Augmented World Models Facilitate Zero-Shot Dynamics Generalization From a Single Offline Environment

Philip Ball*, Cong Lu*, Jack Parker-Holder1, Stephen J. Roberts1

Key Question

How can we generalize to novel environments from offline data on a single environment?

Summary

We propose:

- Dynamics augmentation for offline RL, allowing us to be robust to changing dynamics training only on a single setting.
- A simple self-supervised context adaptation algorithm, significantly increasing zero-shot performance.
- Both approaches offer significant improvement vs. SotA methods.

What happens when the dynamics change?

We train a MOPO agent on the Half-Cheetah D4RL mixed dataset.

At test time we select the mass and damping multipliers from:

\{0.25, 0.50, …, 1.75\}.

We test the zero-shot performance.

The performance can massively decrease from the default, and in many cases, the policy fails to make much progress at all. Existing offline MBRL methods cannot generalize to changed dynamics!

Practical Algorithm: AugWM

Algorithm 1: Augmented World Models Training

- Input: Offline data \(D_{\text{train}}\), Penalty \(\lambda\), Hyperparameters \(H\), and augmentation \(\mathcal{Z}\).
- Initialize: Ensemble of \(N\) dynamics models \(\hat{p}\), policy \(\pi\). Replay buffer \(D_{\text{buf}}\).
- Train \(\hat{p}\) in a supervised fashion using \(D_{\text{train}}\), for epochs \(1, 2, \ldots, \lambda\).
- Sample initial states: \((s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N) \sim D_{\text{train}}\).
- Rollout policy \(\pi\) (in parallel), using a generalized reward, storing all data in \(D_{\text{buf}}\).
- Train policy using \(D_{\text{buf}}\). For each \((s, a, r, \mathcal{Z})\), sample \(z \sim \mathcal{Z}\) and apply \(T_z\).

Return: Policy \(\pi\)

We tested our approach with the oracle augmentation, i.e., the true difference between the train and test environments.

Just augmenting dynamics is sufficient for adaption!

Sampling augmentations during training

How exactly do we approximate test-time dynamics? We consider three approaches with randomly sampled vectors \(z\):

- Random Amplitude Scaling, RAD (Laskin et al., 2020):
- Random Amplitude Nextstate Scaling, RANS:
- Dynamics Amplitude Scaling, DAS:

We train a MOPO agent on the Half-Cheetah D4RL mixed dataset.

At test time we select the mass and damping multipliers from:

\{0.25, 0.50, …, 1.75\}.

We test the zero-shot performance.

The performance can massively decrease from the default, and in many cases, the policy fails to make much progress at all. Existing offline MBRL methods cannot generalize to changed dynamics!

Learning the augmentation online

- We may infer the augmentation at test time and then pass that to the policy as a task-descriptor allowing the policy to further adapt.
- A simple linear model trained online at test time predicting change in state from the current state produces a useful signal for predicting the correct augmentation vector \(z\). An example \(R^2\) of the linear model is shown above.

Results

- The first table shows averaged results for MuJoCo changed mass/damping settings with statistically significant improvements highlighted.
- The second table shows that the DAS augmentation is suitable for more complex modified dynamics such as crippled legs and modified limb sizes.

Future Work

- Meta-learning for few-shot learning: allowing the policy to also change at test time.
- Unlimited dynamics changes during test-time.
- Augmentations in latent space for pixel-based tasks.