Quantile Regression Reinforcement Learning with State Aligned Vector Rewards

Oliver Richter & Roger Wattenhofer
Department of Electrical Engineering and Information Technology
ETH Zurich
Zurich, Switzerland
{richtero,wattenhofer}@ethz.ch

Abstract

The sample efficiency of deep reinforcement learning (DRL) algorithms is limited by the weak scalar training signal. We propose to use state aligned vector rewards to capitalize on the spatiotemporal nature of reaching problems and show that a state change distribution can be learned given an action distribution. Our agent, trained with a new DRL method inspired by quantile regression, is able to learn multiple times faster in high dimensional state spaces than a classical DRL algorithm.

1 Introduction

While neural networks are powerful function approximators, they require large amounts of training data to converge. In reinforcement learning (RL) this means many costly interactions with the environment. The problem is aggravated by the weak training signal of RL – a scalar reward – which was originally inspired by the general "rewarding" role of dopamine activity in mammal brains. However, Pinto and Lammel [18] point out that the diversity of dopamine circuits in the mid brain is better modeled by viral vector strategies. Gershman et al. [7] also show that human RL incorporates effector specific value estimations to cope with the high dimensional action space. Inspired by these new biological insights, we improve the sample efficiency of a deep RL algorithm by capitalizing on the spatiotemporal nature of reaching problems. A vector reward can in metric spaces easily be defined in dimensionwise alignment with the state space. We say that two vector spaces are *aligned* if their dimensions correlate and show that if state and action space are not aligned, a mapping from action distribution to state change distribution can be learned to allow vector reward training.

As a motivating example, consider the agent in Figure 1(a) trying to reach the goal (marked by the blue dot). If we take $\boldsymbol{s}$ as the position state vector of the agent relative to the goal, a sensible reward to guide the agent to the goal is $r = ||\boldsymbol{s}|| - ||\boldsymbol{s} + \boldsymbol{a}||$ where $||\cdot||$ can be any norm in the vector space of the environment. We will focus on the L^1 norm in this paper. During training the agent might try action $\boldsymbol{a}$ which moves it closer to the goal in x direction, but a bit further away from the goal in y direction. The scalar reward conveys the information that the action was rather positive (since the agent got closer to the goal) but misses out on the distinction that the action was good in x-direction but bad in y-direction. To give this distinction, a more informative reward would keep the dimensions separate and therefore be a vector itself: $\boldsymbol{r} = |\boldsymbol{s}| - |\boldsymbol{s} + \boldsymbol{a}|$ where $|\cdot|$ denotes the element-wise absolute value here. Note that this reward is dimensionwise aligned with the position $\boldsymbol{s}$, the *state*, of the agent. (Since we focus on reaching problems in this work, we will use the terms "position" and "state" interchangeably.) The problem with a state aligned vector reward is that the action space is in most cases not state aligned. To see this, consider the schematic robot arm in Figure 1(b): The action dimensions a_1 and a_2 correspond to the torques and do not directly translate to a shift in x and y dimension, respectively. To address this issue we use quantile function networks [5, 4] to

Workshop on Modeling and Decision-Making in the Spatiotemporal Domain, 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

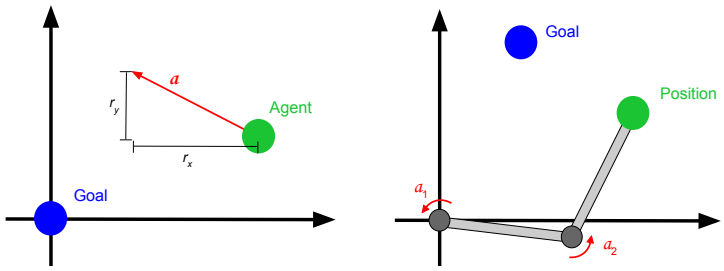

Figure 1: (a) An agent freely moving in a 2D world might try to reach a goal at position (0, 0) by taking action $\boldsymbol{a}$. A sensible reward in this environment is the change in absolute distance to the goal. With a scalar reward this would be summarized as $r = r_x - r_y$, whereas a vector reward would keep the two reward dimensions distinguishable. (b) In most cases action and state space are however not aligned, therefore a mapping from action to state change must be learned.

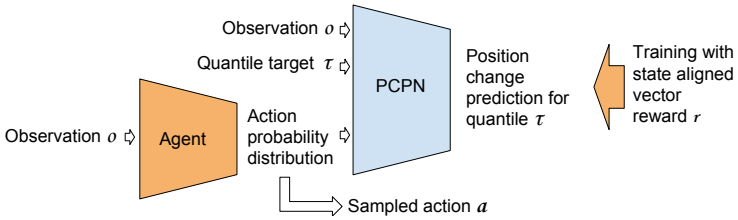

Figure 2: An overview of the architecture setup. The agent and the position change prediction network (PCPN) are instantiated with neural networks and trained through quantile regression reinforcement learning and quantile regression, respectively.

estimate the position change given the current observation and quantile target. Additionally, we give a parameterization of the action probability distribution as input to this position change prediction network (short PCPN). We then train the agent, parameterized by another neural network which maps from observations to action probability distributions, through a new RL method we call quantile regression reinforcement learning (short QRRL). A schematic overview can be seen in Figure 2.

2 Quantile Regression and Implicit Quantile Networks

Quantile regression [11] discusses approximation techniques for the inverse cumulative distribution function F_Z^{-1}, i.e., the *quantile function* $G_Z := F_Z^{-1}$, of some probability distribution Z. We refer to the quantile function as G_Z to avoid confusion with the in this work unrelated Q-function of Q-learning [14, 5, 4]. Recent work [4, 16] shows that a neural network can learn to approximate the quantile function by mapping a uniformly sampled quantile target $\tau \sim \mathcal{U}([0, 1])$ to its corresponding quantile function value $G_Z(\tau) \in \mathbb{R}$. Thereby the trained neural network implicitly models the full probability distribution Z. More formally, Ostrovski et al. [16] show that the expected quantile loss

$$\mathbb{E}_{\tau \sim \mathcal{U}([0,1])} \left[\mathbb{E}_{z \sim Z} \left[\rho_\tau(z - G_{\tilde{Z}_\theta}(\tau)) \right] \right] \text{ with } \rho_\tau(\delta) = (\tau - \mathbf{1}_{\delta<0}) \cdot \delta \tag{1}$$

of a parameterized quantile function $G_{\tilde{Z}_\theta}$ aproximating the quantile function G_Z of some distribution Z is equal to the *quantile divergence* $q(Z, \tilde{Z}_\theta)$ (see [16]) plus some constant not depending on the parameters θ. Here, $\tilde{Z}_\theta$ is the distribution implicitly defined by $G_{\tilde{Z}_\theta}$. Therefore, training a neural network $G_{\tilde{Z}_\theta}(\tau)$ to minimize $\rho_\tau(z - G_{\tilde{Z}_\theta}(\tau))$ with z sampled from the target probability distribution Z effectively models an approximate distribution $\tilde{Z}_\theta$ of Z implicitly in the network parameters θ of the neural network $G_{\tilde{Z}_\theta}(\tau)$. In our setup, we train the PCPN with the quantile regression loss (1) to approximate a position change quantile function per position dimension. Aside from a target quantile $\tau \sim \mathcal{U}([0, 1])$ per position dimension, the network input consists of an observation $\boldsymbol{o}$ of the agent and an action probability distribution parameterized by the output of the agent network.

3 Quantile Regression Reinforcement Learning and SAVER

Our new RL technique described in the beginning of this section is generally applicable to RL problems. We therefore use the terms "action" and "policy" in the common RL meaning here, which is distinct from the meaning of "action" and "policy" in the rest of the paper. In the end of this section we then link the position change estimation of the PCPN to this new meaning of "action".

The idea of QRRL is to model the policy implicitly in the network parameters, therefore allowing for complex stochastic policies. For this we model for each action dimension the quantile function $G_{\tilde{Z}_\theta}(\tau, \boldsymbol{o}) = F^{-1}_{\tilde{Z}_\theta}(\tau, \boldsymbol{o})$ of the implicitly defined action distribution $\tilde{Z}_\theta(\boldsymbol{o})$ by a neural network with an observation $\boldsymbol{o}$ and a target quantile $\tau \in [0, 1]$ as input. An action $\boldsymbol{a}_\tau \in \mathbb{A} \equiv \mathbb{R}^d$, d being the number of action dimensions, can then be sampled by sampling $\boldsymbol{\tau} \sim \mathcal{U}([0, 1]^d)$ and taking the network outputs as action. To train the networks quantile regression comes in handy. Informally put, quantile regression is linked to the Wasserstein metric [5] which is also sometimes referred to as *earth movers distance*. Imagine a pile of earth representing probability mass. In RL we want to move probability mass towards actions that were good and away from actions that were bad, where "good" and "bad" are measured by accumulative reward achieved. Quantile regression can achieve this neatly by shaping the pile of earth according to an advantage estimation and the constraint of monotonicity (a core property of quantile functions)[1]. This leads us to the QRRL actor objective

$$\min_\theta \mathbb{E}_{a_\tau}[\mathbb{E}_{\tau'}[\rho_{\tau'}(a_\tau - G_{\tilde{Z}_\theta}(\tau', \boldsymbol{o}_t)) \cdot (R_{t,n} - V_\psi(\boldsymbol{o}_t)]] \text{ s.t. } G_{\tilde{Z}_\theta} \text{ is monotonically increasing with } \tau'$$

where $(R_{t,n} - V_\psi(\boldsymbol{o}_t))$ is an advantage estimation [15] with $R_{t,n} = \gamma^n V_\psi(\boldsymbol{o}_{t+n}) + \sum_{t'=t}^{t+n} \gamma^{t'-t} r_{t'}$ being the n-step estimate of the discounted future reward and $V_\psi(\cdot)$ being the output of a critic network with parameters ψ, which is trained to approximate $R_{t,n}$. Note that QRRL is a constraint optimization. We found it however sufficient to add this constraint as an additional loss term

$$\mathcal{L}_{mon}(\tau, \tau', \theta) = \mathcal{L}_{Huber}(G_{\tilde{Z}_\theta}(\tau) - G_{\tilde{Z}_\theta}(\tau')) \cdot \mathbf{1}_{\tau<\tau' \,\&\, G_{\tilde{Z}_\theta}(\tau)>G_{\tilde{Z}_\theta}(\tau') \text{ or } \tau>\tau' \,\&\, G_{\tilde{Z}_\theta}(\tau)<G_{\tilde{Z}_\theta}(\tau')}$$

where we use the Huber loss $\mathcal{L}_{Huber}$ [9] for better convergence. We weight this additional loss term with a constant Lagrange multiplier λ_{mon} in the full loss term

$$\mathcal{L}_{QRRL}(a_\tau, \boldsymbol{o}_t, \tau, \tau', \theta, \psi) = \mathcal{L}_a(a_\tau, \boldsymbol{o}_t, \tau', \theta) + \lambda_c \mathcal{L}_c(\boldsymbol{o}_t, \psi) + \lambda_{mon} \mathcal{L}_{mon}(\tau, \tau', \theta) \qquad (2)$$

with $\mathcal{L}_a(a_\tau, \boldsymbol{o}_t, \tau', \theta) = \rho_{\tau'}(a_\tau - G_{\tilde{Z}_\theta}(\tau', \boldsymbol{o})) \cdot (R_{t,n} - V_\psi(\boldsymbol{o}_t))$ and $\mathcal{L}_c(\boldsymbol{o}_t, \psi) = (R_{t,n} - V_\psi(\boldsymbol{o}_t))^2$. λ_c is another constant Lagrange multiplier to weight the critic loss $\mathcal{L}_c$ against the actor loss $\mathcal{L}_a$.

In our State Aligned VEctor Reward (SAVER) agent, we use QRRL to train the agent network AN_η *through* the PCPN. For this we feed the action probability distribution output A of the agent network to a pretrained PCPN and train on the QRRL loss (2) with respect to the agent network parameters η, where we take the actual position change $\boldsymbol{\Delta p}$ introduced by a sampled action $\boldsymbol{a} \sim A$ as QRRL action target $\boldsymbol{a}_\tau = \boldsymbol{\Delta p}$ and compare it to K sampled position change estimations $\boldsymbol{\Delta \hat{p}}(\boldsymbol{\tau}^{(i)}), \boldsymbol{\tau}^{(i)} \sim \mathcal{U}([0, 1]^d)$ for $i \in \{1, ..., K\}$. Note that we pretrain the PCPN in our setup with random observations and action probability distributions and freeze the PCPN weights during agent training. Since the output of the PCPN can be seen as state aligned action, training on vector rewards is straight forward. Instead of having a scalar critic estimating the value function $V(\cdot) \in \mathbb{R}$ we estimate a value per action dimension, i.e., $\boldsymbol{V}(\cdot) \in \mathbb{R}^d$. Similarly, the vector rewards can be summed to a vector n-step discounted reward estimation $\boldsymbol{R}_{t,n} = \gamma^n \boldsymbol{V}(\boldsymbol{o}_{t+n}) + \sum_{t'=t}^{t+n} \gamma^{t'-t} \boldsymbol{r}_{t'}$, which leads us to a vector advantage $\boldsymbol{R}_{t,n} - \boldsymbol{V}(\boldsymbol{o}_t)$ at timestep t. Therefore we can apply loss (2) to each position change estimation dimension individually.

4 Experiments and Results

To test our ideas, we implement a hypercube environment to model high dimensional metric spaces. For proof of concept, we compare our SAVER agent against an A2C agent, the synchronous variant of A3C [15]. We implement a simple feed forward neural architecture architecture and fix all hyperparameters except the initial learning rate to values that work well for SAVER and A2C. The

[1] We can provide you with a more formal derivation, implementation details, more experiments and a link to our code on request.

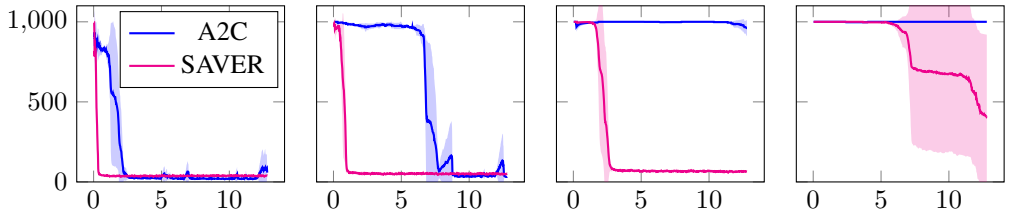

Figure 3: Average number of steps needed to reach the goal over the course of training of an agent that can select the dimension it wants to move in and the amount by how much it wants to move. Results are shown for a 3, 4, 5 and 6 dimensional environment, corresponding plots are ordered from left to right. The x-axis shows the number of training steps in millions while the y-axis shows the average episode length over the last 100 episodes. Training length was fixed to 12,800,000 steps. Plotted is the average of 3 training runs with the shaded area indicating the standard deviation between runs.

agent's action executed in the environment consists of two parts: a softmax distribution from which the dimension to be manipulated is chosen (discrete action part) and a scalar Gaussian distribution defined by network outputs μ and σ from which the step size is sampled (continuous action part). The agent's start position is random and the goal of the agent is to reach the origin of the d-dimensional space. We pretrain the PCPN with 10,000 batches of 128 transitions each by sampling softmax logits and μ uniformly at random from $\mathcal{U}([-1, 1])$ and σ from $\mathcal{U}([0, 1])$. We search for an appropriate learning rate in the 4 dimensional hypercube and settle for 0.01 for SAVER (chosen from $\{0.01, 0.003, 0.001\}$) and 0.0001 for A2C (chosen from $\{0.001, 0.0003, 0.0001\}$). The mean episode length – 100 episode mean averaged across 3 training runs – over the course of agent training is plotted in Figure 3. Note that the higher dimensional the hypercube is, the more advantageous it is to train with vector rewards.

5 Related Work

Klinkhammer [10] discusses multiple problem aligned rewards for better learning, while Brys et al. [3] discuss the effect of correlated rewards on learning performance. Brys et al. [3] also suggest multiple reward shapings of the same reward function for faster learning. Van Seijen et al. [20] decompose the reward function into multiple rewards and train an architecture similar to ours with deep Q learning, assigning a Q-value output to each reward. While all three approaches come to the same conclusion as we do, i.e., increased training performance, they do require hand engineered reward functions, reward shapings and/or reward decompositions. In contrast, our approach is based on the fact that many state spaces are metric spaces and therefore allow for a straightforward vector reward interpretation. This makes our approach easier to apply to a larger set of tasks. While state proximity was already used by McCallum [13] for faster backpropagation of rewards in tabular RL, we are unaware of any deep learning algorithm capitalizing on state aligned vector rewards as we do.

Many recent approaches to deep RL learn the environment dynamics to have a richer training signal [6], imagine action consequences [19], improve exploration [17] or dream up solutions [8]. An interesting line of research [21, 1, 12, 2] in this direction uses successor features to share knowledge between tasks in the same environment. In contrast to most of these works, which often only predict one possible next state or successor feature, our PCPN incorporates the full probability distribution of possible state changes. While Ha and Schmidhuber [8] also predict the full probability distribution of their next state representation by a Gaussian mixture model, our approach is more general in that it is also able to approximate non-Gaussian probability distributions.

6 Conclusion

In this work we present the idea of state aligned vector rewards for faster reinforcement learning. Additionally, we also present a new RL technique we call quantile regression reinforcement learning (QRRL). QRRL allows for complex stochastic policies in continuous action spaces, not limiting agents to Gaussian actions. Combining both, we show that our SAVER agent can be trained through a quantile network pretrained in the environment. Our SAVER agent is capable of training orders of magnitudes faster in high dimensional metric spaces.

References

[1] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado P. van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017. URL http://papers.nips.cc/paper/6994-successor-features-for-transfer-in-reinforcement-learning.

[2] André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel J. Mankowitz, Augustin Zídek, and Rémi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018. URL http://proceedings.mlr.press/v80/barreto18a.html.

[3] Tim Brys, Ann Nowé, Daniel Kudenko, and Matthew E. Taylor. Combining multiple correlated reward and shaping signals by measuring confidence. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1687–1693, 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8390.

[4] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1104–1113, 2018. URL http://proceedings.mlr.press/v80/dabney18a.html.

[5] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17184.

[6] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[7] Samuel J. Gershman, Bijan Pesaran, and Nathaniel D. Daw. Human reinforcement learning subdivides structured action spaces by learning effector-specific values. *Journal of Neuroscience*, 29(43):13524–13531, 2009. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.2469-09.2009. URL http://www.jneurosci.org/content/29/43/13524.

[8] D. Ha and J. Schmidhuber. World models. 2018. doi: 10.5281/zenodo.1207631. URL https://worldmodels.github.io.

[9] Peter J. Huber. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35(1):73–101, 03 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

[10] Eric R Klinkhammer. Learning in complex domains: Leveraging multiple rewards through alignment. 2018.

[11] Roger Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005. doi: 10.1017/CBO9780511754098.

[12] Chen Ma, Junfeng Wen, and Yoshua Bengio. Universal successor representations for transfer reinforcement learning. *CoRR*, abs/1804.03758, 2018. URL http://arxiv.org/abs/1804.03758.

[13] R. Andrew McCallum. Using transitional proximity for faster reinforcement learning. In Derek Sleeman and Peter Edwards, editors, *Machine Learning Proceedings 1992*, pages 316 – 321. Morgan Kaufmann, San Francisco (CA), 1992. ISBN 978-1-55860-247-2. doi: https://doi.org/10.1016/B978-1-55860-247-2.50045-0. URL http://www.sciencedirect.com/science/article/pii/B9781558602472500450.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

[15] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1928–1937, 2016. URL http://jmlr.org/proceedings/papers/v48/mniha16.html.

[16] Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive quantile networks for generative modeling. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3933–3942, 2018. URL http://proceedings.mlr.press/v80/ostrovski18a.html.

[17] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2778–2787, 2017. URL http://proceedings.mlr.press/v70/pathak17a.html.

[18] Daniel F. Cardozo Pinto and Stephan Lammel. Viral vector strategies for investigating midbrain dopamine circuits underlying motivated behaviors. *Pharmacology Biochemistry and Behavior*, 2017. ISSN 0091-3057. doi: https://doi.org/10.1016/j.pbb.2017.02.006. URL http://www.sciencedirect.com/science/article/pii/S0091305716303185.

[19] Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5694–5705, 2017. URL http://papers.nips.cc/paper/7152-imagination-augmented-agents-for-deep-reinforcement-learning.

[20] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5392–5402. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7123-hybrid-reward-architecture-for-reinforcement-learning.pdf.

[21] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017*, 2017. doi: 10.1109/IROS.2017.8206049. URL https://doi.org/10.1109/IROS.2017.8206049.