Exploiting symmetries in Markov Decision Processes for Reinforcement Learning

Gioele Buriani, Giovanni Corvi, Mathijs van Geerenstein, Edoardo Panichi

***Abstract*— In this paper, we present an overview of concepts like Markov Decision Processes (MDPs), homomorphism and symmetry, and how they can be used to accelerate deep reinforcement learning. One novel approach to achieve this is with MDP homomorphic networks. These make use of group symmetries in the joint state-action space of an MDP to find a control policy more efficiently. We first introduce the concepts and place them in context, after which we go into more technical on MDP homomorphic networks. Finally, we provide three explicative examples where structural symmetries of the system lead to more efficient learning for the MDP's policy.**

I. Introduction

The essence of deep learning is built from two simple algorithmic principles: first, the notion of feature learning, whereby adapted, often hierarchical, features capture the appropriate notion of regularity for each task, and second, learning by local gradient-descent, typically implemented as *back-propagation* [1].

A variant of deep learning is called *Geometric Deep Learning*, which works with non-euclidean data or systems with built-in symmetries.

This paper summarises the fundamental concepts of Markov Decision Processes, Reinforcement Learning, Homomorphism, and Geometric Deep Learning, often combined together to efficiently learn a policy $\pi(s)$ for MDP with symmetries.

MDP homomorphic networks are unique types of NNs that allow combining together these concepts. Thanks to them, the learning process is enhanced and allows us to solve a common problem in reinforcement learning: the curse of dimensionality.

In the end, we illustrate that such networks converge faster than unstructured baselines on CartPole, a grid world and Pong.

II. Context

A. Markov Decision Processes

Markov Decision Processes (MDPs) are widely used to model discrete-time stochastic decision making problems [2]. In MDPs, the outcome of the decision making process is partly random, and partly under control of a decision maker, or agent. At each time step, the agent can take any action a that is available from the current state s. With this action, the state s stochastically transitions to a new state s' with a probability determined by action a. For an example, see Figure 1. The probability is described by a state transition function $P_a(s, s')$. With every state transition comes a corresponding reward given by a reward function $R_a(s, s')$.

MDPs can scale quite poorly with the size of the problem that they model [3]. Model minimization methods try to address this issue. These reduce the size of the MDP model by exploiting redundancy in the problem definition. Ravindran et al. [3] pose a new method for minimizing models in MDPs: making use of symmetries.

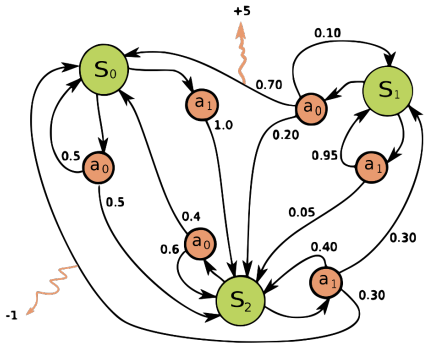

Fig. 1: MDP model with states (green), actions (orange), stochastic state transitions (black) and rewards

B. Reinforcement learning

In Reinforcement Learning (RL), the environment provides a reward and a new state based on the action taken by an agent. There is no ground truth available for learning, but the model can learn through rewards. This description is similar to that of an MDP, to no suprise: MDPs are widely used to mathematically formulate RL problems, especially when the transition probabilities and reward functions are (partly) unknown [4].

C. Unsupervised learning

In unsupervised learning, the model does not learn with labelled data either, but rather through self-organization of patterns in data [5]. Common applications of unsupervised learning include clustering, dimensionality reduction and pattern search.

D. Geometric learning

One way to make use of relationships in data, like symmetries, is with Geometric Deep Learning (GDL) [6]. GDL is a field within deep learning that aims to build neural networks that can learn from non-euclidean data. Graphs are an example of non-euclidean data. These allow for representation of relationships and structure between features, next to the features themselves. A neural network that embeds the structure of a graph - *a graph network* - allows for a strong relational inductive bias [7]. This guides the model towards learning about relations between objects, which is suggested to be essential for human-like intelligence. GDL hereby allows to take advantage of data with inherent relationships, connections and shared properties.

Both reinforcement- and unsupervised learning can be used with geometric learning, even though they have their differences. Where RL uses an agent to learn through feedback by interacting with the environment, unsupervised learning tries to find associations between data entries. RL learns by trail-and-error and unsupervised learning uses patterns in data to build a compact representation without feedback. For example: Bellemare et al. [8] use geometric reinforcement learning to minimize an approximation of the value function for a given environment. Chen et al. [9] represent point clouds on graphs to exploit invariant representations at multiple hierarchies in an geometric unsupervised learning problem.

E. Homomorphism

So, in reinforcement learning, we want to find a policy $\pi(s)$ that maximizes the reward for all possible actions a available from state s. With geometric learning, we can exploit the structure of a relationship within data to accelerate this learning. RL can make use of MDPs to mathematically describe the problem. A group homomorphism is a function that maps a group operation of one group to one of another group. Van der Pol et al. [10] proposes MPD homomorphic networks to combine these techniques. Specifically, it makes use of symmetries in the joint state-action space of an MDP to reduce the size of the solution space, which results in more efficient learning.

III. Background

In order to clearly understand the next section, a brief outline of mathematical concepts is here presented.

Symmetry: Given a transformation operation $L_g : X \to X$, where $g \in G$ is a mathematical group, and a mapping $f : X \to Y$, we say that f is symmetric to L_g if

$$f(x) = f(L_g[x]) \quad \forall\, g \in G, x \in X.$$

An example of this can be seen in Figure 2, in this case the transformation L_g is a reflection symmetry along the vertical axis.

Orbit: Given a point $x \in X$, the orbit O_x is the set of points reachable from x via a transformation operator L_g.

$$O_x = \{L_g[x] \in X \mid g \in G\}.$$

Equivariance: Given a mapping $f : X \to Y$ we say that f is symmetric to a group of transformations $L_g : X \to X$, where $g \in G$, if we can associate every transformation L_g with a second one $K_g : Y \to Y$ acting on the output space of f, such that [11]

$$K_g[f(x)] = f(L_g[x]) \quad \forall\, g \in G, x \in X.$$

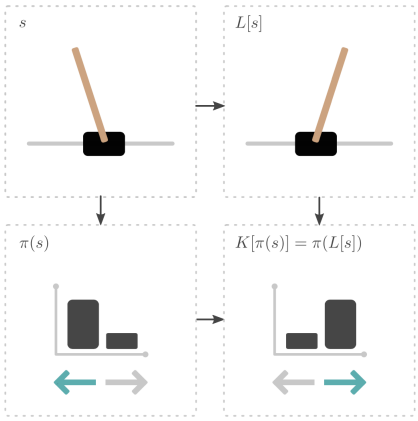

Fig. 2: An example of state-action space symmetry

MDP with symmetries: An MDP (S, A, R, T, γ) can be defined as an MDP with symmetries if there exists a set of state transformations, as $L_g : S \to S$, and a state-dependent action transformation, as $K_g^s : A \to A$,

such that

$$R(s,a) = R(L_g[s], K_g^s[a]),$$
$$T(s'|s,a) = T(L_g[s']|L_g[s], K_g^s[a]),$$
$$\forall\, g \in G, s \in S, a \in A,$$

meaning that the reward and transition function are invariant along the orbits defined by L_g and K_g^s.

IV. Efficient Learning

Generally speaking, the complexity of a system can be quantified through the number of parameters needed to describe and model it. The "curse of dimensionality" problem heavily affects reinforcement learning in the case of large state-action spaces, therefore choosing a convenient representation often makes all the difference allowing not only feasible, but efficient learning. Exploiting symmetries of the system is a way of selecting a convenient representation.

When we work with MDPs, a classic machine learning problem is learning an optimal policy to achieve the highest reward.

To better understand why symmetries help to find more efficient techniques of learning a control policy, let's analyse a simple example. The scenario taken into consideration is an inverted pendulum. For such a system it is easy to observe that it has reflection symmetry along the vertical axis. This means that the system can be divided into two symmetric halves. As a result, balancing an inverted pendulum that falls to the right requires an *equivalent*, but mirrored, strategy to one that falls to the left.

If the system in question presents symmetries, exploiting them results in a simplified learning task allowing to transfer observations made in one particular situation to others that are symmetric to it. This doubles the amount of information learned in every single iteration and update, leading to a significant reduction in the number of training examples needed for the algorithm to converge to the optimal policy.

We use the notion of MDP homomorphisms [3] [12], a map from an MDP to another one while maintaining the essential structure, to formalize these symmetries. This concept, paired with *MDP with symmetries* leads to a smaller abstract MDP where equivalent situations in the original MDP get represented by a single state. In fact, the abstract MDP removes redundancies in the problem description, obtaining a smaller state-action space, upon which we may more easily build a policy.

Considering again the inverted pendulum, the state $(x, \dot{x})$ of the abstract MDP represents both $(\theta, \dot{\theta})$ and $(-\theta, -\dot{\theta})$ in the *original state-space*.

Thanks to the concepts introduced, the process requires learning the policy only in the abstract MDP, then pull it back to the original MDP through a procedure called *lifting* (mathematical details in [10]). Basically, *lifting* the learned policy is a mechanism that takes advantage of the concept of *equivariance*. Hence, it applies the transformation K on $\pi(s)$, inferring so a valid policy for the original MDP.

In practice, the learning happens through an *MDP homomorphic network* [10], i.e. a neural network that is equivariant under symmetries in the joint state-space of an MDP. The introduction of these networks formalizes the connection between equivariant networks and symmetries in reinforcement learning. *Equivariant neural networks* [13] are a class of NNs, which have built-in symmetries, leading to effective parameter sharing. Convolutional Neural Networks (CNNs) are a common example, mostly used in image recognition and processing. CNNs are equivariant to translations but not to other transformations.

For completeness, it is worth mentioning that Van der Pol et al. [10] did not just draw a connection between MDP homomorphisms and group equivariant networks, proposing MDP homomorphic networks. They also introduced a novel numerical algorithm for the automated construction of equivariant layers.

V. Applications

After explaining why the exploitation of symmetries can prove beneficial in learning a control policy, it can now be interesting to analyse some examples of its application. As proposed by the reference paper [10], the three analysed cases will be those of Pong, CartPole and Grid world.

A. Pong

Pong is an Atari video game developed in 1972. It features a very simplified table-tennis game where the player simply has to move a paddle up and down in order to hit the ball and send it to the other side of the screen.

Due to the extremely simple gaming mechanics, it was fairly easy to identify a flip symmetry in the game over the horizontal axis (Figure 3). In fact, "if we flip the observations, the up and down actions also flip" [10]. This leads to Pong having duplicate (up, down) actions that can therefore allow to "mask out the policy

values for the second pair of (up, down) actions" [10]. This greatly simplifies the learning process for this application since it has to be performed only on half of the possible observation-action pairs.

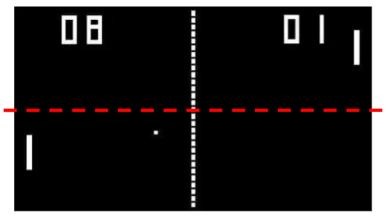

Fig. 3: Symmetry in the Pong game

Interestingly enough, the process to analyse this application, consisting of reducing the resolution and finding the symmetry, is highly resembling how this application was created. In fact, as stated before, Pong was created in 1972 and at that time the microprocessors were not powerful enough to generate complex graphics. For this reason, most Atari games were black and white, with low resolution and, surprisingly, presenting an axial symmetry. In the case of Pong the symmetry is vertical, but in many other examples of Atari games created in the same period "the graphics [...] were drawn only on the left side of the screen and then duplicated on the right half of the screen, either as a repetition or a mirror image of the left side, accounting for much of the horizontal symmetry found in the early Atari games" [14]. This initially necessary condition, caused by the lack of power of the early processors, now gives us a high geometric computational advantage in trying to learn how to automatically play many of these games.

B. CartPole

CartPole is an example of a pole balancing task, where an inverted pendulum has to be balanced on a moving base. This is a common control basic problem, and in this case, it is quite immediate to identify a symmetry across the vertical direction such that "we used a two-element group of reflections about the y-axis" [10] (Figure 4).

In fact, in this example "The agent's state here is given by $(\theta, x, \dot{\theta}, \dot{x})$", while "The action space is the move set $\{Left, Right\}$ for the cart" [15]. By exploiting the aforementioned symmetry it is possible to state that "the state action pairs $\langle(\theta, x, \dot{\theta}, \dot{x}), Left\rangle$ and $\langle(-\theta, -x, -\dot{\theta}, -\dot{x}), Right\rangle$ form an equivalence class under the symmetry exhibited by the domain" [15].

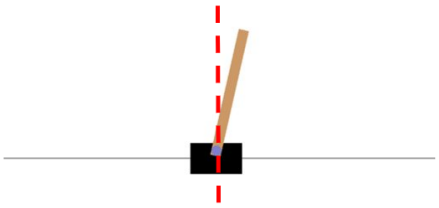

Fig. 4: Symmetry in the CartPole application

C. Grid world

Lastly, also Grid world is an application where symmetries can be exploited to facilitate the learning procedure. In particular, the reference paper [10] considers a "toroidal 7-by-7 predator-prey grid world with agent-centred coordinates [...] chosen due to its four-fold rotational symmetry" [10] (Figure 4).

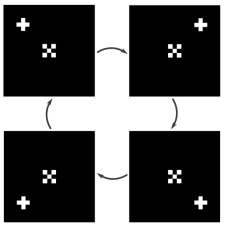

Fig. 5: Symmetry in the Grid world application

In fact, differently from the two previous cases, Grid worlds have a much higher number of possible symmetries, in particular "for the simple case of grid world of dimension d there exist $O(d!2^d)$ fold symmetries" [15]. Notice that in the analysed case the number of symmetries falls down to four, instead of eight as it should be for a 2D grid, because of the presence of the prey that makes the environment less symmetric.

VI. Conclusion

We showed how the exploitation of symmetries in geometric deep reinforcement learning can result in significant advantages, most of all the reduction in the iteration needed for training. In other words, with some geometric expedients, we can reach more efficient algorithms that converge faster. This has been shown in very simple applications such as Pong, CartPole and Grid world, but the possible advantages of this technique can have multiple applications. The increase in efficiency, for example, can be used to improve real-time reinforcement learning applications. Moreover, it could allow using cheaper hardware to perform the computations, resulting in more affordable, and therefore spreadable, technologies.

REFERENCES

[1] Michael M. Bronstein et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *CoRR* (2021).

[2] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[3] Balaraman Ravindran and Andrew G Barto. *Symmetries and model minimization in markov decision processes*. 2001.

[4] Yoav Shoham, Rob Powers, and Trond Grenager. *Multi-agent reinforcement learning: a critical survey*. Tech. rep. Technical report, Stanford University, 2003.

[5] Geoffrey Hinton and Terrence J Sejnowski. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[6] Michael M Bronstein et al. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[7] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[8] Marc Bellemare et al. "A geometric perspective on optimal representations for reinforcement learning". In: *Advances in neural information processing systems* 32 (2019).

[9] Haolan Chen et al. "Unsupervised Learning of Geometric Sampling Invariant Representations for 3D Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 893–903.

[10] Elise van der Pol et al. "MDP homomorphic networks: Group symmetries in reinforcement learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4199–4210.

[11] Daniel E. Worrall et al. "Harmonic Networks: Deep Translation and Rotation Equivariance". In: *CoRR* (2016).

[12] Balaraman Ravindran and Andrew G Barto. *Approximate Homomorphisms: A framework for non-exact minimization in Markov Decision Processes*. 2004.

[13] Carlos Esteves. "Theoretical Aspects of Group Equivariant Neural Networks". In: *CoRR* (2020).

[14] Mark JP Wolf. "Abstraction in the video game". In: *The video game theory reader*. Routledge, 2013, pp. 69–88.

[15] Anuj Mahajan and Theja Tulabandhula. "Symmetry learning for function approximation in reinforcement learning". In: *arXiv preprint arXiv:1706.02999* (2017).