VA-learning as a more efficient alternative to Q-learning

Yunhao Tang [1] **Rémi Munos** [1] **Mark Rowland** [1] **Michal Valko** [1]

Abstract

In reinforcement learning, the advantage function is critical for policy improvement, but is often extracted from a learned Q-function. A natural question is: *Why not learn the advantage function directly?* In this work, we introduce VA-learning, which directly learns advantage function and value function using bootstrapping, *without* explicit reference to Q-functions. VA-learning learns off-policy and enjoys similar theoretical guarantees as Q-learning. Thanks to the direct learning of advantage function and value function, VA-learning improves the sample efficiency over Q-learning both in tabular implementations and deep RL agents on Atari-57 games. We also identify a close connection between VA-learning and the dueling architecture, which partially explains why a simple architectural change to DQN agents tends to improve performance.

1. Introduction

Developed just over three decades ago, Q-learning (Watkins, 1989; Watkins and Dayan, 1992) is one of the most fundamental algorithms of reinforcement learning (RL). Q-learning progresses in an iterative fashion, updating the current value predictions by bootstrapping from its own future value predictions. In addition to its theoretical appeal, the incremental nature of Q-learning is also compatible with powerful deep learning machinery, which has fueled recent breakthroughs in Atari games (Mnih et al., 2013).

Q-learning learns the Q-function $Q(x, a)$, defined as the expected return obtained starting from certain state-action pair, and executing an optimal policy. By splitting the Q-function into a state-dependent value function $V(x)$ and a residual advantage function $A(x, a)$, we arrive at the commonly used decomposition

$$Q(x, a) = V(x) + A(x, a).$$

In many situations accurately approximating the advantage function, which measures the relative performance between actions, is the end goal of the algorithm. Instead of learning advantage functions implicitly via Q-functions, a natural question is whether it is possible to learn advantage functions directly. Unfortunately, unlike Q-functions, the advantage function does not obey a recursive equation (like the Bellman equation for Q-functions) and cannot be learned as a standalone object by bootstrapping from itself.

Our key remedy to resolving the above dilemma is *learning an extra value function at the same time*. We introduce VA-learning (Section 3), an algorithm that derives it name from the fact that it directly learns a value function V and an advantage function A. Importantly, the decomposition $Q = V + A$ does not constrain us from learning just the target value functions. In fact, as we will explain in detail, VA-learning derives its properties by learning a value function adapted to the data collection policy. On a high level, VA-learning is reminiscent of the dueling architecture for Q-learning (Wang et al., 2015), which runs a vanilla Q-learning algorithm with a parameterization that decomposes Q-functions into value and advantage functions. While the dueling architecture is purely empirically motivated, we provide grounded theoretical guarantees to the performance of VA-learning.

Besides theoretical guarantees, we also find that in practice VA-learning is generally more superior to vanilla Q-learning, in both tabular and deep RL settings. A high level explanation is that through the decomposition $Q = V + A$, VA-learning explicitly allows for an extra degree of freedom such that the learning takes place at different rate across different components of the Q-function. Concretely, we generally expect the V to be learned more quickly than A, as the former is shared across all actions. In the case of bootstrapped updates, this technique helps to increase the speed at which the advantage function and target Q-function are learned. We now highlight a few crucial detailed properties enjoyed by VA-learning.

Theoretical guarantee as Q-learning. VA-learning enjoys the same theoretical guarantee as Q-learning (Sec-

[1]Google DeepMind. Correspondence to: Yunhao Tang <robintyh@deepmind.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

tion 3). The *implied* Q-function of VA-learning $Q = V + A$ converges to the same target fixed point as Q-learning, whereas V and A converge to properly defined value and advantage function respectively.

Improved efficiency of tabular algorithms. Through the decomposition $Q = V + A$, VA-learning effectively allows the shared part of the Q-function to be learned *quickly* via the V component, and more *slowly* via the A component. When the learning targets are computed using bootstrapping, the accelerated effect of such a decomposition becomes even more profound, even in simple tabular MDPs (Section 3)

Large-scale value-based learning. VA-learning can also be used as a component within large-scale RL agents (Section 4). When implemented with function approximation, we draw an intriguing connection between VA-learning and the dueling architecture (Wang et al., 2015). On the Atari-57 game suite, VA-learning provides robust improvements over the dueling and Q-learning baselines (Section 6).

2. Background

Consider a Markov decision process (MDP) represented as the tuple $(\mathcal{X}, \mathcal{A}, P_R, P, \gamma)$ where $\mathcal{X}$ is a finite state space, $\mathcal{A}$ the finite action space, $P_R : \mathcal{X} \times \mathcal{A} \to \mathscr{P}(\mathbb{R})$ the reward kernel, $P : \mathcal{X} \times \mathcal{A} \to \mathscr{P}(\mathcal{X})$ the transition kernel and $\gamma \in [0, 1)$ the discount factor. For any policy $\pi : \mathcal{X} \to \mathscr{P}(\mathcal{A})$, important quantities include Q-function $Q^\pi(x, a) \coloneqq \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r_t \mid x_0 = x, A_0 = a\right]$, value function $V^\pi(x) \coloneqq \sum_a \pi(a|x) Q^\pi(x, a)$ and advantage function $A^\pi(x, a) \coloneqq Q^\pi(x, a) - V^\pi(x)$.

In policy evaluation, the aim is to compute the target Q-function Q^π for a fixed target policy π. The target Q-function Q^π can be approximated by applying the recursion $Q_{t+1} = \mathcal{T}^\pi Q_t$, where $\mathcal{T}^\pi : \mathbb{R}^{\mathcal{X}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ is the Bellman evaluation operator. In control, the aim is to find an optimal policy $\pi^\star(\cdot|x) \coloneqq \arg\max_a Q^*(x, a)$ with Q-function $Q^\star(x, a) \coloneqq \max_\pi Q^\pi(x, a)$. It can be can be approximated, by applying the recursion $Q_{t+1} = \mathcal{T}^\star Q_t$, with the Bellman control operator $\mathcal{T}^\star$.

In most applications, it is infeasible to compute the above recursions exactly as they require analytic knowledge of the transition and reward kernel. Instead, from a given state $x \in \mathcal{X}$, it is more common to access a sampled transition (x_t, a_t, r_t, x_{t+1}) tuple at step $t \geq 0$,

$$a_t \sim \mu(\cdot|x_t), r_t \sim P_R(\cdot|x_t, a_t), x_{t+1} \sim P(\cdot|x_t, a_t),$$

where μ is the behavior policy, which for simplicity is assumed fixed and has full coverage over the entire action space $\mu(a|x) > 0, \forall (x, a) \in \mathcal{X} \times \mathcal{A}$. Let $(p_t)_{t=0}^\infty$ and $(q_t)_{t=0}^\infty$ be any number arrays, in the following, we also use $p_{t+1} \overset{\alpha_t}{\leftarrow} q_t$ as shorthand notation for the incremental update $p_{t+1} = p_t + \alpha_t(q_t - p_t)$ with learning rate α_t.

Algorithm 1 Tabular VA-learning

Initializations $V_0 \in \mathbb{R}^{\mathcal{X}}$ and $A_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$; behavior policy μ and learning rate sequence $(\alpha_t)_{t=0}^\infty$.
for $t = 0, 1, 2, \ldots, K$ **do**
Step 1. Sample transition (x_t, a_t, r_t, x_{t+1}).
Step 2. Let $Q_t(x_t, a_t) = V_t(x_t) + A_t(x_t, a_t)$. Compute back-up target $\widehat{\mathcal{T}} Q_t(x_t, a_t)$ based on Eqn (3) for policy evaluation and Eqn (4) for control.
Step 3. Update the value and advantage iterates

$$V_{t+1}(x_t) \overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu),$$
$$A_{t+1}(x_t, a_t) \overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu) - V_t(x_t).$$

end for
Output final V_t and a_t.

Let $(Q_t)_{t=0}^\infty$ be a sequence of estimated Q-functions. First we consider the update for the policy evaluation case, which is commonly known as TD-learning,

$$Q_{t+1}(x_t, a_t) \overset{\alpha_t}{\leftarrow} r_t + \gamma Q_t(x_{t+1}, \pi), \tag{1}$$

where $Q_t(x, \pi) \coloneqq \sum_a \pi(a|x) Q_t(x, a)$. The back-up target $r_t + \gamma Q_t(x_{t+1}, \pi)$ can be understood as a stochastic approximation to the evaluation Bellman recursion back-up target $\mathcal{T}^\pi Q_t$. In the control case, the Q-learning update is

$$Q_{t+1}(x_t, a_t) \overset{\alpha_t}{\leftarrow} r_t + \gamma \max_a Q_t(x_{t+1}, a). \tag{2}$$

With a properly chosen learning rate scheme $(\alpha_t)_{t=0}^\infty$ and mild assumptions on the data process (Watkins and Dayan, 1992; Tsitsiklis, 1994; Jaakkola et al., 1994), TD-learning and Q-learning converge almost surely to Q^π or $Q^\star$ respectively.

3. VA-learning

We now introduce VA-learning, the central object of stufy of the paper. At iteration t, VA-learning maintains a value function estimate $V_t(x)$ and advantage function estimate $A_t(x, a)$. Most importantly, unlike Q-learning, VA-learning does *not* maintain a separate Q-function. To recover a Q-function estimate, VA-learning combines the value and advantage function estimate as

$$Q_t(x, a) \coloneqq V_t(x) + A_t(x, a), \forall (x, a) \in \mathcal{X} \times \mathcal{A}.$$

3.1. Policy evaluation and control

Throughout, we assume access to the transition tuple (x_t, a_t, r_t, x_{t+1}) at time t as in the TD-learning and Q-learning case. To better highlight the difference between

VA-learning and TD-learning (the difference is similar between VA-learning and Q-learning), we start by defining the policy evaluation back-up target for TD-learning,

$$\widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) \coloneqq r_t + \gamma Q_t(x_{t+1}, \pi). \tag{3}$$

The policy evaluation recursion in Eqn (1) rewrites as $Q_{t+1}(x_t, a_t) \overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t)$. In contrast, policy evaluation VA-learning carries out the following recursion:

$$\begin{aligned} V_{t+1}(x_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu), \\ A_{t+1}(x_t, a_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu) - V_t(x_t). \end{aligned}$$

(Policy evaluation VA-learning)

where we similarly define $A_t(x, \mu) \coloneqq \sum_a \mu(a|x) A_t(x, a)$.

Understanding the back-up targets. To better understand the updates, note that the back-up targets for value estimate V_t and advantage estimate A_t share the common back-up target $\widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - A_t(x_{t+1}, \mu)$. To better understand the back-up target, we rewrite it as the estimated Bellman operator $\widehat{\mathcal{T}}^{\pi}$ applied to a transformed Q-function $\widetilde{Q}_t(x_t, a_t)$,

$$\widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu) = \widehat{\mathcal{T}}^{\pi} \widetilde{Q}_t(x_t, a_t),$$

Here, the transformed Q-function $\widetilde{Q}_t(x_t, a_t) = V_t(x_t) + \widetilde{A}_t(x_t, a_t)$ has a special parameterization of its advantage function

$$\widetilde{A}_t(x_t, a_t) = A_t(x_t, a_t) - A_t(x_t, \mu)$$

such that the advantage function has zero mean $\widetilde{A}_t(x_t, \mu) = 0$ under behavior policy μ. Intriguingly, such a transformation is reminiscent of though distinct from the dueling architecture for DQN (Wang et al., 2015). We will draw further connections between VA-learning and dueling in Section 4.

We can interpret the value function estimate $V_{t+1}(x_t)$ as learning the *average* of the common back-up targets, averaged over all actions taken from state x. Meanwhile, the advantage function estimate $A_{t+1}(x_t, a_t)$ learns the *residual* of the back-up target (after subtracting the baseline $V_t(x)$). Intuitively, this hints at the fact that V_t, A_t indeed learn certain value functions and advantage functions respectively. We will make the convergence behavior and fixed points of VA-learning more clear shortly.

Control case. For the control case, we define the control back-up target similar to Q-learning,

$$\widehat{\mathcal{T}}^{\star} Q_t(x_t, a_t) \coloneqq r_t + \gamma \max_a Q_t(x_{t+1}, a). \tag{4}$$

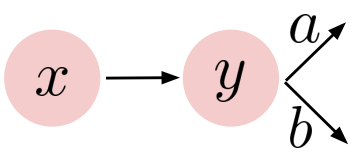


Figure 1. A simple scenario to illustrate the effectiveness of VA-learning over Q-learning. There are two states x, y and from state y there are two actions a, b. Assume there is a back-up target $\widehat{Q}(y, a)$, VA-learning will update the prediction for both $Q(y, a)$ and $Q(y, b)$ thanks to the shared value function $V(y)$. In contrary, Q-learning only updates the prediction $Q(y, a)$. The accelerated learning of $Q(y, b)$ helps accelerate learning $Q(x, \cdot)$ when bootstrapping from $Q(y, \cdot)$.

Then control VA-learning carries out the recursion:

$$\begin{aligned} V_{t+1}(x_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^{\star} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu), \\ A_{t+1}(x_t, a_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^{\star} Q_t(x_t, a_t) - \gamma A_t(x_{t+1}, \mu) - V_t(x_t). \end{aligned}$$

(Control VA-learning)

3.2. Why VA-learning can be more efficient

Before formally presenting the convergence behavior of VA-learning, we provide intuitive explanations and numerical examples to show why VA-learning can be often more efficient than TD-learning and Q-learning.

An illustrative example. Consider a fixed state y with two actions a, b. Imagine so far only action a has been sampled from state y, TD-learning or Q-learning would have updated Q-function estimate $Q(y, a)$, while the Q-function estimate $Q(y, b)$ has never been updated. Now, for any state x that precedes state y, constructing the Q-learning back-up target at state x may require bootstrapping from $Q(y, b)$. Since $Q(y, b)$ is never updated before, such a back-up target for state x is of low quality. Nevertheless, Q-learning can still work by generating more data until the action b at state y is sampled more time. However, the situation above implies that propagating the correct information from y to x can be slowed down by not having enough transitions (y, b) sampled.

For VA-learning, when the action a is sampled at state y, the back-up target for (y, a) will be split into back-up targets for $V(y)$ and $A(y, a)$. This ensures both $V(y)$ and $A(y, a)$ are updated to certain extent. Now, at the preceding state x, when bootstrapping from (y, b) to construct its back-up target, we effectively bootstrap from $Q(y, b) = V(y) + A(y, b)$. Although $A(y, b)$ has not been updated before, the bootstrap target can still utilize information contained in the updated value function estimate $V(y)$. This means the

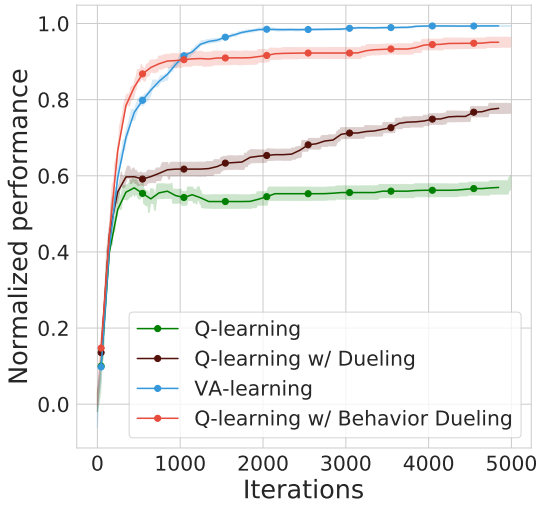


Figure 2. Comparing VA-learning (Section 3), Q-learning with behavior dueling (Section 4), Q-learning with uniform dueling (Wang et al., 2016) and regular Q-learning. We experiment on tabular MDPs with fixed behavior policy $\mu = \varepsilon u + (1-\varepsilon)\pi_{\text{det}}$ for some randomly sampled and fixed deterministic policy π_{det}, uniform policy u and $\varepsilon = 0.8$. The performance evaluates the greedy policy with learned Q-function. New algorithmic variants significantly outperform prior methods.

VA-learning back-up target at state x is already potentially more informative compared to its counterpart in Q-learning.

In summary, the potential benefits of VA-learning come from the decomposition of Q-function into a value function and an advantage function. Since the value function is shared across all actions, it can be learned faster by pooling back-up targets across all actions. When used as bootstrapped targets, the induced Q-function benefits from information contained in the value function, which in turn accelerates the learning process.

To empirically validate the above claims, we examine the performance of Q-learning and VA-learning under the policy evaluation case in tabular MDPs. We carry out recursive updates based on a fixed number of trajectories under behavior policy μ. We examine the error of the advantage estimate $\left\|\widehat{A}_t - A^\pi\right\|_2$ at update iteration k, as the advantage function error is also indicative of performance in the control case. VA-learning provides significant improvements over Q-learning both in terms of convergence speed and asymptotic accuracy. Detailed results are shown in Figure 5 (Appendix C).

Can VA-learning underperform Q-learning? VA-learning is arguably not *always* more sample efficient than Q-learning. The decomposition $Q(x,a) = V(x) + A(x,a)$, from which VA-learning is derived, assumes that it is useful to share information (i.e., $V(x)$) across actions from the same state x. When the Q-function gap from a common state $|Q(x,a) - Q(x,b)|$ is much larger than the gap between different states $|V(x) - V(y)|$, it is potentially better to learn $Q(x,a)$ and $Q(x,b)$ separately rather than sharing a common value function. Nevertheless, in many practical scenarios, we should expect the utility in sharing values across actions starting from a single state. VA-learning should generally outperform Q-learning, as we show in the following tabular and deep RL experiments.

3.3. Convergence of VA-learning

To understand the behavior of VA-learning more precisely, we consider the expected recursive update that sample-based VA-learning approximates, similar to how Q-learning approximates Bellman recursions. To facilitate the discussion, we define the notation $\mu Q \in \mathbb{R}^{\mathcal{X}}$ for any $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ as $\mu Q(x) \coloneqq \sum_a \mu(a|x) Q(x,a)$. Abusing the notation a bit, when the context is clear we also use μQ to denote a vector in $\mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ with the same value for all actions in a single state $\mu Q(x,a) \coloneqq \mu Q(x)$.

We now introduce the *VA recursion* as a counterpart to the Bellman recursion. For both policy evaluation or control, the VA recursion takes a common form

$$\begin{aligned} V_{t+1} &= \mu \mathcal{T}\left(Q_t - \mu A_t\right), \\ A_{t+1} &= \mathcal{T}\left(Q_t - \mu A_t\right) - V_t, \end{aligned} \qquad \text{(VA recursion)}$$

with $\mathcal{T} = \mathcal{T}^\pi$ for policy evaluation and $\mathcal{T} = \mathcal{T}^\star$ for control. As we show later, VA-learning is the stochastic approximation to the VA recursion. As a result, the convergence property of VA recursion obviously determines the behavior of VA-learning. We now show that the VA recursion converges to the target Q-function of interest for both policy evaluation and control.

Theorem 1. (**Convergence of VA recursion**) For the policy evaluation case, define $V_\mu^\pi(x) \coloneqq \sum_a \mu(a|x) Q^\pi(x,a)$ and $A_\mu^\pi(x,a) \coloneqq Q^\pi(x,a) - V_\mu^\pi(x)$. Let $C_\mu^\pi = \left\|V_0 - V_\mu^\pi\right\|_\infty + \left\|A_0 - A_\mu^\pi\right\|_\infty$ be the initial approximation error. The value and advantage estimates converge geometrically

$$\begin{aligned} \left\|A_t - A_\mu^\pi\right\|_\infty &\le \gamma^{t-1}(1+\gamma) C_\mu^\pi, \\ \left\|V_t - V_\mu^\pi\right\|_\infty &\le \gamma^t C_\mu^\pi, \end{aligned} \qquad \text{(policy evaluation)}$$

which also implies $\left\|Q_t - Q^\pi\right\|_\infty = \mathcal{O}(\gamma^t)$. For the control case, we define $V_\mu^\star(x) \coloneqq \sum_a \mu(a|x) Q^\star(x,a)$ and $A_\mu^\star(x,a) \coloneqq Q^\star(x,a) - V_\mu^\star(x)$. Let $C_\mu^\star = \left\|V_0 - V_\mu^\star\right\|_\infty + \left\|A_0 - A_\mu^\star\right\|_\infty$ be the initial approximation error. The value and advantage estimates converge geometrically

$$\begin{aligned} \left\|A_t - A_\mu^\star\right\|_\infty &\le \gamma^{t-1}(1+\gamma) C_\mu^\star, \\ \left\|V_t - V_\mu^\star\right\|_\infty &\le \gamma^t C_\mu^\star, \end{aligned} \qquad \text{(optimal control)}$$

which also implies $\left\|Q_t - Q^\star\right\|_\infty = \mathcal{O}(\gamma^t)$.

Proof. We show a proof sketch for the policy evaluation case, similar result holds for the control case. Define $\widetilde{Q}_t = Q_t - \mu A_t$. From the definition of VA recursion, a few calculations show $\widetilde{Q}_{k+1} = \mathcal{T}^\pi \widetilde{Q}_t$. This implies $\widetilde{Q}_t$ converges to Q^π at a geometric rate. Next, since $V_{t+1} = \mu \mathcal{T}^\pi \widetilde{Q}_t$, we have $V_t \to V_\mu^\pi$. Finally, $A_{t+1} = \mathcal{T}^\pi \widetilde{Q}_t - V_t$ implies $A_t \to Q^\pi - V_\mu^\pi$. □

Intriguingly in general, the converged value function and advantage function differs from the target functions $V_\mu^\pi \neq V^\pi, A_\mu^\pi \neq A^\pi$ (similarly for the control case). The converged value function V_μ^π be understood as the value function obtained by following μ in the first time step and π (resp. π^* for control). Intuitively, this is because the value updates aggregate across all actions according to μ without off-policy corrections. Nevertheless, the Q-function estimate constructed from the value and advantage estimate $Q_t(x,a) = V_t(x) + A_t(x,a)$ *does converge* to the target Q-function (resp., $Q^\star$ for control).

Convergence of VA-learning from VA recursion. Since VA-learning is the stochastic approximation to the VA recursion, the convergent behavior of VA recursion implies that VA-learning should converge too. Indeed, by borrowing the arguments from how TD-learning and Q-learning converge as a result of the convergence of Bellman recursion (Watkins, 1989; Watkins and Dayan, 1992; Jaakkola et al., 1993; Tsitsiklis, 1994), we can show VA-learning converges to the target fixed points above given regular assumptions on the data process and learning rate. We provide the detailed results in Appendix B.

3.4. VA-learning with function approximation

VA-learning is readily compatible with function approximations. In general, consider parameterizing the value function V_θ and advantage function A_ϕ with parameters θ and ϕ. The Q-function can be computed as $Q_{\theta,\phi}(x,a) \coloneqq V_\theta(x) + A_\phi(x,a)$. Let θ^-, ϕ^- be the target network parameter (Mnih et al., 2013) which is slowly updated towards θ and ϕ. Henceforth, we will focus on the policy evaluation case, similar discussions hold for the control case. Given a transition tuple (x_t, a_t, r_t, x_{t+1}), we can construct the back-up value and advantage target based on the tabular VA-learning update,

$$\begin{aligned} \widehat{V}(x_t) &= \widehat{Q}^\pi(x_t,a_t) - \gamma A_{\phi^-}(x_{t+1},\mu), \\ \widehat{A}(x_t,a_t) &= \widehat{Q}^\pi(x_t,a_t) - \gamma A_{\phi^-}(x_{t+1},\mu) - V_{\theta^-}(x_t), \end{aligned} \tag{5}$$

where recall that $\widehat{Q}^\pi(x_t,a_t) = r_t + \gamma Q_{\theta^-,\phi^-}(x_{t+1},\pi)$. The VA-learning update rule can be implemented by minimizing the least square loss function $L_{\text{VA}}(\theta,\phi)$ defined as

$$\frac{1}{2}\left(V_\theta(x_t) - \widehat{V}(x_t)\right)^2 + \frac{1}{2}\left(A_\phi(x_t,a_t) - \widehat{A}(x_t,a_t)\right)^2. \tag{6}$$

When the behavior policy is unknown and we only have access to samples (x_t, a_t), in order to calculate the back-up target defined in Eqn (5), we need a policy μ_ψ that keeps track of the average behavior $\mu_\psi(a|x) \approx \mathbb{E}\left[\mathbb{I}\left[x_t = a\right] \mid x_t = x\right]$. This can be achieved by maximizing the likelihood $\log \mu_\psi(a|x)$ on observed transitions (x_t, a_t). The full VA-learning algorithm with function approximation is shown in Algorithm 2.

Algorithm 2 VA-learning with function approximation

Parameterize value and advantage function $Q_{\theta,\phi}(x,a) = V_\theta(x) + A_\phi(x,a)$. Target network (θ^-, ϕ^-).
for $t = 1, 2...$ **do**
Step 1. Sample transition (x_t, a_t, r_t, x_{t+1}).
Step 2. Learn average behavior policy

$$\psi \leftarrow \psi + \eta \nabla_\psi \log \mu_\psi(a_t|x_t).$$

Step 3. Compute targets $\widehat{V}(x_t), \widehat{A}(x_t,a_t)$ based on Eqn (5), and update online network parameter using gradient based on VA-learning loss function in Eqn (6):

$$(\theta,\phi) \leftarrow (\theta,\phi) - \eta \nabla_{(\theta,\phi)} L_{\text{VA}}(\theta,\phi).$$

end for
Output the final Q-function $Q_{\theta,\phi}$.

4. Behavior dueling architecture

Thus far, we have showed that the VA-learning advantage function estimate A_t converges to A_μ^π for policy evaluation (resp. $A_\mu^\star$ for control). By definition, such advantage functions satisfy the following *zero-mean* property

$$\begin{aligned} A_\mu^\pi(x,\mu) &\coloneqq \sum_a \mu(a|x) A_\mu^\pi(x,a) = 0 \\ A_\mu^*(x,\mu) &\coloneqq \sum_a \mu(a|x) A_\mu^\star(x,a) = 0. \end{aligned} \tag{7}$$

At any finite iteration t, the estimate A_t does not necessarily satisfy the above property. Since we know the zero-mean property that the converged value of A_t satisfies, it is tempting to enforce such a property as a constraint on A_t, which does not change the fixed point of the update. In the function approximation case, such a zero-mean constraint might be a useful inductive bias for parameterizing the advantage function. For example, we let $f_\phi(x,a)$ be an unconstrained function, and define its average over actions $f_\phi(x,\mu) \coloneqq \sum_a \mu(a|x) f_\phi(x,a)$. We parameterize the

zero-mean advantage function as follows

$$A_\phi(x,a) \coloneqq f_\phi(x,a) - f_\phi(x,\mu), \tag{8}$$

such that $A_\phi(x,\mu) \coloneqq \sum_a \mu(a|x) A_\phi(x,a) = 0$. We call the above parameterization *behavior dueling* due to its close connections to the dueling architecture (Wang et al., 2015) and the fact that we parameterize the advantage to be zero-mean under the behavior policy. The regular dueling architecture is a special case when μ is uniform. The full-fledged Q-learning with behavior dueling algorithm is shown in Algorithm 2.

4.1. Connections between VA-learning and Q-learning with behavior dueling

Our key insight is that VA-learning bears close conceptual connections to regular TD-learning (or Q-learning) with behavior dueling parameterization. With behavior dueling, TD-learning or Q-learning might benefit from the value sharing parameteirzation and the inductive bias for learning advantage functions. To better see the connections, note that the TD-learning algorithm minimizes the least square loss function with respect to the parameterized Q-function $Q_{\theta,\phi}$:

$$L_{\mathrm{QL}}(\theta,\phi) = \frac{1}{2}\left(Q_{\theta,\phi}(x_t,a_t) - \widehat{Q}^\pi(x_t,a_t)\right)^2, \tag{9}$$

where $\widehat{Q}^\pi(x_t,a_t) = r_t + \gamma Q_{\theta^-,\phi^-}(x_{t+1},\pi)$ is the one-step back-up target. With behavior dueling, $Q_{\theta,\phi}(x,a) = V_\theta(x) + A_\phi(x,a)$ and $A_\phi(x,a) = f_\phi(x,a) - f_\phi(x,\mu)$. We examine the gradient of Q-learning loss function $L_{\mathrm{QL}}(\theta,\phi)$ with respect to the value parameter $\nabla_\theta L_{\mathrm{QL}}(\theta,\phi)$ is

$$\left(V_\theta(x_t) - \left(\widehat{Q}^\pi(x_t,a_t) - A_\phi(x_t,a_t)\right)\right)\nabla_\theta V_\theta(x_t).$$

We can interpret the gradient for value parameter θ as updating the value function V_θ so as to better fit the value function target $\widehat{Q}^\pi(x_t,a_t) - A_\phi(x_t,a_t)$. This echos with the value updates in VA-learning that which aim to fit a value function target (see Section 3).

Regarding the advantage updates, there is a subtle difference between the advantage updates of VA-learning vs. behavior dueling. In a nutshell, this is because VA-learning carries out separate updates for each advantage function $A(x,a)$. In contrast, behavior dueling couples advantage updates for different actions due to the parameterization $A(x,a) = f(x,a) - f(x,\mu)$; as a result, when action $a \neq b$ is taken, the advantage function $f(x,b)$ is updated as well. Despite the difference, both updates bear the interpretations of fitting the advantage components of the Q-function. Such interpretations imply that the motivational example (Section 3.2) which illustrates that the benefits of VA-learning should intuitively apply to behavior dueling as well, as we will verify empirically. We present a more complete discussion of such connections between VA-learning and behavior dueling in Appendix D.

Algorithm 3 Q-learning with behavior dueling

Behavior dueling $Q_{\theta,\phi}(x,a) = V_\theta(x) + A_\phi(x,a)$ with parameterization $A_\phi(x,a) = f_\phi(x,a) - f_\phi(x,\mu_\psi)$. Target network (θ^-,ϕ^-).
for $t = 1, 2...$ **do**
Step 1. Sample transition (x_t, a_t, r_t, x_{t+1}).
Step 2. Learn average behavior policy

$$\psi \leftarrow \psi + \eta \nabla_\psi \log \mu_\psi(a_t|x_t).$$

Step 3. Compute back-up target

$$\widehat{Q}^\pi(x_t,a_t) = r_t + \gamma Q_{\theta^-,\phi^-}(x_{t+1},\pi).$$

and update online network parameter using gradient based on Eqn (9): $(\theta,\phi) \leftarrow (\theta,\phi) - \eta \nabla_{(\theta,\phi)} L_{\mathrm{QL}}(\theta,\phi)$.
end for
Output the final Q-function $Q_{\theta,\phi}$.

4.2. Why behavior dueling is better than dueling

We can understand the dueling architecture (Wang et al., 2015) as a special case of behavior dueling assuming μ is uniform. Such an implicit assumption can be useful when μ is indeed close to uniform, so that there is no need to parameterize an additional behavior policy μ_ψ to learn. However, when the behavior policy deviates from the uniform policy, learning $\mu_\psi \approx \mu$ seems critical to improved performance. In a few practical setups, we usually find behavior dueling to outperform uniform dueling, as we will demonstrate in both tabular and some large-scale deep RL settings.

In light of the discussion in Section 3.2, both behavior and uniform dueling entail sharing information across actions, so why does the former perform better? We hypothesize that this is because behavior dueling entails a better value sharing between actions, as it is adapted to the behavior policy. Consider the dueling parameterization $A_\nu(x,a) = f(x,a) - f(x,\nu)$ with distribution ν. We are interested in minmizing unshared information $A_\mu(x,a)$ across actions, as characterized by the squared norm

$$\min_\nu \sum_a \mu(a|x) A_\nu(x,a)^2$$

In general, the minimizing distribution is $\nu = \mu$, i.e., the behavior policy. Since behavior dueling at $\nu = \mu$ minimizes the unshared components of Q-functions, it can be interpreted as maximizing the shared components, leading to faster downstream learning. We provide a more precise argument in Appendix D with experiment ablations.

5. Discussion of prior work

We discuss the relation between VA-learning and a few lines of related work in RL.

Advantage learning. Despite the similarity in names, VA-learning differs from advantage learning (Baird III, 1993; Baird, 1995) in critical ways. In a nutshell, VA-learning still aims to learn the original Q-function Q^π (or Q^* in the control case), whereas advantage learning learns to increase the value gaps between actions. Specifically, given a transition (x_t, a_t, r_t, x_{t+1}), advantage learning for optimal control can be understood as the following back-up target for $Q_t(x_t, a_t)$ (Bellemare et al., 2016; Kozuno et al., 2019)

$$\widehat{\mathcal{T}}^\pi Q_t(x_t, a_t) + \beta \left(Q_t(x_t, a_t) - \sum_a \pi(a_t|x_t) Q_t(x_t, a_t) \right)$$

where π is the greedy policy for the control case. The above back-up operation is gap-increasing, in that it enlarges the difference between converged Q-functions at different actions. For example, in the policy evaluation case the Q-function converges to $V^\pi + \frac{1}{1-\beta} A^\pi$. As $\beta \to 1$, the Q-function gap between actions increases.

Compared to gap-increasing operators, a subtle technical difference is that VA-learning constructs the back-up target by subtracting the average Q-function under behavior policy μ instead of target policy π; and at the next state x_{t+1}, instead of the current state x_t. This ensures that VA-learning still retains Q^π as the fixed point. An interesting future direction would be to combine VA-learning with the gap-increasing learning.

Direct advantage learning. With a similar motivation as VA-learning, Pan et al. (2021) proposed to learn advantage functions directly from Monte-Carlo returns, based on the variance-minimization property of advantage function. Their approach is thus far constrained to the on-policy case, and does not allow for bootstrapping out-of-the-box. An interesting direction would be to study the combination of such an approach with VA-learning.

RL with over-parameterized linear function approximation. The tabular dueling parameterization can be understood as a special case of over-parameterized linear function approximation (Sutton and Barto, 1998). Here, *over-parameterized* refers to the fact that dueling introduces an extra degree of freedom to learning Q-functions. We have demonstrated empirically that this extra degree of freedom allows for value sharing across actions, and usually helps speed up convergence. Interesting open questions include the study of off-policy stability of dueling parameterization.

6. Experiments

We start with experiments on tabular MDPs, to understand the improved sample efficiency of VA-learning over Q-learning. Then we evaluate the impacts of VA-learning and behavior dueling in deep RL settings.

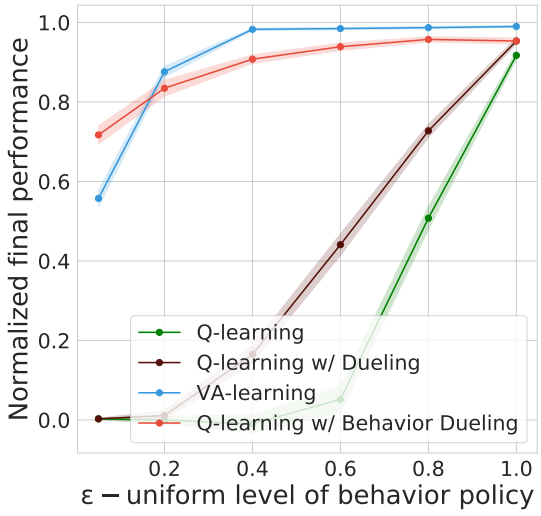


Figure 3. Comparing different algorithmic variants in tabular MDPs with fixed behavior policy $\mu = \varepsilon u + (1-\varepsilon)\pi_{\text{det}}$ for some randomly sampled and fixed deterministic policy π_{det}, uniform policy u and varying degree of ε (x-axis). As $\varepsilon \to 1$ and μ approaches uniform, Q-learning with dueling architecture catches up in performance with behavior dueling and VA-learning.

6.1. Tabular MDP experiments

In Figure 3, we compare four algorithmic variants with behavior policy μ defined as $\mu = \varepsilon u + (1-\varepsilon)\pi_{\text{det}}$ on a family of randomly generated tabular MDPs. Here, π_{det} is a randomly sampled deterministic policy, u is the uniform policy and $\varepsilon \in [0, 1]$ is the mixing coefficient. We calculate the final performance of each algorithm until convergence, and show the mean and standard deviation across 20 independent runs. See Appendix C for more details on the MDP details.

For a wide range of values of ε, VA-learning and Q-learning with behavior dueling outperform other baselines significantly. The performance gap seems the most profound when $\varepsilon \approx 0$ as μ deviates the most from uniform. In this case, we speculate that since the behavior dueling architecture makes an inaccurate implicit assumption on the behavior policy, Q-learning with uniform dueling perform poorly as regular Q-learning. As ε increases, the performance gap decreases. When $\varepsilon \to 1$ and μ is close to a uniform policy, uniform dueling catches up with the two new algorithms. However, there is still a statistically significant gap between regular Q-learning and other algorithms, implying a consistent benefit of VA-learning and its derived Q-learning variants (behavior dueling) over regular Q-learning.

6.2. Deep reinforcement learning experiments

We now evaluate the effects of VA-learning and its variants in large-scale deep RL environments. We use the DQN

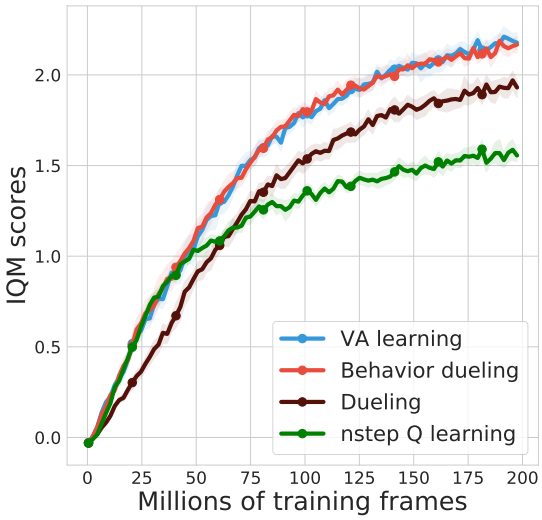


Figure 4. Comparing algorithmic variants implemented with full Atari action set. VA-learning and behavior dueling are significantly better than the uniform dueling architecture, which further improves over the n-step Q-learning baseline. Compared to the standard Atari setup in Figure 6(b), the performance of VA-learning and behavior dueling does not degrade.

agent (Mnih et al., 2013) as the baseline agent and use the Atari 57 game suite as the test bed (Bellemare et al., 2013). Throughout, we report the interquartile mean (IQM) score across multiple random seeds for all algorithmic variants that train for 200M frames (Agarwal et al., 2021). We compare VA-learning, behavior dueling, dueling (Wang et al., 2015) and baseline Q-learning. All variants share the same architecture and hyper-parameters wherever possible.

The behavior policy μ is ε-greedy with respect to the Q-function network $Q_{\theta,\phi}$. Since both the exploration rate ε and Q-function $Q_{\theta,\phi}$ slowly changes over time, the behavior policy μ changes too. VA-learning and behavior dueling trains an additional average behavior policy $\mu_\psi(a|x)$ to approximate the average behavior policy across the entire training history. By default, to improve performance, the baseline Q-learning agent implements n-step bootstrapping and computes back-up targets based on partial trajectories of length n. VA-learning can be easily adapted accordingly, see Appendix C for more details.

Network architecture. The baseline DQN agent network consists of a *torso* convolutional network which processes the input image x into an embedding $g(x)$, and a *head* MLP network which takes the embedding and outputs the Q-function $Q_\theta(g(x), a)$. The dueling architecture parameterizes a separate value *head* network $V_\theta\left(g(x)\right)$ and advantage *head* network $A_\phi(g(x), a)$. In behavior dueling and VA-learning, the behavior policy is parameterized as a policy *head* network that outputs a distribution over actions $\mu_\psi\left(a|g(x)\right)$. Throughout experiments, we design the policy head to share the same torso as the other value heads, but its gradient does not update the torso parameters. This design choice ensures that the loss function for learning average behavior policy does not shape the embedding. Hence, any resulting empirical gains can be more convincingly attributed to the improvements of VA-learning over Q-learning. See Appendix C for more comprehensive details.

Full action set Atari. We focus on a variant of the Atari game suite with *full action set*, where the agent has access to a total of $|\mathcal{A}| = 18$ actions including potentially many actions with no effect. Thus far by default, agents are trained with the restricted action set which makes learning easier (e.g., for Pong the reduced action set has $|\mathcal{A}| = 3$ actions).

In Figure 4, we compare DQN agent variants with the full action set. Almost all algorithms can reach a similar asymptotic performance as with the restricted action set (Figure 6(b)) but the learning speed is generally slowed down. VA-learning and behavior dueling are the least impacted by the increased action set. The dueling architecture slows down more significantly, with the performance margins against VA-learning enlarged over time. As discussed in Section 4, the dueling architecture can be understood as imposing an implicit uniform assumption on the behavior policy. When the action space is large, such an assumption is more easily violated as the agent is much more likely to take certain actions than others over time. Such a comparison highlights the practical importance of using the behavior policy to carry out the average of advantage function, which is the design principle of VA-learning.

For the restricted Atari game setting where for each game only a small subset of full action sets is provided to the agent, we observe that behavior dueling and VA-learning also deliver improvements over dueling and Q-learning baselines. See Appendix C for more results and ablation study in the deep RL setting.

7. Conclusion

In this work, we have developed VA-learning as an alternative value-based RL algorithm to the classic Q-learning. We have discussed a few important theoretical aspects of VA-learning, and how it can be implemented with function approximations. With the extra degree of freedom in place, VA-learning aims to learn a value function and advantage function that is adapted to the behavior policy. Compared to Q-learning, VA-learning makes more efficient use of finite samples and enjoys better empirical performance in both tabular and deep RL settings. VA-learning also inspires the behavior dueling architecture, which generalizes dueling as a special case, and potentially explains why such a seemingly simple architecture change helps improve DQN.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, 2021.

Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*. 1995.

Leemon C. Baird III. Advantage updating. Technical report, Wright Lab Wright-Patterson AFB OH, 1993.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Geoffrey Hinton and Tijman Tieleman. Lecture 6.5 - RMSProp. *COURSERA: Neural Networks for Machine Learning*, 2012.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6, 1993.

Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

Tadashi Kozuno, Dongqi Han, and Kenji Doya. Gap-increasing policy evaluation for efficient and noise-tolerant reinforcement learning. *arXiv preprint arXiv:1906.07586*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *arXiv preprint arXiv:2109.06093*, 2021.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents. URL http://github.com/deepmind/dqn_zoo.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

John N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine learning*, 16(3):185–202, 1994.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2015.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *Proceedings of the International Conference on Learning Representations*, 2016.

Christopher J. C. H. Watkins. Learning from delayed rewards. 1989.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

APPENDICES: VA-learning as a more efficient alternative to Q-learning

A. Proof of theoretical results

Taking policy evaluation as an example, we first show that the VA-learning update

$$V_{t+1}(x_t) \overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^\pi Q_t(X,A) - A_t(x_{t+1},\mu),$$
$$A_{t+1}(x_t,a_t) \overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}^\pi Q_t(X,A) - A_t(x_{t+1},\mu) - V_t(x_t).$$

is reduced to the above VA recursion in expectation,

$$V_{t+1} = \mu\mathcal{T}^\pi\left(Q_t - \mu A_t\right),$$
$$A_{t+1} = \mathcal{T}^\pi\left(Q_t - \mu A_t\right) - V_t.$$

We first consider the value function update. Conditional on the initial state X, we take an expectation over the random variables

$$a_t \sim \mu(\cdot|x_t), r_t \sim P_R(\cdot|x_t,a_t), x_{t+1} \sim P(\cdot|x_t,a_t).$$

This leads to

$$\mathbb{E}\left[\widehat{\mathcal{T}}^\pi Q_t(x_t,a_t) - A_t(x_{t+1},\mu) \;\middle|\; x_t\right] = \sum_a \mu(a|x_t)\mathcal{T}^\pi\tilde{Q}_t(x_t,a),$$

where $\tilde{Q}_t \coloneqq Q_t - \mu A_t$. In our notation, this is equivalent to $\mu\mathcal{T}^\pi(X)$. This means the value function update in expectation is indeed $V_{t+1}(x) = \sum_a \mu(a|x)\mathcal{T}^\pi\tilde{Q}_t(x,a)$. With the same set of argument, we can show the case for the advantage function update too.

Theorem 1. (**Convergence of VA recursion**) For the policy evaluation case, define $V_\mu^\pi(x) \coloneqq \sum_a \mu(a|x)Q^\pi(x,a)$ and $A_\mu^\pi(x,a) \coloneqq Q^\pi(x,a) - V_\mu^\pi(x)$. Let $C_\mu^\pi = \left\|V_0 - V_\mu^\pi\right\|_\infty + \left\|A_0 - A_\mu^\pi\right\|_\infty$ be the initial approximation error. The value and advantage estimates converge geometrically

$$\begin{aligned}\left\|A_t - A_\mu^\pi\right\|_\infty &\le \gamma^{t-1}(1+\gamma)C_\mu^\pi,\\ \left\|V_t - V_\mu^\pi\right\|_\infty &\le \gamma^t C_\mu^\pi,\end{aligned} \qquad \text{(policy evaluation)}$$

which also implies $\left\|Q_t - Q^\pi\right\|_\infty = \mathcal{O}(\gamma^t)$. For the control case, we define $V_\mu^\star(x) \coloneqq \sum_a \mu(a|x)Q^\star(x,a)$ and $A_\mu^\star(x,a) \coloneqq Q^\star(x,a) - V_\mu^\star(x)$. Let $C_\mu^\star = \left\|V_0 - V_\mu^\star\right\|_\infty + \left\|A_0 - A_\mu^\star\right\|_\infty$ be the initial approximation error. The value and advantage estimates converge geometrically

$$\begin{aligned}\left\|A_t - A_\mu^\star\right\|_\infty &\le \gamma^{t-1}(1+\gamma)C_\mu^\star,\\ \left\|V_t - V_\mu^\star\right\|_\infty &\le \gamma^t C_\mu^\star,\end{aligned} \qquad \text{(optimal control)}$$

which also implies $\left\|Q_t - Q^\star\right\|_\infty = \mathcal{O}(\gamma^t)$.

Proof. We first examine the policy evaluation case. Define $\widetilde{Q}_t = Q_t - \mu A_t$. From the definition of VA recursion, we have

$$\begin{aligned}\widetilde{Q}_{k+1} &= V_{t+1} + A_{t+1} - \mu A_{t+1}\\ &= \mu\mathcal{T}^\pi(Q_t - \mu A_t) + \mathcal{T}^\pi(Q_t - \mu A_t) - V_t - \mu\left(\mathcal{T}^\pi(Q_t - \mu A_t)\right)\\ &= \mathcal{T}^\pi(Q_t - \mu A_t)\\ &= \mathcal{T}^\pi\widetilde{Q}_t.\end{aligned}$$

The above equality implies $\widetilde{Q}_t$ converges to Q^π at a geometric rate, since the operator $\mathcal{T}^\pi$ has Q^π as the unique fixed point and is γ-contractive. Formally, we have

$$\left\|\widetilde{Q}_t - Q^\pi\right\|_\infty \le \gamma^t\left\|\widetilde{Q}_0 - Q^\pi\right\|_\infty \le_{(a)} \gamma^t \underbrace{\left(\left\|A_0 - A_\mu^\pi\right\|_\infty + \left\|V_0 - V_\mu^\pi\right\|_\infty\right)}_{=:C_\mu^\pi}.$$

Here, (a) follows from the application of triangle inequality and the fact that $Q^\pi = A^\pi_\mu + V^\pi_\mu$. Now, we can write

$$\left\|V_t - V^\pi_\mu\right\|_\infty = \left\|\mu\mathcal{T}^\pi \widetilde{Q}_{k-1} - V^\pi_\mu\right\|_\infty = \left\|\mu\widetilde{Q}_k - V^\pi_\mu\right\|_\infty \leq \gamma^t C^\pi_\mu.$$

Finally, we consider the advantage function.

$$\begin{aligned}\left\|A_t - A^\pi_\mu\right\|_\infty = \left\|\mathcal{T}^\pi \widetilde{Q}_{k-1} - V_t - A^\pi_\mu\right\|_\infty &= \left\|\widetilde{Q}_t - V_{k-1} - A^\pi_\mu\right\|_\infty \\ &\leq_{(a)} \left\|\widetilde{Q}_t - Q^\pi\right\|_\infty + \left\|V_{k-1} - V^\pi_\mu\right\|_\infty \\ &\leq \gamma^{t-1}(1+\gamma)C^\pi_\mu,\end{aligned}$$

where (a) follows from the application of triangle inequality. This concludes the proof for policy evaluation. For optimal control, the same set of argument applies thanks to the fact that $\mathcal{T}$ is γ-contractive with $Q^\star$ as the unique fixed point. □

B. Convergence of VA-learning

We now present results on the convergence of VA-learning under stochastic approximations. Recall that upon observing the sample (x_t, a_t, r_t, x_{t+1}), VA-learning carries out the following update ,

$$\begin{aligned}V_{t+1}(x_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}Q_t(X,A) - A_t(x_{t+1}, \mu), \\ A_{t+1}(x_t, a_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}Q_t(X,A) - A_t(x_{t+1}, \mu) - V_t(x_t),\end{aligned}$$

where $\widehat{\mathcal{T}}Q_t(X,A)$ is the one-sample stochastic approximation to $\mathcal{T}Q_t(X,A)$ for optimal control and $\mathcal{T}^\pi Q_t(X,A)$ for policy evaluation. We consider a more restrictive setup, where from each state x, we sample action $A_x \sim \mu(\cdot|x)$, and observe the corresponding immediate reward $R_x \sim P_R(\cdot|x,A)$ and next state transition $X_x \sim P(\cdot|x,A)$, where the subscripts are meant to distinguish between samples from different state $x \in \mathcal{X}$. The update is carried out across all states simultaneously, $\forall x \in \mathcal{X}$,

$$\begin{aligned}V_{t+1}(x_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}Q_t(x_t,a_t) - A_t(x_{t+1}, a^\mu_x), \\ A_{t+1}(x_t, a_t) &\overset{\alpha_t}{\leftarrow} \widehat{\mathcal{T}}Q_t(x_t,a_t) - A_t(x_{t+1}, a^\mu_x) - V_t(x_t).\end{aligned} \tag{10}$$

The formal results are stated as follows.

Theorem 2. (**Convergence of VA-learning**) Under the assumption $\sum_{t=0}^\infty \alpha_t = \infty$ and $\sum_{t=0}^\infty \alpha_t^2 \leq C < \infty$ where C is some finite constant, then the above update in Eqn (10) leads to almost sure convergence of the iterates. Concretely,

$$V_t(x) \to V^\pi_\mu, A_t(x) \to A^\pi_\mu, \forall (x,a) \in \mathcal{X} \times \mathcal{A}$$

almost surely for policy evaluation and

$$V_t(x) \to V^\star_\mu, A_t(x,a) \to A^\star_\mu, \forall (x,a) \in \mathcal{X} \times \mathcal{A}$$

for optimal control.

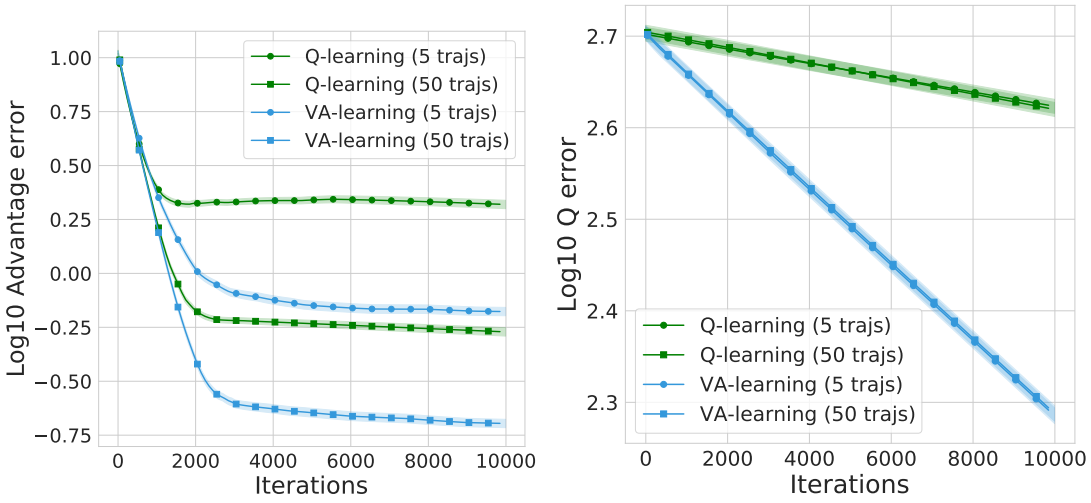
The proof is a straightforward extension of classic proof technique to show the stochastic approximation convergence of Q-learning and TD-learning (Watkins and Dayan, 1992).

C. Experiment details and extra results

We provide further details on the tabular and deep RL experiments in the main paper.

C.1. Tabular experiments

All tabular experiments in the paper are carried out on randomly generated MDPs with $|\mathcal{X}| = 20$ states, $|\mathcal{A}| = 5$ actions and discount factor $\gamma = 0.99$. The transition matrix $p(\cdot|x,a)$ is generated from a Dirichlet distribution with parameter



(a) Advantage error (b) Q-function error

Figure 5. (a) Comparing VA-learning (Section 3) with Q-learning for tabular policy evaluation. We evaluate a target policy π formed as a convex combination of a deterministic policy and uniform policy, using a fixed number of trajectories data collected under uniform policy. The y-axis shows the approximation error to the advantage $\left\|\widehat{A}_t - A^\pi\right\|_2$ at each iteration k. Given any data budget, VA-learning obtains more accurate approximations to the advantage function compared to Q-learning. (b) The same setup as before. The y-axis shows the approximation error to the Q-function $\left\|\widehat{Q}_t - Q^\pi\right\|_2$ at each iteration k. Given any data budget, VA-learning obtains a slightly faster rate of approximating the Q-function compared to Q-learning.

$(\alpha, \alpha ... \alpha)$ for $\alpha = 0.5$. For the control case, the behavior policy is fixed and constructed as $\mu = \varepsilon u + (1-\varepsilon)\pi_{\text{det}}$ for some randomly sampled and fixed deterministic policy π_{det}, uniform policy u and $\varepsilon \in [0, 1]$. By adjusting ε, we can assess different algorithms' robustness to the level of stochasticity in the behavior policy. In the policy evaluation case, μ is set to be uniform and equivalently $\varepsilon = 1$. The trajectories are collected starting from the same state $x = 0$ and under behavior policy μ. Trajectories are truncated at length $T = \text{int}\left(\frac{2}{1-\gamma}\right)$ where $\text{int}(x)$ denotes the closest integer to x. By default, $N = 20$ are collected for each experiment.

In the control case, the performance is calculated as Q^{π_t} where π_t is the greedy policy with learned Q-function Q_t. Recall that for VA-learning, $Q_t(X, A) = V_t(x) + A_t(x, a)$. In plots, we show the average value of Q^{π_t} uniformly across all states. In the policy evaluation case, we calculate $\|Q_t - Q^\pi\|_2$ where π is the randomly chosen deterministic policy.

Figure 2, we demonstrate how VA-learning improves over Q-learning in the policy evaluation case, by measuring the advantage estimation error $\left\|\hat{A}_t - A^\pi\right\|_2$ over time. For VA-learning, $\hat{A}_t = A_t$; for Q-learning, $\hat{A}_t = Q_t - \pi Q_t$.

Gradient descent updates. Throughout tabular experiments, we implement updates as gradient descents on a certain properly defined loss functions. We always adopt tabular parameterizations of Q_t, V_t and a_t. For Q-learning, the loss function is implemented as in L_{QL}; for the dueling architecture, the loss function is the same as Q-learning but with dueling parametrization on $Q_t(X, A)$; for VA-learning, the loss function is implemented as in L_{VA}. At each update, the gradient is averaged across all collected trajectories so as to avoid additional randomness in the update. The learning rate is set as a constant $\alpha_t = 0.1$. The target parameter is copied to be the online parameter every $\tau = 10$ updates.

Extra results on Q-function error. As complementary results to Figure 2, we demonstrate how VA-learning improves over Q-learning in the policy evaluation case in Figure 5(b). We measure the Q-function estimation error $\left\|\hat{Q}_t - Q^\pi\right\|_2$ over time. For VA-learning, $\hat{Q}_t = A_t + V_t$; for Q-learning, $\hat{Q}_t = Q_t$. Notably, VA-learning achieves a slightly faster decaying rate of the approximation error compared to Q-learning.

C.2. Deep RL experiment details

All the deep RL experiments use the DQN agent (Mnih et al., 2013) as the baseline agent and use the Atari 57 game suite as the test bed (Bellemare et al., 2013). To improve the performance, we apply double Q-learning (Van Hasselt et al., 2016) and n-step bootstrapping in general. Throughout, we use $n = 5$; if n is smaller, overall DQN does not benefit fully from multi-step bootstrapping; if n is larger, the performance can suffer due to lack of off-policy corrections. The agent adopts most architecture and hyper-parameters as reported in (Mnih et al., 2013). Our agents are all based on the reference implementation in (Quan and Ostrovski).

We report the interquartile mean (IQM) score across multiple random seeds for all algorithmic variants that train for 200M frames (Agarwal et al., 2021). See the reference for specific procedures for calculating the IQM score and the bootstrapped confidence intervals.

Multi-step bootstrapping, n-step Q-learning and VA-learning. Multi-step bootstrapping usually improves practical performance of deep RL algorithms (Hessel et al., 2018). In n-step Q-learning, the agent samples a partial trajectory $(X_{0:t}, A_{0:t-1}, R_{0:t-1})$ starting from state-action pair (x_t, a_t), and constructs the policy evaluation back-up target with target policy π,

$$\widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n Q_t(x_n, \pi)$$

For control, the back-up target is

$$\widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \max_a Q_t(x_n, a).$$

Finally, n-step Q-learning carries out the update $Q_{t+1}(x_t, a_t) \overset{\alpha_t}{=} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t)$. VA-learning can adapt to n-step bootstrapping as follows:

$$V_{t+1}(x_0) \overset{\alpha_t}{=} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - A_t(x_n, \mu)$$
$$A_{t+1}(x_t, a_t) \overset{\alpha_t}{=} \widehat{\mathcal{T}}^{\pi} Q_t(x_t, a_t) - A_t(x_n, \mu) - V_t(x_0).$$

When $n = 1$, the above recovers the VA-learning introduced in the paper as a special case.

Details on network architecture. We use the standard DQN architecture specified in (Mnih et al., 2013). As described in the main paper, given 4 stacked frames from the Atari game as input state x, the *torso* convolutional neural network processes the image into an embedding $g(x)$. The DQN agent parameterizes a value *head*, which is a MLP that takes $g(x)$ and produces $|\mathcal{A}|$ scalar outputs, each corresponding to a Q-function prediction $Q_\theta(g(x), a)$. The dueling architecture, behavior dueling and VA-learning all parameterize a separate value *head* with one scalar output $V_\theta\left(g(x)\right)$, and an advantage *head* with $|\mathcal{A}|$ scalar outputs $A_\phi\left(g(x), a\right)$. VA-learning and behavior dueling further parameterize a policy *head* network $\mu_\psi\left(g(x), a\right)$ that outputs a probability distribution over actions. The overall Q-function is then produced as

$$Q_{\theta,\phi}\left(g(x), a\right) = V_\theta\left(g(x)\right) + A_\phi(x, a) - \sum_a \mu_\psi\left(g(x), a\right) A_\phi\left(g(x), a\right).$$

The torso parameters g are trained with the Q-learning or VA-learning loss function. We put a stop gradient on the torso embedding $g(x)$ when calculating the learned behavior policy distribution, such that the behavior learning loss function does not impact g.

Tuning learning rate. Learning rate is the only hyper-parameter we tune across DQN agent variants. All agents use the RMSProp optimizer (Hinton and Tieleman, 2012). By default, one-step DQN agent uses the learning rate $\beta = 2.5 \cdot 10^{-4}$. When using n-step Q-learning with $n = 5$, we find the learning rate is best set smaller to be at $5 \cdot 10^{-5}$. When doing VA-learning, behavior dueling and uniform dueling, we find it improves performance further by reducing the learning rate more, to $1.5 \cdot 10^{-5}$. All learning rates are found by grid search: we start with the default learning rate β of DQN, and experiment on a subset of games whether setting learning rates at $\frac{1}{3}\beta$ or 3β improves the performance. We keep iterating until changing the learning rate does not improve performance anymore.

Using online network in VA-learning. In theory, the VA-learning loss function

$$L_{\text{VA}}(\theta, \phi) = \frac{1}{2}\left(V_\theta(x_t) - \widehat{V}(x_t)\right)^2 + \frac{1}{2}\left(A_\phi(x_t, a_t) - \widehat{A}(x_t, a_t)\right)^2,$$

where the back-up targets

$$\widehat{V}(x_t) = \widehat{Q}^\pi(x_t, a_t) - A_{\phi^-}(x_{t+1}, \mu),$$
$$\widehat{A}(X, A) = \widehat{Q}^\pi(x_t, a_t) - A_{\phi^-}(x_t, \mu) - V_{\theta^-}(x_t),$$

are computed from the target network. In deep RL implementations, we find it is important to use online network as the baseline when calculating the back-up target for the advantage function. Effectively, the back-up targets are

$$\widehat{V}(x_t) = \widehat{Q}^\pi(x_t, a_t) - A_{\phi^-}(x_{t+1}, \mu),$$
$$\widehat{A}(x_t, a_t) = \widehat{Q}^\pi(x_t, a_t) - A_{\phi^-}(x_t, \mu) - V_\theta(x).$$

Such a subtle change in implementation brings VA-learning and behavior dueling more similar in practice.

Huber loss. In practice, instead of optimizing the least square loss x^2 function, prior work has identified that optimizing the Huber loss is a more robust alternative (Quan and Ostrovski)

$$\text{huber}(x) = x^2 \mathbb{I}\left[|x| \leq \tau\right] + |x| \mathbb{I}\left[|x| > \tau\right],$$

where by default $\tau = 1$. As a result, the implemented VA-learning loss function is

$$L_{\text{VA}}(\theta, \phi) = \frac{1}{2}\text{huber}\left(V_\theta(x_t) - \widehat{V}(x_t)\right) + \frac{1}{2}\text{huber}\left(A_\phi(x_t, a_t) - \widehat{A}(x_t, a_t)\right),$$

while the implemented Q-learning loss function is

$$L_{\text{QL}}(\theta, \phi) = \frac{1}{2}\text{huber}\left(Q_{\theta,\phi}(x_t, a_t) - \widehat{Q}^\pi(x_t, a_t)\right).$$

In light of this, the equivalence between VA-learning and behavior dueling no longer holds, creating a potentially bigger discrepancy in large-scale settings.

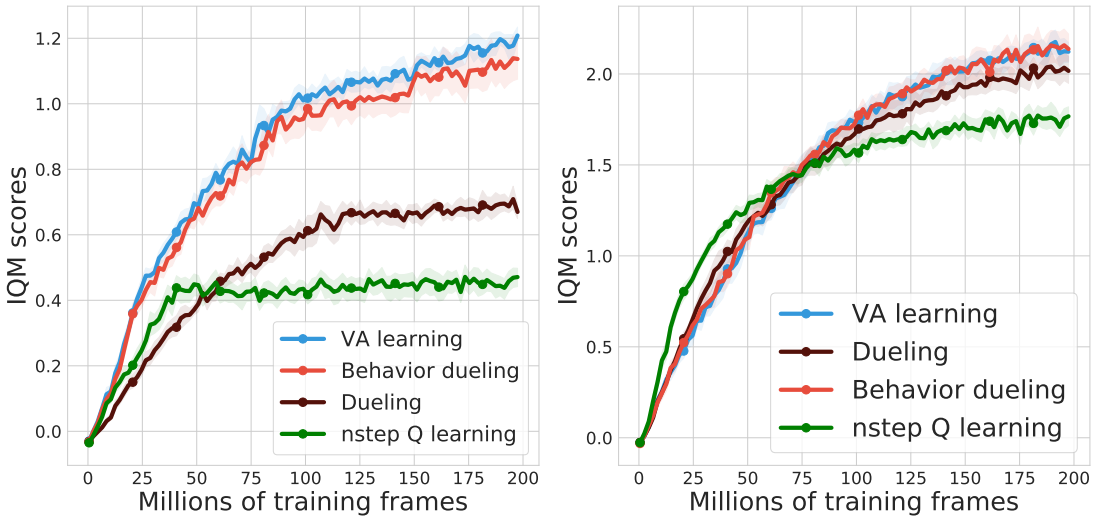
C.3. Deep RL experiments extra results

Robustness to off-policyness. We assess the robustness of various algorithmic variants to the level of off-policyness present in the replay. In DQN agents, the behavior policy μ is ε-greedy with the rate of exploration ε decays from 1 to ε_f over training. By default $\varepsilon_f = 0.01$. To increase the level of off-policyness overall in training, we set $\varepsilon_f = 0.5$. In Figure 6(a), we see that VA-learning and behavior dueling both achieve significant performance gains over dueling, whereas the latter improves upon baseline Q-learning. This shows that VA-learning and behavior dueling are more robust to changes in data distribution which deviates from the standard setting, and is hence more robust in general.

Results for restricted action set. In Figure 6(b), we compare the performance of different DQN agent variants in the standard Atari game setup. Compatible with observations made in prior work (Wang et al., 2015), the dueling architecture achieves significant improvement over the n-step Q-learning baseline. Although n-step Q-learning learns faster initially, other algorithmic variants catch up as the training progresses and obtains higher asymptotic performance.

VA-learning and behavior dueling achieve additional, albeit marginal, performance improvements over the dueling architecture. This is a sign that explicitly learning the behavior policy, rather than implicitly assuming it to be uniform, is potentially valuable. We carry out an ablation study that shows how VA-learning and behavior dueling are more robust than dueling and baseline Q-learning in a number of deep RL setups.

Per-game results. In Table 1, we show the per-game result for the full action Atari game setting. Compatible with Figure 4, the improvement of VA-learning and behavior dueling over dueling and n-step Q-learning is statistically significant in most cases.



(a) Effect of off-policyness (b) Restricted action set

Figure 6. (a) Comparing algorithmic variants implemented with the DQN architecture in the standard Atari setup. The behavior policy is ε-greedy to carry out exploration. The ε decays from 1 to ε_f. By default, $\varepsilon_f = 0.01$. Here, we set $\varepsilon_f = 0.5$ so that there is a large degree of off-policyness throughout training. VA-learning and behavior dueling achieves significant improvements compared to dueling and baseline Q-learning. (b) Comparing algorithmic variants implemented with the DQN architecture. The baseline agent is n-step Q-learning. We further compare with the dueling architecture (Wang et al., 2015), the behavior dueling and VA-learning. All agents are evaluated on Atari 57 games and IQM scores (Agarwal et al., 2021) are shown across 3 seeds. Behavior dueling and VA-learning obtain marginal advantage over dueling.

D. Connections between behavior dueling and VA-learning

We provide an in-depth discussion on the connection between behavior dueling and VA-learning in this section. Recall that in VA-learning, the value function V_θ and advantage function A_ϕ are updated by minimizing the squared losses

$$\frac{1}{2}\left(V_\theta(x_t) - \widehat{V}(x_t)\right)^2 + \frac{1}{2}\left(A_\phi(x_t, a_t) - \widehat{A}(x_t, a_t)\right)^2.$$

as shown in Eqn (6). In behavior dueling, the Q-function is parameterized as $Q_{\theta,\phi}(x, a) = V_\theta(x) + A_\phi(x, a) - A_\phi(x, \mu)$. The parameters are jointly updated with gradient descents on the loss function

$$L_{\mathrm{QL}}(\theta, \phi) = \frac{1}{2}\left(Q_{\theta,\phi}(x_t, a_t) - \widehat{Q}^\pi(x_t, a_t)\right)^2 \tag{11}$$

as in Eqn (9). The gradients with respect to θ and ϕ correspond to the updates for the value function and advantage function components of the Q-function. We examine the value gradient and advantage gradient in detail below. Our main findings are:

- Value gradients are equal in expectation for both VA-learning and behavior dueling, i.e.,

$$\mathbb{E}_\mu\left[\nabla_\theta L_{\mathrm{QL}}(\theta, \phi) \mid x_t\right] = \mathbb{E}_\mu\left[\nabla_\theta L_{\mathrm{VA}}(\theta, \phi) \mid x_t\right].$$

- There are subtle differences between advantage gradients differ for VA-learning and behavior dueling. Both updates bear the interpretation of fitting certain advantage function components.

D.1. Value gradient

We show that the value gradient of VA-learning and TD-learning with behavior dueling are equal in expectation. Similar conclusions apply for Q-learning.

Lemma 3. When the target network is the same as the online network $\theta^- = \theta, \phi^- = \phi$, then in expectation, the VA-learning value gradient is the same as the gradient of TD-learning with behavior dueling

$$\mathbb{E}_\mu \left[\nabla_\theta L_{\text{QL}}(\theta, \phi) \mid x_t\right] = \mathbb{E}_\mu \left[\nabla_\theta L_{\text{VA}}(\theta, \phi) \mid x_t\right],$$

where the expectation is over the action $a_t \sim \mu(\cdot|x_t)$.

Proof. For simplicity, all our derivations below assume $\theta^- = \theta, \phi^- = \phi$. We can write for behavior dueling

$$\nabla_\theta L_{\text{QL}}(\theta, \phi) = \Big(V_\theta(x_t) + A_\phi(x_t, a_t) - \widehat{Q}^\pi(x_t, a_t)\Big) \nabla_\theta V_\theta(x_t).$$

Now, taking expectation over the actions $a_t \sim \mu(\cdot|x_t)$ and note that $A_\phi(x_t, \mu) = 0$ due to the behavior dueling parameterization, we have

$$\mathbb{E}\left[L_{\text{QL}}(\theta, \phi) \mid x_t\right] = \mathbb{E}\left[\Big(V_\theta(x_t) - \widehat{Q}^\pi(x_t, a_t)\Big) \nabla_\theta V_\theta(x_t) \;\Big|\; x_t\right],$$

where $\widehat{Q}^\pi(x_t, a_t) = r_t + \gamma V(x_{t+1}) + \gamma A(x_{t+1}, \pi) - \gamma A(x_{t+1}, \mu)$. Examining the value gradient for the VA-learning case, we have

$$\nabla_\theta L_{\text{VA}}(\theta, \phi) = \Big(V_\theta(x_t) - \widehat{V}^\pi(x_t)\Big) \nabla_\theta V_\theta(x_t).$$

But note that $\widehat{V}^\pi(x_t) = r_t + \gamma V(x_{t+1}) + \gamma A(x_{t+1}, \pi) - \gamma A(x_{t+1}, \mu)$ by definition. This means

$$\mathbb{E}\left[\nabla_\theta L_{\text{VA}}(\theta, \phi) \mid x_t\right] = \mathbb{E}\left[\nabla_\theta L_{\text{QL}}(\theta, \phi) \mid x_t\right]$$

and hence the proof is concluded. □

The above equivalence implies that both VA-learning and behavior dueling carry out value updates that fit the value function targets $\widehat{V}^\pi(x_t)$.

D.2. Advantage gradient

For ease of presentation, we assume a tabular parameterization for the advantage function. For VA-learning, we consider the gradient $\nabla_{A(x,a)}$ for a fixed state-action pair (x, a). We can derive

$$\mathbb{E}\left[\nabla_{A(x,a)} L_{\text{VA}}(\theta, \phi) \;\middle|\; x_t = x\right] = \mu(a|x) \left(V(x) - \mathcal{T}^\pi Q(x, a)\right)$$

To obtain a better intuition for the above update, note that since $V(x)$ is meant to fit the average back-up target $\mathcal{T}^\pi Q(x, a)$, the difference $V(x) - \mathcal{T}^\pi Q(x, a)$ can be understood as the residual learning target. The multiplier $\mu(a|x)$ represents the magnitude of the update, thanks to the sampling behavior distribution.

On the other hand, for behavior dueling with Q-learning, recall that we use the parameterization $A(x, a) = f(x, a) - f(x, \mu)$ and we start by considering the gradient of $\nabla_{f(x,a)}$

$$\mathbb{E}\left[\nabla_{f(x,a)} L_{\text{QL}}(\theta, \phi) \;\middle|\; x_t = x\right] = \mu(a|x) \left[\left(Q(x, a) - \mathcal{T}^\pi Q(x, a)\right) - \sum_b \mu(b|x) \left(Q(x, b) - \mathcal{T}^\pi Q(x, b)\right)\right]$$

As before, the multiplier $\mu(a|x)$ is a result of the sampling distribution. The gradient can be understood as the difference between the TD error $\delta(x, a) = Q(x, a) - \mathcal{T}^\pi Q(x, a)$ and the average TD error $\sum_b \mu(b|x)\delta(x, a)$. Therefore, we can also understand the gradient to f as the residual learning target. In general, however, the advantage gradient update for VA-learning and behavior dueling differ.

D.3. Why behavior dueling is better than dueling

Following the discussion in the main paper, we consider a parameterization $A_\nu(x,a) = f(x,a) - f(x,\nu)$ with distribution ν. The notation A_ν is meant to emphasize that the advantage function depends on the distribution ν. Since $A(x,a)$ represents the unshared commonents of different Q-functions, we might be interested in minimizing such unshared information. Consider the squared norm as such an objective to minimize

$$\min_\nu \sum_a \mu(a|x) A_\nu(x,a)^2.$$

Lemma 4. Across all possible parameterizations with function f, the unique minimizer to the weighted squared norm is $\nu = \mu$. Formally,

$$\mu = \arg\min_\nu \max_f \sum_a \mu(a|x) A_\nu(x,a)^2$$

Proof. We can rewrite the squared norm objective as

$$\sum_a \mu(a|x) A_\nu(x,a)^2 = \mathbb{E}_{a\sim\mu(\cdot|x)}\left[\left(f(x,a) - f(x,\nu)\right)^2\right] \geq_{(a)} \mathbb{V}_{a\sim\mu(\cdot|x)}\left[f(x,a)\right].$$

The equality at (a) is achieved when $f(x,\nu) = \mathbb{E}_{a\sim\mu(\cdot|x)}\left[f(x,a)\right] = f(x,\mu)$. This for any fixed f, the minimizing distribution ν is such that $f(x,\nu) = f(x,\mu)$. For a specific f, the minimizing distribution ν might not be unique. However, it is straightforward to see that across all possible distributions $\nu = \mu$ is the unique minimizer. □

Since the behavior dueling at $\nu = \mu$ minimizes the weighted squared norm of the advantage function, it can be understood as minimizing the unshared information across actions and hence maximizing the shared components.

On tabular experiments, we validate such a theoretical insight in Figure 7. Across 20 randomly generated MDPs, we run Q-learning with behavior dueling vs. dueling. At iteration t, let $A(x,a) = f(x,a) - f(x,\nu)$ be the advantage function of the dueling algorithms ($\nu = u$ where u is uniform for dueling; and $\nu = \mu$ for behavior dueling). We measure three quantities over time: (1) for dueling, we compute $\left(f(x,a) - f(x,\nu)\right)^2$ (red); (2) for dueling, we also compute $\left(f(x,a) - f(x,\mu)\right)^2$ (blue) and (3) for behavior dueling, we compute $\left(f(x,a) - f(x,\mu)\right)^2$ (brown). All statistics are averaged over training samples, generated under the behavior policy. Comparing (2) and (3), we empirically verify that indeed, the behavior dueling parameterization obtains lower weighted advantage norm compared to the uniform dueling.

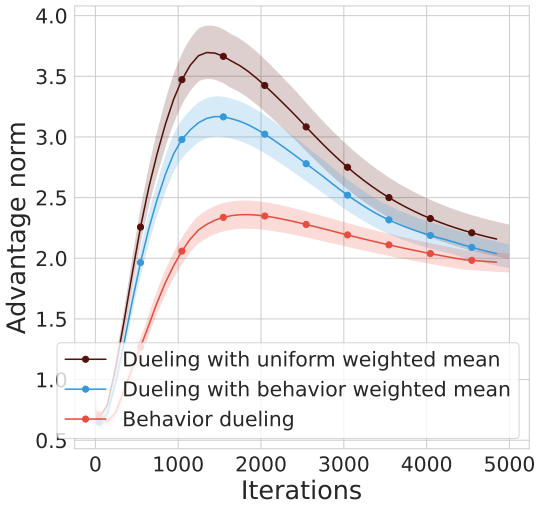


Figure 7. We compare the squared advantage norm over training iterations, across 20 randomly generated MDPs, between behavior dueling and uniform dueling. the behavior dueling parameterization indeed obtains a lower squared norm for the advantage function compared to uniform dueling, as suggested by the theoretical arguments above.

Table 1. Per-game result in the full action set setting. We report the mean $\pm$ standard error of scores averaged across the last $5M$ frames. For each game, we highlight the method with statistically highest mean scores (multiple methods are highlighted if their confidence internval overlap). Though VA-learning, behavior dueling and uniform dueling do not improve over n-step Q-learning in every game, the improvement is statistically significant in most cases. This is also compatible with the aggregate results shown in Figure 4.

Game	VA-learning	Behavior dueling	Dueling	n-step Q-learning
ALIEN	0.82 ± 0.10	0.50 ± 0.09	$\mathbf{1.00 \pm 0.04}$	1.47 ± 0.14
AMIDAR	$\mathbf{1.31 \pm 0.03}$	1.03 ± 0.11	$\mathbf{1.22 \pm 0.09}$	0.93 ± 0.04
ASSAULT	$\mathbf{5.61 \pm 0.28}$	$\mathbf{5.51 \pm 0.19}$	4.44 ± 0.08	4.50 ± 0.12
ASTERIX	2.11 ± 0.14	$\mathbf{2.42 \pm 0.12}$	1.52 ± 0.08	1.92 ± 0.05
ASTEROIDS	$\mathbf{0.12 \pm 0.01}$	$\mathbf{0.10 \pm 0.01}$	0.04 ± 0.00	0.03 ± 0.00
ATLANTIS	51.38 ± 0.37	$\mathbf{53.78 \pm 0.93}$	52.92 ± 1.29	49.79 ± 0.63
BANK HEIST	$\mathbf{1.66 \pm 0.10}$	1.43 ± 0.03	1.44 ± 0.03	1.24 ± 0.04
BATTLE ZONE	$\mathbf{1.36 \pm 0.07}$	1.15 ± 0.06	1.10 ± 0.02	1.18 ± 0.02
BEAM RIDER	$\mathbf{0.95 \pm 0.04}$	0.85 ± 0.03	0.78 ± 0.03	0.81 ± 0.02
BERZERK	0.54 ± 0.11	0.50 ± 0.10	$\mathbf{0.86 \pm 0.06}$	0.54 ± 0.01
BOWLING	0.18 ± 0.06	$\mathbf{0.29 \pm 0.01}$	0.10 ± 0.03	0.16 ± 0.06
BOXING	8.20 ± 0.01	$\mathbf{8.24 \pm 0.01}$	8.20 ± 0.01	8.11 ± 0.01
BREAKOUT	11.73 ± 0.16	10.94 ± 0.28	$\mathbf{12.41 \pm 0.10}$	$\mathbf{12.53 \pm 0.35}$
CENTIPEDE	0.19 ± 0.00	$\mathbf{0.22 \pm 0.01}$	0.12 ± 0.01	0.07 ± 0.02
CHOPPER COMMAND	$\mathbf{1.39 \pm 0.02}$	$\mathbf{1.43 \pm 0.08}$	1.05 ± 0.03	0.84 ± 0.02
CRAZY CLIMBER	4.96 ± 0.06	4.77 ± 0.09	4.98 ± 0.06	$\mathbf{5.27 \pm 0.06}$
DEFENDER	$\mathbf{3.52 \pm 0.10}$	$\mathbf{3.56 \pm 0.11}$	2.85 ± 0.05	1.90 ± 0.08
DEMON ATTACK	$\mathbf{47.18 \pm 2.93}$	$\mathbf{51.03 \pm 4.26}$	6.37 ± 0.21	24.23 ± 1.97
DOUBLE DUNK	$\mathbf{18.60 \pm 0.10}$	$\mathbf{18.50 \pm 0.15}$	17.96 ± 0.09	17.41 ± 0.30
ENDURO	1.87 ± 0.06	1.90 ± 0.07	$\mathbf{2.22 \pm 0.05}$	1.48 ± 0.06
FISHING DERBY	2.64 ± 0.05	$\mathbf{2.78 \pm 0.01}$	2.73 ± 0.03	2.62 ± 0.01
FREEWAY	1.10 ± 0.00	1.10 ± 0.00	1.11 ± 0.00	$\mathbf{1.13 \pm 0.00}$
FROSTBITE	0.49 ± 0.22	$\mathbf{1.08 \pm 0.09}$	$\mathbf{1.03 \pm 0.07}$	0.43 ± 0.19
GOPHER	6.02 ± 0.22	6.13 ± 0.21	5.89 ± 0.30	$\mathbf{9.76 \pm 0.53}$
GRAVITAR	$\mathbf{0.31 \pm 0.05}$	0.17 ± 0.00	0.14 ± 0.02	0.19 ± 0.03
HERO	$\mathbf{1.24 \pm 0.01}$	1.20 ± 0.00	0.58 ± 0.08	0.48 ± 0.03
ICE HOCKEY	1.35 ± 0.03	$\mathbf{1.42 \pm 0.02}$	0.92 ± 0.04	1.14 ± 0.04
JAMESBOND	$\mathbf{35.55 \pm 7.21}$	25.31 ± 6.93	11.81 ± 1.41	16.96 ± 1.17
KANGAROO	4.27 ± 0.12	$\mathbf{4.43 \pm 0.06}$	$\mathbf{4.49 \pm 0.03}$	3.77 ± 0.07
KRULL	7.88 ± 0.08	$\mathbf{8.61 \pm 0.23}$	8.30 ± 0.06	7.70 ± 0.20
MONTEZUMA REVENGE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
MS PACMAN	$\mathbf{0.64 \pm 0.01}$	0.53 ± 0.01	0.58 ± 0.04	$\mathbf{0.67 \pm 0.02}$
NAME THIS GAME	$\mathbf{1.81 \pm 0.04}$	$\mathbf{1.81 \pm 0.04}$	1.72 ± 0.01	1.32 ± 0.02
PHOENIX	$\mathbf{10.00 \pm 0.42}$	$\mathbf{10.39 \pm 0.73}$	4.02 ± 0.32	2.82 ± 0.15
PITFALL	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$
PONG	1.16 ± 0.00	1.15 ± 0.00	$\mathbf{1.17 \pm 0.00}$	$\mathbf{1.17 \pm 0.00}$
PRIVATE EYE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
QBERT	1.51 ± 0.04	1.42 ± 0.04	1.46 ± 0.04	$\mathbf{1.59 \pm 0.06}$
RIVERRAID	1.12 ± 0.03	1.17 ± 0.05	$\mathbf{1.20 \pm 0.05}$	1.18 ± 0.02
ROAD RUNNER	$\mathbf{7.81 \pm 0.07}$	$\mathbf{7.87 \pm 0.05}$	7.38 ± 0.03	7.34 ± 0.21
ROBOTANK	5.97 ± 0.13	6.02 ± 0.04	4.02 ± 0.24	$\mathbf{6.61 \pm 0.13}$
SEAQUEST	0.05 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	$\mathbf{0.43 \pm 0.04}$
SKIING	$\mathbf{0.73 \pm 0.02}$	0.66 ± 0.03	0.69 ± 0.00	-0.51 ± 0.11
SOLARIS	0.01 ± 0.00	-0.01 ± 0.01	0.07 ± 0.01	$\mathbf{0.09 \pm 0.02}$
SPACE INVADERS	$\mathbf{2.52 \pm 0.06}$	1.64 ± 0.02	$\mathbf{2.68 \pm 0.11}$	$\mathbf{2.84 \pm 0.37}$
STAR GUNNER	8.88 ± 0.71	$\mathbf{10.63 \pm 0.27}$	7.53 ± 0.10	7.15 ± 0.14
SURROUND	0.33 ± 0.05	0.65 ± 0.10	$\mathbf{0.72 \pm 0.04}$	0.43 ± 0.03
TENNIS	1.52 ± 0.01	$\mathbf{1.53 \pm 0.00}$	1.43 ± 0.05	1.19 ± 0.11
TIME PILOT	$\mathbf{11.65 \pm 0.36}$	9.07 ± 0.51	6.42 ± 0.22	6.00 ± 0.18
TUTANKHAM	$\mathbf{1.43 \pm 0.07}$	1.25 ± 0.06	1.33 ± 0.05	0.50 ± 0.05
UP N DOWN	6.40 ± 0.36	$\mathbf{7.62 \pm 0.28}$	6.32 ± 0.32	1.15 ± 0.05
VENTURE	0.00 ± 0.00	0.06 ± 0.03	0.16 ± 0.08	$\mathbf{0.79 \pm 0.05}$
VIDEO PINBALL	345.32 ± 27.56	194.08 ± 19.95	149.51 ± 48.98	$\mathbf{417.23 \pm 26.12}$
WIZARD OF WOR	$\mathbf{4.33 \pm 0.34}$	$\mathbf{3.99 \pm 0.10}$	2.70 ± 0.09	0.75 ± 0.02
YARS REVENGE	0.80 ± 0.02	0.57 ± 0.21	1.00 ± 0.02	$\mathbf{1.11 \pm 0.02}$
ZAXXON	2.92 ± 0.06	$\mathbf{3.16 \pm 0.09}$	2.57 ± 0.13	1.43 ± 0.05