Deep Reinforcement Learning with Temporal Abstraction and Intrinsic Motivation

Tejas D. Kulkarni1*
Karthik R. Narasimhan2*
Ardavan Saeedi2
Joshua B. Tenenbaum1
1Brain and Cognitive Sciences, MIT
2Computer Science and Artificial Intelligence Laboratory, MIT
*These authors contributed equally and are listed alphabetically.

Abstract

Learning goal-directed behavior in environments with sparse feedback is a major challenge for reinforcement learning algorithms. The primary difficulty arises due to insufficient exploration, resulting in an agent being unable to learn robust value functions. Intrinsically motivated agents can explore new behavior for its own sake, which could eventually help them solve tasks posed by the environment. We present hierarchical-DQN (h-DQN), a framework to integrate hierarchical value functions, operating at different temporal scales, with intrinsically motivated deep reinforcement learning. A top-level value function learns a policy over intrinsic goals, and a lower-level function learns a policy over atomic actions to satisfy the given goals. h-DQN allows for flexible goal specifications, such as functions over entities and relations, leading to efficient exploration in complicated environments. We demonstrate the strength of our approach on a problem with very sparse, delayed feedback – the classic ATARI game ‘Montezuma’s Revenge’.

1. Introduction

Learning goal-directed behavior with sparse feedback from complex environments is a fundamental challenge for artificial intelligence. Learning in this setting requires the agent to represent knowledge at multiple levels of spatio-temporal abstractions and to explore the environment efficiently. Recently, non-linear function approximators coupled with reinforcement learning (Koutník et al., 2014; Mnih et al., 2015; Silver et al., 2016) have made it possible to learn abstractions over high-dimensional state spaces, but the task of exploration with sparse feedback still remains a major challenge. Existing methods like Boltzmann exploration and Thomson sampling (Stadie et al., 2015; Osband et al., 2016) offer significant improvements over \(\epsilon\)-greedy, but are limited due to the underlying models functioning at the level of basic actions. In this work, we propose a framework that integrates deep reinforcement learning with hierarchical value functions (h-DQN), where the agent is motivated to solve intrinsic goals (via learning options) to aid exploration. These goals provide for efficient exploration and help mitigate the sparse feedback problem. Additionally, we observe that goals defined in the space of entities and relations can help significantly constrain the exploration space for data-efficient learning in complex environments.

Reinforcement learning (RL) formalizes control problems as finding a policy \(\pi\) that maximizes expected future rewards (Sutton & Barto, 1998). Value functions \(V(s)\) are central to RL, and they cache the utility of any state \(s\) in achieving the agent’s overall objective. Recently, value functions have also been generalized as \(V(s, g)\) in order to represent the utility of state \(s\) for achieving a given goal \(g\) \(\in G\) (Sutton et al., 2011; Schaul et al., 2015). When the environment provides delayed rewards, we adopt a strategy to first learn ways to achieve intrinsically generated goals, and subsequently learn an optimal policy to chain them together. Each of the value functions \(V(s, g)\) can be used to generate a policy that terminates when the agent reaches the goal state \(g\). A collection of these policies can be hierarchically arranged with temporal dynamics for learning or planning within the framework of semi-Markov decision processes (Sutton et al., 1999; Szepesvari et al., 2014). In high-dimensional problems, these value functions can be approximated by neural networks as \(V(s, g; \theta)\).
We propose a framework with hierarchically organized deep reinforcement learning modules working at different time-scales. The model takes decisions over two levels of hierarchy – (a) the top level module (meta-controller) takes in the state and picks a new goal, (b) the lower-level module (controller) uses both the state and the chosen goal to select actions either until the goal is reached or the episode is terminated. The meta-controller then chooses another goal and steps (a-b) repeat. We train our model using stochastic gradient descent at different temporal scales to optimize expected future intrinsic (controller) and extrinsic rewards (meta-controller). We demonstrate the strength of our approach on a classic ATARI game (‘Montezuma’s Revenge’) with long-range delayed rewards, where most existing state-of-art deep reinforcement learning approaches fail to learn policies in a data-efficient manner.

2. Related Work

There has been extensive work on trying to learn spatio-temporal abstractions (Sutton et al., 1999; Barto & Mahadevan, 2003; Sorg & Singh, 2010; Schaul et al., 2015; Koutník et al., 2014; Mnih et al., 2015; Silver et al., 2016), intrinsic rewards (Szepesvari et al., 2014; Singh et al., 2010; Schmidhuber, 2010; Singh et al., 2004; Mohamed & Rezende, 2015) and efficient exploration (Stadie et al., 2015; Osband et al., 2016) in the context of reinforcement learning. Due to space limitations, we refer the reader to a longer version of the paper for a more thorough literature review – (Kulkarni et al., 2016).

3. Model

Consider a Markov decision process (MDP) represented by states $s \in S$, actions $a \in A$, and transition function $T : (s, a) \rightarrow s'$. An agent operating in this framework receives a state $s$ from the external environment and can take an action $a$, which results in a new state $s'$. We define the extrinsic reward function as $R : (s) \rightarrow \mathbb{R}$. The objective of the agent is to maximize this function over long periods of time. For example, this function can take the form of the agent’s survival time or score in a game.

**Agents** Effective exploration in MDPs is a significant challenge in learning good control policies. Methods such as $\epsilon$-greedy are useful for local exploration but fail to provide impetus for the agent to explore different areas of the state space. In order to tackle this, we utilize a notion of goals $g \in \mathcal{G}$, which provide intrinsic motivation for the agent. The agent focuses on setting and achieving sequences of goals in order to maximize cumulative extrinsic reward.

We use the temporal abstraction of options (Sutton et al., 1999) to define policies $\pi_g$ for each goal $g$. The agent learns these option policies simultaneously along with learning the optimal sequence of goals to follow. In order to learn each $\pi_g$, the agent also has a critic, which provides intrinsic rewards, based on whether the agent is able to achieve its goals (see Figure 1).

**Temporal Abstractions** As shown in Figure 1, the agent uses a two-stage hierarchy consisting of a controller and a meta-controller. The meta-controller receives state $s_t$ and chooses a goal $g_t \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of all possible current goals. The controller then selects an action $a_t$ using $s_t$ and $g_t$. The goal $g_t$ remains in place for the next few time steps either until it is achieved or a terminal state is reached. The internal critic is responsible for evaluating whether a goal has been reached and providing an appropriate reward $r_t(g)$ to the controller. The objective function for the controller is to maximize cumulative intrinsic reward: $R_t(g) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_t(g)$. Similarly, the objective of the meta-controller is to optimize the cumulative extrinsic reward $F_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} f_t$, where $f_t$ are reward signals received from the environment.

One can also view this setup as similar to optimizing over the space of optimal reward functions to maximize fitness (Singh et al., 2009). In our case, the reward functions are dynamic and temporally dependent on the sequential history of goals. Figure 1 provides an illustration of the agent’s use of the hierarchy over subsequent time steps.

**Learning Algorithm** We learn the parameters of h-DQN using stochastic gradient descent at different time scales – experiences (or transitions) from the controller are collected at every time step but experiences from meta-controller are only collected when the controller terminates (i.e. when a goal is re-picked or the episode ends). Each new goal $g$ is drawn in an $\epsilon$-greedy fashion with the exploration probability $\epsilon_2$ annealed as learning proceeds (from a starting value of 1).

In the controller, at every time step, an action is drawn with a goal using the exploration probability $\epsilon_{1,g}$ which is dependent on the current empirical success rate of reaching $g$. The model parameters $(\theta_1, \theta_2)$ are periodically updated by drawing experiences from replay memories $D_1$ and $D_2$, respectively. For a formal treatment of the model and the algorithm, we refer the reader to a longer version of the paper – (Kulkarni et al., 2016).

4. Experiment

**ATARI game with delayed rewards** We consider ‘Montezuma’s Revenge’, an ATARI game with sparse, delayed rewards (Figure 4). The game requires the player to navigate the explorer through several rooms while collecting treasures. In order to pass through doors (in the top right
Deep Reinforcement Learning with Temporal Abstraction and Intrinsic Motivation

Figure 1. Overview: The agent produces actions and receives sensory observations. Separate deep-Q networks are used inside the meta-controller and controller. The meta-controller that looks at the raw states and produces a policy over goals by estimating the value function $Q_2(s_t, g_t; \theta_2)$ (by maximizing expected future extrinsic reward). The controller takes in states and the current goal, and produces a policy over actions by estimating the value function $Q_1(s_t, a_t; \theta_1, g_t)$ to solve the predicted goal (by maximizing expected future intrinsic reward). The internal critic checks if goal is reached and provides an appropriate intrinsic reward to the controller. The controller terminates either when the episode ends or when $g$ is accomplished. The meta-controller then chooses a new $g$ and the process repeats.

and top left corners of the figure), the player has to first pick up the key. The player has to then climb down the ladders on the right and move left towards the key, resulting in a long sequence of actions before receiving a reward (+100) for collecting the key. After this, navigating towards the door and opening it results in another reward (+300).

Existing deep RL approaches fail to learn in this environment since the agent rarely reaches a state with non-zero reward. For instance, the basic DQN (Mnih et al., 2015) achieves a score of 0 while even the best performing system, Gorila DQN (Nair et al., 2015), manages only 4.16 on average.

Setup. The agent needs intrinsic motivation to explore meaningful parts of the scene before it can learn about the advantage of getting the key for itself. Inspired by the developmental psychology literature (Spelke & Kinzler, 2007) and object-oriented MDPs (Diuk et al., 2008), we use entities or objects in the scene to parameterize goals in this environment. Unsupervised detection of objects in visual scenes is an open problem in computer vision, although there has been recent progress in obtaining objects directly from image or motion data (Fragkiadaki et al., 2015; Eslami et al., 2016; Greff et al., 2015). In this work, we built a custom object detector that provides plausible object candidates for goals. The controller and meta-controller are convolutional neural networks (see Figure 2) that learn representations from raw pixel data. We use the Arcade Learning Environment (Bellemare et al., 2012) to perform experiments. The internal critic is defined in the space of $\langle\text{entity}_1, \text{relation}, \text{entity}_2\rangle$, where relation is a function over configurations of the entities. In our experiments, the agent is free to choose any $\text{entity}_2$. The agent is deemed to have completed a goal (and receives a reward) if the agent entity reaches another entity such as the door.\footnote{Note that some of these objects might not be ‘good’ goals. The agent explores and learns the optimal sequence of goals to choose.} Note that this notion of relational intrinsic rewards can be generalized to other settings. For instance, in the ATARI game ‘Asteroids’, the agent could be rewarded when the bullet reaches the asteroid or if simply the ship never reaches an asteroid. In the game of ‘Pacman’, the agent could be rewarded if the pellets on the screen are reached. In the most general case, we can potentially let the model evolve a parameterized intrinsic reward function given entities. We leave this for future work.

Model Architecture and Training. Our model consists of stacked convolutional layers with rectified linear units (ReLU) (Figure 2). The input to the meta-controller is a set of four consecutive images of size $84 \times 84$. To encode the goal output from the meta-controller, we append a binary mask of the goal location in image space along with the original 4 consecutive frames. This augmented input is passed to the controller. The experience replay memories $D_1$ and $D_2$ were set to be equal to 1E6 and 5E4 respectively. We set the learning rate to be 2.5E−4, with a discount rate
Fig. 2. Architecture: DQN architecture for the controller ($Q_1$). A similar architecture produces $Q_2$ for the meta-controller (without goal as input). In practice, both these networks could share lower level features but we do not enforce this.

of 0.99. We follow a two phase training procedure – (1) In the first phase, we set the exploration parameter $\epsilon_2$ of the meta-controller to 1 and train the controller on actions. This effectively leads to pre-training the controller so that it can learn to solve a subset of the goals. (2) In the second phase, we jointly train the controller and meta-controller.

Fig. 3. Results on Montezuma’s Revenge: These plots depict the joint training phase of the model. As described in Section 4, the first training phase pre-trains the lower level controller for about 2.3 million steps. The joint training learns to consistently get high rewards after additional 2 million steps as shown in (a).

Results Figure 3 shows reward progress from the joint training phase from which it is evident that the model starts gradually learning to both reach the key and open the door to get a reward of around +400 per episode. The agent learns to choose the key more often as training proceeds and is also successful at reaching it. As training proceeds, we observe that the agent first learns to perform the simpler goals (such as reaching the right door or the middle ladder) and then slowly starts learning the ‘harder’ goals such as the key and the bottom ladders, which provide a path to higher rewards.

In order to scale-up to solve the entire game, several key ingredients are missing such as – automatic discovery of objects from videos to aid goal parametrization we considered, a flexible short-term memory, ability to intermittently terminate ongoing options.

We show some screen-shots from a test run with our agent (with epsilon set to 0.1) in Figure 4, as well as a sample animation of the run.3

5. Conclusion

We have presented h-DQN, a framework consisting of hierarchical value functions operating at different time scales. Temporally decomposing the value function allows the agent to perform intrinsically motivated behavior, which in turn yields efficient exploration in environments with delayed rewards. We also observe that parameterizing intrinsic motivation in the space of entities and relations provides a promising avenue for building agents with temporally extended exploration. We also plan to explore alternative parameterizations of goals with h-DQN in the future.

References


3Sample trajectory of a run on ’Montezuma’s Revenge’ – https://goo.gl/3264Ji
Figure 4. **Sample gameplay by our agent on Montezuma’s Revenge:** The four quadrants are arranged in a temporally coherent manner (top-left, top-right, bottom-left and bottom-right). At the very beginning, the meta-controller chooses key as the goal (illustrated in red). The controller then tries to satisfy this goal by taking a series of low level actions (only a subset shown) but fails due to colliding with the skull (the episode terminates here). The meta-controller then chooses the bottom-right ladder as the next goal and the controller terminates after reaching it. Subsequently, the meta-controller chooses the key and the top-right door and the controller is able to successfully achieve both these goals.


Schaul, Tom, Horgan, Daniel, Gregor, Karol, and Silver, David. Universal value function approximators. In *Pro-
Deep Reinforcement Learning with Temporal Abstraction and Intrinsic Motivation


