Information Theoretic Model Predictive Q-Learning

Mohak Bhardwaj[1,2] **Ankur Handa**[2] **Dieter Fox**[1,2] **Byron Boots**[1,2]
[1]**University of Washington** [2]**NVIDIA**

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

Model-free Reinforcement Learning (RL) works well when experience can be collected cheaply and model-based RL is effective when system dynamics can be modeled accurately. However, both assumptions can be violated in real world problems such as robotics, where querying the system can be expensive and real-world dynamics can be difficult to model. In contrast to RL, Model Predictive Control (MPC) algorithms use a simulator to optimize a simple policy class online, constructing a closed-loop controller that can effectively contend with real-world dynamics. MPC performance is usually limited by factors such as model bias and the limited horizon of optimization. In this work, we present a novel theoretical connection between information theoretic MPC and entropy regularized RL and develop a Q-learning algorithm that can leverage biased models. We validate the proposed algorithm on sim-to-sim control tasks to demonstrate the improvements over optimal control and reinforcement learning from scratch. Our approach paves the way for deploying reinforcement learning algorithms on real systems in a systematic manner.

1. Introduction

Deep reinforcement learning has generated great interest due to its success on a range of difficult problems including Computer Go (Silver et al., 2016) and high-dimensional control tasks such as humanoid locomotion (Schulman et al., 2015; Lillicrap et al., 2015). While these methods are extremely general and can learn policies and value functions for complex tasks directly from raw data, they are also sample inefficient, and partially-optimized solutions can be arbitrarily poor, resulting in safety concerns when run on real systems.

One straightforward way to mitigate these issues is to learn a policy or value function entirely in a high-fidelity simulator (Todorov et al., 2012; Shah et al., 2017) and then deploy the optimized policy on the real system. However, this approach can fail due to model bias, external disturbances, the subtle differences between the real robot hardware and poorly modeled phenomena such as friction and contact dynamics. Sim-to-real transfer approaches based on domain randomization (DR) (Tobin et al., 2017; Peng et al., 2018) and model ensembles (Kurutach et al., 2018; Shyam et al., 2019) aim to make the policy robust by training it to be invariant to varying dynamics. However, DR approaches are very sensitive to the choice of distribution, which is often designed by hand.

Model predictive control (MPC) is a widely used method for generating feedback controllers and has a rich history in robotic control, ranging from aggressive autonomous driving (Williams et al., 2017; Wagener et al., 2019) to contact-rich manipulation (Kumar et al., 2014; Fu et al., 2016), and humanoid locomotion (Erez et al., 2013). MPC repeatedly optimizes a finite horizon sequence of controls using an approximate dynamics model that predicts the effect of these controls on the system. The first control in the optimized sequence is executed on the real system and the optimization is performed again from the resulting next state. However, the performance of MPC can suffer due to approximate or simplified models and a limited lookahead. Therefore the parameters of MPC, including the model and horizon H need to be carefully tuned to obtain good performance. While using a longer horizon is generally preferred, real-time requirements may limit the amount of lookahead, and a biased model can result in compounding model errors. In the context of RL, local optimization is an effective way of improving an imperfect value function as noted by Silver et al. (2016, 2017); Sun et al. (2018); Lowrey et al. (2018); Anthony et al. (2017). However, these approaches assume access to a perfect model, which is not the case when dealing with robotic systems. Hence, we argue that it is essential to learn a value function from real data and utilize local optimization to stabilize learning.

In this work, we present an approach to RL that leverages the complementary properties of model-free reinforcement learning and model-based optimal control. Our proposed method views MPC as a way to simultaneously approximate and optimize a local Q function via simulation, and Q-learning as a way to improve MPC using real-world data. We focus on the paradigm of entropy regularized reinforcement learning where the aim is to learn a stochastic policy that minimizes the cost-to-go as well as KL divergence with respect to a prior policy. This has been explored in RL and Inverse RL for its better sample efficiency and exploratory properties (Ziebart et al., 2008; Todorov, 2009; Haarnoja et al., 2017; Schulman et al., 2017). This approach also enables faster convergence by mitigating the over-commitment issue in the early stages of Q-learning (Fox et al., 2015). We discuss how this formulation of reinforcement learning has deep connections to information theoretic stochastic optimal control where the objective is to find control inputs that minimize the cost while staying close to the passive dynamics of the system (Theodorou and Todorov, 2012). This helps in both injecting domain knowledge into the controller as well as mitigating issues caused by over optimizing the biased estimate of the current cost due to model error and the limited horizon of optimization. We explore this connection in depth and derive an infinite horizon information theoretic model predictive control algorithm based on Williams et al. (2017). We test our approach called Model Predictive Q-Learning (MPQ) on simulated continuous control tasks and compare it against information theoretic MPC and soft Q-Learning (Haarnoja et al., 2017), where we demonstrate faster learning with much fewer system interactions (a few minutes with real system parameters) and better performance compared to MPC and soft Q-Learning even in the presence of sparse rewards. The learned Q-function allows us to truncate the MPC planning horizon, which provides additional computational benefits. Finally, we also compare MPQ against domain randomization on sim-to-sim tasks. We conclude that DR approaches can be sensitive to the hand-designed distributions for randomizing parameters which causes the learned Q-function to be biased and suboptimal on the true system's parameters, whereas learning from data generated on true system is able to overcome biases and adapt to the real dynamics.

2. Preliminaries

We first introduce the entropy-regularized RL and information theoretic MPC frameworks and show that they are complimentary approaches to solve a similar problem.

2.1. Reinforcement Learning with Entropy Regularization

A Markov Decision Process (MDP) is defined by the tuple $\mathcal{M}=(\mathcal{S},\mathcal{A},c,P,\gamma,\mu)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, c is the per-step cost function, $s_{t+1}\sim P(\cdot|s_t,a_t)$ is the stochastic transition dynamics, γ is the discount factor and μ is the prior distribution over the initial state. A closed-loop policy $\pi(\cdot|s)$ outputs a distribution over actions given a state. Running π on the system for H-steps starting from time t results in a distribution over trajectories denoted by $d_{\pi,P}^{t,H}$ with $\tau=(s_t,a_t,\ldots,s_{t+H-1},a_{t+H-1})$ being a trajectory sample such that $a_t\sim\pi(a_t|s_t)$ and $s_{t+1}\sim P(s_{t+1}|s_t,a_t)$. The KL divergence between π and a *prior* policy $\bar{\pi}$ at a particular state is $\mathrm{KL}(\pi(\cdot|s)||\bar{\pi}(\cdot|s))=\mathbb{E}_\pi[\log(\pi(a|s)/\bar{\pi}(a|s))]$. Given $c_t=c(s_t,a_t)$ and $\mathrm{KL}_t=\mathrm{KL}(\pi(\cdot|s_t)||\bar{\pi}(\cdot|s_t))$, entropy-regularized RL (Fox et al., 2015) aims to optimize the objective

$$\pi^*=\underset{\pi}{\operatorname{argmin}}\,\mathbb{E}_{\tau\sim d_{\pi,P}^{0,\infty}}\left[\sum_{t=0}^{\infty}\gamma^t(c_t+\lambda\mathrm{KL}_t)\right] \tag{1}$$

where $s_0 \sim \mu$ and λ is a temperature parameter that penalizes deviation of π from $\bar{\pi}$. Given π, we can define the *soft* value functions and their H-timestep versions as[1]

$$V^{\pi}(s_t) = \mathbb{E}_{\tau \sim d_{\pi,P}^{t,\infty}}\left[\sum_{l=0}^{\infty} \gamma^l (c_{t+l} + \lambda \mathrm{KL}_{t+l})\right] = \mathbb{E}_{\tau \sim d_{\pi,P}^{t,H}}\left[\sum_{l=0}^{H-1} \gamma^l (c_{t+l} + \lambda \mathrm{KL}_{t+l}) + \gamma^H V^{\pi}(s_{t+H})\right] \tag{2}$$

$$Q^{\pi}(s_t, a_t) = c_t + \gamma \mathbb{E}_{s_{t+1} \sim P}[V^{\pi}(s_{t+1})] = c_t + \mathbb{E}_{\tau \sim d_{\pi,P}^{t+1,H}}\left[\sum_{l=1}^{H-1} \gamma^l (c_{t+l} + \lambda \mathrm{KL}_{t+l}) + \gamma^H (\lambda \mathrm{KL}_{t+H} + Q(s_{t+H}, a_{t+H}))\right]$$

It can be verified that $V^{\pi}(s_t) = \mathbb{E}_{a_t \sim \pi_t}[\lambda \log(\pi(a_t|s_t)/\bar{\pi}(a_t|s_t)) + Q(s_t, a_t)]$. The optimization in Eq. (1) can be performed either by policy gradient methods that aim to find the optimal policy $\pi \in \Pi$ via stochastic gradient descent (Schulman et al., 2017) or value based methods that try to iteratively approximate the value function of the optimal policy. In either case, the output is a global closed-loop control policy $\pi^*(a|s)$.

2.2. Information Theoretic MPC

Solving the above optimization for a global closed-loop policy can be prohibitively expensive and hard to accomplish during *online* operation, i.e. at every time step, as the system executes, especially when using complex policy classes like deep neural networks. In contrast, MPC computes a closed-loop policy by online optimization of a simple policy class with a truncated horizon. To achieve this, MPC algorithms such as Model Predictive Path Integral Control (MPPI) (Williams et al., 2017) solve a surrogate MDP $\hat{\mathcal{M}} = \left(\mathcal{S}, \mathcal{A}, c, \hat{P}, \gamma, H\right)$ at every timestep with an approximate dynamics model $\hat{P}$, which can be a deterministic simulator such as MuJoCo (Todorov et al., 2012), and a shorter planning horizon H.[2] At timestep t, starting from the current system state s_t, a sequence of actions $A = (a_t, a_{t+1}, \ldots a_{t+H-1})$ is sampled from a parameterized open-loop control distribution $\pi_\theta(A)$, where $\theta = [\theta_t, \theta_{t+1}, \ldots \theta_{t+H-1}]^T$ is a vector of parameters. Since the actions are independent of state, we consider them to be sampled sequentially $\pi_\theta(A) = \pi_{\theta_t}(a_t) \prod_{l=t+1}^{t+H-1} \pi_{\theta_l}(a_l | a_t, a_{t+1}, \ldots a_{l-1})$. The policy and simulator result in a trajectory distribution $d_{\pi_\theta, \hat{P}}^{t,H}$ with each trajectory $\tau = (s_t, a_t, \ldots, s_{t+H-1}, a_{t+H-1})$ sampled such that at timestep $t+l$, $a_{t+l} \sim \pi_{\theta_{t+l}}(a_{t+l} | a_t, \ldots, a_{t+l-1})$ and $s_{t+l+1} \sim \hat{P}(s_{t+l+1} | s_{t+l}, a_{t+l})$. Algorithms like MPPI aim to find an optimal θ^* that optimizes

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \, \mathbb{E}_{\tau \sim d_{\pi,\hat{P}}^{t,H}}\left[\sum_{l=0}^{H-2} \gamma^l (c_{t+l} + \lambda \mathrm{KL}_{t+l}) + \gamma^{H-1} (c_f(s_{t+H-1}, a_{t+H-1}) + \lambda \mathrm{KL}_{t+H-1})\right] \tag{3}$$

where $\mathrm{KL}_{t+l} = \mathrm{KL}\left(\pi_\theta(a_{t+l}|a_t, \ldots, a_{t+l-1}) || \bar{\pi}_\phi(a_{t+l}|a_t, \ldots, a_{t+l-1})\right)$, $\bar{\pi}_\phi(A)$ is the passive dynamics of the system, i.e the distribution over actions produced when the control input is zero with parameters ϕ and c_f is a terminal cost function. Once θ^* is obtained, the first action from the resulting distribution is executed on the system and the optimization is performed again from the next state resulting in a closed-loop controller. The re-optimization and entropy regularization helps in mitigating effects of model-bias and inaccuracies in optimization by avoiding overcommitment to the current estimate of the cost. A shortcoming of MPC is the finite horizon which is especially pronounced in tasks with sparse rewards where a short horizon can make the agent extremely myopic. To mitigate this, an approach known as *infinite horizon MPC* (Zhong et al., 2013) sets the terminal cost c_f as a value function that adds global information to the problem.

1. In this work we consider costs instead of rewards and hence aim to find policies that minimize cumulative cost-to-go.
2. We assume perfect state and cost information, as is common in MPC algorithms (Williams et al., 2017).

3. Approach

We explore the connection between entropy-regularized RL and MPPI and use it to develop an infinite horizon MPC procedure. This enables us to use MPC to approximate the Q-function and Q-learning from real-data as a way to mitigate finite horizon and model-bias issues inherent with MPC. We first derive the expression for the infinite-horizon optimal policy, which is intractable to sample from and then a scheme to iteratively approximate it with a simple policy class similar to Williams et al. (2017).

3.1. Optimal H-step Boltzmann Distribution

Let $\pi(A)$ and $\overline{\pi}(A)$ be the joint control distribution and prior over H-horizon *open-loop* actions respectively, with $\pi_t = \pi(a_t)$ and $\pi_{t+l} = \pi(a_{t+l}|a_{t+l-1},...,a_t)$. Assuming, P is deterministic, the following equations hold (Fox et al., 2015)

$$V^\pi(s_t) = \mathbb{E}_{a_t\sim\pi_t}[\lambda\log(\pi(a_t)/\overline{\pi}(a_t) + Q^\pi(s_t,a_t)] \qquad Q^\pi(s_t,a_t) = c_t + \gamma V^\pi(s_{t+1}) \tag{4}$$

For clarity, we omit the discount factor, γ. Substituting from the equation for $Q^\pi(s,a)$ into $V^\pi(s)$ in Eq. (4)

$$\begin{aligned}
V^\pi(s_t) &= \mathbb{E}_{a_t\sim\pi_t}\left[\lambda\log\left(\frac{\pi(a_t)}{\overline{\pi}(a_t)}\right) + c_t + \mathbb{E}_{a_{t+1}\sim\pi_{t+1}}\left[\lambda\log\frac{\pi_{t+1}(a_{t+1}|a_t)}{\overline{\pi}(a_{t+1}|a_t)} + Q^\pi(s_{t+1},a_{t+1})\right]\right] \\
&= \mathbb{E}_{a_t\sim\pi_t, a_{t+1}\sim\pi_{t+1}}\left[c_t + \lambda\left(\log\frac{\pi(a_t)\pi(a_{t+1}|a_t)}{\overline{\pi}(a_t)\overline{\pi}(a_{t+1}|a_t)}\right) + Q^\pi(s_{t+1},a_{t+1})\right] \\
&= \mathbb{E}_{a_t,a_{t+1}\sim\pi(a_t,a_{t+1})}\left[c_t + \lambda\left(\log\frac{\pi(a_t,a_{t+1})}{\overline{\pi}(a_t,a_{t+1})}\right) + Q^\pi(s_{t+1},a_{t+1})\right]
\end{aligned}$$

where c_t and $\log\frac{\pi(a_t)}{\overline{\pi}(a_t)}$ are taken inside the expectation as they are constants with respect to $\pi(a_{t+1}\,|\,a_t)$. Recursing H times,

$$\begin{aligned}
V^\pi(s_t) &= \mathbb{E}_{(a_t\ldots a_{t+H-1})\sim\pi(A)}\left[\sum_{l=0}^{H-2} c_{t+l} + \lambda\sum_{l=0}^{H-1}\log\frac{\pi(a_{t+l}|a_{t+l-1}\ldots a_t)}{\overline{\pi}(a_{t+l}|a_{t+l-1}\ldots a_t)} + Q^\pi(s_{t+H-1},a_{t+H-1})\right] \\
&= \mathbb{E}_{(a_t\ldots a_{t+H-1})\sim\pi(A)}\left[\sum_{l=0}^{H-2} c_{t+l} + \lambda\log\frac{\pi(A)}{\overline{\pi}(A)} + Q^\pi(s_{t+H-1},a_{t+H-1})\right]
\end{aligned} \tag{5}$$

Eq. (5) is similar to Eq. (2) with the key difference being the use of open-loop policies and the deterministic dynamics assumption, leading to the expectation and KL divergence being applied to the joint action distribution rather than the state-action trajectory distribution $d^{t,H}_{\pi,P}$. Now, consider the following joint action distribution over horizon H

$$\pi(A) = \frac{1}{\eta}\exp\left(\frac{-1}{\lambda}\left(\sum_{l=0}^{H-2} c_{t+l} + Q^\pi(s_{t+H-1},a_{t+H-1})\right)\right)\overline{\pi}(A) \tag{6}$$

where $\eta = \mathbb{E}_{\overline{\pi}(A)}\left[\exp\left(\frac{-1}{\lambda}\left(\sum_{l=0}^{H-2} c_{t+l} + Q^\pi(s_{t+H-1},a_{t+H-1})\right)\right)\right]$ is a normalizing constant. We show that this is the optimal control distribution as $\nabla V^\pi(s_t) = 0$. Substituting Eq. (6) into Eq. (5))

$$V^\pi(s_t) = \mathbb{E}_{\pi(A)}\left[\sum_{l=0}^{H-2} c_t - \lambda\log(\eta) - \sum_{l=0}^{H-2} c_t - Q^\pi(s_{t+H-1},a_{t+H-1}) + Q^\pi(s_{t+H-1},a_{t+H-1})\right] = \mathbb{E}_{\pi(A)}[-\lambda\log(\eta)]$$

Since η is a constant, $V^\pi(s) = -\lambda\log(\eta)$. Therefore, for π in Eq. (6), the soft value function is a constant with gradient zero and is thus the optimal value function, i.e

$$V^{\pi^*}(s_t) = -\lambda\log\mathbb{E}_{\overline{\pi}(A)}\left[\exp\left(\frac{-1}{\lambda}\left(\sum_{l=0}^{H-2} c(s_{t+l},a_{t+l}) + Q^\pi(s_{t+H-1},a_{t+H-1})\right)\right)\right] \tag{7}$$

which is often referred to in optimal control literature as the "free energy" of the system (Theodorou and Todorov, 2012). For H=1, Eq. (7) takes the form of the soft value function from Haarnoja et al. (2018).

3.2. Infinite Horizon MPPI Update Rule

We follow the approach of Williams et al. (2017) to derive our infinite horizon MPPI update rule. Since sampling actions from the optimal control distribution in Eq. (6) is intractable, we consider parameterized control policies $\pi_\theta(A) \in \Pi$ which are easy to sample from. We then optimize for a vector of H parameters θ, such that the resulting action distribution minimizes the KL divergence with the optimal policy

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \, \mathrm{KL}(\pi^*(A)||\pi_\theta(A)) \tag{8}$$

The objective can be expanded as

$$\mathrm{KL}(\pi^*(A)||\pi_\theta(A)) = \int_A \pi^*(A) \log \frac{\pi^*(A)}{\pi_\theta(A)} \mathrm{d}A = \int_A \pi^*(A) \left(\log \frac{\pi^*(A)}{\bar{\pi}(A)} - \log \frac{\pi_\theta(A)}{\bar{\pi}(A)} \right) \mathrm{d}A \tag{9}$$

$$\theta^* = \underset{\pi_\theta(A)}{\operatorname{argmax}} \int_A \pi^*(A) \log \frac{\pi_\theta(A)}{\bar{\pi}(A)} \, \mathrm{d}A \tag{10}$$

where first term was removed as it was independent of θ. Consider Π, to be a time-independent multivariate Gaussian over sequence of the H controls with constant covariance Σ at each timestep. We can write control distribution and prior as

$$\pi_U(A) = \frac{1}{Z} \prod_{t=0}^{H-1} \exp\left(-\frac{1}{2} (u_t - a_t)^T \Sigma^{-1} (u_t - a_t) \right) \qquad \bar{\pi}(A) = \frac{1}{Z} \prod_{t=0}^{H-1} \exp\left(-\frac{1}{2} a_t^T \Sigma^{-1} a_t \right) \tag{11}$$

where u_t and a_t are the control inputs and actions respectively at timestep t and Z is the normalizing constant. Here the prior corresponds to the passive dynamics of the system (Theodorou and Todorov, 2012; Williams et al., 2017), although other choices are possible. The policy parameters θ are the sequence of control inputs $U = [u_1, u_2, ..., u_H]$, which is the mean of the Gaussian. Substituting in Eq. (10):

$$U^* = \underset{\pi_U(A)}{\operatorname{argmax}} \int \pi^*(A) \left(\sum_{t=0}^{H-1} -\frac{1}{2} u_t^T \Sigma^{-1} u_t + u_t^T \Sigma^{-1} a_t \right) \mathrm{d}A \tag{12}$$

The objective can be simplified to the following by integrating out the probability in the first term

$$\sum_{t=0}^{H-1} -\frac{1}{2} u_t^T \Sigma^{-1} u_t + u_t^T \int \pi^*(A) \Sigma^{-1} a_t \, \mathrm{d}A \tag{13}$$

Since this is a concave function with respect to every u_t, we can find the maximum by setting its gradient with respect to u_t to zero and solving for the optimal u_t^*

$$u_t^* = \int \pi^*(A) a_t \mathrm{d}A \quad = \int \pi_U(A) \frac{\pi^*(A)}{\bar{\pi}(A)} \frac{\bar{\pi}(A)}{\pi_U(A)} a_t \mathrm{d}A = \mathbb{E}_{\pi_U(A)} \left[\frac{\pi^*(A)}{\bar{\pi}(A)} \frac{\bar{\pi}(A)}{\pi_U(A)} a_t \right] = \mathbb{E}_{\pi_U(A)}[w(A) a_t] \tag{14}$$

where the second equality comes from importance sampling to convert the optimal controls into an expectation over the control distribution instead of the optimal distribution, which is impossible to sample from. The importance weight $w(A)$ can be written as follows (substituting π^* from Eq. (6))

$$w(A) = \frac{1}{\eta} \exp\left(\frac{-1}{\lambda} \left(\sum_{t=0}^{H-2} c(s_t, a_t) + Q^{\pi^*}(s_{H-1}, a_{H-1}) \right) \right) \frac{\bar{\pi}(A)}{\pi_U(A)} \tag{15}$$

We can form an iterative Monte-Carlo method to estimate u_t^*. At every iteration i we sample N,H-step action sequences $A_1, ..., A_N$ from current distribution $\pi_{U^i}(A)$ and obtain the next control for every t as

$$u_t^{i+1} = \sum_{n=1}^{N} w(A_n) a_t^n \tag{16}$$

where η is also estimated from the Monte-Carlo samples. Eq. (16) is the infinite horizon MPPI update rule.[3] For $H = 1$, it corresponds to soft Q-learning with stochastic optimization to find the optimal action. This leads to a soft Q-learning algorithm, MPQ that uses infinite horizon MPPI to generate actions and Q-targets.

3. Please refer to full version at bit.ly/2YjRpxS for a more complete derivation.

3.3. Information Theoretic Model Predictive Q-Learning

We consider parameterized value functions $Q_\theta(s,a)$ where parameters θ are updated by stochastic gradient descent on the loss $L(\theta)=\frac{1}{K}\sum_{i=1}^{K}(y_i-Q_\theta(s_i,a_i))^2$ for a batch of K experience tuples (s,a,c,s') sampled from a replay buffer. Targets y_i are calculated using the Bellman equation as

$$y=c(s,a)-\gamma\lambda\log\mathrm{E}_{\pi^*(A)}\left[\exp\left(\frac{-1}{\lambda}\left(\sum_{t=0}^{H-2}c(s_t,a_t)+\sum_{t=0}^{H-1}\lambda\log\frac{\pi_t^*}{\bar{\pi}_t}+Q_\theta(s_{H-1},a_{H-1})\right)\right)\middle|s_0=s'\right] \tag{17}$$

The second term is the same as free energy from Eq. (7) with the expectation over prior converted to expectation over the optimal policy π^* using importance sampling. Since the value function updates are performed offline, we can utilize large amounts of computation to obtain $\pi^*(A)$. We do so by performing multiple iterations of the infinite horizon MPPI update in Eq. (16) from s', which allows for better approximation of the free energy (akin to approaches such as Covariance Matrix Adaption, although MPPI does not adapt the covariance). This helps in early stages of learning by providing better quality targets than a random Q function. Intuitively, this update rule leverages the biased dynamics model $\hat{P}$ for H steps and a soft Q function at the end learned from interactions with the real system.

At every timestep t during online rollouts, an H-horizon sequence of actions is optimized using infinite horizon MPPI and the first action is executed on the system. Online optimization with predictive models can look ahead to produce better actions than ad-hoc exploration strategies such as ϵ-greedy. Combined with the better Q estimates, this helps in accelerating learning as we demonstrate in our experiments. Algorithm 1 shows the complete MPQ algorithm. A closely related approach in literature is POLO (Lowrey et al., 2018), which also uses MPPI and offline value function learning, however they do not explore the connection between MPPI and entropy regularized RL, and thus the algorithm does not use free energy targets.

Algorithm 1: MPQ

Input: Approximate model $\hat{P}$, initial Q function parameters θ_1, experience buffer $\mathcal{D}$
Parameter: Number of episodes N, length of episode T, planning horizon H, number of update episodes N_{update}, minibatch-size K, number of minibatches M

for $i=1...N$ **do**
  for $t=1...T$ **do**
    $(a_t,...,a_{t+H})\leftarrow$ Infinite horizon MPPI (Eq. (16))
    Execute a_t on the real system to obtain $c(s_t,a_t)$ and next state s_{t+1}
    $\mathcal{D}\leftarrow(s_t,a_t,c,s_{t+1})$
  if $i\%N_{update}==0$ **then**
    Sample M minibatches of size K from $\mathcal{D}$
    Generate targets using Eq. (17) and update parameters to θ_{i+1}
  return θ_N or best θ on validation.

4. Experiments

We evaluate the efficacy of MPQ on two fronts: (a) overcoming the shortcomings of both stochastic optimal control and model free RL in terms of computational requirements, model bias, and sample efficiency; and (b) learning effective policies on systems for which accurate models are not known.

4.1. Experimental Setup

We focus on sim-to-sim continuous control tasks using MuJoCo (except PENDULUMSWINGUP that uses dynamics equations) to study the properties of MPQ in a controlled manner. We consider robotics-inspired tasks with either sparse rewards or requiring long-horizon planning. The complexity is further aggravated as the agent is not provided with the true dynamics parameters, but a uniform distribution over them with a biased mean and added noise. Details of the tasks are [4]

1. PENDULUMSWINGUP: the agent tries to swingup and stabilize a pendulum by applying torque on the hinge given a biased distribution over its mass and length.

2. BALLINCUPSPARSE: a sparse version of the task from the Deepmind Control Suite (Tassa et al., 2018). Given a cup and ball attached by a tendon, the goal is to swing and catch the ball. The agent is provided with a biased distribution over the ball's mass, moment of inertia and tendon stiffness.

3. FETCHPUSHBLOCK: proposed by Plappert et al. (2018), the agent controls the cartesian position and opening of a Fetch robot gripper to push a block to a goal location. A biased distribution over the mass, moment of inertia, friction coefficients and size of the object is provided.

4. FRANKADRAWEROPEN: based on a real-world manipulation problem from Chebotar et al. (2019) where the agent velocity controls a 7DOF Franka Panda arm to open a cabinet drawer. A biased distribution over damping and frictionloss of robot and drawer joints is provided.

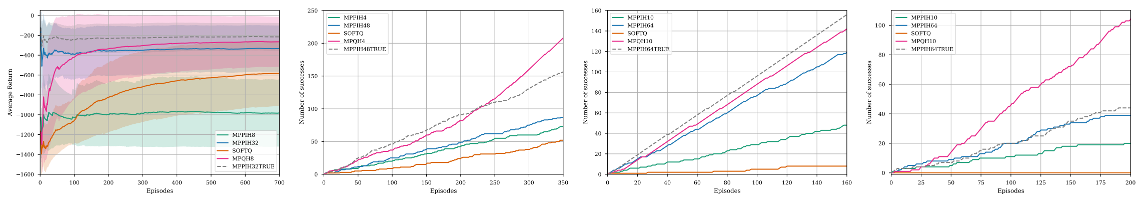
We believe the tasks we chose are more realistic proxies for real-world robotics tasks than standard OpenAI Gym (Brockman et al., 2016) baselines such as ANT and HALFCHEETAH. The parameters we randomize are reasonable in real world scenarios as estimating moment of inertia and friction coefficients is especially error prone. Experiments were performed on a desktop with 12 Intel Core i7-3930K @ 3.20GHz CPUs and 32 GB RAM with only few hours of training. Q-functions are parameterized with feed-forward neural networks with observation and action vector inputs.

4.2. Analysis of Overall Performance

By learning a terminal value function from real data we posit that MPQ will adapt to true system dynamics and allow us to truncate the MPC horizon. Using MPC for Q targets, we also expect to require significantly less data than model-free Q-learning. Hence, we compare MPQ with the following natural baselines: vanilla MPPI with same horizon as MPQ, MPPI with longer horizon, MPPI with longer horizon + true dynamics and SOFTQLEARNING with target networks. Note MPQ does not use a target network. We do not compare with model-based RL methods (Kurutach et al., 2018; Chua et al., 2018) as learning globally consistent neural network models adds an additional layer of complexity beyond the scope of this work. MPQ is complementary to model learning and one can benefit from the other. We make following observations:

O 1 MPQ *can truncate the planning horizon leading to computational efficiency over MPPI.*
Fig. 1 shows that MPQ outperforms MPPI with the same horizon after only a few training episodes and ultimately outperforms MPPI with a much longer horizon. This is due to global information encapsulated in the Q-function, hardness of optimizing longer sequences and compounding model error in longer rollouts. In FETCHPUSHBLOCK, MPPI with a short horizon ($H = 10$) is unable to reach near the block whereas MPQ with $H = 10$ outperforms MPPI with $H = 64$ within 30 training episodes i.e. about 2 minutes of interaction with true sim parameters. In the high-dimensional FRANKADRAWEROPEN, MPQ with $H = 10$ achieves a success rate of $>$5X MPPI with $H = 10$, and outperforms MPPI with $H = 64$ within a few minutes of interaction. We also examine the effects of varying the horizon during training. (Refer to full version for results)

4. Refer to full version at bit.ly/2YjRpxS for more details of tasks, dynamics randomization distributions and learning parameters.



(a) PENDULUMSWINGUP (b) BALLINCUPSPARSE (c) FETCHPUSHBLOCK (d) FRANKADRAWEROPEN

Figure 1: Comparison of MPQ against baselines during training. The number following H in the legend corresponds to MPC horizon. The dashed grey line shows performance of MPPI with access to true dynamics and no terminal Q function, denoting the upper limit on the performance MPPI can achieve. The horizon for MPPI and MPQ were selected after a rough grid search.

O 2 MPQ *mitigates effects of model-bias through a combination of MPC, entropy regularization and a Q function learned from true system.*

Fig. 1 shows that MPQ with short horizon achieves performance close to, or better than, MPPI with true dynamics and a longer horizon (dashed gray line) in all tasks.

O 3 *Using MPC provides stable Q targets leading to sample efficiency over* SOFTQLEARNING

In BALLINCUPSPARSE, FETCHPUSHBLOCK and FRANKADRAWEROPEN, SOFTQLEARNING does not converge to a consistent policy whereas MPQ achieves good performance within few minutes of interaction with true system parameters.

CASE STUDY: LEARNING POLICIES FOR SYSTEMS WITH INACCURATE MODELS

DR aims to make a policy learned in simulation robust by randomizing the simulation parameters. But, such policies can be suboptimal with respect to true parameters due to bias in randomization distribution.

Q 1 *Can a Q-function learned using rollouts on a real system overcome model bias and outperform DR?*

We compare against a DR approach inspired by Peng et al. (2018) where simulated rollouts are generated by sampling parameters at every timestep from a broad distribution whereas real system rollouts use the true parameters[4]. Table 1 shows that a Q function learned using DR with only simulated experience is unable to generalize to the true parameters during testing and MPQ has over 2X the success rate in BALLINCUPSPARSE and 3X in FRANKADRAWEROPEN.

5. Discussion

We presented a theoretical connection between information theoretic MPC and entropy-regularized RL that naturally provides an algorithm to leverage the benefits of both. While the approach is effective on a range of tasks, in the future we wish to investigate the dependence between model error and MPC horizon and adapt the horizon by reasoning about the quality of the Q function, both critical for real-world applications.

Table 1: Average success when training Q function using real system rollouts (ending with REAL) and DR. Test episodes=100. H is horizon of MPC in both training and testing, with $H = 1$ being SOFTQLEARNING

Task	Agent	Avg. success rate
BALLINCUPSPARSE (350 training episodes)	MPQH4REAL	0.85
	MPQH4DR	0.41
	MPQH1REAL	0.09
	MPQH1DR	0.06
FRANKADRAWEROPEN (200 training episodes)	MPQH10REAL	0.53
	MPQH10DR	0.17
	MPQH1REAL	0.0
	MPQH1DR	0.0

Acknowledgments

The authors would like to thank Nolan Wagener for insightful discussions for improving the manuscript.

References

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, pages 5360–5370, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 292–299. IEEE, 2013.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Vikash Kumar, Yuval Tassa, Tom Erez, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815. IEEE, 2014.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018. URL https://arxiv.org/abs/1801.00690.

Evangelos A Theodorou and Emanuel Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1466–1473. IEEE, 2012.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483, 2009.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Nolan Wagener, Ching-An Cheng, Jacob Sacks, and Byron Boots. An online learning approach to model predictive control. *arXiv preprint arXiv:1902.08967*, 2019.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 100–107. IEEE, 2013.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.