Supplementary Material for Explorative Imitation Learning: A Path Signature Approach for Continuous Environments

Nathan Gavenski [a,*], **Juarez Monteiro** [a], **Felipe Meneguzzi** [b], **Michael Luck** [c] **and Odinaldo Rodrigues** [a]

[a]King's College London, London, United Kingdom
[b]University of Aberdeen, Aberdeen, United Kingdom
[c]University of Sussex, Sussex, United Kingdom

Abstract. This Supplementary Material is part of the main submission Explorative Imitation Learning: A Path Signature Approach for Continuous Environments.

1 Environments and samples

In this work, we experiment with five different environments. We now briefly describe each environment and how the expert samples were gathered. We used Stable Baselines 3 [9] coupled with RL Zoo3 [8] to gather expert samples and its weights loaded from HuggingFaces [1]. We believe this will facilitate reproducibility by allowing future work to use the exact same experts. All expert results are displayed in Table 1 in Section 4.2 of the paper. We used a random sample pool of $50,000$ states for all environments (partitioned into $35,000$ states for training and the remaining $15,000$ for validation). It is important to note, that in each environment, a dimension d of these state vectors $\vec{v}$ represents an internal attribute of the robot. Therefore, although they might share a similar number of dimensions, they may carry different meanings. Since I^s grows in size in each iteration, unlike Torabi et al.'s work [11], CILO does not rely on higher sample pools. Figure 1 shows a frame for each environment.

1.1 A note about the expert samples

During our experiments, we observed that not all experts are created equally. Although most experts trained or loaded from HuggingFace share similar results, the behaviour of each expert varies drastically. One could argue that humans also deviate for each trajectory, but using episodes with a more human-like trajectory (less hectic) yielded better results for all IL approaches. By presenting less hectic and more constant movements, we think each policy receives trajectories that vary more and generalise better. All samples used in this work are available in https://github.com/NathanGavenski/CILO.

1.2 Ant-v2

Ant-v2 consists of a robot ant made out of a torso with four legs attached to it, with each leg having two joints [10]. The goal of this environment is to coordinate the four legs to move the ant to the right of the screen by applying force on the eight joints. Ant-v2 requires eight actions per step, each limited to continuous values between -1 and 1. Its observation space consists of 27 attributes for the x, y and z axis of the 3D robot. We use Stable Baselines 3's TD3 weights. The expert sample contains 10 trajectories, each with $1,000$ states consisting of 111 attributes.[2] Ant-v2 shares distribution behaviour with InvertedPendulum-v2, and Hopper-v2, having action spaced in a bell-curve.

1.3 InvertedPendulum-v2

This environment is based on the CartPole environment from Barto et al. [1]. It involves a cart that can move linearly, with a pole attached to it. The agent can push the cart left or right to balance the pole by applying forces on the cart. The goal of the environment is to prevent the pole from reaching a particular angle on either side. The continuous action space varies between -3 and 3, the only one within the five environments outside of the -1 to 1 limit. Its observation space consists of 4 different attributes. We use Stable Baselines 3's PPO weights. The expert sample size is 10 trajectories, which consist of $10,000$ states (with their 4 attributes) and actions (with a single action value per step). The invertedPendulum-v2 environment is the only one that has an expert with the environment's maximum reward. Therefore achieving $\mathcal{P}$ higher than 1 is impossible.

1.4 Swimmer-v2

This environment was proposed by [3]. It consists of a robot with s segments ($s \geqslant 3$) and $j = s - 1$ joints. Following [14], in our experiments we use the default setting $s = 3$ and $j = 2$. The agent applies force to the robot's joints, and each action can range from $[-1, 1] \in \mathbb{R}$. A state is encoded by an 8-dimensional vector representing the angle, velocity and angular velocity of all segments. Swimmer distributions present the same distribution of HalfCheetah-v2 (centred around the lower and upper limits). We used Stable Baselines 3's TD3 weights. The expert sample contains 4 trajectories, with $1,000$ states each plus actions for the $j = 2$ joints. The goal of the agent in this environment is to move as fast as possible towards the right by applying torque on the joints and using the fluid's friction.

* Corresponding Author. Email: nathan.schneider_gavenski@kcl.ac.uk
[1] https://huggingface.co/

[2] With their 111 different attributes - MuJoCo implementation has 27 positions with values and the rest with 0).

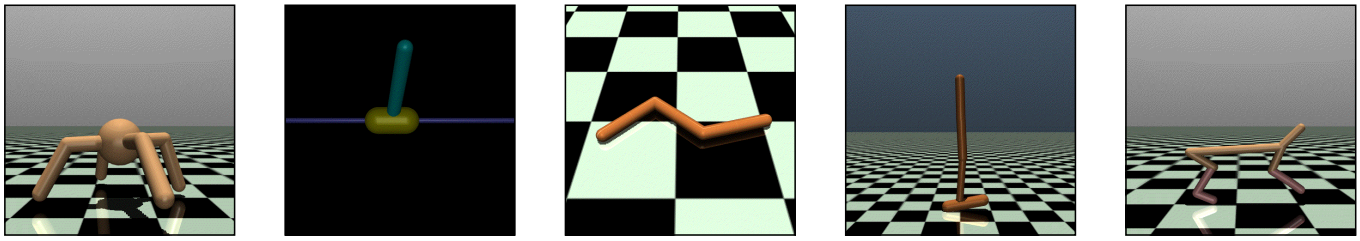


Figure 1. A single frame for each environment used in this work.

Table 1. Layers for each neural network used in this work, where d is the number of dimensions for each state, $\mid a \mid$ is the number of actions, and $\mid \beta \mid$ is given by Eq. 6.

$\mathcal{M}$		π_θ		$\mathcal{D}$	
Layer Name	Input × Output	Layer Name	Input × Output	Layer Name	Input × Output
Input	$2d \times 512$	Input	$d \times 512$	Input	$\mid \beta \mid \times 512$
Activation (Tanh)	-	Activation (Tanh)	-	Activation (Tanh)	-
Fully Connected 1	512×512	Fully Connected 1	512×512	Fully Connected 1	512×512
Activation (Tanh)	-	Activation (Tanh)	-	Activation (Tanh)	-
Self-Attention 1	512×512	Self-Attention 1	512×512	Dropout	0.5%
Fully Connected 2	512×512	Fully Connected 2	512×512	Fully Connected 2	512×512
Activation (Tanh)	-	Activation (Tanh)	-	Activation (Tanh)	-
Self-Attention 2	512×512	Self-Attention 2	512×512	Dropout	0.5%
Fully Connected 3	512×512	Fully Connected 3	512×512	Output	512×2
Activation (Tanh)	-	Activation (Tanh)	-		
Fully Connected 4	512×512	Fully Connected 4	512×512		
Output	$512 \times \mid a \mid$	Output	$512 \times \mid a \mid$		

1.5 Hopper-v2

Hopper-v2 is based on the work done by [4]. Its robot is a one-legged two-dimensional body with four main parts connected by three joints: a torso at the top, a thigh in the middle, a leg at the bottom, and a single foot facing the right. The environment's goal is to make the robot hop and move forward (continuing on the right trajectory). A state consists of 11 attributes representing the z-position, angle, velocity and angular velocity of the robot's three joints. We used Stable Baselines 3's TD3 weights and 10 expert episodes, each with $1,000$ states and actions for the three joints. Each action is limited between $[-1, 1] \in \mathbb{R}$.

1.6 HalfCheetah-v2

HalfCheetah-v2's environment was proposed in [12]. It has a 2-dimensional cheetah-like robot with two "paws". The robot contains 9 segments and 8 joints. Its actions are a vector of 6 dimensions, consisting of the torque applied to the joints to make the cheetah run forward ("thigh", "shin", and "paw" for the front and back parts of the body). All states consist of the robot's position and angles, velocities and angular velocities for its joints and segments. HalfCheetah-v2's goal is to run forward (i.e., to the right of the screen) as fast as possible. A positive reward is allocated based on the distance traversed, and a negative reward is awarded when moving to the left of the screen. We used Stable Baselines 3's TD3 weights. The expert sample size is 10 trajectories, each consisting of $1,000$ states and actions. Each action is limited between the interval of $[-1, 1] \in \mathbb{R}$.

2 Network Topology

We followed the same network topologies employed in the original works. Each model ($\mathcal{M}$ and π_θ) are MLP with 4 fully connected layers, each with 512 neurons, with the exception of the last layer whose size is the same as the number of environment actions, Table 1 displays the topologies alongside the input and output sizes of each layer. Following the implementation in [5], we used a self-attention module after the first and second layers. We experimented with normalisation layers during development, which did not increase the agents' results but helped with weight updates. Although we understand that having more complex architectures could increase our method's performance, for consistency we used the same original architecture to show that CILO achieves expert results and does not rely on the architecture. The implementation of our method can be found within https://github.com/NathanGavenski/CILO.

3 Training and Learning Rate

For training, we used a Nvidia A100 40GB GPU and PyTorch. Although we used this GPU, such hardware is not strictly required since CILO uses $\approx$ 2GB to train with a 1024 mini-batch size. The learning rates for $\mathcal{M}$ and π_θ are shown in Table 2. We note that CILO is robust to different learning rates for π_θ. However, $\mathcal{M}$ is more sensitive since I^s changes at almost every iteration, assuming there is at least one agent's trajectory that $\mathcal{D}$ classifies as expert. Having a high learning rate can make $\mathcal{M}$'s weights update too harshly and result in CILO never learning how to label the $\mathcal{T}^{\pi_\psi}$ properly.

Table 2. Different learning rates for $\mathcal{M}$ and π_θ for all environments.

Environment	$\mathcal{M}$	π_θ	Signature k
Ant	1×10^3	1×10^3	2
InvertedPendulum	1×10^3	1×10^3	4
Swimmer	3×10^3	7×10^4	4
Hopper	5×10^3	1×10^3	4
HalfCheetah	1×10^3	7×10^4	4

4 Path Signatures

In this work we rely on several path signature definitions to discriminate over agent and expert trajectories. In Section 3.2 of our paper, we briefly defined a trajectory τ, in which each state is a vector $\vec{v}$ in $\mathbb{R}^d$, and how to compute the path signature $\beta(\tau)^{i_1,\cdots,i_k}_{1,n}$, where n is the length of the trajectory, and $i_1, \cdots, i_k \in \{1, \cdots, d\}$ ($k > 1$) are indices to elements in $\vec{v}$. Here, we provide some additional information on the process of computing a path signature and the intuition behind it.

4.1 Computing the Path Signature

Given a trajectory τ and a function f that interpolates τ into a continuous map $f : \mathbb{R} \to \mathbb{R}$, the integral of the the trajectory against f can be defined as:

$$\int_1^n f(\tau_t) d\tau_t = \int_1^n f(\tau_t)\dot{\tau}_t dt, \tag{1}$$

where $\dot{\tau}_t = \frac{d\tau_t}{dt}$ for any time $t \in [1, n]$. Note that $f(\tau_t)$ is a real-valued path defined on $[1, n]$, which can be considered the integral of a trajectory τ. Moreover, if we consider that $f(\tau_t) = 1$ for all $t \in [1, n]$, then the path integral of f against any trajectory $\tau : [1, n] \to \mathbb{R}$ is simply the increment of τ:

$$\int_1^n d\tau_t = \int_1^n \dot{\tau} dt = \tau_n - \tau_1. \tag{2}$$

Therefore, by assuming that β is a function of real-valued paths, we can define the signature for any single index $i_k \in \{1, \cdots, d\}$ as:

$$\beta(\tau)^{i_k}_{1,n} = \int_{1<s\leqslant n} d\tau_s^{i_k} = \tau_n^{i_k} - \tau_1^{i_k}, \tag{3}$$

which is the increment of the i_k-th dimension of the path. Now, if we move to any pair of indexes $i_k, j_k \in \{1, \cdots, d\}$, we have to consider the double-iterated integral:

$$\beta(\tau)^{i_k,j_k}_{1,n} = \int_{1<s\leqslant n} \beta(\tau)^{i_k}_{1,s} d\tau_s^{j_k} = \int_{1<r\leqslant s\leqslant n} d\tau_t^{i_k} d\tau_s^{j_k}, \tag{4}$$

where $\beta(\tau)^{i_k}_{1,s}$ is given by Eq. 3. Considering that $\beta(\tau)^{i_k,j_k}_{1,n}$ continues to be a real-values path, then we can define recursively the signature function for any number of indexes $k \geqslant 1$ in the collection of indexes $i_1, \cdots, i_k \in \{1, \cdots, d\}$ as:

$$\beta(\tau)^{i_1,\cdots,i_k}_{1,n} = \int_{1<s\leqslant n} \beta(\tau)^{i_1,\cdots,i_{k-1}}_{1,s} d\tau_s^{i_k}, \tag{5}$$

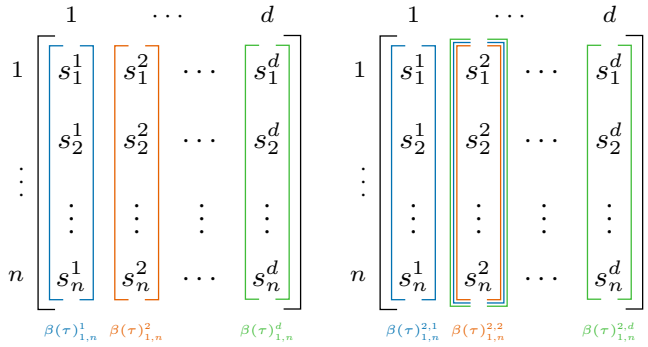
which in our paper is Eq. 6. It is important to note that k is the depth up to which the signature is generated (not its length). At each level $i \leqslant k$, "words" of length i are generated from the alphabet D according to Eq. 6 (main work) to produce the terms of the signature.

Figure 2c shows all possible terms for a trajectory $\tau : [1, n] \to \mathbb{R}^d$ for different depths. For example, a signature with depth 2 will have all the terms in the levels $i = 0, 1, 2$. In an alphabet with d letters, we can construct one word of length 0, d words of length 1, and d^2 words of length 2, giving $1 + d + d^2$ words in total (i.e., the number of terms in the signature). In general, the length of a signature with alphabet size d and depth k is:

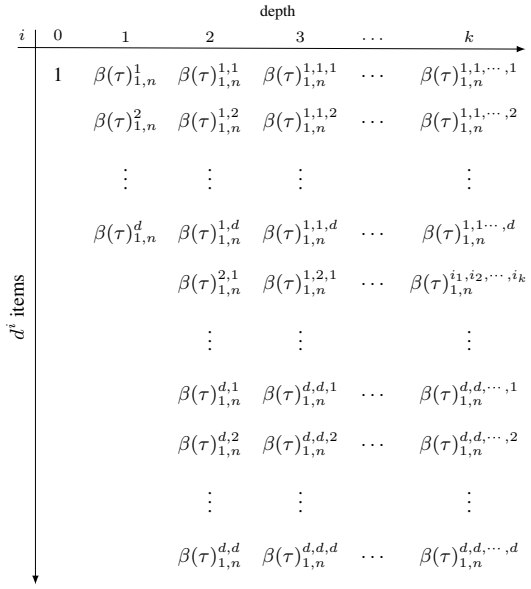
$$\sum_{i=0}^{k} d^i = \frac{d^{k+1} - 1}{d - 1}. \tag{6}$$

We observe that signatures can be computed for any depth k, and are not restricted to $k \leqslant d$.

We give two examples to illustrate how the terms in a signature are computed (we omit the level 0 whose single value 1 is fixed). Figure 2a shows how to generate a signature of depth 1 (with the terms in the first and second columns of Figure 2c). Considering that the length of a signature grows exponentially with the depth k desired ($\frac{d^{k+1}-1}{d-1}$), Figure 4 only shows how to calculate the terms of a signature of depth 2 for a 2-dimensional dictionary, with the first index fixed in 2.



(a) $\beta(\tau)^i_{1,n} \mid \forall_i \in \{1, \cdots, d\}$. (b) $\beta(\tau)^{2,j}_{1,n} \mid \forall_j \in \{1, \cdots, d\}$.



(c) Collection of signatures for τ, where $k \in [0, \infty)$.

Figure 2. Illustration of path signature.

4.2 A Numerical Example

Let us consider a trajectory τ with two two-dimensional states $\{\tau_t^1, \tau_t^2\}$, and the set of multi-indexes $W = \{\ (i_1, \cdots, i_k)\ \mid\ k \geqslant 1, i_1, \cdots, i_k \in \{1, 2\}\ \}$, which is the set of all finite sequences of 1's and 2's. Given the trajectory $\tau : [1, 10] \to \mathbb{R}^2$ illustrated in Figure 3, where the path function for τ is computed according to the function:

$$\tau_t = \{\tau_t^1, \tau_t^2\} = \{5 + t, (5 + t)^2 \mid t \in [1, 10]\} \tag{7}$$

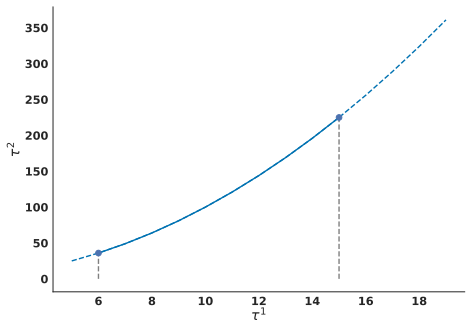


Figure 3. Trajectory $\tau : [1, 10] \to \mathbb{R}^2$.

For the depth k desired, the computation of the signature would be computed as shown in Figure 4. For example, given that states in τ are two-dimensional ($d = 2$), the path signature for τ with depth $k = 2$ will have the $\frac{d^{k+1}-1}{d-1} = \frac{2^3-1}{1} = 7$ terms in the vector $\beta(\tau)_{1,10} = [\ 1, 9, 189, 40.5, 970.5, 730.5, 17860.5\ ]$.

$k = 0$

$$\beta(\tau)_{1,n} = 1$$

$k = 1$ – Eq. 3

$$\beta(\tau)^1_{1,10} = \int_{1<t\leqslant 10} dt = \tau^1_{10} - \tau^1_1 = 9$$

$$\beta(\tau)^1_{1,10} = \int_{1<t\leqslant 10} dt = \tau^2_{10} - \tau^2_1 = 189$$

$k = 2$ – Eq. 4

$$\beta(\tau)^{1,1}_{1,10} = \iint_{1<t_1\leqslant t_2\leqslant 10} d\tau^1_{t_1}, d\tau^1_{t_2} = \int_1^{10}\left[\int_1^{t_2} dt_1\right] dt_2 = 40.5$$

$$\beta(\tau)^{1,2}_{1,10} = \iint_{1<t_1\leqslant t_2\leqslant 10} d\tau^1_{t_1}, d\tau^2_{t_2} = \int_1^{10}\left[\int_1^{t_2} dt_1\right] 2(5 + t_2)dt_2 = 970.5$$

$$\beta(\tau)^{2,1}_{1,10} = \iint_{1<t_1\leqslant t_2\leqslant 10} d\tau^2_{t_1}, d\tau^1_{t_2} = \int_1^{10}\left[\int_1^{t_2} 2(5 + t)dt_1\right] dt_2 = 730.5$$

$$\beta(\tau)^{2,2}_{1,10} = \iint_{1<t_1\leqslant t_2\leqslant 10} d\tau^2_{t_1}, d\tau^2_{t_2} = \int_1^{10}\left[\int_1^{t_2} 2(5 + t)dt_1\right] 2(5 + t)dt_1 = 17,860.5$$

$k \geqslant 3$ – Eq. 5

$$\beta(\tau)^{1,1,1}_{1,10} = \iiint_{1<t_1\leqslant t_2\leqslant t_3\leqslant 10} d\tau^1_{t_1} d\tau^1_{t_2} d\tau^1_{t_3} = \int_1^{t_1}\left[\int_1^{t_2}\left[\int_1^{t_3} dt_1\right] dt_2\right] dt_3 = 121.5$$

$$\vdots$$

Figure 4. Step-by-step computation of a path signature

4.3 Signature Properties

We now describe properties of path signatures that are most relevant to our work. The description is not comprehensive. We recommend the work from Yang et al. [13] and Chevyrev and Kormilitzin [2] for a more in-depth approach to path signatures.

Uniqueness: This property relates to the fact that no two trajectories τ and τ' of bounded variation have the same signature unless the trajectories are tree-equivalent. In light of the invariance under reparametrisations [6], we note that path signatures have no tree-like sections to monotone dimensions, such as acceleration.

Generic nonlinearity of the signature: The second property refers to the product of two terms $\beta(\tau)^{i_1,\cdots,i_k}$ and $\beta(\tau)^{j_1,\cdots,j_k}$, which can also be expressed as:

$$\begin{aligned}\beta(\tau)^1_{1,n} \cdot \beta(\tau)^2_{1,n} &= \beta(\tau)^{1,2}_{1,n} + \beta(\tau)^{2,1}_{1,n} \text{ or,}\\ \beta(\tau)^{1,2}_{1,n} \cdot \beta(\tau)^1_{1,n} &= \beta(\tau)^{1,1,2}_{1,n} + \beta(\tau)^{1,2,1}_{1,n}.\end{aligned} \tag{8}$$

Thus, the nonlinearity of the signature in terms of low-level terms can be expressed by the linear combination of higher-level terms, which adds more nonlinear previous knowledge to the feature vector. This behaviour is better exemplified in the second level of signatures where for any $\beta(\tau)^{i_k,i_k}_{1,n}$, the result will be $\left(\tau^{i_k}_n - \tau^{i_k}_1\right)^2/2$.

Fixed dimension under length variations: The last property refers to the path signature's length invariance under different trajectory lengths. In Section 4.1, we showed that the signature length is a function of the signature depth (k) and the number of dimensions in a state (d). Therefore, path signatures become practical feature vectors for trajectories in machine learning tasks, requiring different inputs to share the same size without recurrent neural networks.

4.4 Motivation for Signatures

Given the nature of deep learning methods operating on vectorial data, which requires the input data to be of a predetermined fixed length, many techniques, such as word embeddings (where a word is represented by a vector), are used to circumvent this length requirement. Moreover, imitation learning tasks, by definition, have to effectively represent expert demonstrations to capture relevant information for learning a desired behaviour. Path signatures provide a solution to represent sequential or trajectory-based expert demonstration in a principled and efficient manner.

In imitation learning, expert demonstration often takes the form of trajectories or sequences of states over time. Path signatures offer a way to encode these trajectories into high-dimensional feature representations that capture the expert behaviour in a geometric and analytic way. Furthermore, the uniqueness property ensures that essential information about the expert demonstrations is preserved in the path signature representation, enabling accurate discrimination over different trajectories' signatures. Lastly, path signatures provide a single hyperparameter (the number of desired collections k). By adjusting k, we can control the trade-off between representational quality and computational complexity, allowing for efficient learning and generalisation. However, we observe that increasing k leads to an exponential increase in the length of the signature, which imposes a limit to agents with limited computation resources.

4.5 Signature Time Complexity

We now briefly discuss the upper-bound complexity for computing path signatures and compare it to Pavse et al.'s work [7], which computes the averages of the trajectory states. Given Eq. 5 and Fig. 2, it should be easy to see that path signatures can be computed in time $\mathcal{O}(t \cdot d^k)$, where t is the number of samples in a trajectory, d is the number of dimensions, and k is the depth of the signature. In contrast, Pavse et al.'s work [7] uses the average over the current and previous states. This does not work well when different trajectories

average to the same value, but it is not an issue for signatures due to their uniqueness. Using averages is not an issue in Pavse et al.'s work (or in IRL in general), which computes an artificial reward signal at each timestep. However, it also quickly becomes costly because the method computes the average of all sub-trajectories t times at each epoch and trajectories are traversed multiple times ($\mathcal{O}(d \cdot t^3)$). The cost of computing signatures increases linearly with respect to the episode length, whereas the cost of computing Pavse et al.'s averages increases exponentially. Moreover, path signatures increase exponentially according to the number of dimensions d, which is constant for all environments. The main parameter affecting the cost of computing signatures in CILO is the depth k. Fig. 5 compares the costs for Ant-v2 — the environment with the largest state representation ($d = 111$). The figure shows that the cost of computing signatures is lower than that of computing averages for signatures with depth up to 5. In Ant-v2 (the environment with the highest number of dimensions) with signature depths up to 5 and episode length at least 729, the cost of computing signatures is lower than computing Pavse et al.'s averages. Moreover, recall we only needed to use depth 2 to obtain superior performance.

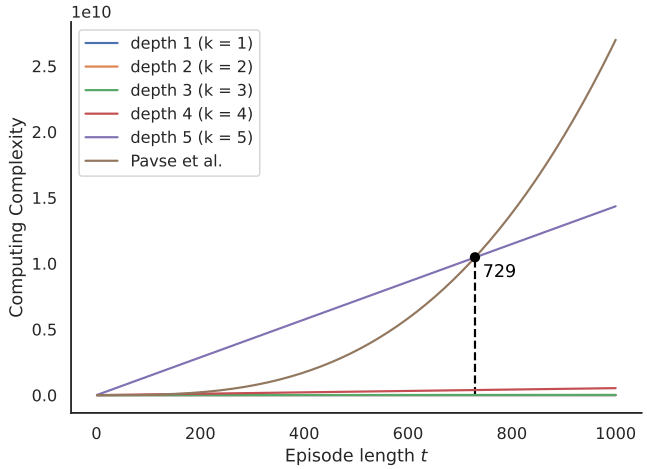


Figure 5. Upper-bound time complexity for signature computation and average.

References

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, 1(5):834–846, Sep 1983.

[2] I. Chevyrev and A. Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[3] R. Coulom. *Reinforcement learning using neural networks, with applications to motor control*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2002.

[4] T. Erez, Y. Tassa, and E. Todorov. Infinite horizon model predictive control for nonlinear periodic tasks. *Manuscript under review*, 4, 2011.

[5] N. Gavenski, J. Monteiro, R. Granada, F. Meneguzzi, and R. C. Barros. Imitating unknown policies via exploration. In *International British Machine Vision Virtual Conference*, pages 1–8, 2020. URL https://www.bmvc2020-conference.com/conference/papers/paper_0774.html.

[6] T. Lyons. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537*, 2014.

[7] B. S. Pavse, F. Torabi, J. Hanna, G. Warnell, and P. Stone. RIDM: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *IEEE Robotics and Automation Letters*, 5(4): 6262–6269, oct 2020.

[8] A. Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[9] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[10] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[11] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4950–4957, 2018.

[12] P. Wawrzyński. A cat-like robot real-time learning to run. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 380–390. Springer, 2009.

[13] W. Yang, T. Lyons, H. Ni, C. Schmid, and L. Jin. Developing the path signature methodology and its application to landmark-based human action recognition. In *Stochastic Analysis, Filtering, and Stochastic Optimization*, pages 431–464. Springer, 2022.

[14] Z. Zhu, K. Lin, B. Dai, and J. Zhou. Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems*, 33: 12402–12413, 2020.