GRAPH CONVOLUTIONAL NETWORKS AS REWARD SHAPING FUNCTIONS

Martin Klissarov & Doina Precup
Department of Computer Science
McGill University
{mklissa,dprecup}@cs.mcgill.ca

ABSTRACT

The framework of reward shaping (Ng et al., 1999) provides an approach for designing reward functions in such a way as to guarantee that the optimal policy remains unchanged while improving learning. However, finding such functions for complex environments automatically is a difficult problem. In this work, we propose a new framework for learning reward shaping functions by drawing from geometric deep learning. Our approach relies on Graph Convolutional Networks (Kipf & Welling, 2016) which we use to leverage the probabilistic inference view of reinforcement learning. More precisely, we use backwards messages from rewarding states, computed during inference and propagated through the environment's underlying graph with the help of a graph convolutional network. We then use these propagated messages for reward shaping in order to speed up the learning process. We demonstrate empirically that our approach can achieve substantial improvements in both tabular and high-dimensional control problems with sparse reward structure.

1 INTRODUCTION

Reinforcement learning algorithms provide a solution to the problem of learning a policy that optimizes an expected, cumulative function of rewards. Hence, a good reward function is critical to the practical success of such algorithms, but designing such a function can be challenging (Amodei et al., 2016). Approaches to this problem include intrinsic motivation (Oudeyer & Kaplan, 2007; Schmidhuber, 2010), optimal rewards (Singh et al., 2010) and reward-shaping (Ng et al., 1999). The latter provides an appealing formulation as it does not change the optimal policy of an MDP while speeding up the learning process. However, the design of potential functions used for reward shaping is still an open question.

In this paper, we propose a solution to this problem by approaching the reinforcement learning task by considering the Markov Decision Process' underlying graph. That is, each state is considered as a node and each transition an edge. By looking at RL as a graph-structured task, we can exploit recent advances in graph representation learning to provide a useful reward shaping function to the agent acting on the graph. To do so, we will represent the backward messages of the probabilistic inference view of reinforcement learning (Toussaint & Storkey, 2006; Rawlik et al., 2013; Levine, 2018). Backward messages are distributions that measure the probability of following an optimal trajectory from a given state, and can therefore provide a dense signal for the agent. We leverage this in the form of a potential-based function for reward shaping. We propose an efficient and flexible implementation for estimating the backward messages through Graph Convolutional Networks as a parametric approximator (Kipf & Welling, 2016).

2 BACKGROUND AND MOTIVATION

2.1 REWARD SHAPING

The framework of reward shaping provides a formal way to transform an MDP's reward function such that its optimal policy remains unchanged. Let the new reward function be: $R'(s, a, s') =$

$r(s,a) + F(s,a,s')$ where $F(s,a,s')$ is the shaping function, which can encode expert knowledge or represent concepts such as curiosity (Schmidhuber, 2010; Oudeyer & Kaplan, 2007). Ng et al. (1999) showed that a necessary and sufficient condition for preserving the MDP's optimal policy when using R' instead of r is for the shaping function to take the following form:

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s)$$

where $\Phi(s)$ is the scalar potential function $\Phi : S \to \mathbb{R}$. In (Ng et al., 1999), the potential function was defined as a distance to a goal position. Different alternatives have been explored since (Harutyunyan et al., 2015; Brys et al., 2015), but they either require a human in the loop or are not easily scalable to large problems. We instead aim to learn the potential functions in an end-to-end manner.

2.2 Reinforcement learning as probabilistic inference

Consider the graphical model in Fig.1, where O_t is a binary variable equal to 1 if the action A_t is optimal at S_t. The distribution over this optimality variable is defined with respect to the reward, as: $p(O_t = 1|S_t, A_t) = \exp(r(S_t, A_t))$ Maximizing likelihood of a state-action sequence being optimal in this model translates to optimizing the sum of rewards (Levine, 2018).

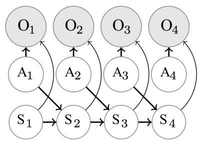


Figure 1: **Graphical model of the control task in reinforcement learning.** O_t is the optimality indicator, with $p(O_t = 1|S_t, A_t) = \exp(r(s_t, a_t))$, while S_t and A_t are the state and action latent variables.

The optimal policy can be computed through message passing, as in an HMM. Backward messages take the form: $\beta_t(S_t, A_t) = p(O_{t:T}|S_t, A_t)$ or, by marginalizing over actions: $\beta_t(S_t) = \int_{\mathbb{A}} p(O_{t:T}|S_t, A_t)p(A_t|S_t)dA = p(O_{t:T}|S_t)$. These messages intuitively represent the probability of following an optimal trajectory from state S_t onwards. Messages are propagated recursively through the graph as follows:

$$\beta_t(S_t, A_t) = p(O_t|S_t, A_t) \int_S p(S_{t+1}|S_t, A_t)\beta_{t+1}(S_{t+1})dS_{t+1} \tag{1}$$

Note that the first factor is dependent on the immediate reward, while the second represents the messages aggregated from all possible next states, with the "base case" message: $\beta_T(s_T, a_T) = p(O_T|s_T, a_T)$. In log-space, the backward messages become soft variants of $Q(S_t, A_t)$ and $V(S_t)$.

3 Method

We will now describe how to use backward messages as potential-based reward shaping functions, defined as: $F(s,a,s') = \gamma\beta(s') - \beta(s)$. We will estimate these messages by approximating Eq.1 through a Graph Convolutional Network (GCN) (Kipf & Welling, 2016; Defferrard et al., 2016).

3.1 Approximating backward messages

GCNs have mainly been used in semi-supervised learning for labeling nodes on a graph. The information of the labeled nodes is propagated through the graph, leading to a probability distribution defined over all nodes. The label propagation mechanism implemented by GCNs will be dependent on the connections between nodes, as well as on their features.

We will apply GCNs on a graph in which each state is a node and edges represent a possible transition based on any action. We define the probability distribution of the GCN as representing the probability of an optimal trajectory from a given state: $\Phi_{GCN}(s) \approx p(O_{t:T}|s) = \beta(s)$. GCNs use a softmax output over a number of classes, in our case there are only two: an optimal trajectory or not. To see how the output of the GCN can approximate the message on states obtained through inference, we need to look at the loss minimized by the network:

$$\mathcal{L} = \mathcal{L}_0 + \eta \mathcal{L}_{prop} \tag{2}$$

$\mathcal{L}_0$ represents the supervised loss for the labels, trained by cross-entropy. The L_2 propagation loss is given by $\mathcal{L}_{prop} = \sum_{i,j} A_{ij} ||\Phi_{GCN}(X_j) - \Phi_{GCN}(X_i)||^2$. The recursive update of Eq. 1 averages the neighboring messages through the transition dynamics $p(S_{t+1}|S_t, A_t)$ while the GCN's propagation loss in Eq. 2 combines them through the graph Laplacian. To see this, recall that the graph Laplacian operator is given by:

$$Lf(s_i) = \sum_j A_{ij} \big(f(s_i) - f(s_j)\big) \tag{3}$$

where f is a function on nodes and A_{ij} is the adjacency matrix entry for nodes i and j. In our case, $f(s)$ will correspond to backward messages $\beta(s)$. We implement the base case through the supervised loss $\mathcal{L}_0 = \sum_{\beta_i \sim \mathbb{B}} \beta_i log\big(\Phi_{GCN}(s_i)\big)$ where $\mathbb{B}$ is the set of base messages given by $\beta(s_i) = \sigma(r(s_i, a_i))$ for states with non-zero rewards.

3.2 Qualitative illustration

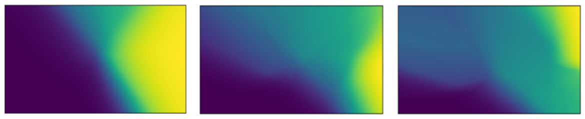

(a) $\eta = 0.0001$ (b) $\eta = 0.1$ (c) $\eta = 1.0$

Figure 2: Comparison of the output of GCN as the potential function for different values of η in Eq. 2, which controls the complexity of the output. The figures show the output after convergence on the PointMassMaze-v0 environment, depicted in Fig.4a. Lower values of η lead to less propagation and therefore a more simple, yet biased, output. As η increases, the output shows more complex structure, notably the position of the walls becomes more evident.

To get a better sense of how the GCN operates, we study the effect of the the hyperparameter η which mediates between $\mathcal{L}_0$ and $\mathcal{L}_{prop}$. Fig. 2 illustrates the result of varying η in the PointMassMaze-v0 environmen shown in Fig.4. In the original formulation, Kipf & Welling (2016) chose the value of η as zero as it is believed that the architectural bias of the GCN already implicitly implements this loss. In our experiments we have found that this hyperparameter can play an important role in the outcome. In Fig2a, we see that for a low value of η, the GCN outputs a simple solution where the states on the left have a low value (blue) while states on right have a higher value (yellow), which matches the position of the goal and the starting states. This solution has a greater bias than variance as it favors a simple output. As we increase the value of η, the propagation loss has greater importance and the output Φ_{GCN} shows more complexity, incorporating more information about the dynamics, such as walls and the specific position of the goal. There exists an interesting parallel with the λ parameter in the TD(λ) algorithm: λ controls the trade-off between bias and variance of the return estimation, while η controls the bias-variance trade-off of the diffusion process.

4 Experiments

4.1 Tabular case

We explore the classic FourRooms domain with different sizes (13x13, 22x22) and the more challenging Maze domain. The results are shown in Fig. 3, in the form of cumulative steps (i.e. regret). We compare our approach, denoted A2C + Φ_{GCN}, to a regular actor-critic algorithm using TD(λ) returns for the critic, denoted A2C +λ. When we use the learned reward shaping function Φ_{GCN}, we notice significant improvements over the classic actor-critic algorithm. We also compare our method to an expert-based approach for designing reward shaping functions, in which we define $\Phi_{expert}(s)$ as the distance between any state s and the position of the goal. Surprisingly, even though $\Phi_{expert}(s)$ essentially provides the perfect reward shaping function an agent would need to solve the task, our approximation Φ_{GCN} achieves similar, though not as good, results on all tasks.

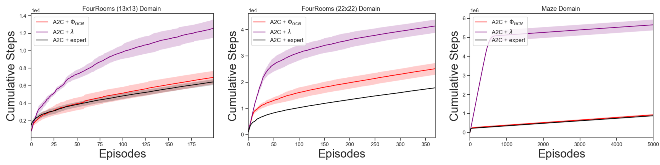
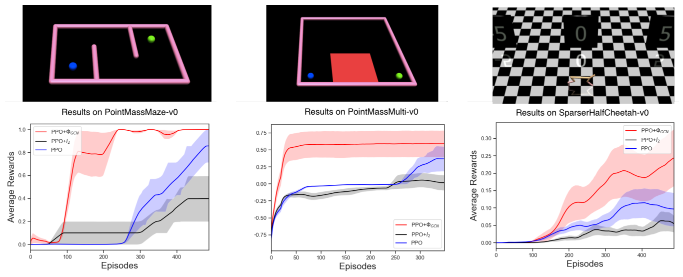


Figure 3: **Results on tabular domains for single-goal environments.** Cumulative steps for each configuration of the FourRooms domain (13x13, 22x22 and 29x29) and the Maze domain. As the complexity of the environment increases, we notice a clear advantage in using reward shaping.

4.2 Continuous control



(a) (b) (c)

Figure 4: **Results on continuous control:** a) PointMassMaze-v0 is a task where the agent (blue mass) has to get to the goal (green). For this environment, we only report the random seeds for which the agent has visited the goal state. b) PointMassMulti-v0 is a similar environment, but it contains a red region where the agent gets a negative reward. c) SparserHalfCheetah-v0 is a sparse variant of HalfCheetah-v0 (Brockman et al., 2016).

We further study how Φ_{GCN} can improve performance in continuous control. In the case of continuous states, it becomes intractable to represent all the possible transitions and states in the environment. To overcome this challenge, we will only sample at intervals the possible transitions in order to approximate the underlying graph. In the environments that we investigate, presented in Fig. 4, it would be challenging to provide the agent with a beneficial hand-crafted reward shaping function. The first environment is PointMassMaze-v0 where the agent (blue dot) has to navigate to the goal (green), where it receives +1 reward. In the second environment, PointMassMulti-v0, the agent has to navigate to the green goal while avoiding the dangerous region defined by the red square, which leads to negative reward. In both environments the actions are continuous. The last environment, SparserHalfCheetah-v0, is much more challenging. It is a very sparse variant of the classic HalfCheetah-v0 defined in OpenAI's Gym Brockman et al. (2016), in which the agent only receives a reward signal for reaching a distance of 15 units from the starting position (the HalfCheetah can only move in the x axis). When the agent reaches this distance, a reward of +1 is given and the episode terminates. As an additional baseline, we provide a reward shaping function that would naively implement the L_2 distance between any state and the goal position. We plot the results in Fig. 4, which shows a clear advantage in using Φ_{GCN} as a reward shaping function, indicating that our method is able to naturally scale to more complex environments.

5 Conclusion and future work

We presented a method for learning a scalable reward shaping function by using GCNs for propagating messages from rewarding states. We showed how the propagation rule of the GCN can approximate the backward messages defined in the probabilistic view of reinforcement learning. We showed empirical evidence that our method can lead to faster learning through better exploitation of the reward signal. Given the recent rise of interest in graph representation learning, it would interesting to investigate further connections with reinforcement learning. In particular, in this work we did not consider constructing the full graph of the underlying MDP (we also did not estimate the transition probabilities). Instead, we sampled trajectories and constructed subgraphs based on these samples. With this strategy, we are still able to implicitly represent the dynamics of the environments, as shown in the plots of Section 3. However, recent advances in graph representation learning (Ying et al., 2018) would potentially allow us to explicitly represent the whole underlying graph and transition dynamics. This approach could then, similarly to SLAM (Durrant-Whyte & Bailey, 2006), be used to perform optimal path planning.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL `http://arxiv.org/abs/1606.06565`.

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

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3352–3358. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL `http://dl.acm.org/citation.cfm?id=2832581.2832716`.

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

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.

Anna Harutyunyan, Tim Brys, Peter Vrancx, and Ann Nowé. Shaping mario with human advice. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pp. 1913–1914, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-3413-6. URL `http://dl.acm.org/citation.cfm?id=2772879.2773501`.

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

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL `http://arxiv.org/abs/1805.00909`.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2. URL `http://dl.acm.org/citation.cfm?id=645528.657613`.

Pierre-Yves Oudeyer and Frédéric Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1:245 – 270, 2007.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 3052–3056. AAAI Press, 2013. ISBN 978-1-57735-633-2. URL `http://dl.acm.org/citation.cfm?id=2540128.2540576`.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (19902010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, Sep. 2010. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2056368.

S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, June 2010. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2051031.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 945–952, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143963. URL `http://doi.acm.org/10.1145/1143844.1143963`.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *CoRR*, abs/1806.01973, 2018. URL `http://arxiv.org/abs/1806.01973`.