Unsupervised Methods For Subgoal Discovery During Intrinsic Motivation in Model-Free Hierarchical Reinforcement Learning

Jacob Rafati and **David C. Noelle**
{jrafatiheravi,dnoelle}@ucmerced.edu
Electrical Engineering and Computer Science
Computational Cognititve Neurosceince Laboratory
University of California, Merced
5200 North Lake Road, Merced, CA 95343, USA

Abstract

Common approaches to Reinforcement Learning (RL) are seriously challenged by large-scale applications involving huge state spaces and sparse delayed reward feedback. Hierarchical Reinforcement Learning (HRL) methods attempt to address this scalability issue by learning action selection policies at multiple levels of *temporal abstraction*. Abstraction can be had by identifying a relatively small set of states that are likely to be useful as subgoals, in concert with the learning of corresponding skill policies to achieve those subgoals. Many approaches to *subgoal discovery* in HRL depend on the analysis of a model of the environment, but the need to learn such a model introduces its own problems of scale. Once subgoals are identified, skills may be learned through *intrinsic motivation*, introducing an internal reward signal marking subgoal attainment. In this paper, we present a novel model-free method for subgoal discovery using incremental unsupervised learning over a small memory of the most recent experiences (trajectories) of the agent. When combined with an intrinsic motivation learning mechanism, this method learns both subgoals and skills, based on experiences in the environment. Thus, we offer an original approach to HRL that does not require the acquisition of a model of the environment, suitable for large-scale applications. We demonstrate the efficiency of our method on two RL problems with sparse delayed feedback: a variant of the rooms environment and the first screen of the ATARI 2600 *Montezuma's Revenge* game.

Introduction

The reinforcement learning problem suffers from serious scaling issues. *Hierarchical Reinforcement Learning* (HRL) is an important computational approach intended to tackle problems of scale by learning to operate over different levels of *temporal abstraction* (Sutton, Precup, and Singh 1999). The acquisition of hierarchies of reusable skills is one of the distinguishing characteristics of biological intelligence, and the learning of such hierarchies is an important open problem in computational reinforcement learning. Also, in the context of games, the development of robust HRL methods will provide a means to acquire relevant knowledge at multiple levels of abstraction, potentially speeding learning and supporting generalization.

A number of approaches to HRL have been suggested. One approach focuses on action sequences, subpolicies, or "options" that appear repeatedly during the learning of a set of tasks. Such frequently reused subpolicies can be abstracted into skills that can be treated as individual actions at a higher level of abstraction. A somewhat different approach to temporal abstraction involves identifying a set of states that make for useful *subgoals*. This introduces a major open problem in HRL: that of *subgoal discovery*.

A variety of researchers have proposed approaches to identifying useful subpolicies and reifying them as skills (Pickett and Barto 2002; Thrun and Schwartz 1995). For example, (Sutton, Precup, and Singh 1999) proposed the *options framework*, which involves abstractions over the space of actions. At each step, the agent chooses either a one-step "primitive" action or a "multi-step" action policy (an option). Each option defines a policy over actions (either primitive or other options) and comes to completion according to a termination condition. Other researchers have focused on identifying subgoals — states that are generally useful to attain — and learning a collection of skills that allow the agent to efficiently reach those subgoals. Some approaches to subgoal discovery maintain the value function in a large look-up table (Sutton, Precup, and Singh 1999; Goel and Huber 2003; Şimşek, Wolfe, and Barto 2005), and most of these methods require building the state transition graph, providing a model of the environment and the agents possible interactions with it (Machado, Bellemare, and Bowling 2017; Şimşek, Wolfe, and Barto 2005; Goel and Huber 2003). Formally, the state transition graph is a directed graph $G = (V, E)$ with a set of vertices, $V \subseteq \mathcal{S}$ and set of edges $E \subseteq \mathcal{A}(\mathcal{S})$, where $\mathcal{S}$ is the set of states and $\mathcal{A}(\mathcal{S})$ is the set of allowable actions. Since actions typically modify the state of the agent, each directed edge, $(s, s') \in E$, indicates an action that takes the agent from state s to state s'. In nondeterministic environments, a probability distribution over subsequent states, given the current state and an action, $p(s'|s, a)$, is maintained as part of the model of the environment. One method of this kind that was applied to a somewhat larger scale task — the first screen of the ATARI 2600 game called *Montezuma's Revenge* — is that of Machado & Bowling (2016). This method constructs the combinatorial transition graph Laplacian matrix, and an eigen-decomposition of that matrix produces candidate subgoals. While it was shown that some of these candidates make for useful subgoals, only heuristic domain-sensitive methods have been reported for identifying useful

subgoals from the large set of candidates (e.g., thousands). Thus, previously proposed subgoal discovery methods have provided useful insights and have been demonstrated to improve learning, but there continue to be challenges with regard to scalability and generalization. Scaling to large state spaces will generally mandate the use of some form of nonlinear function approximator to encode the value function, rather than a look-up table. More importantly, as the scale of reinforcement learning problem increases, the tractability of obtaining a good model of the environment, capturing all relevant state transition probabilities, precipitously decreases.

Once useful subgoals are discovered, an HRL agent should be able to learn the skills to attain those subgoals through the use of *intrinsic motivation* — artificially rewarding the agent for attaining selected subgoals. The nature and origin of "good" intrinsic reward functions is an open question in reinforcement learning, however, and a number of approaches have been proposed. Singh et al. (2010) explored agents with intrinsic reward structures in order to learn generic options that can apply to a wide variety of tasks. Value functions have also been generalized to consider goals along with states (Vezhnevets et al. 2017). Such a parameterized universal value function, $q(s, g, a; w)$, integrates the value functions for multiple skills into a single function taking the current subgoal, g, as an argument.

Recently, Kulkarni et al. (2016) proposed a scheme for temporal abstraction that involves simultaneously learning options and a hierarchical control policy in a deep reinforcement learning framework. Their approach does not use separate Q-functions for each option, but, instead, treats the option as an argument. This method lacks a technique for automatic subgoal discovery, however, forcing the system designer to specify a set of promising subgoal candidates in advance. The approach proposed in this paper is inspired by Kulkarni et al. (2016), which has advantages in terms of scalability and generalization, but it incorporates automatic subgoal discovery.

It is important to note that *model-free* HRL, which does not require a model of the environment, still often requires the learning of useful internal representations of states. When learning the value function using a nonlinear function approximator, such as a deep neural network, relevant features of states must be extracted in order to support generalization at scale. A number of methods have been explored for learning such internal representations during model-free reinforcement learning (Tesauro 1995; Rafati and Noelle 2017; Mnih et al. 2015).

In this paper, we seek to address major open problems in the integration of internal representation learning, temporal abstraction, automatic subgoal discovery, and intrinsic motivation learning, all within the model-free HRL framework. We propose and implement efficient and general methods for subgoal discovery using unsupervised learning and anomaly (outlier) detection. These methods do not require information beyond that which is typically collected by the agent during model-free reinforcement learning, such as a small memory of recent experiences (agent trajectories). Our methods are fundamentally constrained in three ways, by design. First, we remain faithful to a model-free reinforcement learning framework, eschewing any approach that requires the learning or use of an environment model. Second, we are devoted to integrating subgoal discovery with intrinsic motivation learning. Specifically, we conjecture that intrinsic motivation learning can increase appropriate state space coverage, supporting more efficient subgoal discovery. Lastly, we focus on subgoal discovery algorithms that are likely to scale to large reinforcement learning tasks. The result is a unified model-free HRL algorithm that incorporates the learning of useful internal representations of states, automatic subgoal discovery, intrinsic motivation learning of skills, and the learning of subgoal selection by a "meta-controller". We demonstrate the effectiveness of this algorithm by applying it to a variant of the rooms task (illustrated in Figure 2(a)), as well as the initial screen of the ATARI 2600 game called *Montezuma's Revenge* (illustrated in Figure 3(a)).

Reinforcement Learning Problem

The Reinforcement Learning (RL) problem is learning through interaction with an *environment* (Sutton and Barto 1998). At each time step the *agent* receives a representation of the environment's *state*, $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. On that basis, the agent selects an *action*, $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of all available actions. One time step later, as a consequence of the agent's action, the agent receives a *reward*, $r \in \mathbb{R}$, and also an update on the agent's new state, s', from the environment. Each cycle of interaction is called a transition *experience*, $e = (s, a, r, s')$. At each time step, the agent implements a mapping from states to possible actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, called its *policy*. The goal of the RL agent is to find an *optimal policy* that maximizes the expected value of the *return*, i.e. the cumulative sum of future rewards, $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'+1}$, where $\gamma \in (0, 1]$ is the *discount factor* and T is a final step. The Temporal Difference (TD) learning approach is a class of *model-free* RL methods that attempt to learn a policy without learning a model of the environment. It is often useful to define a *value* function $q_\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ to estimate the expected value of the return, following policy π. When the state space is large, or not all states are observable, we can use a nonlinear function approximator, such as an artificial neural network (Mnih et al. 2015), or a linear approximation (Liang et al. 2016), to calculate $Q(s, a; w)$, an estimate the value function q_π, parameterized by w. Q-learning is a TD algorithm that attempts to find the optimal value function by minimizing the loss function $L(w)$, which is defined as the expectation of squared *TD error* over a recent transition *experience memory*, $\mathcal{D}$:

$$L(w) \triangleq \mathbb{E}_{e \sim \mathcal{D}} \Big[\big(r + \gamma \max_{a'} Q(s', a'; w) - Q(s, a; w) \big)^2 \Big].$$

A Unified Model-Free HRL Framework

In Hierarchical Reinforcement Learning (HRL), a central goal is to support the learning of representations at multiple levels of abstraction. As a simple example, consider the task of navigation in the *4-room environment with a key and a*

lock in Figure 2(a). This is a variant of the *rooms* task (Sutton, Precup, and Singh 1999). The 4-room is a grid-world environment consisting of 4 rooms. Each grid square is a state, and the agent has access to the Cartesian location of each grid square. Actions allow the agent to move to an adjacent grid square. The 4 rooms are connected through *doorways*. The agent is rewarded for entering the grid square containing the key, and it is more substantially rewarded for entering the grid square with the lock after obtaining the key. Learning this task based on sparse delayed feedback is challenging for a reinforcement learning agent.

Our intuition, shared with other researchers, is that hierarchies of abstraction will be critical for successfully solving problems of this kind. To be successful, the agent should represent knowledge at multiple levels of spatial and temporal abstraction. Appropriate abstraction can be had by identifying a relatively small set of states that are likely to be useful as *subgoals* and jointly learning the corresponding skills of achieving these subgoals, using intrinsic motivation.

In this section, we introduce a unified method for model-free HRL. The major components of our framework, and the information flow between them, are sketched in Figure 1. Before describing the unified method, we introduce the various components of our framework.

Meta-Controller and Controller Framework

Inspired by Kulkarni et al. (2016), we start by using two levels of hierarchy (Figure 1). The more abstract level of this hierarchy is managed by a *meta-controller* which guides the action selection processes of the lower level *controller*. Separate value functions are learned for the meta-controller and the controller. At time step t, the meta-controller receives a state observation, $s = s_t$, from the environment. It has a policy for selecting a *subgoal*, $g = g_t$, from a set of subgoals, $\mathcal{G}$. In our implementation, the policy arises from estimating the value of each subgoal, $Q(s, g; \mathcal{W})$, and selecting the goal of highest estimated value (except when performing random exploration). With the current subgoal selected, the controller uses its policy to select an action, $a \in \mathcal{A}$, based on the current state, s, and the current subgoal, g. In our implementation, this policy involves selecting the action that results in the highest estimate of the controller's value function, $q(s, g, a; w)$. Actions continue to be selected by the controller while an internal critic monitors the current state, comparing it to the current subgoal, and delivering an appropriate *intrinsic reward*, $\tilde{r}$, to the controller on each time step. Each transition experience, $(s, g, a, \tilde{r}, s')$, is recorded in the controller's experience memory set, $\mathcal{D}_1$, to support learning. When the subgoal is attained, or a maximum amount of time has passed, the meta-controller observes the resulting state, $s_{t'} = s_{t+T+1}$, and selects another subgoal, $g' = g_{t+T+1}$, at which time the process repeats, but not before recording a transition experience for the meta-controller, $(s, g, G, s_{t'})$ in the meta-controller's experience memory set, $\mathcal{D}_2$. The parameters of the value function approximators are adjusted based on the collections of recent experiences. For training the meta-controller value function, we minimize a loss function based on the reward received from the environment:

$$\mathcal{L}_i(\mathcal{W}) \triangleq \mathbb{E}_{(s,g,G,s_{t'})\sim\mathcal{D}_2}\left[\left(\mathcal{Y} - Q(s, g; \mathcal{W})\right)^2\right], \quad (1)$$

where $G = \sum_{t'=t}^{t+T} \gamma^{t'-t} r_{t'}$ is the accumulated external reward (return) between the selection of consecutive subgoals. The target value for the expected return at the time that the meta-controller selected subgoal g is $\mathcal{Y} = G + \gamma \max_{g'} Q(s', g'; \mathcal{W})$. The controller improves its subpolicy, $\pi(a|s, g)$, by learning its value function, $q(s, g, a; w)$, over the set of recorded transition experiences. The controller updates its value function approximator parameters, w, so as to minimize its loss function:

$$L_i(w) \triangleq \mathbb{E}_{(s,g,a,\tilde{r},s')\sim\mathcal{D}_1}\left[\left(y - q(s, g, a; w)\right)^2\right], \quad (2)$$

where $y = \tilde{r} + \gamma \max'_a q(s', g, a'; w)$ is the target expected intrinsic return value.

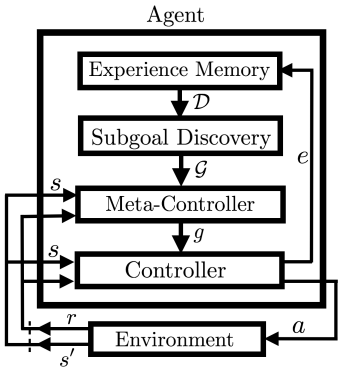

Figure 1: The information flow in the unified Model-Free Hierarchical Reinforcement Learning framework.

Intrinsic Motivation Learning

Intrinsic motivation learning is the core idea behind the learning of value functions in the meta-controller and the controller. In some tasks with sparse delayed feedback, a standard RL agent cannot effectively explore the state space so as to have a sufficient number of rewarding experiences to learn how to maximize rewards. In contrast, the intrinsic critic in our HRL framework can send much more regular feedback to the controller, since it is based on attaining subgoals, rather than ultimate goals. As an example, our implementation typically awards an intrinsic reward of $+1$ when the agent attains the current subgoal, g, and -1 for any other state transition. Successfully solving a difficult task not only depends on such an intrinsic motivation learning mechanism, but also on the meta-controller's ability to learn how to choose the right subgoal for any given state, s, from a set of candidate subgoals. Indeed, identifying a good set of candidate subgoals is an additional prerequisite for success, and it is discussed next.

Unsupervised Subgoal Discovery

The performance of the meta-controller/controller framework depends critically on selecting good candidate subgoals for the meta-controller to consider.

What is a subgoal? In our framework, a subgoal is a state, or a set of related states, that satisfies at least one of these conditions:

1. It is close (in terms of actions) to a rewarding state. For example, in the rooms task in Figure 2(a), the key and lock are rewarding states.
2. It represents a set of states, at least some of which tend to be along a state transition path to a rewarding state.

For example, in the rooms task, the red room should be visited to move from the purple room to the blue room in order to pick up the key. Thus any state in the red room is a reasonably good subgoal for an agent currently in the purple room. Similarly, the states in the blue room are all reasonably good subgoals for an agent currently in the red room. The doorways between rooms can also be considered as good subgoals, since entering these states allows for the transition to a set of states that may be closer to rewarding states.

Our strategy involves leveraging the set of recent transition experiences that must be recorded for value function learning, regardless. Unsupervised learning methods applied to sets of experiences can be used to identify sets of states that may be good subgoal candidates. We focus specifically on two kinds of analysis that can be performed on the set of transition experiences. We hypothesize that good subgoals might be found by (1) attending to the states associated with *anomalous* transition experiences and (2) clustering experiences based on a similarity measure and collecting the set of associated states into a potential subgoal. Thus, our proposed method merges *anomaly (outlier) detection* with the K-means clustering of experiences.

Anomaly Detection The anomaly (outlier) detection process identifies states associated with experiences that differ significantly from the others. In the context of subgoal discovery, a relevant anomalous experience would be one that includes a substantial positive reward in an environment in which reward is sparse. We propose that the states associated with these experiences make for good candidate subgoals. For example, in the rooms task, transitions that arrive at the key or the lock are quite dissimilar to most transitions, due to the large positive reward that is received at that point.

Since the goal of RL is maximizing accumulated (discounted) reward, these anomalous experiences, involving large rewards, are ideal as subgoal candidates. (Experiences involving large negative rewards are also anomalous, but make for poor subgoals. As long as these sorts of anomalies do not greatly outnumber others, we expect that the meta-controller can efficiently learn to avoid poor subgoal choices.) Large changes in state features can also be marked as anomalous. In some computer games, like *Montezuma's Revenge*, each screen represents a room, and the screen changes quickly when the agent moves from one room to another. This produces a large distance between two consecutive states. Such a transition can be recognized simply by the large instantaneous change in state features, marking the associated states as reasonable candidate subgoals. There is a large literature on anomaly detection (Hodge and Austin 2004), in general, offering methods for applying this insight. Heuristic meta-parameter thresholds can be used to identify dissimilarities that warrant special attention, or unsupervised machine learning methods can be used to model the joint probability distribution of state variables, with low probability states seen as anomalous.

K-Means Clustering The idea behind using a clustering algorithm is "spatial" state space abstraction and dimensionality reduction with regard to the internal representations of states. If a collection of transition experiences are very similar to each other, this might suggest that the associated states are all roughly equally good as subgoals. Thus, rather than considering all of those states, the learning process might be made faster by considering a representative state (or smaller set of states), such as the centroid of a cluster, as a subgoal. Furthermore, using a simple clustering technique like K-means clustering to find a small number of centroids in the space of experiences is likely to produce centroid subgoals that are dissimilar from each other. Since rewards are sparse, this dissimilarity will be dominated by state features. For example, in the rooms task, the centroids of K-means clusters, with $K = 4$, lie close to the geometric center of each room, with the states within each room coming to belong to the corresponding subgoal's cluster. In this way, the clustering of transition experiences can approximately produce a coarser representation of state space, in this case replacing the fine grained "grid square location" with the coarser "room location".

A Unified Framework

These conceptual components can be unified into a single model-free HRL framework. The major components of this framework, and the information flow between these components, are schematically displayed in Figure 1. At time t, the meta-controller observes the state, $s = s_t$, from the environment and chooses a subgoal, $g = g_t$, either from the discovered subgoals or from a random set of states (to promote exploration). The controller receives an input tuple, (s, g), and is expected to learn to implement a subpolicy, $\pi(a|s, g)$, that solves the *subtask* of reaching from s to g. The controller selects an action, a, based on its policy, in our case directly derived from its value function, $q(s, g, a; w)$. After one step, the environment updates the state to s' and sends a reward r. The transition experience $(s, g, a, \tilde{r}, s')$ is then stored in the experience memory for the controller, $\mathcal{D}_1$. If the internal critic detects that the resulting state, s', is the current goal, g, the experience $(s_t, g, G, s_{t'})$ is stored in the meta-controller experience memory, $\mathcal{D}_2$, where s_t is the state that prompted the selection of the current subgoal, and $s_{t'} = s_{t+T}$ is the state when the meta-controller assigns the next subgoal, $g' = g_{t'}$. The experience memory sets are typically used to train the value function approximators for the meta-controller and the controller by sampling a random minibatch of recent experiences. The subgoal discovery mechanism exploits the underlying structure in the experience memory sets using unsupervised anomaly detection and experience clustering. A detailed description of this process is outlined in Algorithm 1.

Algorithm 1 Unified Model-Free HRL Algorithm

Initialize discovered subgoals set $\mathcal{G} \leftarrow \emptyset$
Initialize experience memories $\mathcal{D}$, $\mathcal{D}_1$ and $\mathcal{D}_2$
Initialize parameters w and $\mathcal{W}$ randomly
Inputs: learning rate α, exploration rate ϵ
Choose Phase I or Phase II
for episode $= 1, \ldots, M$ **do**
  Initialize state $s_0 \in \mathcal{S}$, $s \leftarrow s_0$
  Initialize episode return $G \leftarrow 0$
  if Phase I of learning **then**
    Choose a subgoal g randomly form $\mathcal{S}$
  else if Phase II of learning **then**
    $g \leftarrow$EPSILON-GREEDY$(Q(s, \mathcal{G}; \mathcal{W}), \epsilon)$
  end if
  Intrinsic Motivation Learning Algorithm:
  repeat for each step $t = 1, \ldots, T$
    compute $q(s, g, a; w)$
    $a \leftarrow$EPSILON-GREEDY$(q(s, g, \mathcal{A}; w), \epsilon)$
    Take action a, observe s' and external reward r
    Compute intrinsic reward $\tilde{r}$ from internal critic
    Store $(s, g, a, \tilde{r}, s')$ to $\mathcal{D}_1$
    Store (s, a, r, s') to $\mathcal{D}$
    Sample $J_1 \subset \mathcal{D}_1$ and compute ∇L
    Update w, $\quad w \leftarrow w - \alpha \nabla L$
    if Phase II of learning **then**
      Sample $J_2 \subset \mathcal{D}_2$ and compute $\nabla \mathcal{L}$
      Update $\mathcal{W}$, $\quad \mathcal{W} \leftarrow \mathcal{W} - \alpha \nabla \mathcal{L}$
    end if
    $s \leftarrow s'$, $\quad G \leftarrow G + r$
    Decay exploration rate ϵ
  until s is terminal or subgoal g is attained
  Store (s_0, g, G, s') to $\mathcal{D}_2$
  if Phase I of training **then**
    Run Unsupervised Subgoal Discovery
  end if
end for
==========
Unsupervised Subgoal Discovery Algorithm
for each $e = (s, a, r, s')$ stored in $\mathcal{D}$ **do**
  if experience e is an outlier (anomaly) **then**
    Store s' to the subgoals set $\mathcal{G}$
    Remove e from $\mathcal{D}$
  end if
end for
Fit a K-means Clustering Algorithm on $\mathcal{D}$ using previous centroids as initial points
Store the updated centroids to the subgoals set $\mathcal{G}$

Experiments

We conducted simulation experiments in order to investigate the ability of our unsupervised subgoal discovery method to discover useful subgoals, as well as the efficiency of our unified model-free hierarchical reinforcement learning framework. The simulations were conducted in two environments with sparse delayed feedback: a variant of the rooms task, shown in Figure 2(a), and the "Montezumas Revenge" game, shown in Figure 3(a).

In these simulations, learning occurred in two phases. In **Phase I** of learning, the controller learned how to navigate in the state space, from any arbitrary state to any other state, using intrinsic motivation learning. The agent's trajectories, (s, a, r, s'), during this pretraining phase were stored in $\mathcal{D}$, and our unsupervised subgoal discovery method extracted the structure of $\mathcal{D}$ by performing K-means clustering and anomaly detection. The discovered subgoals were stored in $\mathcal{G}$. In **Phase II** of learning, the meta-controller learned meta-policies over G, and the controller had opportunities to refine its ability to navigate from any arbitrary state, s, to any assigned subgoal, $g \in \mathcal{G}$. We separated the learning process into phases for two reasons: (1) to focus on the knowledge extracted from game environments using our approach to unsupervised subgoal discovery during intrinsic motivation learning; (2) to avoid complications that arise in meta-controller learning when the set of subgoals, $\mathcal{G}$, is not fixed. Integrating these phases into a uniform and incremental HRL process is a central apsect of our future work.

4-Room Task with Key and Lock

Consider the task of navigation in the *4-room environment with a key and a lock*, as shown in Figure 2(a). This task was inspired by the *rooms* environment introduced by Sutton, et al. (1999), but it is much more complex. The agent not only needs to learn how to navigate form any arbitrary state to any other state, but also it needs to visit some states in a specific temporal order. At the beginning of each episode, the agent is initialized in an arbitrary location in an arbitrary room. The agent has four possible move actions, $\mathcal{A} = \{North,\ South,\ East,\ West\}$, on each time step. The agent receives $r = +10$ reward for reaching the key and $r = +40$ if it moves to the lock while carrying the key (i.e., any time after visiting the key location during the same episode). Bumping into a wall boundary is punished with a reward of $r = -2$. There is no reward for just exploring the space. Learning in this environment with sparse delayed feedback is challenging for a reinforcement learning agent. To successfully solve the task, the agent should represent knowledge at multiple levels of spatial and temporal abstractions. The agent should also learn to explore the environment efficiently.

We first examined the unsupervised subgoal discovery algorithm over the course of a random walk. The agent was allowed to explore the *4-room* environment for 10,000 episodes. Each episode ended either when the task was completed or after reaching a maximum time step limit of 200. The agent's experiences, $e = (s, a, r, s')$, were collected in an experience memory, $\mathcal{D}$. The stream of external rewards for each transition was used to detect *anomalous* subgoals (Figure 2(c)). We applied a heuristic anomaly detection method for the streaming rewards that was able to differentiate between the rare large positive rewards and the regular small ones. These peaks, as shown in Figure 2(c), corresponded to the experiences in which the key was reached ($r = +10$) or the experience of reaching the lock after obtaining the key.

We also applied a K-means clustering algorithm to the experience memory. (See Algorithm 1.) The centroids of the

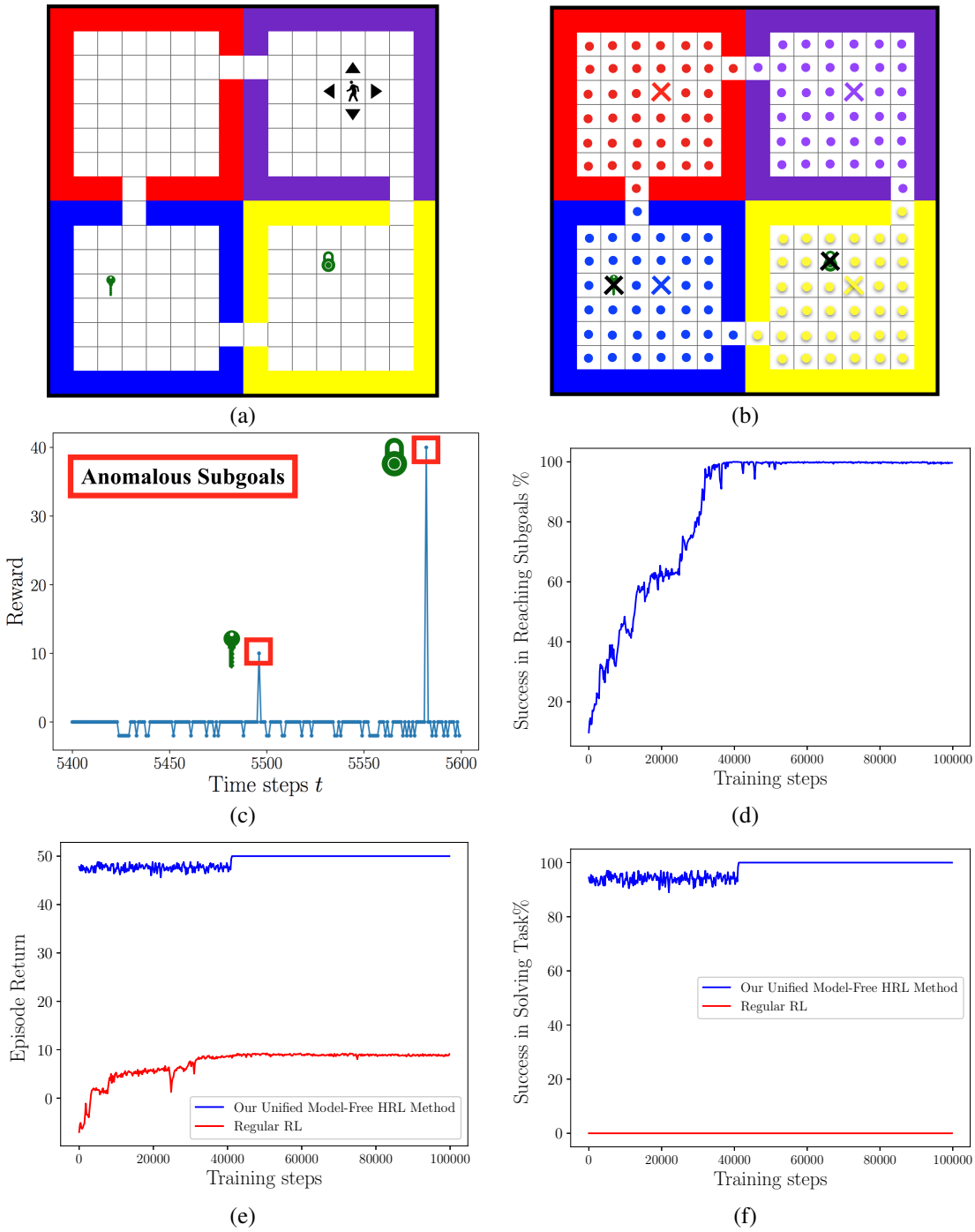

Figure 2: (a) The *4-room* task with a key and a lock. (b) The results of the unsupervised subgoal discovery algorithm with *anomalies* marked with black Xs and *centroids* with colored ones. (c) Reward over an episode, with anomalous points corresponding to the key ($r = +10$) and the lock ($r = +40$). (d) The average success of the controller in reaching subgoals during intrinsic motivation learning in the pretraining phase. (e) The average episode return. (f) The rate of solving the 4-room task.

K-means clusters (with $K = 4$) are plotted in Figure 2(b). We found these centroids to roughly correspond to the centers of the rooms. (We experimented with $K = 8$ and saw equally useful clusters, with each room containing two cluster centroids. We will investigate methods for choosing K in future work.) The clusters, along with the anomalous states, were collected into the set of subgoals.

Phase I learning consisted of 100,000 episodes. Value function approximators were implemented as multi-layer artificial neural networks augmented to encourage the learn-

ing of sparse internal representations of states (Rafati and Noelle 2015). The controller network, $q(s, g, a; w)$, took the state, s, and the goal, g, as inputs. States were presented to the network as Cartesian coordinates, with separate pools of inputs for each of the two dimensions. During this learning phase, the subgoal was initially chosen randomly from the state space. After unsupervised subgoal discovery, the subgoal was chosen randomly from the subgoals set, $\mathcal{G}$. In this way, the controller was trained to navigate from any arbitrary state, s, to any subgoal state. When a centroid was selected as a subgoal, if the agent entered any state in the corresponding cluster, the subgoal was considered attained. Thus, the controller essentially learned how to navigate from any location to any state cluster (*room*) and also to any of the anomalous subgoals (key and door). The learning rate was $\alpha = 0.001$, the discount factor was $\gamma = 0.99$, and the exploration rate was set to $\epsilon = 0.2$. The average success rate (over 10 consecutive episodes) for the first phase of intrinsic motivation learning is shown in Figure 2(d).

Phase II learning involved training both the meta-controller and the controller, together, using the discovered subgoals, $\mathcal{G}$. (See Algorithm 1.) The subgoal set regularly came to contain a total of 6 subgoals: 2 anomalous ones and 4 centroids. The system was trained for 100,000 episodes. The meta-controller's value function approximation network consisted of two layers. The first layer, was a one-hot encoding of the conjunction of the current subgoal and the state, computed by converting the state to the index of the corresponding subgoal. This was connected directly to the output layer. The average return, over 10 consecutive episodes, is shown in Figure 2(e). The agent very quickly converged on the optimal policies and collected the maximum reward ($+50$). The high exploration rate, $\epsilon = 0.2$, caused high stochasticity, but the meta-controller and controller could robustly solve the task on more than 90% of the episodes very early in training. After about 40,000 episodes, the success rate was 100%, as shown in Figure 2(f).

We compared the learning efficiency of our unified HRL method with the performance resulting from training a value approximation network with a regular, non-hierarchical, RL algorithm, TD SARSA (Sutton and Barto 1998). The function approximator that we used for $Q(s, a; w)$ matched that of the controller, equating for computational resources, and we used the same values for the training hyper-parameters. The regular RL agent could only reach the key before becoming stuck in that region, due to the high local reward. Despite the very high exploration rate used, the regular RL agent was not motivated to explore the rest of the state space to reach the lock and solve the task. Results are shown in Figure 2(e) and (f) (red plots).

It is worth noting that this task involves a *partially observable Markov decision process* (POMDP), because information about whether or not the agent has the key is not visible in the state. This hidden state information poses a serious problem for standard RL algorithms, but our HRL agent was able to overcome this obstacle. Through Phase II learning, the hidden information became implicit in the selected subgoal, with the meta-controller changing the current subgoal once the key is obtained. In this way, our HRL framework is able to succeed in task environments that are effectively outside of the scope of standard RL approaches.

Montezuma's Revenge

We applied our HRL approach to the first room of the game *Montezuma's Revenge*. (See Figure 3(a).) The game is well-known as a challenge for RL agents because it requires solving many subtasks while avoiding traps. Having only sparse delayed reward feedback to drive learning makes this RL problem extremely difficult. The agent should learn to navigate the *man* in red to the *key* by: (1) climbing the *middle ladder* (2) climbing the *bottom right ladder* (3) climbing the *bottom left ladder* (4) moving to the key. After picking up the key ($r = +100$), the agent should return back, reversing the previous action sequence, and attempt to reach the *door* ($r = +300$) and exit the room. The moving skull at the bottom of the screen, which ends an episode upon contact, makes obtaining the key extremely difficult. The episode also ends unsuccessfully if the man falls off of a platform.

DeepMind's Deep Q-Learning (DQN) algorithm (Mnih et al. 2015), which surpassed human performance on many ATARI 2600 games, failed to learn this game since the agent did not reach any rewarding state during exploration.

In this problem, the agent requires the skills arising from intrinsic motivation learning in order to explore the environment in a more efficient way (Kulkarni et al. 2016). Our HRL approach supports the learning of such skills. As before, the meta-controller and the controller were trained in two phases. In Phase I, the controller was trained to move the man from any location in the given frame, s, to any other location specified in a subgoal frame, g. An initial set of "interesting" subgoal locations were identified using a custom edge detection algorithm, avoiding empty regions as subgoals. Unsupervised object detection using computer vision algorithms can be challenging (Kulkarni et al. 2016; Fragkiadaki et al. 2015). We made the simplifying assumption that, in many games, edges were suggestive of objects, and the locations of objects made for good initial subgoals. These locations were used in Phase I of training to train the controller through intrinsic motivation. Note that edge detection was only performed to identify Phase I subgoals. Specifically, it was *not* used to change or augment the state representation in any way.

We used a variant of the DQN deep Convolutional Neural Network (CNN) architecture (Figure 3(b)) for approximation of the controller's value function, $q(s, g, a; w)$. The input to the controller network consisted of four consecutive frames of size 84×84, encoding the state, s, and an additional frame binary mask encoding the subgoal, g. The concatenated state and subgoal frames were passed to the network, and the controller then selected one of 18 different joystick actions based on a policy derived from $q(s, g, a; w)$.

During intrinsic motivation learning, the recent experiences were saved in an experience memory, $\mathcal{D}$, with a size of 10^6. In order to support comparison to previously published results, we used the same learning parameters of DeepMind's DQN (Mnih et al. 2015). Specifically, the learning rate, α, was set to to be 0.00025, with a discount rate

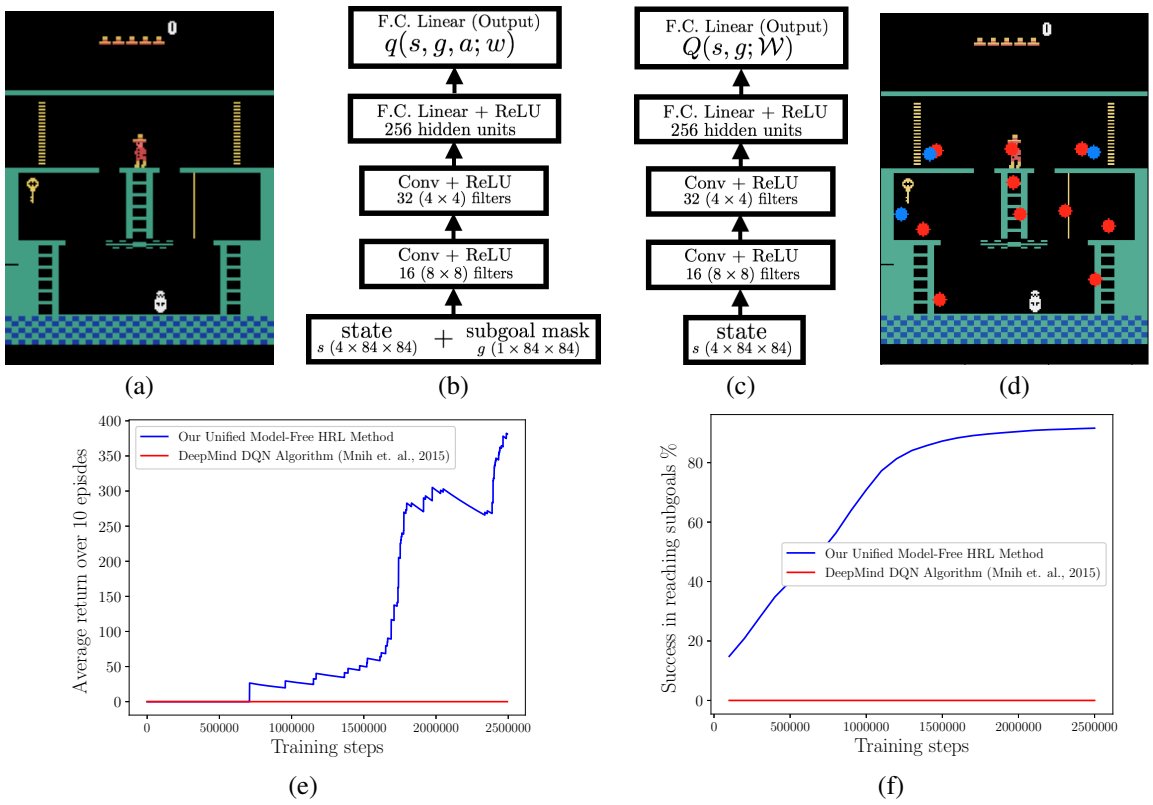

Figure 3: (a) A sample screen from the ATARI 2600 game *Montezuma's Revenge*. (b) The CNN architecture for the controller's value function. (c) The CNN architecture for the meta-controller's value function. (d) The results of the unsupervised subgoal discovery algorithm. The blue circles are the discovered anomalous subgoals and the red ones are the centroid subgoals. (e) The average of return over 10 episodes during the second phase of the learning. (f) The success of the controller in reaching to the subgoals during the second phase of learning.

of $\gamma = 0.99$. During Phase I learning, we trained the network for a total of 2.5×10^6 time steps. The exploration probability parameter, ϵ, decreased from 1.0 to 0.1 in the first million steps and remained fixed after that. After every $100,000$ time steps, we applied our unsupervised subgoal discovery method to the contents of the experience memory in order to find new subgoals, both anomalies and centroids, using K-means clustering with $K = 10$. As shown in Figure 3(d), the unsupervised learning algorithm managed to discover the location of the key and the doors in this way. It also identified useful objects such as ladders, platforms, and the rope. In Phase II, we trained the meta-controller and the controller jointly. We used an architecture based on the DQN CNN (Mnih et al. 2015), as shown in Figure 3(c), for the meta-controller's value function, $Q(s, g; \mathcal{W})$. We used the non-overlapping discovered subgoals, which resulted in a set of 11 subgoals, $\mathcal{G}$. At the beginning of each episode, the meta-controller assigned a subgoal, $g \in \mathcal{G}$, based on an epsilon-greedy policy derived from $Q(s, g; \mathcal{W})$. The controller then attempted to reach these subgoals. The controller's experience memory, $\mathcal{D}_1$, had a size of 10^6, and the size of the meta-controller's experience memory, $\mathcal{D}_2$, was 5×10^4. The cumulative rewards for the game episodes is shown in Figure 3(e). After about 1.5 million time steps, the controller managed to reach the key subgoal more frequently. The success of the intrinsic motivation learning is depicted in Figure 3(f). At the end of the second phase of learning (after 2.5 million learning steps), the meta-controller regularly chose the proper subgoals for collecting the maximum reward ($+400$).

Conclusion

We have proposed and demonstrated a novel model-free HRL method for subgoal discovery using unsupervised learning over a small memory of the most recent experiences (trajectories) of the agent. When combined with an intrinsic motivation learning mechanism, this method learns subgoals and skills together, based on experiences in the environment. Thus, we offer an HRL approach that does not require a model of the environment, making it suitable for larger-scale applications.

Acknowledgments

We acknowledge the valuable comments of the anonymous reviewers of this paper. The computations are conducted on the MERCED cluster at UC Merced, which was funded by National Science Foundation Grant No. ACI-1429783.

References

Şimşek, O.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, 816–823.

Fragkiadaki, K.; Arbelaez, P.; Felsen, P.; and Malik, J. 2015. Learning to segment moving objects in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4083 – 4090.

Goel, S., and Huber, M. 2003. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *FLAIRS Conference*, 346–350. AAAI Press.

Hodge, V. J., and Austin, J. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22(2):85–126.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 3675–3683.

Liang, Y.; Machado, M. C.; Talvitie, E.; and Bowling, M. H. 2016. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 17–25.

Machado, M. C., and Bowling, M. 2016. Learning Purposeful Behaviour in the Absence of Rewards. Presented at the ICML-16 Workshop on Abstraction in Reinforcement Learning.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. H. 2017. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2295–2304.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Pickett, M., and Barto, A. G. 2002. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 506–513.

Rafati, J., and Noelle, D. C. 2015. Lateral inhibition overcomes limits of temporal difference learning. In *37th Annual Cognitive Science Society Meeting, Pasadena, CA, USA*.

Rafati, J., and Noelle, D. C. 2017. Sparse coding of learned state representations in reinforcement learning. In *Conference on Cognitive Computational Neuroscience, New York City, NY, USA*.

Singh, S.; Lewis, R. L.; Barto, A. G.; and Sorg, J. 2010. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transaction on Autonomous Mental Development* 2(2):70–82.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1st edition.

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

Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM* 38(3).

Thrun, S., and Schwartz, A. 1995. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press. 385–392.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. *CoRR* abs/1703.01161.