# Unified Inter and Intra Options Learning Using Policy Gradient Methods

Kfir Y. Levy, Nahum Shimkin

Faculty of Electrical Engineering, Technion, Haifa 32000, Israel
{kfiryl@tx, shimkin@ee}.technion.ac.il

**Abstract.** Temporally extended actions (or macro-actions) have proven useful for speeding up planning and learning, adding robustness, and building prior knowledge into AI systems. The options framework, as introduced in Sutton, Precup and Singh (1999), provides a natural way to incorporate macro-actions into reinforcement learning. In the subgoals approach, learning is divided into two phases, first learning each option with a prescribed subgoal, and then learning to compose the learned options together. In this paper we offer a unified framework for concurrent inter- and intra-options learning. To that end, we propose a modular parameterization of intra-option policies together with option termination conditions and the option selection policy (inter options), and show that these three decision components may be viewed as a unified policy over an augmented state-action space, to which standard policy gradient algorithms may be applied. We identify the basis functions that apply to each of these decision components, and show that they possess a useful orthogonality property that allows to compute the natural gradient independently for each component. We further outline the extension of the suggested framework to several levels of options hierarchy, and conclude with a brief illustrative example.

## 1 Introduction

In complex planning problems it is often useful to utilize macro-actions, where every macro-action is a restricted plan or a policy, and compose these macro-actions in order to form an overall plan or a policy [1–3]. In the terminology of Sutton et al. [4], a macro action is called an *option* and the overall policy, composing these options, is called a *policy over options*. That article suggested how to compose existing options into a policy over options in a manner analogous to standard Reinforcement Learning (RL) algorithms, employing the Semi-Markov Decision process (SMDP) framework. An important problem is how to devise a good option. A popular solution is to define a subgoal task, which is a task that terminates if it arrives at a subgoal state, and an artificial reward is given for arriving there. A plan for the subgoal task can be learned using conventional RL methods, and the optimal policy, which is learned for the subgoal task is then used as an option which halts in a subgoal state. Thus, the learning procedure originally suggested in [4] divides the learning procedure into an intra-option learning phase, where every option is learned using subgoals, and to inter-option learning phase, where the composition of the options is learned, using standard RL algorithms applied to the SMDP framework. A possible shortcoming of this approach is that options are rigid and do not change during the inter-option learning phase. Sutton et al. [4] suggest a partial solution which they call interruption, which corresponds essentially to a single step of greedy policy improvement. Another interesting approach is to find subgoals online, during the learning phase, relying for example on reward or visit statistics of states [5, 6].

The options framework provides temporal abstraction of the planning and learning problem. A different approach towards abstraction of RL problems is to look for the best policy within a parameterized family $\{\pi_\theta\}_{\theta\in\Theta}$. The restriction to a parameterized family allows to introduce prior knowledge, limits the search space, and it allows use of gradient methods, [7–9] , in order to find a local maximum. Policy gradient algorithms have been successfully applied to various reinforcement learning problems in complex domains [8, 10, 11].

Previous work on parametric options includes [1–3]. In [1] gradient learning of the stopping condition is derived through embedding the options into the state space. In [2] the task is first manually decomposed into subtasks which are learned separately and only then the higher decision hierarchy is being optimized. In [3] the SMDP framework is utilized together with a certain regression method in order to learn parametric intra-policies. All of these works do not consider the problem of simultaneous optimization of the intra-options together with the overall policy, and only the first addresses the problem of learning the stopping conditions of options.

In this paper we are interested in schemes that can jointly learn the options policies (intra-option learning) and the policy-over-options (inter-option learning). We first formulate a framework in which the three components of the overall policy: intra-option policies, option termination conditions and the option selection policy, are viewed as a unified stationary policy over an augmented state-action space, to which standard RL algorithms may be applied. We next propose a modular parameterization of these three components, and identify the basis functions that apply to each of these policy components, we then show that these basis functions enjoy a useful orthogonality property that allows to compute the natural gradient independently for each component, leading to great computational savings.

The paper is constructed as follows: In Section 2 we describe the basic MDP model, and provide a brief overview of natural gradient methods and the options framework. In Section 3 we define the augmented state-action spaces and the augmented hierarchial policy (AHP). In Section 4 we propose a parameterization of the AHP for which we prove the orthogonality property of the parameterized base functions, and in Section 5 we briefly consider extending our framework to hierarchical multi-level options, Section 6 presents an illustrative example and in Section 7 we conclude the paper. Due to space limitations we leave some details of the example in Section 6 to the extended online version[1].

[1] http://webee.technion.ac.il/people/shimkin/PREPRINTS/LevyShimkinEWRL2011.pdf

## 2 Model and Background

A discounted-reward Markov Decision Process (MDP) $M$ may be defined by a 5 tuple $< S, A, r, P, \gamma >$. At time $t$, the system is in state $s_t \in S$, the agent chooses an action $a_t \in A$, a reward $r(s_t, a_t)$, is given, and the system transfers to a new state $s_{t+1}$, randomized according to a transition probability function, $P(\cdot|s_t, a_t)$. A policy $\pi = \{\pi_t\}_{t=0}^{\infty}$ is a series of mappings from the collection of possible histories to a probability function over the action space. A stationary policy depends only on the current state, and is therefore defined by the map $\pi : S \rightarrow \Delta(A)$, where $\Delta(A)$ is the probability simplex over the set $A$. For a given stationary policy $\pi$, the value function $V^\pi$ and the action-value function $Q^\pi$ are defined as:

$$V^\pi(s) = E^\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s] \tag{1}$$

$$Q^\pi(s, a) = r(s, a) + E^\pi[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a] \tag{2}$$

We assume that the initial state $s_0$ is chosen according to a fixed initial distribution $\eta(s)$. The general goal is to maximize the discounted return, defined as:

$$J(\pi) = \sum_{s\in S} \eta(s)V^\pi(s) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi(a|s)r(s, a) \tag{3}$$

where $d^\pi(s) = E_{s_0\sim\eta}[\sum_{t=0}^{\infty} \gamma^t p^\pi(s_t = s|s_0)]$.

### 2.1 Natural Policy Gradient

In the policy gradient framework [7], the policy itself is parameterized as $\pi(a|s) = \pi(a|s, \theta)$, and the vector $\theta$ is modified in the direction of gradient of expected return $J(\pi_\theta) \triangleq J(\theta)$. Plain methods of gradient learning estimate the value of $\nabla_\theta J(\theta)$ during the run, and modify the value of $\theta$ greedily in this direction. Natural gradient methods do not follow the steepest direction in parameter space, but rather the steepest direction with respect to the Fisher metric $G(\theta)$ [12, 9, 8], which leads to improved convergence properties. The relation between the standard gradient $\nabla_\theta J(\theta)$ and the natural gradient $\widetilde{\nabla}_\theta J(\theta)$ is given by:

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1}(\theta)\nabla_\theta J(\theta) \tag{4}$$

It is shown in [13] that the Fisher metric captures the geometric structure which is induced by the parametric family of policies. As shown in [7], for a stationary policy $\pi$, the gradient of the discounted return is given by:

$$\nabla_\theta J(\theta) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi(a|s)Q^\pi(s, a)\psi(s, a) \tag{5}$$

where $\psi(s,a) = \nabla_\theta \log \pi(a|s)$. Assuming $\theta \in R^N$, for every element $\theta^{(m)}$ in the parameter vector $\theta$ define:

$$\psi_m(s,a) = \frac{\partial \log \pi(a|s)}{\partial \theta^{(m)}} \tag{6}$$

We shall refer to $\{\psi_m\}_{m=1}^N$ as the $\theta$ *base functions.*
For functions $f, g : S \times A \to R$, consider the following inner product:

$$\langle f, g \rangle = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi(a|s) f(s,a) g(s,a) \tag{7}$$

Using the notation of this inner product, equation (5) can be written as:

$$\frac{\partial J(\pi)}{\partial \theta^{(m)}} = \langle Q^\pi, \psi_m \rangle \tag{8}$$

Hence, if we denote $\Psi_\theta = \mathrm{span}\{\psi_{\theta_1}, \ldots, \psi_{\theta_N}\}$, and $Q^\pi_{\Psi_\theta}(s,a)$ as the orthogonal projection of $Q^\pi(s,a)$ onto $\Psi_\theta$, the function $Q^\pi$ in equation (8) can be replaced with $Q^\pi_{\Psi_\theta}$. The orthogonal projection $Q^\pi_{\Psi_\theta}(s,a)$, by definition, could be written as

$$Q^\pi_{\Psi_\theta}(s,a) = \sum_{i=1}^N w_i \psi_i(s,a) = w^T \nabla_\theta \log \pi(a|s) \tag{9}$$

From the orthogonality principle we know that $\langle Q^\pi - Q^\pi_{\Psi_\theta}, \psi_i \rangle = 0$ for every $i$. From the last two equations one can obtain a linear equation for $w$

$$G_\theta w = c_\theta \tag{10}$$

where

$$\begin{aligned} G_\theta(i,j) &= \langle \psi_i, \psi_j \rangle, \qquad G_\theta \in R^{N \times N} \\ c_\theta(i) &= \langle Q^\pi, \psi_i \rangle, \qquad c_\theta \in R^N \end{aligned} \tag{11}$$

It is shown in [12, 9, 8] that $w$ is the natural gradient of $J(\theta)$. So, if we manage to estimate $G_\theta$ and $c_\theta$, we can calculate $w$ from equation (10) and gradient update $\theta$. In order to estimate the entries of the matrix $G_\theta$, we can simply use the temporal average of $\psi_i(s_t, a_t)\psi_j(s_t, a_t)$. As for the entries of $c_\theta$, it is demonstrated in [7] that if we define the advantage function $A^\pi(s,a) \triangleq Q^\pi(s,a) - V^\pi(s)$, we can replace the $Q^\pi$ function in equation (11) with the advantage function. The latter may be estimated with the time difference (TD) error:

$$\delta_t = r_t + V^\pi(s_{t+1}) - V^\pi(s_t) \tag{12}$$

which is an unbiased estimate of the advantage function. It is suggested in [8] to estimate the entries $c_\theta(i)$ with the temporal average of $\psi_i(s_t, a_t)\delta_t$. We still require an estimate of the value function $V^\pi$, which can be done with TD methods as in [9] or with the least square methods [14, 15]. Once we have an estimate of $G_\theta$ and $c_\theta$, we can calculate the natural gradient: $w = G_\theta^{-1} c_\theta$. We refer the reader to [9] for a complete description of the natural gradient learning algorithm.

### 2.2 The Options framework

An option $o$ is characterized by a 3-tuple $< \mathcal{I}, \pi, \beta >$, where $\mathcal{I} \subseteq S$ is the set from which the option can be initiated, $\beta : S \to [0,1]$ is a termination probability, and $\pi$ is the intra-option policy, which in general, may depend on the entire history since the option was initiated (but not before). Here we restrict attention to stationary intra-option policies. Note that a primitive action can be considered as a single step option. A policy-over-options is defined to be a mapping $\mu : S \to \Delta(\mathcal{O})$, where $\mathcal{O}$ is the set of all options, determines which option should be initiated in the current state. Given the the policy-over-options $\mu$ and options set $\mathcal{O}$ we refer to their combination as the overall policy (OP). We refer to intra-option policies, stopping conditions and policy-over-options as the three *decision components* of the OP. In [4] the SMDP framework is utilized in order to optimize the policy-over-options, $\mu$.

## 3 The Augmented Options Model

In this section we formulate an augmented MDP model that will enable us to utilize existing RL algorithms in order to learn simultaneously the three decision components of the overall policy (OP). In Subsection 3.1 we describe the decisions process made by the OP and in the following Subsection we define augmented state-action spaces and show that the OP in the original state-action spaces is equivalent to a *stationary* policy in the augmented spaces.
We consider a given OP $\{\mu, \mathcal{O}\}$, where $\mu$ is the policy-over-options and $\mathcal{O} = \{\pi_i, \beta_i\}_{i=1}^n$ is the options set. We index the individual option by $i$ or $j$, where $i, j \in \mathcal{O}$, and denote by $\pi_i$ and $\beta_i$ the intra policy and stopping condition of option $i$. Note the use of the same notation $i$ for the option itself and its index.

### 3.1 Overall Policy (OP) Description

The following is an outline of the decision process made by an OP:
1. At time $t$ the process arrives at a state $s_t$ with the option $i_t$, chosen at the previous step. We divide the choices of the policy to three decision phases:

- **Stopping decision phase** ($sp$): Choose whether or not to stop the current option, with the choice made according to the stopping probability $\beta_{i_t}(s_t)$. We relate to the decision to stop or not as an action, chosen from the binary action set $A_{\text{stop}} \triangleq \{stop, cont\}$.
- **Option decision phase** ($op$): Choose a new option $j_t$. If in the previous phase stopping was not chosen, the former option persists ($j_t = i_t$), otherwise, the policy-over-options chooses a new option $j_t$ according to $j_t \sim \mu(\cdot|s_t)$.
- **Action decision phase** ($ap$): Choose a new action $a_t$, according to the option $j_t$ chosen at the previous phase, i.e. $a_t \sim \pi_{j_t}(\cdot|s_t)$.

2. At time $t+1$ the process arrives at a new state $s_{t+1}$ under the policy $i_{t+1}$ chosen in the previous step. Again, three phases of decision will take place as in the previous step. Note that the following holds by definition: $i_{t+1} = j_t$, i.e., at the current step we arrive with the option we chose in the previous step.
We will relate to the option $i_t$ with which we arrive at a state $s_t$ as the *arrive*-option at time $t$, and the option $j_t$ which we choose in time $t$ as the *act*-option.

### 3.2 The Augmented Model

Given the original state and action spaces, $S, A$, and the overall policy $\{\mu, \mathcal{O}\}$, we define the augmented state and actions spaces, $\tilde{S}, \tilde{A}$ :

- **Augmented state** $\tilde{s} = (i, s)$, where $i \in \mathcal{O}$ is the the option with which we *arrive* at the state $s$.
- **Augmented action** $\tilde{a} = (\varphi, j, a)$, where $\varphi \in A_{\text{stop}}$ is the decision whether to stop the current *arrive*-option $i$. The action $j \in \mathcal{O}$ is the choice of the *act*-option, and $a \in A$ is the primitive action chosen by the *act*-option $j$.

State and option transitions in the original state space translate to state transitions in the augmented space. Given the original transition probabilities $P(s'|s,a)$ we can calculate the transition probability function in the augmented one:

$$P(\tilde{s}'|\tilde{s},\tilde{a}) = P((i',s')|(i,s),(\varphi,j,a)) = 1_{\{i'=j\}} P(s'|s,a) \tag{13}$$

Hence, the transition probability is the original $P(s'|s,a)$ if the next *arrive*-option $i'$ is equal to the current *act*-option $j$, otherwise it is 0. Note that $\varphi = \mathit{cont}$ implies $i = j$.
Given a overall policy $\{\mu, \mathcal{O}\}$ in the MDP $M$, we define the following policy $\Pi$ in the augmented space:

$$\Pi(\tilde{a}|\tilde{s}) = \Pi((\varphi,j,a)|(i,s)) = P_{sp}(\varphi|i,s) P_{op}(j|\varphi,i,s) P(a|j,s) \tag{14}$$

where

$$P_{sp}(\varphi|i,s) = 1_{\{\varphi=stop\}} \beta_i(s) + 1_{\{\varphi=cont\}} (1-\beta_i(s)) \tag{15}$$
$$P_{op}(j|\varphi,i,s) = 1_{\{\varphi=cont\}} 1_{\{j=i\}} + 1_{\{\varphi=stop\}} \mu(j|s) \tag{16}$$
$$P(a|j,s) = \pi_j(a|s) \tag{17}$$

We relate to $\Pi$ as the augmented hierarchical policy (AHP). Notice that $\Pi(\tilde{a}|\tilde{s})$ depends only on $(\tilde{a},\tilde{s})$ and is therefore stationary. The term $P_{sp}(\varphi|i,s)$ is the probability that the current *arrive*-option $i$ will stop at the current state $s$. The next term $P_{op}(j|\varphi,i,s)$ is the probability that an *act*-option $j$ will be chosen given the current state and stopping action, and the last term $P(a|j,s)$ is the probability that a new action $a$ is chosen in state $s$, which is done according to the policy of the *act*-option, $j$. In equations (15)-(17), we relate the equivalent policy $\Pi$ to the policy-over-options $\mu$ and options set $\mathcal{O} = \{\pi_i, \beta_i\}_{i=1}^n$. Notice that we can design $\mu$ to be also dependent at the *arrive*-option, i.e., $\mu = \mu(j|i,s)$.

It is easy to show that the policy $\Pi$ over the augmented spaces is equivalent to the overall policy $\{\mu, \mathcal{O}\}$, over the original MDP, in the sense that both induce the same probability measure over augmented state-actions trajectories (which include stopping decisions and option choices), $\{i_t, s_t, \varphi_t, j_t, a_t\}_{t=0}^{T}$ particularly:

$$E^{\{\mu,\mathcal{O}\}}[\sum_{t=0}^{\infty}\gamma^t r_t|s_0, i_0] = E^{\Pi}[\sum_{t=0}^{\infty}\gamma^t r_t|(i_0, s_0)] \tag{18}$$

Therefore, in order to simultaneously learn the three decision components of the overall policy (stopping, intra policies, policy-over-options), we can directly apply standard RL methods to the augmented hierarchical policy $\Pi$. In particular, we can apply natural gradient methods which we mentioned in Subsection 2.1, in the next Section we investigate natural gradient learning of the AHP, $\Pi$.

## 4 Natural Gradient of the AHP

In this Section we offer a modular parameterization of the AHP $\Pi$, for which we prove an orthogonality property (Proposition 1), which enables us to substantially reduce the computational burden of calculating the natural gradient for the parameterized AHP (Corollary 1).

The special structure of the AHP $\Pi$ in equations (14)-(17) suggests the following parameterization:

$$\begin{aligned} \beta_i(s) &= \beta_i(s, \lambda_i) \\ \mu(j|s) &= \mu(j|s, \chi) \\ \pi_j(a|s) &= \pi_j(a|s, \theta_j) \end{aligned} \tag{19}$$

with the parameter vector

$$\Theta = (\theta_1, \theta_2 \ldots, \theta_N, \lambda_1, \lambda_2 \ldots, \lambda_N, \chi) \tag{20}$$

The parameter vector $\Theta$ is composed of three types of sub-vector parameters: $\theta_i$ controls the stationary policy of option $i$, $\lambda_i$ controls the stopping of the $i^{th}$ policy, and $\chi$ controls the choices of the policy over options $\mu$.

Generally, in order to calculate the natural gradient (equation (10)), we should invert an $N \times N$ matrix, where $N = \dim\{\Theta\}$.

In what follows we denote by $N_{\theta_m}, N_{\lambda_k}, N_{\chi}$ the dimensions of the vectors $\theta_m, \lambda_k, \chi$, and by $\theta_j^{(h)}$ the $h^{th}$ element of the sub-vector $\theta_j$.

**Proposition 1** *Let $\{\mu, \mathcal{O}\}$ be an overall policy with stationary options set $\mathcal{O} = \{\pi_i, \beta_i\}_{i=1}^{n}$, and let $\Pi$ be its equivalent augmented hierarchical policy as in equations* (14)-(17)*, parameterized as in equations* (19)-(20)*. Define the following* *linear subspaces of the $\Theta$-base functions from equation* (6)*:*

$$\begin{aligned}
\Psi_{\theta_m} &= \mathrm{span}\{\psi_{\theta_m^{(h)}}\}_{h=1}^{N_{\theta_m}} \qquad \forall m \in \mathcal{O} \\
\Psi_{\lambda_k} &= \mathrm{span}\{\psi_{\lambda_k^{(h)}}\}_{h=1}^{N_{\lambda_k}} \qquad \forall k \in \mathcal{O} \\
\Psi_{\chi} &= \mathrm{span}\{\psi_{\chi^{(h)}}\}_{h=1}^{N_{\chi}} \qquad (21)
\end{aligned}$$

*Then these subspaces are orthogonal under the inner product defined in* (7)*.*

Proof: We may calculate the base function for the elements of the parameter vector $\Theta$ as follows:

$$\psi_{\lambda_k^{(h)}} = \frac{\partial \log \Pi((\varphi,j,a)|(i,s),\Theta)}{\partial \lambda_k^{(h)}} = 1_{\{i=k\}} \frac{1}{P_{sp}(\varphi|i,s,\lambda_i)} \frac{\partial P_{sp}(\varphi|i,s,\lambda_i)}{\partial \lambda_i^{(h)}} \qquad (22)$$

$$\psi_{\chi^{(h)}} = \frac{\partial \log \Pi((\varphi,j,a)|(i,s),\Theta)}{\partial \chi^{(h)}} = 1_{\{\varphi=stop\}} \frac{1}{P_{op}(j|\varphi,i,s)} \frac{\partial \mu(j|s,\chi)}{\partial \chi^{(h)}} \qquad (23)$$

$$\psi_{\theta_m^{(h)}} = \frac{\partial \log \Pi((\varphi,j,a)|(i,s),\Theta)}{\partial \theta_m^{(h)}} = 1_{\{j=m\}} \frac{\partial \log \pi_j(a|s,\theta_j)}{\partial \theta_j^{(h)}} \qquad (24)$$

It is easy to see that the indicator functions in (22)(24) imply that the following orthogonality relations hold:

$$\langle \psi_{\lambda_m^{(f)}}, \psi_{\lambda_k^{(h)}} \rangle = 0 \qquad \forall m \neq k \qquad (25)$$

$$\langle \psi_{\theta_m^{(f)}}, \psi_{\theta_k^{(h)}} \rangle = 0 \qquad \forall m \neq k \qquad (26)$$

Less trivial are the following relations:

$$\langle \psi_{\theta_m^{(f)}}, \psi_{\lambda_k^{(h)}} \rangle = 0, \;\; \langle \psi_{\theta_m^{(f)}}, \psi_{\chi^{(h)}} \rangle = 0 \qquad \forall k, m \qquad (27)$$

$$\langle \psi_{\chi^{(f)}}, \psi_{\lambda_k^{(h)}} \rangle = 0 \qquad \forall k \qquad (28)$$

Let us prove (27) first :

$$\begin{aligned}
&\langle \psi_{\theta_m^{(f)}}, \psi_{\lambda_k^{(h)}} \rangle = \sum_{\tilde{s} \in \tilde{S}} d^{\Pi}(\tilde{s}) \sum_{\tilde{a} \in \tilde{A}} \Pi(\tilde{a}|\tilde{s}) \psi_{\theta_m^{(f)}} \psi_{\lambda_k^{(h)}} = \\
&\sum_{i,s} d^{\Pi}(i,s) \sum_{\varphi,j,a} \Pi(\tilde{a}|\tilde{s}) 1_{\{j=m\}} \frac{\partial \log \pi_j(a|s,\theta_j)}{\partial \theta_j^{(f)}} 1_{\{i=k\}} \frac{\partial \log P_{sp}(\varphi|i,s,\lambda_i)}{\partial \lambda_i^{(h)}} = \\
&\sum_{s} d^{\Pi}(k,s) \sum_{\varphi,a} P_{op}(m|\varphi,k,s,\chi) \frac{\partial P_{sp}(\varphi|k,s,\lambda_k)}{\partial \lambda_k^{(h)}} \frac{\partial \pi_m(a|s,\theta_m)}{\partial \theta_m^{(f)}} = \\
&\sum_{s} d^{\Pi}(k,s) \sum_{\varphi} P_{op}(m|\varphi,k,s,\chi) \frac{\partial P_{sp}(\varphi|k,s,\lambda_k)}{\partial \lambda_k^{(h)}} \frac{\partial}{\partial \theta_m^{(f)}} \sum_{a} \pi_m(a|s,\theta_m) = 0
\end{aligned}$$

where in the last step we used the identity $\sum_a \pi_m(a|s) = 1$. The proof of $\langle \psi_{\theta_m^{(f)}}, \psi_{\chi^{(h)}} \rangle = 0$ is similar. We next prove equation (28):

$$\langle \psi_{\chi^{(f)}}, \psi_{\lambda_k^{(h)}} \rangle = \sum_{\tilde{s} \in \tilde{S}} d^{\Pi}(\tilde{s}) \sum_{\tilde{a} \in \tilde{A}} \Pi(\tilde{a}|\tilde{s}) \psi_{\chi^{(f)}} \psi_{\lambda_k^{(h)}} =$$

$$\sum_{i,s} d^{\Pi}(i,s) \sum_{\varphi,j,a} \Pi(\tilde{a}|\tilde{s}) \frac{1_{\{\varphi=stop\}}}{P_{op}(j|\varphi,i,s)} \frac{\partial \mu(j|s,\chi)}{\partial \chi^{(h)}} 1_{\{i=k\}} \frac{\partial \log P_{sp}(\varphi|i,s,\lambda_i)}{\partial \lambda_i^{(h)}} =$$

$$\sum_{s \in S} d^{\Pi}(k,s) \sum_{j,a} \pi_j(a|s,\theta_j) \frac{\partial \mu(j|s,\chi)}{\partial \chi^{(h)}} \frac{\partial P_{sp}(stop|k,s,\lambda_k)}{\partial \lambda_k^{(h)}} =$$

$$\sum_{s} d^{\Pi}(k,s) \frac{\partial P_{sp}(stop|k,s,\lambda_k)}{\partial \lambda_k^{(h)}} \frac{\partial}{\partial \chi^{(h)}} \{ \sum_j \mu(j|s,\chi) \sum_a \pi_j(a|s,\theta_j) \} = 0$$

where in the last step we used the identities $\sum_a \pi_j(a|s) = 1$, $\sum_j \mu(j|s) = 1$. □
Given the last orthogonality result, it is clear that the matrix $G_\Theta$ in (11) becomes block diagonal with each block corresponding to a component of $\Theta$.

**Corollary 1** *Given a parameterization of the AHP $\Pi$ as in equations* (19)-(20)*, we can estimate each sub-vector of the natural gradient $w$ independently as follows: we can write an independent equation with the form of* (10) *for the relevant sub vector $w_{\theta_m}, w_{\lambda_k}, w_\chi$. The $w_{\theta_m}$ equations:*

$$G_{\theta_m} w_{\theta_m} = c_{\theta_m} \tag{29}$$

*where:*

$$G_{\theta_m}(i,j) = \langle \psi_{\theta_m^{(i)}}, \psi_{\theta_m^{(j)}} \rangle \qquad G_{\theta_m} \in R^{N_{\theta_m} \times N_{\theta_m}} \tag{30}$$

$$c_{\theta_m}(i) = \langle Q^{\pi}(s,a), \psi_{\theta_m^{(i)}} \rangle \qquad c_{\theta_m} \in R^{N_{\theta_m}} \tag{31}$$

*Similar equations apply to $w_{\lambda_k}, w_\chi$.*

Thus, in order to calculate the natural gradient vector, it is sufficient to invert $2n+1$ Fisher matrices, where $n$ denotes the number of options, each matrix has the dimension of the corresponding sub-vector.

## 5 Multilevel Decision Hierarchies

In the previous sections we have considered a hierarchical structure with one level of options below the top (policy-over-options) level, so that every low level option chooses primitive actions. In this section we briefly consider a multi-level hierarchical structure, in which higher-level options can treat options from a lower level as extended actions. We describe options with only two levels of hierarchy, as the extension to structures with more levels of hierarchy is similar.

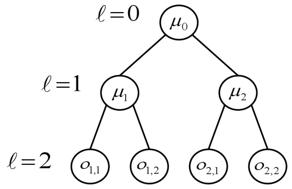


Fig. 1: Hierarchical options structure.

In figure 1 we illustrate the hierarchical structure of an overall policy with two levels of hierarchy, where root option is denoted by $\mu_0$, the first level option-nodes are denoted by $\mu_1, \mu_2$ and beneath every node $i$ in the 1st level lie the leaf options $o_{i,n}$ whose policies are stationary. Similarly to section 3, we can divide the choices made by this hierarchical policy into 5 decision phases as follows. Suppose at time t the process arrives a state $s_t$ with a leaf option $o_{i_t,n_t}$:

1. Decide whether to stop the current leaf option.
2. The hierarchy above, $\mu_{i_t}$, makes a decision whether to stop the execution of the current level 1 decision node.
3. The root policy $\mu_0$ chooses a new level 1 decision node, $\mu_{j_t}$.
4. The level 1 hierarchy, $\mu_{j_t}$ chooses a new leaf option $o_{j_t,m_t}$
5. The new leaf option chooses a primitive action $a_t \sim \pi_{j_t,m_t}(\cdot|s_t)$

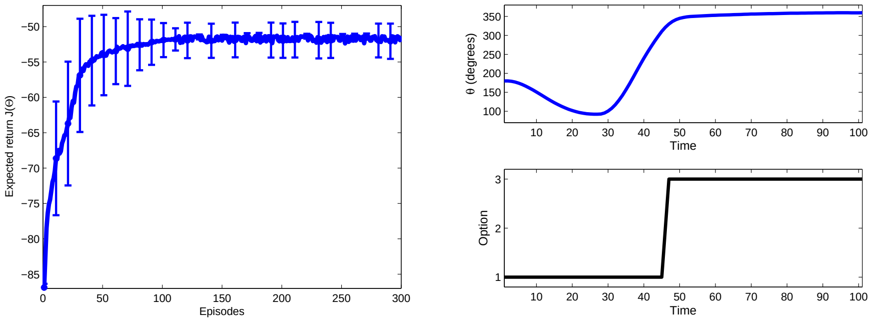
As in section 3, we can define corresponding augmented state and action spaces and an equivalent augmented hierarchical policy above these augmented spaces, it can be shown that the orthogonality property between the basis function of the various policy components (Section 4) will be maintained. We omit the details here due to space constraints.

## 6 Experimental Results – Inverted Pendulum

The inverted pendulum task is a known RL benchmark in which we have to swing up an inverted pendulum from the down position to the up-position and keep it stable. We have two state variables, the angle and it derivative,namely $s = (\theta, \dot{\theta})$. We can apply a limited torque $|a| \leq a_{max}$ at the rotary joint. Motivated by [16] where multiple controllers are designed based on traditional control theory and then combined using an RL scheme, we chose to use the following three *parameterized options* in order to learn an effective control policy:

1. Swinging Option: $a_1 \sim \mathcal{N}(k_1 \text{sign}(\dot{\theta}), \sigma_1^2)$.
2. Decelerating Option: $a_2 \sim \mathcal{N}(-k_2 \text{sign}(\dot{\theta}), \sigma_2^2)$.
3. Stabilizing Option $a_3 \sim \mathcal{N}(k_3\theta + k_4\dot{\theta}, \sigma_3^2)$.

Here $\mathcal{N}(\mu, \sigma^2)$ is gaussian random variable with a fixed variance $\sigma^2$. We further parameterized the options stopping conditions. For instance, the stopping condition of the swinging option is: $\beta_1(s) = \Phi(\frac{\theta-(\pi+\lambda_1)}{\sigma_{sp}}) + \Phi(\frac{-(\theta-(\pi-\lambda_1))}{\sigma_{sp}})$, where $\Phi$ is the cumulative normal distribution[2]. We optimize over the action parameters and stopping parameters:$\{k_i, \lambda_i\}_{i=1}^4$. Results are illustrated in Fig.2. From Fig. 2a, we can see that our algorithm converges within 90 episodes, where in most of the cases it converges within less 40. Interestingly our algorithm eventually neglects the decelerating option, as shown in Figure 2b.



(a) (b)

Fig. 2: Learning curves for the pendulum task. In (a) we can see the expected return averaged over 20 runs. In (b),we can see the options selection and $\theta$ as a function of time for the converged parameters .

## 7 Concluding Remarks

The unified framework that we propose enables to take a structure of overall policy and simultaneously optimize the intra-option policies, the stopping conditions and the policy-over-options. However, the simultaneous optimization of all parameters is only optional, and our framework allows one to freeze some of the parameters while learning the others. For example, we could think of first learning each intra-option policy separately (possibly using subgoals), and then keep the policies parameters fixed while optimizing over the stopping conditions and the composition of options, and only at the final phase optimize all parameters simultaneously. Comparing simultaneous verses alternating optimization schedules is an interesting direction for further research.
The possibility to tune the stopping conditions enables us to obtain "smoother"

[2] For the stopping conditions of the other options as well as further simulation details see the extended online version

transitions in between the options, as opposed to the subgoals approach, where the termination conditions are rigid.
As shown, the orthogonality property between the basis function of the different policy components leads to significant simplification in the computation of the natural gradient. In future research, one could look for similar computational leverages in other learning algorithms such as second-order methods for parameter tuning, and least squares methods for the value function evaluation [14, 15]. More generally, additional theoretical and empirical work on parametric options learning is evidently called for.

## References

1. G. Comanici and D. Precup, "Optimal policy switching algorithms for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 2010, pp. 709–714.
2. M. Ghavamzadeh and S. Mahadevan, "Hierarchical policy gradient algorithms," *Twentieth International Conference on Machine Learning*, pp. 226–233, 2003.
3. G. Neumann, W. Maass, and J. Peters, "Learning complex motions by sequencing simpler motion templates," in *International Conference on Machine Learning*, 2009.
4. R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, pp. 181–211, 1999.
5. O. Simsek and A. Barto, "Using relative novelty to identify useful temporal abstractions in reinforcement learning," in *International Conference on Machine Learning*, vol. 21. Citeseer, 2004, p. 751.
6. I. Menache, S. Mannor, and N. Shimkin, "Q-cutdynamic discovery of sub-goals in reinforcement learning," *Machine Learning: ECML 2002*, pp. 187–195, 2002.
7. R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 2000.
8. J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.
9. S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," *Automatica*, vol. 45, pp. 2471–2482, 2009.
10. S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," *Advances in neural information processing systems*, vol. 19, p. 1169, 2007.
11. O. Buffet, A. Dutech, and F. Charpillet, "Shaping multi-agent systems with gradient reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, 2007.
12. S. Kakade, "A natural policy gradient," *Advances in Neural Information Processing Systems 14*, vol. 2, pp. 1531–1538, 2002.
13. J. Bagnell and J. Schneider, "Covariant policy search," in *International Joint Conference on Artificial Intelligence*, vol. 18. Citeseer, 2003, pp. 1019–1024.
14. J. Boyan, "Technical update: Least-squares temporal difference learning," *Machine Learning*, vol. 49, pp. 233–246, 2002.
15. A. Nedić and D. Bertsekas, "Least squares policy evaluation algorithms with linear function approximation," *Discrete Event Dynamic Systems*, vol. 13, 2003.
16. J. Yoshimoto, M. Nishimura, Y. Tokita, and S. Ishii, "Acrobot control by learning the switching of multiple controllers," *Artificial Life and Robotics*, vol. 9, 2005.