Characterizing the Action-Generalization Gap in Deep Q-Learning

Zhiyuan Zhou
Department of Computer Science
Brown University
Providence, RI 02912
zhouzy@brown.edu

Cameron Allen
Department of Computer Science
Brown University
Providence, RI 02912
csal@brown.edu

Kavosh Asadi
Amazon Web Services
Santa Clara, CA 95054
kavasadi@amazon.com

George Konidaris
Department of Computer Science
Brown University
Providence, RI 02912
gdk@cs.brown.edu

Abstract

We study the action generalization ability of deep Q-learning in discrete action spaces. Generalization is crucial for efficient reinforcement learning (RL) because it allows agents to use knowledge learned from past experiences on new tasks. But while function approximation provides deep RL agents with a natural way to generalize over state inputs, the same generalization mechanism does not apply to discrete action outputs. And yet, surprisingly, our experiments indicate that Deep Q-Networks (DQN), which use exactly this type of function approximator, are still able to achieve modest action generalization. Our main contribution is twofold: first, we propose a method of evaluating action generalization using expert knowledge of action similarity, and empirically confirm that action generalization leads to faster learning; second, we characterize the action-generalization gap (the difference in learning performance between DQN and the expert) in different domains. We find that DQN can indeed generalize over actions in several simple domains, but that its ability to do so decreases as the action space grows larger.

Keywords: Action Generalization, Deep Reinforcement Learning, Action Abstraction, Deep Q Learning

Acknowledgements

This research was supported by a Brown University Undergraduate Teaching and Research Award, as well as by the ONR under the PERISCOPE MURI Contract N00014-17-1-2699, and by the NSF under grant 1955361.

Introduction

In reinforcement learning, generalization (Ponsen et al., 2009) is crucial for achieving efficient learning and leads to better performance over unseen data. Generalization allows agents to extrapolate from environment data they have observed and to adapt to unseen situations. It is well known that deep learning supports generalization (Kawaguchi et al., 2017), and deep reinforcement learning agents, which use deep learning to maximize reward, can therefore be expected to generalize as well. However, this kind of generalization is assumed to come from parameter sharing in the function approximator, and thus only provides a natural way of generalizing over inputs.

In discrete-action domains, deep reinforcement learning (DRL) agents typically only use states as inputs, because incorporating actions as inputs becomes computationally expensive as the size of the action space grows. Therefore, such agents, typified by DQN (Mnih et al., 2015), cannot rely on the same parameter sharing mechanism for generalizing over actions. And yet, many of the domains on which DQN has been shown to perform well have action spaces where generalization is not only possible, but essential. In particular, many of the Atari 2600 games (Machado et al., 2018) include multiple actions with identical effects or actions (e.g. RIGHTFIRE) that combine the effects of two or more other actions (e.g. RIGHT and FIRE). DQN's ability to cope with such action spaces is therefore surprising, given that it is unclear exactly how it is generalizing over these kinds of "similar" actions.

In this work, we seek to better understand DQN's (arguably counter-intuitive) ability to generalize over actions. We first empirically confirm the widely-held belief that action generalization leads to faster learning. We introduce an oracle for characterizing "perfect" action generalization with DQN using expert knowledge: when it is known which actions lead to the same transition effects, all action values for that subset are updated in one Q-update. Our experiments indicate that such generalization indeed improves learning speed. Next, we study action generalization in unmodified DQN by measuring its learning performance against the oracle; the difference is what we term the *action-generalization gap*. We experiment on classic Gym control environments (Brockman et al., 2016) and Atari 2600 games, and find that DQN's ability to generalize over actions depends on the size of the action space. In small action spaces, DQN performs nearly as well as the expert; however, when the action space gets large, DQN performs poorly because of its inability to generalize.

Expert Action Generalization

One way of achieving action generalization is through action abstraction. In the context of reinforcement learning, abstraction is a method to map the representation of the original problem to a new, simpler representation where irrelevant properties are filtered and only properties relevant to decision-making are kept (Abel, 2020; Ponsen et al., 2009). Action abstraction, in particular, is the technique of grouping similar actions together into one abstract action, ignoring their small differences that are not relevant to decision-making. It facilitates generalization by abstracting similar experiences, and allowing information about one experience to apply to related (unseen) experiences.

We introduce an oracle for characterizing perfect action generalization that uses expert knowledge to abstract over actions that are similar to each other. More precisely, in a Markov Decision Process with state space S, action space A, reward function R, and discount factor $\quad$, the expert knowledge is a symmetric $jAj \quad jAj$ similarity matrix K, where each index $K(i; j)$ is a value between 0 and 1 indicating the similarity score of actions a_i and a_j. A score of 1 means two actions are fully similar, and 0 means fully different. Given this expert knowledge, we can adjust the Q-update process during Q-learning: for an experience tuple $(s; a; r; s')$, we update

$$Q(s; a) = Q(s; a) + \quad K(a; a) \quad [(r + \quad V(s')) \quad Q(s; a)] \quad 8\, a \, 2 \, A:$$

In words, during each update step, we not only update $Q(s; a)$ from the experienced action a, but also $Q(s; a)$, proportionally to how similar a and a are. So, a fully similar action a will get a full Q update, and a fully dissimilar action a will not be updated at all, and those in between will be updated proportionally. This process allows generalization to a despite only experiencing a in the environment.

This oracle provides a way of measuring the degree of action generalization through learning performance. To see how much action generalization DQN can do, we can simply compare it with the oracle, which represents a best-case performance ceiling for action generalization methods. Since it is hard to quantify action generalization directly, we compare learning performance instead, using it as a proxy for generalization. We define this performance difference between the oracle and a DRL agent to be the *action-generalization gap*. The rest of this paper aims to characterize the action-generalization gap for DQN.

Action Space Augmentation for Evaluation

In order to evaluate action generalization, we need environments with action spaces where actions may be similar to each other, so that it makes sense for actions to generalize. To that end, we propose to augment the action space of a base environment to include additional similar actions. We propose three such "action augmentation" methods:

1. *Duplicate actions (N)*: augment the original action set N 1 times, so the new action space contains N copies of every action in the original action space.
2. *Semi-duplicate actions*: augment the original action set with 4 sets of reduced-magnitude actions, jAj in each set (where jAj is the size of the original action space), for a total of 5 jAj actions. Each set of these semi-duplicate actions has similar transition effects to the original action set, differing only in the magnitude. The magnitude similarity is controlled by a similarity score h 2 $[0; 1]$, where $h = 1$ corresponds to the same-magnitude action and $h = 0$ corresponds to a zero-magnitude action (No-op). For example, in the Pendulum environment, we can apply a 0:8 torque to the left, or 0:5 torque to the right, etc.
3. *Random actions*: augment the original action set with 4 jAj stochastic actions for a total of 5 jAj actions. Each stochastic action is a uniform random distribution over the actions in the original action space.

Here we are concerned with discrete action spaces, so we take discrete-action versions of CartPole, Pendulum, and LunarLander as the base environments for our experiments, and apply the three action augmentation methods to expand the action space.

Because the action-augmentations above are artificially created, we can supply the oracle with knowledge of which similar actions should be abstracted together. For the "duplicate actions" augmentation, all duplicate copies of each actions are fully similar ($K(i; j) = 1$) to each other, and not to anything else. For the "semi-duplicate actions" augmentation, $K(i; j) = h$ if a_i and a_j are semi-duplicates of each other, else $K(i; j) = 0$. For the "random actions" augmentation, all the random actions are fully similar to each other, and nothing else is similar. For all augmentations, $K(i; i) = 1$. The oracle uses this information to guide its Q updates.

Evaluating Performance

We now experimentally characterize the action-generalization gap between DQN and the oracle. In addition to the augmented action space, we provide the original action space as a baseline for comparison, which we call "baseline". For the "duplicate actions" augmentation in this section, we set $N = 5$. For the semi-duplicate augmentation, we use similarity scores h 2 $f0:2; 0:5; 0:8g$.

In the "duplicate actions" environments, having duplicate actions slows down learning, with the exception of Pendulum, where 5x duplicate actions performs about the same as the baseline (Figure 1). By contrast, the oracle is unaffected by the larger action space and performs just as well as the baseline. This confirms the hypothesis that action generalization does help speed up learning.

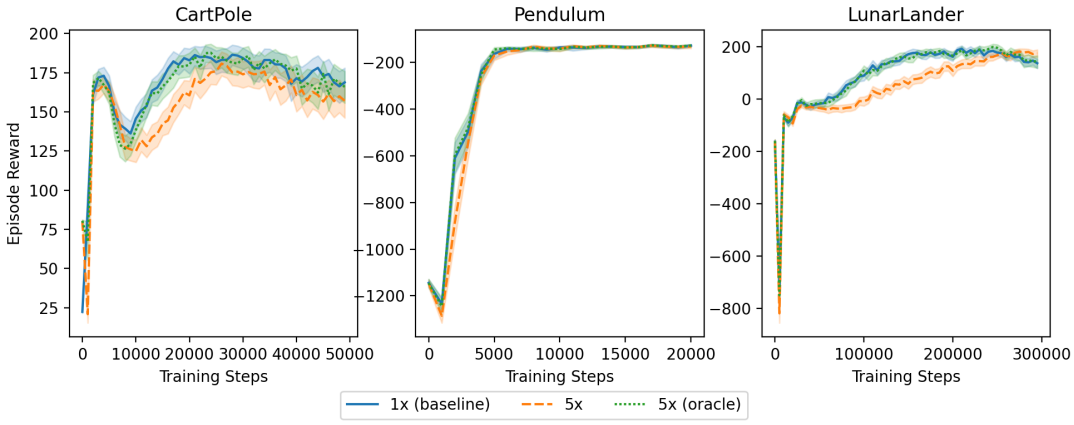

Figure 1: Action-generalization gap for DQN with **5x duplicate actions** on CartPole, Pendulum, and LunarLander

The experiment also provides evidence that DQN is doing some implicit action generalization. Notice the difference in performance between the oracle and unmodified DQN in Figure 1: DQN takes at most twice as long as the oracle to learn with 5x duplicate actions. This is a bit surprising given that the oracle performs the same as the unaugmented environment (baseline), whose action space is one-fifth as large. If DQN isn't doing any action generalization, then learning time should scale linearly with the size of the action space. The fact that it scales sub-linearly suggests that DQN is able to generalize over actions to some degree.

These 5x duplicate action results are somewhat surprising, because even though there are more actions to choose from, the probability of choosing the optimal action under a uniform random policy remains the same. So why does the learning problem become harder? We hypothesize two potential explanations. First, it could be a result of the overestimation

Figure 2: (Left) Action-generalization gap for DQN with **5x semi-duplicate actions**, for similarity score $h \in \{0.2, 0.5, 0.8\}$, on Pendulum. (Right) Action-generalization gap for DQN with **5x random actions** on CartPole, Pendulum, and LunarLander.

bias in Q-Learning (Thrun and Schwartz, 1993), which increases with the size of the action space. Second, it could be that a larger action space means greater opportunity for noise in neural network parameters to produce noisy gradient updates. We leave a full investigation of these hypotheses for future work.

For the semi-duplicate action augmentations, all the experiments take about twice as long to converge, even the ones using an oracle (Figure 2, left). This indicates that semi-duplicate actions form a harder learning problem than exactly-duplicated actions, as expected. But still, we can note here that there is basically no action-generalization gap. This is more evidence to suggest that DQN is able to generalize over actions in small action spaces.

The "random actions" (Figure 2, right) results tell a more complicated story: when random actions are introduced, the learning performance is worse than "baseline", regardless of whether the oracle is used. This shows that having random actions makes learning substantially harder, which is expected because this stochasticity is hard to account for during learning. What's surprising is that in Pendulum and LunarLander the action-generalization gap is actually negative: the oracle performs worse than unmodified DQN. We suspect this may be due to an incorrect assumption: the oracle treats all random actions as similar to each other. However, the random actions are not always similar; in fact, they frequently select from original actions that have completely opposite effects. Performing Q-updates with the assumption that all random actions are fully similar may at times lead the function approximator to incorrectly update shared internal parameters governing the Q-values of the *non-stochastic* actions. If we instead assume the random actions are fully dissimilar, we simply recover the baseline performance. So, in the case of random actions, "baseline" is a better performance ceiling because none of its "abstract" actions contains actions with opposite effects. Using this ceiling, the action-generalization gap is quite noticeable, indicating that DQN is not good at generalizing over stochastic actions.

Evaluation on Large Action Spaces

The experiments from the last section suggest that DQN has some robustness to action augmentation in domains with small action spaces. Here we investigate whether that robustness extends to environments with larger action spaces, and find that, for both Pendulum and Atari 2600 games, it does not.

For Pendulum, we again augment with duplicate actions, and increase N to discover the point at which learning performance starts to degrade (see Figure 3, left). When $N = 5$, we do not observe any action-generalization gap, but when $N \in \{15, 50\}$, the gap becomes quite noticeable. Moreover, when $N = 50$, learning is not just slow, but converges to worse final performance. This deterioration in DQN's performance is not due to the inherent hardness of the domain, because the oracle continues to match the performance of learning on the unaugmented environment. This suggests that the large action-generalization gap must come from DQN's inability to generalize over actions is large action spaces.

For Atari 2600, we chose six commonly-used games to evaluate the action-generalization gap: Beam Rider, Breakout, Pong, Ms Pacman, Qbert, and Space Invaders. Our experiments use three types of action augmentation:

1. *Full action set*: There are 18 legal actions that are shared across all Atari games. However, since some of the actions are not meaningful in certain games, the default action space of each game is usually pruned to only leave the meaningful action space (which we refer to as the "baseline" actions). In the full action set setting, we use all of the 18 legal actions in Atari games as the action space, which increases the size of the action space for all the games we investigate.
2. *Duplicate actions*: augment the baseline actions with N 1 sets of duplicate actions. Here we use $N = 5$.
3. *Noop actions*: augment the baseline actions with N $|A|$ noop actions. Here we use $N = 2$.