
Koopman Q-learning: Offline Reinforcement Learning via Symmetries of Dynamics

Matthias Weissenbacher¹ Samarth Sinha² Animesh Garg² Yoshinobu Kawahara^{1,3}

Abstract

Offline reinforcement learning leverages large datasets to train policies without interactions with the environment. The learned policies may then be deployed in real-world settings where interactions are costly or dangerous. Current algorithms over-fit to the training dataset and as a consequence perform poorly when deployed to out-of-distribution generalizations of the environment. We aim to address these limitations by learning a Koopman latent representation which allows us to infer symmetries of the system’s underlying dynamic. The latter is then utilized to extend the otherwise static offline dataset during training; this constitutes a novel data augmentation framework which reflects the system’s dynamic and is thus to be interpreted as an exploration of the environment’s state space. To obtain the symmetries we employ Koopman theory in which non-linear dynamics are represented in terms of a linear operator acting on the space of measurement functions of the system. We provide novel theoretical results on the existence and nature of symmetries relevant for control systems such as reinforcement learning settings. Moreover, we empirically evaluate our method on several benchmark offline reinforcement learning tasks and datasets including D4RL, Metaworld and Robosuite and find that by using our framework we consistently improve the state-of-the-art of model-free Q-learning methods.

1. Introduction

The recent impressive advances in reinforcement learning (RL) range from robotics, to strategy games and recommendation systems (Kalashnikov et al., 2018; Li et al., 2010). Reinforcement learning is canonically regarded as an active learning process - also referred to as online RL - where the agent interacts with the environment at each training run. In contrast, offline RL algorithms learn from large, previously collected static datasets, and thus do not rely on environment interactions (Agarwal et al., 2020; Ernst et al., 2005; Fujimoto et al., 2019). Online data collection is performed by simulations or by means of real world interactions e.g. robotics and in either scenario interactions may be costly and/or dangerous.

In principle offline datasets only need to be collected once which alleviates the before-mentioned shortcomings of costly online interactions. Offline datasets are typically collected using behavioral policies for the specific task ranging from, random policies, or near-optimal policies to human demonstrations. In particular, being able to leverage the latter is a major advantage of offline RL over online approaches, and then the learned policies can be deployed or fine-tuned on the desired environment. Offline RL has successfully been applied to learn agents that outperform the behavioral policy used to collect the data (Kumar et al., 2020; Wu et al., 2019; Agarwal et al., 2020; Ernst et al., 2005). However algorithms admit major shortcomings in regard to over-fitting and overestimating the true state-action values of the distribution. One solution was recently proposed by Sinha et al. (2021), where they tested several data augmentation schemes to improve the performance and generalization capabilities of the learned policies.

However, despite the recent progress, learning from offline demonstrations is a tedious endeavour as the dataset typically does not cover the full state-action space. Moreover, offline RL algorithms per definition do not admit the possibility for further environment exploration to refine their distributions towards an optimal policy. It was argued previously that is basically impossible for an offline RL agent to learn an optimal policy as the generalization to near data generically leads to compounding errors such as overestimation bias (Kumar et al., 2020). In this paper, we look at

¹RIKEN Center for Advanced Intelligence Project, Japan ² Vector Institute, University of Toronto, Canada ³Institute of Mathematics for Industry, Kyushu University, Japan . Correspondence to: Matthias Weissenbacher <matthias.weissenbacher@a.riken.jp>.

offline RL through the lens of Koopman spectral theory in which non-linear dynamics are represented in terms of a linear operator acting on the space of measurement functions of the system. Through which the representation the symmetries of the dynamics may be inferred directly, and can then be used to guide data augmentation strategies see Figure 1. We further provide theoretical results on the existence on nature of symmetries relevant for control systems such as reinforcement learning. More specifically, we apply Koopman spectral theory by: first learning symmetries of the system’s underlying dynamic in a self-supervised fashion from the static dataset, and second employing the latter to extend the offline dataset at training time by out-of-distribution values. As this reflects the system’s dynamics the additional data is to be interpreted as an exploration of the environment’s state space.

Some prior works have explored symmetry of the state-action space in the context of Markov Decision Processes (MDP’s) (Higgins et al., 2018; Balaraman & Andrew, 2004; van der Pol et al., 2020) since many control tasks exhibit apparent symmetries e.g. the cart-pole is translation symmetric w.r.t. its position. The paradigm introduced in this work is of a different nature entirely. The distinction is twofold: first, the symmetries are learned in a self-supervised way and are in general not apparent to the developer; second: we concern with symmetry transformation from state tuples $(s_t, s_{t+1}) \rightarrow (\tilde{s}_t, \tilde{s}_{t+1})$ which leave the action invariant. The latter, are inferred from the dynamics inherited by the behavioral policy of the underlying offline data. In other words we seek to derive a neighbourhood around a MDP tuple in the offline dataset in which the behavioral policy is likely to choose the same action based on its dynamics in the environment. In practice the Koopman latent space representation is learned in a self-supervised manner by training to predict the next state using a VAE model (Kingma & Welling, 2013).

To summarize, in this paper, we propose Koopman Forward (Conservative) Q-learning (KFC): a model-free Q-learning algorithm which uses the symmetries in the dynamics of the environment to guide data augmentation strategies. We also provide thorough theoretical justifications for KFC. Finally, we empirically test our approach on several challenging benchmark datasets from D4RL (Fu et al., 2021), Meta-World (Yu et al., 2019) and Robosuite (Zhu et al., 2020) and find that by using KFC we can improve the state-of-the-art on most benchmark offline reinforcement learning tasks.

2. Preliminaries and background

2.1. Offline RL & Conservative Q-learning

Reinforcement learning algorithms train policies to maximize the cumulative reward received by an agent who inter-

acts with an environment. Formally the setting is given by a Markov decision process (S, A, ρ, r, γ) , with state space S , action space A , and $\rho(s_{t+1}|s_t, a_t)$ the transition density function from the current state and action to the next state. Moreover, γ is the discount factor and $r(S_t)$ the reward function. At any discrete time the agent chooses an action $a_t \in A$ according to its underlying policy $\pi_\theta(a_t|s_t)$ based on the information of the current state $s_t \in S$ where the policy is parametrized by θ .

We focus on the Actor-Critic methods for continuous control tasks in the following. In deep RL the parameters θ are the weights in a deep neural network function approximation of the policy or Actor as well as the state-action value function Q or Critic, respectively, and are optimized by gradient decent. The agent i.e. the Actor-Critic is trained to maximize the expected γ -discounted cumulative reward $E_\pi[\sum_{t=0}^T \gamma^t r_\pi(s_t, a_t)]$, with respect to the policy network i.e. its parameters θ . For notational simplicity we omit the explicit dependency of the latter in the remainder of this work. Furthermore the state-action value function $Q(s_t, a_t)$, returns the value of performing a given action a_t while being in the state s_t . The Q-function is trained by minimizing the Bellman error as

$$\hat{Q}_{i+1} = \arg \min_Q E \left[\left(r_t + \gamma \hat{Q}_i(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right)^2 \right]. \quad (1)$$

This is commonly referred to as the i^{th} policy evaluation step where the hat denotes the target Q-function. In offline RL one aims to learn an optimal policy for the given the dataset $D = \bigcup_{k=1, \dots, \# \text{data-points}} (s_t, a_t, r_t, s_{t+1})_k$ as the option for exploration of the MDP is not available. The policy is optimized to maximize the state-action value function via the policy improvement

$$\pi_{i+1} = \arg \max_\pi E_{s_t \sim D} [\hat{Q}_i(s_t, \pi(a_t|s_t))]. \quad (2)$$

CQL algorithm: CQL is built on top of a Soft Actor-Critic algorithm (SAC) (Haarnoja et al., 2018), which employs soft-policy iteration of a stochastic policy (Haarnoja et al., 2017). A policy entropy regularization term is added to the policy improvement step in Eq. (2) as

$$\pi_{i+1} = \arg \max_\pi E_{s_t \sim D} \left[\hat{Q}_i(s_t, \pi(a_t|s_t)) - \alpha \log \pi(a_t|s_t) \right] \quad (3)$$

where α either is a fixed hyperparameter or may be chosen to be trainable.

CQL reduces the overestimation of state-values - in particular those out-of distribution from D . It achieves this by regularizing the Q-function in Eq. (1) by a term minimizing its values over out of distribution randomly sampled actions. In the following a_{t+1} is given by the prediction of the policy $\pi_i(a_{t+1}|s_{t+1})$. The policy optimisation step Eq. (1) is

modified by a regularizer term

$$\arg \min_Q \mathbb{E}_{s_t \sim D} \left[\log \sum_a \exp(Q(s_t, a)) \right] - \alpha \mathbb{E}_{\pi(s_t)} [Q(s_t, a)] . \quad (4)$$

Where we have omitted the hyperparameter $\tilde{\alpha}$ balancing the regularizer term in Eq. (4).

2.2. Koopman theory

Historically, the Koopman theoretic perspective of dynamical systems was introduced to describe the evolution of measurements of Hamiltonian systems (Koopman, 1931; Mezić, 2005). The underlying dynamic of most modern reinforcement learning tasks is of non-linear nature, i.e. the agents actions lead to changes of its state described by a complex non-linear dynamical system. In contrast to linear systems which are completely characterized by their spectral decomposition non-linear systems lack such a unified characterisation.

The Koopman operator theoretic framework describes non-linear dynamics via a linear infinite-dimensional Koopman operator and thus inherits certain tools applicable to linear control systems (Mauroy et al., 2020; Kaiser et al., 2021). In practice one aims to find a finite-dimensional representation of the Koopman operator which is equivalent to obtaining a coordinate transformations in which the non-linear dynamics are approximately linear. A general non-affine control system is governed by the system of non-linear ordinary differential equations (ODEs) as

$$\dot{s} = f(s, a) , \quad (5)$$

where s , is the n -dimensional state vector, a the m -dimensional action vector with $(s, a) \in \mathcal{S} \times \mathcal{A}$ the state-action-space. Moreover, $\dot{s} = \partial_t s$ is the time derivative, and f is some general non-linear - at least C^1 -differentiable - vector valued function. For a discrete time system, Eq. (5) takes the form

$$s_{t+1} = F(s_t, a_t) , \quad (6)$$

where s_t denotes the state at time t where F is at least C^1 -differentiable vector valued function.

Definition 2.1 (Koopman operator). Let $\mathcal{K}(\mathcal{S} \times \mathcal{A})$ be the (Banach) space of all measurement functions (observables). Then the Koopman operator $K : \mathcal{K}(\mathcal{S} \times \mathcal{A}) \rightarrow \mathcal{K}(\mathcal{S} \times \mathcal{A})$ is defined by

$$Kg(s_t, a_t) = g(F(s_t, a_t), a_{t+1}) = g(s_{t+1}, a_{t+1}) , \quad (7)$$

for all observables $g \in \mathcal{K}(\mathcal{S} \times \mathcal{A})$ where $g : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$.

Many systems can be modeled by a bilinearisation where the action enters the controlling equations 5 linearly as

$$f(s, a) = f_0(s) + \sum_i^m f_i(s) a_i , \quad (8)$$

for f_i , $i = 0, \dots, m$ C^1 -differentiable-vector valued functions. In that case the action of the Koopman operator takes the simple form

$$g(s_{t+1}) = K(a)g(s_t) = \left(K_0 + \sum_i^m K_i a_i \right) g(s_t) , \quad (9)$$

$g \in \mathcal{K}(\mathcal{S})$ where $\mathcal{K}(\mathcal{S})$ is a (Banach) space of measurement functions K_0, K_i decompose the Koopman operator in a the free and forcing-term, respectively. Details on the existence of a representation as in Eq. (9) are discussed in (Goswami & Paley, 2017). Associated with a Koopman operator is its eigenspectrum, that is, the eigenvalues λ , and the corresponding eigenfunctions $\varphi_\lambda(s, a)$, such that

$$K\varphi_\lambda(s, a) = \lambda\varphi_\lambda(s, a) . \quad (10)$$

In practice one derives a finite set of observable

$$\vec{g} = (g_1, \dots, g_N) ,$$

in which case the approximation to the Koopman operator admits a finite-dimensional matrix representation. The $N \times N$ matrix representing the Koopman operator may be diagonalized by a matrix U containing the eigen-vectors of K as columns. In which case the eigen-functions are derived by $\vec{\varphi} = U\vec{g}$ and one infers from Eq. (5) that $\dot{\varphi}_i = \lambda_i \varphi_i$, for $i = 1, \dots, N$ with eigenvalues $\lambda_{i=1, \dots, N}$. These ODEs admit simple solutions for their time-evolution namely the exponential functions $\exp(\lambda_i t)$.

3. The Koopman forward framework

We initiate our discussion with a focus on symmetries in dynamical control systems in Subsection 3.1 where we additionally present some theoretical results. We then proceed in Subsection 3.2 by presenting the Koopman forward framework for Q-learning based on CQL. Moreover, we discuss the algorithm as well as the Koopman Forward model's deep neural network architecture.

3.1. Symmetries of dynamical control systems

Informal Overview: Our theoretical results culminate in theorem 3.8 and 3.9, which provides a road-map on how specific symmetries of the dynamical control system are to be inferred from a VAE forward prediction model. In particular, theorem 3.8 guarantees that the procedure leads to symmetries of the system (at least locally) and theorem 3.9 that actual new data points can be derived by applying symmetry transformations to existing ones. To make this topic more approachable we have shifted some technical details to the appendix A.

For the reader who wants to skip this technical section directly to the algorithm in section 3.2 let us summarize a

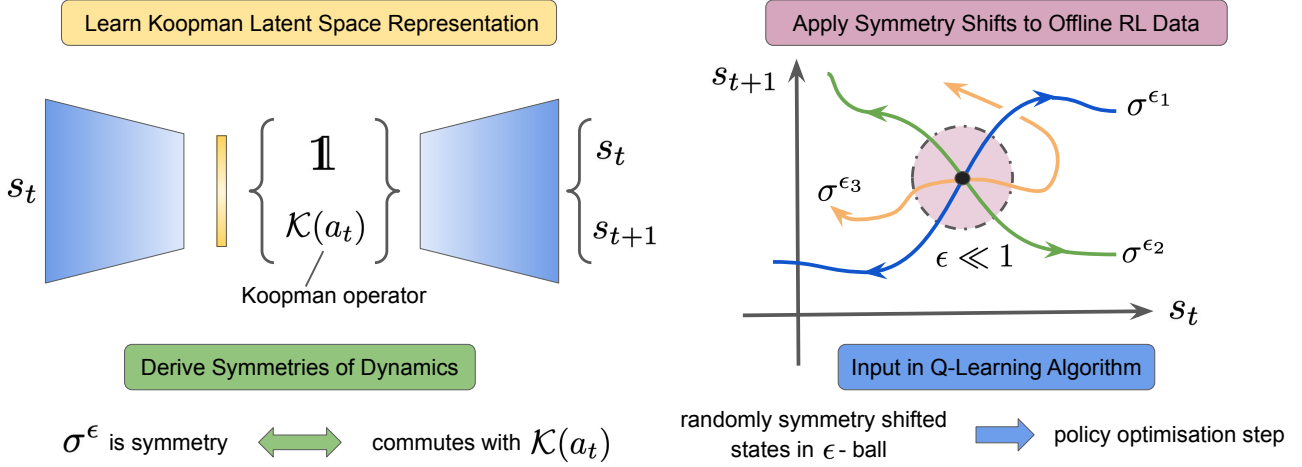


Figure 1. Overview of Koopman Q-learning. The states of the data point (s_t, s_{t+1}, a_t) are shifted along symmetry trajectories parametrized by $\epsilon_{1,2,3}$ for constant a_t . Symmetry transformations can be combined to reach other specific subset of the ϵ -ball region.

couple of take-away messages. Trajectories of many RL-environments are often solutions of ODEs. Symmetries can map solutions i.e. trajectories to each other. Thus having solutions (offline RL data) the symmetry map provides one the tools to generate novel data points. However, symmetry maps are hard to come by. In this section we provide a principled approach on how to derive this symmetry maps for control system using Koopman theory. Vaguely an operator is a symmetry generator if it commutes with the Koopman operator.

In general, a symmetry group Σ of the state space may be any subgroup of the group of isometrics of the Euclidean space. We consider Euclidean state spaces and in particular, Σ invariant compact subsets thereof.

Definition 3.1 (Equivariant Dynamical System). Consider the dynamical system $\dot{s} = f(s)$ and let Σ be a group acting on the state-space S . Then the system is called Σ -equivariant if $f(\sigma s) = \sigma f(s)$, for $s \in S$, $\sigma \in \Sigma$. For a discrete time dynamical system $s_{t+1} = F(s_t)$ one defines equivariance analogously, namely if $F(\sigma s_t) = \sigma F(s_t)$, for $s_t \in S$, $\sigma \in \Sigma$.

A system of ODEs may be represented by the set of its solutions which are locally connected by Lie Groups. In other words the symmetry maps are given by elements of the Lie Group which map between solutions.

Definition 3.2 (Symmetries of ODEs). A symmetry of a system of ODEs on a locally-smooth structure is a locally-defined diffeomorphism that maps the set of all solutions to itself.

Definition 3.3 (Local Lie Group). A parametrized set of transformations $\sigma^\epsilon : S \rightarrow S$ with $s \mapsto \tilde{s}(s, \epsilon)$ for $\epsilon \in (\epsilon_{low}, \epsilon_{high})$ where $\epsilon_{low} < 0 < \epsilon_{high}$ is a one-parameter

local Lie group if¹

1. σ^0 is the identity map; for $\epsilon = 0$ such that $\tilde{s}(s, 0) = s$.
2. $\sigma^{\epsilon_1} \sigma^{\epsilon_2} = \sigma^{\epsilon_2} \sigma^{\epsilon_1} = \sigma^{\epsilon_1 + \epsilon_2}$ for every $j \in \{1, 2\}$.
3. $\tilde{s}(s, \epsilon)$ admits a Taylor series expansion in ϵ ; in a neighbourhood of s determined by $\epsilon = 0$ as $\tilde{s}(s, \epsilon) = s + \epsilon \zeta(s) + O(\epsilon^2)$.

The Koopman operator of equivariant dynamical system is reviewed in the appendix A. Let us next turn to the case relevant for RL, namely control systems. In the remainder of this section we focus on dynamical systems given as in Eq. (6) and Eq. (9).

Definition 3.4 (Action-Equivariant Dynamical System). Let $\bar{\Sigma}$ be a group acting on the state-action-space $S \times A$ of a general control system as in Eq. (6) such that it acts as the identity operation on A i.e. $\sigma(s_t, a_t) = (\sigma s_t, a_t)$, $\forall \sigma \in \bar{\Sigma}$. Then the system is called $\bar{\Sigma}$ -action-equivariant if

$$F(\sigma(s_t, a_t)) = \sigma F(s_t, a_t), \text{ for } (s_t, a_t) \in S \times A. \quad (11)$$

Lemma 3.5. The map $\sigma^\epsilon : \bar{\Sigma} \rightarrow K(S \times A)$ given by $(\sigma^\epsilon g)(s, a) = g(\sigma^{-1} s, a)$ defines a group action on the Koopman space of observables $K(S \times A)$.

Theorem 3.6. A Koopman operator K of a $\bar{\Sigma}$ -action-equivariant system $s_{t+1} = F(s_t, a_t)$ satisfies

$$[\sigma^\epsilon(Kg)](s_t, a_t) = [K(\sigma^\epsilon g)](s_t, a_t). \quad (12)$$

¹The points (1) and (2) in definition 3.3 imply the existence of an inverse element $\sigma^{\epsilon^{-1}} = \sigma^{-\epsilon}$ for $j \in \{1, 2\}$. A local Lie group satisfies the group axioms for sufficiently small parameters values; it may not be a group on the entire set.

In particular, it is easy to see that the bilinearisation in Eq. (9) is Σ -action-equivariant if $f_i(\sigma_{j_S} s) = \sigma_{j_S} f_i(s)$, $\delta i = 0, \dots, m$. Let us thus turn our focus on the relevant case of a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ which admits a Koopman operator description as

$$g(s_{t+1}) = K(a_t)g(s_t), \text{ for } a_t \in A, \delta g \in \mathcal{K}(S), \quad (13)$$

where $fK(a)g_{a \in A}$ is a family of operators with analytical dependence on $a \in A$. Note that the bilinearisation in Eq. (9) is a special case of Eq. (13).²

Definition 3.7 ($\hat{\Sigma}$ -Action-Equivariant Control System). We refer to a control system as in Eq. (13) admitting a symmetry map $\hat{\sigma}_a$ as $\hat{\Sigma}$ -action-equivariant. The definition of the map $\hat{\sigma}_a$ is a bit more involved, see the appendix A.9. Informally, we note that it will connect a symmetry map depending on the action space σ_a and the Koopman observable as $(\sigma_a \hat{\sigma}_a)(s) \stackrel{?}{=} g(\sigma^{-1} s)$. The latter, defines a group action on the Koopman space of observables $\mathcal{K}(S)$.

Theorem 3.8. Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$ -action-equivariant control system i.e. admitting a symmetry action as in definition 3.7 and a Koopman operator representation as

$$g(s_{t+1}) = K(a_t)g(s_t), \text{ for } a_t \in A, \delta g \in \mathcal{K}(S). \quad (14)$$

Then

$$[\sigma_{a_t} \hat{\sigma}_a (K(a_t)g)](s_t) = [K(a_t) (\sigma_{a_t} \hat{\sigma}_a g)](s_t). \quad (15)$$

Moreover, a control system obeying equations 14 and 15 is $\hat{\Sigma}$ -action-equivariant locally if g^{-1} exists for a neighborhood of s_t , i.e. then $\sigma^{-1} \tilde{F}(s_t, a_t) = \tilde{F}(\sigma^{-1}(s_t, a_t))$.

To allow an easier transition to the empirical section we introduce the notation $E : S \rightarrow \mathcal{K}(S)$ and $D : \mathcal{K}(S) \rightarrow S$, denoting the C^1 -differentiable encoder and decoder to and from the finite-dimensional Koopman space approximation, respectively; $E \circ D = D \circ E = id$. Data-points may be shifted by symmetry transformations of solutions of ODEs as discussed next.

Theorem 3.9. Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a control system as in Eq. (14) and σ_{a_t} an operator obeying Eq. (15). Then $\sigma_{a_t}^\epsilon : (s_t, s_{t+1}, a_t) \mapsto (\tilde{s}_t, \tilde{s}_{t+1}, a_t)$ with

$$\begin{aligned} \tilde{s}_t &= D\left((1 + \epsilon \sigma_{a_t}) \hat{\sigma}_a E(s_t)\right), \\ \tilde{s}_{t+1} &= D\left((1 + \epsilon \sigma_{a_t}) \hat{\sigma}_a E(s_{t+1})\right) \end{aligned} \quad (16)$$

is a one-parameter local Lie symmetry group of ODEs. In other words one can use a symmetry transformation to shift both s_t as well as s_{t+1} such that $\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t)$.³

²See Appendix A for the details. We found that, in practice the Koopman operator learned by the neural nets is diagonalizable almost everywhere.

³No assumptions on the Koopman operator are imposed. More-

3.2. KFC algorithm

The previous section provides a theoretical foundation to derive symmetries of dynamical control systems based on a Koopman latent space representation. A vague take-away message is that an operator is a symmetry generator if it commutes with the Koopman operator. Thus the algorithmic challenge of our framework is twofold. Firstly, to find a numerical approximation of the Koopman space and Koopman operator and secondly to derive operators which commute with the latter.

In practice the Koopman latent space representation is finite-dimensional a so called Koopman invariant subspace. As a consequence the Koopman operator and the symmetry generators admit finite matrix representations. To derive the Koopman representation there exist elaborate techniques such as e.g. (Morton et al., 2019) which have shown significant advantages over a basic variational auto-encoder (VAE). However, we use the latter for simplicity. Lastly, after having learned a Koopman representation and derived symmetry maps the objective is to generate augmented data points; which are then used by the Q-learning algorithm such as CQL at training-time.

The KFC-algorithm :

Step 1: Train the Koopman forward model to learn the Koopman latent representation.

Step 2: Compute the symmetry transformations σ_a^ϵ ; we use two approaches denoted by **KFC** and **KFC++**.

Step 3: Integration into Q-learning algorithm; replace input data $(s_t, s_{t+1}, a, r) \mapsto (\sigma_a^\epsilon(s_t), \sigma_a^\epsilon(s_{t+1}), a, r)$.

Step 1: Pre-training of a Koopman forward model

$$F^c(s_t, a_t) = \begin{cases} \mathbf{c} = 0 & \text{VAE : } D(E(s_t)) = s_t \\ \mathbf{c} = 1 & \text{forward prediction :} \\ & D\left((K_0 + \sum_{i=1}^m K_i(a_t, i))E(s_t)\right) = s_{t+1} \end{cases} \quad (17)$$

Both E and D are approximated by Multi-Layer-Perceptrons (MLPs) and the bilinear Koopman-space operator is implemented as fully connected layers (no bias) with weight matrices K_i , $i = 0, \dots, m$, respectively. For training details we refer the reader to Appendix C.

Step 2: Computation of symmetry transformations

From theorem 3.9 one infers that in order for Eq. (18) to be a local Lie symmetry group of the approximate ODEs cap-

over, note that the equivalent theorem holds when $\epsilon \sigma_{a_t} \mapsto \sum_{I=1}^I \epsilon_I \sigma_{a_t}^I$ to be a local N-parameter Lie group.

tured by the neural net Eq. (17) one needs σ_{a_t} to commute with the approximate Koopman operator in Eq. (17).

For the KFC option in Eq. (18) the symmetry generator matrix σ_a is obtained by solving the equation

$$\sigma_{a_t} K(a_t) - K(a_t) \sigma_{a_t} = 0 ,$$

which may be accomplished by employing a Sylvester algorithm (syl). For KFC++ we compute the eigen-vectors of $K(a_t)$ which then constitute the columns of $U(a_t)$. Thus in particular one infers that

$$[\sigma_{a_t}(\vec{\epsilon}), K(a_t)] = 0 ,$$

The latter, is solved by construction of $\sigma_{a_t}(\vec{\epsilon})$ which commutes with the Koopman operator for all values of the random variables.⁴ The symmetry shifts in latent space are given by

$$\text{KFC: } \sigma_{\mathbf{a}_t}^\epsilon : s \mapsto \tilde{s} = D \left((1 + \epsilon \sigma_{a_t}) E(s) \right) \quad (18)$$

$$\text{KFC++: } \sigma_{\mathbf{a}_t}^\epsilon : s \mapsto \tilde{s} = D \left((1 + \sigma_{a_t}(\vec{\epsilon})) E(s) \right) ,$$

$$\sigma_{a_t}(\vec{\epsilon}) = \text{Re} \left(U(a_t) \text{diag}(\epsilon_1, \dots, \epsilon_N) U^{-1}(a_t) \right) ,$$

where $U(a_t)$ diagonalizes the Koopman operator with eigenvalues λ_i , $i = 1, \dots, N$ i.e. the latter can be expressed as

$$K(a_t) = U(a_t) \text{diag}(\lambda_1, \dots, \lambda_N) U^{-1}(a_t) .$$

Note that we abuse our notation as

$$E(s) \quad \vec{g}(s) = (g_1(s), \dots, g_N(s)) ,$$

i.e. the encoder provides the Koopman space observables. Eq. (18) holds for both s_t as well as s_{t+1} thus we have deliberately dropped the subscript. The parameters ϵ , $\vec{\epsilon}$ in Eq. (18) are normally distributed random variables.

Moreover, note that in for KFC++ the symmetry transformations are of the form $\sigma_{a_t}(\vec{\epsilon}) = \sum_{I=1}^N \epsilon_I \sigma_{a_t}^I$. Lastly, in the finite-dimensional Koopman invariant sub-spaces matrix multiplication replaces the formal mapping $\hat{\cdot}$ used in the theorems of section 3.1.

Step 3: Integration into a specific Q-learning algorithm (CQL).

Following (Sinha et al., 2021) we leave the policy improvement Eq. (3) unchanged but modify the policy optimisation step Eq. (1) as

$$\hat{Q}_{i+1} = \tilde{\alpha} \text{ Q-regulizer term} +$$

⁴ $[\cdot, \cdot]$ denotes the commutator of two matrices. The Koopman operators eigenvalues and eigen-vectors generically are \mathbb{C} -valued. However, the definition in Eq. (18) ensures that $\sigma_a(\vec{\epsilon})$ is a \mathbb{R} -valued matrix.

$$\arg \min_Q \left(\mathbb{E}_{\substack{a_t, r_t \\ s_t, s_{t+1}}} \left[D \left[\left(r_t + \gamma \hat{Q}_i \left(\sigma_{a_t}^\epsilon (\tilde{s}_{t+1} | s_{t+1}), a_{t+1} \right) \right) \right. \right. \right. \\ \left. \left. \left. Q \left(\sigma_{a_t}^\epsilon (\tilde{s}_t | s_t), a_t \right) \right)^2 \right] \right) , \quad (19)$$

where the unmodified Q-regulizer term is given in Eq. (4).

Additional remarks: Symmetry shifts are employed with probability $p_K = 0.8$; otherwise a normally distributed state shift is used. The KFC and KFC++ algorithms constitute a simple starting point to extract symmetries. More elaborate studies employing the literature on Koopman spectral analysis are desirable. The reward r_t is not part of the symmetry shift process and will just remain unchanged as an assertion.

3.3. Limitations

Note that in practice the assumptions of theorem 3.9 hold approximately, e.g. $D \approx E \approx id$. The resulting symmetry transformation shifts both s_t as well as s_{t+1} which evades the necessity of forecasting states. Thus the use of an inaccurate fore-cast model is avoided and the accuracy and generalisation capabilities of the VAE are fully utilized. To limit out-of-distribution generalisation errors the magnitude of the induced shift is controlled by the parameter $\epsilon \ll 1$ such that $\|s - \tilde{s}\| = O(\epsilon)$. KFC is computationally less expensive than KFC++ however it provides less freedom to explore different symmetry directions than the latter.

Theorems 3.8 and 3.9 imply that if an operator commutes with the Koopman operator we can find a local symmetry group. However, global conditions may not be easily inferred, see section 3.4 for an example. Practically the VAE model parametrized by a neural net is trained on data from one of many behavior policies and is thus learning an approximate dynamics. The theoretical limitations are twofold, firstly the theorems only hold for dynamical systems with differentiable state-transitions; secondly, we employ a Bilinearisation model for the Koopman operator of the system.

Many RL environments incorporate dynamics with discontinuous ‘‘contact’’ events where the Bilinearisation Ansatz Eq. (9) may not be applicable. However, empirically we find that our approach nevertheless is successful for environments with ‘‘contact’’ events. The latter does not affect the performance significantly (see Appendix D.1).

3.4. Example: Cartpole translation symmetry

We illustrate the KFC-algorithm for deriving symmetries from the Koopman latent representation. Moreover, we highlight that this is possible when purposefully employing imprecise Koopman observables and operators. We refer the reader to the appendix D.3 for a more comprehensive discussion.

The Cartpole environment admits a simple translation symmetry i.e. the dynamics are independent of its position x . We choose the Koopman observables to simply be the state vector $g = s = (x, \dot{x}, \theta, \dot{\theta})$; position, angle and their velocities. We empirically train the Koopman forward model Eq. (17) on trajectories collected by a set of expert policies. We conclude that the translation symmetry operation emerges when imposing it to be commuting with the Koopman operator.

Note that the Mujoco tasks in D4RL also admit a position translation symmetry. However, in the latter case it is common practice to omit the position information from the state vector to simplify the policy learning process. Among other local symmetries our approach can learn those translation symmetries and is thus likely to give additional performance boosts when the position information is not truncated a priori in Mujoco. As most environments do not admit global symmetries the benefit of our approach is due to utilizing local symmetries.

4. Empirical evaluation

In this section, we will first experiment with the popular D4RL benchmark commonly used for offline RL (Fu et al., 2021). The benchmark covers various different tasks such as locomotion tasks with Mujoco Gym (Brockman et al., 2016), tasks that require hierarchical planning such as antmaze, and other robotics tasks such as kitchen and adroit (Rajeswaran et al., 2017). Furthermore, similar to S4RL (Sinha et al., 2021), we perform experiments on 6 different challenging robotics tasks from MetaWorld (Yu et al., 2019) and RoboSuite (Zhu et al., 2020). We compare KFC to the baseline CQL algorithm (Kumar et al., 2020), and two best-performing augmentation variants from S4RL, S4RL- N and S4RL-adv (Sinha et al., 2021). We use the exact same hyperparameters as proposed in the respective papers. Furthermore, similar to S4RL, we build KFC on top of CQL (Kumar et al., 2020) to ensure conservative Q-estimates.

4.1. D4RL benchmarks

We present results in the benchmark D4RL test suite and report the normalized return in Table 1. We see that both KFC and KFC++ consistently outperform both the baseline CQL and S4RL across multiple tasks and data distributions. Outperforming S4RL-variants on various different types of environments suggests that KFC and KFC++ fundamentally improves the data augmentation strategies discussed in S4RL.

KFC-variants also improve the performance of learned agents on challenging environments such as antmaze: which requires hierarchical planning, kitchen and adroit tasks: which are sparse reward and have large action spaces. Similarly, KFC-variants also perform well on difficult data dis-

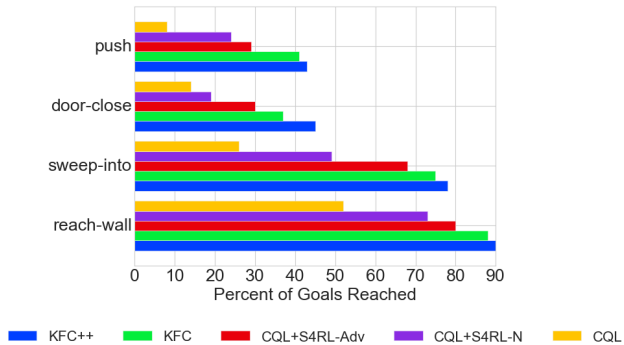


Figure 2. MetaWorld Environments

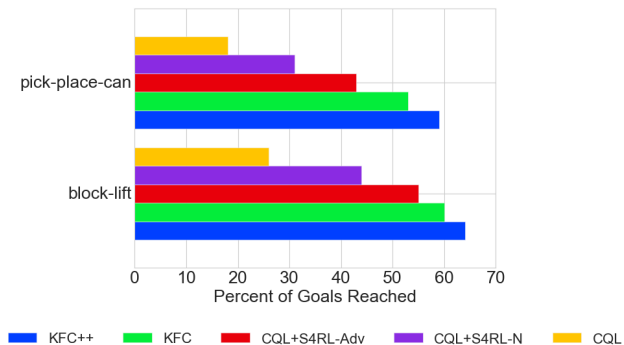


Figure 3. RoboSuite Environments

Figure 4. Results on challenging dexterous robotics environments using data collected using a similar strategy as S4RL (Sinha et al., 2021). We report the % of goals that the agent is able to reach during evaluation, where the goal is set by the environments. We see that KFC and KFC++ consistently outperforms both CQL and the two best-performing S4RL variants.

tributions such as “medium-replay”: which is collected by simply using all the data that the policy encountered while training base SAC policy, and “-human” which is collected using human demonstrations on robotic tasks which results in a non-Markovian behaviour policy (more details can be found in the D4RL manuscript (Fu et al., 2021)).

Furthermore, to our knowledge, the results for KFC++ are state-of-the-art in model-free policy learning from D4RL datasets for most environments and data distributions. We do note that S4RL outperforms KFC on the “-random” split of the data distributions, which is expected as KFC depends on learning a simple dynamics model of the data to use to guide the data augmentation strategy. Since the “-random” split consists of random actions in the environment, our simple model is unable to learn a useful dynamics model. For ablation studies see appendix D.

Table 1. We experiment with the full set of the D4RL tasks and report the mean normalized episodic returns over 5 random seeds using the same protocol as Fu et al. (2021). We compare against 3 competitive baselines including CQL and the two best-performing S4RL-data augmentation strategies. We see that KFC and KFC++ consistently outperforms the baselines. We use the baseline numbers reported in Sinha et al. (2021).

Domain	Task Name	CQL	S4RL-(N)	S4RL-(Adv)	KFC	KFC++
AntMaze	antmaze-umaze	74.0	91.3	94.1	96.9	99.8
	antmaze-umaze-diverse	84.0	87.8	88.0	91.2	91.1
	antmaze-medium-play	61.2	61.9	61.6	60.0	63.1
	antmaze-medium-diverse	53.7	78.1	82.3	87.1	90.5
	antmaze-large-play	15.8	24.4	25.1	24.8	25.6
	antmaze-large-diverse	14.9	27.0	26.2	33.1	34.0
Gym	cheetah-random	35.4	52.3	53.9	48.6	49.2
	cheetah-medium	44.4	48.8	48.6	55.9	59.1
	cheetah-medium-replay	42.0	51.4	51.7	58.1	58.9
	cheetah-medium-expert	62.4	79.0	78.1	79.9	79.8
	hopper-random	10.8	10.8	10.7	10.4	10.7
	hopper-medium	58.0	78.9	81.3	90.6	94.2
	hopper-medium-replay	29.5	35.4	36.8	48.6	49.0
	hopper-medium-expert	111.0	113.5	117.9	121.0	125.5
	walker-random	7.0	24.9	25.1	19.1	17.6
	walker-medium	79.2	93.6	93.1	102.1	108.0
	walker-medium-replay	21.1	30.3	35.0	48.0	46.1
	walker-medium-expert	98.7	112.2	107.1	114.0	115.3
Adroit	pen-human	37.5	44.4	51.2	61.3	60.0
	pen-cloned	39.2	57.1	58.2	71.3	68.4
	hammer-human	4.4	5.9	6.3	7.0	9.4
	hammer-cloned	2.1	2.7	2.9	3.0	4.2
	door-human	9.9	27.0	35.3	44.1	46.1
	door-cloned	0.4	2.1	0.8	3.6	5.6
	relocate-human	0.2	0.2	0.2	0.2	0.2
	relocate-cloned	-0.1	-0.1	-0.1	-0.1	-0.1
Franka	kitchen-complete	43.8	77.1	88.1	94.1	94.9
	kitchen-partial	49.8	74.8	83.6	92.3	95.9

4.2. Metaworld and Robosuite benchmarks

To further test the ability of KFC, we perform additional experiments on challenging robotic tasks. Following (Sinha et al., 2021), we perform additional experiments with 4 MetaWorld environments (Yu et al., 2019) and 2 RoboSuite environments (Zhu et al., 2020). We followed the same method to collect the data as described in Appendix F of S4RL (Sinha et al., 2021), and report the mean percent of goals reached, where the condition of reaching the goal is defined by the environment.

We report the results in Figure 4, where we see that by using KFC to guide the data augmentation strategy for a base CQL agent, we are able to learn an agent that performs significantly better. Furthermore, we see that for more challenging

tasks such as “push” and “door-close” in the MetaWorld, KFC++ outperforms the base CQL algorithm and the S4RL agent by a considerable margin. This further highlights the ability of KFC to guide the data augmentation strategy.

5. Related works

The use of data augmentation techniques in Q-learning has been discussed recently (Laskin et al., 2020a;b; Sinha et al., 2021). In particular, our work shares strong parallels with (Sinha et al., 2021). Our modification of the policy evaluation step of the CQL algorithm (Kumar et al., 2020) is analogous to the one in Sinha et al. (2021). However, the latter randomly augments the data while our augmentation framework is based on symmetry state shifts. Regarding the

connection to world models (Ha & Schmidhuber, 2018) a few comments in order. Here a VAE is used to decode the state information while a recurrent separate neural network predicts future states. Their latent representation is not of Koopman type. Also no symmetries and data-augmentations are derived.

Algebraic symmetries of the state-action space in Markov Decision Processes (MDP) originate (Balaraman & Andrew, 2004) and were discussed recently in the context of RL in (van der Pol et al., 2020). Their goal is to preserve the essential algebraic homomorphism symmetry structure of the original MDP while finding a more compact representation. The symmetry maps considered in our work are more general and are utilized in a different way. Symmetry-based representation learning (Higgins et al., 2018) refers to the study of symmetries of the environment manifested in the latent representation. The symmetries in our case are derived from the Koopman operator not the latent representation directly. In (Caselles-Dupré et al., 2019) the authors discuss representation learning of symmetries (Higgins et al., 2018) allowing for interactions with the environment. A Forward-VAE model which is similar to our Koopman-Forward VAE model is employed. We provide theoretical results on how to derive explicit symmetries of the dynamical systems as well as their utilisation for state-shifts.

In (Sinha et al., 2020) symmetries are used to extend the Koopman operator from a local to a global description. Neither action-equivariant dynamical control systems nor data augmentation is discussed. In (Salova et al., 2019) the imprint of known symmetries on the block-diagonal Koopman space representation for non-control dynamical systems is discussed. This is close to the spirit of disentanglement (Higgins et al., 2018). Our results are on control setups and deriving symmetries.

On another front, the application of Koopman theory in control or reinforcement learning has also been discussed recently. For example, (Li et al., 2020) propose to use compositional Koopman operators using graph neural networks to learn dynamics that can quickly adapt to new environments of unknown physical parameters and produce control signals to achieve a specified goal. (Kaiser et al., 2021) discuss the use of Koopman eigenfunction as a transformation of the state into a globally linear space where the classical control techniques is applicable. (Iwata & Kawahara, 2021) propose a framework for policy learning based on linear quadratic regulator in a space where the Koopman operator is defined. And, (Ohnishi et al., 2021) discuss a method of controlling nonlinear systems via the minimization of the Koopman spectrum cost: A cost over the Koopman operator of the controlled dynamics. However, to the best of our knowledge, this paper is the first to discuss Koopman latent space for data augmentation.

The study of Bisimulation metrics of action-state spaces has led to performance improvements of Q-learning algorithms (Gelada et al., 2019; Zhang et al., 2021). We do not alter the metric of the action-state space as well do not take the reward into consideration. It would be interesting to study the KFC-algorithm trained on a Bisimulation metric space.

6. Conclusions

In this work we have proposed a symmetry-based data augmentation technique derived from a Koopman latent space representation. It enables a principled extension of offline RL datasets describing dynamical systems. The approach is based on our theoretical results on symmetries of dynamical control systems and symmetry shifts of data. Both hold for systems with differentiable state transitions and with a Bilinearisation model for the Koopman operator. However, the empirical results show that the framework is successfully applicable beyond those limitations.

We empirically evaluated our method on several benchmark offline reinforcement learning tasks D4RL, Metaworld and Robosuite and find that by using our framework we consistently improve the state-of-the-art of model-free Q-learning algorithms. Our framework is naturally applicable to model-based approaches and thus such an extension of our study would be of great interest.

Acknowledgments

This work was supported by JSPS KAKENHI (Grant Number JP22H00516), and JST CREST (Grant Number JP-MJCR1913). And, we would like to thank Y. Nishimura for technical supports.

References

- Sylvester algorithm - scipy.org. URL https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.linalg.solve_sylvester.html.
- Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.
- Balaraman, R. and Andrew, G. B. Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes. In *International Conference on Knowledge Based Computer Systems*, 2004.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv: 1606.01540*, 2016.
- Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N. Modern Koopman theory for dynamical systems. *arXiv: 2102.12086*, 2021.
- Caselles-Dupré, H., Garcia Ortiz, M., and Filliat, D. Symmetry-based disentangled representation learning requires interaction with environments. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005.
- Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv: 2004.07219*, 2021.
- Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 09–15 Jun 2019.
- Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. DeepMDP: Learning continuous latent space models for representation learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/gelada19a.html>.
- Goswami, D. and Paley, D. A. Global bilinearization and controllability of control-affine nonlinear systems: A Koopman spectral approach. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 6107–6112, 2017.
- Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf>.
- Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 06–11 Aug 2017.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018.
- Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. Towards a definition of disentangled representations. *arXiv: 1812.02230*, 2018.
- Iwata, T. and Kawahara, Y. Controlling nonlinear dynamical systems with linear quadratic regulator-based policy networks in Koopman space. In *Proc. of the 60th IEEE Conf. on Decision and Control (CDC'21)*, pp. 5086–5091, 2021.
- Kaiser, E., Kutz, J. N., and Brunton, S. L. Data-driven discovery of Koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2:035023, 2021.
- Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Scalable deep reinforcement learning for vision-based robotic manipulation. In Billard, A., Dragan, A., Peters, J., and Morimoto, J. (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 29–31 Oct 2018.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv: 1312.6114*, 2013.
- Koopman, B. O. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of*

- Sciences of the United States of America*, 17(5):315–318, 1931.
- Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020.
- Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19884–19895. Curran Associates, Inc., 2020a.
- Laskin, M., Srinivas, A., and Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020b.
- Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.
- Li, Y., He, H., Wu, J., Katabi, D., and Torralba, A. Learning compositional Koopman operators for model-based control. In *Proc. of the 8th Int’l Conf. on Learning Representation (ICLR’20)*, 2020.
- Mauroy, A., Mezić, I., and Susuki, Y. *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer, 2020.
- Mezić, I. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41: 309–325, 2005.
- Morton, J., Witherden, F. D., and Kochenderfer, M. J. Deep variational Koopman models: Inferring Koopman observations for uncertainty-aware dynamics modeling and control. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI’19*. AAAI Press, 2019.
- Nair, V. and Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *Icml*, 2010.
- Ohnishi, M., Ishikawa, I., Lowrey, K., Ikeda, M., Kakade, S., and Kawahara, Y. Koopman spectrum nonlinear regulator and provably efficient online learning. *arXiv: 2106.15775*, 2021.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.
- Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv: 1709.10087*, 2017.
- Salova, A., Emenheiser, J., Rupe, A., Crutchfield, J. P., and D’Souza, R. M. Koopman operator and its approximations for systems with symmetries. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(9):093128, 2019. doi: 10.1063/1.5099091.
- Sinha, S., Nandanoori, S. P., and Yeung, E. Koopman operator methods for global phase space exploration of equivariant dynamical systems. *arXiv: 2003.04870*, 2020.
- Sinha, S., Mandlekar, A., and Garg, A. S4RL: Surprisingly Simple Self-Supervision for Offline Reinforcement Learning. In *Conference on Robot Learning*, 2021.
- Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., and Heess, N. dm-control: Software and tasks for continuous control. *arXiv: 2006.12983*, 2020.
- van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. Mdp homomorphic networks: Group symmetries in reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4199–4210. Curran Associates, Inc., 2020.
- Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv: 1911.11361*, 2019.
- Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv: 1910.10897*, 2019.
- Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=-2FCwDKRREu>.
- Zhu, Y., Wong, J., Mandlekar, A., and Martín-Martín, R. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv: 2009.12293*, 2020.

A. Koopman theory and Symmetries

This section provides some additional introductory material on Koopman theory as well as a more detailed presentation of our results on symmetries of dynamical systems.

In general, a symmetry group Σ of the state space may be any subgroup of the group of isometrics of the Euclidean space. We consider Euclidean state spaces and in particular, Σ invariant compact subsets thereof.

Definition A.1 (Equivariant Dynamical System). Consider the dynamical system $\dot{s} = f(s)$ and let Σ be a group acting on the state-space S . Then the system is called Σ -equivariant if $f(\sigma s) = \sigma f(s)$, for $s \in S$, $\delta\sigma \in \Sigma$. For a discrete time dynamical system $s_{t+1} = F(s_t)$ one defines equivariance analogously, namely if $F(\sigma s_t) = \sigma F(s_t)$, for $s_t \in S$, $\delta\sigma \in \Sigma$.

A system of ODEs may be represented by the set of its solutions which are locally connected by Lie Groups.

Definition A.2 (Symmetries of ODEs). A symmetry of a system of ODEs on a locally-smooth structure is a locally-defined diffeomorphism that maps the set of all solutions to itself.

Definition A.3 (Local Lie Group). A parametrized set of transformations $\sigma^\epsilon : S \rightarrow S$ with $s \mapsto \tilde{s}(s, \epsilon)$ for $\epsilon \in (\epsilon_{low}, \epsilon_{high})$ where $\epsilon_{low} < 0 < \epsilon_{high}$ is a one-parameter local Lie group if⁵

1. σ^0 is the identity map; for $\epsilon = 0$ such that $\tilde{s}(s, 0) = s$.
2. $\sigma^{\epsilon_1} \sigma^{\epsilon_2} = \sigma^{\epsilon_2} \sigma^{\epsilon_1} = \sigma^{\epsilon_1 + \epsilon_2}$ for every $j \in \mathbb{1}, j \in \mathbb{2} \quad \mathbb{1}$.
3. $\tilde{s}(s, \epsilon)$ admits a Taylor series expansion in ϵ ; in a neighbourhood of s determined by $\epsilon = 0$ as $\tilde{s}(s, \epsilon) = s + \epsilon \zeta(s) + O(\epsilon^2)$.

The Koopman operator of equivariant dynamical system is reviewed in the following .

Lemma A.4. The map $\sigma : \Sigma \rightarrow \mathcal{K}(S) \rightarrow \mathcal{K}(S)$ given by $(\sigma^{-1}g)(s) \mapsto g(\sigma^{-1}s)$ defines a group action on the Koopman space of observables $\mathcal{K}(S)$.

Theorem A.5. Let K be the Koopman operator associated with a Σ -equivariant system $s_{t+1} = F(s_t)$. Then

$$[\sigma^{-1}(Kg)](s) = [K(\sigma^{-1}g)](s) . \quad (20)$$

Theorem A.5 states that for a Σ -equivariant system any symmetry transformation commutes with the Koopman operator. For the proof see (Sinha et al., 2020).

Let us next turn to the case relevant for RL, namely control systems. In the remainder of this section we focus on dynamical systems given as in Eq. (6) and Eq. (9).

⁵The points (1) and (2) in definition 3.3 imply the existence of an inverse element $\sigma^{\epsilon^{-1}} = \sigma^{-\epsilon}$ for $j \in \mathbb{1}$. A local Lie group satisfies the group axioms for sufficiently small parameters values; it may not be a group on the entire set.

Definition A.6 (Action Equivariant Dynamical System).

Let $\bar{\Sigma}$ be a group acting on the state-action-space $S \times A$ of a general control system as in Eq. (6) such that it acts as the identity operation on A i.e. $\sigma^{-1}(s_t, a_t) = (\sigma^{-1}s_t, a_t)$, $\delta\sigma \in \bar{\Sigma}$. Then the system is called $\bar{\Sigma}$ -action-equivariant if $\delta\sigma \in \bar{\Sigma}$

$$F(\sigma^{-1}(s_t, a_t)) = \sigma^{-1}F(s_t, a_t), \text{ for } (s_t, a_t) \in S \times A . \quad (21)$$

Lemma A.7. The map $\sigma : \bar{\Sigma} \rightarrow \mathcal{K}(S \times A) \rightarrow \mathcal{K}(S \times A)$ given by $(\sigma^{-1}g)(s, a) \mapsto g(\sigma^{-1}(s, a))$ defines a group action on the Koopman space of observables $\mathcal{K}(S \times A)$.

Theorem A.8. A Koopman operator K of a $\bar{\Sigma}$ -action-equivariant system $s_{t+1} = F(s_t, a_t)$ satisfies

$$[\sigma^{-1}(Kg)](s_t, a_t) = [K(\sigma^{-1}g)](s_t, a_t) . \quad (22)$$

In particular, it is easy to see that the bilinearisation in Eq. (9) is Σ -action-equivariant if $f_i(\sigma^{-1}s) = \sigma^{-1}f_i(s)$, $\delta i = 0, \dots, m$. Let us thus turn our focus on the relevant case of a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ which admits a Koopman operator description as

$$g(s_{t+1}) = K(a_t)g(s_t), \text{ for } a_t \in A, \delta g \in \mathcal{K}(S), \quad (23)$$

where $fK(a)g_{a \in A}$ is a family of operators with analytical dependence on $a \in A$. Note that the bilinearisation in Eq. (9) is a special case of Eq. (13). Furthermore, let $fU(a)g_{a \in A}$ be a family of invertible operators s.t. $U(a) : \mathcal{K}(S) \rightarrow \mathcal{F}(S \times A)$ is a mapping to the (Banach) space of eigenfunctions $\mathcal{F}(S \times A)$ with eigenfunctions $\varphi(s, a) := U(a)g(s)$ which obeys $U(a)K(a)U(a)^{-1} = \Lambda(a)$, with $\Lambda(a)\varphi(s, a) = \lambda_\varphi(a)\varphi(s, a)$ and where $\lambda_\varphi(a) : A \rightarrow \mathbb{R}$. The existence of such operators on an infinite dimensional space requires the Koopman operator in Eq. (13) to be self-adjoint or a finite-dimensional (approximate) matrix representation to be diagonalizable.⁶

Lemma A.9. The map $\hat{\phi} : \bar{\Sigma} \rightarrow \mathcal{K}(S) \rightarrow fU(a)g_{a \in A} \rightarrow \mathcal{K}(S)$ given by

$$(\sigma_a \hat{g})(s) \mapsto \left(U^{-1}(a)(\sigma^{-1}(U(a)g)) \right)(s) = g(\sigma^{-1}(s, a)), \quad (24)$$

defines a group action on the Koopman space of observables $\mathcal{K}(S)$. Where σ is defined analog to Lemma A.7 but acting on $\mathcal{F}(S \times A)$ instead of $\mathcal{K}(S \times A)$ by $(\sigma^{-1}\varphi)(s, a) \mapsto \varphi(\sigma^{-1}(s, a))$. We refer to a control system in Eq. (23) admitting a $\hat{\phi}$ -symmetry as $\hat{\Sigma}$ -action-equivariant.

Theorem A.10. Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$ -action-equivariant control system with a symmetry action as in

⁶See Appendix A for the details. We found that, in practice the Koopman operator learned by the neural nets is diagonalizable almost everywhere.

Lemma A.9 which furthermore admits a Koopman operator representation as

$$g(s_{t+1}) = K(a_t)g(s_t), \text{ for } a_t \in A, \forall g \in \mathcal{K}(S). \quad (25)$$

Then

$$[\sigma_{a_t}^\epsilon(K(a_t)g)](s_t) = [K(a_t)(\sigma_{a_t}^\epsilon g)](s_t). \quad (26)$$

Moreover, a control system obeying equations 25 and 26 is $\hat{\Sigma}$ -action-equivariant locally if g^{-1} exists for a neighborhood of s_t , i.e. then $\sigma^{-1} \tilde{F}(s_t, a_t) = \tilde{F}(\sigma^{-1}(s_t, a_t))$.

To allow an easier transition to the empirical section we introduce the notation $E : S \rightarrow \mathcal{K}(S)$ and $D : \mathcal{K}(S) \rightarrow S$ denote the C^1 -differentiable encoder and decoder to and from the finite-dimensional Koopman space approximation, respectively, i.e. $E \circ D = D \circ E = id$. Data-points may be shifted by symmetry transformations of solutions of ODEs as discussed next.

Theorem A.11. Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a control system as in Eq. (25) and σ_{a_t} an operator obeying Eq. (26). Then $\sigma_{a_t}^\epsilon : (s_t, s_{t+1}, a_t) \rightarrow (\tilde{s}_t, \tilde{s}_{t+1}, a_t)$ with

$$\begin{aligned} \tilde{s}_t &= D\left((1 + \epsilon \sigma_{a_t})^\wedge E(s_t)\right), \\ \tilde{s}_{t+1} &= D\left((1 + \epsilon \sigma_{a_t})^\wedge E(s_{t+1})\right) \end{aligned} \quad (27)$$

is a one-parameter local Lie symmetry group of ODEs. In other words one can use a symmetry transformation to shift both s_t as well as s_{t+1} such that $\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t)$.

No assumptions on the Koopman operator are imposed. Moreover, note that the equivalent theorem holds when $\epsilon \sigma_{a_t} \in \sum_{I=1}^N \epsilon_I \sigma_{a_t}^I$ to be a local N-parameter Lie group.

Additional Remarks. Let us re-evaluate some information given on the operators used previously. Let $\tilde{F}(a)g_{a \in A}$ be a family of invertible operators s.t. $U(a) : \mathcal{K}(S) \rightarrow \mathcal{F}(S \rightarrow A)$ is a mapping to the (Banach) space of eigenfunctions $\varphi(s, a) := U(a)g(s) \in \mathcal{F}(S \rightarrow A)$. Which moreover obeys

$$\begin{aligned} U(a)K(a)U(a)^{-1} &= \Lambda(a), \text{ with} \\ \Lambda(a)\varphi(s, a) &= \lambda_\varphi(a)\varphi(s, a), \end{aligned} \quad (28)$$

and where $\lambda_\varphi(a) : A \rightarrow \mathbb{R}$. The existence of such operators puts further restriction on the Koopman operator in Eq. (13). However, as our algorithm employs the finite-dimensional approximation of the Koopman operator i.e. its matrix representation this amounts simple for $K(a)$ to be diagonalizable and $U(a)$ is the matrix containing its eigenvectors as columns. To evolve a better understanding on the required criteria for the infinite-dimensional case we employ an alternative formulation of the so called spectral theorem below.

Theorem A.12 (Spectral Theorem). Let K be a bounded self-adjoint operator on a Hilbert space H . Then there is a measure space (S, Σ, μ) and a real-valued essentially bounded measurable function λ on S and a unitary operator $U : H \rightarrow L^2_\mu(S)$ i.e. $U^{-1}KU = id$ such that

$$U^{-1}KU = K, \text{ with } [\Lambda\varphi](s) = \lambda(s)\varphi(s). \quad (29)$$

In other words every bounded self-adjoint operator is unitarily equivalent to a multiplication operator. In contrast to the finite-dimensional case we need to slightly alter our criteria to $U(a)K(a)U(a)^{-1} = \Lambda(a)$, with $\Lambda(a)\varphi(s, a) = \lambda_\varphi(a, s)\varphi(s, a)$ and where $\lambda_\varphi(s, a) : S \rightarrow \mathbb{R}$. Concludingly, a sufficient condition for our criteria to hold in terms of operators on Hilbert spaces is that the Koopman operator $K(a)$ is self-adjoint i.e. that

$$K(a) = K(a)^* \quad (30)$$

B. Proofs

In this section we provide the proofs of the theoretical results of Section 3.

Proof of Lemma A.4 and Theorem A.5 The proofs of the lemma as well as the theorem can be found in (Sinha et al., 2020).

Proof of Lemma 3.5: We aim to show that $\text{map}^{-1} : \bar{\Sigma} \times \mathcal{K}(S \rightarrow A) \rightarrow \mathcal{K}(S \rightarrow A)$ given by $(\sigma^{-1}g)(s, a) = g(\sigma^{-1}(s, a))$ defines a group action on the Koopman space of observables $\mathcal{K}(S \rightarrow A)$, where $\bar{\Sigma}$ defines the symmetry group of definition 3.4. Firstly, let $g \in \mathcal{K}(S \rightarrow A)$ and $\sigma_0 \in \bar{\Sigma}$ be the identity element, then we see that it provides existence of an identity element of $\bar{\Sigma}$ by

$$(\sigma_0^{-1}g)(s, a) = g(\sigma_0^{-1}(s, a)) = g(s, a).$$

Secondly, let $\sigma_1, \sigma_2 \in \bar{\Sigma}$ and $\bar{\Sigma}$ denoting the group operation i.e. $\sigma_1 \circ \sigma_2 = \sigma_3 \in \bar{\Sigma}$. Then

$$\begin{aligned} (\sigma_2^{-1}(\sigma_1^{-1}g))(s, a) &= \sigma_2^{-1}\left(g(\sigma_1^{-1}(s, a))\right) \\ &= g(\sigma_2^{-1}(\sigma_1^{-1}(s, a))) \\ &= g((\sigma_2^{-1} \circ \sigma_1^{-1})(s, a)) \\ &\stackrel{(I)}{=} g((\sigma_1 \circ \sigma_2)^{-1}(s, a)) \\ &= \left((\sigma_1 \circ \sigma_2)^{-1}g\right)(s, a) \\ &= (\sigma_3^{-1}g)(s, a), \end{aligned} \quad (31)$$

where in (I) we have used the invertibility of the group property of $\bar{\Sigma}$. Lastly it follows analogously that for $\sigma, \sigma^{-1} \in \bar{\Sigma}$

that

$$\begin{aligned} (\sigma^-(\sigma^{-1}g))(s, a) &= ((\sigma^{-1} \sigma)^-g)(s, a) \quad (32) \\ &= (\sigma_0^-g)(s, a) = g(s, a) . \end{aligned}$$

Thus the existence of an inverse is established which concludes to show the group property of $\bar{\cdot}$.

Proof of Theorem 3.6: We aim to show that with \mathcal{K} being the Koopman operator associated with a $\bar{\Sigma}$ -action-equivariant system $s_{t+1} = F(s_t, a_t)$. Then

$$[\sigma^-(\mathcal{K}g)](s, a) = [\mathcal{K}(\sigma^-g)](s, a) .$$

First of all note that by the definition of the Koopman operator of non-affine control systems it obeys

$$[\mathcal{K}g](s_t, a_t) = g(F(s_t, a_t), a_{t+1}) .$$

Using the latter one thus infers that

$$\begin{aligned} [\sigma^-(\mathcal{K}g)](s_t, a_t) &= \sigma^-g(F(s_t, a_t), a_{t+1}) \quad (33) \\ &= g(\sigma^{-1} F(s_t, a_t), a_{t+1}) \\ &\stackrel{(I)}{=} g(F(\sigma^{-1} s_t, a_t), a_{t+1}) , \end{aligned}$$

where in (I) we have used that it is a $\bar{\Sigma}$ -action-equivariant system. Moreover, one finds that

$$\begin{aligned} g(F(\sigma^{-1} s_t, a_t), a_{t+1}) &= \mathcal{K}g(\sigma^{-1} s_t, a_t) \quad (34) \\ &= [\mathcal{K}(\sigma^-g)](s_t, a_t) , \end{aligned}$$

which concludes the proof.

Proof of Lemma A.9: We aim to show that the map $\hat{\phi} : \bar{\Sigma} \times \mathcal{K}(S) \rightarrow \mathcal{K}(S)$ given by

$$(\sigma_a \hat{g})(s) \stackrel{!}{=} (U^{-1}(a)(\sigma^-(U(a)g)))(s) = g(\sigma^{-1} s) ,$$

defines a group action on the Koopman space of observables $\mathcal{K}(S)$. Where $\bar{\cdot}$ is defined analog to Lemma 3.5 but acting on $F(S \rightarrow A)$ instead of $\mathcal{K}(S \rightarrow A)$ by $(\sigma^- \varphi)(s, a) \stackrel{!}{=} \varphi(\sigma^{-1} s, a)$. First of all note that

$$\begin{aligned} (U^{-1}(a)(\sigma^-(U(a)g)))(s) &= U^{-1}(a)(\sigma^- \varphi(s, a)) \\ &= U^{-1}(a)\varphi(\sigma^{-1} s, a) \\ &= g(\sigma^{-1} s) \quad (35) \end{aligned}$$

We proceed analogously as in the proof of Lemma 3.5 above. Firstly, let $g \in \mathcal{K}(S)$ and $\sigma_0 \in \bar{\Sigma}$ be the identity element

then one infers from Eq. (35) that it provides existence of an identity element of $\hat{\phi}$ by

$$(\sigma_{a,0} \hat{g})(s) = g(\sigma_0^{-1} s) = g(s) ,$$

where we have used the notation for $i = 0, \dots$

$$(\sigma_{a,i} \hat{g})(s) = (U^{-1}(a)(\sigma_i^-(U(a)g)))(s) , .$$

Secondly, let $\sigma_1, \sigma_2 \in \bar{\Sigma}$ and $\bar{\cdot}$ denoting the group operation i.e. $\sigma_1 \sigma_2 = \sigma_3 \in \bar{\Sigma}$. Then

$$\begin{aligned} (\sigma_{a,2} \hat{g})(\sigma_{a,1} \hat{g})(s) &= \sigma_{a,2} \hat{g}(g(\sigma_1^{-1} s)) \quad (36) \\ &= g(\sigma_2^{-1} (\sigma_1^{-1} s)) \\ &= g((\sigma_2^{-1} \sigma_1^{-1}) s) \\ &\stackrel{(I)}{=} g((\sigma_1 \sigma_2)^{-1} s) \\ &= g(\sigma_3^{-1} s) \\ &= (\sigma_{a,3} \hat{g})(s) , \end{aligned}$$

where in (I) we have used the invertibility of the group property of $\bar{\Sigma}$. Lastly, it follows analogously that for $\sigma, \sigma^{-1} \in \bar{\Sigma}$ that

$$\begin{aligned} (\sigma \hat{g})(\sigma^{-1} \hat{g})(s) &= ((\sigma^{-1} \sigma) \hat{g})(s) \quad (37) \\ &= (\sigma_0 \hat{g})(s) = g(s) . \end{aligned}$$

Thus the existence of an inverse is established which concludes to show the group property of $\hat{\phi}$.

Proof of Theorem 3.8: We aim to show twofold.

\Rightarrow : Firstly, that with $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$ -action-equivariant control system with a symmetry action as in Lemma A.9 which furthermore admits a Koopman operator representation as

$$g(s_{t+1}) = K(a_t)g(s_t) , \text{ for } a_t \in A , \delta g \in \mathcal{K}(S) .$$

Then

$$[\sigma_{a_t} \hat{g}(K(a_t)g)](s_t) = [K(a_t)(\sigma_{a_t} \hat{g})](s_t) .$$

\Leftarrow : Secondly, the converse. Namely, if a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ obeys Eqs. (14) and (15), then it is $\hat{\Sigma}$ -action-equivariant, i.e. $\sigma \tilde{F}(s_t, a_t) = \tilde{F}(\sigma^{-1}(s_t, a_t))$. For notational simplicity we drop the subscripts referring to time in the remainder of this proof i.e. $s_t \stackrel{!}{=} s$ and $a_t \stackrel{!}{=} a$.

Let us start with the first implication i.e. \Rightarrow .

First of all note that by the definition of the Koopman operator one has

$$[K(a)g](s) = g(\tilde{F}(s, a)) .$$

Using the latter one infers that

$$\begin{aligned} [\sigma_a \hat{\cdot} (K(a)g)](s) &= \sigma_a \hat{\cdot} g(\tilde{F}(s, a)) , \\ &= U^{-1}(a) \left(\sigma^{-}(U(a)g(\tilde{F}(s, a))) \right) , \\ &= U^{-1}(a) \left(\sigma^{-}\varphi(\tilde{F}(s, a), a) \right) , \\ &= U^{-1}(a) \left(\varphi(\sigma^{-1} \tilde{F}(s, a), a) \right) , \\ &= U^{-1}(a) \left(\varphi(\tilde{F}(\sigma^{-1} s, a), a) \right) , \\ &= g(\tilde{F}(\sigma^{-1} s, a)) . \end{aligned}$$

Moreover, one derives that

$$\begin{aligned} g(\tilde{F}(\sigma^{-1} s, a)) &= K(a)g(\sigma^{-1} s) , \\ &= K(a) \underbrace{U^{-1}(a)U(a)}_{=id} g(\sigma^{-1} s) , \\ &= K(a)U^{-1}(a)\varphi(\sigma^{-1} s, a) , \\ &= K(a) \left(U^{-1}(a) (\sigma^{-}\varphi(s, a)) \right) , \\ &= K(a) \left(U^{-1}(a) (\sigma^{-}(U(a)g(s))) \right) , \\ &= \left[K(a) (\sigma_a \hat{\cdot} g) \right](s) , \end{aligned}$$

which concludes the proof of the first part of the theorem. Let us next show the converse implication i.e. (\Leftarrow) . For this case it is practical to use the discrete time-system notation explicitly. Let $\sigma \in \Sigma$ be a symmetry of the state-space and be $\tilde{s}_t = \sigma^{-1} s_t$ and $\tilde{s}_{t+1} = \sigma^{-1} s_{t+1}$ the σ -shifted states. Then

$$\begin{aligned} g(s_{t+1}) &= g(\tilde{F}(s_t, a_t)) , \\ &= K(a_t)g(s_t) , \\ &= K(a_t)g(\sigma^{-1} \tilde{s}_t) , \\ &= K(a_t) \left(\sigma_a \hat{\cdot} g(\tilde{s}_t) \right) , \\ &= \left[K(a_t) (\sigma_a \hat{\cdot} g) \right](\tilde{s}_t) , \\ &\stackrel{(J)}{=} \left[\sigma_a \hat{\cdot} (K(a_t)g) \right](\tilde{s}_t) , \end{aligned}$$

where in (I) we have used that the symmetry operator com-

mutates with the Koopman operator. Thus in particular

$$\begin{aligned} \sigma_a \hat{\cdot} g(s_{t+1}) &= \left[\sigma_a \hat{\cdot} (\sigma_a \hat{\cdot} (K(a_t)g)) \right](\tilde{s}_t) \\ &\stackrel{\text{Lemma A.9}}{=} \left[K(a_t)g \right](\tilde{s}_t) , \end{aligned} \quad (38)$$

Moreover, one finds that

$$\begin{aligned} \sigma_a \hat{\cdot} g(s_{t+1}) &= g \left((\sigma^{-1})^{-1} s_{t+1} \right) \\ &= g(\sigma^{-1} s_{t+1}) = g(\tilde{s}_{t+1}) , \end{aligned} \quad (39)$$

from which one concludes that

$$g(\tilde{s}_{t+1}) = [K(a_t)g](\tilde{s}_t) = g(\tilde{F}(\tilde{s}_t, a_t)) .$$

Finally, we use the invertibility of the Koopman space observables i.e. $g^{-1}(g(s)) = s$ to infer

$$\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) .$$

However, in general s the Koopman space observables are not invertible globally. The "inverse function theorem" guarantees the existence of a local inverse if $g(s)$ is C^1 differentiable for maps between manifolds of equal dimensions; see any classical reference/text-book on differential calculus. However, we assume inevitability locally.

$$\begin{aligned} \tilde{s}_{t+1} = \sigma^{-1} s_{t+1} & \quad \tilde{F}(\tilde{s}_t, a_t) = \sigma^{-1} \tilde{F}(s_t, a_t) \quad (40) \\ & \quad \tilde{F}(\sigma^{-1} s_t, a_t) = \sigma^{-1} \tilde{F}(s_t, a_t) , \end{aligned}$$

which at last concludes our proof of the second part of the theorem.

Extension of Theorem 3.9: Moreover, one may account for practical limitations i.e. an error by the assumption $[\sigma_{a_t}^\epsilon, K(a_t)] = \epsilon_a \mathbb{1}$. One then finds that $\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) + O(\epsilon_a)$. Thus the error becomes suppressed when $\epsilon_a \rightarrow 0$. The error may be due to practical limitations of capturing the true dynamics as well as the symmetry map.

Proof of Theorem 3.9: We will show here theorem 3.9 as well as the extension mentioned in the paragraph prior to this proof. Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$ -action-equivariant control system as in Theorem 3.8. For symmetry maps

$$\tilde{s}_t = D \left((1 + \epsilon \sigma_{a_t}) \hat{\cdot} E(s_t) \right) , \quad (41)$$

$$\tilde{s}_{t+1} = D \left((1 + \epsilon \sigma_{a_t}) \hat{\cdot} E(s_{t+1}) \right) .$$

we aim to show that one can use a symmetry transformation to shift both s_t as well as s_{t+1}

$$\sigma_{a_t}^\epsilon : (s_t, s_{t+1}, a_t) \mapsto (\tilde{s}_t, \tilde{s}_{t+1}, a_t) \quad (42)$$

such that

$$\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t). \quad (43)$$

From Eq. (41) and the definition of the Koopman operator one infers that⁷

$$\tilde{s}_{t+1} = D\left(\sigma_{a_t} \hat{E}(s_{t+1})\right) = D\left(\sigma_{a_t} \hat{K}(a_t)E(s_t)\right). \quad (44)$$

By using that $[\sigma_{a_t}, K(a_t)] = 0$ and Eq. (41) one finds that

$$\begin{aligned} \tilde{s}_{t+1} &= D\left(K(a_t)\sigma_{a_t} \hat{E}(s_t)\right) \quad (45) \\ &= D\left(K(a_t)\underbrace{E(D(\sigma_{a_t} \hat{E}(s_t)))}_{=1}\right) \\ &= D\left(K(a_t)E(\tilde{s}_t)\right), \end{aligned}$$

which concludes the proof of the first part of the theorem with $\sigma_{a_t} \hat{E}(s_t) = (1 + \epsilon \sigma_{a_t})E(s_t)$.

Local diffeomorphism. Per definition the maps D, E are differentiable and invertible which implies that they provide a local diffeomorphism from and to Koopman-space. Also the linear map i.e. a matrix multiplication by $(1 + \epsilon \sigma_{a_t})$ is a diffeomorphism. Thus the symmetry map $\sigma_{a_t}^\epsilon$ i.e. Eq. (16) constitutes a local diffeomorphism. The above proof culminating in Eq. (41) implies that we have a local diffeomorphism mapping solutions of the ODEs to solutions, thus a symmetry of the system of ODEs.

Limitations. However D, E to be invertible is a strong assumption as it implies that the Koopman space approximation admit the same dimension as the state space. Note that however in practice as we only require D, E *id* one may choose other Koopman space dimensions.

Local Lie group. What is left to show is that the symmetry of the system of ODEs locally is a Lie group. In definition 3.3 we need to show points (1)-(3).

1. For $\epsilon = 0$ one finds that $\sigma_{a_t}^0$ is the identity map i.e for such that

$$\tilde{s}_t(s_t, 0) = D((1 + 0 \sigma_{a_t})E(s_t)) = D(E(s_t)) = s_t \quad (46)$$

- 2.

$$\begin{aligned} \sigma^{\epsilon_1} \sigma^{\epsilon_2} &= D((1 + \epsilon_1 \sigma_{a_t})(1 + \epsilon_2 \sigma_{a_t})E(s_t)) \quad (47) \\ &= D((1 + (\epsilon_1 + \epsilon_2)\sigma_{a_t} + O(\epsilon_1 \epsilon_2))E(s_t)) \\ &= \sigma^{\epsilon_1 + \epsilon_2} \end{aligned}$$

for every $j \in \{1, 2\}, j \in \{1, 2\}$ 1.

3. $\tilde{s}_t(s_t, \epsilon)$ admits a Taylor series expansion in ϵ , i.e. in a neighbourhood of s determined by $\epsilon = 0$. We may

⁷For notational simplicity we study the general case $(1 + \epsilon \sigma_{a_t}) \hat{E}(s_t)$.

Taylor expand D around the point $E(s_t)$ g as

$$\tilde{s}_t(s_t, \epsilon) = D(E(s_t)) + \epsilon \sum_{I=1}^N (\sigma_{a_t} E(s_t))_I \frac{\partial D}{\partial g_I} + O(\epsilon^2) \quad (48)$$

thus

$$\tilde{s}_t(s_t, \epsilon) = s_t + \epsilon \zeta(s_t) + O(\epsilon^2) \quad (49)$$

$$\text{for } \zeta(s_t) = \sum_{I=1}^N (\sigma_{a_t} E(s_t))_I \frac{\partial D}{\partial g_I}.$$

4. The existence of an inverse element $(\sigma_{a_t}^\epsilon)^{-1} = \sigma_{a_t}^{-\epsilon}$ follows from (1) and (2).

Numerical errors. Let us next turn to the second part of the theorem incorporating for practical numerical errors. We aim to show that under the assumption $[\sigma_{a_t}^\epsilon, K(a_t)] = \epsilon_a 1$ one finds $\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) + O(\epsilon_a)$. Note that $\sigma_{a_t} \hat{E}(s_t) = (1 + \epsilon \sigma_{a_t})E(s_t)$ admits an expansion in the parameter ϵ . For $\epsilon \ll 1$ one can Taylor expand the differentiable functions D, E to find that

$$\tilde{s}_t = s_t + \delta(\epsilon), \quad \text{with } \delta(\epsilon) = \epsilon \sum_I g_I^\sigma \frac{\partial D}{\partial g_I} \Big|_{E(s_t)} + O(\epsilon^2), \quad (50)$$

where $g^\sigma = \sigma_{a_t} \hat{E}(s_t)$ and $g = E(s_t)$, and with the index $I = 1, \dots, N$. The analog expression holds for $t + 1$ i.e. $\tilde{s}_{t+1} = s_{t+1} + O(\epsilon)$. If $a_t = a_t(s_t)$ is differentiable function of the states⁸ one may Taylor expand the Koopman operator using Eq. (50) as⁹

$$\begin{aligned} K(a_t(\tilde{s}_t)) &= K(a_t(s_t)) + \sum_i \sum_n \delta_n(\epsilon) a_{in}^0(s_t) K_i \\ &= K(a_t(s_t)) + \epsilon \Delta + O(\epsilon^2). \quad (51) \end{aligned}$$

where and $a_{in}^0(s_t) = \frac{\partial a_t}{\partial s_{t,n}} j_{s_t}$ and $n = 1, \dots, \dim(s_t)$ and with

$$\Delta = \sum_i \sum_n \sum_I g_I^\sigma \frac{\partial D_n}{\partial g_I} \Big|_{E(s_t)} a_{in}^0(s_t) K_i. \quad (52)$$

However, by the definition of the symmetry map $a_t(\tilde{s}_t) = a_t(s_t)$, thus the Koopman operators in Eq. (51) match $\mathcal{B}\epsilon$. By using the assumption on the error one infers from Eq. (44)

⁸Although neural networks modeling the policies modeling the data-distribution contain non-differentiable components they are differentiable almost everywhere.

⁹The reader may wonder about the explicit use of the index $\delta_n(\epsilon)$ expressing the dimension of the state. it is necessary here although we have simply used s_t as an implicit vector without explicitly specifying any components throughout the work.

that

$$\begin{aligned}
 \tilde{s}_{t+1} &= D\left((1 + \epsilon\sigma_{a_t}) \hat{K}(a_t) E(s_t)\right) \\
 &= D\left((K(a_t) + \epsilon_a \mathbf{1}) (1 + \epsilon\sigma_{a_t}) \hat{E}(s_t)\right) \\
 &= D\left(K(a_t)(1 + \epsilon\sigma_{a_t}) \hat{E}(s_t) + \epsilon_a E(s_t) + O(\epsilon_a \epsilon)\right) \\
 &= D\left(K(a_t) E(\tilde{s}_t) + \epsilon_a E(s_t) + O(\epsilon_a \epsilon)\right),
 \end{aligned} \tag{53}$$

One may Taylor expand D by making use of $\epsilon_a \ll 1$ as

$$\begin{aligned}
 \tilde{s}_{t+1} &= D(K(a_t) E(\tilde{s}_t)) + \epsilon_a \sum_I E_I(s_t) \frac{\partial D}{\partial g_I} \Big|_{K(a_t) E(\tilde{s}_t)} \\
 &\quad + O(\epsilon_a \epsilon) + O(\epsilon_a^2). \tag{54}
 \end{aligned}$$

Thus under the assumption that the numerical error is given by a violation of the commutation relation one finds $\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) + O(\epsilon_a)$ where

$$\epsilon_a \sum_I E_I(s_t) \frac{\partial D}{\partial g_I} \Big|_{K(a_t) E(\tilde{s}_t)} = O(\epsilon_a). \tag{55}$$

By comparison with Eq. (50) one infers that the ϵ -expansion is a good approximation to the real dynamic if $\epsilon_a \ll \epsilon$. This concludes the proof of theorem 3.9. Let us note that the proof for state shifts arising from a sum of symmetry transformations as

$$\begin{aligned}
 \tilde{s}_t &= D\left(\left(1 + \sum_{I=1}^N \epsilon_I \sigma_{a_t}^I\right) \hat{E}(s_t)\right), \tag{56} \\
 \tilde{s}_{t+1} &= D\left(\left(1 + \sum_{I=1}^N \epsilon_I \sigma_{a_t}^I\right) \hat{E}(s_{t+1})\right).
 \end{aligned}$$

is completely analogous, and an analogous theorem holds. We choose the number of symmetries to run over the dimension of the Koopman latent space N as it is relevant for the **KFC++** setup.

C. Implementation details

The Koopman forward model The KFC algorithm requires pre-training of a Koopman forward model $F : S \rightarrow A \times S$ as

$$F^c(s_t, a_t) = \begin{cases} \mathbf{c} = \mathbf{0} & \text{VAE: } D(E(s_t)) = s_t \\ \mathbf{c} = \mathbf{1} & \text{forward prediction:} \\ & D\left(\left(K_0 + \sum_{i=1}^m K_i a_{t,i}\right) E(s_t)\right) = s_{t+1} \end{cases} \tag{57}$$

where both of E and D are approximated by Multi-Layer-Perceptrons (MLP's) and the bilinear Koopman-space operator approximation are implemented by a single fully

connected layer for $K_{i=0,\dots,m}$, respectively. The model is trained on batches of the offline data-set tuples (s_{t+1}, s_t, a_t) and optimized via the loss-function

$$\bar{l}_2(F^1(s_t, a_t), s_{t+1}) + \gamma_2 \bar{l}_2(F^0(s_t + s_\epsilon, a_t), s_t + s_\epsilon), \tag{58}$$

where s_ϵ is a normally distributed random state vector shift with zero mean and a standard deviation of $6 \cdot 10^{-2}$ where the \bar{l}_2 -loss is the Huberloss. For training the Koopman forward model in Eq. (17) we split the dataset in a randomly selected training and validation set with ratios 70%/30%. We then loop over the dataset to generate the required symmetry transformations and augment the replay buffer dataset tuple $(s_t, s_{t+1}, a_t, r_t, \sigma_{a_t})$ according to Eq. (18) as

$$\text{KFC: } (s_t, s_{t+1}, a_t, r_t, \sigma_{a_t}) \tag{59}$$

$$\text{KFC++: } (s_t, s_{t+1}, a_t, r_t, U(a_t), U(a_t)^{-1}).$$

It is more economic to compute the symmetry transformation as a pre-processing step rather than at runtime of the RL algorithm.

Let us reemphasize a couple of points addressed already in the main text to make this section self-contained. Note that we have built upon the implementation of CQL (Kumar et al., 2020), which is based on SAC (Haarnoja et al., 2018). **We also use the same hyperparameters for all the experiments as presented in the CQL and S4RL papers for baselines.**

Koopman forward model: For the encoder as well as the decoder we choose a three layer MLP architecture, with hidden-dimensions $[512, 512, N]$ and $[512, 512, m]$, respectively. Where $N = 32$ is the Koopman latent space dimension and m is the state space dimensions following our notation from section 2. The activation functions are ReLU (Nair & Hinton, 2010). We use an ADAM optimizer with learning rate $3 \cdot 10^{-4}$. We train the model for 75 epochs. The batch size is chosen to be 256. Lastly, we employ a random normally distributed state-augmentation for the VAE training i.e. $s \rightarrow s + s_\epsilon$ with $s_\epsilon \sim \mathcal{N}(0, 1 \cdot 10^{-2})$.

Hyper-parameters KFC: We take over the hyperparameter settings from the CQL paper (Kumar et al., 2020) except for the fact that we do not use automatic entropy tuning of the policy optimisation step Eq. (3) but instead a fixed value of $\alpha = 0.2$. The remaining hyper-parameters of the conservative Q-learning algorithm are as follows: $\gamma = 0.99$ and $\tau = 5 \cdot 10^{-3}$ for the discount factor and target smoothing coefficient, respectively. Moreover, the policy learning rate is $1 \cdot 10^{-4}$ and the value function learning rate is $3 \cdot 10^{-4}$ for the ADAM optimizers. We enable Lagrange training for $\tilde{\alpha}$ with threshold of 10.0 and learning rate of $3 \cdot 10^{-4}$ where the optimisation is done by an ADAM optimizer. The minimum Q-weight is set to 10.0 and the number of random samples of the Q-minimizer is 10. For

more specifics on these quantities see (Kumar et al., 2020). The model architectures of the policy as well as the value function are three layer MLP’s with hidden dimension 256 and *ReLU*-activation. The batch size is chosen to be 256. Moreover, the algorithm performs behavioral cloning for the first $40k$ training-steps, i.e. time-steps in an online RL notation.

The KFC specific choices are as follows: the split of random to Koopman-symmetry state shifts is 20/80 i.e. $p_K = 80\%$, whereas the option for the symmetry-type is either "Eigenspace" or "Sylvester" referring to case (I) and (II) of Eq. (18), respectively. Regarding the random variables in the symmetry transformation generation process, we choose $\epsilon_i \sim \mathcal{N}(0, 1 \cdot 10^{-4})$ for case (II) and $\epsilon \sim \mathcal{N}(0, 5 \cdot 10^{-5})$ for case (I) in Eq. (18) after normalizing σ_a by its mean. While for the random shift we use $\tilde{\epsilon} \sim \mathcal{N}(0, 3 \cdot 10^{-3})$ which is the hyper-parameter used in (Sinha et al., 2021).

Computational setup: We performed the empirical experiments on a system with PyTorch 1.9.0a (Paszke et al., 2019) The hardware was as follows: NVIDIA DGX-2 with 16 V100 GPUs and 96 cores of Intel(R) Xeon(R) Platinum 8168 CPUs and NVIDIA DGX-1 with 8 A100 GPUs with 80 cores of Intel(R) Xeon(R) E5-2698 v4 CPUs. The models are trained on a single V100 or A100 GPU.

D. Ablation study

D.1. Training on non contact events

To test the theoretical limitations in a practical setting one interesting experiment is to use the S4RL- \mathcal{N} augmentation scheme when the current state is a “contact” state and to choose KFC++ otherwise. We define a “contact” state as one where the agent makes contact with another surface; in practice we looked at the state dimensions which characterize the velocity of the agent parts, and if there is a sign difference between s_t and s_{t+1} , then contact with a surface must have been made. The details regarding which dimension maps to agent limb velocities is available in the official implementation of the Open AI Gym suite (Brockman et al., 2016).

Note that the theoretical limitations are twofold, firstly the theorems 3.8 and 3.9 only hold for dynamical systems with differentiable state-transitions; secondly, we employ a Bilinearisation Ansatz for the Koopman operator. Contact events lead to non-continuous state transitions and moreover would require incorporating for external forces in our Koopman description, thus both theoretical assumptions are practically violated in our main empirical evaluations of KFC and KFC++. By performing this ablation study which is more suitable for our Ansatz, i.e. the KFC++ part is only trained on "differentiable" trajectories without contact events. In other words, the Koopman description is unlikely

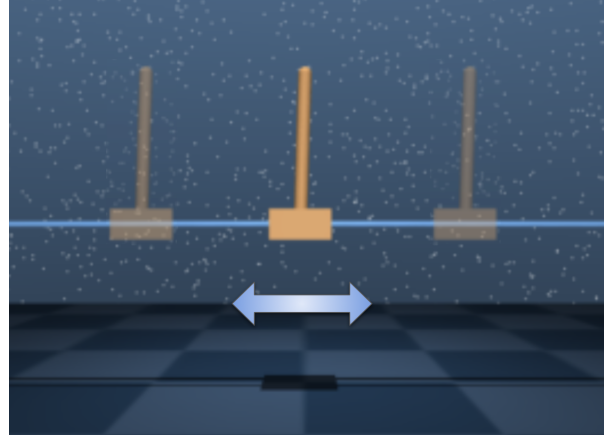


Figure 5. Depiction of translation symmetry of the cartpole environment in the Deepmind control suite (Tassa et al., 2020). The dynamics is invariant of the position along the horizontal bar, i.e. under shifts to the left and right symbolized by the arrow.

to give good information regarding s_{t+1} if there is a “contact event”, we simply use S4RL- \mathcal{N} when training the policies, which we expect to give slightly better performance than the KFC++ baseline.

We report the results in Table 2, where we perform results on the OpenAI Gym subset of the D4RL benchmark test suite. We denote the proposed scheme as **KFC++-contact** and compare to both S4RL- \mathcal{N} and KFC++. We see that KFC++-contact performs comparably and slightly better in some instances, compared to KFC++. This is expected as contact events may be rare, or the data augmentation given by the KFC++ model during contact events is pseudo-random and itself is normally distributed, which may suggest that KFC++ behaves similarly to KFC++-contact. However, we do not say this conclusively.

D.2. Magnitude of KFC augmentation

To further try to understand the effect of using KFC for guiding data augmentation, we measure the mean L2-norm of the strength of the augmentation during training. We measured the mean L2-norm between the original state and the augmented state, and found the distance to be $2.6 \cdot 10^{-4}$. For comparison, S4RL uses a fixed magnitude of $1 \cdot 10^{-4}$ (Sinha et al., 2021) for all experiments with S4RL-Adv. Since S4RL-Adv does a gradient ascent step towards the direction with the highest change, its reasonable for the magnitude of the step to be smaller in comparison. An adversarial step too large may be detrimental to training as the step can be too strong. KFC minimizes the need for making such a hyperparameter search decision.

D.3. Symmetries

Task Name	S4RL-(N)	KFC	KFC++	KFC++-contact
cheetah-random	52.3	48.6	49.2	49.4
cheetah-medium	48.8	55.9	59.1	61.4
cheetah-medium-replay	51.4	58.1	58.9	59.3
cheetah-medium-expert	79.0	79.9	79.8	79.8
hopper-random	10.8	10.4	10.7	10.7
hopper-medium	78.9	90.6	94.2	95.0
hopper-medium-replay	35.4	48.6	49.0	49.1
hopper-medium-expert	113.5	121.0	125.5	129
walker-random	24.9	19.1	17.6	18.3
walker-medium	93.6	102.1	108.0	108.3
walker-medium-replay	30.3	48.0	46.1	48.5
walker-medium-expert	112.2	114.0	115	118.1

Table 2. We study the effect of combining S4RL- N based augmentation training on “contact” event transitions (state transitions where the agent makes contact with a surface, read D.1 for more details), and KFC++ augmentation training on non-“contact” events. We see that KFC++-contact performs similarly and in most cases, slightly better than the KFC++ baseline, which is expected. We experiment with the Open AI gym subset of the D4RL tasks and report the mean normalized returns over 5 random seeds.

Global translation symmetry CartPole: The cartpole’s dynamics admits a translation symmetry w.r.t. to its position i.e. it is invariant under left and right shifts, see fig.5. In the following we employ the KFC-algorithm for deriving symmetries from the Koopman latent representation in the cartpole gym environment (Brockman et al., 2016). Moreover, we highlight that this is possible when purposefully employing imprecise Koopman observables and operators. We choose the Koopman observables to simply be the state vector $g = s = (x, \dot{x}, \theta, \dot{\theta})$; position, angle and their velocities. The dynamics is not linear in these observables, thus the latter constitute a rough approximation. Nevertheless we will see that our approach is successful. **Note that symmetries can be inferred from the approximate Koopman space representation.**

The trajectories are collected by a set of expert policies which admit the simple analytical form

$$\pi(s, z) = \Theta(0.015x + 0.066\dot{x} + 1.8\theta + 0.32\dot{\theta} + z), \quad (60)$$

where Θ is the Heaviside step function and $z \sim U(-0.2, 0.2)$ a uniformly distributed random variable. The latter is sampled once at the beginning of each episode and is then fixed until the next one. This allows for a wide variety of different expert policies from which we sample 100 trajectories each 1000 time-steps long. Note that Eq. (60) leads to an action set of $\bar{f}0, 1g$ as required by the cartpole gym environment (Brockman et al., 2016).¹⁰

We empirically train a modified Koopman forward model

¹⁰The physical action set is $\bar{f} - 1, 1g$ which we use to train the Koopman model. Note that naturally a physical action space is required for our approach.

Eq. (17) with trivial state embedding $g = s$ on the collected datasets and find that the approximate Koopman operator takes the form

$$K(a=0) = \begin{pmatrix} 1 & 0.02 & 0.01 & 0.001 \\ 0 & 1 & 0.6 & 0.8 \\ 0 & 0 & 1 & 0.02 \\ 0 & 0.01 & 0.7 & 0.2 \end{pmatrix} \quad (61)$$

$$K(a=1) = \begin{pmatrix} 1 & 0.02 & 0.009 & 0 \\ 0 & 1 & 0.6 & 0.8 \\ 0 & 0 & 1 & 0.02 \\ 0 & 0.009 & 0.7 & 0.2 \end{pmatrix} \quad (62)$$

for the actions $\bar{f}0, 1g$, respectively.¹¹

The translation symmetry generation matrix emerges when imposing it to be commuting with the Koopman operator taking the form

$$\sigma_0 = \begin{pmatrix} 1 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (63)$$

where $c_1 = 2.35, c_2 = 25.95, c_3 = 2$ for $a = 0$ and $c_1 = 2.60759, c_2 = 29.0296, c_3 = 2.22222$ for $a = 1$.

¹¹We have set numerical values at order $O(10^{-4})$ and smaller to zero. While this truncation the subsequently derived relations will hold approximately instead of precisely, e.g. the vanishing commutation relation with the symmetry operation. Moreover, we have rounded the numbers to one significant digit.

The translation symmetry shift then takes the general form

$$\sigma_a^\epsilon = 1 + \epsilon \sigma_0 = \begin{pmatrix} 1 & \epsilon & \epsilon c_1 & \epsilon c_2 & \epsilon c_3 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \quad (64)$$

for $\epsilon \in \mathbb{R}$. Note that the Decoder and Encoder from and to Koopman space are simply the identity map in this case as we chose $g = s$. It is easy to see that Eq. (64) indeed is a translation symmetry. Note that the blue-shaded area guarantees that the position does not affect any of the other state variables. The latter, are mapped to themselves by the identity matrix in the lower right corner. Moreover, the red-shaded area implies that the initial position is mapped to another position by the symmetry map, controlled by the arbitrary ϵ -parameter. We conclude that the translation symmetry operation emerges when imposing it to be commuting with the Koopman operator.

We show in theorem 3.9 that for $\epsilon = 1$ Eq. (64) is a local Lie symmetry. However, in this case we may show that the local symmetry can be extended to any global value. We take the matrix exponential of the generator σ_0 and find that it is equivalent to

$$\exp(\sigma_0 t) = \begin{pmatrix} e^{-t} & c_1(1 - e^{-t}) & c_2(1 - e^{-t}) & c_3(1 - e^{-t}) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

for a parameter $t \in \mathbb{R}$. By comparison to Eq. (64) one infers that

$$\exp(\sigma_0 t) = \sigma_a^\epsilon \quad \text{if} \quad \epsilon = 1 - e^{-t}. \quad (65)$$

In other words the composition of local symmetry transformations lead to extended "global" ones.

KFC++ algorithm: The translation symmetry can easily be inferred in a self supervised manner algorithmically by the KFC++ approach in Eq. (18) as

$$\begin{aligned} \sigma_a^\epsilon &: s \mapsto \tilde{s} = (1 + \sigma_0(\vec{\epsilon}))(s), \\ \sigma_0(\vec{\epsilon}) &= \text{Re}(U(a) \text{diag}(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4) U^{-1}(a)), \end{aligned}$$

for $a = 0, 1$. One computes the eigenvalues λ_i , $i = 1, \dots, 4$ and vectors $U(a = 0, 1)$ of the Koopman operators Eq. (61) and Eq. (62). Notable the systems admit an eigenvector $U_0 = (1, 0, 0, 0)$ to the eigenvalue $\lambda_0 = 1$. One infers that

$$\sigma_a^{\epsilon_1} = 1 + \sigma_0(\epsilon_1, \epsilon_2 = 0, \epsilon_3 = 0, \epsilon_4 = 0) = \begin{pmatrix} 1 + \epsilon_1 & \epsilon_1 c_1 & \epsilon_1 c_2 & \epsilon_1 c_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Thus the translation symmetry is obtained by symmetry shifts w.r.t. the first eigenvalue. Moreover, we note that for the general setting i.e. $\epsilon_i \neq 0$, $i = 2, 3, 4$ one finds that the translation symmetry is naturally composed by other local symmetries as

$$\sigma_a^{\vec{\epsilon}} = 1 + \sigma_0(\vec{\epsilon}) = \begin{pmatrix} 1 + \epsilon_1 & & & \\ 0 & & & \\ 0 & & & \\ 0 & & & \end{pmatrix}, \quad (66)$$

where we have used the symbol " " as proxy for in generally non-zero elements.

Local precision symmetry shifts Mujoco: In this section we provide a quantitative analysis of the symmetry shifts generated by KFC++ in comparison to random shifts of the latent states. For the latter we choose the variance such that absolute state shift

$$\Delta S := \|\tilde{s}_t - s_t\| + \|\tilde{s}_{t+1} - s_{t+1}\| \quad (67)$$

averaged over all sampled data points (D4RL) is comparable between the two setups. Moreover we simply use the encoder We then compare how well the shifted states are in alignment with the actual dynamic using the online Mujoco environment.¹² To do so we set the environment state to the value \tilde{s}_t and use the action a_t to perform the environment step which we denote by $s_{t+1} = M(s_t, a_t)$ ¹³ The performance metric is then given by

$$\Delta E := \|j_{s_{t+1}} - M(\tilde{s}_t, a_t)j\|. \quad (68)$$

Note that ΔE is simply the error to the actual dynamic of the environment. We take the model trained on the hopper-medium-expert dataset from section appendix D.1 to compute the symmetry shift. Moreover, for the comparison in Eq. (68) we split the state space into positions and velocities of the state vector, i.e. the first five elements of $s_{t+1} = M(\tilde{s}_t, a_t)$ give position while the remaining ones given velocities. The norm is taken subsequently. For the results see figure 8.

We conclude that there is a qualitative difference between the distributions obtained. Most notable the symmetry shift allows for a controlled shift by a larger ΔS while still maintaining a small error ΔE . This amounts to a wide but accurate exploration of the environments state-space. Although we cannot say this conclusively we expect this to be the reason for the performance gains of KFC++ over random shifts.

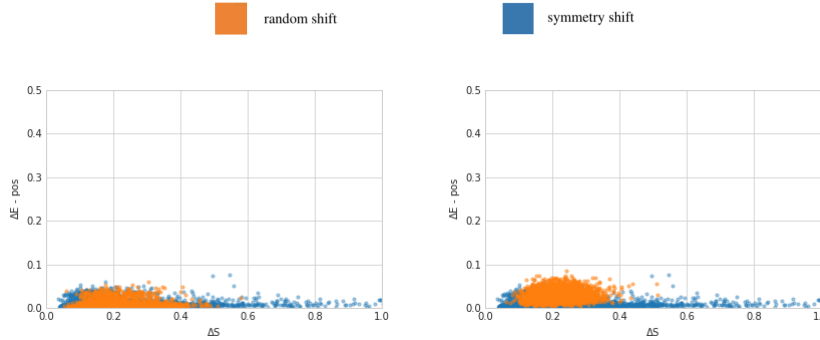


Figure 6. Hopper position state shifts

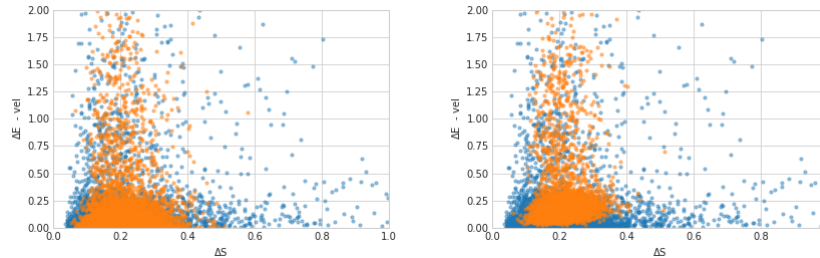


Figure 7. Hopper velocity state shifts

Figure 8. Symmetry shifts vs. random latent space shifts of same magnitude in the mean of ΔS compared by its accuracy in the online evaluation in the gym Hopper-v3 environment. The first line shows the result where s_t and s_{t+1} are shifted by the same N-dimensional random variable. For the ablation study in line two we sample twice from the random variable for the shifts s_t of s_{t+1} , respectively.

Task Name	S4RL-(Adv)	KFC	KFC++	VAE
Gym	61.61	66.36	67.76	52.90
Antmaze	62.88	65.52	67.35	54.26

Table 3. Results CQL with data-augmentation by a VAE with additive random latent space shifts. The results are averaged on the Gym and Antmaze tasks, respectively. We report the best results of the VAE over a hyperparameter search of the variance of the random variable.

D.4. VAE without Koopman Operator

We have performed an additional ablation study to compare KFC to a conventional VAE, see Table 3 for the results over the OpenAI Gym and Antmaze subset of the D4RL datasets. We find increased performance of KFC/KFC++ over data-augmentation by a common VAE with random latent space shifts. This ablation study is similar to the VAE baseline used in S4RL (Sinha et al., 2021), but the

¹²Note that this requires a minor modification of the environment code by a function which allows to set states to specific values.

¹³ $M(s_t, a_t)$ denotes the step done by the active gym environment.

main difference is that we replace the Koopman operator in KFC with adding random noise in the learned latent representation of the VAE. We add noise of the form of $\mathcal{N}(0, \epsilon^{-1})$; we perform a hyperparameter sweep over values for ϵ between 10^{-1} and 10^{-5} . We report results with the best-performing noise value ($\epsilon = 3 \cdot 10^{-3}$). The VAE does not improve the performance over the S4RL baseline or KFC variants. We conclude that KFC/KFC++ is still the best-performing method due to the use of symmetry shifts and its Koopman latent representation, which is significantly better than adding random noise to the latent representation.

D.5. Prediction Model

As a further ablation study it is of interest to compare the symmetry induced state shifts tuple $(\tilde{s}_t, \tilde{s}_{t+1})$ to one obtained by the forward prediction of our VAE-forward model Eq. (17).

KFC++-prediction: In particular we use the forward prediction on \tilde{s}_t obtained by the KFC++ shift, i.e. $\tilde{s}_{t+1} = F^{c=1}(\tilde{s}_t, a_t)$. We refer to this setup in the following as KFC++-prediction. See table 4 for the results.

Fwd-prediction: Moreover, we study the case where we \tilde{s}_t is obtained by a shift with a normally distributed random variable $\mathcal{N}(0, 6 \cdot 10^{-3})$ and then simply forward predict

as $\tilde{s}_{t+1} = F^{c=1}(\tilde{s}_t, a_t)$. This is comparable to conventional model based approaches employing our simple VAE forward model. This constitutes an ideal systematic comparison as the model, hyper-parameters and training procedure are identical to the one used in the KFC variants. Moreover, the variance of the random variable is such that the distance to the augmented states is comparable to the ones in KFC, see appendix D.2. See table 5 for the results.

We conclude that **KFC++prediction** falls behind both **KFC** as well as **KFC++**. This was expected as the symmetry shift in the latter alters the original data points by means of the VAE which admits not only a much higher accuracy but also advanced generalisation capabilities to out of distribution values.

More interesting however is that **Fwd-prediction** falls behind both **KFC** as well as **KFC++** as well as **KFC++prediction**. This is a strong indicator that the symmetry shifts provide superior out-of-distribution data for training a Q-learning algorithm.

D.6. Discussion

Although our KFC framework is suited best for the description of environments described by control dynamical systems the learned (in self-supervised manner) linear Koopman latent space representation may be applicable to a much wider set of tasks. However, there are notable shortcomings to the current implementation both conceptually as well as practically. The bilinearisation in Eq. (17) of the latent space theoretically assumes the dynamical system to be governed by Eq. (9), which is rather restrictive. Although a general non-affine control system admits a bilinearisation (Brunton et al., 2021) it generically requires the observables to depend on the action-space variables implicitly. Secondly, the Koopman operator formalism is theoretically defined by its action on an infinite dimensional observable space. The finite-dimensional approximation i.e. the latent space representation of the Koopman forward model in Eq. (17) lacks accuracy due to that. On the practical side our formalism requires data pre-processing which is computationally expensive, i.e. solving the Sylvester or eigenvalue problem for every data-point. Moreover, the Koopman forward model in Eq. (17) serves as a function approximation to two distinct tasks. Thus one faces a twofold accuracy vs. over-estimation problem which needs to be balanced. The systematic error in the VAE directly imprints itself on the state-data shift in Eqs. (16) and (18) and may thus conceal any potential benefit of the symmetry considerations. Lastly, the dynamical symmetries do not infer the reward. Thus the underlying working assumption is that the reward should not vary much in Eq. (16).

A note on the simplicity of the current algorithm: Let us stress a crucial point. Algorithmically the symmetry

maps are derived in two distinct ways **KFC** and **KFC++**. The latter, constitute a simple starting point to extract symmetries from our setup. More elaborate studies employing the extended literature on Koopman spectral analysis are desirable. Moreover, it is desirable to extend our framework to more complex latent space descriptions. It is our opinion that by doing so there is significant room for improvement both in the accuracy of the derived symmetry transformations as well their induced performance gains of Q-learning algorithms. Note that currently our VAE model is of very simple nature and the symmetries are extracted in a rather uneducated way. While the Sylvester algorithm simply converges to one out of many symmetry transformations for the KFC++ algorithm we omit all the information of the imaginary part, let alone utilize concrete spectral information.

Koopman Q-learning

Task Name	KFC	KFC++	KFC++-prediction
cheetah-random	48.6	49.2	46.5
cheetah-medium	55.9	59.1	53.7
cheetah-medium-replay	58.1	58.9	55.3
cheetah-medium-expert	79.9	79.8	76.3
hopper-random	10.4	10.7	10.8
hopper-medium	90.6	94.2	90.5
hopper-medium-replay	48.6	49.0	44.2
hopper-medium-expert	121.0	125.5	121.2
walker-random	19.1	17.6	15.6
walker-medium	102.1	108.0	105.3
walker-medium-replay	48.0	46.1	45.2
walker-medium-expert	114.0	115.6	114.5

Table 4. Results with prediction model KFC++-prediction on the Open AI Gym subset of the D4RL tasks. We report the mean normalized episodic rewards over 5 random seeds similar to the original D4RL paper (Fu et al., 2021).

Domain	Task Name	KFC	KFC++	Fwd-prediction
AntMaze	antmaze-umaze	96.9	99.8	92.7
	antmaze-umaze-diverse	91.2	91.1	90.1
	antmaze-medium-play	60.0	63.1	60.8
	antmaze-medium-diverse	87.1	90.5	88.0
	antmaze-large-play	24.8	25.6	23.1
	antmaze-large-diverse	33.1	34.0	29.3
Gym	cheetah-random	48.6	49.2	50
	cheetah-medium	55.9	59.1	50.1
	cheetah-medium-replay	58.1	58.9	56.4
	cheetah-medium-expert	79.9	79.8	71.3
	hopper-random	10.4	10.7	10.4
	hopper-medium	90.6	94.2	82.3
	hopper-medium-replay	48.6	49.0	40.8
	hopper-medium-expert	121.0	125.5	120.3
	walker-random	19.1	17.6	18.4
	walker-medium	102.1	108.0	103.2
walker-medium-replay	48.0	46.1	41.7	
walker-medium-expert	114.0	115.3	111.8	
Adroit	pen-human	61.3	60.0	49.4
	pen-cloned	71.3	68.4	50.2
	hammer-human	7.0	9.4	6.1
	hammer-cloned	3.0	4.2	4.2
	door-human	44.1	46.1	41.8
	door-cloned	3.6	5.6	1.2
	relocate-human	0.2	0.2	0.2
relocate-cloned	-0.1	-0.1	-0.1	
Franka	kitchen-complete	94.1	94.9	90.0
	kitchen-partial	92.3	95.9	84.6

Table 5. Results with prediction model Fwd-prediction on the D4RL tasks. We report the mean normalized episodic rewards over 5 random seeds similar to the original D4RL paper (Fu et al., 2021).