Symbolic-Model-Based Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We investigate using symbolic regression (SR) to model dynamics with mathematical expressions in model-based reinforcement learning (MBRL). While the primary promise of MBRL is to enable sample-efficient learning, most popular MBRL algorithms rely, in order to learn their approximate world model, on black-box over-parametrized neural networks, which are known to be data-hungry and are prone to overfitting in low-data regime. In this paper, we leverage the fact that a large collection of environments considered in RL is governed by physical laws that compose elementary operators e.g $\sin$, $\sqrt{\ }$, $\exp$, $\frac{\mathrm{d}}{\mathrm{d}t}$, and we propose to search a world model in the space of interpretable mathematical expressions with SR. We show empirically on simple domains that MBRL can benefit from the extrapolation capabilities and sample efficiency of SR compared to neural models.

1 Introduction

Motivated by real-world applications, such as robotic controlling [1, 2], one of the main goals of the field of Reinforcement Learning (RL) is to learn to control systems with non-linear (potentially stochastic) dynamics. Most control systems, irrespective of the task to solve (i.e. rewards maximization) have in common the fact that their dynamics are governed by physical laws. They are usually expressed with mathematical equations connecting the next state with the system's past states and controller's actions with operators such as time-derivatives, trigonometric operators, power functions. Solving a task usually involves implicitly a good understanding of the dynamics, e.g. goal reaching.

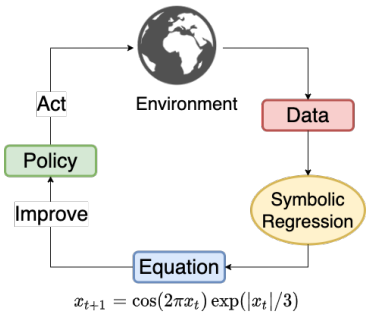


Figure 1: Symbolic-MBRL

For instance, in the classic CartPole environment [3, 4], the system's state is described by $(x, \dot{x}, \theta, \dot{\theta})$ where x (resp. $\dot{x}$) denotes the position (resp. speed) of the cart along the x-axis and θ (resp. $\dot{\theta}$) is the angle (resp. rotation) of the pole w.r.t the cart. The agent's action a affects the system's state according to the following equations [4, 5]:

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-K_{\text{mag}}a - m_p l\dot{\theta}^2\sin\theta}{m_c+m_p}\right)}{l\left(\frac{4}{3} - \frac{m_p\cos^2\theta}{m_c+m_p}\right)} \quad \ddot{x} = \frac{K_{\text{mag}}a + m_p l\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right)}{m_c+m_p} \tag{1}$$

where $g, K_{\text{mag}}, m_p, m_c, l$ are all constants. Systems with impossible states, safety or physical constraints, e.g. the joint of a robotic arm cannot exceed a certain angle [6, 7], can be expressed via piece-wise expressions. When the environment is stochastic, probabilistic distributions appear. Note that the parametrization of the state and action spaces has an impact on the symbols that are necessary to describe the system dynamics; for instance, if $\dot{x}$ and $\dot{\theta}$ were not in the CartPole agent's

Submitted to NeurIPS 2022 AI for Science Workshop, Attention Track.

observation, derivative operators would have to appear in the equation. Generally, the larger the state-space parametrization, the "shorter" equation is.[1]

Model-Based Reinforcement Learning (MBRL) is a class of RL algorithms that involves a two-step procedure repeated until task termination; a) learn from data a forward dynamics model (possibly stochastic) function f that maps current state s_t and action a_t to next state s_{t+1} and b) derive a policy from this model. Though they have been shown to learn faster than model-free algorithms in many applications [8, 9], MBRL algorithms have proven to be hard to train as their performance depends on mutually dependent components [10]. In this work, we challenge the go-to approach of using over-parametrized feed-forward neural networks to approximate f as they are prone to overfitting when collected data does not have enough coverage of state and action spaces. We propose to leverage prior knowledge on operators that *could* appear in the environment dynamics equations, e.g. $\sin, \sqrt{\ }, \exp, \frac{\mathrm{d}}{\mathrm{dt}}$.

Manipulating expressions to fit data is exactly the objective of SR algorithms that select f within a large family of expressions through composition of operators, constants and variables, as opposed to gradient-descent of over-parametrized models, e.g. neural networks (NNs). The latter have more degrees of freedom and are easier to optimize, but prone to overfitting in low-data regimes, whereas SR has recently shown excellent extrapolation capabilities [11, 12, 13]. Furthermore, SR provides an interpretable and differential model, interesting properties for RL, which we develop in section 4.

Contributions. We propose a novel approach to dynamics modelling in control problems, which we call Symbolic-Model-Based RL, that uses mathematical expressions to model dynamics. To our knowledge, this is the first work that proposes to leverage SR to find an interpretable function f that best maps state-action pairs (s_t, a_t) to the next state s_{t+1}. We provide empirical evidences in simple domains, where our method largely outperforms the over-parametrized approaches, that SR provides faster better dynamics models that generalize to unseen states-action pairs.

2 Related work

MBRL. We study the RL problem where an agent interacts with an unknown environment, formalized by a Markov Decision Process $(\mathcal{S}, \mathcal{A}, p, r)$ where $\mathcal{S}$ (resp. $\mathcal{A}$) represents the state (resp. action) space, p the transition dynamics and r the reward functions. MBRL algorithms are a class of RL algorithms that ground control policies on a model of the dynamics of environment. Most of approaches alternate two steps repeatedly: i) collect data $\mathcal{D}$ with the current policy and learn an approximate model f of the environment's dynamics, fitting $\mathcal{D}$ as in supervised learning (SL), i.e.:

$$f^* = \arg\max_{f \in \mathcal{F}} \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \, \mathcal{L}(s_{t+1}, f(s_t, a_t)) \tag{2}$$

where $\mathcal{F}$ is a family of functions, e.g. neural networks or Gaussian processes, and $\mathcal{L}$ a loss function that depends on the nature of f; ii) then, simulate transitions with the model f and optimize the policy accordingly. Note that in this work we consider reward and termination functions learned in the same way as dynamics, even though MBRL algorithms [14, 15] often consider them provided to the agent. We refer the reader to an exhaustive MBRL review [16]. Step i) usually faces classic under/over-fitting issues of SL (w.r.t the state and action space), causing sub-optimal task performance. The design of predictive model has proven to be very challenging; though they are fast to train, Gaussian Processes (GPs) under-fit on complex dynamical systems [17], whereas over-parametrized functions, i.e. neural networks, can express complex (and high-dimensional) dynamics, but are prone to overfitting. To avoid overfitting, one can a) acquire more data, but this often comes with exploration challenges, b) use regularization, i.e. making the model simpler, in the form of priors, e.g. GPs' kernel function or Bayesian neural networks [18, 19], or c) use ensembles [20, 21, 22]. [14] uses probabilistic NNs (b) in combination with ensembles (c) to make MBRL agents uncertainty-aware for better planning.

Symbolic regression. SR, the science of inferring mathematical laws from experimental data by searching over the space of interpretable mathematical expression by manipulating operators, constants and variables, has already proved useful in a variety of domain such as physics [23, 24, 25, 26]. The leading algorithms for SR, excellently reviewed in [11], mainly rely on genetic programming (GenP) [27, 28, 29] by iterating through steps of selection, mutation and crossover until a satisfactory accuracy level is achieved. More recently, neural networks have been applied to SR to identify

[1] The best parametrization to decrease the complexity of predicting an equation from data is an open problem.

qualitative patterns to reduce the search space, either in combination with GenP algorithms [23, 30] or without [31, 32, 33, 34, 12]. Recently, SR has been applied to model-free RL to learn interpretable policies explicitly [35, 36, 37] or via the associated value function [38], but to our knowledge we are the first to consider SR for MBRL. In addition to the immediate interpretability benefit, considering the dynamics model search space $\mathcal{F}$ in Eq. (2) to be the family of short mathematical expressions that contain constants, variables and operators from a given dictionary has the advantages of injecting prior knowledge, smoothness properties, as well as to significantly reduce the size of the search space.

3 Experiments

Our experiments investigate the following questions: **i)** Can SR algorithms help learn predictive models with less samples, **ii)** Do the obtained dynamics equations have interesting properties? **iii)** How is this manifesting in terms of performance (task solving)? We consider deterministic environments, which already represent a substantial of environments in the RL literature [6, 7, 39].

Description of the algorithm. Though the symbolic approach is applicable to any MBRL algorithm, we use Probabilistic Ensembles with Trajectory Sampling (PETS) [14] implemented with *MBRL-lib* [10] as a building block and trade the dynamics neural architecture for an expression optimized via SR. In particular, we use Operon [29], an efficient GenP algorithm, as our base symbolic regressor. Preliminary experiments on fitting data with Eq. (3), showed that Operon had better performance that numerous algorithms, e.g. gplearn [40] (inaccurate), PS-Tree [41] (unnecessarily complicated and piece-wise functions), AI-Feynman [23] (inference is too-long). As in PETS, we maintain an ensemble of 7 dynamics models and select the sequence of actions that maximize rewards using the cross-entropy method [42] on the dynamics simulated with trajectory sampling. We call this model Symbolic-PETS and compare it to MLP-PETS, the original version of PETS.

An illustrating example. For illustrative purposes, we consider a simple one-dimensional state MDP, with episode length 10 and the following dynamics:

$$s_{t+1} = s_t + a_t, \quad r_t = \cos(2\pi s_{t+1}) \exp(|s_{t+1}|/3) \tag{3}$$

As illustrated on Fig. 2, solving this MDP is challenging as it requires sufficient exploration to learn the reward function and avoid falling into local minimas, i.e. staying in relative integers x values without going to better ones. As shown in Fig. 2, MLP-PETS' dynamics models over-fits on the training distribution. Even using an ensemble of 7 models, the uncertainty of the ensemble is not good enough to be leveraged for efficient exploration, leading to a sub-optimal policy. On the other

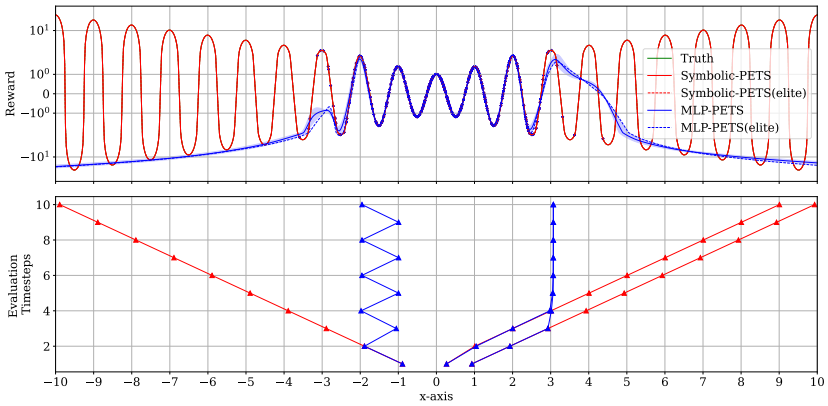


Figure 2: We train the dynamics models on 500 transitions collected by a random (uniform $[-1, 1]$). The top row is the immediate reward function predicted by learned models (evaluated with $a_t = 0$ for clarity) and dots correspond to training data. Elite is the best dynamics model w.r.t to an evaluation set. The bottom row represents 3 evaluation roll-outs after the predictive model was updated:we observe that Symbolic-PETS allows agents to reach better rewarded states. Symbolic-PETS's predicted reward function is $(1.0 \exp((0.333 s_t + 0.333 a_t)) + 2.14e^{-4}) \sin(6.283 x_t + 6.283 a_t - |0| - 4.712)$.

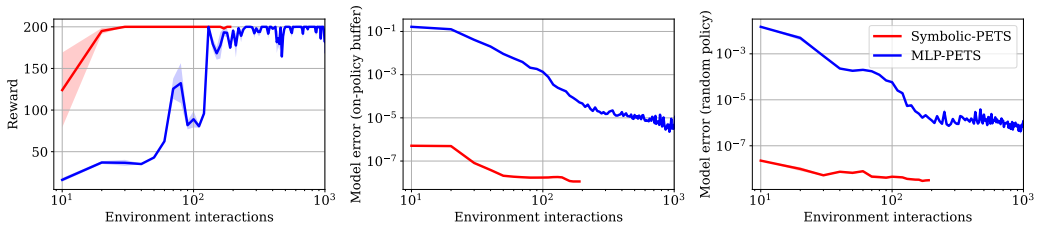


Figure 3: **Symbolic-PETS solves CartPole *very* fast.** Agents are evaluated on 3 episodes with 3 random seeds every 10 transitions. MLP-PETS solves CartPole despite significant model error, suggesting that solving CartPole does not require a perfect understanding of the environment.

hand, Symbolic-PETS' dynamics models extrapolate really well on unseen states, thus achieving optimal behavior in just one update. In Appendix A.1.2, we present the evolution of Symbolic-PETS every episodes after just 20 observed random transitions.

CartPole. We consider the continuous CartPole [6], where, as in [14, 10], the termination and reward function are made available to the agent, therefore control is restricted to be a problem of dynamics modelling. We define the model error $\mathcal{L}$ in Eq. (2) as the MSE averaged over output dimensions.

In a first experiment, we explore the capabilities of both symbolic and MLP regressors on the data generated by a random policy on CartPole. We collect an evaluation dataset of $5e^4$ transitions, enough to have good state-action coverage. We then train the two dynamics models on training datasets of growing sizes and plot the model error in Fig. 5 (App. A.2.2). The symbolic model predicts the following equations with the accuracy reached by the MLP in two order of magnitude less interactions:

$$
\begin{aligned}
x_{t+1} &= x_t + 0.02\dot{x}_t \\
\theta_{t+1} &= \theta_t + (0.02\dot{\theta}_t + 0.015)/\cos(0.035 * \dot{\theta}_t) - 0.015 \\
\dot{x}_{t+1} &= (0.002\theta_t + 2.34e^{-4}\dot{\theta}_t a_t + 1.0) \times (\dot{x}_t + 0.195a_t - \sin(0.015\theta_t) + 3.23e^{-5}) \\
\dot{\theta}_{t+1} &= \cos(0.195\theta_t)(0.314\theta_t + \dot{\theta}_t - 8.97e^{-1}a_t \times (-0.031\dot{\theta}_t - 2.014)\frac{(0.016\dot{\theta}_t - \cos(1.053\theta_t))}{(6.173 - 0.002\theta_t)})
\end{aligned}
\tag{4}
$$

Interestingly, predicted equations are a bit different than in Eq. (1), though we can notice constants such as the time-discretization interval 0.02. What could be as missing terms can be explained by limited development as θ and x have small values because of CartPole's constraints. In Fig 3, we demonstrate that Symbolic-PETS is able to solve CartPole in just 20 interactions with the environment, a *state-of-the-art* performance to our knowledge.

4 Discussion

We demonstrated on simple environments that SR can learn better predictive models with many less samples than neural network and that they extrapolate well to unseen state-action pairs much better. Though [11] showed promising results on real-world regression datasets with input dimensions up to 124 (with a single output dimension), SR still remains to be scaled to higher output dimensions, with challenges including parallelizing regressor training of each output dimension or pixel observations.

SR can have significant impact on multiple RL research topics, e.g partially-observable environments, or meta- and continual-RL thanks to sample efficiency of SR. Having an interpretable (and differentiable) dynamics model can enable constrained reinforcement learning [43], environment design [44] and domain randomization [45] to enable "Sim2Real" transfer to new environments. Yet, SR current tools are not as modulable as spatio-temporal NNs. To be useful in the aforementioned RL domains, we aim at designing algorithms and their implementation to respect these non-exhaustive properties of neural networks; i) better uncertainty-awareness (can express distributions), ii) learning can be initialized to some known expression, iii) allow multi-step predictions and learning, iv) allow parallelization for all output dimensions and v) preprocessing of inputs.

References

[1] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

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

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[5] Razvan V Florian. Correct equations for the dynamics of the cart-pole system. *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

[6] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[7] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.

[8] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

[9] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. *Advances in neural information processing systems*, 27, 2014.

[10] Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021.

[11] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio Olivetti de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351*, 2021.

[12] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *arXiv preprint arXiv:2204.10532*, 2022.

[13] Gabriel Kronberger, Fabricio Olivetti de França, Bogdan Burlacu, Christian Haider, and Michael Kommenda. Shape-constrained symbolic regression—improving extrapolation with prior knowledge. *Evolutionary computation*, 30(1):75–98, 2022.

[14] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[15] Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *Advances in Neural Information Processing Systems*, 33:14156–14170, 2020.

[16] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[17] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression, 2014.

[18] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[19] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. *Advances in neural information processing systems*, 30, 2017.

[20] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993.

[21] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS workshop on bayesian deep learning*, volume 192, 2016.

[22] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[23] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: a physics-inspired method for symbolic regression, 2020.

[24] Silviu-Marian Udrescu and Max Tegmark. Symbolic pregression: Discovering physical laws from raw distorted video. *Physical review. E*, 103 4-1:043307, 2021.

[25] M. Cranmer, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, Rui Xu, Kyle Cranmer, David N. Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *ArXiv*, abs/2006.11287, 2020.

[26] Anja Butter, Tilman Plehn, Nathalie Soybelman, and Johann Brehmer. Back to the formula – lhc edition. 2021.

[27] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[28] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H Moore. Learning concise representations for regression by evolving networks of trees. *arXiv preprint arXiv:1807.00981*, 2018.

[29] Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon c++: An efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, page 1562–1570, New York, NY, USA, 2020. Association for Computing Machinery.

[30] Brenden K Petersen, Mikel Landajuela Larma, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

[31] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

[32] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[33] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales, 2021.

[34] Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences. *arXiv preprint arXiv:2201.04600*, 2022.

[35] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2021.

[36] Jacob F Pettit, Brenden K Petersen, FL Silva, Dale B Larie, RC Cockrell, Gary An, and Daniel M Faissol. Learning sparse symbolic policies for sepsis treatment. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2021.

[37] Mathurin Videau, Alessandro Leite, Olivier Teytaud, and Marc Schoenauer. Multi-objective genetic programming for explainable reinforcement learning. In *European Conference on Genetic Programming (Part of EvoStar)*, pages 278–293. Springer, 2022.

[38] Jiří Kubalík, Erik Derner, Jan Žegklitz, and Robert Babuška. Symbolic regression methods for reinforcement learning. *IEEE Access*, 9:139697–139711, 2021.

[39] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

[40] Trevor Stephens. gplearn. `https://github.com/trevorstephens/gplearn`, 2016.

[41] Hengzhe Zhang, Aimin Zhou, Hong Qian, and Hu Zhang. Ps-tree: A piecewise symbolic regression tree. *Swarm and Evolutionary Computation*, 71:101061, 2022.

[42] Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

[43] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International Conference on Machine Learning*, pages 7390–7399. PMLR, 2021.

[44] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

[45] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

A Experiment details

We perform our experiments with 1 GPU and 1 CPU.

A.1 Toy example

A.1.1 Hyper-parameters.

MLP-specific hyper-parameters. We consider a deterministic neural network with 4 hidden layers of size 200 with SiLU activation, trained with Adam optimizer during a maximum of 2000 epochs with batch size 256 and patience epochs 25 (meaning training stops when loss/ evaluation score does not progress more than 0.01 relatively), learning rate $7.5e-4$ and weight decay $3e-5$. Inputs are normalized.

Symbolic-specific hyper-parameters. We use Operon with 5 local iterations, 10000 generations, 10 threads, population size 5000 and allowed symbols are `add,sub,mul,div,constant,variable,sin,exp,abs`.

Action optimizer. We use CEM with planning horizon 3, 10 iterations, elite ratio 0.1, population size 1000, alpha 0.1 and clipped normal action distribution. Out of the ensemble of 7 predictive models, only the 3 elite (w.r.t evaluation set) ones are used.

A.1.2 Results

We present the evolution of Symbolic-PETS every episodes after just 20 random transitions in Fig. 4

A.2 CartPole

A.2.1 Hyper-parameters.

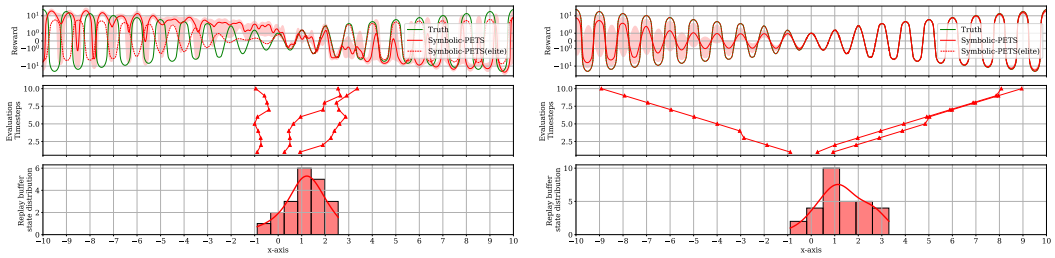
MLP-specific hyper-parameters. Same as for the toy example.

Symbolic-specific hyper-parameters. We use Operon with 5 local iterations, 10000 generations, 10 threads, population size 5000 and allowed symbols are `add,sub,mul,div,constant,variable,sin,cos,pow`.

Action optimizer. We use CEM with planning horizon 15, 5 iterations, elite ratio 0.1, population size 350, alpha 0.1. Out of the ensemble of 7 predictive models, only the 3 elite (w.r.t evaluation set) ones are used.

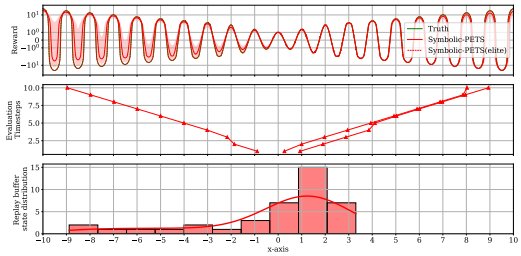
A.2.2 Analysis of results

Interestingly, one can notice in Fig. 3 the performance (in terms of reward and model error) fluctuate a bit as the number of interactions grows (contrary to Fig. 5). This can be explained by the fact that random samples are the most informative transitions in terms on environment understanding, whereas on-policy transitions (whose proportion grows during learning) are all located in a very narrow part of the state-space (pole standing still)

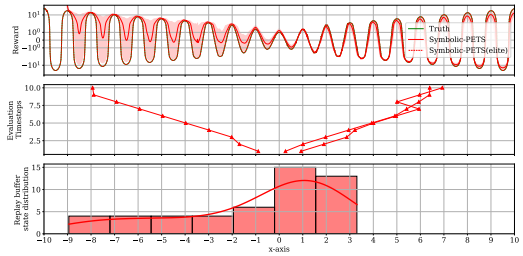


(a) 20 environments steps

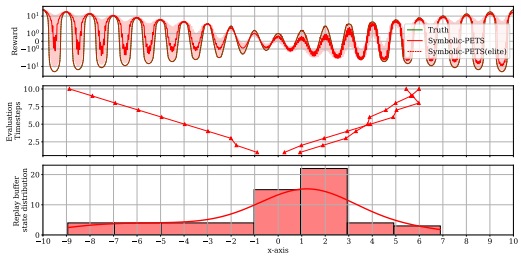
(b) 30 environments steps



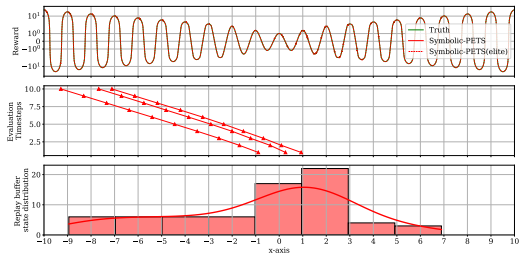
(c) 40 environments steps



(d) 50 environments steps



(e) 60 environments steps



(f) 70 environments steps

Figure 4: Top row is the reward function evaluated with $a_t = 0$ for clarity learned by the Symbolic-PETS agents (elite is the best dynamics model w.r.t to an evaluation set). Middle rows represents 3 evaluation roll-outs after a predictive model update. Bottom row is the training replay buffer state distribution.

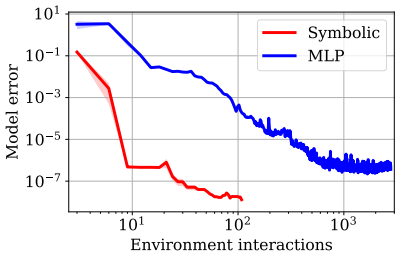


Figure 5: Model error of the MLP and symbolic regressors (averaged over different seeds) on data generated by a random policy.