Oregon State University

Temporal Difference Model Predictive Control

Radhika Tekade, Josyula Krishna

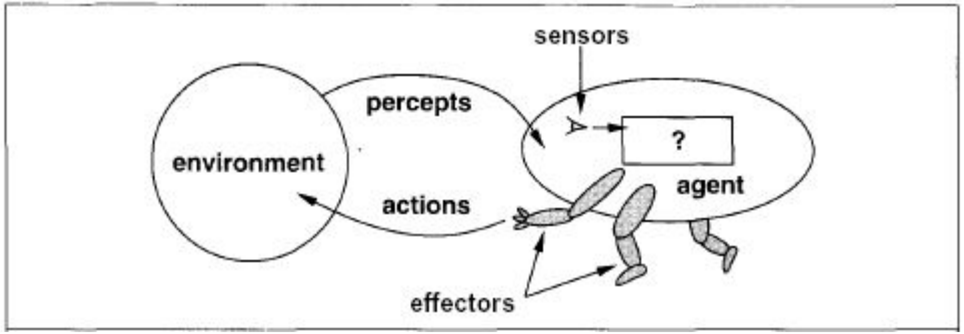
ROB 545: Kinematics, Dynamics, and Control

1 June 2022

Motivation behind this problem

- Achieving a desired behavior for an agent = iterative interaction + consolidation of knowledge about the environment, Often intractable

- Planning is a powerful approach to such sequential decision making problems, assumes a known cost function



Motivation behind this problem

Oregon State University

- TDMPC utilizes an agent's learned model of the environment
  - So that agent can plan a trajectory of actions ahead of time that leads to the desired behavior (MPC)
  - unlike model-free algorithms, which is learning a policy purely through trial-and-error

- In this work, we combine Model-free and Model Based methods for planning called TD-MPC[1].

[1] *Hansen, Nicklas and Wang, Xiaolong and Su, Hao,* Temporal Difference Learning for Model Predictive Control, arXiv:2203.04955.

What is TD-MPC?

Model Predictive Control

Models the dynamics of environment

Task Oriented Latent Dynamics (TOLD)

+

Terminal Value Function (TVF)

Retains information predictive of final reward

Model based = Model free

TD- Learning

Task Oriented Latent Dynamics(TOLD)

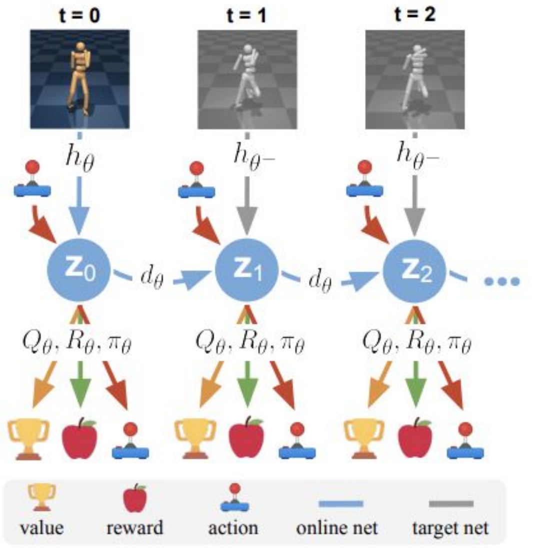
TOLD predicts

(i) The latent state dynamics (latent representation z_{t+1} of the following timestep)

(ii) the single-step reward received;

(iii) its state-action (Q) value

(iv) an action that approximately maximizes the Q-function.



Representation:	$\mathbf{z}_t = h_\theta(\mathbf{s}_t)$
Latent dynamics:	$\mathbf{z}_{t+1} = d_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Reward:	$\hat{r}_t = R_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Value:	$\hat{q}_t = Q_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Policy:	$\hat{\mathbf{a}}_t \sim \pi_\theta(\mathbf{z}_t)$

Oregon State University

Terminal Value Function (TVF)

- Terminal value function is to estimate long-term return
- Which is learned jointly by temporal difference learning with TOLD by minimizing J learning a reward prediction.

$$\mathcal{J}(\theta;\Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} \mathcal{L}(\theta;\Gamma_i)$$

$$\begin{aligned}\mathcal{L}(\theta;\Gamma_i) &= c_1 \underbrace{\| R_\theta(\mathbf{z}_i, \mathbf{a}_i) - r_i \|_2^2}_{\text{reward}} \\ &+ c_2 \underbrace{\| Q_\theta(\mathbf{z}_i, \mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1}, \pi_\theta(\mathbf{z}_{i+1}))) \|_2^2}_{\text{value}} \\ &+ c_3 \underbrace{\| d_\theta(\mathbf{z}_i, \mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1}) \|_2^2}_{\text{latent state consistency}}\end{aligned}$$

Algorithm Description

Algorithm 1 TD-MPC *(inference)*

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ $\triangleleft$ *Assuming TOLD model*
for each iteration $j = 1..J$ **do**
  Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathrm{I})$
  Sample N_π traj. of length H using π_θ, d_θ
  // Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
  for all $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Latent transition*
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ $\triangleleft$ *Terminal value*
  // Update parameters μ, σ for next iteration:
  μ^j, σ^j = Equation 4 (and Equation 5)
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

Encoding latent states

Algorithm Description

Oregon State University

Algorithm 1 TD-MPC *(inference)*

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

1: Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ ◁ *Assuming TOLD model*
2: **for** each iteration $j = 1..J$ **do**
3: Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathrm{I})$
4: Sample N_π traj. of length H using π_θ, d_θ
// Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
5: **for all** $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H})$ **do**
6: **for** step $t = 0..H-1$ **do**
7: $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Reward*
8: $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Latent transition*
9: $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ ◁ *Terminal value*
// Update parameters μ, σ for next iteration:
10: μ^j, σ^j = Equation 4 (and Equation 5)
11: **return** $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

Sampling Trajectories

Algorithm Description

Oregon State University

Algorithm 1 TD-MPC *(inference)*

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ $\triangleleft$ *Assuming TOLD model*
for each iteration $j = 1..J$ **do**
  Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathrm{I})$
  Sample N_π traj. of length H using π_θ, d_θ
  // Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
  for all $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Latent transition*
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ $\triangleleft$ *Terminal value*
  // Update parameters μ, σ for next iteration:
  μ^j, σ^j = Equation 4 (and Equation 5)
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

Learning TVF and latent dynamics

Algorithm Description

Oregon State University

Algorithm 1 TD-MPC *(inference)*

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ $\triangleleft$ *Assuming TOLD model*
for each iteration $j = 1..J$ **do**
  Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2 \mathrm{I})$
  Sample N_π traj. of length H using π_θ, d_θ
  // Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
  for all $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Latent transition*
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ $\triangleleft$ *Terminal value*
  // Update parameters μ, σ for next iteration:
  μ^j, σ^j = Equation 4 (and Equation 5)
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2 \mathrm{I})$

Return action

Advantages of TD-MPC

Oregon State University

- Combines the strengths of model-based planning and model-free learning methods

  ⇒ ↑ sample η

  ⇒ better performance (over just data-driven MPC)

Advantages of TD-MPC

Oregon State University

- Key technical contribution - “How” the model is learned
  - Representation of model purely from rewards

    ⇒ sample efficient
  - Back propagation through multiple rollout steps of the model

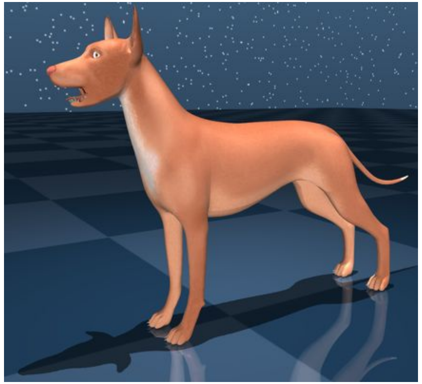
    ⇒ alleviates error compounding
  - modality-agnostic prediction loss in latent space

    ⇒ enforces temporal consistency

Simulation/Experiments

-Task list:

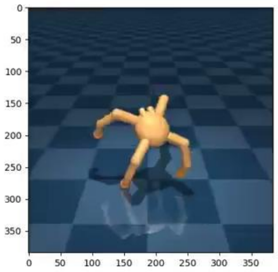
- Humanoid
- Cartpole
- Quadruped
- Dog



Preliminary Results

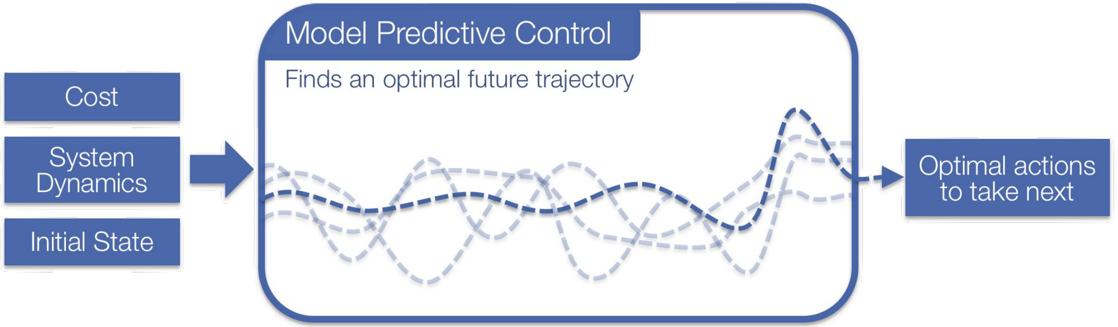
Oregon State University

SAC | **TD-MPC**



Future Work

- Incorporating better exploration strategies
- Improving the architecture of learning latent dynamics (right now uses MLP model)



Oregon State University
College of Engineering

Thank you for listening!
Any Questions?

Conclusion