

# Expert-guided Symmetry Detection in Markov Decision Processes

Giorgio Angelotti,<sup>1,2</sup> Nicolas Drougard<sup>1,2</sup><sup>a</sup> and Caroline P. C. Chanel<sup>1,2</sup><sup>b</sup>

<sup>1</sup>ISAE-SUPAERO, University of Toulouse, France

<sup>2</sup>ANITI, University of Toulouse, France

**Keywords:** Offline Reinforcement Learning, Batch Reinforcement Learning, Markov Decision Processes, Symmetry Detection, Homomorphism, Density Estimation, Data Augmenting.

**Abstract:** Learning a Markov Decision Process (MDP) from a fixed batch of trajectories is a non-trivial task whose outcome's quality depends on both the amount and the diversity of the sampled regions of the state-action space. Yet, many MDPs are endowed with invariant reward and transition functions with respect to some transformations of the current state and action. Being able to detect and exploit these structures could benefit not only the learning of the MDP but also the computation of its subsequent optimal control policy. In this work we propose a paradigm, based on Density Estimation methods, that aims to detect the presence of some already supposed transformations of the state-action space for which the MDP dynamics is invariant. We tested the proposed approach in a discrete toroidal grid environment and in two notorious environments of OpenAI's Gym Learning Suite. The results demonstrate that the model distributional shift is reduced when the dataset is augmented with the data obtained by using the detected symmetries, allowing for a more thorough and data-efficient learning of the transition functions.

## 1 INTRODUCTION

Model-based Offline Reinforcement Learning (ORL) is the branch of Machine Learning that first fits a dynamical model and then obtains a behavioural (optimal) system control policy with the aim of maximizing a predetermined utility function (Sutton, 1990). The model learning phase is usually based on a finite batch of trajectories followed by the system. Although it is always possible to compute confidence intervals on the parameters of the learnt categorical distribution, it is hard to determine the amount of data needed for the model optimisation to return a sufficiently improved policy.

Following (Levine et al., 2020) the statistical distance between reality and the learnt model is defined as *distributional shift*. Thus, in order to be able to compute reliable strategies that can be applied to real world scenarios (e.g. planning in healthcare, autonomous driving) the distributional shift must be minimal. Consequently, limiting it by considering strategies close to the one used for data collection, is the main challenge of ORL (Levine et al., 2020). Indeed, such strategies are more likely to drive the sys-


tem into areas of the state-action space whose transitions have been thoroughly explored in the batch, allowing a better estimation of the model than in other areas.


Interestingly, the detection of symmetries in physics has always provided greater representability and generalization power to the learnt models (Gross, 1996). Moreover, symmetry detection can potentially provide knowledge about what will be the time evolution of a system state that has never been explored before. Such an idea can be explored to allow a more data efficient model learning phase for ORL taking advantage of the knowledge of regularities in the system dynamics that repeat themselves in different situations, independently of the initial conditions.

In this context, a method to detect symmetries for ORL relying on expert guidance is presented. The described method is built in the classical framework of discrete time Markov Decision Processes (MDPs) for both discrete and continuous state-action spaces (Mausam and Kolobov, 2012).

### 1.1 Illustrative Example

One of the most emblematic examples is checking whether or not a dynamical system is symmetric with

<sup>a</sup> <https://orcid.org/0000-0003-0002-9973>

<sup>b</sup> <https://orcid.org/0000-0003-3578-4186>

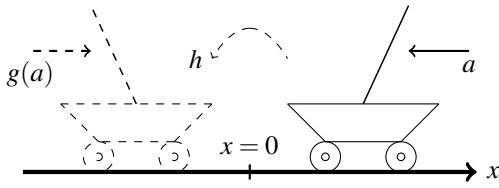


Figure 1: The cart in the right is a representation of a Cart-Pole’s state  $s_t$  with  $x_t > 0$  and action  $a_t = \leftarrow$ . The dashed cart in the left is the image of  $(s_t, a_t)$  under the transformation  $h$  which inverts state  $f(s) = -s$  and action  $g(a) = -a$ .

respect to some particular transformation of the system of reference. For instance, consider the well-known CartPole Reinforcement Learning (RL) domain of the OpenAI’s Gym Learning Suite (Brockman et al., 2016). In CartPole the purpose of the automated agent is to avoid a rotating pole situated on a sliding cart falling down due to gravity. The state of the system is expressed as a tuple  $s = (x, v, \alpha, \omega)$  where  $x$  is the position of the cart with respect to a horizontal track upon which it can slide,  $v$  is its longitudinal velocity,  $\alpha$  is the angle between the rotating pole and the axis pointing along the direction of the gravitational acceleration, and  $\omega$  the angular velocity of the pole. The agent can push the cart left ( $\leftarrow$ ) or right ( $\rightarrow$ ) at every time step (in the negative or positive direction of the track) providing to the system a fixed momentum  $|p|$ . A pictorial representation of a state-action pair  $(s_t, a_t)$  can be found in Figure 1.

Let us suppose there exists a function  $h : S \times A \rightarrow S \times A$  that maps a state-action pair  $(s_t, a_t)$  to  $(f(s_t), g(a_t))$ , where  $f : S \rightarrow S$  and  $g : A \rightarrow A$ , such that the dynamics of the pair  $(s, a)$  is the same as the one of  $h(s, a)$ . Note that in this example, the system dynamics is symmetric with respect to a flip around the vertical axis. In other words, its dynamics is invariant by multiplication by minus one, assuming that if  $a = \leftarrow$  then  $g(a) = -a = \rightarrow$  and vice versa. Indeed, if the state-action pair  $(s_t, a_t)$  leads to the state  $s_{t+1}$ , this property will imply that  $h(s_t, a_t) = (-s_t, -a_t)$  leads to  $f(s_{t+1}) = -s_{t+1}$ .

When learning the dynamics from a finite batch of experiences (or trajectories), resulting in a set of transitions  $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i=1}^n$  with  $n \in \mathbb{N}$  the size of the batch, we might for instance fit a function  $\hat{s}(s, a) = s'$  with the aim of minimizing a loss, e.g. the Mean Squared Error. However, imagine that in the batch  $\mathcal{D}$  there were many transitions regarding the part of the state-action space with  $x > 0$  and very few with  $x < 0$ . Unfortunately, we may learn a good model to forecast what will happen when the cart is at the right of the origin and a very poor model at its left side. We can then suppose that also the optimal control policy will perform well when  $x > 0$  and poorly when  $x < 0$ .

Nevertheless, if it were possible to be confident of the existence of the symmetry  $f(s) = -s$  and  $g(a) = -a$  (where the opposite of the action  $a$  is the transformation stated above), we might extend the batch of experiences without additional interaction with the system, and then improve the accuracy of the model also to the regions where  $x < 0$ .

## 1.2 Definition of the Problem and Related Work

How can we automatically detect these sorts of symmetries in the batch of experiences?

In this work we propose a paradigm to check whether a pre-alleged structure of this kind is present in the batch and consequently utilize it to augment the data.

Data augmentation is a common practice in Machine Learning to improve performances in data-limited regimes. For example, it has been used in recent Computer Vision DNN architectures (Shorten and Khoshgoftaar, 2019) where, if the purpose is to detect an object in an image independently of its position or orientation the original data set gets augmented with several copies of the images providing the object translated or rotated.

We build our approach upon the Markov Decision Process (MDP) framework (Mausam and Kolobov, 2012), whether discrete or continuous state and action spaces can be considered, although hypothesizing a non stochastic dynamics.

The detection of structures and abstractions in MDPs has been widely studied in the literature. In (Dean and Givan, 1997; Givan et al., 2003) the notions of MDP homomorphism (structure-preserving maps between the original MDP and one characterized by a factored representation) and stochastic bisimulation (are introduced and used to automatically partition the state space of an MDP and to find aggregated and factored representations. In (Ravindran and Barto, 2001) the previous works on state abstractions have been extended to include the concept of symmetry, and in (Ravindran and Barto, 2004) approximate homomorphisms are considered.

Later on, (Narayanamurthy and Ravindran, 2008) showed that the fully automatic discovery of symmetries in a discrete MDP is as hard as verifying whether two graphs are isomorphic. Concurrently, (Taylor et al., 2009) relaxed the notion of bisimulation to allow for the attainment of performance bounds for approximate MDP homomorphisms. Approximate homomorphisms are of particular interest in continuous state MDPs where a hard mapping to an aggregated representation could be impractical. In this context,

(Ferns et al., 2004) developed a bisimulation pseudometric to extend the concept of bisimulation relation. The automatic discovery of representations using the bisimulation pseudometric has been investigated in recent years using Deep Neural Networks and obtaining theoretical guarantees for such a methodology (Ruan et al., 2015; Castro, 2020; Abel et al., 2020).

From a more theoretical perspective (Li et al., 2006) classified different kinds of possible state abstractions, showing for each one of them which function would be preserved under the new refinement: the value function, the Q-value function, the optimal policy, and so on. From a different perspective (Mandel et al., 2016) developed an algorithm that aims to cluster MDPs states in a Bayesian sense in order to solve the MDP in a more data efficient way, even when an underlying homomorphic or symmetric structure is not present.

Recently, (van der Pol et al., 2020a) used a contrastive loss function that enforces action equivariance on a to be learnt representation of an MDP. Their approach resulted in the automatic learning of a structured latent space which can be used to plan in a more data efficient fashion. Finally, (van der Pol et al., 2020b) introduced MDP Invariant Networks, a specific class of Deep Neural Network (DNN) architectures that guarantees by construction that the optimal MDP control policy obtained through other Deep RL approaches will be invariant under some set of symmetric transformations and hence providing more sample efficiency to the baseline when the symmetry is actually present.

### 1.3 Contribution

In summary, the literature can be divided in two categories: 1) learning a representation using only or the data contained in the batch or some a priori knowledge about the environment; 2) exploiting an alleged symmetry to obtain a behavioural policy that converges faster to the optimal one.

The present work gets inspiration from all the reported literature, but rather than automatically looking for a state space abstraction or directly exploiting a symmetry to act in the world it focuses on checking whether the dynamical model to be learnt is symmetric with respect to some transformation that is presumed by other means. In other words, such transformation can be given by external expert knowledge or proposed by the user insight and its existence is verified by estimating the chance of its occurrence with machine learning tools.

In order to verify the presence of a symmetry, a preliminary estimate of the transition model is needed. More specifically, we first perform a probability mass function (pmf) estimation or a probability density function (pdf) estimation of the transitions in the batch  $\mathcal{D}$  depending on the typology of the MDP we are tackling (respectively discrete or continuous). In the discrete case this amounts to learning a set of categorical distributions. In the continuous case we opt for using Normalizing Flows (Dinh et al., 2015; Kobyzev et al., 2020), a Deep neural network architecture that allows to adapt and estimate a pdf while being always able to exactly compute the density value for new samples. In this way, once a pmf/pdf has been estimated from the batch of transitions  $\mathcal{D}$  we can compute the probability of an alleged symmetric transition that is supposed to be sampled from the same distribution.

When the probability (or the density in the continuous case) is bigger than a given threshold for a high fraction of samples, we decide to *trust* in the presence of this alleged symmetry and hence augment the batch by including the symmetric transitions.

In the end the dynamics of the model is learnt over the augmented data set. When the approach detects a symmetry that is really present in the true environment then the accuracy of the learnt model increases, otherwise the procedure could also result in detrimental effects.

The paper is organized as follows: in Section 2 we provide the definitions of MDP, MDP homomorphism, symmetric transformation and pmf/pdf estimation; next, in Section 3 the two algorithms for expert-guided detection of symmetries are presented along with a discussion of the limitations of both (these are the main contributions); results in proof-of-concept environments are shown in Section 4; conclusion and future perspective are illustrated in Section 5.

## 2 BACKGROUND

In this section, we introduce the basic concepts about MDPs, homomorphisms and symmetries.

**Definition 2.1** (Markov Decision Process). A discrete-time MDP (Bellman, 1966) is a tuple  $\mathcal{M} = (S, A, R, T, \gamma)$ , where  $S$  is the set of states,  $A$  is the set of actions,  $R : S \times A \rightarrow \mathbb{R}$  is the reward function,  $T : S \times A \rightarrow \text{Dist}(S)$  is the transition pdf, and  $\gamma \in [0, 1)$  is the discount factor. At each time step, the agent observes a state  $s = s_t \in S$ , takes an action  $a = a_t \in A$  drawn from a policy  $\pi : S \times A \rightarrow [0, 1]$ , and with probability  $T(s'|s, a)$  transits to a next state  $s' = s_{t+1}$ , earn-

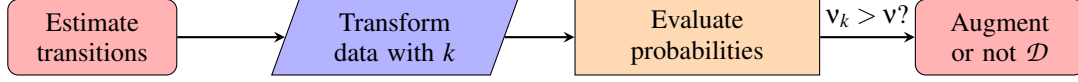


Figure 2: Pseudo flow chart of Algorithms 1 and 2.

ing a reward  $R(s_t, a_t)$ . The value function of the policy  $\pi$  is defined as:  $V_\pi(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right]$ . The optimal value function  $V^*$  is the maximum of the previous expression over every possible policy  $\pi$ . When  $\forall (s, a) \in S \times A$ ,  $T$  dictates the transition to one and only state  $s' \in S$  the MDP is said to be deterministic or non stochastic.

A homomorphism from a dynamic system  $\mathcal{M}$  to a dynamic system  $\mathcal{M}'$  is a mapping that preserves  $\mathcal{M}$ 's dynamics, while in general eliminating some of the details of the full system  $\mathcal{M}$ .

**Definition 2.2** (MDP Homomorphism). An MDP homomorphism  $h$  (Ravindran and Barto, 2004) from an MDP  $\mathcal{M} = (S, A, T, R, \gamma)$  to an MDP  $\mathcal{M}' = (S', A', T', R', \gamma')$  is a surjection from  $S \times A$  to  $S' \times A'$ , defined by a tuple of surjections  $(f, g)$ , with  $h(s, a) = (f(s), g(a))$ , where  $f: S \rightarrow S'$  and  $g: A \rightarrow A'$  such that  $\forall (s, s') \in S^2, a \in A$ :

$$T'(f(s), g(a), f(s')) = \sum_{s'' \in [s']_f} T(s, a, s''), \quad (1)$$

$$R'(f(s), g(a)) = R(s, a). \quad (2)$$

where  $[s']_f = f^{-1}(\{f(s')\})$ , i.e.  $[s']_f$  is the set of states for which the application of  $f$  results in the state  $f(s') \in S'$ .

**Definition 2.3** (MDP Symmetry). Let  $k$  be a surjection from an MDP  $\mathcal{M}$  to itself defined by the tuple of surjections  $(f, g, l)$  where  $f: S \rightarrow S$ ,  $g: A \rightarrow A$  and  $l: S \rightarrow S$ . The transformation  $k$  is a symmetry if  $\forall (s, s') \in S^2, a \in A$  both the dynamics  $T$  and the reward  $R$  are invariant with respect to the transformation induced by  $k$ :

$$T(f(s), g(a), l(s')) = T(s, a, s'), \quad (3)$$

$$R(f(s), g(a)) = R(s, a). \quad (4)$$

If  $\mathcal{M}' = \mathcal{M}$  and  $l = f$  then an MDP symmetry is also an MDP homomorphism.

In this work we will focus only on symmetries with respect to the dynamics and not the reward. In particular, we will check for transformations  $k$  that map a whole transition  $k(s, a, s') \rightarrow (f(s), g(a), l(s'))$ . Therefore the only condition that will be examined and fulfilled is Equation 3. Detecting the said structure will help reduce the distributional shift between the true environment dynamics and the one learnt from experience.

**Probability Mass Function Estimation for Discrete MDPs.** Let  $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i=1}^n$  be a batch of recorded transitions. Performing mass estimation over  $\mathcal{D}$  amounts to compute the probabilities that define the unknown categorical transition distribution  $T$  by estimating the frequencies of transition in  $\mathcal{D}$ . In other words we compute:

$$\hat{T}(s, a, s') = \begin{cases} \frac{n_{s,a,s'}}{\sum_{s'} n_{s,a,s'}} & \text{if } \sum_{s'} n_{s,a,s'} > 0, \\ |S|^{-1} & \text{otherwise;} \end{cases} \quad (5)$$

where  $n_{s,a,s'}$  is the number of times the transition  $(s_t = s, a_t = a, s_{t+1} = s') \in \mathcal{D}$ .

**Probability Density Function Estimation for Continuous MDPs.** Performing density estimation over  $\mathcal{D}$  amounts to finding an analytical expression for the probability density of a transition  $(s, a, s')$  given  $\mathcal{D}$ :  $\mathcal{L}(s, a, s' | \mathcal{D})$ . Normalizing flows (Dinh et al., 2015; Kobyzev et al., 2020) allow defining a parametric flow of continuous transformations that reshapes a known pdf (e.g. a multivariate gaussian) to one that best fits the data. Since the transformations are known, the Jacobians are computable at every step and the probability value can always be assessed. (Dinh et al., 2015).

### 3 EXPERT-GUIDED DETECTION OF SYMMETRIES

We propose a paradigm to check whether the dynamics of a to-be-learned (i) deterministic discrete MDP (see Algorithm 1) or (ii) a deterministic continuous MDP (see Algorithm 2) is endowed with the invariance of the dynamics with respect to some transformation. A pseudo representation of the flow chart of both algorithms is shown in Figure 2.

Let  $\mathcal{M}$  be a deterministic MDP, let  $\mathcal{D}$  be a batch of pre-collected transitions and let  $k$  be an alleged symmetric transformation of  $\mathcal{M}$ 's dynamics.

In order to check whether  $k$  can be considered or not as a symmetry of the dynamics, in the discrete case we will first estimate the most likely set of transition categorical distributions  $\hat{T}$  given the batch of transitions (Line 1, Algorithm 1) while in the continuous case we will estimate the density of transition in the batch obtaining the probability density  $\mathcal{L}(s, a, s' | \mathcal{D})$  (Line 1, Algorithm 2). In the continuous

---

Algorithm 1: Symmetry detection and data augmenting in a deterministic discrete MDP.

---

**Input:** Batch of transitions  $\mathcal{D}$ ,  $v \in (0, 1)$   
percentage threshold,  $k$  alleged  
symmetry  
**Output:** Possibly augmented batch  $\mathcal{D} \cup \mathcal{D}_k$

- 1  $\hat{T} \leftarrow$  Most Likely Categorical pmf from ( $\mathcal{D}$ )
- 2  $\mathcal{D}_k = k(\mathcal{D})$
- 3  $v_k = \frac{1}{|\mathcal{D}_k|} \sum_{(s,a,s') \in \mathcal{D}_k} \mathbb{1}_{\{\hat{T}(s'|s,a)=1\}}$
- 4 **if**  $v_k > v$  **then**
- 5 | **return**  $\mathcal{D} \cup \mathcal{D}_k$
- 6 **else**
- 7 | **return**  $\mathcal{D}$
- 8 **end**

---

case we will then compute the density value of every transition  $(s, a, s') \in \mathcal{D}$ , resulting in a set of real values from  $\mathcal{L}$  denoted by  $\Lambda$  (Line 2, Algorithm 2). We will select the  $q$ -order quantile of  $\Lambda$  to be a threshold  $\theta \in \mathbb{R}$  (Line 3, Algorithm 2) that will determine whether we can trust the symmetry to be present, and hence to augment or not the starting batch. Next, we will map any  $(s, a, s') \in \mathcal{D}$  to its alleged symmetric image  $(f(s), g(a), l(s'))$  (Line 4, Algorithm 2). The map of  $\mathcal{D}$  under the transformation  $k$  will be denoted as  $\mathcal{D}_k$ . Let then,  $\forall (s, a, s') \in \mathcal{D}_k$ ,  $\mathcal{L}(s, a, s' | \mathcal{D})$  be the probability density of the symmetric image of a transition in the batch. We will assume that the system dynamics is invariant under the transformation  $k$  if  $v_k$ , the percentage of transitions whose density is greater than  $\theta$ , is bigger than the percentage threshold  $v$  (Lines 5-10, Algorithm 2). In the discrete case we will just count how many images of the symmetric transformation  $\mathcal{D}_k$  are present in the batch (Line 3, Algorithm 1) and then decide whether to perform data augmenting according to both  $v_k$  and the threshold  $v$  (Lines 4-8, Algorithm 1). If data augmenting is performed, the boosted batch  $\mathcal{D} \cup \mathcal{D}_k$  will be returned as output. Figure 3 provides an intuition about Algorithm 2.

### 3.1 Intuition for the Discrete Case

In the discrete case we are performing a sort of “proof by induction“. If the majority of symmetric transitions (with respect to the threshold  $v$ ) were already sampled, then we can imagine that the other ones were not sampled just because our data set is not big enough. Therefore we trust that the dynamics is symmetric with respect to  $k$  and augment the batch.

---

Algorithm 2: Symmetry detection and data augmenting in a deterministic continuous MDP.

---

**Input:** Batch of transitions  $\mathcal{D}$ ,  $q \in [0, 1)$   
order of the quantile,  $v \in (0, 1)$   
percentage threshold,  $k$  alleged  
symmetry  
**Output:** Possibly augmented batch  $\mathcal{D} \cup \mathcal{D}_k$

- 1  $\mathcal{L} \leftarrow$  Density Estimate ( $\mathcal{D}$ )
- 2  $\Lambda \leftarrow$  Distribution  $\mathcal{L}(\mathcal{D})$  ( $\mathcal{L}$  evaluated over  $\mathcal{D}$ )
- 3  $\theta = q$ -order quantile of  $\Lambda$
- 4  $\mathcal{D}_k = k(\mathcal{D})$
- 5  $v_k = \frac{1}{|\mathcal{D}_k|} \sum_{(s,a,s') \in \mathcal{D}_k} \mathbb{1}_{\{\mathcal{L}(s,a,s'|\mathcal{D}) > \theta\}}$
- 6 **if**  $v_k > v$  **then**
- 7 | **return**  $\mathcal{D} \cup \mathcal{D}_k$
- 8 **else**
- 9 | **return**  $\mathcal{D}$
- 10 **end**

---

### 3.2 Intuition for the Continuous Case

The intuition behind the approach in the continuous case is that if in the original batch  $\mathcal{D}$  the density of transitions that are “not so different” from some of the symmetric images  $\mathcal{D}_k$  of  $\mathcal{D}$ , then  $\mathcal{L}(\mathcal{D}_k | \mathcal{D})$  will not be “too small”. How small is small when we are considering real valued, continuous pdf? In order to insert a comparable scale we take the threshold  $\theta$  to be a  $q$ -quantile of the set of the estimated density values of the transitions in the original batch  $\mathcal{D}$ , *i.e.*

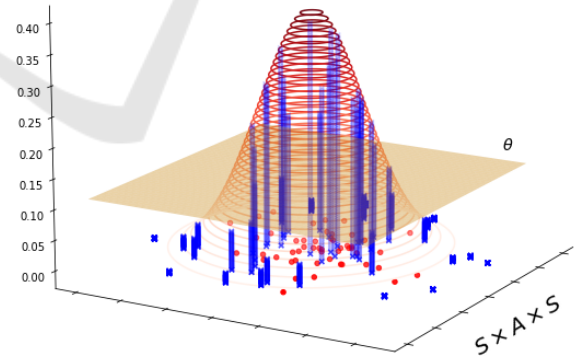


Figure 3: Intuition for the continuous case. The  $xy$  plane is the space of transitions  $S \times A \times S$ , the  $z$  axis is  $\mathcal{L}$ , the value of the probability density of a given transition. The red points represent  $\mathcal{D}$ , the blue crosses  $\mathcal{D}_k$  for a given transformation  $k$ . We display as a red contour plot the pdf  $\mathcal{L}$  learnt in Line 1 of Algorithm 2. The orange hyperplane has height  $\theta$  which is the threshold computed in Line 3 of Algorithm 2. The blue vertical bars have as height the value of  $\mathcal{L}$  evaluated for that specific transition. The algorithm counts the fraction  $v_k$  of samples (blue crosses) which have a vertical bar higher than the hyperplane.

$\{\mathcal{L}(s, a, s' \mid \mathcal{D}) \mid (s, a, s') \in \mathcal{D}\}$ . It goes without saying that since the purpose of Algorithm 2 is to perform data augmentation, it is necessary to select a small  $q$ -order quantile, otherwise the procedure would bear no meaning: it would be pointless to augment the batch with transitions that are already very likely in the original one (see Figure 4). In this case we won't insert any new information.

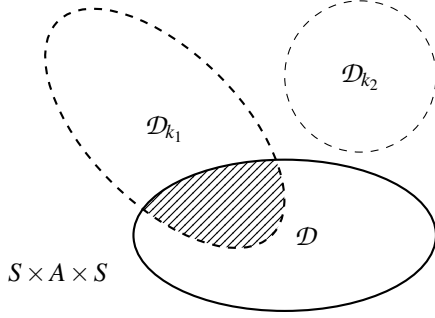


Figure 4: Representation of the support in  $S^2 \times A$  of the of  $\mathcal{D}$ ,  $\mathcal{D}_{k_1}$  and  $\mathcal{D}_{k_2}$ .  $k_1$  and  $k_2$  are two different alleged transformations. The shape and the position of the sets is decided according to the value of the log likelihood of the density estimate  $\mathcal{L}$  and quantile threshold  $\theta$ .  $\mathcal{D}_{k_1} \cap \mathcal{D} \neq \emptyset$  and  $\mathcal{D}_{k_2} \cap \mathcal{D} = \emptyset$ . Depending on the user chosen percentage threshold  $v$ ,  $k_1$  could be a symmetry while  $k_2$  no. If  $k_1$  is detected as a symmetry then data augmenting  $\mathcal{D}$  with  $\mathcal{D}_{k_1}$  means training the model over  $\mathcal{D} \cup \mathcal{D}_{k_1}$ . Notice that the data contained in  $\mathcal{D}_{k_1} \setminus \mathcal{D}$  are not present in the original batch  $\mathcal{D}$ .

### 3.3 Limitations

Both in the discrete and in the continuous case the approach is limited by the size and the variety of the data set. It is hard to tell how good an initial batch is, but this aspect could non negligibly affect the output of the paradigm. The approach in the continuous case is also limited by the impact that the transformation has on the representation of the transition. More in detail, if the distance in Euclidean space between  $(s, a, s')$  and  $k(s, a, s')$  is small, or if only a small fraction of the total number of components that represent the transition is affected by the transformation then the probability of the image of the alleged symmetry will be high, resulting in a false positive detection. With this in mind, appropriately preprocessing the data before density estimation could enormously affect the outcome of the procedure.

## 4 EXPERIMENTS

We test the algorithm in one discrete grid environment with periodic boundary conditions and in two

famous environments of OpenAI's Gym Learning Suite: CartPole and Acrobot.

### 4.1 Setup

We first collect a batch of transitions  $\mathcal{D}$  by acting in the environment with a uniform random policy. We suppose the presence of a symmetry  $k$  and we try to detect it using Algorithm 1 or Algorithm 2. We report  $\bar{v}_k$ , the average value plus or minus the standard deviation of the quantity  $v_k$  computed over an ensemble of  $N$  different iterations of the procedure (with  $N$  different batches  $\mathcal{D}$ ). We do not consider any threshold  $v$ , thus limiting the analysis at the phase of the calculation of  $v_k$  since the detection or not of a symmetry is highly dependent of an expert-wise domain dependent choice of  $v$ . In the end we show that detecting the right symmetry really results in learning more accurate models by computing  $\bar{\Delta}$ , the average of an estimate of the distributional shift  $\Delta$  over the various simulations. Since  $\Delta$  is measured differently depending on the typology of the environment (discrete or continuous) more details about the performances are illustrated in the next subsections and the results are reported in Table 4. In Tables 1, 2, 3 we adopted the following contracted notation: let  $a_1, a_2, a_3, a_4$  be some actions  $\in A$ , for space's sake we will use the notations  $g(a_1, a_2, \dots) = (a_3, a_4, \dots)$  to indicate  $g(a_1) = a_3, g(a_2) = a_4, \dots$ , etc.

### 4.2 Grid (Discrete)

In this environment the agent can move along fixed directions over a torus by acting with any  $a \in A = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ . The positions on the torus are the states  $s = (i, j)$  and the set  $S$  is represented as a grid of fixed size  $l$  and periodic boundary conditions (see Figure 5). Since there are no obstacles and the dynamics is deterministic this environment is endowed of many symmetric transformations and therefore can serve as an useful proof-of-concept.

We tested Algorithm 1 with six different alleged transformations  $k$  in a Grid with size  $l = 100$  over  $N =$

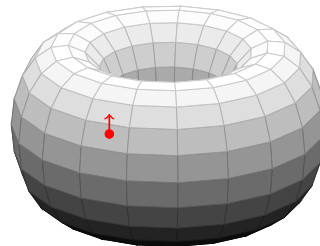


Figure 5: Representation of the Grid Environment. The red dot is the position of a state  $s$  on the torus. The displacement obtained by acting with action  $a = \uparrow$  is shown as a red arrow.

50 different simulations.

1. **Time Reversal Symmetry with Action Inversion (TRSAI).** Assuming that  $\downarrow$  is the reverse of  $\uparrow$  and  $\leftarrow$  is the reverse of  $\rightarrow$  we proposed the following transformation:  $k = (f(s) = s', g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow), l(s') = s)$ . The symmetric transformation is present in the batch with a percentage  $\bar{v}_k = 0.6 \pm 0.1$ .
2. **Same Dynamics with Action Inversion (SDAI).** When  $k = (f(s) = s, g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow), l(s') = s')$  the results clearly show that this transformation is not a symmetry. Indeed in this case  $\bar{v}_k = 0.0 \pm 0.0$ .
3. **Opposite Dynamics and Action Inversion (ODAI):**  $k = (f(s) = s, g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow), l(s') = s' \mp (2, 0) \vee (0, 2))$ . In other words we revert the action but also the final state is changed in order to reproduce the correct destination.  $\bar{v}_k = 0.4 \pm 0.1$  showing that the latter could be a symmetry of the environment.
4. **Opposite Dynamics but Wrong Action (ODWA).** The alleged transformation is like the one of Point 3, but the action is switched on the wrong axis (e.g.  $g(\uparrow) = \rightarrow$ ). Here  $\bar{v}_k = 0.0 \pm 0.0$  and not a symmetry.
5. **Translation Invariance (TI).**  $k = (f(s) = s', g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\uparrow, \downarrow, \leftarrow, \rightarrow), l(s') = s' \pm (1, 0) \vee (0, 1))$ . The proposed transformation proposes a displacement with similar effects when the same action is applied to the next state.  $\bar{v}_k = 0.5 \pm 0.1$ .  $k$  could be a symmetry.
6. **Translation Invariance with Opposite Dynamics (TIOD).** In this case the action is the same as Point 5 but the agent returns to the previous state. With  $\bar{v}_k = 0.0 \pm 0.0$  the latter is not a symmetry.

The said transformations are resumed in the Table 1 along with the results in Table 4.

With the scope of showing that augmenting the data set with the image of the symmetry  $k$  leads to better model we compute the distributional shift between the true model  $T$  and  $\hat{T}$ , the one learnt from  $\mathcal{D}$ , as the sum over every possible state-action pair of the Total Variation Distance for each transition. In other words let

$$d(T, \hat{T}) = \frac{1}{2} \sum_{(s, s') \in \mathcal{S}^2, a \in A} |T(s, a, s') - \hat{T}(s, a, s')| \quad (6)$$

be the distance between  $T$  and  $\hat{T}$  and let

$$d(T, \hat{T}_k) = \frac{1}{2} \sum_{(s, s') \in \mathcal{S}^2, a \in A} |T(s, a, s') - \hat{T}_k(s, a, s')| \quad (7)$$

Table 1: Grid. Proposed transformations and label.

$k$	Label
$f(s) = s'$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$ $l(s') = s$	TRSAI
$f(s) = s$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$ $l(s') = s'$	SDAI
$f(s) = s$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$ $l(s') = s' \mp (2, 0) \vee (0, 2)$	ODAI
$f(s) = s$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\rightarrow, \leftarrow, \uparrow, \downarrow)$ $l(s') = s' \mp (2, 0) \vee (0, 2)$	ODWA
$f(s) = s'$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\uparrow, \downarrow, \leftarrow, \rightarrow)$ $l(s') = s' \pm (1, 0) \vee (0, 1)$	TI
$f(s) = s'$ $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\uparrow, \downarrow, \leftarrow, \rightarrow)$ $l(s') = s$	TIOD

where  $\hat{T}_k$  is the dynamics inferred from  $\mathcal{A}_k$ , we report  $\bar{\Delta}$  in Table 4: the difference

$$\Delta = d(T, \hat{T}) - d(T, \hat{T}_k) \quad (8)$$

averaged over  $N = 50$  simulations, showing that when  $k$  is a true symmetry of the dynamics the said quantity is positive ( $d(T, \hat{T}_k) < d(T, \hat{T})$ ) and when  $k$  is not a real symmetry it is negative. Hence, augmenting the data set after having detected a true symmetry leads to better model learning, but the false detection of a symmetric transformation could also result in less representative models.

### 4.3 CartPole (Continuous)

As stated in the Introduction, the dynamics of CartPole is invariant with respect to the transformation  $k = (f(s) = -s, g(a) = -a, l(s') = -s) \forall (s, s') \in \mathcal{S}^2$ . In order to use Algorithm 2 we first map the actions to real numbers:  $\leftarrow = -1.5$  and  $\rightarrow = 1.5$ . We then normalize every state feature in the range  $[-1.5, 1.5]$ . We tested Algorithm 2 with four different alleged transformations  $h$  over  $N = 5$  different simulations, a batch of size  $|\mathcal{D}| = 10^3$  collected with a random policy, quantile order to compute the thresholds  $q = 0.1$ .

1. **State and Action Reflection with Respect to an Axis in  $x = 0$  (SAR).** Assuming that  $\leftarrow$  is the reverse of  $\rightarrow$  we proposed the following transformation:  $k = (f(s) = -s, g(\leftarrow, \rightarrow) = (\rightarrow, \leftarrow), l(s') = -s')$ . The symmetric transformation is present in the batch with a percentage  $\bar{v}_k = 0.80 \pm 0.05$ .
2. **Initial State Reflection (ISR).** We then tried the same transformation as before but without reflect-

ing the next state  $s'$ :  $k = (f(s) = -s, g(\leftarrow, \rightarrow) = (\rightarrow, \leftarrow), l(s') = s')$ . This is clearly not a symmetry because  $\bar{v}_k = 0.00 \pm 0.00$ .

3. **Action Inversion (AI).** What about reversing only the actions?  $k = (f(s) = s, g(\leftarrow, \rightarrow) = (\rightarrow, \leftarrow), l(s') = s')$  and  $\bar{v}_k = 0.00 \pm 0.00$ . Therefore this is not a symmetry of the environment according to our criterion.
4. **Single Feature Inversion (SFI).** We also tried to reverse only one single feature of the starting state:  $k = (f(x, v, \alpha, \omega) = (-x, v, \alpha, \omega), g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow), l(s') = s')$ . In this case  $\bar{v}_k = 0.05 \pm 0.03$ .
5. **Translation Invariance (TI).** We translated the position of the initial state  $x$  and that of the final state  $x'$  by an arbitrary value (0.3):  $k = (f(x, v, \alpha, \omega) = (x + 0.3, v, \alpha, \omega), g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow), l(x', v', \alpha', \omega') = (x' + 0.3, v', \alpha', \omega'))$ . The percentage  $v_k = 0.25 \pm 0.23$ . Given the high variance this could be detected as a symmetry or not depending on the simulation.

The proposed transformations are resumed in Table 2 along with the results in Table 4.

Table 2: CartPole. Proposed transformations and label.

$k$	Label
$f(s) = -s$ $g(\leftarrow, \rightarrow) = (\rightarrow, \leftarrow)$ $l(s') = -s'$	SAR
$f(s) = -s$ $g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow)$ $l(s') = s'$	ISR
$f(s) = s$ $g(\leftarrow, \rightarrow) = (\rightarrow, \leftarrow)$ $l(s') = s'$	AI
$f((x, \dots)) = (-x, \dots)$ $g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow)$ $l(s') = s'$	SFI
$f((x, \dots)) = (x + 0.3, \dots)$ $g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow)$ $l((x', \dots)) = (x' + 0.3, \dots)$	TI

With the aim of computing the distributional shift we fit over  $\mathcal{D}$  the evolution  $s'(s, a)$  with a Multi Layer Perceptron by minimizing the Mean Squared Error (MSE) between  $\hat{s}'$  and  $s$ . We reiterate the same procedure over  $\mathcal{D} \cup \mathcal{D}_k$  obtaining a next state  $\hat{s}'_k(s, a)$ . We then randomly generate  $10^6$  other transitions building up an evaluation batch  $\mathcal{D}_{ev}$  from the environment and we evaluate with respect to the MSE these two models over the new data set. We report in Table 4 the

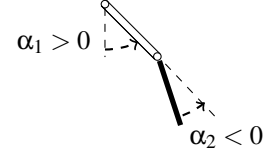


Figure 6: Representation of a state of the Acrobot environment.

difference:

$$\Delta = MSE_{\mathcal{D}_{ev}}(s', \hat{s}'(s, a)) - MSE_{\mathcal{D}_{ev}}(s', \hat{s}'_k(s, a)) \quad (9)$$

Again, when  $\Delta > 0$  augmenting the data set leads to better models  $\hat{s}'_k$ , but when  $\Delta < 0$  exploiting the transformation  $h$  is detrimental to the dynamics.

#### 4.4 Acrobot (Continuous)

The Acrobot environment consists of two poles linked with a rotating joint at one end. One of the poles is pinned to a wall with a second rotating joint (see Figure 6). The system is affected by gravity and hence the poles are hanging down. An agent can apply a negative torque to the lower pole ( $a = -1$ ), a positive one ( $a = 1$ ) or do nothing ( $a = 0$ ). The goal is to push the lower pole as high as possible. The state consists of the sine and cosine of the two rotational joint angles  $(\alpha_1, \alpha_2)$  and the joint angular velocities  $(\omega_1, \omega_2)$ :  $s = (\sin \alpha_1, \cos \alpha_1, \sin \alpha_2, \cos \alpha_2, \omega_1, \omega_2)$ . The dynamics is invariant under the transformation  $k = (f((\alpha_1, \alpha_2, \omega_1, \omega_2)) = (-\alpha_1, -\alpha_2, -\omega_1, -\omega_2)$  and  $g(a) = -a, l((\alpha'_1, \alpha'_2, \omega'_1, \omega'_2)) = (-\alpha'_1, -\alpha'_2, -\omega'_1, -\omega'_2) \forall (s, s') \in \mathcal{S}^2$ . In order to apply Algorithm 2 we first normalize the state features and the action in the interval  $[-3, 3]$ . We tested Algorithm 2 with four different alleged transformations  $k$  over  $N = 5$  different simulations, a batch of size  $|\mathcal{D}| = 10^3$  collected with a random policy, quantile order to compute the thresholds  $q = 0.1$ . The label of the transformations here after explained are resumed in Table 3.

##### 1. Angles and Angular Velocities Inversion (AAVI).

$$k = (f(\sin \alpha_1, \sin \alpha_2, \cos \alpha_1, \cos \alpha_2, \omega_1, \omega_2) = (-\sin \alpha_1, -\sin \alpha_2, \cos \alpha_1, \cos \alpha_2, -\omega_1, -\omega_2), g(a) = -a, l(\sin \alpha'_1, \sin \alpha'_2, \cos \alpha'_1, \cos \alpha'_2, \omega'_1, \omega'_2) = (-\sin \alpha'_1, -\sin \alpha'_2, \cos \alpha'_1, \cos \alpha'_2, -\omega'_1, -\omega'_2)).$$

This transformation could be a symmetry since the percentage of images with a probability higher than the threshold is  $\bar{v}_k = 0.86 \pm 0.03$ .

##### 2. Cosines and Angular Velocities Inversion (CAVI).

$$k = (f(\sin \alpha_1, \sin \alpha_2, \cos \alpha_1, \cos \alpha_2, \omega_1, \omega_2) = (\sin \alpha_1, \sin \alpha_2, -\cos \alpha_1, -\cos \alpha_2, -\omega_1, -\omega_2),$$



Table 3: Acrobot. Proposed transformations and label.

$k$	Label
$f((s_1, s_2, \omega_1, \omega_2, \dots)) = -(s_1, s_2, \omega_1, \omega_2, \dots)$ $g(-1, 0, 1) = (1, 0, -1)$	AAVI
$l((s'_1, s'_2, \omega'_1, \omega'_2, \dots)) = -(s'_1, s'_2, \omega'_1, \omega'_2, \dots)$ $f((c_1, c_2, \omega_1, \omega_2, \dots)) = -(c_1, c_2, \omega_1, \omega_2, \dots)$ $g(-1, 0, 1) = (1, 0, -1)$	CAVI
$l((c'_1, c'_2, \omega'_1, \omega'_2, \dots)) = -(c'_1, c'_2, \omega'_1, \omega'_2, \dots)$ $f(s) = s$ $g(-1, 0, 1) = (1, 0, -1)$ $l(s') = s'$	AI
$f(s) = -s$ $g(-1, 0, 1) = (-1, 0, 1)$ $l(s') = s'$	SSI

$$g(a) = -a, l(\sin \alpha'_1, \sin \alpha'_2, \cos \alpha'_1, \cos \alpha'_2, \omega'_1, \omega'_2) = (\sin \alpha'_1, \sin \alpha'_2, -\cos \alpha'_1, -\cos \alpha'_2, -\omega'_1, -\omega'_2).$$

This is not a symmetry since  $\bar{v}_k = 0.00 \pm 0.00$ .

- Action Inversion (AI).**  $k = (f(s) = s, g(a) = -a, l(s') = s')$ . In this case the results are not so indicative because  $\bar{v}_k = 0.35 \pm 0.09$  and therefore the transformation could be falsely detected as a symmetry if a low threshold  $v$  is chosen.
- Starting State Inversion (SSI).**  $k = (f(s) = -s, g(a) = a, l(s') = s')$ . This is clearly not a symmetric transformation because  $\bar{v}_k = 0.00 \pm 0.00$ .

The results can be found in Table 4 along with model performance. Using the said metrics  $\Delta$  to evaluate the gains of data augmentation, we notice that in this case the high dimensionality of the state space makes it harder for the MLP to reconstruct the next state and that even the discovery of the true symmetry  $k = (f((\alpha_1, \alpha_2, \omega_1, \omega_2)) = -(\alpha_1, \alpha_2, \omega_1, \omega_2), g(a) = -a, l((\alpha'_1, \alpha'_2, \omega'_1, \omega'_2)) = -(\alpha'_1, \alpha'_2, \omega'_1, \omega'_2))$  seems to not lead to the expected results because the confidence interval of  $\bar{\Delta}$  also falls on the negative part of the real axis.

## 4.5 Discussion

In the light of the results we can state that, when used with care, the proposed approaches manage to detect the alleged transformations of the transitions which are symmetries of the dynamics.

However, note that the present experiments in the continuous environments were performed with a quantile of order  $q = 0.1$ . The outcomes can greatly be affected by the choice of  $q$  and further studies should be carried out in the next future to address this limitation.

Moreover, the expert choice of  $v$  can also have a huge impact on the final detection. Indeed,  $v = 0.30$

in the Acrobot environment could lead to the misdetection of AI as a symmetry, while the TI symmetry of CartPole requires a very low ( $v < 0.20$ ) in order to be detected (Table 4).

## 5 CONCLUSIONS

The present work proposed an expert-guided approach to data augment the batch of trajectories used to learn a model in Offline Model Based RL. More specifically, new information is created assuming that the dynamics of the system is invariant with respect to some symmetric transformation of the state and action representations. The alleged symmetry is detected exploiting both expert knowledge and the fulfillment of an a priori established criterion. Experimental results in proof-of-concepts deterministic environments suggest that the proposed method could be an useful tool for the developer of Reinforcement Learning algorithms when he is ought to apply them to real world scenarios. Nevertheless, the strong dependency from preprocessing, from neural network hyperparameters to be fine tuned before training and the from the choice of the quantile order  $q$  and of the percentage threshold  $v$  impose that Algorithms 1 and 2 should be handled with care. Indeed, the false detection of a symmetry and the following data augmenting could lead to catastrophic aftermaths when the retrieved optimal policy is going to be applied to the real environment.

An immediate next step consists in estimating to what extent the solution to an MDP learnt offline benefits from the reduction in distributional shift obtained with the proposed paradigm.

Future research perspectives involve (1) using more recent Normalizing Flows approaches to perform density estimation, e.g. we do not exclude that state-of-the art continuous time Normalizing Flows who exploit ODE solvers like FJORD (Grathwohl et al., 2019) might yield even better results; (2) extending the approach to also tackle stochastic environments.

## ACKNOWLEDGEMENTS

This work was carried out during a visit at the Pompeu Fabra University (UPF) in Barcelona and was funded by the Artificiality and Natural Intelligence Toulouse Institute (ANITI) - Institut 3iA (ANR-19-PI3A-0004). We thank Hector Geffner, Anders Jonsson and the Ar-

Table 4: Results of the algorithms in all environments. The columns on the right point to the tested alleged symmetry, indexed by its label as reported in Table 1 for the toroidal Grid, Table 2 for CartPole and Table 3 for Acrobot. All experiments were performed with  $q = 0.1$ . The true symmetries are displayed in bold.

Environment	Metrics	Alleged Transformation					
		TRSAI	SDAI	ODAI	ODWA	TI	TIOD
Grid	$\bar{V}_k$	<b>0.6 ± 0.1</b>	0.0 ± 0.0	<b>0.4 ± 0.1</b>	0.0 ± 0.0	<b>0.5 ± 0.1</b>	0.0 ± 0.0
	$\bar{\Delta}$	<b>27 ± 9</b>	-14 ± 3	<b>47 ± 12</b>	-14 ± 3	<b>39 ± 9</b>	-17 ± 2
CartPole	$\bar{V}_k$	<b>0.80 ± 0.05</b>	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.03	<b>0.25 ± 0.23</b>	
	$\bar{\Delta}$	<b>4 ± 2 × 10<sup>-4</sup></b>	-10 ± 1 × 10 <sup>-2</sup>	-8 ± 1 × 10 <sup>-3</sup>	-4 ± 3 × 10 <sup>-3</sup>	<b>3 ± 2 × 10<sup>-4</sup></b>	
Acrobot	$\bar{V}_k$	<b>0.86 ± 0.03</b>	0.00 ± 0.00	0.35 ± 0.09	0.00 ± 0.00		
	$\bar{\Delta}$	<b>3.3 ± 6.6 × 10<sup>-3</sup></b>	-3.9 ± 1.9 × 10 <sup>-2</sup>	-0.6 ± 1.3 × 10 <sup>-2</sup>	-9.5 ± 4.3 × 10 <sup>-2</sup>		

tificial Intelligence and Machine Learning group of the UPF for their warm hospitality.

## REFERENCES

- Abel, D., Umbanhowar, N., Khetarpal, K., Arumugam, D., Precup, D., and Littman, M. (2020). Value Preserving State-Action Abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR.
- Bellman, R. (1966). Dynamic Programming. *Science*, 153(3731):34–37.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.
- Castro, P. S. (2020). Scalable Methods for Computing State Similarity in Deterministic Markov Decision Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076.
- Dean, T. and Givan, R. (1997). Model Minimization in Markov Decision Processes. In *AAAI/IAAI*, pages 106–111.
- Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: Non-linear Independent Components Estimation. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.
- Ferns, N., Panangaden, P., and Precup, D. (2004). Metrics for Finite Markov Decision Processes. In *UAI*, volume 4, pages 162–169.
- Givan, R., Dean, T., and Greig, M. (2003). Equivalence Notions and Model Minimization in Markov Decision Processes. *Artificial Intelligence*, 147(1-2):163–223.
- Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2019). FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Gross, D. J. (1996). The role of symmetry in fundamental physics. *Proceedings of the National Academy of Sciences*, 93(25):14256–14259.
- Kobyzev, I., Prince, S., and Brubaker, M. (2020). Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643.
- Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a Unified Theory of State Abstraction for MDPs. *ISAIM*, 4:5.
- Mandel, T., Liu, Y.-E., Brunskill, E., and Popovic, Z. (2016). Efficient Bayesian Clustering for Reinforcement Learning. In *IJCAI*, pages 1830–1838.
- Mausam and Kolobov, A. (2012). *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers.
- Narayanamurthy, S. M. and Ravindran, B. (2008). On the Hardness of Finding Symmetries in Markov Decision Processes. In *Proceedings of the 25th international conference on Machine learning*, pages 688–695.
- Ravindran, B. and Barto, A. G. (2001). Symmetries and Model Minimization in Markov Decision Processes. Technical report, USA.
- Ravindran, B. and Barto, A. G. (2004). Approximate Homomorphisms: A Framework for Non-exact Minimization in Markov Decision Processes.
- Ruan, S. S., Comanici, G., Panangaden, P., and Precup, D. (2015). Representation Discovery for MDPs Using Bisimulation Metrics. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.
- Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):1–48.
- Sutton, R. S. (1990). Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In Porter, B. and Mooney, R., editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA).
- Taylor, J., Precup, D., and Panagaden, P. (2009). Bounding Performance Loss in Approximate MDP Homomorphisms. In Koller, D., Schuurmans, D., Bengio,

- Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.
- van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. (2020a). Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '20*, page 1431–1439, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. (2020b). MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4199–4210. Curran Associates, Inc.

