Modular Latent Space Transfer with Analytic Manifold Learning

Rika Antonova[‡], Maksim Maydanskiy, Danica Kragic[‡], Sam Devlin[§], Katja Hofmann[§]
[‡]EECS, KTH, Stockholm, Sweden
[§]Microsoft Research

***Abstract*—Sim2Real can be effective when using photorealistic simulators with accurate physics. However, the target appearance might be unknown a-priori and simulated dynamics might resemble reality only in some aspects. We propose an approach that uses simulation to help learning latent state representations without requiring a match in visual appearance or domain randomization. For this, we learn to encode the dynamics properties of simulated/source domains in a set of independent analytic relations that hold on sequences of low-dimensional simulation states. Then, we impose these relations onto the latent space when learning on the target domain (e.g. reality). Defining independence rigorously allows us to obtain modular representations and ensure that each relation captures a new aspect of the dynamics. This approach also enables transferring common properties from a set of related domains, instead of being confined to modeling a specific task. We show that our approach improves the quality of the latent space of unsupervised learners that train from non-stationary high-dimensional observations. We also outline potential for adaptive partial transfer, which would entail adapting the strength of imposing relations during transfer.**

I. Introduction

Consider the problem of learning latent state representations from streaming high-dimensional observations. Unsupervised learning approaches, such as variational autoencoders [1], have shown promising results. However, they lack data efficiency to learn from hardware data in non-stationary settings, e.g. during reinforcement learning (RL). Previous work proposed using domain knowledge to structure the latent space during training. For example, imposing continuity between consecutive states [2], maximizing mutual information with prior states [3], enforcing consistency with a forward or inverse model (see [4] for a survey). This improves data efficiency, but domain expertise is needed to construct such objectives/relations and ensure that they hold for target tasks/domains.

We propose[1] a more general approach that allows learning flexible relations from source domains (e.g. simulation). For this, we formalize the notion of learning a set of independent relations, without imposing restrictive simplifying assumptions or requiring domain-specific information. These relations help structure the latent space learning on target domains where data efficiency is required (e.g. learning on hardware). Their modularity/independence yields potential for partial transfer: a possibility to downweight relations that hold only in simulation and do not hold on the real underlying dynamics.

[1]This is a 2-page extended abstract summarizing the parts of our work in [5] relevant for Sim2Real. A 1-minute video summary of this RSS Sim2Real workshop abstract is given here: https://youtube.com/watch?v=6J_J6kFemLM

Consider an example 'continuity' relation [2], [6], which imposes an auxiliary loss $L_{cont}(\mathcal{D}_x, \phi) = \mathbb{E}\big[||s_{t+1} - s_t||^2\big]$, where s_t is a low-dimensional or latent state, x_t is the corresponding high-dimensional state (e.g. RGB image), $D_x = \{x_t, x_{t+1}, ...\}$ & encoder $\phi(x) = s$. Such heuristics draw from intuition and prior knowledge, and it is tedious to manually incorporate a comprehensive set of these into the overall optimization.

We take a broader perspective. Let $g(\mathcal{D}_\tau) = 0$ define a relation that holds on a set of sequences $\mathcal{D}_\tau = \{\tau^{(i)}\}_{i=1}^{M}$. $\mathcal{D}_\tau$ contains state sequences $\tau = [s_t, ..., s_{t+T}]$ from a set of source domains. We learn a set of relations $g_1, ..., g_k$ that are (approximately) independent, and we define independence rigorously. Formulating the problem as learning analytic relations that cut out the latent data manifold allows us to use neural networks as function approximators, in contrast to using polynomials to express algebraic relations [7], [8], [9].

II. Analytic Manifold Learning (AML): Proposed Mathematical Formulation and Algorithm

Let $\mathbb{R}^N$ be the ambient space of possible latent state sequences τ (of some fixed length). Let $\mathcal{M}$ be the submanifold of actual state sequences that our dynamical system could generate (under any control policy). A submanifold can be specified by describing all equations (i.e. relations) that have to hold for points in the submanifold. We aim to find relations that are in some sense independent. In linear algebra, a dependency is a linear combination of vectors with constant coefficients. In our nonlinear setting the analogous notion is that of *syzygy*. A collection of functions $\mathfrak{f}^\ddagger = \{f_1, ..., f_k\}$ is called a *syzygy* if $\sum_{j=0}^{k} f_j g_j$ is zero. Observe that this sum is a linear combination of relations $g_1, ..., g_k$ with coefficients in the ring of functions. If there is no syzygy $\mathfrak{f}^\ddagger$ s.t. $\sum_{j=0}^{k} f_j g_j = 0$, then $g_1, ..., g_k$ are independent. However, this notion of independence deems any g_1, g_2 dependent: $g_1 \cdot g_2 - g_2 \cdot g_1 = 0$ holds for any g_1, g_2. Hence, we define *restricted syzygies*.

Definition II.1 (Restricted Syzygy)**.** *Restricted syzygy for relations $g_1, ..., g_k$ is a syzygy with the last entry f_k equal to -1, i.e. $\mathfrak{f} = \{f_1, ..., f_{k-1}, f_k = -1\}$ with $\sum_{j=1}^{k} f_j g_j = 0$.*

Definition II.2 (Restricted Independence)**.** *g_k is independent from $g_1, ..., g_{k-1}$ in a restricted sense if $\sum_{j=1}^{k} f_j g_j = 0$ implies $f_k \neq -1$, i.e. if there exists no restricted syzygy for $g_1, ..., g_k$.*

We also give an alternative definition of independence via transversality, which ensures g_ks differ to first order and yields guarantees on the dimensionality of the learned submanifold.

Definition II.3 (Transversality). *If for all points $\tau^{(i)} \in \mathcal{M}$ the gradients of $g_1, .., g_k$ at τ, i.e. $v = \nabla_\tau g|_{\tau^{(i)}}$, are linearly independent, we say that g_k is transverse to the previous relations: $g_k \pitchfork g_1, ..., g_{k-1}$.*

Using the above definitions, we construct Algorithm 1, with g_ks represented by neural networks. The overall idea is: while learning g_ks, we are also looking for restricted syzygies $\mathfrak{f}(\tau, g_1, ..., g_k) = 0$. Finding them would mean g_ks are dependent, so we augment the loss for learning g_k to push it away from being dependent. We proceed sequentially: first learning g_1, then g_2 while ensuring no restricted syzygies appear for $\{g_1, g_2\}$, then g_3 and so on. For training g_ks we use *on-manifold* data: τ sequences from the simulator. Restricted syzygies $\mathfrak{f}$ train on *off-manifold* data: $\tau_{off} = \{s_{off_t}, s_{off_{t+1}}, ..., s_{off_T}\}$ (to get independence of g_ks on $\mathbb{R}^N$, not restricted to $\mathcal{M}$).

We prove that the process of adding new g_ks terminates when using syzygies, and prove guarantees related to transversality: see [5] for our proofs, further details and illustrations.

Algorithm 1 Analytic Manifold Learning (AML)

$\{\tau^{(i)}\}_{i=1}^{d} \leftarrow$ rollouts from RL actors
train g_0 with loss $L = g_d(\tau) - \log \|v\|$ (Eq.1)
for $k = 1, 2, ...,$ **do**
  if *aiming_for_transversality* **then**
    train g_k with loss L_{tr} from Eq.2
  else // using syzygies
    train g_k with loss L from Eq.1
    for $j = 1, 2, ...,$ **do**
      generate $\tau_{off}, \tau_{off}^{test}$
      train $\mathfrak{f}_j$ with $L_{\mathfrak{f}} = |\mathfrak{f}_j(\tau_{off})|$
      if $\mathfrak{f}_j \neq 0$ *on* τ_{off}^{test} **then** break //$g_k \approx$indep.
      while $\mathfrak{f}_j(\tau_{off}^{test}) \approx 0$ **do**
        freeze $\mathfrak{f}_j$; train g_k with L_{syz} (Eq.3)

$$L(g) = d_g(\tau) - \log \|v\| \;\; ; \quad d_g(\tau) = |g(\tau)| / \|v\| \tag{1}$$

$$L_{tr}(g_k) = L(g_k) - \log \prod_{j=1}^{k-1} \sin^2(\text{angle}(v_j, v_k)) \tag{2}$$

$$\nabla L_{syz}(g_k; \mathfrak{f}) = \nabla L(g_k) - \nabla_{g_k} \Big[\big| \mathfrak{f}(\tau_{off}, g_1, ..., g_k) \big| \Big] \tag{3}$$

III. Evaluating AML Potential for Sim2Real

To demonstrate potential for Sim2Real, we first learn relations from simulation states of a source domain with simple shapes sliding down an incline (*Geom-on-incline*). Incline angle, friction and object pose are initialized randomly. Actions are random forces that push objects along the incline. AML is given incline, position & velocity at two subsequent steps, and the applied action. Then, we train an unsupervised learner on the target domain: *YCB-on-incline*, which uses 3D scans of real objects from the YCB dataset [10]. PPO RL [11] drives the distribution of RGB frames, which is non-stationary, since they are sampled using the current (changing) RL policy. RL gets high rewards for pushing objects to stay in the middle of the incline. *Geom-on-incline* plays the role of a simulator, while frames from *YCB-on-incline* act as surrogates for 'real' observations. YCB objects have realistic appearances and their

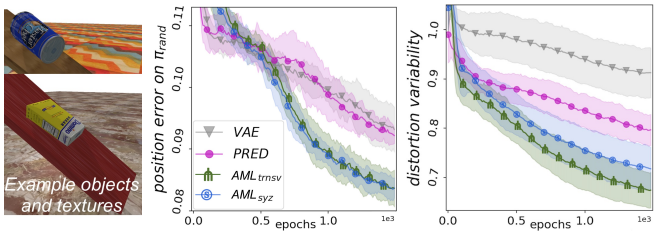

Fig. 1: *YCB-on-incline*: mean of 6 training runs, STD as shaded areas.

dynamics is dictated by meshes from scans of real objects. So there is a non-trivial gap between *Geom-* vs *YCB-on-incline*.

As baselines, we use two kinds of unsupervised learners: VAE [1] – a variational autoencoder, and $PRED$ – a sequential VAE that, given a sequence of frames $x_1, ..., x_t$, constructs a predictive sequence $x_1, ..., x_{t+k}$. Both use a 4-layer convolutional encoder & de-convolutional decoder; $PRED$ adds a fully-connected last encoder layer (512 units). We test ability of AML relations to improve latent space learning as follows: we impose AML relations by extending the latent part of an ELBO-based loss of $PRED$ (with $z_{t,t+1}$ as encoder outputs): $\mathcal{L} = -\big[\underbrace{\log p(x|z_{t,t+1}) - KL\big(q(z_{t,t+1}) \| \mathcal{N}(0,1)\big)}_{\text{standard } ELBO \text{ for } PRED \text{ version of } VAE}\big] + \underbrace{\sum_{k=1}^{K} \big|g_k(z_{t,t+1}, a_t)\big|}_{\text{impose } AML \text{ relations}}$

The resulting $\text{AML}_{\text{trnsv}}$ (AML_{syz} when using syzygies) gets a better latent state alignment for object position compared to VAE and $PRED$ without AML relations imposed (the left plot in Figure 1). To quantify the alignment, we do a regression fit using a small fully-connected neural network, which takes latents as inputs and is trained to produce low-dimensional states as outputs (object positions). The alignment is characterized by the resulting test error rate and quantifies latent space quality without needing detailed reconstructions (see [5] for details). Another important quality measure of a latent space mapping is how much it distorts the true data manifold. We quantify this as follows (on 10K test points): take pairs of low-dimensional representations τ_1^{true}, τ_2^{true} and the corresponding pixel-based representations x_1, x_2, then compute distortion coefficient $\rho_{distort} = \log \big[d_{L2}\big(\phi_{enc}(x_1), \phi_{enc}(x_2)\big) / d_{L2}\big(\tau_1^{true}, \tau_2^{true}\big) \big]$, with d_{L2} as Euclidean distance. An encoder that yields low variance of these coefficients better preserves the geometry of the low-dimensional manifold (up to scale). The right plot in Figure 1 confirms that AML helps to lower distortion variability.

IV. Future Work: Partial Transfer

AML can build a modular representation of relations encoded in the latent/low-dimensional space. Hence, AML can enable a dynamic partial transfer and thus help recover from negative transfer in cases of large Sim2Real mismatch. In future work we intend to dynamically adjust the strength of imposing each latent relation. We would combine the learned relations $g_1, ..., g_k$ using prioritization weights $w_1, ..., w_k$. These weights would be optimized by propagating the gradients of the RL loss w.r.t. the latent state representation (that these weights would influence). Further extensions could include lifelong learning: expanding the set of relations with new ones and discarding the old ones with low weights as lifelong learning proceeds.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[3] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm, "Unsupervised state representation learning in ATARI," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8766–8779.

[4] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018.

[5] R. Antonova, M. Maydanskiy, D. Kragic, S. Devlin, and K. Hofmann, "Analytic manifold learning: Unifying and evaluating representations for continuous control," *arXiv preprint arXiv:2006.08718*, 2020.

[6] V. R. Kompella, M. Luciw, and J. Schmidhuber, "Incremental slow feature analysis: Adaptive and episodic learning from high-dimensional input streams," *arXiv preprint arXiv:1112.2113*, 2011.

[7] R. Livni, D. Lehavi, S. Schein, H. Nachliely, S. Shalev-Shwartz, and A. Globerson, "Vanishing component analysis," in *International Conference on Machine Learning*, 2013, pp. 597–605.

[8] T. Sauer, "Approximate varieties, approximate ideals and dimension reduction," *Numerical Algorithms*, vol. 45, no. 1-4, pp. 295–313, 2007.

[9] D. Heldt, M. Kreuzer, S. Pokutta, and H. Poulisse, "Approximate computation of zero-dimensional polynomial ideals," *Journal of Symbolic Computation*, vol. 44, no. 11, pp. 1566–1591, 2009.

[10] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.