2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Online Baum-Welch algorithm for Hierarchical Imitation Learning

Vittorio Giammarino[1] and Ioannis Ch. Paschalidis[2]

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683044

***Abstract*— The options framework for hierarchical reinforcement learning has increased its popularity in recent years and has made improvements in tackling the scalability problem in reinforcement learning. Yet, most of these recent successes are linked with a proper options initialization or discovery. When an expert is available, the options discovery problem can be addressed by learning an options-type hierarchical policy directly from expert demonstrations. This problem is referred to as hierarchical imitation learning and can be handled as an inference problem in a Hidden Markov Model, which is done via an Expectation-Maximization type algorithm. In this work, we propose a novel online algorithm to perform hierarchical imitation learning in the options framework. Further, we discuss the benefits of such an algorithm and compare it with its batch version in classical reinforcement learning benchmarks. We show that this approach works well in both discrete and continuous environments and, under certain conditions, it outperforms the batch version.**

I. Introduction

Hierarchical Reinforcement Learning (HRL) addresses the scalability problem in classical Reinforcement Learning (RL) [1] by dividing the agent policy in decisions that are temporally extended over several steps (higher-level) and in others taken at each step (lower-level). Most of the recent successes of HRL [2] rely in learning a good hierarchical structure which divides the main problems in sub-problems and tackles them separately by means of single *options* [3]. In the literature, the hierarchical learning problem is either decoupled in option initialization, also called option discovery, and in optimal option selection [4], [5], [6], or it is performed in an end-to-end fashion where the entire hierarchy is learnt while solving the task [7], [8]. When for a specific task an expert is available, initializing policies by direct observation of the expert behavior is beneficial to speed up the learning [9]. The procedure of learning policies from expert data is called imitation learning [10], [11] and, in hierarchical frameworks, recent works have focused on inferring not only the expert policy but also its underlying hierarchical structure. These studies are generally divided in: Hierarchical Inverse Reinforcement Learning (HIRL), which infers a hierarchical reward function either from expert demonstrations (state-action pairs) [12], or only observations (states) [13]; and Hierarchical Imitation Learning, (HIL) which directly learns the expert policy in a hierarchical fashion [14], [15], [16], [17]. In this paper, we assume that the expert follows an options-type hierarchical policy and we formulate an online algorithm to perform end-to-end HIL. We leverage the idea that the *Options Probabilistic Graphical Model (OPGM)* in Fig. 1 can be handled as a special case of a *Hidden Markov Model (HMM)* [18], [19] and that inference in HMM can be performed via an *Expectation-Maximization (EM)* recursion, also known as the *Baum-Welch (BW)* algorithm [20], [21]. Given the expert demonstrations, this algorithm alternates between an Expectation step (E-step), which computes a surrogate of the log-likelihood, and a Maximization step (M-step), which maximizes such a function over the policy space. By alternating the E-step and the M-step several times, the BW algotrithm is able to find a policy which (locally) maximizes the log-likelihood.

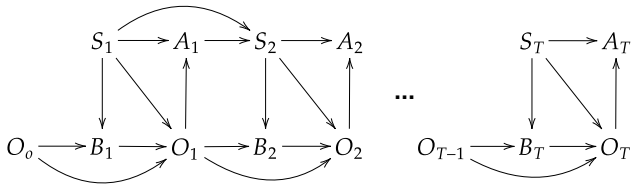

Fig. 1: Graphical model for the options framework.

Related Work and Contributions: Works related to our method are [22], [23], [24], which exploit a batch version of the BW algorithm to perform end-to-end HIL. In these algorithms, the E-step is carried out through a forward-backward recursion [25], which needs a sweep through the entire data set at each iteration. As such, for environments where many training samples are required, this procedure is expensive and motivates the development of an online algorithm which processes the data on-the-fly. In addition to efficiency, online algorithms are also memory-wise efficient, since, at each iteration, a single sample is processed and then discarded. Note that we examine the two algorithms in competition; however, batch and online versions are complementary and in practical applications can be used together in sequence. We now summarize the main contributions of this work: (i) the batch version of the BW algorithm for HIL in [22], [23], [24] requires to process the entire data set at each E-step; to tackle this issue, we build upon the works in [26], [27] for the HMM setting and develop an online recursion for the OPGM in Fig. 1 which processes the data on-the-fly. To the best of our knowledge, this is the first online end-to-end algorithm for HIL. (ii) Both [26] and [27] make assumptions on the policy parameterization, we relax these assumptions and formulate a recursion that can use Neural Networks (NN) as functions approximations to parameterize the hierarchical

[1]V. Giammarino is with Division of Systems Engineering, Boston University, Boston, MA 02446, USA. vittoriogiammarino@gmail.com

[2] Ioannis Ch. Paschalidis is with Dept. of Electrical and Computer Engineering, Division of Systems Engineering, and Dept. of Biomedical Engineering, Boston University, 8 St. Mary's St., Boston, MA 02215, USA. yannisp@bu.edu

policy. (iii) We compare the two versions of the BW for HIL algorithm via empirical experiments on classical OpenAi RL benchmarks [28].[1]

Outline: In Section II, we introduce the OPGM and the imitation learning problem. Section III introduces the batch BW as in [22], [23], [24] and in Section IV we formulate the recursion for the online BW for HIL and provide an overview of the algorithm. Finally, Section V presents the regularization penalties we add to the cost function in order to obtain versatile options and Section VI compares empirically online and batch versions.

Notation: We use uppercase letters (e.g., S_t) for random variables, lowercase letters (e.g., s_t) for values of random variables, script letters (e.g., $\mathcal{S}$) for sets, and bold lowercase letters (e.g., $\boldsymbol{\theta}$) for vectors. Let $[t_1 : t_2]$ be the set of integers t such that $t_1 \leq t \leq t_2$; we write S_t such that $t_1 \leq t \leq t_2$ as $S_{t_1:t_2}$. Moreover, we refer to $\mathbb{1}[S_t = s_t]$ as the indicator function, which is 1 when $S_t = s_t$ and zero otherwise and to $\delta(\cdot)$ as the Kronecker delta. Finally, $\mathbb{E}[\cdot]$ represents expectation, $\mathbb{P}(\cdot)$ probability, $|\mathcal{S}|$ the cardinality of a set, and $||\cdot||_2$ the ℓ_2-norm.

II. Preliminaries

In the following we introduce the OPGM as illustrated in Fig. 1 and the imitation learning problem. The index t represents time and (S_t, A_t, O_t, B_t) denote state, action, option and termination indicator at time t, respectively. For all t, S_t is defined on the set of states $\mathcal{S}$, possibly infinite, A_t and O_t are respectively defined on the set of actions $\mathcal{A}$ and the set of options $\mathcal{O}$, both finite, and B_t is defined on the binary set $\mathcal{B} = \{0, 1\}$. Moreover, define the parameter $\boldsymbol{\theta} := (\boldsymbol{\theta}_{hi} \in \Theta_{hi}, \boldsymbol{\theta}_{lo} \in \Theta_{lo}, \boldsymbol{\theta}_b \in \Theta_b)$ where $\Theta := (\Theta_{hi} \times \Theta_{lo} \times \Theta_b) \subset \mathbb{R}^d$. Given any $(O_0, S_1) = (o_0, s_1)$, the joint distribution on the rest of the OPGM is determined by the following components: an unknown environment transition probability function $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ where $\Delta_{\mathcal{S}}$ denotes the space of probability distributions over $\mathcal{S}$, and a triplet of stationary policies $\{\pi_{hi}^{\boldsymbol{\theta}_{hi}}, \pi_{lo}^{\boldsymbol{\theta}_{lo}}, \pi_b^{\boldsymbol{\theta}_b}\}$ where $\pi_{hi}^{\boldsymbol{\theta}_{hi}}$ is the high level policy parameterized by $\boldsymbol{\theta}_{hi}$, $\pi_{lo}^{\boldsymbol{\theta}_{lo}}$ the low level policy parameterized by $\boldsymbol{\theta}_{lo}$ and $\pi_b^{\boldsymbol{\theta}_b}$ the termination policy parameterized by $\boldsymbol{\theta}_b$. The hierarchical decision process starts at $t = 0$, where the agent decides whether to terminate or not the current option o_0. This decision is encoded in the termination indicator b_1 sampled from $\pi_b^{\boldsymbol{\theta}_b}(\cdot|s_1, o_0)$, where $\pi_b^{\boldsymbol{\theta}_b} : \mathcal{S} \times \mathcal{O} \to \Delta_{\mathcal{B}}$. If $b_1 = 1$, the option o_0 terminates and the next sample o_1 is sampled from $\pi_{hi}^{\boldsymbol{\theta}_{hi}}(\cdot|s_1)$, where $\pi_{hi}^{\boldsymbol{\theta}_{hi}} : \mathcal{S} \to \Delta_{\mathcal{O}}$; otherwise, if $b_1 = 0$, the option o_0 continues and $o_1 = o_0$. Next, the action a_1 is sampled from $\pi_{lo}^{\boldsymbol{\theta}_{lo}}(\cdot|s_1, o_1)$, where $\pi_{lo}^{\boldsymbol{\theta}_{lo}} : \mathcal{S} \times \mathcal{O} \to \Delta_{\mathcal{A}}$, and the agent interacts with the environment through the low level policy associated with the option o_1. Finally, the next state s_2 is sampled from $P(\cdot|s_1, a_1)$, and the rest of the samples $(s_{3:T}, a_{2:T}, o_{2:T}, b_{2:T})$ are generated analogously. The just described decision process, based on the triplet $\{\pi_{hi}^{\boldsymbol{\theta}_{hi}}, \pi_{lo}^{\boldsymbol{\theta}_{lo}}, \pi_b^{\boldsymbol{\theta}_b}\}$, encodes the hierarchical agent policy in the options framework. For the sake of completeness, we define $\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}$ as

$$\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o_t|o_{t-1}, s_t, b_t) := \begin{cases} \pi_{hi}^{\boldsymbol{\theta}_{hi}}(o_t|s_t), & \text{if } b_t = 1, \\ 1, & \text{if } b_t = 0, o_t = o_{t-1}, \\ 0, & \text{if } b_t = 0, o_t \neq o_{t-1}. \end{cases} \tag{1}$$

Fixing the initial state $S_1 = s_1$ and the initial option $O_0 = o_0$, the joint distribution of $\{S_{2:T}, A_{1:T}, O_{1:T}, B_{1:T}\}$ becomes

$$\begin{aligned} &\mathbb{P}_{o_0,s_1}^{\boldsymbol{\theta}}(S_{2:T} = s_{2:T}, A_{1:T} = a_{1:T}, O_{1:T} = o_{1:T}, B_{1:T} = b_{1:T}) \\ &= \left[\prod_{t=1}^{T} \pi_b^{\boldsymbol{\theta}_b}(b_t|s_t, o_{t-1}) \tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o_t|o_{t-1}, s_t, b_t) \pi_{lo}^{\boldsymbol{\theta}_{lo}}(a_t|s_t, o_t)\right] \\ &\quad \times \left[\prod_{t=1}^{T-1} P(s_{t+1}|s_t, a_t)\right]. \end{aligned} \tag{2}$$

Concerning the Imitation Learning (IL) problem, it is defined as inferring the underlying expert distribution via a set of demonstrations (state-action samples) generated while solving a task [29]. When we assume the expert behavior follows a hierarchical policy with true parameters $(\boldsymbol{\theta}_{hi}^*, \boldsymbol{\theta}_{lo}^*, \boldsymbol{\theta}_b^*)$, and given initial conditions (o_0, s_1), the process of estimating $(\boldsymbol{\theta}_{hi}^*, \boldsymbol{\theta}_{lo}^*, \boldsymbol{\theta}_b^*)$ through a finite sequence of expert demonstrations $(s_{2:T}, a_{1:T})$ with $T \geq 2$ is called HIL. One way to address this problem is by solving:

$$\max_{(\boldsymbol{\theta}_{hi}, \boldsymbol{\theta}_{lo}, \boldsymbol{\theta}_b) \in \Theta} \mathcal{L}(\boldsymbol{\theta}_{hi}, \boldsymbol{\theta}_{lo}, \boldsymbol{\theta}_b), \tag{3}$$

where $\mathcal{L}(\boldsymbol{\theta}_{hi}, \boldsymbol{\theta}_{lo}, \boldsymbol{\theta}_b)$ denotes the marginal log-likelihood and is equivalent to the logarithm of the joint probability of generating the expert demonstrations $(s_{2:T}, a_{1:T})$ given (o_0, s_1) and the parameters $\boldsymbol{\theta}_{hi}, \boldsymbol{\theta}_{lo}, \boldsymbol{\theta}_b$, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}_{hi}, \boldsymbol{\theta}_{lo}, \boldsymbol{\theta}_b) = \log \sum_{o_{1:T}, b_{1:T}} \mathbb{P}_{o_0,s_1}^{\boldsymbol{\theta}}(s_{2:T}, a_{1:T}, o_{1:T}, b_{1:T}). \tag{4}$$

Note that, $\mathbb{P}_{o_0,s_1}^{\boldsymbol{\theta}}(s_{2:T}, a_{1:T}, o_{1:T}, b_{1:T})$ in (4) is the same as (2), but we have dropped the random variables $S_{2:T}, A_{1:T}, O_{1:T}, B_{1:T}$ to streamline the notation. The optimization problem in (3) is hard to evaluate for our framework, considering also that for a long sequence of demonstrations Eq. (4) gets close to zero. Yet, the BW algorithm provides an iterative procedure based on EM which solves (3) by maximizing a surrogate of (4). The way we compute this surrogate during the E-step determines the main difference between the batch and our online version of the algorithm.

III. Batch Baum-Welch for Hierarchical Imitation Learning

In this section we draw the main ingredients of the batch BW for HIL as in [23]. As mentioned, this algorithm alternates between the E-step and the M-step: during the E-step we compute a surrogate of (4), the Baum's auxiliary function [21], with respect to the previously obtained vector of parameters $\boldsymbol{\theta}^{\text{old}}$. Then, in the M-step, we optimize this

[1]All the code is available at `https://github.com/VittorioGiammarino/Online_BWforHIL`.

function with respect to a new vector of parameters $\boldsymbol{\theta}$. Given $(O_0, S_1) = (o_0, s_1)$, we obtain the following Baum's auxiliary function for the OPGM (cf. [30] for the complete derivation)

$$Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}}) = \frac{1}{T} \sum_{o_{1:T}, b_{1:T}} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{1:T}, b_{1:T}|(s_t,a_t)_{1:T}) \times \log \mathbb{P}^{\boldsymbol{\theta}}_{o_0,s_1}(s_{2:T}, a_{1:T}, o_{1:T}, b_{1:T}). \tag{5}$$

By replacing $\mathbb{P}^{\boldsymbol{\theta}}_{o_0,s_1}(s_{2:T}, a_{1:T}, o_{1:T}, b_{1:T})$ with (2), Eq. (5) becomes

$$\begin{aligned} Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}}) = \frac{1}{T}\Bigg\{ & \sum_{t=2}^{T} \sum_{o_{t-1}} \sum_{b_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{t-1}, b_t|(s_t,a_t)_{1:T}) \\ & \times \log \pi^{\boldsymbol{\theta}_b}_b(b_t|s_t, o_{t-1}) \\ & + \sum_{t=1}^{T} \sum_{o_{t-1}} \sum_{b_t} \sum_{o_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{t-1}, b_t, o_t|(s_t,a_t)_{1:T}) \\ & \times \log \tilde{\pi}^{\boldsymbol{\theta}_{hi}}_{hi}(o_t|o_{t-1}, s_t, b_t) \\ & + \sum_{t=1}^{T} \sum_{o_t} \sum_{b_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_t, b_t|(s_t,a_t)_{1:T}) \log \pi^{\boldsymbol{\theta}_{lo}}_{lo}(a_t|s_t, o_t) \\ & + \sum_{b_1} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(b_1|(s_t,a_t)_{1:T}) \log \pi^{\boldsymbol{\theta}_b}_b(b_1|s_1, o_0) + C \Bigg\}, \end{aligned} \tag{6}$$

where C is constant with respect to $\boldsymbol{\theta}$. In (6), $\mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{t-1}, b_t|(s_t,a_t)_{1:T})$ and $\mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_t, b_t|(s_t,a_t)_{1:T})$ are referred to as the smoothing distributions of the latent variables given the expert demonstrations $(s_t,a_t)_{1:T}$ and are computed, during the E-step of the batch algorithm, via forward-backward decomposition (cf. [30] and [23]). Moreover, C contains all constant terms (independent on $\boldsymbol{\theta}$), $\mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(b_1|(s_t,a_t)_{1:T}) \log \pi^{\boldsymbol{\theta}_b}_b(b_1|s_1,o_0)$ is neglected, for T large enough, for reasons linked with the forward-backward decomposition [23], and $\tilde{\pi}^{\boldsymbol{\theta}_{hi}}_{hi}$ depends on $\boldsymbol{\theta}_{hi}$ only through $\pi^{\boldsymbol{\theta}_{hi}}_{hi}$ in (1) for $b_t = 1$. Hence, using the convention $0 \log 0 = 0$, we can replace $Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$ by

$$\begin{aligned} Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}}) = \frac{1}{T}\Bigg\{ & \sum_{t=2}^{T} \sum_{o_{t-1}} \sum_{b_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{t-1}, b_t|(s_t,a_t)_{1:T}) \\ & \times \log \pi^{\boldsymbol{\theta}_b}_b(b_t|s_t, o_{t-1}) \\ & + \sum_{t=1}^{T} \sum_{o_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_t, b_t = 1|(s_t,a_t)_{1:T}) \log \pi^{\boldsymbol{\theta}_{hi}}_{hi}(o_t|s_t) \\ & + \sum_{t=1}^{T} \sum_{o_t} \sum_{b_t} \mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_t, b_t|(s_t,a_t)_{1:T}) \log \pi^{\boldsymbol{\theta}_{lo}}_{lo}(a_t|s_t, o_t) \Bigg\}. \end{aligned} \tag{7}$$

We summarize the batch BW for HIL recursion in Algorithm 1. As discussed, the main shortcoming of this algorithm is the need of processing the entire set of demonstrations, multiple times, at each iteration.

Algorithm 1 Batch Baum-Welch algorithm for HIL

1: **Require:** Observation sequence $(s_t,a_t)_{1:T}$; $o_0 \in \mathcal{O}$; $s_1 \in \mathcal{S}$; $N \in \mathbb{N}_+$ and $\boldsymbol{\theta}^{(0)} \in \Theta$.
2: **for** $n = 1, \dots, N$ **do**
3: Compute $\{\mathbb{P}^{\boldsymbol{\theta}^{(n-1)}}_{o_0,s_1}(o_{t-1}, b_t|(s_t,a_t)_{1:T})\}_{t=2}^T$ and $\{\mathbb{P}^{\boldsymbol{\theta}^{(n-1)}}_{o_0,s_1}(o_t, b_t|(s_t,a_t)_{1:T})\}_{t=1}^T$ ▷ E-step
4: Update $\boldsymbol{\theta}^{(n)} \in \arg\max_{\boldsymbol{\theta}\in\Theta} Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n-1)})$ based on (7) ▷ M-step
5: **end for**

IV. Online Baum-Welch for Hierarchical Imitation Learning

In the following, we replace the smoothing distributions $\mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_{t-1}, b_t|(s_t,a_t)_{1:T})$ and $\mathbb{P}^{\boldsymbol{\theta}^{\text{old}}}_{o_0,s_1}(o_t, b_t|(s_t,a_t)_{1:T})$ in (7) with a sufficient statistic $\phi^{\boldsymbol{\theta}}_T$ which is updated as soon as the new state-action pair (s_T, a_T) becomes available.

Given $O_0 = o_0$ and $S_1 = s_1$, the sufficient statistic $\phi^{\boldsymbol{\theta}}_T : \mathcal{O}^2 \times \mathcal{B} \times \tilde{\mathcal{A}} \times \tilde{\mathcal{S}} \to \mathbb{R}$, is defined as

$$\begin{aligned} & \phi^{\boldsymbol{\theta}}_T(o', b, o, s, a) = \\ & \frac{1}{T} \mathbb{E}^{\boldsymbol{\theta}}_{o_0,s_1}\Bigg[\sum_{t=1}^{T} \mathbb{1}[O_{t-1} = o', B_t = b, O_t = o, S_t = s, A_t = a] \Bigg| (s_t,a_t)_{1:T} \Bigg]. \end{aligned} \tag{8}$$

Where $\mathcal{O}$ is the set of options, $\mathcal{B}$ is the termination binary set, $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ is the finite and countable set of actions taken by the expert and $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ is the finite and countable set of states explored by the expert. Note that, to avoid confusion and distinguish between o_t as the value of the random variable O_t at time t and o_t as an element of the set $\mathcal{O}$, we change the notation compared to Section III. Hence, in (8) we use $o', o \in \mathcal{O}$, $b \in \mathcal{B}$, $s \in \tilde{\mathcal{S}}$, $a \in \tilde{\mathcal{A}}$ while we keep the notation $(s_t,a_t)_{1:T}$ for the expert demonstrations. In Proposition 1, a new Baum's auxiliary function for the online BW for HIL is obtained in terms of (8).

Proposition 1. *Given $\phi^{\boldsymbol{\theta}}_T$ defined in* (8)*, we can rewrite* (5) *as*

$$\begin{aligned} Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{old}) = & \sum_{o'} \sum_{o} \sum_{s} \sum_{a} \Big\{ \sum_{b} \phi^{\boldsymbol{\theta}^{old}}_T(o', b, o, s, a) \\ & \times \log \pi^{\boldsymbol{\theta}_b}_b(b|s, o') + \phi^{\boldsymbol{\theta}^{old}}_T(o', b = 1, o, s, a) \log \pi^{\boldsymbol{\theta}_{hi}}_{hi}(o|s) \\ & + \sum_{b} \phi^{\boldsymbol{\theta}^{old}}_T(o', b, o, s, a) \log \pi^{\boldsymbol{\theta}_{lo}}_{lo}(a|s, o) \Big\}, \end{aligned} \tag{9}$$

where $\pi^{\boldsymbol{\theta}_{hi}}_{hi}(o|s)$, $\pi^{\boldsymbol{\theta}_{lo}}_{lo}(a|s,o)$ and $\pi^{\boldsymbol{\theta}_b}_b(b|s,o)$ are arbitrarily parameterized.
Proof: See [30].

Remark 1. The new $Q^T_{o_0,s_1}(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$ in (9) needs to sum over $s \in \tilde{\mathcal{S}}$, $a \in \tilde{\mathcal{A}}$, where both $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{A}}$ are a finite and countable subsets of $\mathcal{S}$ and $\mathcal{A}$, in place of the sum over T in (7). Note that in many practical situations $|\tilde{\mathcal{A}}| \times |\tilde{\mathcal{S}}| < T$.

We proceed introducing the online recursion to update $\phi_T^{\boldsymbol{\theta}}$ in (8). First, $\phi_T^{\boldsymbol{\theta}}$ can be decomposed through the following two filters

$$\chi_T^{\boldsymbol{\theta}}(o) = \mathbb{P}_{o_0,s_1}^{\boldsymbol{\theta}}\big(O_T = o \big| (s_t, a_t)_{1:T}\big), \tag{10}$$

$$\rho_T^{\boldsymbol{\theta}}(o', b, o, s, a, o'') = \frac{1}{T}\mathbb{E}_{o_0,s_1}^{\boldsymbol{\theta}}\Bigg[\sum_{t=1}^{T}\mathbb{1}[O_{t-1} = o', B_t = b, O_t = o, S_t = s, A_t = a] \Bigg| O_T = o'', (s_t, a_t)_{1:T}\Bigg], \tag{11}$$

where $\chi_T^{\boldsymbol{\theta}} : \mathcal{O} \to \Delta_{\mathcal{O}}$ and $\rho_T^{\boldsymbol{\theta}} : \mathcal{O}^3 \times \mathcal{B} \times \tilde{\mathcal{A}} \times \tilde{\mathcal{S}} \to \mathbb{R}$. It follows that

$$\phi_T^{\boldsymbol{\theta}}(o', b, o, s, a) = \sum_{o''} \rho_T^{\boldsymbol{\theta}}(o', b, o, s, a, o'')\chi_T^{\boldsymbol{\theta}}(o''). \tag{12}$$

Proposition 2 shows the recursion to update (10)-(11).

Proposition 2. ***Initialization:***

$$\chi_0^{\boldsymbol{\theta}}(o) = P(O_0 = o), \tag{13}$$

$$\rho_0^{\boldsymbol{\theta}}(o', b, o, s, a, o'') = 0, \tag{14}$$

Recursion: *for $T > 0$ and the new state-action pair (s_T, a_T) it holds that*

$$\chi_T^{\boldsymbol{\theta}}(o) = \sum_{o'}\sum_{b} \pi_{lo}^{\boldsymbol{\theta}_{lo}}(a_T|s_T, o)\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o|b, s_T, o')\pi_b^{\boldsymbol{\theta}_b}(b|s_T, o')\chi_{T-1}^{\boldsymbol{\theta}}(o') \times \Big(\sum_{o'}\sum_{b}\sum_{o} \pi_{lo}^{\boldsymbol{\theta}_{lo}}(a_T|s_T, o)\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o|b, s_T, o') \pi_b^{\boldsymbol{\theta}_b}(b|s_T, o')\chi_{T-1}^{\boldsymbol{\theta}}(o')\Big)^{-1}, \tag{15}$$

$$\rho_T^{\boldsymbol{\theta}}(o', b, o, s, a, o'') = \sum_{o'''}\sum_{b'} \Big\{\frac{1}{T}\kappa(o' - o''', b - b', o - o'', s - s_T, a - a_T) + \Big(1 - \frac{1}{T}\Big)\rho_{T-1}^{\boldsymbol{\theta}}(o', b, o, s, a, o''')\Big\} \times \frac{\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o''|b', s_T, o''')\pi_b^{\boldsymbol{\theta}_b}(b'|s_T, o''')\chi_{T-1}^{\boldsymbol{\theta}}(o''')}{\sum_{o'''}\sum_{b'}\tilde{\pi}_{hi}^{\boldsymbol{\theta}_{hi}}(o''|b', s_T, o''')\pi_b^{\boldsymbol{\theta}_b}(b'|s_T, o''')\chi_{T-1}^{\boldsymbol{\theta}}(o''')}. \tag{16}$$

where $\kappa(o' - o''', b - b', o - o'', s - s_T, a - a_T) = \delta(o' - o''')\delta(b - b')\delta(o - o'')\delta(s - s_T)\delta(a - a_T)$.
Proof: See [30].

Based on Propositions 1-2, we formulate Algorithm 2 which is, to the best of our knowledge, the first online EM type algorithm suitable for end-to-end HIL within the options framework. In Algorithm 2, note that, we do not have to specify the number of iterations (N, in Algorithm 1) as we perform an E-step after each state-action pair available. Additionally, we inhibit the M-step for $t < T_{\text{min}}$ to ensure that $Q_{o_0,s_1}^T(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$ is numerically well-behaved which is not always the case for a small number of demonstrations.

Algorithm 2 Online Baum-Welch algorithm for HIL

Require: Observation sequence $(s_t, a_t)_{1:T}$; $O_0 = o$; $S_1 = s$; $\theta^{(0)} \in \Theta$.
for $t = 0, \dots, T$ **do**
  if $t = 0$ **then**
    Initialize $\rho_0^{\boldsymbol{\theta}^{(0)}}$ (14) and $\chi_0^{\boldsymbol{\theta}^{(0)}}$ in (13)
  end if
  if $t > 0$ **then**
    Compute $\rho_t^{\boldsymbol{\theta}^{(t-1)}}$ in (16) and $\chi_t^{\boldsymbol{\theta}^{(t-1)}}$ in (15)
    if $t > T_{\text{min}}$ **then**
      Compute $\phi_t^{\boldsymbol{\theta}^{(t-1)}}$ in (12) and Update $\boldsymbol{\theta}^{(t)} \in \arg\max_{\boldsymbol{\theta}^{(t)} \in \Theta} Q_{o_0,s_1}^T(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t-1)})$ with $Q_{o_0,s_1}^T(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}})$ in (9)
    else
      $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$
    end if
  end if
end for

V. Regularization Penalties

As additional requirement in HIL, we want to learn a set of interpretable and transferable options. To achieve this goal, we penalize the Baum's auxiliary functions in (7) and (9) with regularizers on both the high and low level policies [13].

High level policy regularizers: For $\pi_{hi}^{\boldsymbol{\theta}_{hi}}$, we introduce two regularizers L_b and L_v in (17). By minimizing L_b, we encourage the activation of each option with a target sparsity value $\tau = 1/|\mathcal{O}|$ in expectation over the training set. On the other hand, maximizing L_v, where var denotes the variance, we encourage the options activation to be varied and force each option to have a high probability for certain states and low for the rest.

$$L_b = \sum_{o} \big|\big|\mathbb{E}_s[\pi_{hi}^{\boldsymbol{\theta}_{hi}}(o|s)] - \tau\big|\big|_2, \quad L_v = \sum_{o} \text{var}_s[\pi_{hi}^{\boldsymbol{\theta}_{hi}}(o|s)]. \tag{17}$$

Low level policy regularizer: Additionally, we maximize a numerical approximation (cf. Appendix in [30]) of the Kullback–Leibler divergence (D_{KL}) of each low level policy over the set of demonstrations (18). This in order to enhance differentiation in $\pi_{lo}^{\boldsymbol{\theta}_{lo}}$ given different options:

$$L_{D_{KL}} = \sum_{o}\sum_{o', o \neq o'} D_{KL}\big(\pi_{lo}^{\boldsymbol{\theta}_{lo}}(a|s, o)||\pi_{lo}^{\boldsymbol{\theta}_{lo}}(a|s, o')\big). \tag{18}$$

Overall, at each M-step, we solve the following optimization problem

$$\boldsymbol{\theta}^{(T)} \in \arg\max_{\boldsymbol{\theta}^{(T)} \in \Theta} Q_{o_0,s_1}^T(\boldsymbol{\theta}|\boldsymbol{\theta}^{\text{old}}) - \lambda_b L_b + \lambda_v L_v + \lambda_{D_{KL}} L_{D_{KL}}.$$

Note that, given the different ways we construct the Baum's auxiliary functions for Algorithm 1 and 2, respectively in (7) and (9), the three penalties are differently implemented in the two settings. For more details on this regard refer to the Appendix in [30].

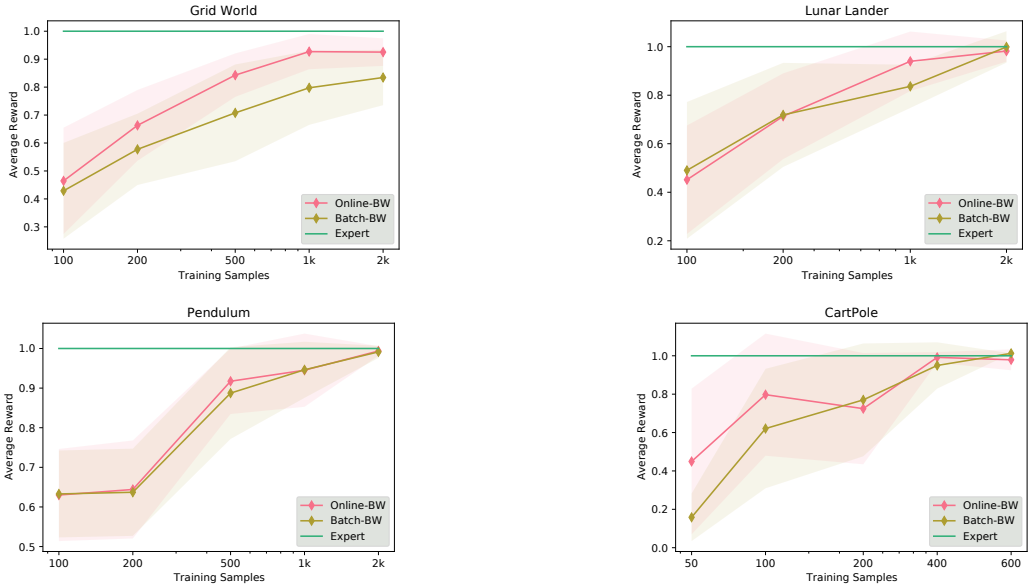

Fig. 2: Performance for different environments. The y-axis shows the reward averaged over the random seeds and scaled such that the expert achieves 1. The x-axis shows the size of the training set used for each trial. The shaded area indicates the standard deviation over the 10 reruns for cartpole, lunar lander and pendulum and over 30 reruns for the grid world.

VI. Comparison and Discussion

A. Implementation and Numerical Complexity

In the following we focus on the numerical complexity of the E-step for both Algorithms 1 and 2, since this is where the two algorithms differ the most. Algorithm 1 uses the forward-backward decomposition (Appendix in [30]): at each iteration updates a vector of size $T \times |\mathcal{O}| \times |\mathcal{B}|$ in $O(T \times |\mathcal{O}|^2 \times |\mathcal{B}|)$. On the other hand, Algorithm 2 updates a vector of dimension $|\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}| \times |\mathcal{O}|^2 \times |\mathcal{B}|$ with a numerical complexity of $O(|\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}| \times |\mathcal{O}|^3 \times |\mathcal{B}|^2)$. The bottleneck of the batch algorithm is T, the size of the expert demonstrations; while, in the online algorithm it is $|\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}|$ which is the combination of states explored and actions used by the expert. Generally, in order to obtain satisfactory learning in stochastic environments we have $T >> |\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}|$ which implies that a single online E-step in Proposition 2 is more efficient than a single forward-backward decomposition. However, consider that the batch BW (Algorithm 1) requires N iterations while, the online (Algorithm 2) T iterations where usually $T >> N$. Therefore, as acknowledged in [27], [31], this mere comparison is not always meaningful and requires to be further investigated via empirical experiments.

B. Experiments

We evaluate the two algorithms on 4 different tasks from the classic RL literature: three of them come from the OpenAI gym library [28], the cartpole, pendulum and lunar lander; and have a continuous state-space. The last is a grid-world type environment with discrete state-space and high stochasticity in transition and reward. We first generate the expert demonstrations running value iteration on the grid-world, Q-learning on the pendulum and cartpole [1] and a heuristic on lunar lander. After generating the demonstrations, we use Algorithms 1 and 2 to train an options-type hierarchical policy on the same triplet of feed-forward fully connected neural networks. For both $\pi_{lo}^{\boldsymbol{\theta}_{lo}}$ and $\pi_{b}^{\boldsymbol{\theta}_{b}}$ we use a number of $|\mathcal{O}| = 2$ networks made of a single hidden layer of 30 units, with ReLu activation function; while, $\pi_{hi}^{\boldsymbol{\theta}_{hi}}$ uses the same architecture but with 100 units. All the networks weights are initialized randomly with a uniform distribution $\mathcal{U}(-0.5, 0.5)$ at the beginning of each trial. In different trials, the algorithms are fed with a different number of demonstrations (training samples) and they are trained for the same amount of time, on the same hardware and in exactly the same conditions. We run all trials 30 times over different random seeds for the grid world and 10 times for the others. After the training is completed, we measure the average reward obtained over 100 episodes for each trial given a seed and finally, average again over the seeds. More information on the used hyperparameters are provided in the Appendix in [30]. The results are illustrated in Figure 2. As Figure 2 depicts, there is no tangible deterioration in the performance when using the online setting with respect to the batch. For the environments with a continuous state space, i.e., lunar lander, pendulum and cart pole, where $T \approx |\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}|$ we observe similar performance; on the other hand, for the grid-world, which has a discrete state space, the online algorithm outperforms its batch version since we have $T > |\tilde{\mathcal{S}}| \times |\tilde{\mathcal{A}}|$. Finally, note that these experiments are conducted on a reasonably small number of demonstrations T, in order to facilitate the comparison, and the training hyperparameters (Appendix in [30]) are selected such that the two algorithms perform an equivalent amount of gradient

steps. For greater T, the gap would have been larger since the forward-backward decomposition in the batch algorithm is more expensive. Overall, the results are encouraging and experiments on more realistic setups will be the subject of subsequent work.

VII. Conclusions

In this work, we develop an online version of the BW algorithm for HIL. Specifically, we formulate an online smoothing recursion suitable for the options framework and leverage it to obtain our online BW algorithm for HIL. In addition, we empirically compare online and batch versions on classical control tasks: the two algorithms show similar performance in all the environments with a continuous state-space; while, when the size of the training set becomes larger compared to the portion of the state-space explored by the expert, e.g the grid-world, we show the online algorithm to be convenient since the forward-backward decomposition used in the batch algorithm becomes more expensive than the online recursion in Proposition 2.

VIII. Acknowledgements

We thank Zhiyu Zhang for all the comments and useful discussions. This work was supported in part by NSF under grants DMS-1664644, CNS-1645681, IIS-1914792, by ARPA-E under grant DE-AR0001282, by the ONR under grant N00014-19-1-2571, and by the NIH under grant R01 GM135930.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in neural information processing systems*, 2018, pp. 3303–3313.

[3] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[4] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver, "Learning and transfer of modulated locomotor controllers," *arXiv preprint arXiv:1610.05182*, 2016.

[5] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, pp. 3675–3683, 2016.

[6] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1703.01161*, 2017.

[7] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] A. S. Vezhnevets, V. Mnih, J. Agapiou, S. Osindero, A. Graves, O. Vinyals, and K. Kavukcuoglu, "Strategic attentive writer for learning macro-actions," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 3494–3502.

[9] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots, "Fast policy learning through imitation and reinforcement," *arXiv preprint arXiv:1805.10413*, 2018.

[10] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[11] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[12] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg, "Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards," *arXiv preprint arXiv:1604.06508*, 2016.

[13] P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, and D. Precup, "Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning," *arXiv preprint arXiv:1709.06683*, 2017.

[14] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé III, "Hierarchical imitation and reinforcement learning," *arXiv preprint arXiv:1803.00590*, 2018.

[15] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv preprint arXiv:1810.11043*, 2018.

[16] R. Fox, R. Berenstein, I. Stoica, and K. Goldberg, "Multi-task hierarchical imitation learning for home automation," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 1–8.

[17] P. Sharma, D. Pathak, and A. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," in *Advances in Neural Information Processing Systems*, 2019, pp. 2597–2607.

[18] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[19] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[20] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, vol. 73, no. 3, pp. 360–363, 1967.

[21] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[22] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann, "Probabilistic inference for determining options in reinforcement learning," *Machine Learning*, vol. 104, no. 2-3, pp. 337–357, 2016.

[23] Z. Zhang and I. Paschalidis, "Provable hierarchical imitation learning via em," *arXiv preprint arXiv:2010.03133*, 2020.

[24] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg, "Multi-level discovery of deep options," *arXiv preprint arXiv:1703.08294*, 2017.

[25] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, no. 1, pp. 1–8, 1972.

[26] G. Mongillo and S. Deneve, "Online learning with hidden markov models," *Neural computation*, vol. 20, no. 7, pp. 1706–1716, 2008.

[27] O. Cappé, "Online em algorithm for hidden markov models," *Journal of Computational and Graphical Statistics*, vol. 20, no. 3, pp. 728–749, 2011.

[28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016, cite arxiv:1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540

[29] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[30] V. Giammarino and I. Paschalidis, "Online baum-welch algorithm for hierarchical imitation learning," *arXiv preprint arXiv:2103.12197*, 2021.

[31] O. Cappé, E. Moulines, and T. Rydén, *Inference in hidden Markov models*. Springer Science & Business Media, 2006.