Explore the Context: Optimal Data Collection for Context-Conditional Dynamics Models - Supplementary Materials

Jan Achterhold **Joerg Stueckler**
Embodied Vision Group
Max Planck Institute for Intelligent Systems, Tübingen, Germany
{jan.achterhold, joerg.stueckler}@tuebingen.mpg.de

1 ARCHITECTURAL DETAILS

In the following, we describe the architectural details of our model. Notation-wise, $[\cdot;\cdot;...]$ denotes a sequence of neural network layers. `Linear`(M, N) indicates a linear layer with M input features and N output features, `ReluLinear`(M, N) is a linear layer with non-negative weights $\boldsymbol{y} = \zeta(W)\boldsymbol{x} + \boldsymbol{b}$ where $\zeta(\cdot)$ is the (elementwise) ReLU function $\zeta(x) = \max(0, x)$. `ReLU` and `Tanh` represent ReLU and hyperbolic tangent nonlinearities, respectively. `Negate` is a negation of the input features $y = -x$. `SoftplusOffset`(γ) symbolizes a softplus nonlinearity with additive offset: $y = \ln(1 + e^x) + \gamma$.

1.1 Transition model

The transition model consists of encoders g_{state}, g_{action} and $g_{\boldsymbol{\beta}}$ to lift state (dimensionality X), action (dimensionality U) and latent context variable (dimensionality B) to an embedding space with dimensionality $E = 200$. A GRU cell (Cho et al., 2014) operates in the embedding space to model the dynamics. The decoders parameterize mean and diagonal covariance on the state space given a propagated embedding. Due to the constant additive noise assumption in the toy problem environment, in those experiments, the diagonal state space covariance of the model is learned as a constant and not modeled by a decoder.

State encoder g_{state}:
[`Linear`(X, 200); `ReLU`(); `Linear`(200, E); `Tanh`()]

Action encoder g_{action}:
[`Linear`(U, 200); `ReLU`(); `Linear`(200, E); `ReLU`()]

Latent context encoder $g_{\boldsymbol{\beta}}$:
[`Linear`(B, 200); `ReLU`(); `Linear`(200, E); `ReLU`()]

GRU cell h_{RNN}:
GRU cell with input dimension 2*E (concatenation of action and latent context), state dimension E.

State decoder (mean) $d_{\text{state},\mu}$:
[`Linear`(E, 200); `ReLU`(); `Linear`(200, X)]

State decoder (diagonal covariance) $d_{\text{state},\sigma^2}$:
[`Linear`(E, 200); `ReLU`(); `Linear`(200, X); `SoftplusOffset`($1e^{-4}$)]

1.2 Context encoder

Components of the context encoder are the transition encoder and latent context decoder networks for mean and (diagonal) standard deviation. The dimensionality of the transition embedding space F is 32 for the toy problem and 128 for all other experiments. In ablation experiments in which we do not enforce the variance of the context encoder to be strictly decreasing with the number of context observations (referred to as "- Decr. variance"), we replace the `ReluLinear` layers by standard `Linear` layers.

Transition encoder:
[`Linear`($2X + U$, 200); `ReLU`(); `Linear`(200, F); `ReLU`()]

Latent context decoder (mean):
[`Linear`(F, 200); `ReLU`(); `Linear`(200, B)]

Latent context decoder (diagonal standard deviation):
[`ReluLinear`(F, 200); `ReLU`(); `ReluLinear`(200, B); `Negate`(); `SoftplusOffset`($1e^{-2}$)]

2 Training details

Table 1 gives values for the hyperparameters used for each experiment. During training, the number of context observations for each batch element is uniformly sampled from $\{0, ..., 10\}$ for the toy problem experiment and from $\{0, ..., 50\}$ for Pendulum and MountainCar.

Parameter	Toy Problem	Pendulum	MountainCar
Number of training steps	50k	100k	100k
Batchsize	64	512	512
Latent context dimensionality B	1	16	16
Transition embedding space dimensionality F	32	128	128

Table 1: Hyperparameters for the toy problem, Pendulum and MountainCar experiments

2.1 Data sampling

As detailed in the respective environment sections, for data collection, we first randomly sample instances of the parameterized toy problem, Pendulum, and MountainCar environments. On each environment instance we generate two independent rollouts (a *rollout pair*), each with a randomly sampled initial state and randomly sampled actions. We use rollout pairs from 5k instances of the toy problem, 100k instances of the Pendulum environment and 50k instances of the MountainCar environment as training data. For computing a validation loss during training, we generate rollout pairs from additional 1k toy problem, 10k Pendulum and 10k MountainCar environment samples. Enviroment instances used to evaluate the performance of the predictive model after calibration do not overlap with instances used for training and validation.

Each rollout pair we sample per environment is composed of a target rollout and a context rollout. The target chunk D^α is a random, contiguous subsequence of length 50 of the target rollout. Each transition in the context set C^α is independently sampled from the target rollout with a probability $p_{\text{ctx}-\text{from}-\text{target}}$ or from the context rollout with a probability $p_{\text{ctx}-\text{from}-\text{context}} = 1 - p_{\text{ctx}-\text{from}-\text{target}}$. For the toy problem experiments, we fix $p_{\text{ctx}-\text{from}-\text{target}} = 0$. For the Pendulum and MountainCar experiments, we set $p_{\text{ctx}-\text{from}-\text{target}} = 0.5$ for the first 30k training steps, then reduce it linearly to $p_{\text{ctx}-\text{from}-\text{target}} = 0$ until step 60k, and keep it at this value until the end of training. We motivate this scheduling strategy to simplify the learning problem by increasing the average amount of context observations which are informative for the target chunk. When context set and target chunk are sampled from different rollouts, they may cover disjoint parts of the state space, increasing the average amount of non-informative transitions in the context set.

2.2 Validation loss computation

We randomly sample 5 batches from the validation data (with a batchsize of 64 for the toy problem and 512 for Pendulum and MountainCar) to construct a validation dataset (see section 2.1). For sampling the validation

batches, we fix $p_{\text{ctx}-\text{from}-\text{target}} = 0$. The validation loss is calculated by applying the loss objective used for training (main paper, equation 14) on the validation dataset. We report results on models yielding the lowest validation loss within the given number of training steps.

2.3 Optimization

We train our models using the Adam optimizer (Kingma and Ba, 2015) with parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1\text{e}^{-4}$ and a learning rate of 1e^{-3}. We scale gradients such that the vector of concatenated gradients has a maximal 2-norm of 1000. As the latent context belief $p(\boldsymbol{\beta}|\cdot)$ is a multivariate Gaussian distribution with diagonal covariance matrix, the KL divergence term $\text{KL}(p(\boldsymbol{\beta}|D^\alpha \cup C^\alpha) \,||\, p(\boldsymbol{\beta}|C^\alpha))$ decomposes into a sum of KL divergences between scalar Gaussian distributions

$$\text{KL}(p(\boldsymbol{\beta}|D^\alpha \cup C^\alpha) \,||\, p(\boldsymbol{\beta}|C^\alpha) = \sum_i \text{KL}(p(\beta_i|D^\alpha \cup C^\alpha) \,||\, p(\beta_i|C^\alpha)) \tag{1}$$

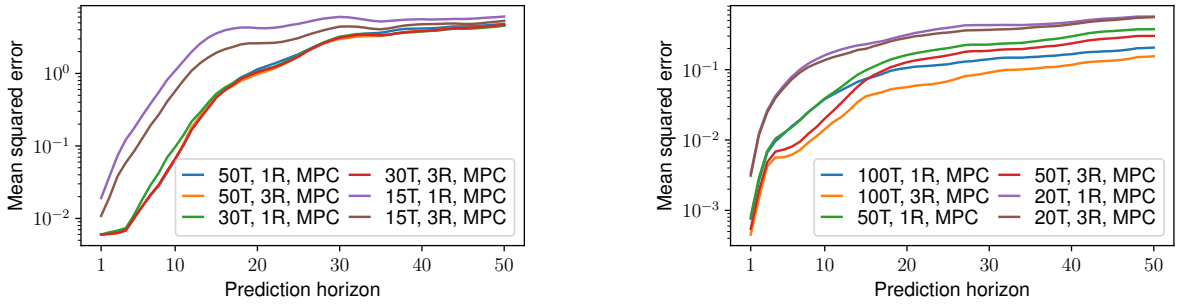
We clip $\text{KL}(p(\beta_i|D^\alpha \cup C^\alpha) \,||\, p(\beta_i|C^\alpha))$ at a minimum of 0.1 during training. This avoids local minima during the beginning of training, in which the KL divergence approaches 0 through $p(\boldsymbol{\beta}|\cdot)$ modeling a constant distribution independent of the context observations, while other loss components are not properly minimized.

3 CEM algorithm

To plan an optimal action sequence for calibration, we use a planning algorithm based on the cross-entropy method (Rubinstein, 1999) (CEM, see algorithm 1). We set the number of optimization iterations $T = 10$, number of candidates $N_{\text{cand}} = 1000$ and number of elite candidates $N_{\text{elites}} = 100$. The planning horizon is task dependent, during Open-Loop calibration, we use the full calibration horizon as planning horizon ($N = 30$ for the Pendulum, $N = 50$ for the MountainCar). During MPC calibration, the planning horizon is given by H as defined in section 3.1 of the main paper.

Input: Objective function $J : \mathcal{U}^N \to \mathbb{R},\ \mathcal{U} = [-u_{\max}, u_{\max}]$
Maximal action magnitude $u_{\max}$,
Number of optimization iterations T,
Number of candidates N_{cand},
Number of elites N_{elites}
Result: Optimal action sequence $(u_1^*, ..., u_N^*)$
Initialize $\mu_n = 0$, $\sigma_n^2 = u_{\max}^2\ \forall n \in \{1, ..., N\}$;
for $t = \{1, ..., T\}$ **do**
  Sample N_{cand} candidate sequences
    $(u_1^k, ..., u_N^k), k \in \{1, ..., N_{\text{cand}}\}$ with $\hat{u}_n^k \sim \mathcal{N}(\mu_n, \sigma_n^2)$, $u_n^k = \text{clip}(\hat{u}_n^k, -u_{\max}, u_{\max})$;
  Evaluate objective $J_k = J(u_1^k, ..., u_N^k)$;
  Obtain set of elites $\mathcal{S}_{\text{elites}} \subset \{1, ..., N_{\text{cand}}\}$
    with $|\mathcal{S}_{\text{elites}}| = N_{\text{elites}},\ J_{k'} \geq J_k\ \forall k' \in \mathcal{S}_{\text{elites}}, k \in \{1, ..., N_{\text{cand}}\} \setminus \mathcal{S}_{\text{elites}}$;
  Re-fit beliefs
    $\mu_n \leftarrow \frac{1}{N_{\text{elites}}} \sum_{k \in \mathcal{S}_{\text{elites}}} u_n^k$;
    $\sigma_n^2 \leftarrow \frac{1}{N_{\text{elites}} - 1} \sum_{k \in \mathcal{S}_{\text{elites}}} (u_n^k - \mu_n)^2$;
end
Set $(u_1^*, ..., u_N^*) \leftarrow (\mu_1, ..., \mu_N)$;

Algorithm 1: Cross-entropy method (CEM) for optimization



(a) Pendulum (b) MountainCar

Figure 1: Prediction error (lower is better) of the learned (a) Pendulum and (b) MountainCar models, for MPC calibration procedures with varying number of transitions per rollout (xT) and varying number of calibration rollouts (xR). Each line represents the mean squared error over 3000 rollouts.

4 MountainCar environment

The terrain profile of the MountainCar environment is generated by a linear combination of Gaussian functions $g(x; l, w) = \exp\left(-\frac{1}{2}\frac{(x-l)^2}{w^2}\right)$

$$y = 0.5 \cdot g(x; -1, 0.3) + 0.5 \cdot g(x; 1, 0.3) + \sum_{n=1}^{N} h_n \cdot g(x; l_n, w_n) \tag{2}$$

with $x \in [-1, 1]$, $N \sim \text{Uniform}\{2, ..., 7\}$, $h_n \sim \text{Uniform}[0.1, 0.3]$, $l_n \sim \text{Uniform}[-1.5, 1.5]$, $w_n \sim \text{Uniform}[0.1, 0.5]$. The Markovian state of the environment is represented by the current horizontal position and horizontal velocity. An external tangential acceleration $u \in [-3, 3]$ can be applied to the car. We locally approximate the dynamics of the car as a sliding block on an inclined plane with friction, where the slope of the plane is given by the average of the profile's gradient at the current point and the gradient at the simulated next point, akin to Heun's method for solving ordinary differential equations (Butcher, 2016).

Data collection For training and validation data collection, we generate 60k MountainCar environment instances (50k training / 10k validation) with randomly sampled terrain profiles. On each environment instance, we generate two randomly initialized ($x_0 \sim \text{Uniform}[-0.8, 0.8]$, $\dot{x}_0 \sim \text{Uniform}[-2, 2]$) rollouts with random actions $u_n \sim \text{Uniform}[-3, 3]$.

5 Ablation experiment: Number of calibration interactions

As an ablation experiment, we investigate the relation between the model prediction error of a calibrated model and the number of system interactions performed during calibration. To this end, we vary the number of calibration transitions per rollout and the number of calibration rollouts we perform for calibrating a single system. In case of multiple rollouts, we add the transitions of previous calibration rollouts to the set of already observed system transitions $\mathcal{T}_0$ in the MPC calibration scheme.

We vary the number of transitions per rollout as $\{15, 30, 50\}$ transitions for the Pendulum environment and $\{20, 50, 100\}$ transitions for the MountainCar environment. The number of calibration rollouts is selected from $\{1, 3\}$. The results reported in the main paper use a single rollout with 30 transitions for the Pendulum environment and 50 transitions for the MountainCar environment.

See Figure 1 for a depiction of the prediction error of the calibrated models.

For the Pendulum environment we observe that short calibration rollouts with 15 transitions yield significantly worse prediction results compared to rollouts of length 30 or 50, even when performing multiple calibration rollouts. With too short rollouts, the calibration sequence can not swing-up the Pendulum to cover all (especially the upper two) quadrants. On the other hand, longer calibration rollouts (with 50 transitions) or more calibration

attempts (3 rollouts with 30 transitions) do not yield significantly better results than a single calibration rollout with 30 transitions, because all quadrants have already been covered.

For the MountainCar environment, an environment which exhibits more complex dynamics variations than the Pendulum environment, we observe that more calibration data yields models with lower prediction error for short-horizon predictions (< 15 steps). However, for long-horizon predictions, a single long rollout (100 transitions) performs better than 3 short rollouts (50 transitions), although the total amount of calibration transitions is higher in the latter case. We hypothesize that long calibration rollouts accurately explore regions which long system rollouts reach (for long prediction horizons) and thus perform better for the long-horizon case than short calibration rollouts.

References

Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. ACL.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. of the International Conference on Learning Representations (ICLR)*.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190.