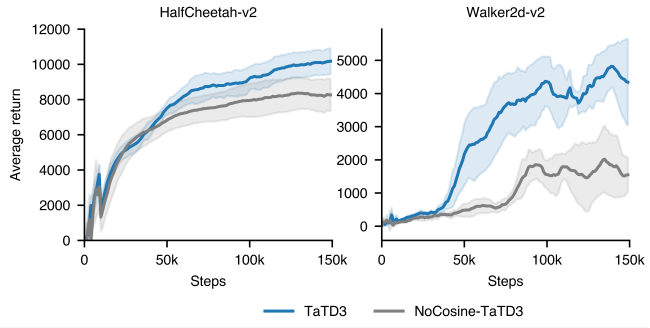

Figure 6. Performance in terms of average returns for TaTD3 compared to a version of TaTD3 that computes the dot product of state and action gradient terms, instead of cosine similarity (5 runs, 95% c.i.).

H. Computing

All experiments were run on a cluster of GPUs, including NVIDIA GeForce RTX 2080, 3090 and NDVIDIA A100. Here, we report the difference in computing time between standard TD-learning and Taylor TD based on the same GPU for each comparison. We do so for a low dimensional (Pendulum), a 'medium' dimensional (Walker2d) and a high dimensional (Humanoid) environment to span a broad range of settings.

Environment	TD-learning time	Taylor TD time	n. time steps
Pendulum	24s	38s	200
Walker2d	50s	68s	1000
Humanoid	94s	117s	1000

I. Hyperparameters settings

Below, we reported the hyperparameter settings for TaTD3 (and MC Expected-TD3),

	Pendulum-v1	HalfCheetah-v2	Walker2d-v2
Steps	10000	150000	150000
Model ensemble size	8	8	8
Model architecture (MLP)	4 h-layers of size 512	4 h-layers of size 512	4 h-layers of size 512
Reward model architecture (MLP)	3 h-layers of size 256	3 h-layers of size 256	3 h-layers of size 256
Actor-critic architecture (MLP)	2 h-layers of size 400	2 h-layers of size 400	2 h-layers of size 400
Dyna steps per environment step	10	10	10
Model horizon	1	1	1
λ_a	0.25	0.25	0.25
λ_s	1e-5	1e-5	1e-5

	Hopper-v2	Ant-v2	Humanoid-v2
Steps	10000	150000	150000
Model ensemble size	8	8	8
Model architecture (MLP)	4 h-layers of size 512	4 h-layers of size 512	4 h-layers of size 512
Reward model architecture (MLP)	3 h-layers of size 256	3 h-layers of size 512	3 h-layers of size 512
Actor-critic architecture (MLP)	2 h-layers of size 400	4 h-layers of size 400	4 h-layers of size 400
Dyna steps per environment step	10	10	10
Model horizon	1	1	1
λ_a	0.05	0.05	0.25
λ_s	1e-5	1e-5	1e-5

Note, "h-layers" stands for hidden layers and the size is in terms of number of units.

We found we could achieve good performance for TaTD3 across all tested environments without needing to fine tune the value of λ_a and λ_s to each environment (i.e., $\lambda_a = 0.05$ and $\lambda_s = 0.00005$). These parameters were founded by running a grid search over potential values of these parameters based on a single environment (i.e., Pendulum), then using the best values for all other environments. Nevertheless, we reached top performance in HalfCheetah, Walker2d and Humanoid by using a larger λ_a (i.e., $\lambda_a = 0.25$). This finding suggests these 3 environments benefit from learning a broader distribution of Q-values over the actions, we believe for better exploration.