Symmetry Detection and Exploitation for Function Approximation in Deep RL

(Extended Abstract)

Anuj Mahajan
Conduent labs India
Bangalore, India
anujmahajan.iitd@gmail.com

Theja Tulabandhula
University of Illinois Chicago
Chicago, USA
tt@theja.org

ABSTRACT

With recent advances in the use of deep networks for complex reinforcement learning (RL) tasks which require large amounts of training data, ensuring sample efficiency has become an important problem. In this work we introduce a novel method to detect environment symmetries using reward trails observed during episodic experience. Next we provide a framework to incorporate the discovered symmetries for functional approximation to improve sample efficiency. Finally, we show that the use of potential based reward shaping is especially effective for our symmetry exploitation mechanism. Experiments on classical problems show that our method improves the learning performance significantly by utilizing symmetry information.

Keywords

Deep Reinforcement Learning; Functional Approximation; Symmetry; Representation Learning.

1. INTRODUCTION

In many RL scenarios, like training a rover to move on Martian surface, the cost of obtaining samples for learning can be high (in terms of robot's energy expenditure), and so sample efficiency is an important subproblem which deserves special attention. Very often the environment has intrinsic symmetries which can be leveraged by the agent to improve performance and learn more efficiently. Moreover, in many environments, the number of symmetry relations tend to increase with the dimensionality of the state-action space, e.g. for the simple case of grid world of dimension d there exist $O(d!2^d)$ fold symmetries. This can provide substantial gains in sample efficiency while learning as we would ideally need to consider only the equivalence classes formed under the induced symmetry relations. However, discovering these symmetries can be a challenging problem owing to noise in observations and complicated dynamics of the environment. With recent advances in deep reinforcement learning [5, 4], it has been demonstrated that a lot of seemingly complex tasks which pose challenges in the form of large state action spaces and difficulty of learning good representations [1, 8]; can be handled very well with the use of deep networks as functional approximators. Training these networks, however requires large amounts of data and learning their parameters necessitates coming up with careful update procedures and defining the right objectives(see Glorot et al [3]) and is a topic of study in its own right. Thus established methods for MDP abstraction and minimization can't be practically extended for use in function approximation for RL using deep networks which builds up the premise for this work. To the best of our knowledge, we are the first to motivate the use of symmetry in the context of deep reinforcement learning. In this paper we investigate methods for discovering state space symmetries and their inclusion as prior information via a suitable cost objective.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

2. PRELIMINARIES

Symmetries in MDP: We represent MDP as a tuple $M := \langle S, A, \Psi, T, R\rangle$, here $\Psi \subset S \times A$ is the set of admissible state-action pairs, $R : \Psi \times S \to \mathbb{R}$, $T : \Psi \times S \to [0, 1]$ are reward and transition functions respectively. The notion of symmetries in MDP can be rigorously treated using the concept of MDP *homomorphisms*[7]. MDP homomorphism h from $M = \langle S, A, \Psi, T, R\rangle$ to $M' = \langle S', A', \Psi', T', R'\rangle$ is defined as a surjection $h : \Psi \to \Psi'$, which is itself defined by a tuple of surjections $\langle f, \{g_s, s \in S\}\rangle$. In particular, $h((s, a)) := (f(s), g_s(a))$, with $f : S \to S'$ and $g_s : A_s \to A'_{f(s)}$, & satisfies following: Firstly it preserves the rewards (i.e., $R'(f(s), g_s(a), f(s')) = R(s, a, s')$) & secondly it commutes with dynamics of M (i.e., $T'(f(s), g_s(a), f(s')) = T(s, a, [s']_{B_{h|S}})$). Here we use the notation $[\cdot]_{B_{h|S}}$ to denote the *projection* of equivalence classes B that partition Ψ under the relation $h((s, a)) = (s', a')$ on to S. Symmetries $\chi : \Psi \to \Psi$ can then be formally defined as *automorphisms* on M with the underlying functions f, g_s being bijective. The homomorphism requirements for a symmetry reduce to:

$$T(f(s), g_s(a), f(s')) = T(s, a, s') \quad (1)$$

$$R(f(s), g_s(a), f(s')) = R(s, a, s') \quad (2)$$

The set of *equivalence classes* $C[(s, a)]$ of state action pairs formed under the relation $\chi((s, a)) = (s', a')$ partition Ψ and can thus be used to form a quotient MDP, represented as $M_Q = M/C$, which is smaller and can be efficiently solved. However in RL setting, we do not know the underlying system dynamics, hence we estimate $C[(s, a)]$ on the go using the agent's experience and use estimated equivalent classes

to drive identical updates for equivalent state-action pairs during the process of learning the optimal policy.

Reward Shaping: Reward shaping is a technique that augments the reward function R of a MDP $M = \langle S, A, \Psi, T, R\rangle$ by a shaping function $F : \Psi \times S \rightarrow \mathbb{R}$. Thus, the agent sees a modified reward $R'(s, a, s') = R(s, a, s') + F(s, a, s')$. Ng et al. [6] have shown that if the reward shaping function is potential based i.e., is of the form: $F(s, a, s') = \gamma\Theta(s') - \Theta(s) \ \ \forall s, a, s'$ for some $\Theta : S \rightarrow \mathbb{R}$, then the policy invariance property is guaranteed. Shaping helps distinguish these pairs by making the rewards sufficiently distinct and consequently preventing spurious similarity estimates.

3. METHODOLOGY

Symmetry Detection: Given an MDP $M = \langle S, A, \Psi, T, R\rangle$, we define set $\Pi_{sa,j} = \{(\sigma, N_\sigma)\}$ where σ is a sequence of rewards of length j and N_σ is the number of times it is seen starting with state s taking action a during the execution of policy π. We use the notation $|\Pi_{sa,j}| = \sum_{|\sigma|=j} N_\sigma$ and $\Pi_{sa,j} \cap \Pi_{s'a',j} = \{(\sigma, \min(N_\sigma, N'_\sigma))\}$. We define the notion of similarity between two state action pairs $\langle s, a\rangle$ and $\langle s'a'\rangle$ as follows:

$$\chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) = \frac{\sum_{j=l_0}^{i} |\Pi_{sa,j} \cap \Pi_{s'a',j}|}{(\sum_{j=l_0}^{i} |\Pi_{sa,j}| * \sum_{j=l_0}^{i} |\Pi_{s'a',j}|)^{1/2}} \quad (3)$$

To efficiently compute the similarities between all the state action pairs, we use an auxiliary structure called the reward history tree P, which stores the prefixes of reward sequences of length up to i for the state action pairs observed during policy execution and also maintains a list of state, action, occurrence frequency tuples $[\langle s, a, o\rangle]$. The similarities estimates can be computed by doing a breadth first traversal on P. We consider state pairs

$$\chi_{sym} := \{\langle s, a\rangle, \langle s'a'\rangle | \chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) \geq \Delta\}$$

as similar for the given length (i and l_0) and threshold parameters (Δ). Note that the definition of χ_{i,l_0} enables finding state action symmetries even when the actions are not invariant under the symmetry transform ie. $g_s(a) \neq a \forall s, a$ (indeed, this is the case with the Cart-Pole problem where $\forall s \in S,\ g_s(Left) = Right, g_s(Right) = Left$). Previous work in [2] is unable to do this. It can be shown that the similarity measure $\chi_{l_0,i}$ is complete ie. any state action pair which is equivalent under the given symmetry should be identifiable using 3.

Symmetry Inclusion Priors & Learning: Let $Q(s, a; \theta)$ be the function approximator network. Having found some symmetric state action pairs χ_{sym}, we next use this information while training the network.

$$L_{i,total}(\theta_i) = L_{i,TD}(\theta_i) + \lambda L_{i,Sym}(\theta_i), \quad (4)$$

Eq. 4 gives training loss, where λ is weighing parameter and the loss components are:

$$L_{i,TD}(\theta_i) = \mathbb{E}_{\mathbb{B}}[((r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})) - Q(s, a; \theta_i))^2] \quad (5)$$

$$L_{i,Sym}(\theta_i) = \mathbb{E}_{\chi_{sym}}[(Q(s', a'; \theta_i) - Q(s, a; \theta_i)^2] \quad (6)$$

Here $\mathbb{B}$ is the set of observed (s, a, r, s') tuples following a ϵ-greedy behavioral policy. Differentiating the total loss (4) with respect to the weights, we arrive at a combination of gradients coming from the two loss objectives:

$$\nabla_{\theta_i} L_{i,TD}(\theta_i) = \mathbb{E}_{\pi,s}[(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i))\nabla_{\theta_i} Q(s, a; \theta_i)] \quad (7)$$

Table 1: Iterations to convergence(rounded)

Setup	Naive	Girgin	Sym
9x9 Θ_1	136 ± 7	113 ± 5	$\mathbf{37 \pm 6}$
13x13 Θ_1	237 ± 6	166 ± 7	$\mathbf{46 \pm 6}$
9x9 Θ_2	174 ± 7	116 ± 6	$\mathbf{31 \pm 5}$
13x13 Θ_2	241 ± 5	187 ± 6	$\mathbf{38 \pm 6}$
5x5x5 Θ_2	253 ± 8	177 ± 6	$\mathbf{41 \pm 7}$
7x7x7 Θ_2	275 ± 9	229 ± 10	$\mathbf{54 \pm 8}$

$$\nabla_{\theta_i} L_{i,Sym}(\theta_i) = \mathbb{E}_{\pi,\chi_{sym}}[(Q(s', a'; \theta_i) - Q(s, a; \theta_{i-1}))\nabla_{\theta_i} Q(s', a'; \theta_i)] \quad (8)$$

Eq. 7 represents the familiar Q-learning gradient. Eq. 8 is defined so to prevent the network from destroying the knowledge gained from current episode. A symmetric version of the DQN algorithm is given below.

Algorithm 1 Sym DQN

```
Initialize: Memory 𝔻 ← {}, P ← {{root}, {}}
Initialize action-value function Q with random weights(θ)
for episode ≤ M do
    Initialize start state
    for t = 1 to T do
        With probability ϵ select action a_t
        Otherwise select a_t = argmax_a Q(s_t, a, θ)
        Execute action a_t and observe reward r_t state s_{t+1}
        Store transition (s_t; s_t; r_t; s_{t+1}) in 𝔻
        Sample random minibatch 𝔹 from 𝔻
        Find 𝔹_s the batch of symmetric pairs of 𝔹 from P
        Set targets for 𝔹 & 𝔹_s
        Perform gradient descent step as in eq 7,8
    Update P with the episode
```

4. EXPERIMENTS

Grid World: In the grid world domain we consider the problem of finding the shortest path to a goal in a nXn square grid world. The discount factor is set to be $\gamma = 0.9$ and exploration $\epsilon = 0.1$. The goal state(x_G, y_G) is chosen randomly at the start of each iteration. We test two kinds of reward shaping settings $\Theta_1(x, y) = (|x - x_G| + |y - y_G|)$, $\Theta_2(x, y) = (|x - x_G| + |y - y_G|)\gamma^{|x-x_G|+|y-y_G|}$. Table 1 gives the number of episodes required for convergence to optimal policy for different setups. The agent running our algorithm (labeled Sym) learns the optimal policy much faster than baseline (labeled $Naive$) and previous work. On performing the two sided Welsh's t-test on the null hypothesis the p-values are found to be $< 10^{-4}$ for all the setups.

5. CONCLUSIONS

We have proposed a novel framework for discovering environment symmetries and exploiting them for the paradigm of function approximation. The experiments verify the effectiveness of our approach. For future work we would like to explore methods for learning the symmetries using deep networks.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new

perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] S. Girgin, F. Polat, and R. Alhajj. State similarity based approach for improving performance in RL. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 817–822, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

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

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[6] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

[7] B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes. Technical report, Amherst, MA, USA, 2001.

[8] D. Williams and G. Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.