PROJECT REPORT - BCS541

An interactive hyper-beam based visualization method of n-th dimensional reward landscapes

Author
ELRHARBI-FLEURY Yannis - 20235708
Brain and Cognitive Science department
Korea Advanced Institute of Science & Technology
17th of December 2023

1 Background

Reinforcement Learning is a heuristic search framework whose purpose is to find the strategy maximizing an agent's reward in a given environment. This framework's ultimate performance relies on environment-dependent hyper-parameters [1] and is inherently subject to deception [2], additionally it benefits from parameters space of high dimensions.

Hence we believe that designers or researchers should have tools allowing for the intuitive visualization of these spaces. Ultimately, such tools could be leveraged in other areas of Machine Learning, specifically evolutionary computation which requires a thorough understanding of the genotype-phenotype mapping (parameters and output) [3] [4].

All of the following results are reproducible, and the tool's output was meant to be interactive. Hence we highly encourage the reader to visit the project's GitHub repository [5].

2 Problem

Given an agent's parameters $\Theta = (\theta_i)_{1 \leq i \leq N} \in E$, we wish to visualize its surrounding reward landscape as it learns.There are two problems at hand for the design of our tool:

1. Intuitively visualize the immediate surrounding reward landscape $\mathcal{B}'(\Theta, r) = \{M \in E \mid d(\Theta_M, \Theta) \leq r\}$.
2. Observe the dynamics of its learning process (i.e. account for movements in N dimensions).

3 Methods

Firstly, we use a decompositional algorithm [6] to sample $\mathcal{B}'(\Theta, r)$:

- With $S \sim \mathcal{N}(0, 1)$, we uniformly sample the centered hyper-sphere of radius r by using $S' \sim r \times \frac{S}{\|S\|}$.
- Then $\mathcal{B}'(\Theta, r)$ is found by sampling the segments $[\Theta - S', \Theta + S']$ with any distribution U.

As seen in Figure 3, very much like an inverse of the Box-Muller transform [7], this decomposition allows us to simply map our samples to an image or 3D landscape. Indeed, the segments $[\Theta + S', \Theta - S']$ are arranged into lines, with the reward of the sampled policies from U as values for its pixels. To ensure a better continuity of surrounding structures on our mapping, these lines are arranged by proximity.

A natural visualization of an agent's surroundings as it follows a trajectory, would be to repeat that visualization process along the hyper-beam defined by subsequent hyper-planes shifted along its vector trajectory. In Figure 1, we show the steps of the final method given a trajectory $\vec{AB} = u$:

- Find an orthonormal basis of the hyperplane orthogonal to u with a Gram-Schmidt orthogonalization process [8] applied to the random family $(< u, u > v - < v, u > u)_v \sim_{\text{uniform}}$ (step (a)).
- Use the hyper-plane's basis to sample the hyper-ball (step (b) and (c)).
- Shift it along u (step (d)).

Finally, Figure 3 shows how the user can interactively move along the beam and visualize the appropriate reward landscape after each policy has been evaluated (like an n-th dimensional MRI).

Note that any distribution U can be used (see Figure 2). However we found that due to the relatively low number of samples used, uniform distributions gave less meaning to the observed structures as there is a greater variance between the distance of two points sampled along different directions, hence the arrangement of the lines in our mapping was perturbed. We preferred to use bounded normal distributions instead (but it's up to the user!).

4 Results

We developed our tool in Python to be compatible with Stable-baselines 3 [9], hence users easily have access to diverse state-of-the-art algorithms to test our tool.

As an example, in Figure 4, we trained a low parameter SAC model on the Cartpole environment and observed the reward landscape between its 1,000th and 40,000th training step. In the early stage of its learning process (depth 0), the agent is in a low reward hole surrounded by policies of higher rewards. As it is learning, (depth 30), it progressively starts to find bubbles of reward and becomes better than its immediate surroundings. Finally (depth 49), it becomes clear that the agent is locally on the right track to maximize its reward.

5 Discussion

A low parameter SAC model was used for the example because the complexity of the Gram-Schmidt process applied to find the basis of an hyperplane is $o(N^3)$. Hence it becomes a major bottleneck in higher dimensions. But is that process truly necessary for our tool?

Indeed, the reason we sample surfaces on the orthogonal hyperplane is to prevent overlapping between subsequent layers. However, two random uniform unit vectors are almost surely orthogonal as $N \to \infty$ [10], therefore it appears that we could forgo of that hyper-plane constraint in higher dimensions and drastically increase the scaling potential of our tool.

References

[1] M. Franceschetti, C. Lacoux, R. Ohouens, A. Raffin, and O. Sigaud, "Making reinforcement learning work on swimmer," *arXiv preprint arXiv:2208.07587*, 2022.

[2] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, 2011.

[3] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[4] S. Gavrilets, "A dynamical theory of speciation on holey adaptive landscapes," *The American Naturalist*, vol. 154, no. 1, pp. 1–22, 1999.

[5] "Project github repository," https://github.com/yannisEF/NNMRI, accessed: 2023-12-17.

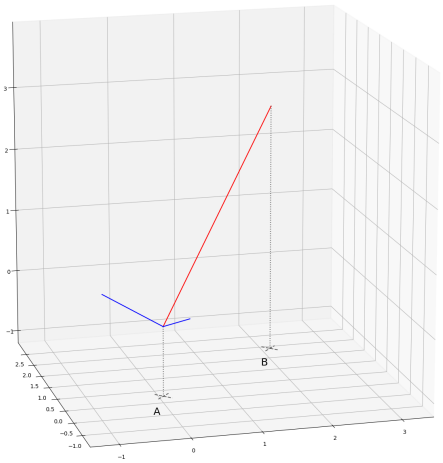
[6] R. Harman and V. Lacko, "On decompositional algorithms for uniform sampling from n-spheres and n-balls," *Journal of Multivariate Analysis*, vol. 101, no. 10, pp. 2297–2304, 2010.

[7] D. W. Scott, "Box–muller transformation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 3, no. 2, pp. 177–179, 2011.

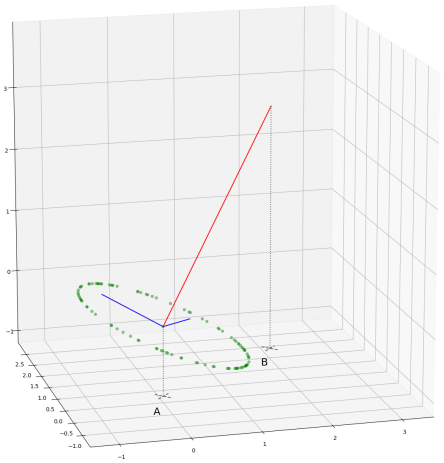
[8] S. J. Leon, Å. Björck, and W. Gander, "Gram-schmidt orthogonalization: 100 years and more," *Numerical Linear Algebra with Applications*, vol. 20, no. 3, pp. 492–532, 2013.

[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*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

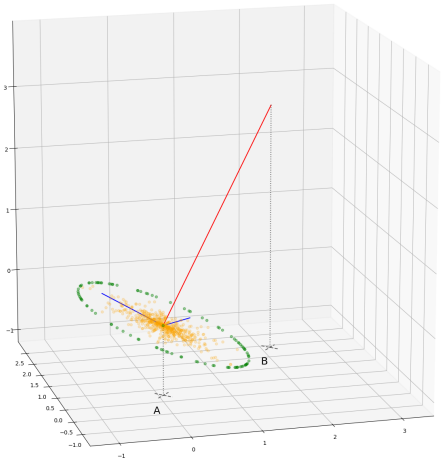
[10] "Why are randomly drawn vectors nearly perpendicular in high dimensions," Mathematics Stack Exchange, accessed: 2023-12-17. [Online]. Available: https://math.stackexchange.com/q/995678



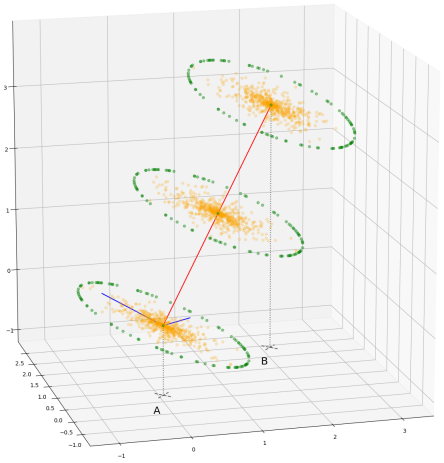
(a) Find orthogonal hyperplane



(b) Sample an hyper-sphere

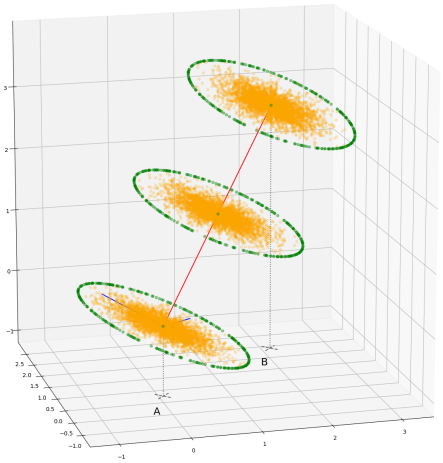


(c) Sample the hyper-ball

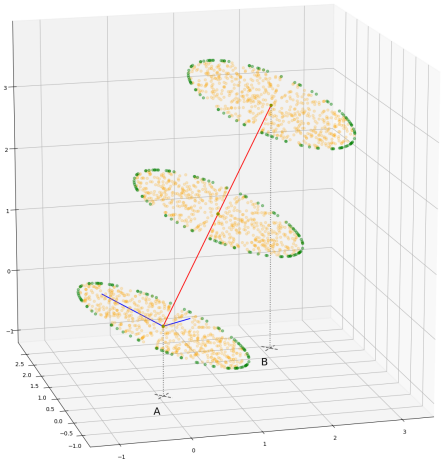


(d) Create and sample the hyper-beam

Figure 1: 3D illustration of the steps taken to sample the hyper-beam between two input policies A and B. (a) A basis of the orthogonal hyperplane is found, which allows us to compute the coordinate of each future point. (b and c) We then use a decompositional algorithm to sample an hyper-ball around our central point. (d) Finally, we shift the hyperplane along the input vector to create an hyper-beam between A and B.



(a) Increase the number of samples



(b) Uniform Monte Carlo sampling

Figure 2: Higher number of samples or better environment-suited distributions can increase the likelihood of finding nearby structures (here (a) bounded normal (b) uniform Monte-Carlo).

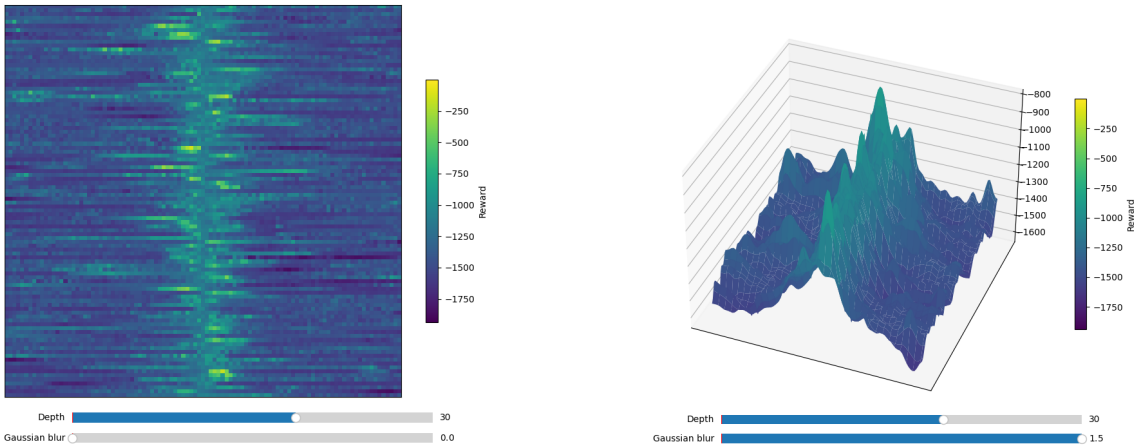
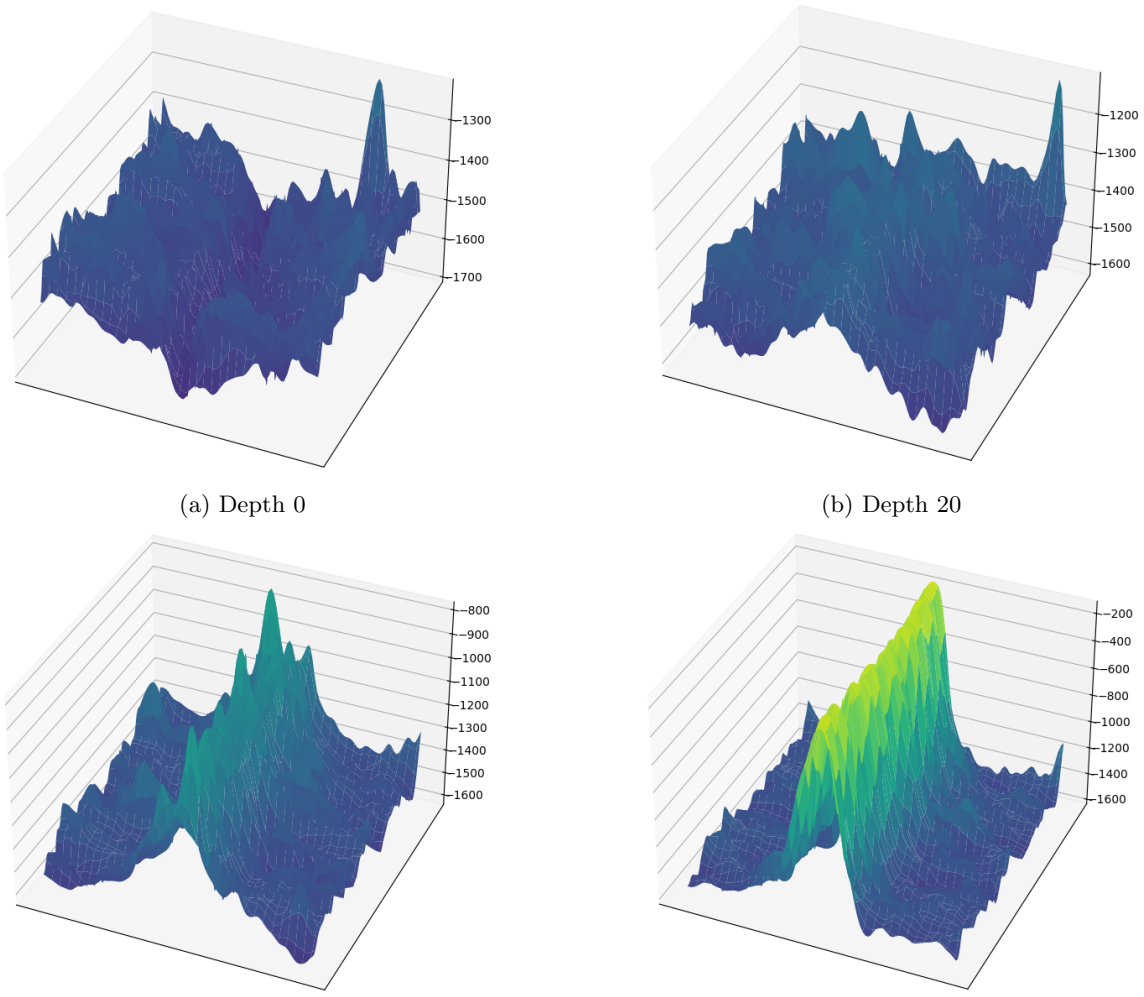


Figure 3: Example output of the tool, available in 2D or 3D. Segments become lines of pixels, whose values are the sampled policies' rewards on a given environment. Once the hyper-beam has been fully evaluated, the user can use sliders to freely move along the layers of the beam.



(a) Depth 0

(b) Depth 20

(c) Depth 30

(d) Depth 49

Figure 4: Snapshot of the 50-layers deep hyper-beam of a policy of 4,000 parameters trained by SAC with SB3 default's parameters on the Cart-pole environment between its 1,000 and 40,000 training steps, with a Gaussian blur of variance 1.5 applied to the final surfaces.