ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

A model-based approach to meta-Reinforcement Learning: Transformers and tree search

Brieuc Pinon and Raphaël Jungers and Jean-Charles Delvenne [*]

UCLouvain, Department of Mathematical engineering
Louvain-la-Neuve - Belgium

Abstract. Meta-learning is a line of research that develops the ability to leverage past experiences to efficiently solve new learning problems. In the context of Reinforcement Learning (RL), meta-RL methods demonstrate a capability to learn behaviors that efficiently acquire and exploit information on a set of related tasks.

The Alchemy benchmark has been proposed in [1] to test such methods. Alchemy features a rich structured latent space that is challenging for state-of-the-art model-free RL methods. These methods fail to learn to properly explore then exploit.

We develop a model-based algorithm. We train a model whose principal block is a Transformer Decoder to fit the symbolic Alchemy environment dynamics. Then we define an online planner with the learned model using a tree search method. This algorithm significantly outperforms previously applied methods on the symbolic Alchemy problem.

Our results reveal the relevance of model-based approaches with online planning to perform exploration and exploitation successfully in meta-RL.

1 Introduction

Deep Learning methods have been successfully applied to various problems such as in image processing, natural language processing, and games. However, these solutions usually require many samples to solve any new task. The developing field of meta-learning addresses this issue. To solve a task of interest, the meta-learning paradigm supposes access to data from other tasks in relation to the task of interest. A meta-learning method can then take advantage of this supplementary data to learn an efficient learning algorithm for the task of interest.

Recently proposed in [1], the Alchemy benchmark aims to test and understand proposed meta-RL algorithms. In this benchmark, the agent faces an environment with a new hidden dynamics in each episode. To maximize his rewards, the agent must understand this dynamics through experimentation, then exploit that knowledge over the episode. Two versions of the benchmark have been proposed, a 3D visual version and a symbolic version. We focus on the symbolic version in this paper.

[*]Raphaël Jungers is a FNRS honorary Research Associate. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 864017 - L2C. RJ is also supported by the FNRS (Chist-Era Druid-net).

One approach to meta-RL is to cast the problem as a partially observable Markov Decision Process (POMDP), where a latent space represents the hidden dynamics. Model-free RL methods that support partial observability can then be used for meta-RL [2, 3]. The authors of Alchemy tested state-of-the-art model-free RL methods on it. Their experimentation revealed a failure of these model-free RL methods to learn a policy that efficiently explores for information and then exploits it both on the 3D visual and on the symbolic case [1].

We investigate the use of a model-based algorithm with online planning on the symbolic version of Alchemy and show significant improvements. This result shows both:

- the capability of Deep Learning methods, in particular the Transformer architecture, to fit complex dynamics in environments where model-free RL methods fail. These dynamics emerge naturally in meta-RL problems where the latent space is a critical part;
- the strength of online planning algorithms in challenging environments, such as those that arise in meta-RL where the reward is delayed between the gain in information from exploration and its exploitation.

2 Methods

We define the general problem we address in 2.1, and describe the symbolic Alchemy benchmark in 2.2. We present the training process and architecture of the neural network model fitting the dynamics in 2.3. We then explain the online planner we use on top of this model in 2.4.

2.1 Problem formulation

We define a partially observable Markov decision process (POMDP) as a tuple $(S, A, T, P, \Omega, O, N)$ where S is the set of states, A is the set of actions, T is the conditional transition probability distribution over states and rewards $T(s_{t+1}, r_{t+1}|s_t, a_t)$ where $s_{t+1}, s_t \in S$, $a_t \in A$ and $r_{t+1} \in \mathbb{R}$, and P is the probability distribution over the initial states. The set Ω defines the possible observations and O is a conditional probability distribution $O(o_t|s_t, a_t)$ describing the link between the state and action pairs with the observations given to the agent. The number N gives the number of steps in one episode.

In our meta-learning problem, at the start of each episode, latent variables are sampled. These hidden variables define the dynamics until the end of the episode. This fits in the definition of a POMDP. Our goal is to define an agent that maximizes its expected sum of rewards in a POMDP, allowing us to apply it to the meta-learning problem.

2.2 Symbolic Alchemy

Symbolic Alchemy as defined in [1] links an unknown "chemistry" to the dynamics of an environment which consists of a variety of stones and potions. Where

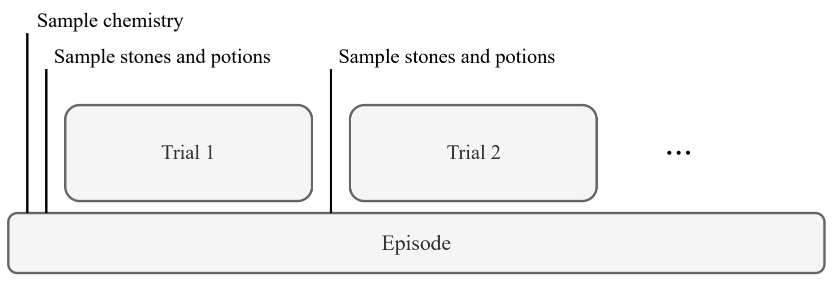

Fig. 1: Decomposition of one episode of symbolic Alchemy: a chemistry is sampled for the episode, then trials are interleaved with resamples of new potions and stones.

potions can be used to change stones, and render them more valuable. Here, the chemistry determines the effect of applying a potion on a stone and a new chemistry is sampled each time an episode starts. For example, applying the red potion on a small purple stone might make it large in one episode and increase its value, but might change its color in another and decrease its value.

In symbolic Alchemy, there are three types of objects that determine a state: the chemistry; the stones, their respective 3 visual features and associated reward; and the potions with their respective colors. The features of the stones and potions are directly observable, but the chemistry is not.

The agent can do several actions. He can apply a potion on a stone, this can change its perceptual features and reward. He can also collect the reward associated with a stone (which can only be done one time for each stone). The hidden chemistry determines the effects of potions on stones and the possible features of stones.

An episode unrolls as follows: a chemistry is sampled among 167,424 others at the start and is kept constant during the whole episode; then a sequence of 10 trials follows. Each trial is further decomposed in: sampling a set of stones and potions; then 20 time steps where actions can be taken to understand the chemistry through experiments, generating high-value stones by using potions, and collecting rewards associated to the stones. The decomposition of one episode is given on Figure 1.

2.3 Learning a model

2.3.1 Data

We sample a dataset of trajectories prior to any learning and planning. We simply use a uniform policy over the action space to generate the trajectories. We define a supervised learning problem from this data to learn a model of the dynamics.

While this method is sufficient to learn an accurate model in the case of symbolic Alchemy, other environments might need interleaving data generation

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

and model learning to get sufficient coverage of the state and action spaces.

2.3.2 Model architecture

The architecture is presented in Figure 2. Our model is based on a Transformer Decoder using causal attention for its main block (3 layers with 256 units each). It receives as input a sequence of observations, rewards, and actions. From the sequence, it outputs a prediction for the next observation and reward at each time step.

In the prediction head for the next observation, a Recurrent Neural Network (RNN) allows our neural network to model non-independent probability distributions over the dimensions of the next observation. In addition to the RNN, a direct linear layer is used to improve the optimization process.

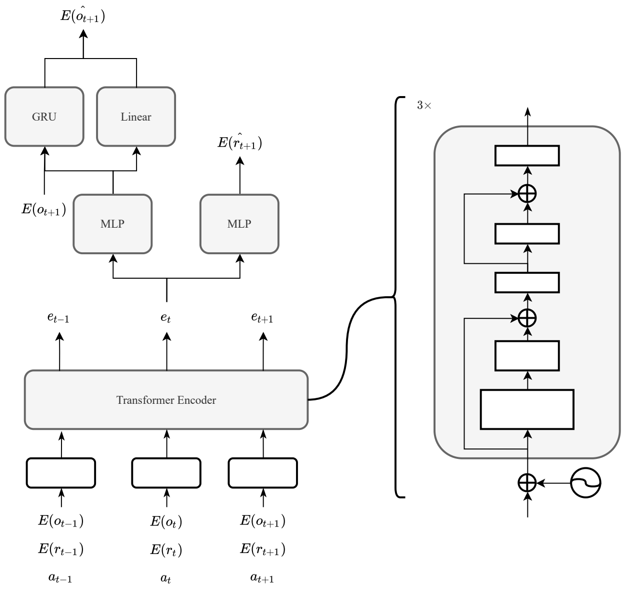

Fig. 2: A neural network architecture to fit trajectories. At the bottom, the sequence of observations and rewards is first transformed into categorical variables. Then the sequence is passed into a Transformer block with positional encoding and causal attention. Independent heads predict the next observation and reward in the sequence from the output of the Transformer.

2.4 Tree search with stochastic transitions

We define the model-based online planning algorithm we use at each step to decide the action to take. Note that this planning algorithm uses for its simulations the learned model defined in the previous section.

We construct our online planner on top of the tree search planner presented in [4]. The tree search algorithm grows iteratively a tree where each branch represents a potential sequence of actions starting at the current state. The search procedure balances exploration and exploitation of promising branches by learning Q-values at each node. The original algorithm supposes full observability and deterministic transitions. We adapt it to support partial observability and stochastic transitions.

To deal with partial observability, we use a common transformation that casts POMDPs as equivalent MDPs where states correspond to the complete history of observations up to that point. Note that even if the original POMDP was deterministic, the resulting MDP is stochastic in general.

To support stochastic dynamics, we adapt the tree search algorithm by adding chance nodes as children of decision nodes. At chance nodes, we branch different paths representing different stochastic realizations of the system dynamics. For each chance node, we use a fixed number of samples from which stochastic branches are created.

3 Results and discussion

We compare our method against the results obtained in [1]. They tested V-MPO, a state-of-the-art model-free RL method, applied with a Transformer architecture improved for model-free RL [5]. They also implemented an ideal observer, which is a Bayes-optimal agent that has the best performance achievable on the benchmark. This agent is not directly comparable to the others, since its implementation relies on prior knowledge of the problem. For further insights into the agent performances, they also implemented Random Heuristic which is a simple baseline.

Table 1: Comparison with the V-MPO RL method and the baselines given in [1]. We give the scores with the standard error (250 runs). The number of expansions refers to the number of nodes explored in the tree search procedure.

Agent	**Score on symbolic Alchemy**
Ours (# expansions 100)	79.3 ± 1.6
Ours (# expansions 500)	161.8 ± 3.9
Ours (# expansions 1.250)	207.1 ± 3.5
Ours (# expansions 2.500)	220.5 ± 3.2
Ours (# expansions 5.000)	229.5 ± 3.9
Ours (# expansions 10.000)	251.5 ± 4.5
V-MPO	155.4 ± 1.6
Ideal Observer	284.4 ± 1.6
Random Heuristic	145.7 ± 1.5

We sample 10^6 episodes to construct our dataset of trajectories from which we learn our model. In contrast, the V-MPO agent is trained using 10^9 episodes.

See Table 1 for the comparison, we test our method with different magnitudes of computational resources going into online planning.

The comparison reveals that our method surpasses the model-free RL method V-MPO with several orders of magnitude less data, given sufficient computational resources for online planning. It also shows the importance of planning on this benchmark, since the performance increases significantly with more allocation of resources into planning. However, the marginal gains diminish too fast to reach the optimal performance in our tests.

We also tested variations of the architecture of our model of the dynamics: replacing the Transformer block with an RNN; removing the RNN in the head to predict observations; ablating the direct linear layer in the head to predict the next observation. Each of these changes independently resulted in a failure to correctly fit the dynamics.

Our results demonstrate that a model-based approach with online planning can substantially surpass the performance of a state-of-the-art model-free method on the symbolic Alchemy meta-RL benchmark. Our work shows that while most of the research effort has focused on model-free methods in meta-RL [6], learning to learn might need explicit modeling of the dynamics and planning. Moreover, the use of the Transformer architecture can be critical to model the complex relationship between histories of observations and beliefs upon the system transitions.

References

[1] Jane X Wang, Michael King, Nicolas Pierre Mickael Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, H Francis Song, et al. Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[2] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[3] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[5] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

[6] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.