ELSIM: End-to-end learning of reusable skills through intrinsic motivation

Arthur Aubret[1] **Laetitia Matignon**[1] **Salima Hassas**[1]

Abstract

Taking inspiration from developmental learning, we present a novel reinforcement learning architecture which hierarchically learns and represents self-generated skills in an end-to-end way. With this architecture, an agent focuses only on task-rewarded skills while keeping the learning process of skills bottom-up. This bottom-up approach allows to learn skills that 1- are transferable across tasks, 2- improves exploration when rewards are sparse. To do so, we combine a previously defined mutual information objective with a novel curriculum learning algorithm, creating an unlimited and explorable tree of skills. We test our agent on simple gridworld environments to understand and visualize how the agent distinguishes between its skills. Then we show that our approach can scale on more difficult MuJoCo environments in which our agent is able to build a representation of skills which improve over a baseline both transfer learning and exploration when rewards are sparse.

1. Introduction

In reinforcement learning (RL), an agent learns by trial-and-error to maximize the expected rewards obtained from actions performed in its environment (Sutton & Barto, 1998). However, many RL agents usually strive to achieve one goal using only low-level actions. In contrast, as human being, when we want to go to work, we do not think about every muscle we contract in order to move; we just take abstract decisions such as *Go to work*. Low-level behaviors such as how to walk are already learned and we do not need to think about them. Learning to walk is a classical example of babies developmental learning, which refers to the ability of an agent to spontaneously explore its environment and acquire new skills (Barto, 2013). Babies do not try to get walking behaviors all at once, but rather first learn to move their legs, to crawl, to stand up, and then, eventually, to walk. They are **intrinsically motivated** since they act for the inherent satisfaction of learning new skills (Ryan & Deci, 2000) rather than for an **extrinsic reward** assigned by the environment.

Several works are interested in learning abstract actions, also named **skills** or **options** (Sutton et al., 1999), in the framework of deep reinforcement learning (DRL) (Aubret et al., 2019). Skills can be learned with extrinsic rewards (Bacon et al., 2017), which facilitates the credit assignment (Sutton et al., 1999). In contrast, if one learns skills with intrinsic motivation, the learning process becomes bottom-up (Machado et al., 2017), i.e. the agent learns skills before getting extrinsic rewards. When learning is bottom-up, the agent commits to a time-extended skill and avoids the usual wanderlust due to the lack, or the sparsity, of extrinsic rewards. Therefore, it can significantly improve exploration (Machado & Bowling, 2016; Nachum et al., 2019b). In addition, these skills can be used for different tasks, emphasizing their potential for transfer learning (Taylor & Stone, 2009). These properties make intrinsic motivation attractive in a *continual learning* framework, which is the ability of the agent to acquire, retain and reuse its knowledge over a lifetime (Thrun, 1996).

Several works recently proposed to intrinsically learn such skills using a diversity heuristic (Eysenbach et al., 2019; Achiam et al., 2018), such that different states are covered by the learned skills. Yet several issues remain: 1- the agent is often limited in the number of learned skills or requires *curriculum learning* (Achiam et al., 2018); 2- most skills target uninteresting parts of the environment relatively to some tasks; thereby it requires prior knowledge about which features to diversify (Eysenbach et al., 2019); 3- the agent suffers from catastrophic forgetting when it tries to learn a task while learning skills (Eysenbach et al., 2019); 4- discrete time-extended skills used in a hierarchical setting are often sub-optimal for a task. With diversity heuristic, skills are indeed not expressive enough to efficiently target a goal (Eysenbach et al., 2019; Achiam et al., 2018).

In this paper, we propose to address these four issues so as to **improve the approaches for continually learning increasingly difficult skills with diversity heuristics**. We introduce ELSIM (End-to-ended Learning of reusable Skills through Intrinsic Motivation), a method for learning rep-

[1]Univ Lyon, Université Lyon 1, CNRS, LIRIS F-69622, Villeurbanne, France. Correspondence to: Arthur Aubret <arthur.aubret@liris.cnrs.fr>.

resentations of skills in a bottom-up way. The agent autonomously builds a tree of abstract skills where each skill is a refinement of its parent. First of all, skills are learned independently from the tasks but along with tasks; it guarantees they can be easily transferred to other tasks and may help the agent to explore its environment. We use the optimization function defined of DIAYN (Eysenbach et al., 2019) which guarantees that states targeted by a skill are close to each other. Secondly, the agent selects a skill to refine with extrinsic or intrinsic rewards, and learns new subskills; it ensures that the agent learns specific skills useful for tasks through an intelligent *curriculum*, among millions of possible skills.

Our approach contrasts with existing approaches which either bias skills towards a task (Bacon et al., 2017), reducing the possibilities for transfer learning, or learn skills during pretraining (Eysenbach et al., 2019). We believe our paradigm, by removing the requirement of a *developmental period* (Metzen & Kirchner, 2013) (which is just an unsupervised pretraining), makes naturally compatible developmental learning and *lifelong learning*. Therefore, we emphasize three properties of our ELSIM method. 1-**Learning is bottom-up**: the agent does not require an expert supervision to expand the set of skills. It can use its skills to solve different sequentially presented tasks or to explore its environment. 2- **Learning is end-to-end**: the agent never stops training and keeps expanding its tree of skills. It gradually self-improves and avoids catastrophic forgetting. 3- **Learning is focused**: the agent only learns skills useful for its high-level extrinsic/intrinsic objectives when provided.

Our contributions are the following: we introduce a new curriculum algorithm based on an adaptation of diversity-based skill learning methods. Our objective is not to be competitive when the agent learns one specific goal, but **to learn useful and reusable skills along with sequentially presented goals in an end-to-end fashion**. We show experimentally that ELSIM achieves **good asymptotic performance** on several single-task benchmarks, **improves exploration** over standard DRL algorithms and manages to easily **reuse its skills**. Thus, this is a step towards *lifelong learning* agents.

This paper is organized as follows. First, we introduce the concepts used in ELSIM, especially diversity-based intrinsic motivation (Section 2). In Section 3, the core of our method is presented. Then, we explain and visualize how ELSIM works on simple gridworlds and compare its performances with state-of-the-art DRL algorithms on single and sequentially presented tasks learning (Section 4). In Section 5, we detail how ELSIM relates to existing works. Finally, in Section 6, we take a step back and discuss ELSIM. Pseudo-codes, full experiments, additional details and hyper-parameters can be found in the long version of the paper [1].

2. Background

2.1. Reinforcement learning

A Markov decision process (MDP) (Puterman, 2014) is defined by a set of possible states S; a set of possible actions A; a transition function $P : S \times A \times S \rightarrow \mathbb{P}(s'|s,a)$ with $a \in A$ and $s, s' \in S$; a reward function $R : S \times A \times S \rightarrow \mathbb{R}$; the initial distribution of states $\rho_0 : S \rightarrow [0;1]$. A stochastic policy π maps states to probabilities over actions in order to maximize the discounted cumulative reward defined by $\varsigma_t = [\sum_{t=0}^{\infty} \gamma^t r_t]$ where $\gamma \in [0,1]$ is the discount factor. In order to find the action maximizing ς in a state s, it is common to maximize the expected discounted gain following a policy π from a state-action tuple defined by:

$$Q_\pi(s,a) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t) \\ s_{t+1} \sim P(s_{t+1}|s_t,a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right] \tag{1}$$

where $s_0 = s$, $a_0 = a$. To compute this value, it is possible to use the Bellman Equation (Sutton & Barto, 1998).

2.2. Obtaining diverse skills through mutual information objective

We characterize a **skill** by its intra-skill policy; thereby a skill is a mapping of states to probabilities over actions. One way to learn, without extrinsic rewards, a set of different skills along with their intra-skill policies is to use an objective based on mutual information (MI).

In (Eysenbach et al., 2019), learned skills should be as **diverse** as possible (different skills should visit different states) and **distinguishable** (it should be possible to infer the skill from the states visited by the intra-skill policy). It follows that the learning process is 4-step with two learning parts (Eysenbach et al., 2019): 1- the agent samples one skill from an uniform distribution; 2- the agent executes the skill by following the corresponding intra-skill policy (randomly initialized); 3- a discriminator learns to categorize the resulting states to the assigned skill; 4- at the same time, these approximations reward intra-skill policies (cf. eq. 5).

The global objective can be formalized as maximizing the MI between the set of skills G and states S' visited by intra-skill policies, defined by (Gregor et al., 2017):

$$I(G;S') = \mathbb{H}(G) - \mathbb{H}(G|S') \tag{2}$$

$$= \mathop{\mathbb{E}}_{\substack{g \sim p(g) \\ s' \sim p(s'|\pi_\theta^g, s)}} [\log p(g|s') - \log p(g)] \tag{3}$$

[1]https://arxiv.org/abs/2006.12903

where π_θ^g is the intra-skill policy of $g \in G$ and is parameterized by θ; $p(g)$ is the distribution of skills the agent samples on; and $p(g|s')$ is the probability to infer g knowing the next state s' and intra-skill policies. This MI quantifies the reduction in the uncertainty of G due to the knowledge of S'. By maximizing it, states visited by an intra-skill policy have to be informative of the given skill.

A bound on the MI can be used as an approximation to avoid the difficulty to compute $p(g|s')$(Barber & Agakov, 2003; Gregor et al., 2017) :

$$I(G, S') \geq \mathbb{E}_{\substack{g\sim p(g) \\ s'\sim p(s'|\pi_\theta^g, s)}} [\log q_\omega(g|s') - \log p(g)] \quad (4)$$

where $q_\omega(g|s')$ is the **discriminator** approximating $p(g|s')$. In our case, the discriminator is a neural network parameterized by ω. q_ω minimizes the standard cross-entropy $-\mathbb{E}_{g\sim p(g|s')} \log q_\omega(g|s')$ where $s' \sim \pi_\theta^g$.

To discover skills, it is more efficient to set $p(g)$ to be uniform as it maximizes the entropy of G (Eysenbach et al., 2019). Using the uniform distribution, $\log p(g)$ is constant and can be removed from eq. 4. It follows that one can maximize eq. 4 using an intrinsic reward to learn the intra-skill policy of a skill $g \in G$ (Eysenbach et al., 2019):

$$r^g(s') = \log q_\omega(g|s'). \quad (5)$$

Similarly to (Eysenbach et al., 2019), we use an additional entropy term to encourage the diversity of covered states. In practice, this bonus is maximized through the use of DRL algorithms: Soft Actor Critic (SAC) (Haarnoja et al., 2018) for continuous action space and Deep Q network (DQN) with Boltzmann exploration (Mnih et al., 2015) for discrete one.

3. Method

In this section, we first give an overview of our method and then detail the building of the tree of skills, the learning of the skill policy, the selection of the skill to refine and how ELSIM integrates this in an end-to-end framework.

3.1. Overview: building a tree of skills

To get both bottom-up skills and interesting skills relatively to some tasks, our agent has to choose the skills to improve thanks to the extrinsic rewards, but we want that our agent improves its skills without extrinsic rewards. The agent starts by learning a discrete set of diverse and distinguishable skills using the method presented in Section 2.2. Once the agent clearly distinguishes these skills using the covered skill-conditioned states with its discriminator, it splits them into new sub-skills. For instance, for a creature provided with proprioceptive data, a *moving forward* skill could be separated into *running* and *walking*. The agent only trains on sub-skills for which the parent skill is useful for the global task. Thus it incrementally refines the skills it needs to accomplish its current task. If the agent strives to sprint, it will select the skill that provides the greater speed. The agent repeats the splitting procedure until its intra-skill policy either reach the maximum number of splits or become too deterministic to be refined.

The **hierarchy of skills** is maintained using a tree where each node refers to an abstract skill that has been split and each leaf is a skill being learned. We formalize the hierarchy using sequence of letters where a letter's value is assigned to each node:

- The set of skills G is the set of leaf nodes. A skill $g \in G$ is represented by a sequence of $k+1$ letters : $g = (l^0, l^1, ..., l^k)$. When g is split, a letter is added to the sequence of its new sub-skills. For instance, the skill $g = (l^0 = 0, l^1 = 1)$ can be split into two sub-skills $(l^0 = 0, l^1 = 1, l^2 = 0)$ and $(l^0 = 0, l^1 = 1, l^2 = 1)$.
- The vocabulary V refers to the values which can be assigned to a letter. For example, to refine a skill into 4 sub-skills, we should define $V = \{0, 1, 2, 3\}$.
- The length $L(g)$ of a skill is the number of letters it contains. Note that the length of a skill is always larger than its parent's.
- $l^{:k}$ is the sequence of letters preceding l^k (excluded).

We use two kind of policies: the first are the **intra-skill policies**. The learning of these intra-skill policies is described in Section 3.2. The second type of policy is task-dependent and responsible to choose which skill to execute; we call it the **tree-policy** (see Section 3.3).

3.2. Learning intra-skill policies

In this section, we detail how intra-skill policies are learned. We adapt the method presented in Section 2.2 to our hierarchical skills context. Two processes are simultaneously trained to obtain diverse skills: the intra-skill policies learn to maximize the intrinsic reward (cf. eq. 5), which requires to learn a discriminator $q_\omega(g|s')$. Given our hierarchic skills, we can formulate the probability inferred by the discriminator as a product of the probabilities of achieving each letter of g knowing the sequence of preceding letters, by applying the chain rule:

$$\begin{aligned} r^g(s') &= \log q_\omega(g|s') = \log q_\omega(l^0, l^1, \ldots, l^k|s') \\ &= \log \prod_{i=0}^{k} q_\omega(l^i|s', l^{:i}) = \sum_{i=0}^{k} \log q_\omega(l^i|s', l^{:i}). \end{aligned} \quad (6)$$

Gathering this value is difficult and requires an efficient discriminator q_ω. As it will be explained in Section 3.4, in practice, we use **one different discriminator for each node** of our tree: $\forall i,\ q_\omega(l^i|s',l^{:i}) \equiv q_\omega^{:i}(l^i|s')$.

For instance, if $|V|=2$, one discriminator $q_\omega^{\emptyset}$ will be used to discriminate $(l^0=0)$ and $(l^0=1)$ but an other one, $q_\omega^{l^0=0}$ will discriminate $(l^0=0,l^1=0)$ and $(l^0=0,l^1=1)$.

It would be difficult for the discriminators to learn over all letters at once; the agent would gather states for several inter-level discriminators at the same time and a discriminator would not know which part of the gathered states it should focus on. This is due to the fact that discriminators and intra-skill policies simultaneously train. Furthermore, there are millions of possible combinations of letters when the maximum size of sequence is large. We do not want to learn them all. To address these issues, we introduce **a new curriculum learning algorithm that refines a skill only when it is distinguishable**. When discriminators successfully learn, they progressively extends the sequence of letters; in fact, we split a skill (add a letter) only when its discriminator has managed to discriminate the values of its letter. Let's define the following probability:

$$p_{finish}^{:k}(l^k) = \mathbb{E}_{s_{final}\sim\pi^{:k+1}}\left[q_\omega^{:k}(l^k|s_{final})\right]. \quad (7)$$

where s_{final} is the state reached by the intra-skill policy at the last timestep. We assume the discriminator $q_\omega^{:k}$ has finished to learn when: $\forall v \in V,\ p_{finish}^{:k}(l^k=v) \geq \delta$ where $\delta \in [0,1]$ is an hyperparameter. Choosing a δ close to 1 ensures that the skill is learned, but an intra-skill policy always explores, thereby it may never reach an average probability of exactly 1; we found empirically that 0.9 works well.

To approximate eq. 7 for each letters' value v, we use an exponential moving average $p_{finish}^{:k}(l^k=v) = (1-\beta)p_{finish}^{:k}(l^k=v) + \beta q_\omega^{:k}(l^k=v|s_{final})$ where $s_{final} \sim \pi^{:k+1}$ and $\beta \in [0;1]$. Since we use buffers of interactions (see Section 3.4), we entirely refill the buffer before the split.

Let us reconsider eq. 6. $\sum_{i=0}^{k-1}\log q_\omega(l^i|s',l^{:i})$ is the part of the reward assigned by the previously learned discriminators. It forces the skill to stay close to the states of its parent skills since this part of the reward is common to all the rewards of its parent skills. In contrast, $\log q_\omega(l^k|s',l^{:k})$ is the reward assigned by the discriminator that actively learns a new discrimination of the state space. Since the agent is constrained to stay inside the area of previous discriminators, the new discrimination is **uncorrelated** from previous parent discriminations. In practice, we increase the importance of previous discriminations with a hyper-parameter $\alpha \in \mathbb{R}$:

$$r^g(s') = \log q_\omega(l^k|s',l^{:k}) + \alpha\sum_{i=0}^{k-1}\log q_\omega(l^i|s',l^{:i}). \quad (8)$$

This hyper-parameter is important to prevent the agent to deviate from previously discriminated areas to learn more easily the new discrimination.

3.3. Learning which skill to execute and train

For each global objective, a stochastic policy, called *tree-policy* and noted π_T (with T the tree of skills), is responsible to choose the skill to train by navigating inside the tree at the beginning of a task-episode. This choice is critical in our setting: while expanding its tree of skills, the agent cannot learn to discriminate every leaf skill at the same time since discriminators need states resulting from the intra-skill policies. We propose to **choose the skill to refine according to its benefit in getting an other reward** (extrinsic or intrinsic), thereby ELSIM executes and learns only interesting skills (relatively to an additional reward).

To learn the *tree-policy*, we propose to model the tree of skills as an MDP solved with a Q-learning and Boltzmann exploration. The action space is the vocabulary V; the state space is the set of nodes, which include abstract and actual skills; the deterministic transition function is the next node selection; if the node is not a leaf, the reward function R_T is 0, else this is the discounted reward of the intra-skill policy executed in the environment divided by the maximal episode length. Each episode starts with the initial state as the root of the tree, the *tree-policy* selects the next nodes using Q-values. Each episode ends when a leaf node has been chosen, i.e. a skill for which all its letters has been selected; the last node is always chosen uniformly (see Section 3.4).

Let us roll out an example using the *tree-policy* displayed in Figure 1. The episode starts at the root of the tree; the *tree-policy* samples the first letter, for example it selects $l^0=0$. Until it reaches a leaf-node, it samples new letters, e.g. $l^1=1$ and $l^2=0$. The *tree-policy* has reached a leaf, thereby it will execute and learn the skill $(0,1,0)$. Then, the state-action tuple $((0,1),(0))$ is rewarded with the scaled discounted reward of the task. This reward is propagated via the Q-learning update to previous state-action tuples $((\emptyset),(0))$ and $((0),(1))$ to orientate the *tree-policy* to $(0,1,0)$.

The MDP evolves during the learning process since new letters are progressively added. The Q-values of new skills are initialized with their parent Q-values. However, eq. 6 ensures that adding letters at the leaf of the tree monotonically increases Q-values of their parent nodes. The intuition is that, when splitting a skill, at least one of the child is equal or better than the skill of its parent relatively to the task.

We experimentally show this in Section 4.2. The resulting curriculum can be summarized as follows: the tree will be small at the beginning, and will grow larger in the direction of feedbacks of the environment.

We now sum up the process of the *tree-policy*: 1-an agent runs an episode inside the MDP of skills; the sequence of actions represents a skill; 2- the agent executes the intra-skill policy of the skill; 3- the *tree-policy* is rewarded according to how well the intra-skill policy fits the task and the Q-learning applies.

3.4. Simultaneous training of the *tree-policy* and intra-skill policies

The MI objective requires the skill distribution to remain uniform (cf. eq. 5), however that is not our case: the agent strives to avoid some useless skills while focusing on others. In our preliminary experiments, ignoring this leads us to catastrophic forgetting of the learned skills since discriminators forget how to categorize states of the skills they never learn on. To bypass this issue and sample uniformly, we assign to each node i of our tree a replay buffer containing interactions of the intra-skill policy with the environment, a RL algorithm and a discriminator ($q_\omega^{:i}$). At each split, intra-skill policies and buffers of a node are copied to its children; for the first node, its intra-skill policies are randomly initialized and its buffer is empty.

This way, the entire training is off-policy: the intra-skill policy fills the replay buffer while the discriminator and intra-skill policies learn from the interactions that are uniformly extracted from their buffers. We split the lifetime of a node into two phases: 1-the **learning phase** during which next letter's values are sampled uniformly; the *tree-policy* is uniform at this node; 2-the **exploitation phase** during which the *tree-policy* chooses letters with its Boltzmann policy (Section 3.3).

Then, at each step, the agent runs the *tree-policy* to select the discriminator in the learning phase that will learn. The discriminator samples a mini-batch of data from its children's (all leaves) buffers and learns on it. Then, all children intra-skill policies learn from the intrinsic feedback of the same interactions, output by the selected discriminator and all its parents according to eq. 6.

Once a node enters the exploitation phase, an hyper-parameter η regulates the probability that each parent's discriminator learn on its children data. Their learning interactions are recursively sampled uniformly on their children. This post-exploration learning allows a node to expand its high-reward area. Without this mechanism, different uncovered states of the desired behaviour may be definitively attributed to different fuzzy skills, as shown in Section 4.1.

Figure 1 gives an example of a potential tree and how different phases coexist; the skills starting by $(0, 1)$ seem to be the most interesting for the task since each letter sampling probability is high. Skills $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0)$ and $(1, 1)$ are being learned, therefore the sampling probability of their last values is uniform.

4. Experiments

The first objective of this section is to study the behavior of our ELSIM algorithm on basic gridworlds to make the visualization easier. The second purpose is to show that ELSIM can scale with high-dimensional environments. We also compare its performance with a non-hierarchical algorithm SAC (Haarnoja et al., 2018) in a single task setting. Finally we show the potential of ELSIM for transfer learning.

4.1. Study of ELSIM in gridworlds

In this section, we analyze how skills are refined on simple gridworlds adapted from gym-minigrid (Chevalier-Boisvert & Willems, 2018). Unless otherwise stated, **there is no particular task (or extrinsic reward)**, thereby the *tree-policy* is uniform. The observations of the agent are its coordinates; its actions are the movements into the four cardinal directions. To maximize the entropy of the intra-skill policy with a discrete action space, we use the DQN algorithm (Mnih et al., 2015). The agent starts an episode at the position of the arrow (see figures) and an episode resets every 100 steps, thus the intra-skill policy lasts 100 steps. At the end of the training phase, the skills of all the nodes are evaluated through an evaluation phase lasting 500 steps for each skill. In all figures, each tile corresponds to a skill with $|V| = 4$ that is displayed at the top-left of the tile. Figure 2 and 4 display the density of the states visited by intra-skill policies during the evaluation phase: the more red the state, the more the agent goes over it.

Do the split of skills improve the exploration of an agent ? Figure 2 shows some skills learned in an environment of 4 rooms separated by a bottleneck. We first notice that the agent clearly separates its first skills $(0), (1), (2), (3)$ since the states covered by one skill are distinct from the states of the other skills. However it does not escape from its starting room when it learns these first skills. When it develops the skills close to bottlenecks, it learns to go beyond and invests new rooms. It is clear for skills (1) and (2) which, with one refinement, respectively explore the top-right (skill $(1, 0)$) and bottom-left (skills $(2, 0), (2, 2), (2, 3)$) rooms. With a second refinement, $(2, 3, 0)$ even manages to reach the farthest room (bottom-right). This stresses out that the refinement of a skill also allows to expand the states covered by the skill, and thus can improve the exploration of an agent when the rewards are sparse in an environment.

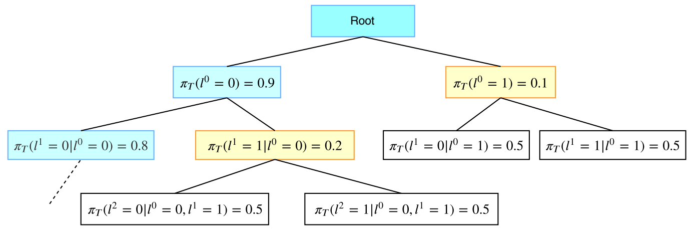
Root
$\pi_T(l^0 = 0) = 0.9$
$\pi_T(l^0 = 1) = 0.1$
$\pi_T(l^1 = 0|l^0 = 0) = 0.8$
$\pi_T(l^1 = 1|l^0 = 0) = 0.2$
$\pi_T(l^1 = 0|l^0 = 1) = 0.5$
$\pi_T(l^1 = 1|l^0 = 1) = 0.5$
$\pi_T(l^2 = 0|l^0 = 0, l^1 = 1) = 0.5$
$\pi_T(l^2 = 1|l^0 = 0, l^1 = 1) = 0.5$

Figure 1. Representation of a part of the tree of skills with $|V| = 2$ and the value of *tree-policy* in each node. White nodes are actual leaves of the tree; the discriminator is inactive. Yellow nodes represent nodes for which the discriminator can not differentiate its sub-skills; the *tree-policy* samples uniformly. Nodes are blue when the discriminator can distinguish its sub-skills; the *tree-policy* samples using Q-values.

Do the split of skills correct a wrong over-generalization of a parent skill ? Figure 3 shows the evolution of the intrinsic reward function for some skills . The environment contains a vertical wall and settings are the same as before, except the Boltzmann parameter set to 0.5. At the beginning, skill (1) is rewarding identically left and right sides of the wall. This is due to the generalization over coordinates and to the fact that the agent has not yet visited the right side of the wall. However it is a wrong generalization because left and right sides are not close to each other (considering actions). After training, when the agent begins to reach the right side through the skill $(3, 2)$, it corrects this wrong generalization. The reward functions better capture the distance (in actions) between two states: states on the right side of the wall are attributed to skill (3) rather than (1). We can note that other parts of the reward function remain identical.

Can the agent choose which skill to develop as a priority ? In this part, we use the same environment as previously, but states on the right side of the wall give an extrinsic reward of 1. Thus the agent follows the *tree-policy* to maximize its rewards, using Boltzmann exploration, and focus its refinement on rewarding skills. Figure 4 shows all the parent skills of the most refined skill which reaches $L(g) = 6$. The agent learns more specialized skills in the rewarding area than when no reward is provided .

Summary. We illustrated the following properties of ELSIM: 1- it expands a previously learned rewarding area when it discovers new states; we show in Section 4.2 that it improves exploration when the rewards are sparse; 2- adding letters corrects over-generalization of their parent discriminator; 3- it can focus the skill expansion towards task-interesting areas.

4.2. Performance on a single task

In this part, we study the ability of ELSIM to be competitive on high-dimensional benchmarks [2] **without any prior knowledge**. For ELSIM, we set the maximum skill length to 10, which is reached in *HalfCheetah*.

Figure 5 respectively shows the average reward per episode for different environments. Shaded areas color are upper-bounded (resp. lower-bounded) by the maximal (resp. minimal) average reward.

First, the *MountainCarContinuous* environment represents a challenge for the exploration as it is a sparse reward environment: the agent receives the reward only when it reaches the goal. In this environment, ELSIM outperforms SAC by getting a higher average reward. It confirms our results (cf. Section 4.1) on **the positive impact of ELSIM on the exploration**. There is a slight decrease after reaching an optima, in fact, ELSIM keeps discovering skills after finding its optimal skill. On *Pendulum* and *LunarLander*, ELSIM achieves the same asymptotic average reward than SAC, even though ELSIM may require more timesteps. On *HalfCheetah*, SAC is on average better than ELSIM. However we emphasize that **ELSIM also learns other skills**. For example in *HalfCheetah*, ELSIM learns to walk and flip while SAC, that is a non-hierarchical algorithm, only learns to sprint.

4.3. Transfer learning

In this section, we evaluate the interest of ELSIM for transfer learning. We take skills learned by intra-policies in Section 4.2, reset the *tree-policy* and restart the learning process on *HalfCheetah* and *HalfCheetah-Walk*. *HalfCheetah-Walk* is a slight modification of *HalfCheetah* which makes the agent target a speed of 2 . Intra-skill policies learning was stopped in *HalfCheetah*.

The same parameters as before are used, but we use MBIE-

[2] *HalfCheetah* has a state space and action space respectively of 17 and 6 dimensions.

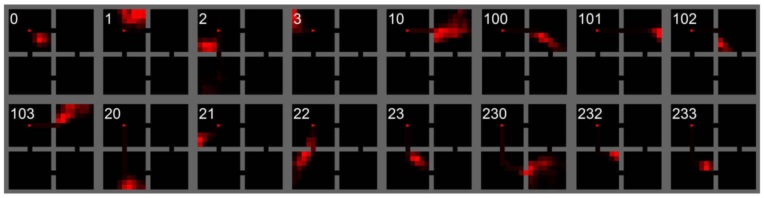

Figure 2. Some skills learned by the agent in an environment composed of four rooms.

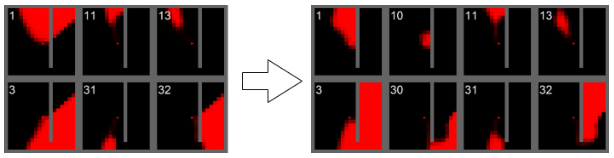

Figure 3. Discriminator's probability of achieving skills (1), (3) and their sub-skills in every state (*i.e.* $q(g|s)$). The more red the state, the more rewarding it is for the skill. The left side corresponds to the preliminary stage of the learning process (timestep 128.10^4); the right side corresponds to the end of the learning process (timestep 640.10^4).

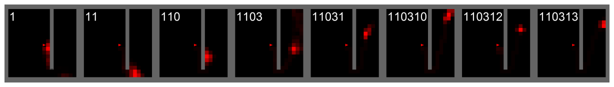

Figure 4. One path in the tree of skills learned by the agent with an extrinsic reward of 1 on the upper right side of the wall.

EB (Strehl & Littman, 2008) to explore the tree . Figure 6 shows that the *tree-policy* learns to reuse its previously learned skills on *HalfCheetah* since it almost achieves the same average reward as in Figure 5. On *HalfCheetah-Walk*, we clearly see that the agent has already learned skills to walk and that it easily retrieves them. In both environments, ELSIM learns faster than SAC, which learn from scratch. It demonstrates that skills learned by ELSIM can be used for **other tasks** than the one it has originally been trained on.

5. Related work

Intrinsic motivation in RL is mostly used to improve exploration when rewards are sparse or to learn skills (Aubret et al., 2019). The works that learn skills with intrinsic rewards are close to our approach and can be classified in two major categories. The first category strives to explicitly target states. The reward is defined either as a distance between agent's state and its goal state (Levy et al., 2019), or as the difference between the change in the state space and the required change (Nachum et al., 2018). However, to efficiently guide the agent, the reward functions require a good state representation (Nair et al., 2018; Nachum et al., 2019a).

Intrinsic motivation as diversity heuristic. Our work mostly falls into this category, which strives to define a skill based on a MI objective (cf. Section 2.2). Seminal works already learn a discrete set of diverse skills (Florensa et al., 2017; Eysenbach et al., 2019). In contrast to them, we manage to learn both the skill and the skill-selection policy in an end-to-end way and we propose an efficient way to learn a large number of skills useful for the tasks the agent wants to accomplish. Recent works (Warde-Farley et al., 2019; Co-Reyes et al., 2018) try to learn a continuous embedding of skills, but do not integrate their work into an end-to-end hierarchical learning agent. DADS (Sharma et al., 2019) learn skills using a generative model over observations rather than over skills. While this is efficient on environments with simple observation space, this is computationally ineffective. In our work, rather than learning a continuous skill embedding, we strive to select and learn skills among a very large number of discretized skills. As a consequence, we focus our learned skill distribution only on task-interesting skills and we do not rely on a parametric state distribution. **Continual learning.** Other works proposed a *lifelong learning* architecture. Some assume that skills are already learned and learn to reuse them; for example H-DRLN (Tessler et al., 2017) uses a hierarchical policy to choose between ground actions and skills. They also propose to distill previously learned skills into a larger architecture, making their approach scalable. In contrast, we tackle the problem of learning skills in an end-to-end fashion, thereby our approach may be compatible. Similarly to us, CCSA (Kompella et al., 2017) addresses the catastrophic forgetting problem by freezing the learning of some experts. They mix two unsupervised learning methods to find and represent goal states, and then learn to reach them. However, their unsupervised algorithm only extracts linear features and they manually define a first set of skills. One particular aspect of continual learning is Meta-RL: how can an agent learn how to learn ? Traditional methods assume there exists a task distributions and try to generalize over it (Finn et al., 2017; Duan et al., 2016); this task distribution serves as prior knowledge. In (Gupta et al., 2018), the authors address this issue and apply MAML (Finn et al., 2017) on an uniform distribution of tasks learned by DIAYN (Eysenbach et al., 2019). However, learning is neither focused, nor end-to-end. In the continuity of this work, CARML (Jabri et al., 2019) mixes the objective of DADS (Sharma

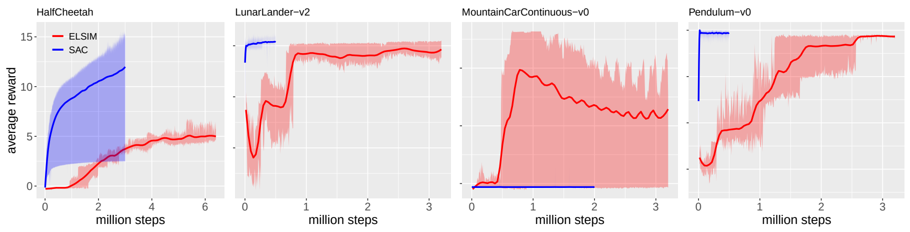

Figure 5. Average reward per episode in classical environments (*HalfCheetah-v2* (Todorov et al., 2012), *LunarLanderContinuous-v0* (Shariff & Dick, 2013), *MountainCarContinuous-v0* (Moore, 1990) and *Pendulum-v0*) for SAC and ELSIM (averaged over 4 seeds). We use our own implementation of SAC except for *HalfCheetah* for which the blue curve is the average reward of SAC on 5 seeds, taken from (Haarnoja et al., 2018). We stopped the simulation after convergence of SAC.

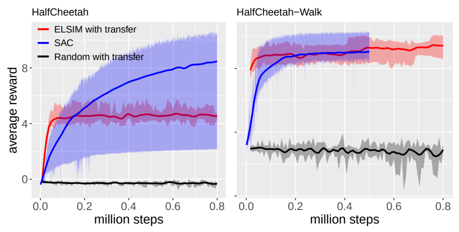

Figure 6. Average reward per episode in *HalfCheetah* and *HalfCheetah-Walk*. We use our own implementation of SAC for *HalfCheetah-Walk*. The black curve is the average reward of a random *tree-policy* that uses the transfered skills.

et al., 2019) and Meta-RL; it alternates between generating trajectories of the distribution of tasks and fitting the task distribution to new trajectories. While CARML discovers diverse behaviors with pixel-level state space, it cannot learn a global objective end-to-end like ELSIM.

State abstraction. Our method can be viewed as a way to perform state abstraction (Li et al., 2006). Rather than using this abstraction as inputs to make learning easier, we use it to target specific states. The application of our refinement method bounds the suboptimality of the representation, while the task-independent clustering ensures that skills are transferable. In contrast to our objective, existing methods usually tackle suboptimality for a task without addressing transfer learning or exploration (Akrour et al., 2018; Abel et al., 2016). The k-d tree algorithm (Friedman et al., 1977) has been used to perform state abstraction over a continuous state space (Uther & Veloso, 1998), but as above, the splitting process takes advantage of extrinsic reward and previously defined partitions are not adapted throughout the learning process. In the domain of developmental robotics, RIAC and SAGG-RIAC (Baranes & Oudeyer, 2009; 2010) already implement a splitting algorithm building a tree of subregions in order to efficiently explore the environment and learn a forward model. More precisely, they split the state space to maximize either the sum of variance of interactions already collected or the difference of learning progress between subregions. However, these heuristics do not scale to larger continuous environments. In contrast, we assign states to subregions according to the proximity of states and use these subregions as reusable skills to solve several tasks. ASAP (Mankowitz et al., 2016) partitions the goal space, but does not use intrinsic motivation and the partitions are limited to hyper-plans.

6. Conclusion

We proposed ELSIM, a novel algorithm that continually refines discrete skills using a recently defined diversity heuristic (Eysenbach et al., 2019). To do so, the agent progressively builds a tree of different skills in the direction of a high-level objective. As shown in Section 4, **ELSIM expands the area associated to a skill** thanks to its exploratory behavior which comes from adding latent variables to the overall policy. **ELSIM also focuses its training on interesting skills relatively to some tasks**. Even though the agent is often learning a task, the skills can be defined independently from a specific task and we showed that ELSIM possibly makes them **transferable across different tasks** of a similar environment. Since the agent does not need extrinsic reward to learn, we show that it can **improve exploration** on sparse rewards environments. We believe that such a paradigm is appropriate for *lifelong learning*.

Currently, our method allows to avoid the problem of catastrophic forgetting, but the counterpart is an increase of the memory footprint, which is a recurrent issue in methods based on trees. Several works addressing catastrophic forgetting may be adapted to our work, e.g. (Lopez-Paz & Ranzato, 2017) and could potentially improve transfer learning between neural networks at different levels of our tree. In addition, ELSIM quickly gets stuck in local optimas in more difficult environments such as *BipedalWalker-v2* or Pybullet environments. The main limitation of our approach is that we cannot select several skills in one episode, such as one would make within the *option* framework (Sutton et al., 1999). To be adapted, the *tree-policy* should be dependent on the true state and the diversity heuristic should maximize

$\mathbb{E}_{s\sim\mathbb{U}(s)}\, I(G, S'|S)$ rather than eq. 6 like in (Sharma et al., 2019). Thus the curriculum algorithm should be modified. It would result that the semantic meaning of a skill would be no longer to target an area, but to produce a change in the state space. We plan to address these issues in future work.

Acknowledgment

We thank O. Sigaud and A. Dutech for their useful feedbacks on the paper and NVIDIA Corporation for the donation of a Titan V.

References

Abel, D., Hershkowitz, D. E., and Littman, M. L. Near optimal behavior via approximate state abstraction. In *ICML*, pp. 2915–2923, 2016.

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Akrour, R., Veiga, F., Peters, J., and Neumann, G. Regularizing reinforcement learning with state abstraction. In *IROS*, pp. 534–539. IEEE, 2018.

Aubret, A., Matignon, L., and Hassas, S. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Baranes, A. and Oudeyer, P.-Y. R-iac: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009.

Baranes, A. and Oudeyer, P.-Y. Intrinsically motivated goal exploration for active motor learning in robots: A case study. In *IROS*, pp. 1766–1773, 2010.

Barber, D. and Agakov, F. V. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, pp. 201–208, 2003.

Barto, A. G. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47. Springer, 2013.

Chevalier-Boisvert, M. and Willems, L. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *ICML*, pp. 1008–1017, 2018.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl\$^2\$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*, 2017.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM (TOMS)*, 3(3):209–226, 1977.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. In *ICLR 2017*, 2017.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *CoRR*, abs/1806.04640, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pp. 1856–1865, 2018.

Jabri, A., Hsu, K., Gupta, A., Eysenbach, B., Levine, S., and Finn, C. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 10519–10530, 2019.

Kompella, V. R., Stollenga, M., Luciw, M., and Schmidhuber, J. Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335, 2017.

Levy, A., Platt, R., and Saenko, K. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Machado, M. C. and Bowling, M. Learning purposeful behaviour in the absence of rewards. *arXiv preprint arXiv:1605.07700*, 2016.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *ICML*, volume 70, pp. 2295–2304. JMLR. org, 2017.

Mankowitz, D. J., Mann, T. A., and Mannor, S. Adaptive skills adaptive partitions (asap). In *Advances in Neural Information Processing Systems*, pp. 1588–1596, 2016.

Metzen, J. H. and Kirchner, F. Incremental learning of skill collections based on intrinsic motivation. *Frontiers in neurorobotics*, 7:11, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Moore, A. W. Efficient memory-based learning for robot control. 1990.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3303–3313. 2018.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. In *ICLR*, 2019a.

Nachum, O., Tang, H., Lu, X., Gu, S., Lee, H., and Levine, S. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019b.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9209–9220, 2018.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ryan, R. M. and Deci, E. L. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

Shariff, R. and Dick, T. Lunar lander: A continous-action case study for policy-gradient actor-critic algorithms, 2013.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, 2017.

Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pp. 640–646, 1996.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ IROS*, pp. 5026–5033. IEEE, 2012.

Uther, W. T. and Veloso, M. M. Tree based discretization for continuous state space reinforcement learning. In *Aaai/iaai*, pp. 769–774, 1998.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T. D., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. In *ICLR*, 2019.