STRESZCZENIE

Słowa kluczowe:

Dziedziny nauki i techniki zgodne z wymogami OECD:

ABSTRACT

Keywords:

OECD field of science and technology:

CONTENTS

LIST OF ABBREVIATIONS AND NOMENCLATURE

1. INTRODUCTION

Computer science defines Artificial Intelligence (AI) as the intelligent agents in form of any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Progress has been made in developing capable agents for numerous domains using deep neural networks in conjunction with model-free reinforcement learning [7][9][10], where raw observations directly map to agent's actions. However, current state-of-the-art approaches usually are sample inefficient, they require thousands of millions of interactions with the environment, and lack the behavioural flexibility of human intelligence, hence the resulting policies poorly generalize to novel task in the same environment.

The other branching of reinforcement learning algorithms is called model-based reinforcement learning, which gives the agent access to (or learns) a model of the environment. There are many orthogonal ways of using the model: one can use the model for data augmentation for model-free methods[4], some methods use the model as the imagined environment to learn model-free policy in it[5], other methods focus on simulation-based search using the model[12] and there are even methods that integrate model-free and model-based approaches[14]. The model allows the agent to simulate an outcome of an action taken in a given state. The main upside to having the model is that it allows the agent to plan by thinking ahead, seeing what would happen for a range of possible choices, and explicitly deciding between possible options without the risk of the adverse consequences of trial-and-error in the environment - including making poor, irreversible decision. Agents can then distill the results from planning ahead into a policy. Even if the model needs to be learned first it can exploit additional unsupervised learning signals, thus it results in a substantial improvement in sample efficiency over methods that don't have a model. Furthermore, the same model can be used by the agent to complete other tasks in the same environment. Planning is different from learning. In the former the agent samples episodes of simulated experience and updates its policy based on them. In the latter the agent also updates its policy, but this time based on real experience gained through interaction with the environment. It's worth noting the symmetry which yields one important implication: algorithms for reinforcement learning can also become algorithms for planning, simply by substituting simulated experience in place of real experience.

Model-free methods are more popular and have been more extensively developed and tested than model-based methods. While model-free methods forego the potential gains in sample efficiency from using a model, they tend to be easier to implement and tune. The main downside of model-based reinforcement learning is that a ground-truth model of the environment is usually not available to the agent. If an agent wants to use a model in this case, it has to learn the model purely from experience, which creates several challenges. The biggest challenge is that bias in the model can be exploited by the agent, resulting in an agent which performs well with respect to the learned model, but behaves sub-optimally in the real environment. Model-learning in complex domains is fundamentally hard and requires function approximation, so resulting models are inherently imperfect. The performance of agents employing standard planning methods usually suffer from model errors. Those errors compound during planning, causing more and more inaccurate predictions the further plans' horizon.

There are many real-world problems that could benefit from application of general AI system. Company called DeepMind, driven by their experience from creating winning Go search algorithm AlphaZero[12], published AlphaFold[3], a system that predicts protein structure. The 3D models of proteins that AlphaFold generates are far more accurate than any that have come before making significant progress on one of the core challenges in biology. Real-world applications of AI algorithms like this are often limited by the problem of sample inefficiency. In setting with e.g. a physical robot the AI agent can't afford much trial-and-error behaviour, that could cause damage to the robot, and do this over thousands of millions of time steps, for each task separately, in order to build a sufficiently large training dataset. Those machines work in the real world, not accelerated computer simulation, and often need a human assistance. To apply sample efficient model-based systems, that can generalize their knowledge, accurate models are needed.

This work explores progress in the model-learning domain and examine a possibility of using learned models in simulation-based search algorithms. Recently, there have been major steps taken in the domain of model-learning algorithms[2][8][1][6]. More accurate models, at least in short horizon, open the path for application of proofed simulation-based search planning algorithms like TD-Search[11] or AlphaZero[12] to complex planning problems in environments with discrete state-action spaces and without access to a ground-truth model. This should allow for improvements in data efficiency, generalization and agents performance for these problems. This work focuses on two benchmarks: a complex puzzle environment, Sokoban and a multi-task environment, MiniPacman.

2. THEORETICAL BACKGROUND

2.1. Markov Decision Processes

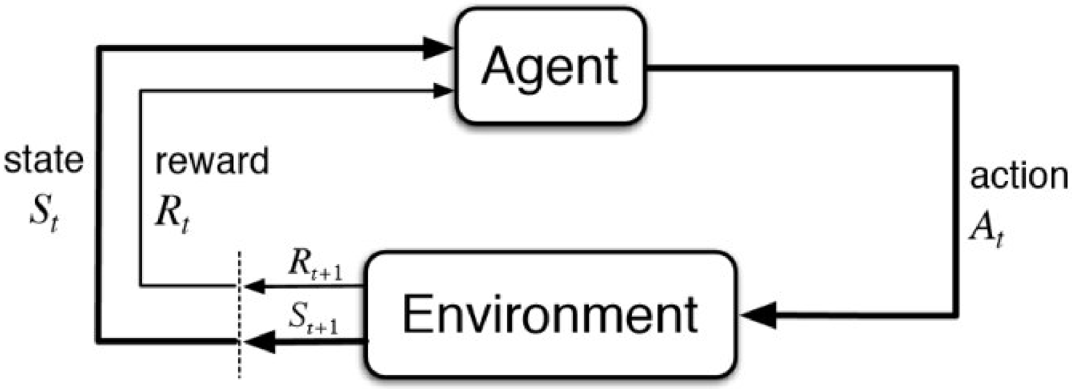
Markov Decision Processes (MDPs) are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations (states) and through those future rewards. Thus MDPs involve delayed reward and the need to tradeoff immediate and delayed reward.

A Markov Decision Process consists of a set of states S and a set of actions A. The dynamics of the MDP, from any state $s \in S$ and for any action $a \in A$, are determined by transition probabilities, $P^a_{ss'} = Pr(s_{t+1} = s'|s_t = s, a_t = a)$, specifying the distribution over the next state $s' \in S$. Finally, a reward function, $R^a_{ss'} = \mathbb{E}[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s']$, specifies the expected reward for a given state transition. Episodic MDPs terminate with probability 1 in a distinguished terminal state, $s_T \in S$, after finite number of transitions. Continuous MDPs doesn't include the terminal state and can run endlessly. The return $R_t = \sum_{k=t}^{T} r_k$ is the total reward accumulated in that episode from time t until reaching the terminal state at time T. In continuous MDPs this is the sum of an infinite sequence of rewards from time t. A policy, $\pi(s, a) = Pr(a_t = a|s_t = s)$, maps a state s to a probability distribution over actions. The value function, $V_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s]$, is the expected return from state s when following policy π. The action value function, $Q_\pi(s, a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a]$, is the expected return after selecting action a in state s and then following policy π. The optimal value function is the unique value function that maximises the value of every state, $V^*(s) = maxV_\pi(s), \forall_{s \in S}$ and $Q^*(s, a) = maxQ_\pi(s, a), \forall_{s \in S, a \in A}$. An optimal policy $\pi^*(s, a)$ is a policy that maximises the action value function from every state in the MDP, $\pi^*(s, a) = \underset{a}{\text{argmax}} Q^*(s, a)$.

MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made. In reinforcement learning the dynamics and the reward function are hidden behind an environment. Consequently, we can't directly use them for planning, but we can learn through interaction with the environment.

2.2. Reinforcement Learning

Reinforcement learning (RL) is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal.[13] This mapping is called a policy π. RL consists of an agent that, in order to learn a good policy, acts in an environment. The environment provides a response to each agent's action a that is interpreted and fed back to the agent. Reward r is used as a reinforcing signal and state s is used to condition agent's decisions. Fig. 21 explains it in the diagram. Each action-response-interpretation sequence is called a step or a transition. Multiple steps form an episode. The episode finishes in a terminal state s_T and the environment is reset in order to start the next episode from scratch. Very often, RL agents need dozens and dozens of episodes to gather enough experience to learn the (near) optimal policy.



Rys. 21. Reinforcement Learning[13]

[Should I describe here General Policy Iteration, Monte-Carlo Control, TD-Learning, ... what else? Rather than guessing what needs to be described, continue work and see what needs more attention.]

2.3. Deep Learning

It's like Machine Learning... but deeper.

Rys. 22. Deep Learning

REFERENCES

[1] Lars Buesing et al. “Learning and Querying Fast Generative Models for Reinforcement Learning”. In: *arXiv e-prints* (2018). arXiv: `1802.03006`.

[2] Silvia Chiappa et al. “Recurrent Environment Simulators”. In: *arXiv e-prints* (2017). arXiv: `1704.02254`.

[3] Richard Evans et al. “De novo structure prediction with deep-learning based scoring”. Dec. 2018. URL: `https://deepmind.com/blog/alphafold/`.

[4] Vladimir Feinberg et al. “Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning”. In: *arXiv e-prints* (2018). arXiv: `1803.00101`.

[5] David Ha and Jürgen Schmidhuber. “World Models”. In: *arXiv e-prints* (2018). arXiv: `1803.10122`.

[6] Danijar Hafner et al. “Learning Latent Dynamics for Planning from Pixels”. In: *arXiv e-prints* (2018). arXiv: `1811.04551`.

[7] Matteo Hessel et al. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. In: *arXiv e-prints* (2017). arXiv: `1710.02298`.

[8] Felix Leibfried, Nate Kushman, and Katja Hofmann. “A Deep Learning Approach for Joint Video Frame and Reward Prediction in Atari Games”. In: *arXiv e-prints* (2016). arXiv: `1611.07078`.

[9] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *arXiv e-prints* (2016). arXiv: `1602.01783`.

[10] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *arXiv e-prints* (2017). arXiv: `1707.06347`.

[11] David Silver, Richard S. Sutton, and Martin Müller. “Temporal-difference search in computer Go”. In: *Machine Learning* 87.2 (2012), pp. 183–219. ISSN: 1573-0565. DOI: `10.1007/s10994-012-5280-0`. URL: `https://doi.org/10.1007/s10994-012-5280-0`.

[12] David Silver et al. “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”. In: *arXiv e-prints* (2017). arXiv: `1712.01815`.

[13] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction 2nd*. The MIT Press, 2018. ISBN: 9780262039246.

[14] Théophane Weber et al. “Imagination-Augmented Agents for Deep Reinforcement Learning”. In: *arXiv e-prints* (2017). arXiv: `1707.06203`.

LIST OF FIGURES

LIST OF TABLES