Diffusion-Based Approximate Value Functions

Martin Klissarov [1] **Doina Precup** [1]

Abstract

We present a novel model-based framework inspired by spectral graph theory and deep geometric learning: the Diffusion-based Approximate Value Function. Our approach efficiently approximates the graph Laplacian of an MDP's underlying graph by using Graph Convolutional Networks (GCN). By generating an approximate value function, we diffuse the reward signal much faster than traditional Reinforcement Learning algorithms such as TD(0). This leads to substantial improvements on sparse rewards environments where efficient credit assignment is most demanding.

1. Introduction

Model-free reinforcement learning agents have recently achieved impressive results by solving complex tasks such as the Atari games (Mnih et al., 2015) and simulated robotics tasks (Schulman et al., 2015). However, a main challenge still remains: the ability to perform well on sparse rewards environments. This comes on one hand from the need of better exploration, but also from the fact that back-ups, as performed by temporal difference updates, require an agent to visit the goal state multiple times for the information to spread back towards initial states.

A way to approach this challenge is through model-based reinforcement learning (Sutton, 1991; Silver et al., 2016b;a), which gives the agent the fundamental ability to plan. This ability is clearly desirable when an agent has limited access to the environment. It is also important when addressing the temporal credit assignment problem. Given a sparse rewards environment, this model can be used to intentionally train the agent in states that lie somewhere between the initial states and the rewarding states, propagating the information more efficiently by reducing the states' distance. The drawback comes when the agent has access to an imperfect model of the environment (which is usually the case) and generates unrealistic trajectories.

In this work we present a model-based approach inspired by spectral graph theory (Chung, 1997) to overcome the temporal credit assignment problem: the diffusion-based approximate value function. The basic intuition in our approach is that value functions can be seen as the outcome of diffusing the reward signal throughout state-space. As in the framework of Proto-Value Functions (Mahadevan, 2005; Mahadevan & Maggioni, 2007), we will approximate the transition matrix of an MDP by a diffusion model: the graph Laplacian. We will leverage recent advances in deep geometric learning (Bronstein et al., 2016) to implement a fast, flexible and efficient approximation to applying the graph Laplacian through Graph Convolutional Networks (Kipf & Welling, 2016; Defferrard et al., 2016).

We first present results in discrete domains in which we improve the speed of convergence by a factor of up to 20x by using diffusion-based approximate value functions. To verify that our method scales to more complex environments, we then perform experiments on two sparse rewards environments: SparseMountainCar-v0 and SparseHalfCheetah-v0. On these more complex tasks, our method shows once again significant improvements.

2. Background

2.1. Reinforcement Learning

A Markov Decision Process $\mathcal{M}$ is a tuple $\doteq(\mathcal{S}, \mathcal{A}, \gamma, r, P)$ with $\mathcal{S}$ the state set, $\mathcal{A}$ the action set and the scalar $\gamma \in [0, 1)$ the discount factor. The reward function maps states and actions to a scalar reward $r : S \times A \rightarrow Dist(\mathbb{R})$ and the transition matrix $P : S \times A \rightarrow Dist(S)$ specifies the environment's dynamics. A policy π is a set of probability distributions over actions conditioned on states π: $S \rightarrow Dist(A)$. For a given policy, the value function defines the expected return obtained by following π:

$$V_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t r(S_t, A_t) \middle| S_0 = s \right]$$

[1]Reasoning and Learning Lab/MILA, McGill, Montreal, Canada. Correspondence to: Martin Klissarov <mklissa@cs.mcgill.ca>, Doina Precup <dprecup@cs.mcgill.ca>.

Preliminary work. Under review by the ECA workshop at the International Conference on Machine Learning (ICML). Do not distribute.

V_π satisfies the following Bellman equations:

$$V_\pi(s) = \sum_a \pi(a|s) \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_\pi(s') \right)$$

The policy gradient theorem (Sutton et al., 1999) provides the gradient of a parametrized stochastic policy π_θ with respect to the expected discounted return from an initial state distribution $d_0 \in \text{dist}(S)$. For simplicity, we write the policy as π, making its parametrization (θ) implicit.

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_s d(s;\theta) \sum_a \frac{\partial \pi(a|s)}{\partial \theta} Q_\pi(s,a)$$

where $d(s;\theta) = \sum_{s_0} d(s_0) \sum_{t=0}^{\infty} \gamma^t P^\pi(S_t = s | S_0 = s_0)$ is a weighting of states along the trajectories generated by π and passing through s. Using the log-likelihood trick (Williams, 1992),

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial \log \pi(A_t|S_t)}{\partial \theta} A^\pi(S_t, A_t) \right]$$

where $A^\pi(S_t, A_t) = Q_\pi(S_t, A_t) - V_\pi(S_t)$ is the advantage function. The term $V_\pi(s_t)$ acts as a *baseline* (Williams, 1992; Sutton et al., 1999) which reduces the variance of the resulting estimator. This particular choice of policy gradient is called actor-critic (Barto et al., 1983), which is what all our agents use for training. To avoid modeling $Q_\pi(S_t, A_t)$ as well as $V_\pi(S_t)$, we make use of the identity: $Q_\pi(S_t, A_t) = \mathbb{E}[r(S_t, A_t) + \gamma V_\pi(S_{t+1})]$.

2.2. Diffusion Models and Proto-Value Functions

Spectral analysis of Markov Decision Processes has been explored in various ways (Mahadevan, 2005; Mahadevan & Maggioni, 2007; Osentoski & Mahadevan, 2007; Şimşek et al., 2005; Machado et al., 2017). We will focus on prior work based on Proto-Value Functions (PVFs), in which the graph Laplacian plays a central role. Given a reversible MDP[1], its transition matrix is diagonalizable in the following way:

$$P^\pi = \Phi \Lambda \Phi^T$$
$$P^\pi = \sum_{i=1}^{n} \lambda_i \phi_i \phi_i^T$$

where Λ is the diagonal matrix of eigenvalues and Φ is the matrix of eigenvectors. These eigenvectors form a complete orthogonal basis. The expected reward function when following policy π can be defined on the state space, therefore we can express it as: $R^\pi = \Phi\alpha$, where α is a vector of coefficients. The value function V^π can then be defined with respect to the eigenvectors ϕ_k:

$$\begin{aligned} V^\pi &= \sum_i^\infty \gamma^i (P^\pi)^i R^\pi \\ &= \sum_{i=0}^\infty \gamma^i (P^\pi)^i \Phi\alpha \\ &= \sum_{i=0}^\infty \sum_{k=1}^n \gamma^i (P^\pi)^i \phi_k \alpha_k \\ &= \sum_{i=0}^\infty \sum_{k=1}^n \gamma^i (\lambda_k)^i \phi_k \alpha_k \\ &= \sum_{k=1}^n \frac{1}{1-\gamma\lambda_k} \phi_k \alpha_k \\ &= \sum_{k=1}^n \beta_k \phi_k \end{aligned} \tag{1}$$

To obtain a compact approximation of the value function, it is possible to choose the top m largest values of β_k. Since the transition model is almost never available, a good surrogate candidate is the Laplacian as it is always diagonalizable and it produces the smoothest approximation to the value function by respecting the underlying graph topology. Proto-value functions make use of a particular form of the Laplacian, that is the *normalized graph Laplacian* $L = D^{\frac{-1}{2}}(D - A)D^{\frac{-1}{2}}$. As in equation (1), this results in the value function being approximated by a sum of basis functions $V^\pi = \beta_1 V_1^G + \beta_2 V_2^G + ... + \beta_n V_n^G$, where each basis function V_k^G is an eigenvector of the normalized graph Laplacian.

3. Difference Operators

An important shortcoming of the PVF framework is that it relies on the eigendecomposition of the underlying graph of an MDP. In large or continuous state-space this computation becomes intractable, although some approaches have been proposed (Mahadevan & Maggioni, 2007). To leverage the many interesting properties of the graph Laplacian, we propose to use it mainly as a *difference operator* (defined as L) on the space of functions on a graph (or equivalently the MDP) $\mathcal{F} : S \to \mathbb{R}$. Indeed, it is possible to verify that:

$$Lf(s) = \sum_{s'} \big(f(s) - f(s') \big)$$

where s' are the adjacent states of state s. Given sparse rewards environments leading to terminal goal states, it is straightforward to draw a connection between policy evaluation (Sutton & Barto, 1998) and the recurrent application of the difference operator on a value function:

[1] We make this assumption only to simplify the derivation.

$$v_{k+1}(s) = \big(R^{\pi}(s) + \gamma \sum_{s'} \mathcal{P}^{\pi}_{ss'} v_k(s')\big)$$

$$v_{k+1}(s) = v_k(s) + \alpha\bigg(R^{\pi}(s) + \gamma \sum_{s'} \mathcal{P}^{\pi}_{ss'} v_k(s') - v_k(s)\bigg)$$

$$V_{k+1}(s) = V_k + \alpha\bigg(\gamma \sum_{s'} \mathcal{P}^{\pi}_{ss'} V_k(s') - V_k(s)\bigg)$$

$$V_{k+1}(s) \approx V_k - \alpha L V_k(s) \tag{2}$$

$$\text{where } V_k(s) = \begin{cases} r(s,a) & \text{if } v_k(s) \text{ is terminal} \\ v_k(s) & \text{elsewhere} \end{cases}$$

The approximation made is similar to the one in the Proto-Value Function framework: we approximate the transition model by a diffusion model, in this case the graph Laplacian. An efficient implementation of equation (2) would require a matrix multiplication between an $|S| \times |S|$ matrix (where $|S|$ is the number of saved states) and a vector of length $|S|$. Such an operation implemented naively would result in a $O(n^3)$ complexity. To circumvent this problem, we will approximate applying the graph Laplacian through a recently proposed architecture in deep geometric learning: the Graph Convolutional Network ((Defferrard et al., 2016), (Kipf & Welling, 2016)). This architecture has been used for semi-supervised tasks such as node labeling. The loss function the network is minimizing is the following:

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg}, \text{ with } \mathcal{L}_{reg} = \sum_{i,j} A_{ij} ||f(X_j) - f(X_i)||^2$$

where $\mathcal{L}_0$ represents the supervised loss (the labeled nodes of the graph). The regularization loss, $\mathcal{L}_{reg}$, will propagate the label information smoothly through the graph as this loss is defined by A_{ij}, the adjacency matrix taken between node i and j. We notice that the regularization loss takes the same form as the graph Laplacian used as a difference operator. However, the authors argue that by conditioning the function f (the neural network) on the adjacency matrix, we would naturally be able to propagate the labels through gradient descent: we can therefore ignore the $\mathcal{L}_{reg}$ loss. The forward pass then takes (for a 2-layer Graph Convolutional Network) this exact form:

$$f(X, A) = \text{softmax}\big(\hat{A}\ \text{ReLU}\big(\hat{A} X W^{(0)}\big) W^{(1)}\big) \tag{3}$$

where $\hat{A}$ is a transformed adjacency matrix used to simplify the notation. The details of this derivation are available in (Kipf & Welling, 2016). The computation complexity of equation (3) becomes linear with the number of edges. We therefore propose using GCNs as diffusion-based approximate value functions to complement actor-critic algorithms (Barto et al., 1983). Let an agent store its past transitions from N episodes in a graph. As soon as it reaches the goal, it would use the stored transitions as input to the GCN, labeling the starting states as sources and the state preceding to the goal state as a sink (0 and 1 respectively). By repeatedly applying the forward/backward pass on this dataset, the GCN diffuses the reward signal throughout the MDP's underlying graph. We resume the process in Algorithm 1.

Algorithm 1 Diffusion-Based Approximate Value Functions for Actor-Critic agents.

```
Create empty graph G
for Episode=0,1,2,.... do
    for t=0,1,2...T do
        Add transition (s_{t-1}, s_t) to graph G
        if goal is reached then
            Label the starting states as 0
            Label the pre-goal states as 1
            Train GCN on dataset for M epochs
            Use final output to update the value function
        Proceed with actor-critic algorithm
    end
    if mod(Episode,N) then
        Reset G to empty graph
end
```

The next figure is a qualitative illustration of the result of such process in the domain FourRooms: it gives the agent an approximate value function to use as a critic. In this environment, the agent starts in the upper-left room while the goal is located in the bottom-right room. The upper-right room has not been explored yet by the agent (black color), therefore the GCN cannot diffuse the reward in that direction.

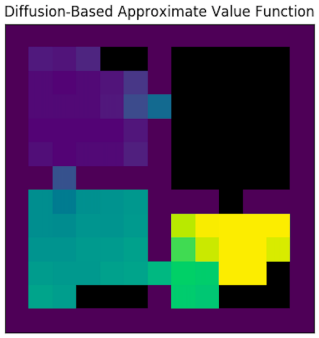

Figure 1. Diffusion-Based Approximate Value-Function on the FourRooms domain.

We can notice that most of the states in the bottom-right room (near the goal) have unrealistically high values. This is a consequence of using an approximate diffusion model to estimate the transition matrix as well as a consequence of limited exploration of the environment. To correct this error,

we continue training the agent using standard actor-critic, fine-tuning the diffusion-based approximate value function.

When we move to environments with continuous state space it is necessary to use function approximation for the critic. Similarly to discrete state environments, each visited state will be represented as a node in the underlying graph. However, the state-space being continuous implies that we will unlikely visit the same state twice. Therefore we will end up forming mostly path-graphs as inputs to the GCN, which aren't rich representations of an environment's dynamics. Thus, we propose to perform a radius-based nearest neighbor search on the stored states, adding edges between those that are relatively near. The value of the radius is a hyperparameter that has to be set and depends on the environment's state-space dimensionality. Indeed, high-dimensional space naturally leads to sparser data. Thus, we should expect a larger radius length in order to capture the same number of neighours as for a low-dimensional environment. To perform a nearest-neighbours search we also have to decide on a metric. As the space dimensionality grows, it is recommended to use L_k norms with k taking low values, such as the Manhattan metric or the fractional metric (Aggarwal et al., 2001; Hinneburg et al., 2000), as they behave better in these spaces. However, in our experiments we used the Euclidean metric as the environments we explored had limited dimensionality and the algorithm produced satisfying results.

In continuous state-space environments, we notice that diffusion-based value function can once again give accurate outputs when compared to fully trained value functions.

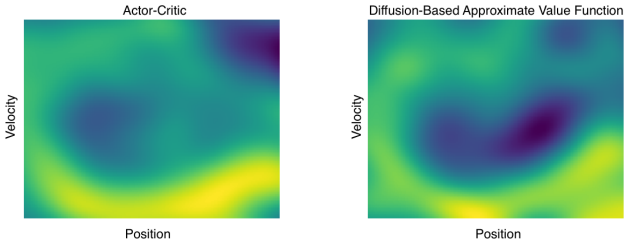

Figure 2. A comparison between the value functions obtained on SparseMountainCar-v0.

In figure 2, we compare a value function trained using actor-critic (left) to a diffusion-based approximate value function (right) obtained by repeatedly applying the difference operator, in our case the GCN, and fine-tuning the critic in a handful of episodes. This comparison is conducted on SparseMountainCar-v0 as it is possible to visualize the critic's output without dimensionality reduction. We notice that both value functions, even though obtained from different processes, have very similar outputs. This suggest that diffusion-based approximate value functions are able to accurately capture the environment's dynamics.

4. Empirical Results

4.1. Tabular case

In this section we present the results obtained on tabular domains. We will show that even in the tabular case, where credit assignment is less daunting, it is possible to obtain large improvements. We will explore the FourRooms domain with variable grid sizes (13x13, 22x22 and 29x29) and the more challenging Maze domain.[2] In both cases, the agent starts in one of the rooms and needs to get to the goal state, at which point the episode ends. In the figure 3, we plot the cumulative steps with respect to the number of episodes (i.e. the regret) for all configurations. We plot this variable instead of the cumulative rewards as it shows more clearly that diffusion-based approximate value functions reduce the exploration time and lead to more efficient credit assignment. As the number of states becomes larger, exploration becomes increasingly more difficult. More importantly, the backups performed by temporal differences algorithm require multiple visits to the goal in order for the reward signal to diffuse, therefore leading to misguided wandering. In the case of diffusion-based approximate value functions, we notice that after the first few visits to the goal, the algorithm converges quickly to a good solution. In Maze domain, our method leads to a 20x improvement in the number of steps needed for convergence.

4.2. Function Approximation

As mentioned in section 3, in the case of continuous state-space environments, we will perform an extra step before diffusing the reward with GCNs. We proceed to a radius-based nearest-neighbours search on the stored transitions in order to contribute more edges between very close states, therefore facilitating the diffusion of information. This procedure naturally assumes that the value function is locally smooth.

We perform experiments on SparseMountainCar-v0 and SparseHalfCheetah-v0 from OpenAI's Gym (Brockman et al., 2016). Both of these environments were previously used in experiments on sparse rewards environments (Houthooft et al., 2016; Plappert et al., 2017). In the original version of MountainCar-v0, a reward of -1 is given at every time-step until the goal is reached. Solving the original version is not a challenging task: by giving a reward of -1 we actually encourage the agent to move away from the initial state distribution, which points the agent towards the goal and simplifies the learning process. In

[2]The code for all experiments is available in the following repository: https://github.com/mklissa/davf

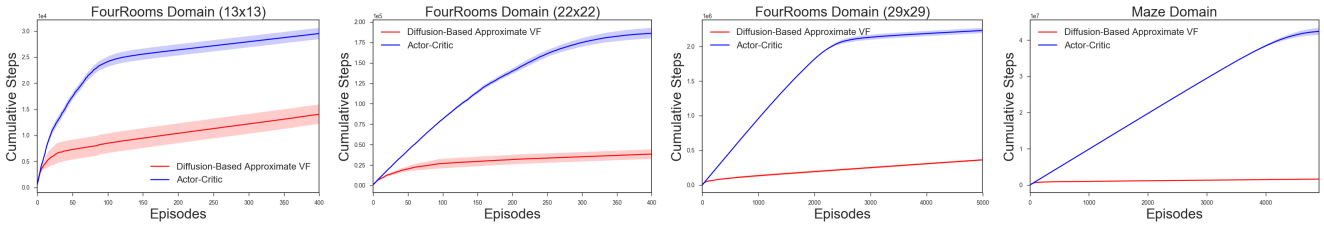

Figure 3. Cumulative steps for each of the configuration of the FourRooms domain (13x13, 22x22 and 29x29) and the Maze domain.

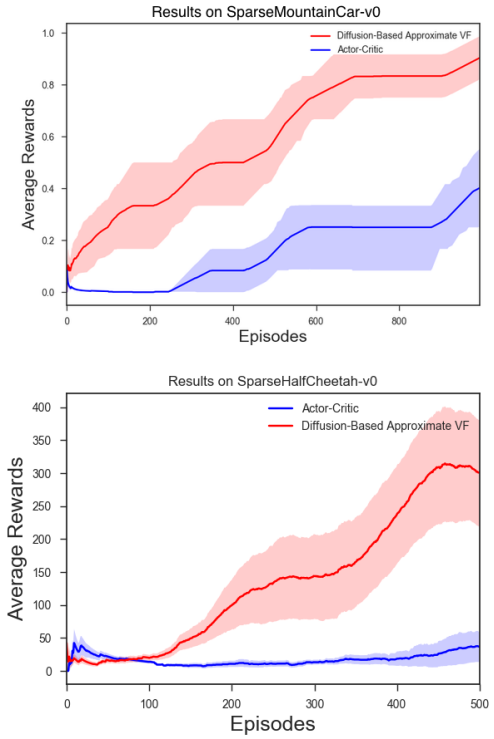

Figure 4. Cumulative rewards (number of episodes solved) for the sparse-rewards version of MountainCar-v0.

this modified version (SparseMountainCar-v0) we give a reward of 0 for every time-step and a reward of 1 when the goal is reached, which results in a truly sparse environment. In the case of SparseHalfCheetah-v0, no reward is given to the agent until it reaches a distance of 5 units from the starting position. Afterwards, a reward of 1 is given for each time-step spent beyond the 5 units mark. In figure 4, we plot the average rewards for both environments. In the case of SparseMountainCar-v0, we use standard actor-critic, while on SparseHalfCheetah-v0 we use the Proximal Policy Optimization algorithm (Schulman et al., 2017).

Without the additional signal provided by diffusion-based approximate value functions, agents will sometimes visit the goal, but will rarely remember the path leading to it. We notice that this behaviour is corrected on both domains, given that the agent visits the goal at least once. In SparseHalfCheetah-v0, despite the fact that no other reward than the one given for crossing a distance is available to the agent, it will still learn to move forward in a peculiar gait, reaching distances beyond the goal mark.[3]

5. Conclusion

We presented a model-based approach inspired by spectral graph theory and deep geometric learning (Chung, 1997; Bronstein et al., 2016; Kipf & Welling, 2016). Our work is closely related to the Proto-Value Function framework (Mahadevan & Maggioni, 2007) as it uses the graph Laplacian to approximate the transition matrix of a given MDP. By using the graph Laplacian as a difference operator, it is possible to diffuse the reward signal in a sparse rewards environment, which leads to significant improvements. In this work we have focused only on sparse rewards environments, however approximating the value function of any MDP is possible through a similar process.

We have performed experiments on environments with state-space dimensionality that were reasonable (up to 17D). However, high dimensionality can lead to many complications (Beyer et al., 1999; Houle et al., 2010), such as the fact that as the dimensionality tends to infinity, the distance between the farthest point and the nearest point reaches a ratio of 1. Therefore, for very high dimensional environments, a possible avenue would be to perform distance queries on the feature-space of a neural network's last layer. Another possibility would be to directly try to reduce the dimensionality through Variational Autoencoders (Kingma & Welling, 2013; Rezende et al., 2014).

We have focused on diffusing the reward signal through GCNs for sparse rewards environments, exploiting the environment's information in a more efficient way. However, by taking a deeper look into the Graph Convolutional Network architecture, we notice that they also take as input a feature vector for each node. In our implementation, we

[3] A video of the learning process is available at https://www.youtube.com/watch?v=Wl_MlIR5HOg .

used a one-hot encoding vector of length $|S|$ (the number of nodes), therefore leading to a diffusion process only guided by the adjacency matrix. However, using information extracted from the environment as features for each node could modify this diffusion process. In an setting where rarely visited states are of importance, their different feature representation could be exploited, possibly leading to better exploration. This is left as future work.

References

Aggarwal, Charu C., Hinneburg, Alexander, and Keim, Daniel A. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pp. 420–434, Berlin, Heidelberg, 2001. Springer-Verlag. ISBN 3-540-41456-8. URL http://dl.acm.org/citation.cfm?id=645504.656414.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, Sept 1983. ISSN 0018-9472. doi: 10.1109/TSMC.1983.6313077.

Beyer, Kevin S., Goldstein, Jonathan, Ramakrishnan, Raghu, and Shaft, Uri. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pp. 217–235, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-65452-6. URL http://dl.acm.org/citation.cfm?id=645503.656271.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Bronstein, Michael M., Bruna, Joan, LeCun, Yann, Szlam, Arthur, and Vandergheynst, Pierre. Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097, 2016. URL http://arxiv.org/abs/1611.08097.

Chung, Fan R. K. *Spectral Graph Theory*. American Mathematical Society, 1997.

Şimşek, Özgür, Wolfe, Alicia P., and Barto, Andrew G. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pp. 816–823, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102454. URL http://doi.acm.org/10.1145/1102351.1102454.

Defferrard, Michaël, Bresson, Xavier, and Vandergheynst, Pierre. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016. URL http://arxiv.org/abs/1606.09375.

Hinneburg, Alexander, Aggarwal, Charu C., and Keim, Daniel A. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pp. 506–515, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-715-3. URL http://dl.acm.org/citation.cfm?id=645926.671675.

Houle, Michael E., Kriegel, Hans-Peter, Kröger, Peer, Schubert, Erich, and Zimek, Arthur. Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management, 22nd International Conference, SSDBM 2010, Heidelberg, Germany, June 30 - July 2, 2010. Proceedings*, pp. 482–500, 2010. doi: 10.1007/978-3-642-13818-8_34. URL https://doi.org/10.1007/978-3-642-13818-8_34.

Houthooft, Rein, Chen, Xi, Duan, Yan, Schulman, John, Turck, Filip De, and Abbeel, Pieter. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *CoRR*, abs/1605.09674, 2016. URL http://arxiv.org/abs/1605.09674.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL http://arxiv.org/abs/1312.6114.

Kipf, Thomas N. and Welling, Max. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Machado, Marlos C., Bellemare, Marc G., and Bowling, Michael H. A laplacian framework for option discovery in reinforcement learning. *CoRR*, abs/1703.00956, 2017. URL http://arxiv.org/abs/1703.00956.

Mahadevan, Sridhar. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pp. 553–560, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102421. URL http://doi.acm.org/10.1145/1102351.1102421.

Mahadevan, Sridhar and Maggioni, Mauro. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007. URL http://dl.acm.org/citation.cfm?id=1314570.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Osentoski, Sarah and Mahadevan, Sridhar. Learning state-action basis functions for hierarchical mdps. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 705–712, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273585. URL http://doi.acm.org/10.1145/1273496.1273585.

Plappert, Matthias, Houthooft, Rein, Dhariwal, Prafulla, Sidor, Szymon, Chen, Richard Y., Chen, Xi, Asfour, Tamim, Abbeel, Pieter, and Andrychowicz, Marcin. Parameter space noise for exploration. *CoRR*, abs/1706.01905, 2017. URL http://arxiv.org/abs/1706.01905.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In Xing, Eric P. and Jebara, Tony (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/rezende14.html.

Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael I., and Abbeel, Pieter. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Silver, David, Huang, Aja, Maddison, Christopher J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, and Hassabis, Demis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016a. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Silver, David, van Hasselt, Hado, Hessel, Matteo, Schaul, Tom, Guez, Arthur, Harley, Tim, Dulac-Arnold, Gabriel, Reichert, David P., Rabinowitz, Neil C., Barreto, André, and Degris, Thomas. The predictron: End-to-end learning and planning. *CoRR*, abs/1612.08810, 2016b. URL http://arxiv.org/abs/1612.08810.

Sutton, Richard S. Dyna, an integrated architecture for learning, planning, and reacting, 1991.

Sutton, Richard S. and Barto, Andrew G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Sutton, Richard S., McAllester, David A., Singh, Satinder P., and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pp. 1057–1063, 1999.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.