Hierarchical model-based policy optimization: from actions to action sequences and back

Daniel McNamee
University College London
d.mcnamee@ucl.ac.uk

Abstract

We develop a normative framework for hierarchical model-based policy optimization based on applying second-order methods in the space of all possible state-action paths. The resulting *natural path gradient* performs policy updates in a manner which is sensitive to the long-range correlational structure of the induced stationary state-action densities. We demonstrate that the natural path gradient can be computed exactly given an environment dynamics model and depends on expressions akin to higher-order successor representations. In simulation, we show that the priorization of local policy updates in the resulting policy flow indeed reflects the intuitive state-space hierarchy in several toy problems.

1 Introduction

Reinforcement learning algorithms can leverage internal models of environment dynamics to facilitate the development of good control policies [1]. Dynamic programming methods iteratively implement one-step, full-width backups in order to propagate reward information across a state-space representation and then use this information to perform policy updates [2]. Stochastic approximations of this approach underpin a wide range of model-free reinforcement learning algorithms which can be enhanced by the ability to query samples from an "internal" environment model as in the DYNA architecture [3]. State-space search strategies apply heuristic principles to efficiently sample multi-step paths from internal models and have formed a core component of recent state-of-the-art game playing agents [4]. Model-based policy search algorithms can use paths sampled from a model in order to approximate policy gradients [5]. Such methods rely on alternating between simulating paths over various horizons and then using this information to improve the policy either directly or based on bootstrapped value estimates [1]. In this study, we introduce a model-based policy optimization procedure which normatively improves policies in a manner sensitive to the distribution of all future paths without requiring simulations or cached value functions. In contrast to dynamic programming techniques [1], this algorithm has the normative feature that it generates the unique trajectory through policy space which iteratively maximizes the expected cumulative reward obtained. Therefore, it obtains the maximum amount of cumulative reward on each algorithmic step.

In our analysis, the central object of interest is not a state-action pair, as is the standard perspective in reinforcement learning in discrete MDPs, but complete state-action sequences or *paths* [6–8]. We show that the cumulative reward objective can be re-written in terms of a log-likelihood over paths and so gradient ascent in the space of policies over paths integrates information over all possible future paths in expectation on each step. We demonstrate that, when performing Newton steps, both the path gradient and the path likelihood Hessian reflect the correlational structure of the policy path density between state-actions across time. In Section 2, we summarize the mathematical framework of discrete Markov decision processes in the sum-over-paths formalism and define our notation. In Section 3, we derive our hierarchical model-based policy optimization algorithm (HIMO). In Section 4, we apply the algorithm in simple state-spaces drawn from a variety of domains and

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

interrogate the resulting policy optimization dynamics. We conclude with a brief discussion in Section 5.

2 Background and notation

We consider discounted infinite-horizon Markov decision processes defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, s_\emptyset)$ comprised of a states-space $\mathcal{S}$, an action-space $\mathcal{A}$, a dynamics tensor P, a reward function R, and an initial state $s_\emptyset$. The action set $\mathcal{A}$ is the union of the sets of actions available at each state $\mathcal{A} = \cup_{s_i \in \mathcal{S}} \mathcal{A}_i$. In each state $s_i \in \mathcal{S}$, the agent selects actions $a_j \in \mathcal{A}_i$ according to a stochastic policy[1] $\pi(a_j|s_i) \equiv \pi_{ij}$ resulting in a stochastic state transition governed by the dynamics tensor $P(s_k|s_i, a_j) \equiv p_{ijk}$ and the receipt of a reward $R(s_i, a_j, s_k) \equiv R_{ijk}$. Bold-typed notation, $\mathbf{s} \in \boldsymbol{\mathcal{S}}$, $\mathbf{a} \in \boldsymbol{\mathcal{A}}$, and $\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}$ denotes sequences of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, and state-actions $\tau \equiv (s_\tau, a_\tau) \in \mathcal{T}$ respectively. A valid (i.e. possible) state-action sequence $\boldsymbol{\tau} := (\ldots, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots)$ is referred to as a *path*. The path probability $\mathbf{p}(\boldsymbol{\tau})$ is defined as the joint distribution over states $\mathbf{s}$ and actions $\mathbf{a}$

$$\mathbf{p}(\boldsymbol{\tau}) \quad := \quad \prod_{t=0}^{\infty} \mathrm{p}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t) = \mathbf{p}(\mathbf{s}_{+1}|\mathbf{s}, \mathbf{a})\boldsymbol{\pi}(\mathbf{a}|\mathbf{s}) \tag{1}$$

where

$$\boldsymbol{\pi}(\mathbf{a}|\mathbf{s}) \quad := \quad \prod_{t=0}^{\infty} \pi(\mathbf{a}_t|\mathbf{s}_t) \quad , \quad \mathbf{p}(\mathbf{s}_{+1}|\mathbf{s}, \mathbf{a}) := \prod_{t=0}^{\infty} \mathrm{p}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \ . \tag{2}$$

A discount parameter γ is embedded in the environment dynamics tensor P implying that an episode may end on every time step with probability $1 - \gamma$. We represent the policy in terms of natural parameters A_{ij} in an exponential parameterization $A_{ij} := \log \pi_{ij}$. These natural parameters are examples of action preferences in reinforcement learning parlance [1]. In general, $A_{ij}(\theta)$ may be parametrized and our formalism extended to policy gradients in θ-space which we will study in future work. In this manuscript, we consider the action preferences as parameters themselves in order to focus on theory exposition and demonstrations of hierarchical processing during planning.

To ensure that the policy probabilities $\pi_{ij} = e^{A_{ij}}$ take positive values, the action preferences $A_{ij} \in \mathbb{R}^-$ are constrained to take negative real values. To ensure that the set of policy probabilities forms a normalized density at each state, we eliminate a redundant action preference at each state. This is accomplished by defining an arbitrary transition probability at each state in terms of the probabilities of alternative transitions at that state. We index this dependent action preference using an ω subscript as in A_{ii_ω} in order to distinguish it from the independent action preferences which will be directly modified during policy optimization. We define i_ω as the action index of an arbitrary action available in state s_i. Due to the fact that a policy must be normalized at each state, we have

$$\pi_{ii_\omega} \quad = \quad 1 - \sum_{a_{i_\omega} \neq a_j \in \mathcal{A}_i} \pi_{ij} \ . \tag{3}$$

Under this local policy normalization constraint, the "fixed" action preferences are equivalently constrained via

$$A_{ii_\omega} \quad = \quad \log \left(1 - \sum_{a_{i_\omega} \neq a_j \in \mathcal{A}_i} e^{A_{ij}} \right) \ . \tag{4}$$

Given a complete action preference parametrization $A := (\ldots, A_{ij}, \ldots)_{s_i \in \mathcal{S}, a_j \in \mathcal{A}_j}$, the path policy $\boldsymbol{\pi}(\boldsymbol{\tau}) \equiv \boldsymbol{\pi}(\mathbf{a}|\mathbf{s})$ and the path density $\mathbf{p}(\boldsymbol{\tau})$ have the following log-linear forms

$$\log \boldsymbol{\pi}(\boldsymbol{\tau}) \quad = \quad A \cdot n(\boldsymbol{\tau}) = \sum_{\substack{s_i \in \mathcal{S} \\ a_j \in \mathcal{A}_j}} A_{ij} n_{ij}(\boldsymbol{\tau}) \tag{5}$$

$$\log \mathbf{p}(\mathbf{s}_{+1}|\mathbf{s}, \mathbf{a}) \quad = \quad C \cdot n(\boldsymbol{\tau}) = \sum_{\substack{s_i, s_k \in \mathcal{S} \\ a_j \in \mathcal{A}_i}} C_{ijk} n_{ijk}(\boldsymbol{\tau}) \tag{6}$$

$$\log \mathbf{p}(\boldsymbol{\tau}) \quad = \quad A \cdot n(\boldsymbol{\tau}) + C \cdot n(\boldsymbol{\tau}) \tag{7}$$

[1] The policy converges to the optimal deterministic policy asymptotically.

where n_{ijk} counts the number of occurrences of the state-action-state event (s_i, a_j, s_k) in the path $\boldsymbol{\tau}$ and $C_{ijk} := \log p_{ijk}$. Considering the set of probabilities $e^{(A+C)\cdot n(\boldsymbol{\tau})}$ parametrized by A as an exponential family [9], the vector $n(\boldsymbol{\tau})$ of transition counters $n_{ijk}(\boldsymbol{\tau})$ constitutes a sufficient statistic for the path $\boldsymbol{\tau}$. These counter variables will be crucial in computing gradients with respect to the stationary state-action density induced by a given policy. This, in turn, will facilitate a novel *path gradient* relating policy gradients across distinct states. We now turn to optimizing the policy objective.

3 Second-order policy optimization in space of paths

The cumulative expected reward objective as a sum-over-paths [8] is

$$\begin{aligned} A^* &:= \operatorname*{argmax}_A \mathcal{R}(A) \\ \mathcal{R}(A) &:= \left\langle \mathbf{R}(\boldsymbol{\tau}) \right\rangle_{\mathbf{p}} \qquad (8) \\ \mathbf{R}(\boldsymbol{\tau}) &= \sum_{t=0}^{\infty} \mathrm{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \qquad (9) \end{aligned}$$

where the angled brackets $\langle\cdot\rangle_{\mathbf{p}}$ denote the expectation operation over the path density $\mathbf{p}$. Consider the gradient $\nabla_A \mathcal{R}$ of $\mathcal{R}$ with respect to the action preferences A. Leveraging "likelihood trick" frequently used in policy gradient methods, we have that

$$\begin{aligned} \nabla_A \mathcal{R}(A) &= \sum_{\boldsymbol{\tau}\in\boldsymbol{\mathcal{T}}} \nabla_A \mathbf{p}(\boldsymbol{\tau}) \mathbf{R}(\boldsymbol{\tau}) \qquad (10) \\ &= \sum_{\boldsymbol{\tau}\in\boldsymbol{\mathcal{T}}} \mathbf{p}(\boldsymbol{\tau}) \left[\nabla_A \log \mathbf{p}(\boldsymbol{\tau})\right] \mathbf{R}(\boldsymbol{\tau}) \qquad (11) \\ &= \left\langle \nabla_A \log \left[\mathbf{p}(\boldsymbol{\tau}) e^{\mathbf{R}(\boldsymbol{\tau})}\right] \right\rangle_{\mathbf{p}} . \qquad (12) \end{aligned}$$

It is observed that the gradient $\nabla_A \mathcal{R}(A)$ has the form of a score function in expectation over paths with $\mathcal{L}_{\boldsymbol{\tau}}(A) = \mathbf{p}(\boldsymbol{\tau}) e^{\mathbf{R}(\boldsymbol{\tau})}$ playing the role of a path likelihood function.

Our theoretic hypothesis is that the curvature information utilized in a second-order optimization procedure [10] of the path-based objective (Eqn. 9) will enforce policy updates which are sensitive to the state-action correlational structure over all horizons embedded in the path density $\mathbf{p}(\boldsymbol{\tau})$. Therefore, we implement the Newton step

$$A^{t+1} \leftarrow A^t + \left\langle \mathrm{H}_{\mathcal{L}_{\boldsymbol{\tau}}}\left(A^t\right) \right\rangle_{\mathbf{p}}^{-1} \nabla_A \mathcal{R}\left(A^t\right) . \qquad (13)$$

where $\left\langle \mathrm{H}_{\mathcal{L}_{\boldsymbol{\tau}}}\left(A^t\right) \right\rangle_{\mathbf{p}}$ is the expected Hessian of the path likelihood function $\mathcal{L}_{\boldsymbol{\tau}}(A)$ at the current action preferences A^t. As a trust region method [10], this can be interpreted as optimizing a gradient step ΔA under a local approximation of $\mathcal{R}(A^t)$:

$$\Delta A^* = \operatorname*{argmax}_{\Delta A} \left[\nabla_A \mathcal{R}\left(A^t\right) \cdot \Delta A + \frac{1}{2} ||\Delta A||^2_{\left\langle \mathrm{H}_{\mathcal{L}_{\boldsymbol{\tau}}}(A^t)\right\rangle_{\mathbf{p}}}\right] \qquad (14)$$

where $||\cdot||_{\left\langle \mathrm{H}_{\mathcal{L}_{\boldsymbol{\tau}}}(A^t)\right\rangle_{\mathbf{p}}}$ is the expected Hessian norm of the path likelihood function $\mathcal{L}_{\boldsymbol{\tau}}$ at A^t. It can be shown [11, 12] that the expectation of the likelihood Hessian (in this case over paths) is equivalent to a Fisher information expression $\left\langle \mathrm{H}_{\mathcal{L}_{\boldsymbol{\tau}}}(A) \right\rangle_{\mathbf{p}} \equiv \mathcal{I}(A)$ with components[2]

$$[\mathcal{I}(A)]_{ij,kl} := \left\langle \left[\partial_{A_{ij}} \log \mathbf{p}(\boldsymbol{\tau})\right] \left[\partial_{A_{kl}} \log \mathbf{p}(\boldsymbol{\tau})\right] \right\rangle_{\mathbf{p}} . \qquad (15)$$

Therefore, the *natural path gradient* step

$$A^{t+1} \leftarrow A^t + \mathcal{I}^{-1} \nabla_A \mathcal{R}\left(A^t\right) \qquad (16)$$

is equivalent to the Newton step (Eqn. 13) and is optimal with respect to the local path-based approximation of the cumulative reward objective (Eqn. 14). This planning-as-optimization scheme

[2]The Fisher information matrix $\mathcal{I}$ for the path likelihood is derived in Section 7.2 of the Supplementary Material (SM).

is covariant with respect to the natural gradient of the path probability $\boldsymbol{\pi}$ parametrization of the cumulative reward objective $\mathcal{R}$. Since $\mathcal{R}$ is convex in $\boldsymbol{\pi}$ [13], Eqn. 13 can be expected to converge to the globally optimal policy and does so in all simulated problems. In the SM (Section 7), we show that exact expressions for all the quantities required to implement this policy optimization procedure can be derived analytically. Notably, the policy path gradient is a function of state-action correlation functions which are themselves composed of products of successor representations [14]. Based on these derivations, the complete algorithm which implements the hierarchical model-based path gradient updates (Eqn. 13) is elaborated in subsection 7.4 (SM).

4 Simulations

We simulate hierarchical model-based policy optimization (Eqn. 27) in two toy environments in order to gain insight into the dynamics of the policy optimization process and demonstrate its hierarchical sensibility. Specifically, the Tower of Hanoi problem and a four-room grid world environment. In all simulations, we fix the foresight $\lambda = 0.99$ (thus simulating an "expert" planner with "deep" foresight), the stepsize $\alpha = 0.001$ (chosen so that the policy evolution process is tracked at high precision for the purposes of visualization). After running HIMO until convergence, its dynamics are interrogated using two measures. The first measure is the KL-divergence between the policy densities at each iteration π^t and the prior policy π^0. We compute this *policy divergence* measure PD locally at each state $s \in \mathcal{S}$:

$$\mathrm{PD}(s,t) := \mathrm{KL}\left[\pi^t_{s\cdot} || \pi^0_{s\cdot}\right] \tag{17}$$

Policy divergence quantifies the degree to which the algorithm is modifying the local policy at each state as a function of planning time. The second measure is the difference between the expected number of times a state will be occupied under the currently optimized policy versus the prior policy. Specifically, the *counter difference* measure CD is

$$\mathrm{CD}(s,t) := D^t_{\emptyset s} - D^0_{\emptyset s} \ . \tag{18}$$

where $s_\emptyset$ is the initial state. Counter differences shows how HIMO prioritizes the occupation of states in time. We study these measures as well as their time derivatives in their original form as well as after max-normalizing per state in order to facilitate comparisons across states:

$$\widetilde{\mathrm{PD}}(s,t) := \frac{\mathrm{PD}(s,t)}{\max_t \mathrm{PD}(s,t)} \ , \ \widetilde{\mathrm{CD}}(s,t) := \frac{\mathrm{CD}(s,t)}{\max_t |\,\mathrm{CD}(s,t)|} \ . \tag{19}$$

The Tower of Hanoi example highlights the intuitive hierarchical qualities of the algorithm, and, in the grid worlds, the capacity of the algorithm to radically alter its dynamics in response to the relatively minor modifications of the state-space is observed. This demonstates the algorithm's sensitivity to all future paths without sampling.

In our Tower of Hanoi simulation (Fig. 1), the agent is endowed with the ability to remain at a state thus the optimal policy is to transit to state G and then choose to remain there (since it can then accumulate further reward on every time step). Of all actions in all states in the environment, HIMO policy improvements prioritize this goal-stay action. This can be observed in the relatively rapid policy divergence $\widetilde{\mathrm{PD}}$ at the goal state (Fig. 1B) and the fact that the policy divergence velocity peaks for the goal state before all others (Fig. 1D). The second highest priority is assigned to bottleneck states along the optimal path. The optimization of the local policy at the start state is deferred to last. Through the counter difference measure $\widetilde{\mathrm{CD}}$, we can observe how HIMO increases the occupation density of all states in the same cluster as the goal state (in blue) before subsequently reducing the occupation density of non-goal states in the goal cluster (Fig. 1E). These non-monotonic counter difference trajectories suggest that HIMO treats all blue states as a single unit initially before refining its planning strategy to distinguish individual states within the goal cluster. Increasing the resolution at which it distinguishes states over time as well as prioritizing local policy adaptations starting with the goal state through the bottleneck states and ending with the start state, suggests that HIMO is dynamically sensitive to the hierarchical structure of the state-space.

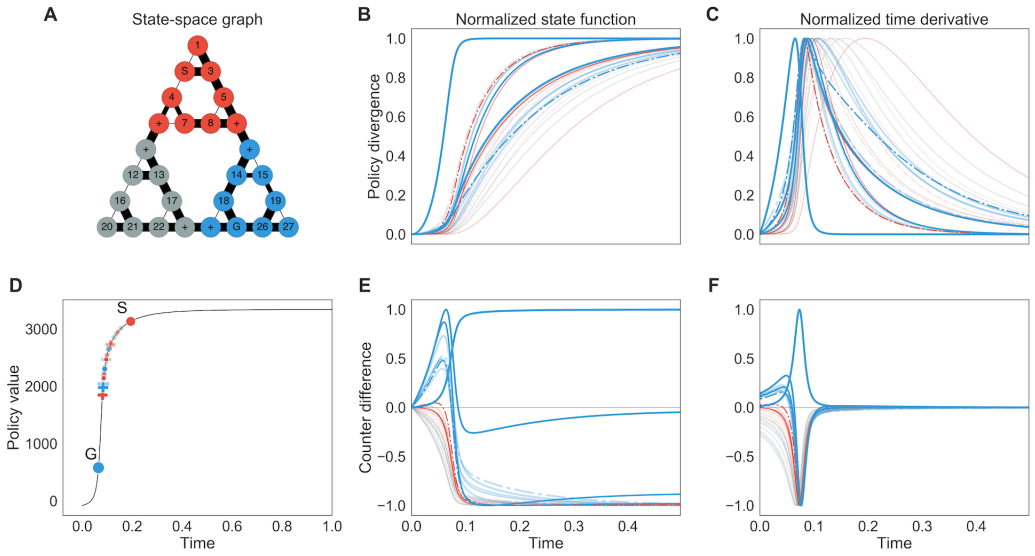

Figure 1: **Hierarchical policy optimization dynamics in the Tower of Hanoi game. A.** Tower of Hanoi state-space graph. **B-C.** Normalized policy divergence $\widetilde{\mathrm{PD}}$ and its time derivative for each state. The color of the curve indicates which state it corresponds to in panel A. Dotted lines correspond to bottleneck states marked + in panel A. Lines for states which are not along the optimal path are plotted transparently. **D.** Policy value as a function of planning time. Time-to-max policy divergence velocities (i.e. the peaks of the curves in panel C) are dotted along the policy value curve for states along the optimal path. **E-F.** Normalized counter difference $\widetilde{\mathrm{CD}}$ and its time derivative.

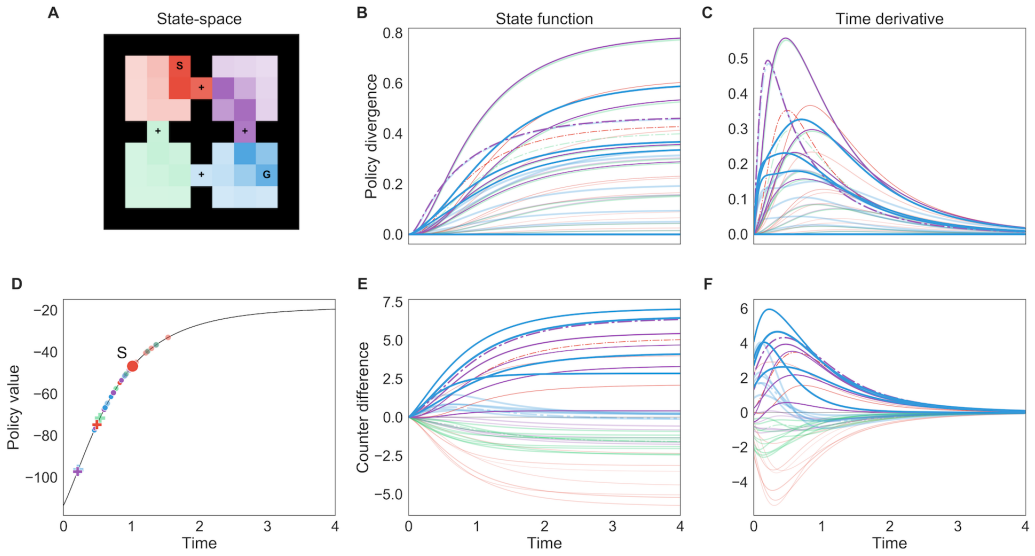

Figure 2: **Model-based policy optimization the optimal policy in a grid world.** Panels as in Fig. 1. Dotted lines with short dashes correspond to bottleneck states marked + in panel **A**. Darker state colors indicate higher densities of state occupation under the optimal policy.

In the room world simulation (Fig. 2), the agent must navigate from the start state S in the northwest room to the goal state G in the southeast room (panel A). HIMO prioritizes the adaptation of the local policies at the critical bottleneck states entering the goal room (Fig. 2D). In contrast, when a "wormhole" is available (Fig. S1, SM), the algorithm switches avoids the now-suboptimal route

through the doorway bottleneck and prioritizes the local adaptation of the policies at the entrance and exit of the wormhole.

5 Discussion

We introduced a novel model-based policy optimization procedure which we demonstrated to be sensitive to the hierarchical structure of the state-action space of the MDP. This feature is due to the use of second-order gradient information drawn from a sum-over-paths representation of the MDP objective. This causes the natural path gradient updates to take the steepest ascent in the space of policies defined over state-action paths through environment.

In previous work, natural policy gradient and actor-critic methods [15, 6, 16] have modified standard policy gradient steps using Fisher information matrices in order to perform parametrized policy updates in a manner that is sensitive to the KL-divergence between old and new local policies on average at each state. However, by averaging distinct natural gradients localized at each state based on the policy-induced stationary state distribution, these methods do not relate the components of the policy gradient across states or state-actions as in our method and thus is agnostic to the hierarchical structure of the state-space. As previously mentioned, the action preferences A employed in HIMO may be parametrized e.g. $A(\theta)$. Therefore, the natural gradient in terms of A with respect to path densities (sensitive to the global structure across states) established here may be combined with the natural gradient in terms of θ with respect to state-action probabilities (reflecting local structure at each state) through the reparametrization rule for Fisher informations. In this setup, we suggest that, as the foresight parameter $\lambda \to 1$ and the horizon becomes infinite, the natural path gradient converges on this previous definition of the natural policy gradient [6].

Hierarchical reinforcement learning [17, 18] refers to the acquisition and use of hierarchical representations in order to facilitate exploration, policy compression, and transfer [19]. Theoretic approaches typically make use of ad hoc objectives which do not pertain directly to the fundamental goal of policy improvement [20–22]. This is in contrast to the intuitive hierarchies emergent in the dynamics of HIMO. This motivates its use as a theoretic tool for analyzing the hierarchical structure of policy space since functional relationships between actions over all spatiotemporal scales are explicitly embedded within policy path gradients. In the classic hierarchical tasks simulated here, HIMO dynamically clusters then distinguishes state occupation densities (Fig. 1), implicitly prioritizes policy improvements at critical bottleneck states (Fig. 2), and restructures the policy flow in order to take advantage of shortcuts when available at the earliest stages of processing (Fig. S1). All such effects emerge from the single normative principle of performing policy gradient ascent in path space (Eqn. 13). Whereas these effects are manifest in the output of the algorithm, it will be informative to explore the internal dynamic of the algorithm by analyzing how the counter correlation functions and expected Hessians evolve over time.

References

[1] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[2] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[3] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *International Conference on Machine Learning*, pages 216 – 224, 1990.

[4] D Silver, A Huang, CJ Maddison, A Guez, L Sifre, G van den Driessche, J Schrittwieser, I Antonoglou, V Panneershelvam, M Lanctot, S Dieleman, D Grewe, J Nham, N Kalchbrenner, I Sutskever, T Lillicrap, M Leach, K Kavukcuoglu, T Graepel, and D Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[5] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. *International Conference on Machine Learning*, pages 465–472, 2011.

[6] J Andrew Bagnell and Jeff Schneider. Covariant policy search. *International Joint Conference on Artificial Intelligence*, pages 1019–1024, 2003.

[7] H J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005:21, 2005.

[8] Evangelos Theodorou, Krishnamurthy Dvijotham, and Emo Todorov. From information theoretic dualities to path integral and kullback leibler control: continuous and discrete time formulations. *Sixteenth Yale Workshop on Adaptive and Learning Systems*, 2013.

[9] Hiroshi Nagaoka. The exponential family of markov chains and its information geometry. *Proc. of the 28th Symposium on Information Theory and Its Applications, 2005 pp. 601-604*, 2005.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[12] James Martens. New insights and perspectives on the natural gradient method. *arXiv*, 2014.

[13] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 2006.

[14] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.

[15] Sham M Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems*, pages 1531–1538, 2001.

[16] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. *European Conference on Machine Learning*, pages 280–291, 2005.

[17] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181 – 211, 1999.

[18] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13:41–77, 2003.

[19] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning. *arXiv*, page 1909.10618v1, 2019.

[20] Sander Van Dijk and Daniel Polani. Grounding subgoals in information transitions. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 105–111, 2011.

[21] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G Barto, Yael Niv, and Matthew M Botvinick. Optimal behavioral hierarchy. *PLoS Computational Biology*, 10:e1003779, 2014.

[22] Daniel McNamee, Daniel Wolpert, and Máté Lengyel. Efficient state-space modularization for planning: theory, behavioral and neural signatures. *Advances in Neural Information Processing Systems 30*, 2016.

[23] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Springer-Verlag, 1983.

Hierarchical model-based policy optimization
Supplementary Material

Daniel McNamee
University College London
d.mcnamee@ucl.ac.uk

Contents

6 Extended simulations

6.1 Room world with wormhole

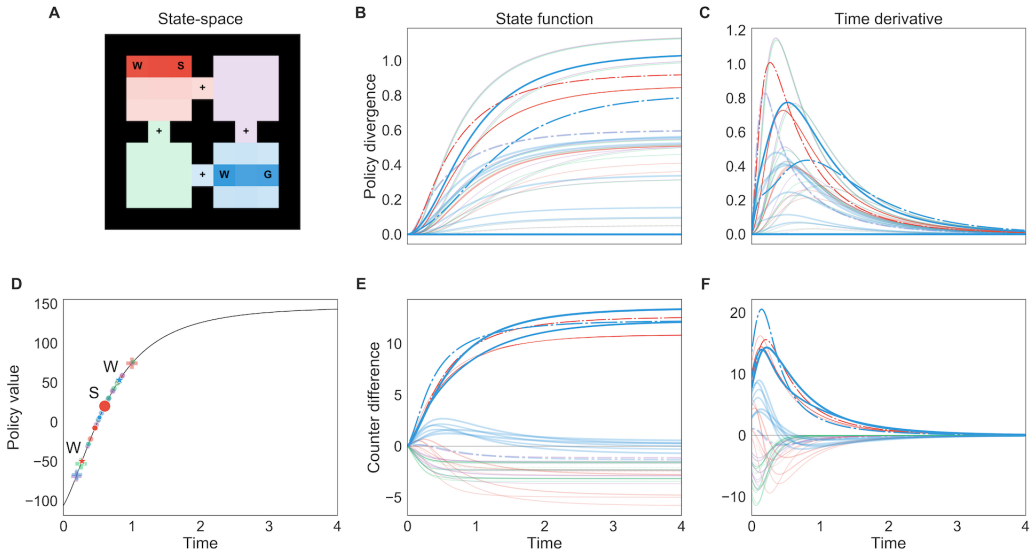

Figure S1: **Hierarchical model-based policy optimization in a grid world with a wormhole.** Panels as in Fig. 1. Dotted lines with short dashes correspond to bottleneck states marked + in panel **A**. Dotted lines with long dashes correspond to wormhole states marked W in panel **A**. Darker state colors indicate higher densities of state occupation under the optimal policy.

7 Technical derivations

7.1 Partial derivatives

In this section, we record some complementary calculations (Eqn. 20) and a proposition (7.1) which will be applied in Theorem 7.2.

The partial derivative $\partial_{A_{ij}} A_{kk_\omega}$ of a dependent action preference A_{kk_ω} with respect to an independent action preference A_{ij} is

$$\begin{aligned}
\partial_{A_{ij}} A_{kk_\omega} &= \partial_{A_{ij}} \left[\log \left(1 - \sum_{a_{k_\omega} \neq a_l \in \mathcal{A}_k} e^{A_{kl}} \right) \right] \\
&= \left(1 - \sum_{a_{k_\omega} \neq a_l \in \mathcal{A}_k} e^{A_{kl}} \right)^{-1} \delta_{ik} \left[-e^{A_{kj}} \right] \\
&= -\delta_{ik} e^{A_{kj} - A_{kk_\omega}} \quad . \qquad (20)
\end{aligned}$$

Proposition 7.1. The partial derivative of the path density $\mathbf{p}(\boldsymbol{\tau})$ and the path policy $\boldsymbol{\pi}(\boldsymbol{\tau})$ with respect to A_{ij} is

$$\begin{aligned}
\partial_{A_{ij}} \mathbf{p}(\boldsymbol{\tau}) &= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \log \mathbf{p}(\boldsymbol{\tau}) \qquad (21) \\
&= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \log \boldsymbol{\pi}(\boldsymbol{\tau}) \qquad (22) \\
&= \mathbf{p}(\boldsymbol{\tau}) \left[n_{ij}(\boldsymbol{\tau}) - e^{A_{ij} - A_{ii_\omega}} n_{ii_\omega}(\boldsymbol{\tau}) \right] \quad . \qquad (23)
\end{aligned}$$

Proof. Using the log-derivative trick

$$\begin{aligned}
\partial_{A_{ij}} \mathbf{p}(\boldsymbol{\tau}) &= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \left[\log \mathbf{p}(\boldsymbol{\tau}) \right] \\
&= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \left[\log \boldsymbol{\pi}(\boldsymbol{\tau}) + \log \mathbf{p}(\mathbf{s}_{+1} | \mathbf{s}, \mathbf{a}) \right] \\
&= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \left[\mathbf{A} \cdot \mathbf{n}(\boldsymbol{\tau}) \right] \\
&= \mathbf{p}(\boldsymbol{\tau}) \partial_{A_{ij}} \left\{ \sum_{k \in \mathcal{X}} \left[\sum_{k_\omega \neq l \in \mathcal{X}_k} A_{kl} n_{kl}(\boldsymbol{\tau}) + A_{kk_\omega} n_{kk_\omega}(\boldsymbol{\tau}) \right] \right\} \\
&= \mathbf{p}(\boldsymbol{\tau}) \left[n_{ij}(\boldsymbol{\tau}) + \left(\partial_{A_{ij}} A_{ii_\omega} \right) n_{ii_\omega}(\boldsymbol{\tau}) \right] \\
&= \mathbf{p}(\boldsymbol{\tau}) \left[n_{ij}(\boldsymbol{\tau}) - e^{A_{ij} - A_{ii_\omega}} n_{ii_\omega}(\boldsymbol{\tau}) \right] \qquad (24)
\end{aligned}$$

re-using Eqn. 20. □

7.2 Fisher information

State transition occupations are not independent. Modifying one transition occupation probability under the policy π may change the occupation probability of another transition. This is in contrast to the expected reward objective in path space where policy modifications are independent along each path dimension (apart from an overall normalization factor). In order to identify a policy gradient in transition space with independent gradient components, we will transform the gradient derived in Section 7.3 into the natural path gradient pulled back to transition space. In order to make this gradient ascent natural in the space of transitions, we pre-multiply the gradient by the inverse Fisher information $\mathcal{I}^{-1}$ [15] which relates the policy densities in path space $\boldsymbol{\pi}$ and transition space π. The Fisher information matrix $\mathcal{I}$ has components

$$\begin{aligned}
\mathcal{I}_{ij,kl} &:= \left\langle \left[\partial_{A_{ij}} \log \boldsymbol{\pi}(\boldsymbol{\tau}) \right] \left[\partial_{A_{kl}} \log \boldsymbol{\pi}(\boldsymbol{\tau}) \right] \right\rangle_{\mathbf{p}} \\
&= \left\langle \left[n_{ij}(\boldsymbol{\tau}) - e^{A_{ij} - A_{ii_\omega}} n_{ii_\omega}(\boldsymbol{\tau}) \right] \left[n_{kl}(\boldsymbol{\tau}) - e^{A_{kl} - A_{kk_\omega}} n_{kk_\omega}(\boldsymbol{\tau}) \right] \right\rangle_{\mathbf{p}} \\
&= \left\langle n_{ij}(\boldsymbol{\tau}) n_{kl}(\boldsymbol{\tau}) \right\rangle_{\boldsymbol{\pi}} - e^{A_{kl} - A_{kk_\omega}} \left\langle n_{ij}(\boldsymbol{\tau}) n_{kk_\omega}(\boldsymbol{\tau}) \right\rangle_{\mathbf{p}} + \\
&\quad - e^{A_{ij} - A_{ii_\omega}} \left\langle n_{kl}(\boldsymbol{\tau}) n_{ii_\omega}(\boldsymbol{\tau}) \right\rangle_{\boldsymbol{\pi}} + e^{A_{ij} - A_{ii_\omega}} e^{A_{kl} - A_{kk_\omega}} \left\langle n_{ii_\omega}(\boldsymbol{\tau}) n_{kk_\omega}(\boldsymbol{\tau}) \right\rangle_{\mathbf{p}} \\
&= \mathcal{C}_{ij,kl} - e^{A_{kl} - A_{kk_\omega}} \mathcal{C}_{ij,kk_\omega} - e^{A_{ij} - A_{ii_\omega}} \mathcal{C}_{kl,ii_\omega} + e^{A_{ij} + A_{kl} - A_{ii_\omega} - A_{kk_\omega}} \mathcal{C}_{ii_\omega,kk_\omega} \quad . \qquad (25)
\end{aligned}$$

The Fisher information $\mathcal{I}$ depends on the counter correlation functions $\mathcal{C}_{ij,kl} := \langle n_{ij}(\boldsymbol{\tau}) n_{kl}(\boldsymbol{\tau}) \rangle_{\mathbf{p}}$ which can be analytically derived based on Markov chain theory [23].

7.3 Model-based policy path gradient

Theorem 7.2. The policy path gradient in the exponential parametrization is defined by the partial derivatives

$$\partial_{A_{ij}} \mathcal{R}(\mathbf{A}) \quad = \quad \sum_{\substack{s_k \in \mathcal{S} \\ a_l \in \mathcal{A}_k}} \left[\mathcal{C}_{ij,kl} - e^{A_{ij} - A_{ii_\omega}} \mathcal{C}_{ii_\omega,kl} \right] \mathrm{R}_{kl} \tag{26}$$

where $\mathcal{C}_{ij,kl} := \langle n_{ij}(\boldsymbol{\tau}) n_{kl}(\boldsymbol{\tau}) \rangle_{\mathbf{p}}$ are state-action correlation functions and $\mathrm{R}_{kl} := \langle \mathrm{R}(s_i, a_j, s_k) \rangle_{\mathrm{p}(s_k | s_i, a_j)}$.

Proof.

$$\begin{aligned}
\partial_{A_{ij}} \mathcal{R}(\mathbf{A}) &= \partial_{A_{ij}} \sum_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}} \mathbf{p}(\boldsymbol{\tau}) \left[\nabla_{\mathbf{A}} \log \mathbf{p}(\boldsymbol{\tau}) \right] \mathbf{R}(\boldsymbol{\tau}) \\
&= \sum_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}} \mathbf{p}(\boldsymbol{\tau}) \left[n_{ij}(\boldsymbol{\tau}) - e^{A_{ij} - A_{ii_\omega}} n_{ii_\omega}(\boldsymbol{\tau}) \right] \mathbf{R}(\boldsymbol{\tau}) \qquad (\Leftarrow \text{Proposition 7.1}) \\
&= \sum_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}} \left\{ \mathbf{p}(\boldsymbol{\tau}) \left[n_{ij}(\boldsymbol{\tau}) - e^{A_{ij} - A_{ii_\omega}} n_{ii_\omega}(\boldsymbol{\tau}) \right] \right\} \left[\sum_{\substack{s_k, s_m \in \mathcal{S} \\ a_l \in \mathcal{A}_k}} n_{klm}(\boldsymbol{\tau}) R_{klm} \right] \\
&= \sum_{\substack{s_k, s_m \in \mathcal{S} \\ a_l \in \mathcal{A}_k}} \left[\langle n_{ij}(\boldsymbol{\tau}) n_{klm}(\boldsymbol{\tau}) \rangle_{\mathbf{p}} - e^{A_{ij} - A_{ii_\omega}} \langle n_{ii_\omega}(\boldsymbol{\tau}) n_{klm}(\boldsymbol{\tau}) \rangle_{\mathbf{p}} \right] R_{klm} \\
&= \sum_{\substack{s_k, s_m \in \mathcal{S} \\ a_l \in \mathcal{A}_k}} \left[\mathcal{C}_{ij,kl} - e^{A_{ij} - A_{ii_\omega}} \mathcal{C}_{ii_\omega,kl} \right] p_{klm} R_{klm} \\
&= \sum_{\substack{s_k \in \mathcal{S} \\ a_l \in \mathcal{A}_k}} \left[\mathcal{C}_{ij,kl} - e^{A_{ij} - A_{ii_\omega}} \mathcal{C}_{ii_\omega,kl} \right] R_{kl} \ .
\end{aligned}$$

□

Given a policy π and an environment dynamics model, the state-action correlation functions $\mathcal{C}$ can be derived using Markov chain theory [23] and therefore the policy path gradient can be expressed analytically. In order to ensure convergence of these quantities, we assume a discounted horizon with parameter λ which is upper bounded by the discount parameter $\lambda < \gamma$. That is, from the perspective of the agent, an episodic may end on every timestep with probability $1 - \lambda$. This parameter may reflect a reward discount or a limitation on how far the agent can "see" into the future.

7.4 Algorithm

input: initialized policy π with corresponding action preferences $A_{ij} := \log \pi_{ij}$
while π *not converged* **do**
  compute state-state (D) and (state-action)-state (E) counter correlations
  foreach $(s_i, s_k) \in \mathcal{S} \times \mathcal{S}$ **do**

$$\begin{aligned} T_{ik} &= \sum_{a_j \in \mathcal{A}_i} \pi_{ij} p_{ijk} \\ D_{ik} &= \left[(I - \lambda T)^{-1} \right]_{ik} \end{aligned}$$

    foreach $(a_j) \in \mathcal{A}_i$ **do**

$$E_{(ij)k} = \sum_{s_{k'} \in \mathcal{S}} p_{ijk'} D_{k'k}$$

    end
  end
  compute (state-action)-(state-action) counter correlations $\mathcal{C}$ and Fisher information $\mathcal{I}$
  foreach $(s_i, a_j, s_k, a_l) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A}$ **do**

$$\begin{aligned} \mathcal{C}_{ij,kl} &= D_{\emptyset i} \pi_{ij} \delta_{ik} \delta_{jl} + \left[D_{\emptyset i} E_{(ij)k} + D_{\emptyset k} E_{(kl)i} \right] \pi_{ij} \pi_{kl} \\ \mathcal{I}_{ij,kl} &= \mathcal{C}_{ij,kl} - \pi_{kl} \pi^{-1}_{kk_\omega} \mathcal{C}_{ij,kk_\omega} - \pi_{ij} \pi^{-1}_{ii_\omega} \mathcal{C}_{kl,ii_\omega} + \pi_{ij} \pi^{-1}_{ii_\omega} \pi_{kl} \pi^{-1}_{kk_\omega} \mathcal{C}_{ii_\omega,kk_\omega} \end{aligned}$$

  end
  update action preferences
  foreach $(s_i, a_j) \in \mathcal{S} \times \mathcal{A}$ **do**

$$A_{ij} \leftarrow A_{ij} + \alpha \sum_{s_m \in \mathcal{S}, a_n \in \mathcal{A}_m} \left[\mathcal{I}^{-1} \right]_{ij,mn} \left\{ \sum_{s_k \in \mathcal{S}, a_l \in \mathcal{A}_k} \left[\mathcal{C}_{mn,kl} - \pi_{mn} \pi^{-1}_{mm_\omega} \mathcal{C}_{mm_\omega,kl} \right] R_{kl} \right\} \tag{27}$$

  end
  normalize action preferences
  foreach $s_i \in \mathcal{S}$ **do**

$$A_{ii_\omega} = \log \left(1 - \sum_{a_{i_\omega} \neq a_j \in \mathcal{A}_i} e^{A_{ij}} \right)$$

  end
  compute policy
  foreach $(s_i, a_j) \in \mathcal{S} \times \mathcal{A}$ **do**

$$\pi_{ij} = e^{A_{ij}}$$

  end
end
output : optimal policy π^*
Algorithm 1: Hierarchical model-based policy optimization in the exponential parametrization.

where (ij) is a univariate index of state-action (s_i, a_j) combinations. The parameter λ is a free parameter $0 < \lambda < 1$ controlling the agent's "foresight". The parameter $\alpha > 0$ has been introduced to modulate the stepsize. Note that the transition reward R, transient transition matrix T, successor representation D, Fisher information $\mathcal{I}$, and counter correlations $\mathcal{C}$, all depend on the current policy estimate π^t.

The path gradient (Eqn. 27) has several intuitive properties. The matrix D is the successor representation [14]. An entry D_{ij} counts the expected number of times that state s_j will be occupied

after starting from state s_i. Therefore the counter correlations $\mathcal{C}$, which is quadratic in successor representations, reflect the rate of co-occurrence of pairs of state-actions on average under the policy-generated path distribution. This enables the algorithm to understand the correlative structure of state occupations under the current policy as proposed. For example, if a temporally remote state-action (s_k, a_l) has high reward R_{kl} and if there is a high counter correlation $\mathcal{C}_{ij,kl}$ between a local state-action (s_i, a_j) and the remote action (over all horizons), then the reward R_{kl} associated with the remote action will be weighted heavily in the path gradient update and added to the local action preference A_{ij}. The magnitude of this backup is explicitly normalized with respect to a baseline counter correlation $\mathcal{C}_{ii_\omega,kl}$ associated with the dependent action preference. That is, if the action (s_i, a_{i_ω}) is also strongly correlated with (s_k, a_l) then the backup to A_{ij} is attenuated since the unique contribution of (s_i, a_j) in generating (s_k, a_l) is diminished. Using such attributional logic, optimal model-based policy optimization updates action preferences based on the degree to which a state-action independently generate to rewarding state-space paths over all depths.

7.5 Initialization

The prior policy π^0 can be set to any policy with corresponding initial action preferences

$$A_{ij}^0 \quad = \quad \log \pi_{ij}^0 \ . \tag{28}$$

Assuming that π^0 is initialized at the random policy, we have

$$\pi_{ij}^0 \quad = \quad \frac{1}{|\mathcal{A}_i|} \tag{29}$$

$$A_{ij}^0 \quad = \quad -\log |\mathcal{A}_i| \tag{30}$$

for all $a_j \in \mathcal{A}$.