Temporal Difference Model Predictive Control Applied to Deep Mind Control Suite Tasks

Radhika Tekade, Josyula Gopala Krishna

Abstract—**In this project we explore the method of learning model predictive control with temporal difference learning, which is a combination of optimal control and reinforcement learning paradigms, a model predictive control learns the best sequence of actions by interpolation of states while following a trajectory and estimating the optimal action based on a cost function, in this work, we learn the state interpolation by encoding the state vector through a Mutlilayer Perceptron, which learns the dynamics of the latent state transitions and the terminal reward function using the TD-Learning simultaneously thus behaving as a model predictive control, as shown in [1] we reproduce the methodology in [1] experiment with deep mind control suite tasks and compare with the Soft Actor Critic(SAC)[2].**

I. Introduction

In classical control to achieve a the desired behaviour for the agent, an objective function is optimized without and the information about the agent dynamics is modelled from a Euler-Lagrange formulation however in many cases this is hard to accomplish and a model of the agent cannot be obtained and has to be estimated by a system identification method [3] To bridge this gap in learning based control there have been numerous methods developed in the domain of reinforcement learning one of them is the temporal difference learning which is a model-free method which learns the value function of a state by an exponential moving average over the reward function.[4]. Where reward function is a reinforcement for achieving a target behaviour, a penalising reward can be given for undesired behaviour. This paradigm requires the agent to iteratively interact with the environment and learn the state-value function, however this requires a model-based function which can predict the state-transitions and a value function of a state both simultaneously, which requires an efficient exploration strategy along with sample efficiency in learning the value function of a state. Therefore the current work[1] considers a acting over a locally optimal trajectory as in Model Predictive Control and extends to a globally optimal solution by learning an approximate terminal value function which can estimate rewards on a trajectory beyond the current panning horizon. In the upcoming sections of our report we briefly describe previous work done in sequential planning decision making and proceed to describe the TD-MPC method and describe the advantages it presents over the existing method and its applicability to manipulators, and Deep mind control suite tasks.[5]

II. Previous Work

Previous works include classical approaches to solving an optimal control problem called Model Predictive Control, MPC uses a model of the system to make predictions about the system's future behavior. MPC solves an online optimization algorithm to find the optimal control action that drives the predicted output to the reference. MPC can handle multi-input multi-output systems that may have interactions between their inputs and outputs. It can also handle input and output constraints. MPC has preview capability; it can incorporate future reference information into the control problem to improve controller performance. MPC is also called a receding horizon control problem as time horizon is moved iteratively till it reaches a goal. The control problem is formulated as a (deterministic) optimization problem

$$\begin{aligned} & \min_{u_i} \sum_{i=0}^{p} \phi_i\left(x_i, u_i\right) \\ & g_i\left(x_i, u_i\right) \geqslant 0 \\ & x_{i+1} = F\left(x_i, u_i\right) \end{aligned} \tag{1}$$

where the objective ϕ in (1) is usually a quadratic cost function and the state is given as a linear function of control input u

$$\phi = \sum_{i=0}^{p} x_i^T Q x_i + \sum_{i=0}^{m-1} u_i^T R u_i \quad \text{and } x_{k+1} = A x_k + B u_k \\ y_k = C x_k \tag{2}$$

Unconstrained linear least squares problem 2 has an analytical solution, when inequality constraints are present there is no analytical solution and has to be solved iteratively using Sequential Quadratic Programming or other optimization methods found in [6]. It is apparent this method works best only when the function F in (1) is linear and the optimization problem has a guaranteed solution.

To this end model free methods in Reinforcement learning are known to perform well without a known model as in (1) and incorporating the non-linearities of the system as a whole and functions on continuous action spaces, the soft-actor critic is a reinforcement learning approach which works on maximising the reward of an agent using maximum entropy RL, which maximises the reward shown in the equation (3)

$$J\left(\pi_\theta\right) = E_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t R\left(s_t, a_t\right) + \alpha H\left(\pi\left(. \mid s_t\right)\right)\right] \tag{3}$$

The soft-actor critic is also seen as a combination of policy-based and value based approaches, Learning of the actor is based on policy-gradient approach and critic is learned in value-based fashion. The actor is a decision maker and is modelled as neural network which decides an action(policy) to be taken in the current state, and the critic is a neural network which computes the value of the state.

In SAC, there are three networks: the first network represents state-value(V) parameterised by ψ, the second one is a policy function $\pi_\phi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)$ that is parameterised by ϕ, and the last one represents soft $Q_\theta\left(\mathbf{s}_t, \mathbf{a}_t\right)$ function parameterised by θ.

The value network is trained by minimising the following error, which works to decrease the squared difference between the prediction of value network and the expected prediction of the Q function and the entropy of the policy function π_ϕ

$$J_V(\psi)=E_{\mathbf{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_\psi\left(\mathbf{s}_t\right)-E_{\mathbf{a}_t \sim \pi_\phi}\left[Q_\theta\left(\mathbf{s}_t, \mathbf{a}_t\right)-\right.\right.\right. \log \pi_\phi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)^2 \tag{4}$$

The Q-function is trained by (5)

$$J_Q(\theta)=E_{\left(\mathbf{s}_t, \mathbf{a}_t\right) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(\mathbf{s}_t, \mathbf{a}_t\right)-\hat{Q}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)^2\right] \tag{5}$$

where

$$\hat{Q}\left(\mathbf{s}_t, \mathbf{a}_t\right)=r\left(\mathbf{s}_t, \mathbf{a}_t\right)+\gamma E_{\mathbf{s}_{t+1} \sim p}\left[V_{\bar{\psi}}\left(\mathbf{s}_{t+1}\right)\right] \tag{6}$$

finally the policy function trains on minimising

$$J_\pi(\phi)=E_{\mathbf{s}_t \sim \mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi\left(\cdot \mid \mathbf{s}_t\right) \| \frac{\exp \left(Q_\theta\left(\mathbf{s}_t, \cdot\right)\right)}{Z_\theta\left(\mathbf{s}_t\right)}\right)\right] \tag{7}$$

More details on SAC can be found in [2]

SAC has been shown to significant results on Deep mind control suite however it doesn't combine the capabilities of a model based approach. To compensate this TD-MPC learns the Q-Value network (5) and state transitions simultaneously and is the first algorithm (as claimed in [1] and verified by our results on dog environment) to solve the Deepmind control suite dog environment. In the following section we describe TD-MPC.

III. TD-MPC

In this section we expand on TD-MPC [1] begining with notation

A. Notation

An infinite-horizon Markov Decision Processes (MDP) characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, p_0)$, where $\mathcal{S} \in R^n$ and $\mathcal{A} \in R^m$ are continuous state and action spaces, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto R_{+}$is the transition (dynamics) function, $\mathcal{R}: \mathcal{S} \times \mathcal{A} \mapsto R$ is a reward function, $\gamma \in [0,1)$ is a discount factor, and p_0 is the initial state distribution. We aim to learn a parameterized mapping $\Pi_\theta: \mathcal{S} \mapsto \mathcal{A}$ with parameters θ such that discounted return $E_{\Gamma \sim \Pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right], r_t \sim \mathcal{R}\left(\cdot \mid \mathbf{s}_t, \mathbf{a}_t\right)$ is maximized along a trajectory $\Gamma=\left(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_{\infty}\right)$ following Π_θ by sampling an action $\mathbf{a}_t \sim \Pi_\theta\left(\cdot \mid \mathbf{s}_t\right)$ and reaching state $s_{t+1} \sim \mathcal{T}(.|s_t, a_t)$.

B. MPC as RL

The Model Predictive control, when a value function is known via a a heuristic or approximated using a neural network, to estimate discounted return at state s_{t+H} and beyond such methods are known as *MPC with a terminal value function*, where H is the finite time horizon. Therefore the MPC is given by the objective in (8)

$$\Pi_\theta^{\mathrm{MPC}}\left(\mathbf{s}_t\right)=\arg \max_{\mathbf{a}_{t: t+H}} E\left[\sum_{i=t}^H \gamma^i \mathcal{R}\left(\mathbf{s}_i, \mathbf{a}_i\right)\right] \tag{8}$$

In model-free RL the Q-value is estimated and updated according to the equation (9) where the target $y(s)$ is given by the equation (6) which should work as an value approximation for the state.

$$\theta^{k+1} \leftarrow \arg \min_\theta E_{(\mathbf{s}, \mathbf{a}, \mathbf{s}^{\prime}) \sim \mathcal{B}}\left\|Q_\theta(\mathbf{s}, \mathbf{a})-y(\mathbf{s})\right\|_2^2 \tag{9}$$

C. TD-Learning for MPC

TD-MPC works by combining MPC with a task-oriented latent dynamics(TOLD) model and terminal value function jointly learned using TD-learning in an online RL setting.

TOLD attempts to model the elements of the environment which are predictive of rewards rather the environment itself. TD-MPC framework leverages the learned TOLD model for trajectory optimization and estimating short-term rewards using model rollouts and long-term returns using the terminal value function.

TOLD iteratively learns the following:

- Improving the learned TOLD model using data collected from previous environment interaction.
- Collecting new data from the environment by online planning of action sequences with TD-MPC, using TOLD for generating imagined rollouts.

TOLD is modeled as a deterministic MLP, During training, we minimize a temporally weighted objective as shown in (10) the TOLD model is pictured in Fig.1,

$$\mathcal{J}(\theta ; \Gamma)=\sum_{i=t}^{t+H} \lambda^{i-t} \mathcal{L}\left(\theta ; \Gamma_i\right) \tag{10}$$

$$\text{Where } \Gamma \sim \mathcal{B} \text{ is a trajectory } \left(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}\right)_{t: t+H} \tag{11}$$

The variables of TOLD are

$$\begin{aligned}
\mathbf{z}_t & =h_\theta\left(\mathbf{s}_t\right) \\
\mathbf{z}_{t+1} & =d_\theta\left(\mathbf{z}_t, \mathbf{a}_t\right) \\
\hat{r}_t & =R_\theta\left(\mathbf{z}_t, \mathbf{a}_t\right) \\
\hat{q}_t & =Q_\theta\left(\mathbf{z}_t, \mathbf{a}_t\right) \\
\hat{\mathbf{a}}_t & \sim \pi_\theta\left(\mathbf{z}_t\right)
\end{aligned} \tag{12}$$

where z_t is the latent state representation, $\hat{r}$ is the reward, $\hat{q}$ is the value and $\hat{a}$ is the action

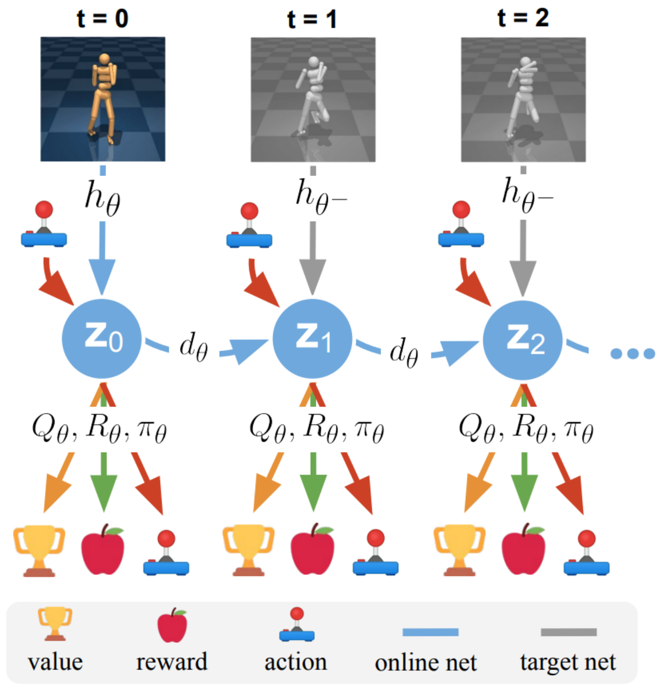

Fig. 1. TOLD model a trajectory $\Gamma_{0:H}$ of length H is sampled from a replay buffer, and the first observation, s_0 is encoded by h_θ into a latent representation z_0. Then, TOLD recurrently predicts the following latent states $z_1, z_2, ..., z_H$ as well as a value $\hat{q}$, reward $\hat{r}$, and action $\hat{a}$ for each latent state.

are learned by optimizing for reward prediction, value prediction, and a latent state consistency loss that regularizes the learned representation as shown in equation (13).

$$\begin{aligned}\mathcal{L}(\theta;\Gamma_i) &= c_1 \underbrace{\left\|R_\theta(\mathbf{z}_i,\mathbf{a}_i) - r_i\right\|_2^2}_{\text{reward}} \\ &+ c_2 \underbrace{\left\|Q_\theta(\mathbf{z}_i,\mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1}, \pi_\theta(\mathbf{z}_{i+1})))\right\|_2^2}_{\text{value}} \\ &+ c_3 \underbrace{\left\|d_\theta(\mathbf{z}_i,\mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1})\right\|_2^2}_{\text{latent state consistency}}\end{aligned} \tag{13}$$

This allows TOLD to predict a terminal value and latent state dynamics simultaneously. The training procedure is shown in Algorithm 2.(Fig.2)

IV. Experimental Results

We have utilised the TOLD public repository to train on the Deep mind control suite tasks. The latent state representation is given by a 100-dimensional vector for human and dog and 50-dimensional for other tasks. MLP is trained using ADAM with a momentum coefficient $\tau = 0.99$ and exploration parameters ϵ is changed from 0.5 to 0.05 over the first 25k steps, sampling is done with 512 trajectories.

- Input: The input to TD-MPC is the current state of the agent(joint angles, positions and velocities) for learning the TOLD as described in Algorithm 2.

```
Algorithm 2  TOLD (training)
Require: $\theta, \theta^-$: randomly initialized network parameters
         $\eta, \tau, \lambda, \mathcal{B}$: learning rate, coefficients, buffer
 1: while not tired do
 2:   // Collect episode with TD-MPC from $\mathbf{s}_0 \sim p_0$:
 3:   for step $t = 0...T$ do
 4:     $\mathbf{a}_t \sim \Pi_\theta(\cdot|h_\theta(\mathbf{s}_t))$   $\triangleleft$ Sample with TD-MPC
 5:     $(\mathbf{s}_{t+1}, r_t) \sim \mathcal{T}(\cdot|\mathbf{s}_t,\mathbf{a}_t), \mathcal{R}(\cdot|\mathbf{s}_t,\mathbf{a}_t)$   $\triangleleft$ Step env.
 6:     $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$   $\triangleleft$ Add to buffer
 7:   // Update TOLD using collected data in $\mathcal{B}$:
 8:   for num updates per episode do
 9:     $\{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}\}_{t:t+H} \sim \mathcal{B}$   $\triangleleft$ Sample traj.
10:     $\mathbf{z}_t = h_\theta(\mathbf{s}_t)$   $\triangleleft$ Encode first observation
11:     $J = 0$   $\triangleleft$ Initialize $J$ for loss accumulation
12:     for $i = t...t+H$ do
13:       $\hat{r}_i = R_\theta(\mathbf{z}_i, \mathbf{a}_i)$   $\triangleleft$ Equation 8
14:       $\hat{q}_i = Q_\theta(\mathbf{z}_i, \mathbf{a}_i)$   $\triangleleft$ Equation 9
15:       $\mathbf{z}_{i+1} = d_\theta(\mathbf{z}_i, \mathbf{a}_i)$   $\triangleleft$ Equation 10
16:       $\hat{a}_i = \pi_\theta(\mathbf{z}_i)$   $\triangleleft$ Equation 11
17:       $J \leftarrow J + \lambda^{i-t}\mathcal{L}(\mathbf{z}_{i+1}, \hat{r}_i, \hat{q}_i, \hat{\mathbf{a}}_i)$   $\triangleleft$ Equation 7
18:     $\theta \leftarrow \theta - \frac{1}{H}\eta\nabla_\theta J$   $\triangleleft$ Update online network
19:     $\theta^- \leftarrow (1-\tau)\theta^- + \tau\theta$   $\triangleleft$ Update target network
```

Fig. 2. Algorithm for training TOLD

- Ouput: TD-MPC outputs a policy(an action to be taken at time step 't') with mean μ and variance σ

The results of TD-MPC on the control suite tasks dog-run, humanoid-run, cheetah-run, and cartpole-swingup are shown below.

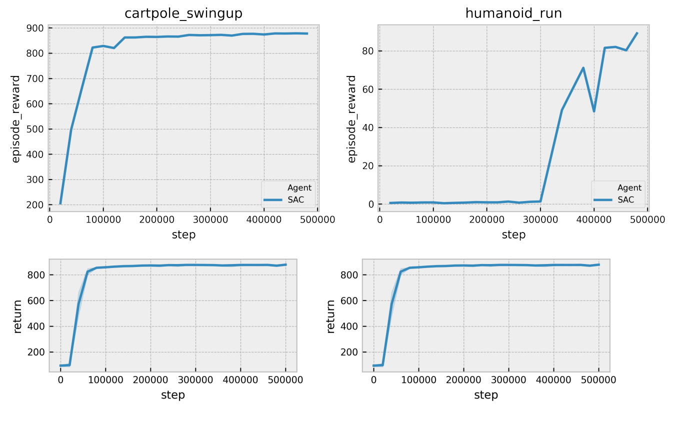

Fig. 3. top row shows SAC and the bottom row shows TDMPC results for cartpole and humanoid the value for cartpole swing is 867 after 5e6 episodes for tdmpc and 872 for SAC however on humanoid run td-mpc obtains exceedingly better results than SAC-86, TDMPC-95 and after 30e6 runs attains 800+ return

Results for dog run(Fig.5) are obtained only with TDMPC as SAC is unable to solve them.

We modified and used TDMPC and PyTorch SAC official repositories to produce these results,

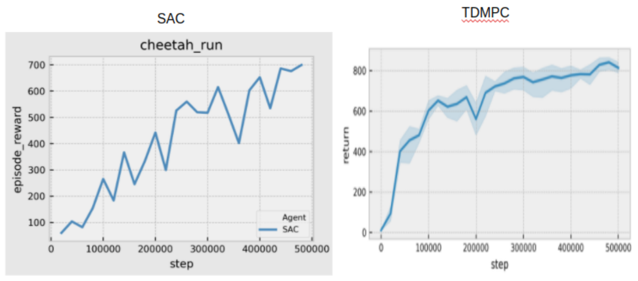

Fig. 4. left shows SAC and right shows TDMPC results for cheetah run SAC-708, TDMPC-875.

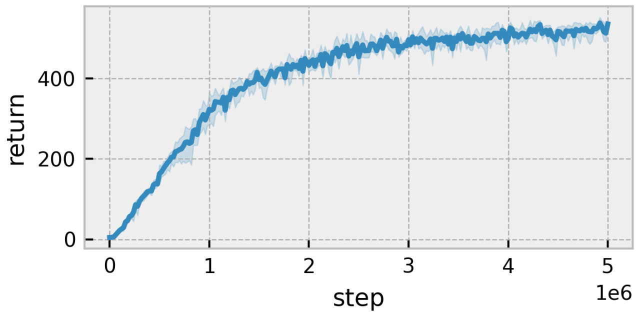

Fig. 5. Dog run task is only solvable with TDMPC with a return of 537

V. Conclusion

We have validated the results from the TOLD paper as stated in our project goals we intend to extend the TD-MPC method of solution to other domains which include a UR5 arm as shown in the meta-world in a real world application of tree pruning trained in a simulator and transferred to a real robot.

References

1 Hansen, N., Wang, X., and Su, H., “Temporal difference learning for model predictive control,” *arXiv preprint arXiv:2203.04955*, 2022.

2 Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

3 Ljung, L., “Perspectives on system identification,” *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

4 Tesauro, G., “Temporal difference learning and td-gammon,” *Commun. ACM*, vol. 38, no. 3, p. 58–68, mar 1995. [Online]. Available: https://doi.org/10.1145/203330.203343

5 Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A. *et al.*, “Deepmind control suite,” *arXiv preprint arXiv:1801.00690*, 2018.

6 Boyd, S., Boyd, S. P., and Vandenberghe, L., *Convex optimization*. Cambridge university press, 2004.