POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Unsupervised Pre-Training for Reinforcement Learning via Recursive History Encoders

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Pietro Maldini

Advisor: Prof. Marcello Restelli

Co-advisor: Dott. Mirco Mutti

Academic year: 2020-2021

1. Introduction

Reinforcement Learning (RL) is a sub-field of machine learning that aims at solving sequential decision problems formalized as Markov Decision Processes (MDPs). In an MDP a decision maker, the *agent*, is placed inside an environment. The agent can perceive the current conditions of this environment, the *state*. The agent interacts with the environment by performing actions. The behaviour of the agent, i.e. its action choosing strategy, is called a *policy*. When an action is performed the agent receives a scalar signal, the *reward*, which indicates how the action is compliant with a specified task to learn. The learning process starts from an initial behaviour, often random, that is used to interact with the environment. Using the reward signal as feedback the behaviour is changed to improve its performance. The design of a reward signal is hard and the shape of the reward defines the hardness of the subsequent learning process. A sparse reward is chosen so that a reward is given only when a specific task is performed or the desired goal is reached. A sparse reward is difficult to collect with a random policy, this choice often leads to inefficient learning. If the reward is not found, the agent has no feedback to modify its policy, so learning is not possible. To overcome this limitation a new approach has been proposed, which is similar to that of unsupervised pre-training of supervised learning models. Unsupervised pre-training for RL aims to learn policies with no supervised feedback from the environment so that it is agnostic to the task. The main goal is to learn to explore the environment, reach a wide range of states and master various skills. A policy with those features could then be fine-tuned to perform different tasks in the environment. Using an exploratory policy as initial behaviour for a subsequent RL task helps collect sparse rewards which can't be collected by random initial policies. This leads to faster learning of a subsequent task. In particular, we consider the extension of the unsupervised pre-training problem to multiple environments. An agent can be in one environment within a class of environments and the objective is to learn to explore every environment in that class. In this setting Markovian policies are known to be sub-optimal [3]. The contribution of this thesis is the proposal of an architecture to represent non-Markovian policies and show empirical results of the advantages of a non-Markovian policy in the unsupervised pre-training in multiple envi-

ronments.

2. Notation

In this chapter, we introduce the notation used in the remained of this document. The interaction process between an agent an environment happens in a sequence of discrete time-steps, $t = 0, 1, 2, 3, ..., T$. At each time-step t the agent uses the current state of the environment s_t to choose an action to perform a_t. After the action is performed, the agent receives the next state s_{t+1} and a reward signal r_{t+1}. A finite Markov Decision Process (MDP) is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, d_0, T \rangle$. In order, they represent, the set of possible states, the set of available actions, the dynamics of the environment, the initial state distribution and the time horizon. In particular, we consider the multiple environments MDPs where the agent is placed in one among a set of environments $\mathcal{M} = \mathcal{M}_1, ..., \mathcal{M}_I$, every environment has its dynamics whereas the state and action space are in common. At the beginning of an episodic interaction, the agent is placed inside one of the environments following a distribution $p_{\mathcal{M}}$ defined over $\mathcal{M}$. An agent placed in an environment interacts with it at discrete time-steps t, using its policy function π to choose the action to perform. A Markovian policy uses the current state of the environment s_t as an input to π. A non-Markovian policy, instead uses the history $h_t = (s_t, s_{t-1}, ..., s_0)$. A typical measurement used to assess the performance of a policy is the cumulative reward:

$$\mathcal{J} = \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{k=0}^{T-1} r_{k+1}(\tau),$$

where $r_{k+1}(\tau)$ represents the reward received at time step $k+1$ in the trajectory τ. We define trajectory $\tau = \langle s_t, a_t, r_t \rangle_{t=0,...,T-1}$ as a sequence of states, actions and reward collected in an episode. The interaction between an agent following a policy π and an MDP induces a distribution over the states d^π, $d^\pi(s)$ represents the probability of being in state s following the policy π. Lastly an important measure we consider is an estimated version of the differential entropy used in [5], which can be computed by sampling from a distribution. Given N samples obtained from a distribution $f(x)$, and a number $k < N$, we can compute a k-nearest neighbor estimate of the differential entropy:

$$\hat{H}_k(f) = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{k}{N V_i^k} + \log k - \Psi(k),$$

where Ψ is the digamma function and V_i^k is the volume of the hyper-sphere of radius $R_i = |x_i - x_i^{knn}|$ that represents the euclidean distance between x_i and its k^{th} nearest neighbor x_i^{knn}.

3. State of the Art

In this Section, we provide a quick overview of the current state of the art of unsupervised pre-training in RL.

3.1. Maximum State Entropy Methods

To solve the unsupervised pre-training problems several approaches have been proposed. The objective of those algorithms can be formalized as the maximization problem proposed by Hazan et al. in [1] :

$$\pi^* \in \arg\max_{\pi \in \Pi} H(d_\pi),$$

where $H(d_\pi)$ is the entropy of the state distribution induced by the policy π. Those algorithms showed outstanding results compared with state-of-the-art solutions in single environments. *Maximum Entropy POLicy optimization* (Mepol) [5] using the estimate $\hat{H}_k(d_\pi)$. From here we will refer to this estimate as H.

3.2. Unsupervised Pre-Training in Multiple Environments

The families of algorithms described up to now are mainly focused on learning a single policy for a single environment. Some works proposed different approaches to extend these results to the multiple environments setting. *Change-Based Exploration Transfer* (C-BET) [6] extended count-based methods to allow learning in multiple environments by considering interesting changes in the environment as an additional bonus. αMepol [4] extended Mepol by changing its objective to a risk-averse variant of it. Their risk measure is the Conditional Value-at-Risk (CVaR) of the entropies of the collected trajectories, that is the mean of the α percentile of the distribution of collected entropies. This objective prioritizes the entropy in the worst case

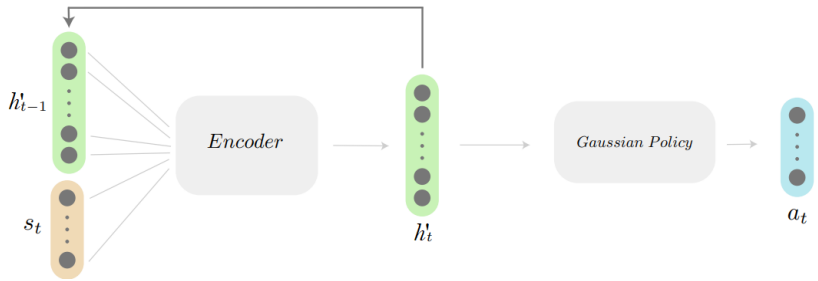


Figure 1: A representation of the policy architecture learned by HMepol.

allowing for a better exploration in rare or particularly difficult environments, with a trade-off with easier and more frequently visited environments.

4. Motivations

Policies learned using unsupervised pre-training algorithms showed promising results when used as initial policies for subsequent reinforcement learning tasks. With the increase in research interest, some limitations started to arise. Mutti et al. in [3] showed theoretical limits of Markovian policies in finite samples regime compared with non-Markovian policies. When exploring an environment requires visiting a state multiple times, and to act differently every time, a Markovian policy is forced to randomize its action. While being more complex, a non-Markovian policy can use information from history to overcome this limit and act deterministically. Further limitations are introduced in the multiple environments setting. A markovian agent cannot understand in which environment it is placed, leading to a suboptimal behaviour. A non-Markovian policy can instead identify the environment and adapt the strategy to the current environment, allowing it to reach an optimal behaviour.

5. An Architecture to Represent Non-Markovian Policies

Our main contribution is the proposal of a new architecture to represent non-Markovian policies. The proposed architecture is composed of two parts. A *recursive history encoder*, that allows to create and to update a compact representation, as a limited size vector, of the history at each time-step, which we call h'_t. At the beginning of each trajectory, the history vector is initialized with a zero vector, namely $h'_0 = \mathbf{0}$. At each time step t, the previous history vector h'_{t-1} and the current state s_t are encoded into a new history vector h'_t according to the following recursive equation:

$$\begin{cases} h'_0 = \mathbf{0} \\ h'_t = g_{\boldsymbol{\theta}}(s_t, h'_{t-1}), \end{cases}$$

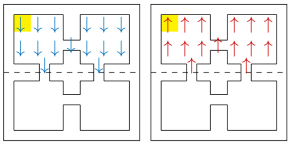
where $g_{\boldsymbol{\theta}}(\cdot)$ is the function computed by the history encoder, which is implemented as a multilayer perceptron parametrized by $\boldsymbol{\theta}$. Using this recursive definition, it is possible to highlight the dependence of the history at each time-step with the states visited in all the previous time-steps. By employing a gradient ascent procedure it is possible to learn a function $g_{\boldsymbol{\theta}}$ that retains only the information relevant for future decisions. The second component is a diagonal Gaussian policy represented with a neural network. This component is often used to represent stochastic Markovian policies. In the Markovian case, it receives as input the current state s_t, in our case, it receives the current history vector h'_t. It uses this input to compute the mean and the covariance matrix of a multivariate Gaussian distribution. The action is then sampled from the resulting distribution. The combination of these two components (see Figure 1) allows to represent a non-Markovian policy. The underlying learning algorithm is a modified version of the model-free, policy-gradient algorithm MEPOL introduced in [5], which we call History Mepol (HMepol).

6. Experimental Results

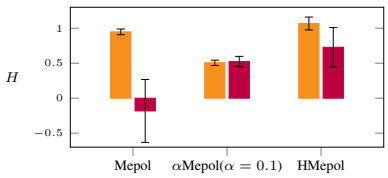
In this section, we present some of the main results obtained in our experimental evaluation. We compare entropies obtained by Mepol [5], αMepol [4] and HMepol over different domains.

6.1. GridWorld with Slope

We consider a class $\mathcal{M}$ of two configurations of a continuous GridWorld domain with 2D states and actions, which was used in [4] with the name *GridWorld with Slope*. The two configurations are composed of four rooms connected by narrow hallways. The action space is a (bounded) increment representing the movement of the agent along with the two coordinates. The difference between the two environments lies in the transition dynamics. As shown in Fig. 2a, the environments have slopes of opposed directions. As a consequence, the movement of the agent is altered in different ways. The yellow area represents the initial position while the arrows represent the slope direction. To make the setting more interesting, we consider an unbalanced distribution over the class of environments $\mathcal{M}$, with $p_{\mathcal{M}} = [0.8, 0.2]$. In Figure 2, we compare the performance of optimal policies obtained by Mepol [5], αMepol [4] ($\alpha = 0.2$), and HMepol within 500 epochs. We show the advantage of HMepol in both the weighted average over the class and the performance in single environments. We can notice that a policy learned with HMepol outperforms the one learned by αMepol in Gridworld with a northwards slope (red arrows), which is rarely visited. The introduction of non-Markovianity allows reaching better performance in the worst case using the average entropy objective and not a specific risk-averse objective like in αMepol.



(a) GridWorld with Slope



(b) Entropy Results

Figure 2: Pre-training performance by αMepol ($\alpha = 0.2$), and HMepol in the GridWorld with Slope domain (a). In (b), on the left bar the average performance and on the right the performance in the rare configuration (northwards slope). For every plot, we provide 95% c.i. over 8 runs.

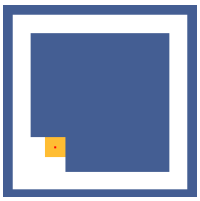
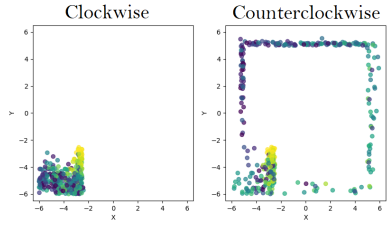


Figure 3: A visual representation of GridWorld with Corridor environment. The shaded area represents the initial position of the agent.

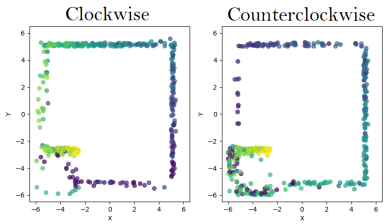
6.2. GridWorld with Corridor

To highlight the benefits of the identification of the environment within a class, we created another set of GridWorlds. A graphical representation of the domain is presented in Figure 3, and the action space is the same as in the previously seen environment. We consider two configurations sampled with equal probability, both with a wind that modifies the agent's actions everywhere except in the bottom-left room. One configuration has clockwise wind and the other one has counterclockwise wind. The hardness of this environment can be controlled by modifying the strength of the wind. Up to a certain value of wind, the agent can enter the corridor also with a headwind, allowing it to explore the corridor in one direction without a considerable loss in performance. This is no longer true with stronger winds, which stop the agent from entering the corridor in the wrong direction. In Figure 4a and Figure 4b, we visualize ten trajectories generated using the policy learned by Mepol and by HMepol. We can see that a Markovian policy, once the wrong entrance is picked, will move around that same entrance. Instead, a non-Markovian policy can identify the environment from what happens at that entrance. In case of a mistake, it can correct itself and move towards the other entrance. In particular, we repeated the training process with different wind forces and in Figure 5a we can see how a Markovian policy can perform quite well up to certain wind strength, while a non-Markovian policy can reach consistent entropy results. In Figure 5b and Figure 5c) we show also results obtained by fine-tuning with the TRPO algorithm [7] the policies pre-trained with Mepol and HMe-

pol with stronger wind force (0.4). We can see how the advantage in entropy results in a faster fine-tuning process, in the counterclockwise configuration this difference is more evident, with HMepol reaching in less than 10 epochs the same average return reached by Mepol in 50 epochs.



(a) Mepol

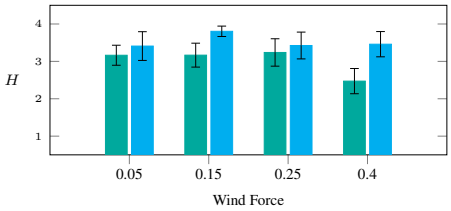


(b) HMepol

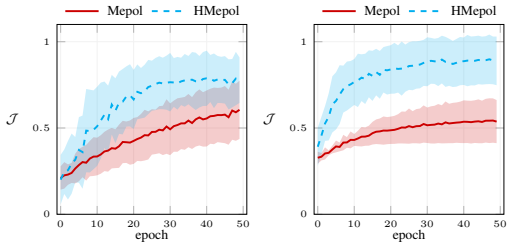
Figure 4: Sample of 10 trajectories over both configurations of GridWorld with Corridor using Mepol (a) and HMepol (b), on the left with clockwise wind and on the right with counterclockwise wind.

6.3. Multi Grid

To provide evidence of the ability of HMepol to scale to larger classes of environments we employed the setting used in [4], which is called MultiGrid. We consider a class of ten different GridWorld configurations sampled with equal probability. We compare the performance obtained by αMepol [4] ($\alpha = 0.1$) and HMepol within 500 epochs. In Figure 6, we have on the left bar the average entropy, on the right bar the entropy in the worst configuration. We can see that also with an increased number of environments HMepol can reach better average performance, but also a higher entropy in the worst case (first configuration). To further understand the advantages of our architecture, we provide in Figure 7a and 7b heatmaps of the state distribution obtained by running 100 trajectories with policies learned by αMepol and HMepol. We can notice how the non-Markovian policies reach a better coverage of the state space.



(a) Different Wind forces



(b) clockwise (c) counterclockwise

Figure 5: Average entropy (a), on the clockwise configuration (left bars) and on the counterclockwise configuration (right bars), with 95% c.i. over 8 runs. We also provide average returns obtained by fine-tuning the trained policies using TRPO [7] with 5 random goal locations for both clockwise (b) and counterclockwise (c) configurations.

7. Conclusions

In this thesis, we proposed HMepol, a policy-search algorithm for unsupervised pre-training in multiple environments. We explained the motivations that led our attention towards non-Markovian policies. Then, we showed the advantage of the proposed algorithm over state-of-the-art algorithms. We have also suggested the ability of the algorithm to scale to large classes of environments. We now propose some direc-

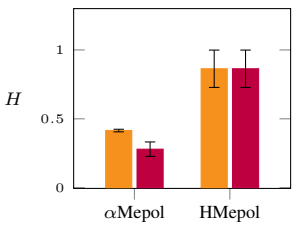
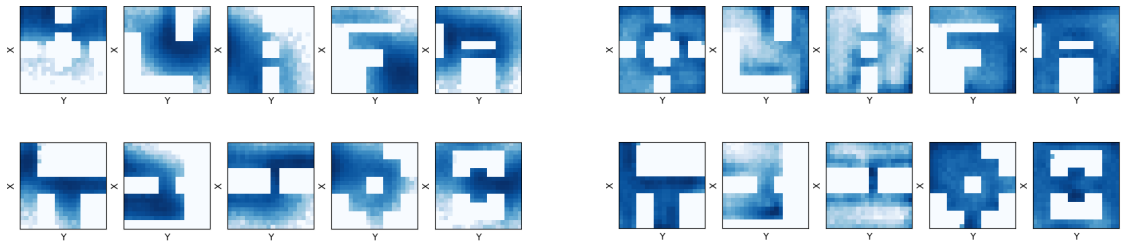


Figure 6: Average entropy obtained with policies trained with αMepol and HMepol, considering the uniform distribution over the configurations (left bar) and on the first configuration (right bar). We provide 95% c.i. over 8 runs.



(a) αMepol (b) HMepol

Figure 7: Heatmaps of log probability of state visitation for all configurations in MultiGrid induced by a policy learned using αMepol (a) and HMepol (b).

tions for possible future research. HMepol, as it is proposed in the thesis, is unable to handle vision-based domains. In the field of unsupervised pre-training for reinforcement learning, some work [4] uses compact state representation resulting from randomly initialized convolutional encoders. One possible approach would be to introduce the recursive history encoder after these convolutional encoders, but alternative architectures might be considered as well. Another direction consists in finding a more efficient and effective architecture to represent a non-Markovian policy. The current architecture is composed of two neural networks, which are quite small in comparison with other deep models. At the same time, scaling to high dimensional environments may require increasing the dimension of the history vector, which should be big enough to keep all the required information about the past. One possible architectural change that can be done to the current policy is to combine the two networks into one single network with two outputs, the action and the history vector. Other interesting architectures may also use recurrent neural networks with Long Short-Term Memory (LSTM) [2], which can control what to remember, and can better learn long-term dependencies. The use of more complex architectures may also lead to overall better results, at the cost of additional computational resources and computational complexity. Finally, we believe that this work motivates the study of non-Markovian architectures for unsupervised reinforcement learning and that it represents a further step towards the development of artificial general intelligence.

References

[1] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Mirco Mutti, Riccardo De Santi, and Marcello Restelli. The importance of non-markovianity in maximum state entropy exploration. *arXiv preprint arXiv:2202.03060*, 2022.

[4] Mirco Mutti, Mattia Mancassola, and Marcello Restelli. Unsupervised reinforcement learning in multiple environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[5] Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9028–9036, 2021.

[6] Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[7] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.