10- Model-Based Reinforcement Learning

Melih Kandemir

University of Southern Denmark
Department of Mathematics and Computer Science (IMADA)
kandemir@imada.sdu.dk

Fall 2022

Approach 1: Dyna-style MBRL

$\mathcal{D}_{real} := \emptyset$

repeat

$u \sim \mu_\theta(u|i)$

$i' := \texttt{env.step}(i, u)$ ▷ act in the real environment

$\mathcal{D}_{real} := \mathcal{D}_{real} \cup (i, u, g_i, i')$

$\psi := \arg\min_\psi \mathcal{L}(f_\psi(\cdot,\cdot), \mathcal{D}_{real})$ ▷ dynamics model update

– –

$\mathcal{D}_{sim} := \emptyset$

for do $i = 1 \to K$

$u \sim \mu_\theta(u|i)$

$i', g_i \sim f_\psi(i, u)$ ▷ imagination in the world model

$\mathcal{D}_{sim} := \mathcal{D}_{sim} \cup (i, u, g_i, i')$ ▷ policy update

$\theta, \phi := \texttt{policy-iteration-algo}(\mu_\theta(\cdot|\cdot), Q_\phi(\cdot,\cdot), \mathcal{D}_{sim})$

end for

until $|\mathcal{D}_{real}| = \tau$ ▷ interaction budget exhausted

Approach 2: Model Predictive Control (MPC)

$\mathcal{D}_{real} := \emptyset$
repeat
  $u \sim \texttt{planning-algo}(f_\psi(\cdot,\cdot), i)$
  $i' := \texttt{env.step}(i, u)$ ▷ act in the real environment
  $\mathcal{D}_{real} := \mathcal{D}_{real} \cup (i, u, g_i, i')$
  $\psi := \arg\min_\psi \mathcal{L}(f_\psi(\cdot,\cdot), \mathcal{D}_{real})$ ▷ dynamics model update
until $|\mathcal{D}_{real}| = \tau$ ▷ interaction budget exhausted

Top-trend MBRL algorithms

[Dyna] Probabilistic Inference for Learning Control (PILCO)[1]:

$$i' = f_\psi(i,u) := p(i'|i,u) = \mathcal{GP}$$

$$\texttt{policy-iteration-algo} := \texttt{DPG/Moment-Matching}$$

[Dyna] Deep PILCO[2]:

$$i' = f_\psi(i,u) := p(i'|i,u) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(i'|\mu_\psi^k(i,u), \Sigma_\psi^k(i,u)),$$

$$\psi \sim \texttt{Dropout}(\psi)$$

$$\texttt{policy-iteration-algo} := \texttt{REINFORCE}$$

[1] https://arxiv.org/abs/1502.02860

[2] http://mlg.eng.cam.ac.uk/yarin/PDFs/DeepPILCO.pdf

Top-trend MBRL algorithms

[Dyna] Model Based Policy Optimization (MBPO)[3]:

$$i', g_i = f_\psi(i,u) := p(i', g_i | i, u) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(i', g_i | \mu_\psi^k(i,u), \Sigma_\psi^k(i,u))$$

$\texttt{policy-iteration-algo} := \texttt{SAC}$

[Dyna] Guided Policy Search (GPS)[4]:

$i' = f_\psi(i,u) := \texttt{iterative-Linear-Quadratic-Regulator}$

$\texttt{policy-iteration-algo} := \texttt{REINFORCE+Importance Sampling}$

[3] https://arxiv.org/1906.08253

[4] http://proceedings.mlr.press/v28/levine13.pdf

Top-trend MBRL algorithms

[MPC] Probabilistic Ensembles with Trajectory Sampling (PETS)[5]:

$$i' = f_\psi(i,u) := p(i'|i,u) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(i'|\mu_\psi^k(i,u), \Sigma_\psi^k(i,u))$$

$$\texttt{planning-algo} := \texttt{Cross-Entropy-Method}$$

[MPC] Model-Based RL with Model-Free Fine Tuning (MBMF) [6]

$$i' = i + f_\psi(i,u) = \text{Neural Net}$$

$$\texttt{planning-algo} := \texttt{Random-Shooting}$$

[5] https://arxiv.org/abs/1805.12114

[6] https://arxiv.org/abs/1708.02596

Planning Algo 1: Random Shooting (RS)

```
for do n := 1 → N                                   ▷ Trials
    i_1 := i
    G_n := 0
    for do t := 1 → K                               ▷ Plan in a horizon of K steps
        u_t^n ~ U(u_min, u_max)                     ▷ Shoot uniformly at random
        i_{t+1}, g_{i_t} := f_ψ(i_t, u_t^n)
        G_n := G_n + g_{i_t}
    end for
end for
n_* := arg min_n G_n
return u_1^{n_*}                                    ▷ First action of the best shot
```

for do $n := 1 \to N$ ▷ Trials

  $i_1 := i$

  $G_n := 0$

  for do $t := 1 \to K$ ▷ Plan in a horizon of K steps

    $u_t^n \sim \mathcal{U}(u_{min}, u_{max})$ ▷ Shoot uniformly at random

    $i_{t+1}, g_{i_t} := f_\psi(i_t, u_t^n)$

    $G_n := G_n + g_{i_t}$

  end for

end for

$n_* := \arg\min_n G_n$

return $u_1^{n_*}$ ▷ First action of the best shot

Planning Algo 2: Cross Entropy Method (CEM)

for do $n := 1 \rightarrow N$ ▷ Trials
  $i_1 := i$
  $G_n := 0$
  for do $t := 1 \rightarrow K$ ▷ Plan in a horizon of K steps
    $u_t^n \sim \mathcal{N}(\mu, \Sigma)$ ▷ Shoot normally at random
    $i_{t+1}, g_{i_t} := f_\psi(i_t, u_t^n)$
    $G_n := G_n + g_{i_t}$
  end for
end for
$\mathcal{E} := \{n_j | G_{n_1} \geq G_{n_2} \geq \ldots \geq G_{n_R}\}$ ▷ Choose R elite sequences
$\mu := \frac{1}{R} \Sigma_{n_j \in \mathcal{E}} \Sigma_{t=1}^K u_{t,n_j}$ ▷ Update shooter parameters
$\Sigma := \frac{1}{RK-1} \Sigma_{n_j \in \mathcal{E}} \Sigma_{t=1}^K (\mu - u_{t,n_j})^2$
return u_{1,n_1} ▷ First action of the best shot

(Recap) Partially Observable MDPs

The key idea is to do MBRL by latent imagination, i.e. find a latent state space where planning is easier.

Defined as a tuple of six entities $\langle \mathcal{S}, \mathcal{A}, \mathcal{Y}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is the set of environment states: $S_t = s$ with $s \in \mathcal{S}, \ \forall t$.
- $\mathcal{A}$ is the set of actions: $A_t = a$ with $a \in \mathcal{A}, \ \forall a$.
- $\mathcal{R}$ is the set of rewards: $R_t = r$ with $r \in \mathcal{R}, \ \forall r$.
- $\mathcal{Y}$ is the set of observations: $Y_t = o$ with $o \in \mathcal{Y}, \ \forall o$.
- $\gamma \in [0, 1]$ is the **discount factor**.
- $\mathcal{P} = P(R_{t+1}, Y_{t+1}, S_{t+1} | S_t, A_t)$ is the **environment dynamics model** that naturally decomposes according to the chain rule as

$$P(R_{t+1}, Y_{t+1}, S_{t+1} | S_t, A_t) = \underbrace{P(Y_{t+1} | S_t)}_{\text{Observation model}} \underbrace{P(R_{t+1} | S_t, A_t)}_{\text{Reward model}} \underbrace{P(S_{t+1} | S_t, A_t)}_{\text{Transition model}}.$$

State-space models

State-space models

Given a sequence of random variables $Y_{1:T} = \{y_1, \cdots, y_T\}$ and $S_{1:T} = \{s_1, \cdots, s_T\}$, we refer to the following factorization as a **state-space model**:

$$p(S_{1:T}, Y_{1:T}) = p(s_0) \prod_{t=1}^{T} p(y_t|s_t)p(s_t|s_{t-1})$$

for some distributions

- $p(s_0)$, called the **recognition model**,
- $p(s_t|s_{t-1})$, called the **transition model**, and
- $p(y_t|s_t)$ called the **emission model**.

Filtering versus smoothing

One may be interested in three outcomes of this model

- **Filtering:** $p(s_T|Y_{1:T})$
- **Smoothing:** $p(s_t|Y_{1:T})$, for some $t < T$.
- **Prediction:** $p(s_{T+k}|Y_{1:T})$, for some $k > 0$.

Only filtering and prediction are relevant for the context of reinforcement learning. Hence we restrict our discussion below to these two target outcomes.

Sequential Bayesian inference

The online update lemma

For any probabilistic model of the form $p(a,b,\theta) = p(a|\theta)p(b|\theta)p(\theta)$, the following equality holds

$$p(\theta|a,b) = \frac{p(b|\theta)p(\theta|a)}{\int p(b|\theta)p(\theta|a)d\theta}.$$

Proof. Denote $Z = \int p(a|\theta)p(\theta)d\theta$, then

$$\begin{aligned}
\frac{p(b|\theta)p(\theta|a)}{\int p(b|\theta)p(\theta|a)d\theta} &= \frac{p(b|\theta)p(a|\theta)p(\theta)/Z}{\int p(b|\theta)p(a|\theta)p(\theta)/Zd\theta} \\
&= \frac{p(b|\theta)p(a|\theta)p(\theta)}{\int p(b|\theta)p(a|\theta)p(\theta)d\theta} \\
&= p(\theta|a,b) \qquad \square
\end{aligned}$$

Online posterior calculation

Online posterior calculation lemma

For any probabilistic state-space model, the following equality holds

$$p(s_t|Y_{1:t}) = \frac{p(y_t|s_t)p(s_t|Y_{1:t-1})}{\int p(y_t|s_t)p(s_t|Y_{1:t-1})ds_t}.$$

Proof. Choose $\theta := s_t$, $a := Y_{1:t-1}$, $b := y_t$ and apply the online update lemma $\square$

Bayesian filtering

Bayesian filtering for SSMs

$Y_{1:T} = \{y_1, \cdots, y_T\}$, $p(s_0)$
$\mathcal{S}_f := \emptyset$ (Filtering distributions) and $\mathcal{S}_p := \emptyset$ (Predictive distributions)
for do$t = 1 \rightarrow T$:
$\quad p(s_t|Y_{1:t-1}) := \int p(s_t|s_{t-1})p(s_{t-1}|Y_{1:t-1})ds_{t-1}$ ▷ Marginalization
$\quad p(y_t|Y_{1:t-1}) := \int p(y_t|s_t)p(s_t|Y_{1:t-1})ds_t$ ▷ Marginalization
$\quad p(s_t|Y_{1:t}) := p(y_t|s_t)p(s_t|Y_{1:t-1})/p(y_t|Y_{1:t-1})$ ▷ Inference
end for

Parameterizing the filtering model

Let us next introduce free parameters into our state-space model

$$p(S_{1:T}, Y_{1:T}|\theta) = p(s_0) \prod_{t=1}^{T} p(y_t|s_t) p(s_t|s_{t-1}, \theta).$$

For observed $Y_{1:T} = \{y_1, \cdots, y_T\}$, fit the parameters by MLE

$$\underset{\theta}{\operatorname{argmax}}\, p(Y_{1:T}|\theta).$$

This can be done by applying the product rule

$$p(Y_{1:T}|\theta) = \prod_{t=1}^{T} p(y_t|Y_{1:t-1}, \theta)$$

together with the filtering outcome for each time step t sequentially

$$\theta_* = \underset{\theta}{\operatorname{argmax}} \sum_{t=1}^{T} \log \underbrace{\int p(y_t|s_t) p(s_t|Y_{1:t-1}, \theta) ds_t}_{p(y_t|\theta, Y_{1:t-1})}.$$

Making predictions

One can predict future steps $Y_{T+1:T+K}$ using

$$p(Y_{T+1:T+K}|Y_{1:T}) = \int \prod_{t=T+1}^{T+K} p(y_t|s_t)p(s_t|s_{t-1},\theta_*)p(s_T|Y_{1:T},\theta_*)ds_{T:t}.$$

Linear SSMs: Kalman filter

Kalman filter

A Kalman Filter is defined as the state-space model that follows the distributions below

$$
\begin{aligned}
p(s_0) &= \mathcal{N}(s_0|m_0, S_0), \\
p(s_t|s_{t-1}) &= \mathcal{N}(s_t|E_t s_{t-1}, F_t), \\
p(y_t|s_t) &= \mathcal{N}(y_t|C s_t, D).
\end{aligned}
$$

such that

$$dE_t = f_{\theta_e}(E_t)dt, \quad dF_t = g_{\theta_f}(F_t)dt.$$

Both the filtering distribution and the predictive distribution remain normal distributed throughout the marginalization and inference operations due to the identities below.

Useful identities

Given $p(x) = \mathcal{N}(x|m, S)$ and $p(y|x) = \mathcal{N}(y|Cx, D)$, then the following identities hold

i) **Marginalization:**

$$p(y) = \mathcal{N}(y|Cm, D + CSC^T),$$

ii) **Inference:**

$$p(x|y) = \mathcal{N}(y|\Sigma(C^T D^{-1}(y - c) + S^{-1}m), \Sigma)$$

where $\Sigma = (S^{-1} + C^T D^{-1} C)^{-1}$.

Bayesian Kalman Filter

Input. $Y_T = \{y_1, \cdots, y_T\}$, $p(s_0) = \mathcal{N}(s_0 | m_0, S_0)$, $\theta = \{\theta_c, \theta_f, C, D\}$, E_0, F_0

for do $t = 1 \rightarrow T$:

$E^{t+1} = \int_t^{t+1} f_{\theta_c}(E_\tau) d\tau$ ▷ Roll out parameter dynamics

$F^{t+1} = \int_t^{t+1} g_{\theta_f}(F_\tau) d\tau$

$p(s_t | Y_{t-1}) := \mathcal{N}(s_t | \mu_t, \Sigma_t)$ ▷ Marginalization

$\Sigma := F_t + E_t S_{t-1} E_t^T, \qquad \mu := E_t m_{t-1}$

$p(y_t | Y_{t-1}) := \mathcal{N}(s_t | u_t, V_t)$ ▷ Marginalization

$V_t := D + C \Sigma C^T, \qquad u_t := C\mu$

$p(s_t | Y_t) := \mathcal{N}(s_t | m_t, S_t)$ ▷ Inference

$S_t := (\Sigma^{-1} + C^T D^{-1} C)^{-1}$

$m_t := S_t (C^T D^{-1} y_t + \Sigma^{-1} \mu)$

end for

Variational State-Space Model Inference

If the inference step of Bayesian filtering is no longer analytically tractable, once can approximate the posterior on the latent states by variational inference. The approximate posterior would then be

$$q(s_{0:T}|Y_{1:T}, \psi) = \prod_{t=1}^{T} q(s_t|s_{0:t-1}, Y_{1:T}, \psi)$$

which amortizes over the whole sequence of observations $Y_{1:T}$ and reparameterizes samples

$$\epsilon \sim p(\epsilon), \qquad s_t = g_\psi(s_{1:t-1}, Y_{1:T}, \epsilon)$$

for some transformation g_ψ. This is called **temporal auto-regressive factorization**[7] as the samples of $s_{1:t-1}$ are used to sample s_t.

[7] https://arxiv.org/abs/1906.10264

Variational State-Space Model Inference

If the transition model has free parameters θ to be fitted $p(s_t|s_{t-1}, \theta)$, then the corresponding ELBO with $\lambda = \{\psi, \theta\}$ would be

$$\mathcal{L}(\lambda) = \sum_{t=1}^{T} \mathbb{E}_{q(s_{0:t}|\psi, Y_{1:T})} \left[\log p(y_t|s_t) - \log \frac{q(s_t|s_{0:t-1}, \psi, Y_{1:T})}{p(s_t|s_{t-1}, \theta)} \right]$$

We can evaluate the gradient of this ELBO with respect to its free parameters by reparameterization and applying Monte Carlo integration on the expectations by ancestral sampling

$$\tilde{\epsilon}_t \sim p(\epsilon), \qquad \tilde{s}_t = g_\psi(\tilde{s}_{1:t-1}, Y_{1:T}, \tilde{\epsilon}_t), \qquad \forall t = \{1, \ldots, T\},$$

$$\begin{aligned}\nabla_\lambda \tilde{\mathcal{L}}(\lambda) = \sum_{t=1}^{T} \Big\{ & \nabla_\lambda \log p(y_t|\tilde{s}_t) - \nabla_\lambda \log q(\tilde{s}_t|\tilde{s}_{1:t-1}, \psi, Y_{1:T}) \\ & + \nabla_\lambda \log p(\tilde{s}_t|\tilde{s}_{t-1}, \theta) \Big\}.\end{aligned}$$

Note that the gradients of all three terms depend on both ψ and θ via the reparameterized sample sequence $\tilde{s}_{1:t}$.

Variational State-Space Model Inference

Given the learned parameters ψ_*, θ_* that maximize the ELBO, we can make predictions about a time point $k > 0$ steps ahead of the latest available observation as

$$p(y_{T+k}|Y_{1:T}) = \int p(y_{T+k}|s_{T+k}) \prod_{t=T+1}^{T+k} p(s_t|s_{t-1}, \theta_*) p(s_T|Y_{1:T}) ds_{T:T+k}$$

$$\approx \int p(y_{T+k}|s_{T+k}) \prod_{t=T+1}^{T+k} p(s_t|s_{t-1}, \theta_*) q(s_T|Y_{1:T}m, \psi_*) ds_{T:T+k},$$

where the integrals over s_T can be approximated by ancestral sampling from $q(s_t|s_{t-1}, Y_{1:T}, \psi_*)$ for time points up to T, collecting s_T, and continuing to sample from $p(s_t|s_{t-1}, \theta_*)$ until $T + k$.

VSSM Posterior Alternative 1: Mean-field

Capture the effect of the latent states of the previous time steps by conditioning only on data $q(s_t|s_{1:t-1}, Y_{1:T}, \psi) := q(s_t|Y_{1:T}, \psi)$ and model each time point with an independent distribution

$$\mathcal{L}(\lambda) = \sum_{t=1}^{T} \mathbb{E}_{q(s_t|Y_{1:t},\psi)}\Big[\log p(y_t|s_t)\Big] \\ - \mathbb{E}_{q(s_{t-1}|Y_{1:t},\psi)}\Big[D_{KL}(q(s_t|Y_{1:t}, \psi)||p(s_t|s_{t-1}, \theta))\Big],$$

where $p(s_0|s_{-1}, \theta) = p(s_0)$.

VSSM Posterior Alternative 2: Markovian

Mimic the true posterior $p(s_{0:T}) = p(s_0) \prod_{t=1}^{T} p(s_t|s_{t-1}, Y_{1:T})$ and factorize[8]

$$q(s_{0:T}|Y_{1:T}, \psi) = q(s_0) \prod_{t=1}^{T} q(s_t|s_{t-1}, Y_{1:T}, \psi)$$

giving rise to ELBO

$$\begin{aligned}\mathcal{L}(\lambda) = &\sum_{t=1}^{T} \mathbb{E}_{q(s_t|Y_{1:T},\psi)}[\log p(y_t|s_t)] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q(s_{t-1}|Y_{1:T},\psi)}\left[D_{KL}(q(s_t|s_{t-1}, Y_{1:T}, \psi)||p(s_t|s_{t-1}, \theta))\right]\end{aligned}$$

where the marginals $q(s_t|\psi)$ and $q(s_t|s_{t-1}, \psi)$ can be obtained by ancestral sampling from $q(s_{0:t}|\psi)$ and discarding the previous steps.

[8] https://arxiv.org/abs/1609.09869

Modeling long-term dependencies

- Precondition for accurate long-term planning tasks such as MBRL.
- Auto-regressive models map the past H observations directly to a prediction, while SSMs model state transitions explicitly.
- Auto-regressive models often train more stably, while SSMs are more interpretable and data-efficient.
- A middle ground is a **Recurrent SSM (R-SSM)**, which builds a dependency on the whole history while maintaining an explicit state transition distribution.

Recurrent State-Space Models

i) Global random function modulation. Recurrent Gaussian Process Models such as PR-SSM and VCDT assume the design

$$
\begin{aligned}
f &\sim \mathcal{GP}, \\
s_t|s_{t-1}, f &\sim p(s_t|s_{t-1}, f), \\
y_t|s_t &\sim p(y_t|s_t)
\end{aligned}
$$

but apply different variational inference schemes. Similar properties are maintained by replacing the GP with a deterministic mapping f with random parameters

$$
\begin{aligned}
\theta &\sim p(\theta), \\
s_t|s_{t-1}, \theta &\sim p(s_t|s_{t-1}, f_\theta), \\
y_t|s_t &\sim p(y_t|s_t).
\end{aligned}
$$

Recurrent State-Space Models

ii) Deterministic transition with random state input. An early example of this approach[9] is later on elaborated by the PlaNet design

$$\begin{aligned} h_t &= f_\theta(h_{t-1}, s_{t-1}), \\ s_t | s_{1:t-1} &\sim p(s_t | h_t), \\ y_t | s_{1:t} &\sim p(y_t | s_t, h_t) \end{aligned}$$

where the dependency of y_t and s_t on the whole history of s_t is due to

$$\begin{aligned} h_t &= f_\theta(h_{t-1}, s_{t-1}) \\ &= f_\theta(f_\theta(h_{t-2}, s_{t-2}), s_{t-1}) \\ &= f_\theta(f_\theta(f_\theta(h_{t-3}, s_{t-3}), s_{t-2}), s_{t-1}). \end{aligned}$$

Feeding h_t into y_t enforces complex observation models with direct historical feedback. Also used in continual neural processes[10].

[9] https://arxiv.org/abs/1506.02216

[10] https://arxiv.org/abs/1906.10264

Attentive SSM

iii) Memory modulation. Maintain a memory of past observations. How to incorporate memory into SSM inference is open question. **Attentive SSM**[11] uses the observation history $y_{1:t-1}$ to build an attendable context on the true model

$$p(Y_{1:T}, s_{1:T}) = \prod_{t=1}^{T} p(y_t|s_t)p(s_t|s_{1:t-1}, Y_{1:t-1})$$

by decomposing the transition dynamics as

$$p(s_t|s_{1:t-1}, Y_{1:t-1}) = \alpha_{1:t-1}^T P_\lambda(s_t, s_{1:t-1})$$

with respect to attention weights $\alpha_{1:t-1} = \mathrm{softmax}(\texttt{seq2seq}(Y_{1:t-1}))$ obtained from the past observations and a set of baseline transition kernels $P_\lambda(s_t, s_{1:t-1}) = \{p_\lambda(s_t, s_{t'})|t' = 1, \ldots, t-1\}$ learned as bare parameters or as a siamese network with parameters λ.

[11] https://papers.nips.cc/paper/2019/file/1d0932d7f57ce74d9d9931a2c6db8a06-Paper.pdf

Attentive SSM

Using the property

$$p(s_{1:T}|Y_{1:T}) = p(s_1|Y_{1:T}) \prod_{t=2}^{T} p(s_t|Y_{1:t-1}, s_{1:t-1}, Y_{t:T}),$$

the model chooses the approximate posterior that mimics the true posterior

$$q(s_{1:T}|Y_{1:T}) = q(s_1|Y_{1:T}) \prod_{t=2}^{T} q(s_t| \underbrace{\alpha_{1:t-1}}_{\text{share with fwd model}}, \underbrace{s_{1:t-1}}_{\text{sample}}, \underbrace{Y_{t:T}}_{\text{backward RNN}}).$$

The attention mechanism is not intuitive.

Generative Temporal Model (GTM)

The closest match for external memory based SSM inference is the **Generative Temporal Model** [12] which is a design tailored specifically to incorporation of spatial information. Given context $(X_{1:\tau}, A_{1:\tau}) = \{(x_1, a_1), \ldots, (x_\tau, a_\tau)\}$, GTM predicts the values of the observable variable $X_{\tau+1:\tau+T}$. The design has an external memory because inferring locations from visual input requires a long-term memory and building such memories with classical SSMs demand parameter sizes that grow quadratically with the state space dimensionality. The environment is assumed to consist of a latent variable s_t that encodes spatial information, while z_t encodes the frame embedding. The model assumes the data generating process below

$$
\begin{aligned}
&p(X_{\tau+1:T}, Z_{\tau+1:T}, S_{\tau+1:T} | X_{1:\tau}, A_{1:\tau}, A_{\tau+1:T}) \\
&= \prod_{t=\tau+1}^{\tau+T} p(x_t|z_t) p\Big(z_t \Big| s_t, m(X_{1:\tau}, \widehat{Z}_{1:\tau}, \widehat{S}_{1:\tau}, s_t)\Big) p(s_t|s_{t-1}, a_t).
\end{aligned}
$$

[12] https://proceedings.mlr.press/v80/fraccaro18a.html

Generative Temporal Model (GTM)

The function $m(X_{1:\tau}, \widehat{Z}_{1:\tau}, \widehat{S}_{1:\tau}, s_t)$ is an attentive memory queried by hidden state s_t with $t > \tau$. The memory maps each context element to the hidden state space using the inference network $q(s_t)$. Learn in two phases: a)memorize $(1, \tau)$, and b) infer $(\tau + 1, \tau + T)$. Memorization:

i) Mapping $X_{1:\tau}$ to $Z_{1:\tau}, S_{1:\tau}$ using an ad-hoc algorithm that approximates the filtering distribution

$$p(s_t, z_t | X_{1:t}, A_{1:t}) \approx q(s_t, z_t | X_{1:t}, A_{1:t}) = q(z_t | x_t) p(s_t | A_{1:t}),$$

where $p(s_t | A_{1:t})$ can be evaluated by ancestral sampling from $p(s_k | s_{k-1}, a_k)$ in the direction of time $k = 1, \ldots, t$.

ii) Construct KNN set $m(X_{1:\tau}, \widehat{Z}_{1:\tau}, \widehat{S}_{1:\tau}, s_t) := \{(d_k, \widehat{s}_k, \widehat{z}_k) | k = 1, \ldots, K\}$ for the query item s_t, where $d_k = dist(s_k, s_t)$ for some distance metric between the random query element s_t and a statistic $\widehat{s}_k$ sampled from $p(s_k | A_{1:k})$ for $k \leq \tau$.

Generative Temporal Model (GTM)

Then the inference phase builds the ELBO and backpropagates the gradients through the parameters:

$$\log p(X_{\tau+1:\tau+T}|X_{1:\tau+T}, A_{1:\tau+T}) \geq$$
$$\sum_{t=\tau+1}^{\tau+T} \left\{ \mathbb{E}_{q(x_t|z_t)}[\log p(x_t|z_t)] - \mathbb{E}_{p(s_t|A_{\tau+1:t})}\Big[KL(q(z_t|s_t)||p(z_t|s_t, M)\Big] \right\}.$$

where $M := m(X_{1:\tau}, \widehat{Z}_{1:\tau}, \widehat{S}_{1:\tau}, s_t))$. GTM also tries factorizing the joint in the reverse order of the original model compliantly with the amortization spirit of the adopted VAE design

$$q(Z_{\tau+1:T}, S_{\tau+1:T}|X_{1:\tau+T}, A_{1:\tau+T}) = \prod_{t=\tau+1}^{\tau+T} q(z_t|x_t) q(s_t|s_{t-1}, z_t, a_t, M).$$

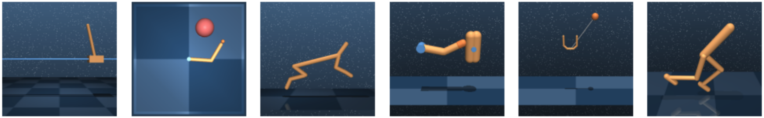
The factor $q(s_t|s_{t-1}, z_t, a_t, M)$ is assumed to query the memory using z_t as the key and s_t the value.

Generative Temporal Model (GTM)

The eventual ELBO will then be given as

$$
\begin{aligned}
&\log p(X_{\tau+1:\tau+T}|X_{1:\tau+T}, A_{1:\tau+T}) \geq \\
&\quad \sum_{t=\tau+1}^{\tau+T} \Bigg\{ \mathbb{E}_{q(x_t|z_t)}[\log p(x_t|z_t)] \\
&\qquad - \mathbb{E}_{q(s_t|z_t, A_{\tau+1:t}, M)} \Big[D_{KL}(q(z_t|x_t)||p(z_t|s_t, M) \Big] \\
&\qquad - \mathbb{E}_{q(z_t|x_t)q(s_t|z_t, a_t, M)} \Big[D_{KL}(q(s_t|s_{t-1}, z_t, a_t, M)||p(s_t|s_{t-1}, a_t)) \Big] \Bigg\}.
\end{aligned}
$$

Planning Networks (PlaNet): MPC for POMDPs



(a) Cartpole (b) Reacher (c) Cheetah (d) Finger (e) Cup (f) Walker

Figure: https://arxiv.org/pdf/1811.04551.pdf

- Visual control: Input is $64 \times 64 \times 3$ image. Easier to plan in a lower-dimensional latent space. Embed from y_t to s_t
- Learn embedding as variational inference of R-SSM: $q(s_t' | s_t, a_t, Y_{1:T})$
- Plan on the latent space s_t using CEM.

DREAMER: Dyna for POMDPs

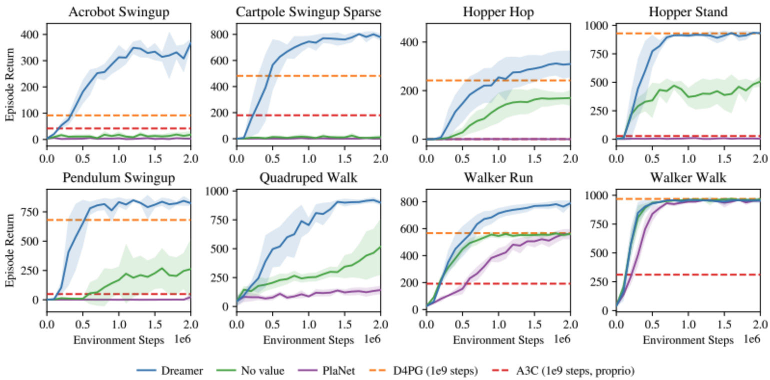


Figure: https://arxiv.org/pdf/1912.01603.pdf

- Learn embedding as variational inference of R-SSM $q(s_t'|s_t, a_t, Y_{1:T})$ supported by a contrastive loss.
- Learning policy and value networks $q_\phi(a_t|s_t)$ and $v_\psi(s_t)$.