Discovering Intrinsically Motivated Goals

Juan Duque
Microsoft Corp.
juduqu@microsoft.com

Charles Isbell
Georgia Institute of Technology
isbell@cc.gatech.edu

Michael loss
Georgia Institute of Technology
loss@math.gatech.edu

Abstract

One of the fundamental problems in reinforcement learning is optimizing a policy in environments with sparse and delayed rewards. In this paper we propose a framework that extends from the hierarchical DQN algorithm (Kulkarni, Narasimhan, Saeedi, & Tenenbaum, 2016) to allow agents to explore the environment motivated by intrinsically defined goals. We show that our methodology effectively yields better results than DQN in complex environments that are often hard to explore. Our contribution to the h-DQN algorithm is using a spectral framework to determine goals, which we define as a discrete number of sub-regions of the state space. We name this new algorithm spectral h-DQN (sh -DQN) and it runs end to end without human intervention.

1 Introduction

Deep reinforcement learning has provided a methodology to improve decision making in a variety of domains, including Atari games (Mnih et al., 2015), Go (Silver et al., 2016) and autonomous driving (Chen, Liu, Everett, & How, 2017). In these tasks, reinforcement learning agents train a non-linear function approximator to map the state space to the action space in such way as to maximize a defined notion of reward in an environment. This approach, nonetheless, has shown significant limitations in environments with sparse and delayed rewards (Liu, Machado, Tesauro, & Campbell, 2017). Many of the efforts in reinforcement learning have tried to deal with these restrictions using temporally extended actions (Sutton, Precup, & Singh, 1999) and temporally extended sub-policies (Kulkarni et al., 2016), which simplify the exploration space by using layers of abstraction .

In nature, humans consistently decompose complicated tasks in smaller sub-problems that are easier to solve. Hierarchical Reinforcement Learning (HRL) is a mathematical framework derived from this idea, which formalizes decision processes as a composition of multiple optimization problems (Kulkarni et al., 2016). Despite their success in complex problems (Vezhnevets et al., 2017), many HRL algorithms rely on manual intervention for determining sub-goals. In this work, we introduce a methodology that allows h-DQN agents to discover sub-goals in an unsupervised manner. In such way, our framework allows the usage of Hierarchical Reinforcement Learning in complex, more general problems without human intervention.

Our contributions are formulating sub-goals as subsets of the state space and finding a reasonable partitioning of the state space into sub-goals that can be effectively differentiated. We do so by discretizing the state space into small sub-regions and constructing an adjacency matrix during an initial exploration phase. Then, we associate highly interconnected regions into clusters using spectral graph partitioning. Finally we train a non linear function approximator to serve as a critic that distinguishes between different clusters and provides the intrinsic reward.

2 Literature Review

2.1 The Options framework

Modelling higher levels of abstraction is an intuitive approach for dealing with complicated tasks. In reinforcement learning, the options framework (Sutton et al., 1999) allows the modelling of higher temporal abstractions in Markov decision processes (MDPs). This framework defines a two-level hierarchy in which the bottom is given by options: temporal policies, over a set of time steps, that terminate under a stochastic function . The top of the hierarchy is a policy over options that terminates when the overarching goal is met. Options are popular in reinforcement learning because they decompose tasks into more atomic subtasks that are easier to learn.

2.2 Hierarchical Deep Reinforcement Learning

Deep reinforcement learning techniques have gained significant attention in the past years for their success in problems with high-dimensional inputs (Mnih et al., 2015). Still, these methods perform poorly in tasks with convoluted state spaces and delayed rewards. A recent paper by Tejas Kulkarni(Kulkarni et al., 2016) integrates deep reinforcement learning with temporal abstractions. In this paper, a hierarchy is established with a controller and a meta-controller. The controller learns a policy, over a sequence of states, using an internal critic that provides a reward when an internal goal is achieved. The meta-controller learns a policy over goals, that maximizes the extrinsic reward on the environment. Our paper builds upon this hierarchical deep reinforcement learning framework.

2.3 Option discovery in RL

Unsupervised option discovery is an open problem in the state of the art of reinforcement learning. In the past, many approaches have been tried to deal with this issue. A formulation by (Mannor, Menache, Hoze, & Klein, 2004) uses k-means clustering to partition the state space and then defines options as transitions between the means of these clusters. More recent advances use clustering to define options, but without a predetermined number of clusters. Instead, new clusters are created for each state that is more than a selected number of deviations away from the closest cluster (Karimpanal & Wilhelm, 2017).

A different proposal uses the recent history of the states visited by the agent to create a local transition probability graph. Then the graph is partitioned into two, by finding a cut with the lowest between-states transition probability. Potential options are identified as the states that are endpoints of the edges that cross the cut (Şimşek, Wolfe, & Barto, 2005).

Yet, another approach from (Liu et al., 2017) proposes a Laplacian framework to discover options. A Laplacian matrix is constructed by subtracting a diagonal matrix from the adjacency matrix representation of the Markov decision process. Then the Laplacian matrix is diagonalized and the eigenvectors are used as proto value functions, i.e. alternative representations of the MDP. These proto value functions can be interpreted as options, in which the reward function is dependent on the geometry of the environment.

3 Model

Spectral graph partitioning: Given a Markov decision process with finite states and actions, $\langle\ \mathcal{S},\ \mathcal{A},\ p, r, \gamma \rangle$, we can represent the state space, $\mathcal{S}$, as a state-transition graph where each state, s, corresponds to a vertex in the graph. For each state pair, $s_1 \times s_2 \in \mathcal{S}\times\mathcal{S}$, we define a directed edge $e(s_1, s_2) \in E$ if there exists an action $a \in \mathcal{A}$, for which $p(s_2|s_1, a) > 0$. We can construct an estimate of this state-transition graph by allowing the agent to interact with the environment throughout a fixed number of random walks.

To formalize the notion of goal we define the finite set of all goals, G, as a partition of the state space, $\mathcal{S}$. Intuitively, we want this partitioning to capture a notion of similarity between states. Here we define this similarity metric as the interconnectedness between states, with the objective of separating regions of $\mathcal{S}$ that share few transitions between them and capture bottlenecks. We use assume symmetrical transitions, i.e. undirected edges to simplify the following optimization problem:

Without loss of generality[1]*, we want to find some labeling* $y \in [-1, +1]^n$*, where* $n = |S|$ *is the number of states, such that* $\sum\limits_{(i,j)\in E} (y_i - y_j)^2$ *is minimal.*

This is the formulation of spectral graph partitioning, we want to find a labeling of the graph into two subsets such that the number of edges crossing the split is minimal. As shown by Miroslav Fiedler, we can relax the constraints of the formulation above and choose instead some soft labeling $x \in \mathcal{R}^n$ such that x is a unit vector, orthogonal to the first eigenvector λ_1. Thus we get:

$$\operatorname*{argmin}_x \frac{\sum\limits_{(i,j)\in E} (x_i - x_j)^2}{||x||^2} = \operatorname*{argmin}_x \frac{\sum\limits_{i,j=1}^{n} (A_{ij} - D_{ij})(x_i x_j)}{||x||^2} = \operatorname*{argmin}_x \frac{x^T L x}{x^T x} = \lambda_2 \tag{1}$$

Where A is the adjacency matrix of the state transition graph, D is the graph's degree matrix, and L is the graph's Laplacian matrix. The spectral graph partitioning algorithm then sorts the components of the second smallest eigenvector of the graph's Laplacian and chooses some point, usually the median, to split the vertices of the graph into two subsets. Other, more expensive, methods of splitting exist that attempt to minimize normalized cut in 1-dimension. We assign each subset resulting from this partitioning, g, to be one of the goals in our algorithm.

[1]Note that this labeling partitions the graph into only two subsets, but we can easily overcome this by recursively partitioning the subsets until reaching the desired number of partitions.

Hierarchical Deep Reinforcement Learning: As described in the paper of the same name by Kulkarni et al., we define a Q_1^* function approximator for the controller, which maximizes the expected implicit reward given a state s, a goal, g, and an action, a:

$$\begin{aligned} Q_1^*(s,a;g) &= \max_{\pi_{a\,g}} E[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} | s_t = s, a_t = a, g_t = g, \pi_{a\,g}] \\ &= \max_{\pi_{a\,g}} E[r_t + \gamma \max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) | s_t = s, a_t = a, g_t = g, \pi_{a\,g}] \end{aligned} \tag{2}$$

A similar Q_2^* function approximator maximizes the expected extrinsic reward from the environment for the meta-controller:

$$Q_2^*(s,g) = \max_{\pi_g} E[\sum_{t'=t}^{t+N} f_{t'} + \gamma \max_{g'} Q_2^*(s_{t+N}, g') | s_t = s, g_t = g, \pi_g] \tag{3}$$

Where N denotes the number of time steps until the meta-controller stops the current goal, $\pi_{a\,g} = P(a|s,g)$ is the policy over the actions and $\pi_g = P(g|s)$ is the policy over goals. $Q^*(s,a;g) \approx Q(s,a;g)$ and $Q^*(s,g) \approx Q(s,g)$, Q_1 and Q_2 namely, are neural networks parametrized by θ_1 and θ_2 respectively. We can then minimize the square of the temporal difference error as follows:

$$L_1(\theta_1, i) = E_{(s,a,g,r,s') \sim D_1}[(r + \gamma \max_{a'} Q_1(s', a'; g, i, \theta_1)) - Q_1^*(s,a;g,i,\theta_1)]^2 \tag{4}$$

Where i is the training episode number and D_1 denotes the replay buffer of the experiences (s,a,g,r,s') stored by the controller. We can differentiate with respect to θ_1 to find the gradient for the controller:

$$\begin{aligned} &\nabla_{\theta_{1},i} L_1(\theta_1, i) \\ &= E_{(s,a,g,r,s') \sim D_1}[((r + \gamma \max_{a'} Q_1(s', a'; g, i, \theta_1)) - Q_1^*(s,a;g,i,\theta_1) \nabla_{\theta_{1},i} Q_1(s,a;g,i,\theta_1))] \end{aligned} \tag{5}$$

A similar procedure is used to derive the loss $L_2(\theta_2, i)$ and its gradient $\nabla_{\theta_2,i} L_2(\theta_2, i)$.

Other considerations for spectral graph partitioning: Since spectral graph partitioning only works well in fully connected graphs, we use different methods to discretize the state space into a tractable number of regions either using k-means clustering or rounding the values of each of the components of the state tuple, s. This vastly reduces the state space size and increases the connectivity between vertices of the state transition graph.

The other important consideration with regards to spectral graph partitioning is that, once the labeling has been assigned, there is no easy way to make predictions for states that have never been seen before. To deal with this problem, we use the states and the labels generated by the spectral graph partitioning algorithm to define a supervised learning problem and train another function approximator, C, with parameters θ_3. Where $C^*(s)$ is the correct spectral graph label of the state, s, and $C^*(s) \approx C(s;\theta_3)$.

Learning algorithm: We learn the parameters θ_1 and θ_2 using stochastic gradient descent at each iteration (i.e. step of the algorithm). The controller collects experience tuples,$([s,g], a, r_{in}, [s', g])$, at every time step and the meta-controller collects experience tuples,(s, g, r_{ex}, s') when the goal terminates (i.e. when a goal is re-picked or the episode ends). Here r_{in} is the intrinsic reward provided by an internal critic when the goal is reached and r_{ex} is the extrinsic reward of the environment.

4 Experiments

Environment description: We test our approach on "Mountain Car"as described by Andrew Moore: an environment in which a car located in a valley between two mountains receives a reward if it reaches the top of the mountain at its right. However, the car's engine is not strong enough to climb the mountain in a single pass. Thus, the car must accumulate enough momentum, by swinging between the two mountains, in order to reach its objective. We use an OpenAI gym implementation with discrete actions (push left, no push, push forward) and a state tuple composed by velocity and position values. This environment is well known in the reinforcement learning literature for requiring agent to perform very good exploration to be solved.

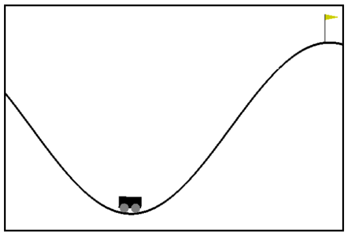
Figure 1: A sample frame of the Mountain Car environment

Setup: We allow the agent to explore the environment throughout 10 random walk episodes, where each episode finalizes after a maximum number of steps 10000 or when the goal is reached. We use this experience to build the state-transition graph and perform spectral graph partitioning with 10 partitions. We introduce an extra goal that only terminates when the state is terminal.

Algorithm 1: Spectral Hierarchical DQN

```
$graph \leftarrow []$;
for $i \leftarrow 1$ to $n$ do
    Initialize environment and get initial state $s_{current}$;
    $j \leftarrow 0$;
    $episodeStates \leftarrow []$;
    while not ($s_{current}$ is terminal or $j > m$) do
        Select random action $a$;
        Execute $a$ and obtain next state $s_{next}$;
        Append $(s_{current}, s_{next})$ to $episodeStates$;
        $s_{current} \leftarrow s_{next}$;
        $j \leftarrow j + 1$;
    end
    Append $episodeStates$ to $graph$;
    $i \leftarrow i + 1$;
end
$A \leftarrow ConstructAdjacencyMatrix(graph)$;
$labels \leftarrow SpectralGraphPartitioning(A)$;
$goals \leftarrow Set(labels)$;
Append $length(goals) + 1$ to $goals$;
Initialize $C$ with parameters $\theta_3$;
Train $C$ with states in $graph$ and $labels$;
Run the Hierarchical DQN algorithm with $goals$ and $critic(C, s, g, goals, done)$;
```

Algorithm 2: Critic

```
if ($g == C(Discretize(s))$) or ($g == length(goals)$ and $done$) then
    return 1;
end
return 0;
```

Results: We plot the output of the spectral graph partitioning algorithm on our sample of the state space to get an intuition of the geometry of the environment. As seen in Figure 2, the spiral-like pattern seems to indicate that clusters are formed primarily according to the total energy of the car, i.e., the sum of its potential and kinetic energies. The shape of the clusters captures important information about the dynamic of the environment that can be useful to optimize a wide range of different objectives.

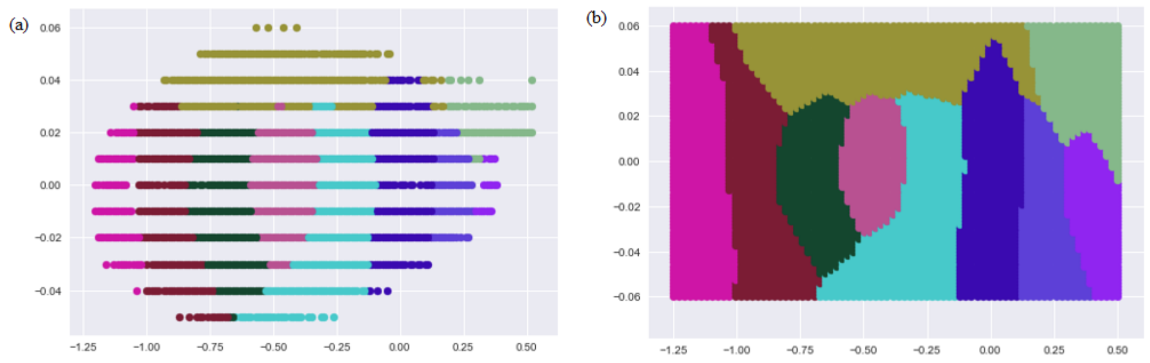

Figure 2: (a) Spectral graph partitioning on the discretized sampled state space. The x axis is the x position of the car, the y axis is the velocity. (b) The prediction of the cluster approximator, C, to a generalization of the state space.The x axis is the x position of the car, the y axis is the velocity.

We ran the training phase of the sh-DQN algorithm 10 times, calculated its average cumulative reward and compared it with a similar average from the training phase of the DQN algorithm. The sh-DQN algorithm converged to a local maximum in all but one of the training runs. The DQN algorithm, on the other hand, only converged in two of its runs. It is clear that the sh-DQN algorithm reaches a sub-optimal maxima and fairly easily outperforms DQN in this environment. The following are the episode reward averages within a 95% confidence interval:

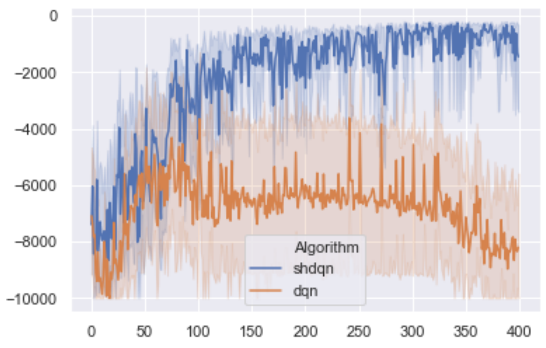

Figure 3: **Results in the Mountain Car environment:** Total rewards, at the end of each episode, for the sh-DQN and DQN algorithms during their training phases . The x axis is the number of episodes, the y axis is the total cumulative reward

5 Conclusion

As stated by Kulkarni et al. (2016), the main strength of h-DQN, by extension also of sh-DQN, is that intrinsically motivated goals allow for more efficient exploration of the environment. This ultimately allows sh-DQN to easily beat DQN in environments like Mountain Car, which require agents to do throughout exploration to converge to local optima. These results were expected as we were extending from h-DQN.

Ultimately we have shown that our formulation of goals as spectral partitions effectively captures the geometry of the environment, allowing the agent to easily learn tasks using these goals as abstractions. In such way, we have been successful in contributing to the completion of the h-DQN algorithm. In the future, however, we would like to extend the sh-DQN formulation to not depend on the dicretization of the environment and allow goals to be learned online.

6 Acknowledgements

We would like to thank Himanshu Sahni for all the useful discussions that provided invaluable guidance for this project. Special thanks to Saurabh Kumar for proposing using clusters as possible goals. Finally, thanks to Alejandro Aristizabal and Alen Polakof for providing assistance in the implementation of the algorithm.

References

Chen, Y. F., Liu, M., Everett, M., & How, J. P. (2017). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 ieee international conference on robotics and automation (icra)* (pp. 285–292).

Karimpanal, T. G., & Wilhelm, E. (2017). Identification and off-policy learning of multiple objectives using adaptive clustering. *Neurocomputing*, *263*, 39–47.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems* (pp. 3675–3683).

Liu, M., Machado, M. C., Tesauro, G., & Campbell, M. (2017). The eigenoption-critic framework. *arXiv preprint arXiv:1712.04065*.

Mannor, S., Menache, I., Hoze, A., & Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on machine learning* (p. 71).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... others (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484.

Şimşek, Ö., Wolfe, A. P., & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on machine learning* (pp. 816–823).

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 3540–3549).