Towards Reinforcement Learning of Human Readable Policies

Riad Akrour[1], Davide Tateo[1], and Jan Peters[1,2]

[1] IAS, TU Darmstadt, Germany
{first name}@robot-learning.de
[2] Max Planck Institute for Intelligent Systems, Tübingen, Germany.

Abstract. Reinforcement learning (RL) has demonstrated its ability to solve high dimensional tasks by leveraging non-linear function approximators. These successes however are mostly confined to simulated domains. When deploying RL to the real world, several concerns regarding the use of a 'black-box' policy might be raised. In an effort to make RL more interpretable, we propose in this paper a policy iteration scheme that retains a complex function approximator for its internal value predictions but constrains the policy to have a simple, human readable structure. We show that our proposed algorithm can solve continuous action deep RL benchmarks and return policies that can be fully visualized and interpreted by a human non-expert.

Keywords: Hierarchical Reinforcement Learning · Interpretable Machine Learning

1 Introduction

Reinforcement Learning (RL) [16, 17] has led to several practical breakthroughs [10, 15] despite the high dimensionality of the state-action space of the problems at hand. To do so, recent RL algorithms leverage high dimensional function approximators for estimating the value function [10, 18] or both the value function and the policy [9, 7]. However, in applying RL to real-world problems, the policy returned by the learner might need to be scrutinized to ensure that it is safe, ethical and fair. To facilitate the latter we propose to replace the policy, typically a 'black-box' deep neural network, with a more interpretable structure.

We choose to use a simpler structure for the policy instead of the value function because the policy usually has a simpler functional shape [12, 4], and can thus be more easily approximated. Our proposed algorithm yields a clustering of the state space, where a single action is associated to each cluster. Such clustering of the state space could be seen as a form of state abstraction [6, 1, 8, 3]. Previous work in high-dimensional state abstraction however, only considered non-intepretable state-to-cluster mappings, in the form of polynomial functions [8] or neural networks [3]. Recently, [11] proposed to learn a policy linear in state for a medical treatment problem, where interpretability is critical. The

resulting treatment policy was easy to interpret in the bi-variate case but linear-in-state policies are not ideal for all settings. First, their interpretability in higher dimensional spaces can be questionable. Secondly, a linear policy class might severely limit the quality of the policy.

As a computer program, the policy returned by our algorithm can be seen as a sequence of `IF` blocks having the structure `IF close(state, center[k]) DO action[k]`; where `state` is the current state, `center[k]` is a cluster center and `action[k]` its associated action. To ensure that such a policy is interpretable, we impose a series of limitations that make the underlying optimization problem challenging. First, we limit beforehand the number of clusters to a fixed number K. Second, and most importantly, we do not allow the cluster centers to be optimized. Instead, we only allow the discrete decision of picking a cluster center out of the states encountered during learning. Doing so ensures that the cluster centers are within the potentially lower dimensional manifold that is the state space and hence, are interpretable. The final component that is critical for interpretability is the function `close` that discriminates if a state belongs to the specified cluster, i.e., the state is sufficiently close to the cluster prototype. In this paper, we assume that this function is known, but we later discuss ways of learning it that would preserve interpretability.

2 Learning the interpretable policy

We describe more formally in this section the policy structure and an associated learning algorithm. Let $Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{1} \gamma^t R(s_t,a_t) \mid s_0 = s, a_0 = a\right]$, where the expectation is w.r.t. states s_t and actions a_t for $t > 0$, be the Q function of policy π, $V_\pi(s) = \mathbb{E}_{a \sim \pi(.|s)}[Q_\pi(s,a)]$ be its value function and $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ its advantage function.

Policy structure. As described in the previous section, the policy samples an action by comparing the current state to a list of cluster centers. It can then for instance select the action associated with the closest cluster. This would yield a discrete optimization problem. However, learning a satisfactory policy in this setting is a hard problem. We relax our model, by considering instead a smoothed version of the previous problem with fuzzy memberships [19] to each cluster. Formally, the policy at state s is Gaussian distributed $\pi(a|s) = \mathcal{N}(a|K\varphi(s), \Sigma)$; where K is a matrix, stacking actions associated to all clusters, φ is the membership function and Σ a state-independent full covariance matrix. The k-th component of the membership function φ^k is proportional to $\varphi^k(s) \propto c_k \exp(-\|s - s_k\|_2^2)$; where c_k is a cluster weight and s_k a cluster center. The membership is normalized such that $\|\varphi(s)\|_1 = 1$.

A final component of the policy is the default action. When a state is far from all cluster centers, the policy becomes too sensitive to small changes of the state. This sensitivity introduces both numerical problems and hinders interpretability. We thus introduce a default action, `action[0]` that has unormalized membership $\varphi_0(s) \propto 1$ fixed to one, independently of the input state.

Policy evaluation. The algorithm for learning the interpretable policy is based on the approximate policy iteration framework [5, 13]. At each iteration we sample trajectories, evaluate the policy and update the cluster weights, the cluster actions and the exploration noise. The advantage function of the current policy is evaluated on the generated samples by learning a neural approximated value function, following standard procedures described in e.g. [14]. In addition to updating the policy parameters of the current clustering, the advantage function is also used as a heuristic for adding states to the cluster center list.

Policy update. The policy parameters—cluster weights, cluster actions and covariance—are updated by solving a constrained optimization problem. The objective of the problem is to maximize the expected advantage function. The constraints are a Kullback-Leibler (KL) divergence constraint between successive policies (akin to a step-size) and en entropy constraint (to sustain exploration). We refer the reader to Sec. 2.2 of [2] for a more expansive definition and justification of the policy update optimization problem. From [2] we also borrow the optimization scheme for optimizing the cluster actions and covariance. Namely, we make use of the projection defined in Alg. 2 to transform any linear-Gaussian distribution into a linear-Gaussian distribution complying with the two policy update constraints; and then optimize the composition of the objective and the projection using gradient ascent.

For a fixed clustering, the main difference between the policy introduced in this paper and that of [2] is the feature function of the state. In [2], state features are given by the hidden layers of a neural network while they are the cluster memberships in this paper. These features are affected by the cluster weights. To learn these weights without violating the KL divergence constraint, we derive a projection that takes as input any vector of cluster weights and returns cluster weights satisfying the constraint. Similar to [2], we phrase this as an interpolation between the input and the previous parameters, derive an upper bound of the KL divergence that is simple in the interpolation parameter and solve for the interpolation parameter.

Let $w(s)$ be the unnormalized membership, that we denote w for short, which is given by $w^k = c_k \exp(-\left\|s - s_k\right\|_2^2)$. Let w_q be the unnormalized membership of the data generating policy, i.e. computed with the old cluster weights, and ϕ_q its normalization. Let $\varphi_\eta \propto \eta w + (1-\eta) w_q$ and $m_s(\eta) = \|K_q(\varphi_\eta - \varphi_q)\|^2_{\Sigma_q}$. We use the bound in inq. (1), which is tighter when $\|w\|_1 \approx \|w_q\|_1$ and inq. (2), which is tighter when $\|w\|_1 \gg \|w_q\|_1$, to project any input cluster weight to a cluster weight complying with the KL constraint. We then use this projection to transform the constrained optimization problem into an unconstrained one. Proofs for both inequalities are deferred to the appendix.

$$m_s(\eta) \leq \eta^2 \max\left(\frac{\|w\|_1^2}{\|w_q\|_1^2}, 1\right) m_s(1), \tag{1}$$

$$m_s(\eta) \leq \eta \frac{\|w\|_1^2}{\|w_q\|_1^2 + 2\|w_q\|_1 (\|w\|_1 - \|w_q\|_1)} m_s(1). \tag{2}$$

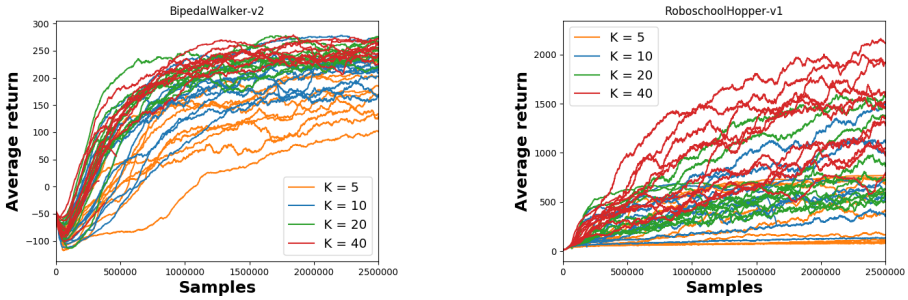

Fig. 1. Average return of the interpretable policies during learning for variable number of clusters on two continuous control tasks with 11 runs for each setting.

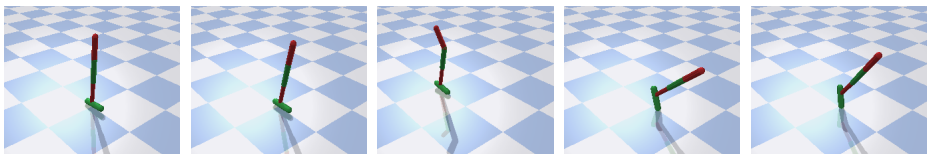

Fig. 2. Learned clusters on the hopping task with five cluster centers. Although not hopping as fast as a neural network, the policy still learns a successful hopping motion.

Adding and deleting cluster centers. The final element of our algorithm is to maintain the cluster center list. As stated before, the number of clusters is fixed beforehand to a hyper-parameter K. Starting from an empty cluster list, if at the beginning of the policy update, the length of the cluster list is less than K, we keep adding cluster centers until we reach K centers. The centers are selected by sorting the state-action pairs in the dataset by decreasing order of their advantage value. For instance, the first cluster added has a center and associated action set to the state-action pair with highest advantage. The rational behind this is that a high advantage value indicates that the action is surprisingly, w.r.t. the current policy, good in that state and the learner might want to repeat it more frequently when close to said state. The cluster weights for newly added clusters is initially set to zero such as not to change the policy, and hence the KL divergence.

As for deleting clusters, we periodically run a clean-up routine that computes the average membership of each cluster in the current dataset and deletes all clusters from lowest to highest membership as long as the KL divergence constraint is not violated. We perform this operation every 5 iterations in order to give time for the cluster weights of the newly added centers to be learned.

3 Experiments

We evaluate the proposed algorithm on two continuous control tasks, Roboschool-Hopper and BipedalWalker. The tasks respectively have a state space of 15 and

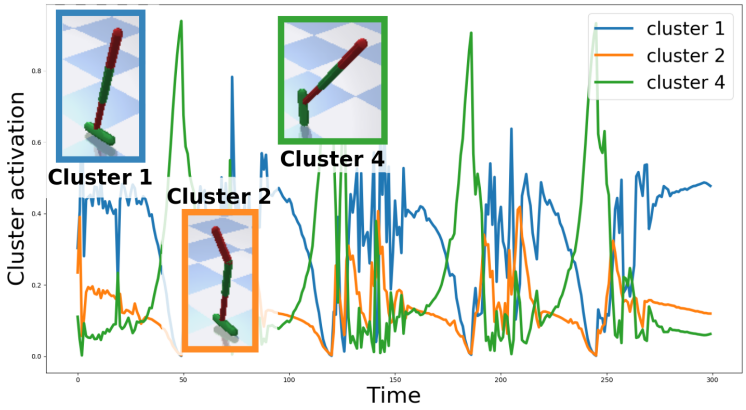

Fig. 3. Cluster activation across time during one rollout of the hopping policy. The cyclical nature of the task is clearly apparent in the cluster activation. The cluster centers show the different phases of the gait: stabilization, flexion and extension.

24 dimensions and an action space of 3 and 4 dimensions. For each task we set a variable cluster count ranging from 5 to 40 and launch 11 runs for each setting. Results reported in Fig. 1 show, perhaps unsurprisingly, that the higher the cluster count is the better performance gets. We note however that even though performance is sub-optimal on the BipedalWalker task, the algorithm always returns policies successfully walking with as low as 5 cluster centers. The RoboschoolHopper task appears more challenging and with 5 clusters the algorithm seem to only learn to stabilize itself. However, on rare runs it also learn successful hopping motions with only 5 cluster centers. An example of such policy is shown in Fig. 2. The cluster centers appear to be spread evenly along the hopping motion. Although the motion itself is not as fast as with a more complex policy class. Fig. 3 shows the membership of each cluster across time for one sampled rollout of the hopping policy. Only the 3 more active cluster centers from Fig. 2 are displayed. The cyclical nature of the task is clearly apparent in the figure. The figure also shows the different phases of the gait: stabilization upon landing, flexion of the leg and extension of the leg right before the hop.

4 Discussion

We have presented in this paper some preliminary results towards interpretable RL. Transitioning from a fully differentiable policy class to a class with discrete variables certainly presents a challenge. At a technical level, many questions remain open, such as training a policy with hard memberships to the clusters. The deletion mechanism of the cluster centers can also be largely improved since in the current state, the cluster list converges too early. One solution is to pose a joint optimization problem for deleting, adding and updating clusters all at once.

At a more general level, to increase interpretability of RL we proposed in this paper to split it into two sub-problems: i) learning a similarity function between states, and ii) leveraging the similarity function to learn the optimal policy. While we solely focused on the second sub-problem, solving the first sub-problem in an efficient and interpretable way can prove to be as challenging. One way to ensure interpretability of the similarity function is to force it to respect the underlying dynamics of the MDP, and the intuitive knowledge of which states are likely to come after which other states. To ensure efficiency and scalability to higher dimensional states such as images, the similarity function will need to build a more abstract and semantic understanding of the state space.

References

1. Abel, D., Hershkowitz, D.E., Littman, M.L.: Near optimal behavior via approximate state abstraction. In: International Conference on Machine Learning (ICML). pp. 2915–2923 (2016)
2. Akrour, R., Pajarinen, J., Peters, J., Neumann, G.: Projections for approximate policy iteration algorithms. In: International Conference on Machine Learning (ICML) (2019)
3. Akrour, R., Veiga, F., Peters, J., Neumann, G.: Regularizing reinforcement learning with state abstraction. In: International Conference on Intelligent Robots and Systems (IROS) (2018)
4. Anderson, C.W.: Approximating a policy can be easier than approximating a value function. Computer Science Technical Report (2000)
5. Bertsekas, D.P.: Approximate policy iteration: a survey and some new methods. Journal of Control Theory and Applications **9**(3), 310–335 (Aug 2011)
6. Li, L., Walsh, T.J., Littman, M.L.: Towards a unified theory of state abstraction for mdps. In: International Symposium on Artificial Intelligence and Mathematics (ISAIM) (2006)
7. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. CoRR (2015)
8. Mankowitz, D.J., Mann, T.A., Mannor, S.: Adaptive skills adaptive partitions (ASAP). In: Advances in Neural Information Processing Systems (NIPS). pp. 1588–1596 (2016)
9. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning (ICML) (2016)
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (02 2015)
11. Nie, X., Brunskill, E., Wager, S.: Learning When-to-Treat Policies. arXiv e-prints (2019)
12. Rexakis, I., Lagoudakis, M.G.: Classifier-based policy representation. In: Seventh International Conference on Machine Learning and Applications. pp. 91–98. IEEE (2008)

13. Scherrer, B.: Approximate policy iteration schemes: A comparison. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. pp. 1314–1322 (2014)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017)
15. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (Jan 2016)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Boston, MA (1998)
17. Szepesvari, C.: Algorithms for Reinforcement Learning. Morgan & Claypool (2010)
18. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
19. Zadeh, L.A.: Fuzzy sets. Information and control **8**(3), 338–353 (1965)

Appendix

We prove here the inequalities (1) and (2). They both start from the same reasoning, bounding the mean component of the KL divergence given w_η.

$$
\begin{aligned}
m(w_\eta) &= \left(M_\mu \frac{\eta w + (1-\eta)w_q}{\|\eta w + (1-\eta)w_q\|_1} - M_\mu \frac{w_q}{\|w_q\|_1}\right)^T {\Sigma_q}^{-1} \left(M_\mu \frac{\eta w + (1-\eta)w_q}{\|\eta w + (1-\eta)w_q\|_1} - M_\mu \frac{w_q}{\|w_q\|_1}\right) \\
&= \left(\frac{\eta w + (1-\eta)w_q}{\|\eta w + (1-\eta)w_q\|_1} - \frac{w_q}{\|w_q\|_1}\right)^T M_\mu^T {\Sigma_q}^{-1} M_\mu \left(\frac{\eta w + (1-\eta)w_q}{\|\eta w + (1-\eta)w_q\|_1} - \frac{w_q}{\|w_q\|_1}\right) \\
&= \left\| \frac{\eta w + (1-\eta)w_q}{\|\eta w + (1-\eta)w_q\|_1} - \frac{w_q}{\|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \left\| \frac{\eta w \|w_q\|_1 + (1-\eta)w_q\|w_q\|_1 - w_q\|\eta w + (1-\eta)w_q\|_1}{\|\eta w + (1-\eta)w_q\|_1 \|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \left\| \frac{\eta w \|w_q\|_1 + (1-\eta)w_q\|w_q\|_1 - \eta w_q\|w\|_1 - (1-\eta)w_q\|w_q\|_1}{\|\eta w + (1-\eta)w_q\|_1 \|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \left\| \frac{\eta w \|w_q\|_1 - \eta w_q\|w\|_1}{\|\eta w + (1-\eta)w_q\|_1 \|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \frac{\eta^2}{\|\eta w + (1-\eta)w_q\|_1^2} \left\| \frac{w\|w_q\|_1 - w_q\|w\|_1}{\|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \frac{\eta^2 \|w\|_1^2}{\|\eta w + (1-\eta)w_q\|_1^2} \left\| \frac{w}{\|w\|_1} - \frac{w_q}{\|w_q\|_1} \right\|^2_{M_\mu^T \Sigma_q^{-1} M_\mu} \\
&= \frac{\eta^2 \|w\|_1^2}{\|\eta w + (1-\eta)w_q\|_1^2} m(w). \qquad (3)
\end{aligned}
$$

Then, inequalities (1) and (2) differ in the way of bounding $\|\eta w + (1-\eta)w_q\|_1^2$. For the first inequality we have the following.

$$
m(w_\eta) \leq \begin{cases} \eta^2 \dfrac{\|w\|_1^2}{\|w_q\|_1^2} m(w) & \|w\|_1 \geq \|w_q\|_1, \\ \eta^2 m(w) & \text{otherwise.} \end{cases} \qquad (4)
$$

We can write the bound compactly as

$$
m(w_\eta) \leq \eta^2 \max\left(\frac{\|w\|_1^2}{\|w_q\|_1^2}, 1\right) m(w). \qquad (5)
$$

The second bound uses the convexity of $\|w_\eta\|_1^2$ in η. Let $f(\eta) = \|w_\eta\|_1^2$, then

$$
\begin{aligned}
f(\eta) &\geq f(0) + f'(0)\eta, \\
&= \|w_q\|_1^2 + 2\|w_q\|_1 (\|w\|_1 - \|w_q\|_1)\eta, \\
&\geq \eta(\|w_q\|_1^2 + 2\|w_q\|_1 (\|w\|_1 - \|w_q\|_1)).
\end{aligned}
$$

Giving rise to this other upper bound

$$m(p_\eta) \leq \eta \frac{\|w\|_1^2}{\|w_q\|_1^2 + 2\|w_q\|_1 (\|w\|_1 - \|w_q\|_1)} m(p).$$

If $\|w\| \approx \|w_q\|$ the first bound should be tighter. If $\|w\| \gg \|w_q\|$ the second one should be better. In practice we compute both and select the tightest. Note that this is only for a given state, but for the KL divergence we need to compute the expectation over all states in our dataset.