A First-order Taylor expansion

A.1 Action expansion update proof

Transposing the scalar inner product, $\boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})$, in the first-order Taylor expansion in the main text (Eq. 10) gives,

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \text{E}_{\boldsymbol{\xi}_\text{a}}\left[\left(\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right) \nabla_\theta \left(Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \left(\nabla_\mathbf{a} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right)^T \boldsymbol{\xi}_\text{a}\right) \middle| \text{s}, \boldsymbol{\mu}_\text{a}\right] \tag{25}$$

As ∇_θ is a linear operator, and the action noise, $\boldsymbol{\xi}_\text{a}$, does not depend on the policy parameters, θ,

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \text{E}\left[\left(\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right) \left(\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\boldsymbol{\xi}_\text{a}\right) \middle| \text{s}, \boldsymbol{\mu}_\text{a}\right] \tag{26}$$

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

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \text{E}_{\boldsymbol{\xi}_\text{a}}\left[\left(\boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right)\left(\nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\boldsymbol{\xi}_\text{a}^T\right) | \text{s}, \boldsymbol{\mu}_\text{a}\right] \tag{27}$$

Swapping the order of the terms in the expectation (which is valid, as $\left(\boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right)$ is a scalar),

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \text{E}_{\boldsymbol{\xi}_\text{a}}\left[\left(\nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\boldsymbol{\xi}_\text{a}\right)\left(\boldsymbol{\xi}_\text{a}^T \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\right) | \text{s}, \boldsymbol{\mu}_\text{a}\right] \tag{28}$$

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

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\, \text{E}_{\boldsymbol{\xi}_\text{a}}\left[\boldsymbol{\xi}_\text{a}\boldsymbol{\xi}_\text{a}^T\right] \nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) \tag{29}$$

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

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \boldsymbol{\mu}_\text{a})\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\boldsymbol{\Sigma}_\text{a}\nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) \tag{30}$$

If we assume the action covariance is isotropic, $\boldsymbol{\Sigma}_\text{a} = \lambda_\text{a}\mathbf{I}$, we get the (1st-order) Taylor TD-update for the action expansion,

$$\Delta_\text{Ta}\theta(\text{s}, \boldsymbol{\mu}_\text{a}) = \delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\nabla_\theta Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) + \lambda_\text{a}\nabla^2_{\theta,\mathbf{a}} Q_\theta(\text{s}, \boldsymbol{\mu}_\text{a})\nabla_\mathbf{a}\delta_\theta(\text{s}, \boldsymbol{\mu}_\text{a}) \tag{31}$$

A.2 State expansion update proof

Applying the first-order Taylor expansion to Eq. (16), then following the approach from the previous section,

$$\begin{aligned}
\Delta_\text{Ta}\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) &= \text{E}_{\boldsymbol{\xi}_\text{s}}\left[\left(\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\right) \nabla_\theta \left(Q(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \left(\nabla_\text{s} Q(\boldsymbol{\mu}_\text{s}, \mathbf{a})\right)^T \boldsymbol{\xi}_\text{s}\right) | \boldsymbol{\mu}_\text{s}, \mathbf{a}\right] && (32)\\
&= \text{E}_{\boldsymbol{\xi}_\text{s}}\left[\left(\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\right) \left(\nabla_\theta Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \nabla^2_{\theta,\mathbf{s}} Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\boldsymbol{\xi}_\text{s}\right) | \boldsymbol{\mu}_\text{s}, \mathbf{a}\right] && (33)\\
&= \delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\nabla_\theta Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \text{E}_{\boldsymbol{\xi}_\text{s}}\left[\left(\nabla^2_{\theta,\mathbf{s}} Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\boldsymbol{\xi}_\text{s}\right)\left(\boldsymbol{\xi}_\text{s}^T \nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\right) | \boldsymbol{\mu}_\text{s}, \mathbf{a}\right] && (34)\\
&= \delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\nabla_\theta Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \nabla^2_{\theta,\mathbf{s}} Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\, \text{E}_{\boldsymbol{\xi}_\text{s}}\left[\boldsymbol{\xi}_\text{s}\boldsymbol{\xi}_\text{s}^T\right] \nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) && (35)
\end{aligned}$$

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

$$\Delta_\text{Ta}\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) = \delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\nabla_\theta Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \nabla^2_{\theta,\mathbf{s}} Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\boldsymbol{\Sigma}_\text{s}\nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) \tag{36}$$

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

$$\Delta_\text{Ta}\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) = \delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\nabla_\theta Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) + \lambda_\text{s}\nabla^2_{\theta,\mathbf{s}} Q_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a})\nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \mathbf{a}) \tag{37}$$

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):

$$\Delta_{\text{Ta}}\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}}) = \delta_\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}})\nabla_\theta Q_\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}}) + \lambda_{\text{a}}\nabla^2_{\theta,\boldsymbol{\mu}_{\text{a}}} Q_\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}})\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}}) \tag{38}$$

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,

$$Q_\theta(\mathbf{s}, \boldsymbol{\mu}_{\text{a}}) = \theta^T\mathbf{x} \qquad\qquad Q_\theta(\mathbf{s}', \mathbf{a}') = \theta^T\mathbf{x}' \tag{39}$$

where,

$$\mathbf{x} = \phi(\mathbf{s}, \boldsymbol{\mu}_{\text{a}}) \in \mathbb{R}^N \qquad\qquad \mathbf{x}' = \phi(\mathbf{s}', \mathbf{a}') \in \mathbb{R}^N \tag{40}$$

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,

$$\begin{aligned}\delta_\theta(\mathbf{s}, \mathbf{a})\nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a}) &= (r + \gamma\theta^T\mathbf{x}' - \theta^T\mathbf{x})\mathbf{x} \\ &= r\mathbf{x} - \mathbf{x}(\mathbf{x} - \gamma\mathbf{x}')^T\theta.\end{aligned} \tag{41}$$

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

$$\begin{aligned}\nabla^2_{\theta,\mathbf{a}} Q_\theta(\mathbf{s}, \mathbf{a})\nabla_{\mathbf{a}}\delta_\theta(\mathbf{s}, \mathbf{a}) &= \nabla_{\theta,\mathbf{a}}\left(\mathbf{x}^T\theta\right)\left(\nabla_{\mathbf{a}} r + \gamma\nabla_{\mathbf{a}}(\mathbf{x}'^T\theta) - \nabla_{\mathbf{a}}(\mathbf{x}^T\theta)\right) \\ &= (\nabla_{\mathbf{a}}\mathbf{x})^T\left(\nabla_{\mathbf{a}} r + \gamma(\nabla_{\mathbf{a}}\mathbf{x}')\theta - (\nabla_{\mathbf{a}}\mathbf{x})\theta\right) \\ &= (\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_{\mathbf{a}} r + \gamma(\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_{\mathbf{a}}\mathbf{x}'\theta - (\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_{\mathbf{a}}\mathbf{x}\theta \\ &= (\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_{\mathbf{a}} r - (\nabla_{\mathbf{a}}\mathbf{x})^T\left(\nabla_{\mathbf{a}}\mathbf{x} - \gamma\nabla_{\mathbf{a}}\mathbf{x}'\right)\theta\end{aligned} \tag{42}$$

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. 41 & 42 and factorising the terms multiplying θ_t, we can write the expected next weight vector as:

$$\mathrm{E}\left[\theta_{t+1} \mid \theta_t\right] = \theta_t + \eta\Delta\theta = (\mathbf{I} - \eta(\mathbf{A} + \lambda_{\text{a}}\tilde{\mathbf{A}}))\theta_t + \eta\mathbf{u} \tag{43}$$

where:

$$\mathbf{u} = \mathrm{E}\left[r\mathbf{x} + (\nabla_{\mathbf{a}}\mathbf{x})^T\nabla_a r\right] \tag{44}$$

$$\mathbf{A} = \mathrm{E}\left[\mathbf{x}(\mathbf{x} - \gamma\mathbf{x}')^T\right] \in \mathbb{R}^{N\times N} \tag{45}$$

$$\tilde{\mathbf{A}} = \mathrm{E}\left[(\nabla_{\mathbf{a}}\mathbf{x})^T\left(\nabla_{\mathbf{a}}\mathbf{x} - \gamma\nabla_{\mathbf{a}}\mathbf{x}'\right)\right] \in \mathbb{R}^{N\times N} \tag{46}$$

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 [11] provides a proof that $\mathbf{A}$ is always positive definite [see also 12]. 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,

$$\mathbf{s}' = \Delta t\, f(\mathbf{s}, \mathbf{a}) + \mathbf{s} \tag{47}$$

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),

$$\frac{\partial x'}{\partial a} = \frac{\partial x'}{\partial a'}\frac{\partial a'}{\partial s'}\frac{\partial s'}{\partial a} + \frac{\partial x'}{\partial s'}\frac{\partial s'}{\partial a}\frac{\partial x'}{\partial a} = \left(\frac{\partial x'}{\partial a'}\frac{\partial a'}{\partial s'} + \frac{\partial x'}{\partial s'}\right)\frac{\partial s'}{\partial a} \tag{48}$$

This is proportional to Δt, as

$$\frac{\partial s'}{\partial a} = \Delta t \frac{\partial f(s,a)}{\partial a} \tag{49}$$

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

$$0 \leq \mathbf{b}^T \tilde{\mathbf{A}} \mathbf{b}. \tag{50}$$

Substituting the definition of $\tilde{\mathbf{A}}$ from Eq. (46),

$$0 \leq \mathbf{b}^T E[(\nabla_{\mathbf{a}} \mathbf{x})^T \nabla_{\mathbf{a}} \mathbf{x}] \mathbf{b} - \gamma \mathbf{b}^T E[(\nabla_{\mathbf{a}} \mathbf{x})^T \nabla_{\mathbf{a}} \mathbf{x}'] \mathbf{b}. \tag{51}$$

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

$$0 \leq E[(\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})^T (\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{\mathbf{a}} \mathbf{x}\mathbf{b})^T (\nabla_{\mathbf{a}} \mathbf{x}'\mathbf{b})] \tag{52}$$

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,

$$0 \leq \sum_{i=1}^{A} E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{53}$$

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

$$\nabla_{\mathbf{a}} \mathbf{x} = \begin{pmatrix} \nabla_{a_1} \mathbf{x} \\ \nabla_{a_2} \mathbf{x} \\ \vdots \\ \nabla_{a_A} \mathbf{x} \end{pmatrix} \tag{54}$$

Critically, the overall inequality in Eq. (52) holds if a similar inequality holds for every term in the sum in Eq. (53),

$$0 \leq E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}\mathbf{b})] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^T (\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{55}$$

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

$$0 \leq E[(\nabla_{a_i} \mathbf{x}\mathbf{b})^2] - \gamma E[(\nabla_{a_i} \mathbf{x}\mathbf{b})(\nabla_{a_i} \mathbf{x}'\mathbf{b})] \tag{56}$$

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. (56) holds, in which case $\tilde{\mathbf{A}}$ is positive semi-definite.

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:

$$\nabla_{\mathbf{a}} \delta_\theta(\mathbf{s}, \mathbf{a}) \quad = \quad \frac{\partial \hat{r}(\mathbf{s}, \mathbf{a})}{\partial \mathbf{a}} \; - \; \frac{\partial Q_\theta(\mathbf{s}, \mathbf{a}_t)}{\partial \mathbf{a}} \; + \; \gamma \frac{\partial \hat{\mathbf{s}}'}{\partial \mathbf{a}} \left(\frac{\partial Q_\theta(\hat{\mathbf{s}}', \mathbf{a}')}{\partial \hat{\mathbf{s}}'} + \frac{\partial \mathbf{a}'}{\partial \hat{\mathbf{s}}'} \frac{\partial Q_\theta(\hat{\mathbf{s}}', \mathbf{a}')}{\partial \mathbf{a}'} \right) \tag{57}$$

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 Computing the updates as the gradient of a loss

Here we report how to easily implement the Taylor TD-update (i.e. Eq. 18) as a loss to be passed to an optimizer (e.g. PyTorch optimizer).

$$\begin{aligned}\mathcal{L}_\theta^{\text{critic}} = &- \text{ stopgrad}_\theta\{\delta_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})\}\; Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) \\ &- \lambda_\text{a}\frac{\text{stopgrad}_\theta\{\nabla_\mathbf{a}\delta_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})\} \cdot \nabla_\mathbf{a} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})}{\text{stopgrad}_\theta\{\|\,(\nabla_\mathbf{a}\delta_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}))\,\|\|\nabla_\mathbf{a} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})\|\}} \\ &- \lambda_\text{s}\frac{\text{stopgrad}_\theta\{(\nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}))\} \cdot \nabla_\mathbf{s} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})}{\text{stopgrad}_\theta\{\|\,(\nabla_\mathbf{s}\delta_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}))\,\|\|\nabla_\mathbf{s} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a})\|\}}\end{aligned}$$

"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 Model learning

Taylor TD-learning requires a differentiable model of the transitions. We learned this model according to maximum-likelihood based on observed transitions sampled from a reply buffer,

$$\mathcal{L}_w^{\text{model}} = \hat{p}_w(s' \mid s, a); \qquad\qquad (s, a, s') \sim \mathcal{B} \tag{58}$$

where $\mathcal{B}$ denotes a reply buffer from which (observed) transitions can be sampled. In TaTD3, the model of the transitions consists of an ensemble of 8 models. These models return a Gaussian distribution over the next state, with zero correlations, and with mean and variance given by applying a neural network to the previous state-action pair. Conversely, the model of the rewards consists of a neural network trained with mean-square error on the observed rewards sampled from the reply buffer.

F Variance reduction

Here we describe in more details how we computed the variance of Taylor TD and standard TD-learning updates on the 6 continuous control benchmark tasks (i.e. Section 4.1). 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 (sample-based) TD batch updates across all sampled states (under the policy). Note, we ensured the exact same (random) states were used to compute the Taylor and the standard (sample-based) TD batch updates. 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 10 different seeds and plotted the mean update variance and standard error for both Taylor and the standard TD updates (i.e. Fig.1).

G Variance reduction across training

In this section we test how the variance of Taylor TD and standard TD-learning updates changes across training. In particular, we want to assess how the variance advantage of Taylor TD updates (over standard TD updates) changes across 3 different points in training: at the start of training (i.e., for an untrained critic and an untrained model of the transitions), early in training (i.e., after 5000 training steps) and finally late in training (i.e., after 50000 training steps).

In Fig. 4, we can see that across all training points, Taylor TD updates provide lower variance updates compared to standard TD updates.

H 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 sample-based estimates of the same expected

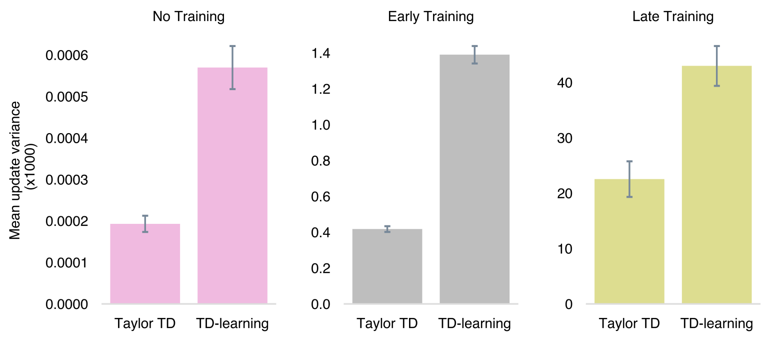

Figure 4: Mean update variance (and standard error) between Taylor TD and standard (sample-based) TD-learning (batch) updates, computed across 3 time points in training: No Training (i.e., for an untrained critic), Early training (i.e., after 5000 training steps) and finally late in training (i.e., after 50000 training steps) . All results are based on 10 runs with corresponding error bars on the Half-Cheetah-v2 environment.

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):

$$J(\theta) = \frac{1}{2}\,\mathrm{E}_{\boldsymbol{\xi}_\mathrm{x}}\left[(y(x+\boldsymbol{\xi}_\mathrm{x}) - \hat{y}_\theta(x+\boldsymbol{\xi}_\mathrm{x}))^2\right] \tag{59}$$

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 "Sample-based targets". This approach aims to mimic standard TD-learning, where TD-targets are sampled at each time step to train the critic (i.e. providing sample-based estimates). The second approach aims to mimic Taylor TD, applying a first-order Taylor expansion to the objective in Eq 59,

$$J_{\mathrm{Ta}_{\boldsymbol{\xi}_\mathrm{a}}}(\theta) = \frac{1}{2}\,\mathrm{E}_{\boldsymbol{\xi}_\mathrm{x}}\left[(y + \boldsymbol{\xi}_\mathrm{x}^T\nabla_\mathrm{x} y\; - \hat{y}_\theta - \boldsymbol{\xi}_\mathrm{x}^T\nabla_\mathrm{x}\hat{y}_\theta)^2\right] \tag{60}$$

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:

$$J_{\mathrm{Ta}_{\boldsymbol{\xi}_\mathrm{a}}}(\theta) = \frac{1}{2}(y - \hat{y}_\theta)^2 + \tfrac{1}{2}\lambda_\mathrm{x}(\nabla_x y - \nabla_x\hat{y}_\theta)^T(\nabla_x y - \nabla_x\hat{y}_\theta) \tag{61}$$

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

$$\begin{aligned}\nabla_\theta J(\theta) \sim\; & \hat{y}_\theta\nabla_\theta\hat{y}_\theta \\ & + \lambda_\mathrm{x}\nabla_\theta(\nabla_x\hat{y}_\theta^T\nabla_x\hat{y}_\theta) \\ & - y\;\nabla_\theta\hat{y}_\theta \\ & - \lambda_\mathrm{x}\nabla_\theta(\nabla_x y^T\nabla_x\hat{y}_\theta^T)\end{aligned} \tag{62}$$

Note, this update is reminiscent of of double backpropagation settings in the supervised learning literature [49, 50]. The toy example allows us to compare the Taylor expansion with the sample-based 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

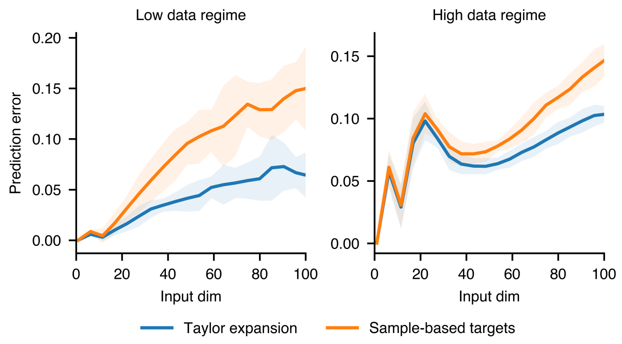

Figure 5: Average performance of the Taylor expansion approach relative to sample-based estimates on unseen input examples across several input dimensions (x-axis) and two data regimes (5 runs, 95% c.i.).

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. 5 show that the benefits of the Taylor approach over sample-based 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.

I Code

All the code is available at `https://github.com/maximerobeyns/taylortd`

J Baseline algorithms

Plotted performance of MBPO [18] was directly taken from the official algorithm repository on GitHub at `https://github.com/jannerm/mbpo`. Plotted performance of SAC [2] was also obtained taken from the official algorithm repository on GitHub at `https://github.com/haarnoja/sac`. Plotted performances of both MAGE and Dyna-TD3 [19] were obtained by re-running these algorithms on the benchmark environments, taking the implementations from the official algorithms' repository available at `https://github.com/nnaisense/MAGE`.

K Ablations

K.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. 18). Fig. (6) shows that including the state expansion is beneficial to both environments.

K.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. 18) with a version of TaTD3 trained with a loss without cosine similarity,

$$\mathcal{L}_\theta = \delta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) + \lambda_\text{a} \, \nabla_\mathbf{a} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) \cdot \, \nabla_\mathbf{a} \delta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) + \lambda_\text{s} \, \nabla_\mathbf{s} Q_\theta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) \cdot \nabla_\mathbf{s} \delta(\boldsymbol{\mu}_\text{s}, \boldsymbol{\mu}_\text{a}) \tag{63}$$

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. 7). In Fig. 7, we can see the cosine similarity does improve performance on both tasks.

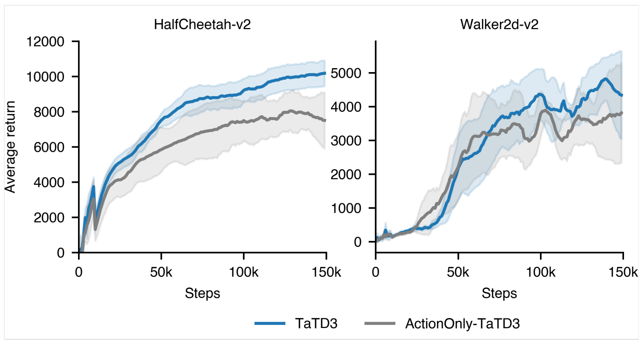

Figure 6: Performance in terms of average returns for TaTD3 (with both state and action expansions) compared to a version of TaTD3 that uses the action expansion only, on two benchmark continuous control tasks. Including the state expansion in TaTD3 seem to improve performance on both tasks (5 runs, 95% c.i.).

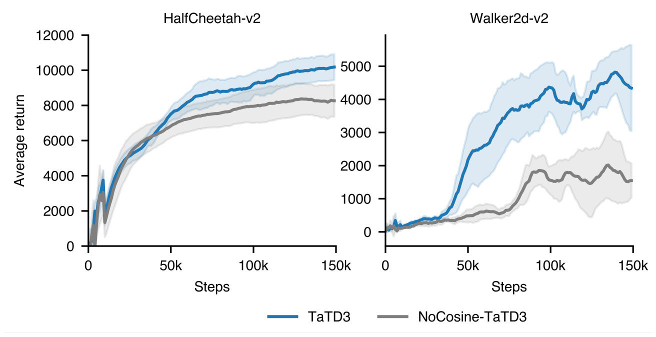

Figure 7: Performance in terms of average returns for TaTD3 compared to a version of TaTD3 that computes the dot product of state and action gradient terms, instead of cosine similarity (5 runs, 95% c.i.).

L 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, Taylor TD and the two strongest baselines (i.e., MAGE and MBPO) 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. For MBPO, we report the average running time across all training epochs using the horizon lengthening schedule outlined in the original paper [18].

Environment	TD-learning time	Taylor TD time	MAGE time	MBPO time	n. time steps
Pendulum	24s	38s	36s	52s	200
Walker2d	50s	68s	63s	133s	1000
Humanoid	94s	117s	127s	235s	1000

MAGE has about the same runtime as our method (TaTD3), while MBPO is a lot slower than our method.

M Hyperparameters settings

Below, we reported the hyperparameter settings for TaTD3 (and sample-based 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	0.06	0.06	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., $\lambda_a = 0.06$ 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. Additionally, you may notice the much larger values for λ_a compared to λ_s. However, we should stress that any direct comparison between the state covariance λ_s and the action covariance λ_a may not be very meaningful. This is because the distribution of initial states and the distribution of initial actions may be very different, making any direct comparison between λ_s and λ_a hard to assess.