通过精确 Q-损失及其高斯－牛顿近似，从非线性模型预测控制中进行模仿学习

Andrea Ghezzi*, 1, Jasper Hoffman*, 2, Jonathan Frey[1,3],Joschka Boedecker[2],Moritz Diehl[1,3]

arXiv:2304.01782v1 [cs.LG] 3 Apr 2023

摘要——**本文提出一种新颖的损失函数，用于通过模仿学习获得非线性模型预测控制策略。标准的模仿学习方法忽略了有关专家的信息，通常采用基于专家控制与学习所得控制之间距离的损失函数。本文提出一种直接基于 Q 函数的损失，将相关最优控制问题（OCP）的性能目标和约束满足情况纳入其中。然而，使用 Q 损失训练神经网络需要针对每个新样本求解相应的 OCP。为减轻计算负担，我们基于 OCP 的高斯－牛顿近似推导出第二种 Q 损失，从而缩短训练时间。我们在带约束的非线性系统控制任务中，将所提出的损失与模仿学习的标准方法——行为克隆——进行比较验证。最终结果表明，基于 Q 函数的损失显著减少了约束违反，同时实现了相当或更低的闭环代价。**

I. 引言

模型预测控制（MPC）是一种基于优化的动力系统控制方法 [1]。由于其灵活性、处理约束的能力以及对稳定性的保证，它被广泛应用于各种领域。对于每个新的状态，MPC 都通过求解一个包含性能目标、系统动力学和系统属性的最优控制问题（OCP）来计算控制量。然而，当需要达到特定控制频率，或在计算资源受限的嵌入式系统上运行时，实时优化的计算复杂度可能会严重限制 MPC 的适用性。

解决这一问题的一种方法是显式模型预测控制。不同于通过在线优化推导出的隐式控制策略，显式模型预测控制会在离线环境中预先计算控制策略，之后只需在线评估即可。对于线性模型预测控制，可以用分段仿射函数表示状态反馈策略，并将相应的增益存储为查找表。然而，对于非线性模型预测控制（NMPC），无法精确表示该策略，现有方法也会产生近似误差。因此，通常的做法是依赖在线隐式计算控制策略。

与显式 模型预测控制相关的一种有前景的方法是模仿学习（IL）。这种方法试图用神经网络等参数化策略来模仿模型预测控制策略的行为。其主要问题是在模仿过程中会引入近似误差。另一方面，即使在非线性模型预测控制的情形下，模仿学习也能大幅降低在线计算成本。以往的研究主要采用行为克隆（BC）等方法来模仿模型预测控制策略。然而，行为克隆所使用的损失函数只是一种替代损失，它最小化模型预测控制策略与学习型策略之间的差异 [2]。采用这种替代损失，我们会失去模型预测控制最初为何采取某个控制动作的所有信息。因此，在训练过程中，学习型策略无法获得有关约束满足和性能目标的反馈。

在这项工作中，我们不再像行为克隆那样采用替代损失，而是提出一种新的损失形式，使策略在训练期间能够获知模型预测控制的内在目标。为此，我们引入一个 Q 函数，它对应于给定初始状态和固定初始控制时，相关最优控制问题在最优解处的代价。

具体而言，我们的贡献有两点：

1) 我们提出一种基于给定最优控制问题（OCP）的 Q 函数的模仿学习损失，使该损失能够直接融入最优控制问题的特性；2) 我们引入 Q 函数损失的二次规划近似，以降低计算负担，并使其更适合训练神经网络。

我们在非线性小车倒立摆系统的稳定控制任务上，将两种所提损失函数与行为克隆（BC）进行比较。与行为克隆策略相比，使用 Q 函数损失学习得到的策略实现了更低的代价，且约束违反情况显著减少。这一结果令人鼓舞，因为我们预计，在更复杂的示例中，所提损失与标准损失之间的潜在性能差异会更加明显。

相关工作：对于显式线性模型预测控制，可以通过分段仿射函数精确表示策略 [3], [4]。通过对这类策略进行近似，显式模型预测控制已扩展到非线性情形 [5]。已有研究探讨了使用神经网络表示最优控制策略 [6]，并将其应用于化工过程控制 [7]。得益于深度学习的成功，使用神经网络进行显式模型预测控制的研究日益增多，

* 这些作者贡献相同。[1] 弗赖堡大学微系统工程系（IMTEK），德国弗赖堡 79110[2] 弗赖堡大学计算机科学系[3] 弗赖堡大学数学系，德国弗赖堡 79104 通讯作者：{andrea.ghezzi@imtek.uni-freiburg.de, hoffmaja@informatik.uni-freiburg.de}本研究由德国研究联合会（DFG）通过研究组 FOR 2401 和项目 424107692 资助，并由欧盟通过 ELO-X 953348 资助。

©2023 IEEE。允许个人使用本材料。对于任何其他用途，包括在当前或未来媒体中重印或再版本材料用于广告或宣传目的、创作新的合集作品、转售或再分发至服务器或列表，或在其他作品中重用本作品的任何受版权保护部分，均须事先获得 IEEE 的许可。

近期应用包括电力电子、建筑控制和机器人操作臂 [8]–[10]。

许多研究已开始探究这些学习型控制器的性质与保证。在 [11] 中，通过强化学习训练近似线性模型预测控制器，并通过将控制投影到系统的控制不变集上，保证控制可行。对于非线性模型预测控制， [12] 基于学习型模型预测控制的近似误差条件，推导出稳定性和约束满足的统计保证。在 [13]，中，研究者引入控制屏障函数，将专家的安全性迁移到学习型控制器，并提供输入到状态稳定性的形式化保证。在 [14] 中，轨迹不仅针对代价进行优化，还要考虑学习型策略能否复现这些轨迹。然而，上述方法均基于行为克隆，因此它们最小化的是替代损失函数。

另一类与本文密切相关的研究关注从模型预测控制进行模仿学习，见 [15] 和 [16]，，其中模仿学习被重新表述为最小化连续时间最优控制问题表述中的控制哈密顿量。与我们的方法相比，其一项主要区别在于，除了我们研究的是离散时间最优控制问题之外，采用哈密顿量时，不必针对每个新控制重新求解最优控制问题，而只需使用代价到达函数关于状态的梯度。为将不等式约束的信息纳入该梯度，他们在最优控制问题表述中引入了对数障碍函数。

A. 符号说明

给定 $a \in \mathbb{R}^{n_a}$ 和 $b \in \mathbb{R}^{n_b}$，我们用 $c = (a, b)$表示向量$c = [a^\top \ b^\top]^\top$ 。给定 $a \in \mathbb{R}^{n_a}$，对于每个 $W \in \mathbb{R}^{n_a \times n_a}$，我们用$\|a\|_W^2 := a^\top W a$表示正定矩阵。用 $\mathcal{U}(a, b)$表示边界分别为 a，b 的均匀分布。

II. 背景

本节将介绍最优控制和模仿学习所需的背景知识。

A. 最优控制

在本文中，我们希望近似非线性模型预测控制策略，这些策略由反复求解最优控制问题来定义。具体而言，本文将以下通用离散时间最优控制问题视为研究对象

$$\min_{\substack{x_0, u_0, s_0, \ldots, \\ u_{N-1}, x_N, s_N}} \quad \sum_{k=0}^{N-1} \tilde{L}(x_k, u_k, s_k) + \tilde{E}(x_N, s_N) \tag{1a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{1b}$$

$$x_{k+1} = f(x_k, u_k), k = 0, \ldots, N-1, \tag{1c}$$

$$h(x_k, u_k) \le s_k, \quad k = 0, \ldots, N-1, \tag{1d}$$

$$r(x_N) \le s_N, \tag{1e}$$

$$s_k \ge 0, \qquad k = 0, \ldots, N, \tag{1f}$$

包含 N 个射击区间。这里， $x_k \in \mathbb{R}^{n_x}$ 和 $u_k \in \mathbb{R}^{n_u}$分别表示状态和控制轨迹，它们遵循式（1c）中的可能为非线性的系统动力学 f。为了保证可行性，我们引入松弛变量$s_k \in \mathbb{R}^{n_{s,\ k}}$并在代价函数中对其使用进行惩罚。因此，阶段代价定义为 $\tilde{L}(x_k, u_k, s_k) := L(x_k, u_k) + z^\top s_k + \|s_k\|_Z^2$，终端代价定义为 $\tilde{E}(x_N, s_N) := E(x_N) + z_e^\top s_N + \|s_N\|_{Z_e}^2$。不等式（1d）施加路径约束，（1e）则编码了关于系统终端状态的条件。我们假设函数 L，E，f，h，r 关于各自变量二阶连续可微。请注意，对于某些正松弛惩罚项取值 z，Z ，可以实现对约束的精确惩罚 [17]。在约束满足至关重要的场景中，我们可以调整松弛变量的权重，使可行性优先于最优性。在非线性模型预测控制方案中，每个控制步都会针对新的初始状态 $\bar{x}_0$ 求解一次最优控制问题（1），并将最优控制$u_0^\star(\bar{x}_0)$施加于系统。

请注意，为方便记号，后续所有最优控制问题表述均省略松弛变量。

B. 模仿学习

我们希望利用模仿学习（IL）来模仿通过求解式（1）所述离散时间最优控制问题而得到的控制律。我们可以将待模仿的专家策略定义为：针对给定的 $\bar{x}_0$，求解结果中的第一个最优控制 $\pi^\star(x) := u_0^\star(\bar{x}_0)$。我们的目标是通过参数化策略 $\pi(\cdot; w) : \mathbb{R}^{n_x} \to R^{n_u}$ 尽可能准确地逼近 $\pi^\star$ 。例如，参数化策略可以是一个神经网络，其中参数 $w \in \mathbb{R}^{n_w}$ 是神经网络的权重。

下面我们将简要介绍模仿学习框架。模仿学习目标可定义为

$$\mathcal{L}(w) := \mathbb{E}_{x \sim \mathcal{D}} \left[\ell \left(x, \pi(\cdot\,; w) \right) \right], \tag{2}$$

其中， $\mathcal{D}$ 是定义在 $\mathbb{R}^{n_x}$ 上的给定状态分布，而 ℓ 是给定状态 x [2]下策略 $\pi(\cdot\,; w)$的逐点损失函数。模仿学习（IL）的最终目标是找到参数 $w^\star$ 的最优组合，使期望损失 $\mathcal{L}(w)$最小化：

$$w^\star = \arg\min_w \quad \mathcal{L}(w). \tag{3}$$

在最一般的形式下，模仿学习（IL）假设对专家策略的内部目标一无所知。例如，专家可能是人类。因此，行为克隆等方法会用替代损失函数 ℓ 取代内部目标，该函数用于衡量策略 π 与专家策略 $\pi^\star$之间的行为差异。常见的选择包括定义如下的二次损失函数 ℓ_2 ：

$$\ell_2(x, \pi) := \left(\pi(x) - \pi^\star(x) \right)^2, \tag{4}$$

Huber 损失 [18]，、 ℓ_1 损失，或随机策略情况下的交叉熵损失。本文将使用二次损失函数 ℓ_2 进行比较，由此得到以下期望二次损失：

$$\mathcal{L}^2(w) := \mathbb{E}_{x \sim \mathcal{D}} \left[\ell_2 \left(x, \pi(\cdot\,; w) \right) \right]. \tag{5}$$

III. 模仿学习的 Q损失

接下来我们将看到，当专家策略是某个最优控制问题（OCP）的解时，我们不一定需要替代损失。

A. 精确 Q-损失

事实上，我们可以直接基于专家的内部目标定义一种损失，称之为精确 Q-损失。这个损失函数直接嵌入了最优控制问题（OCP）中包含的信息，例如其代价和约束。其核心思想是重用最优控制问题表述 (1)，并将最优控制问题（OCP）的第一个控制量 u_0 固定为策略返回的值 $\bar{u}_0 = \pi(x; w)$。通过求解由此得到的最优控制问题，我们可以为策略输出 $\pi(x; w)$赋予一个代价。

具体而言，我们重新表述最优控制问题（1），使其仅显式包含第一个控制。给定 $\bar{x}_0$，我们将精确 Q-损失定义为以下“Q函数最优控制问题”

$$
\begin{aligned}
Q(\bar{x}_0, \bar{u}_0) \coloneqq \min_{\substack{x_0, u_0, \ldots, \\ u_{N-1}, x_N}} \quad & \sum_{k=0}^{N-1} L\left(x_k,\ u_k\right) + E\left(x_N\right) && \text{(6a)} \\
\text{s.t.} \quad & \\
& x_0 - \bar{x}_0 = 0, && \text{(6b)} \\
& u_0 - \bar{u}_0 = 0, && \text{(6c)} \\
& x_{k+1} - f(x_k, u_k) = 0, && \text{(6d)} \\
& h(x_k, u_k) \leq 0,\ k = 1, \ldots, N-1, && \text{(6e)} \\
& r(x_N) \leq 0. && \text{(6f)}
\end{aligned}
$$

提醒一下，为了记号上的方便，我们在最优控制问题表述中省略了松弛变量。使用精确 Q-损失后，模仿学习目标变为

$$
\mathcal{L}^{\mathrm{Q}}(w) \coloneqq \mathbb{E}_{x \sim \mathcal{D}} \left[Q(x, \pi(x; w)) \right]. \tag{7}
$$

“Q-损失”这一名称源于强化学习中相关的 Q函数概念。$\mathcal{L}^{\mathrm{Q}}(w)$ 的梯度定义为

$$
\nabla_w \mathcal{L}^{\mathrm{Q}}(w) = \mathbb{E}_{x \sim \mathcal{D}} \left[\nabla_w \pi(x; w)\ \nabla_u Q(x, u)\big|_{u=\pi(x;w)} \right]. \tag{8}
$$

引理 *1*：Q损失的梯度由与约束（6c）对应的拉格朗日乘子 λ_u 给出，该约束对应于最优解

$$
\bar{\lambda}_u = \nabla_u Q(x, u)\big|_{u=\pi(x;w)}. \tag{9}
$$

证明：这可以通过考察动态规划中的代价到达函数以及最优性的一阶必要条件推导得出，[1, §8.8.3], [19, §3.3.3]. □

引理 *2*：如果 $\pi^\star(\bar{x}_0)$ 是（1）的唯一最小化解，那么对于任意 $\bar{u}_0 \neq \pi^\star(\bar{x}_0)$，$Q(\bar{x}_0,\ \bar{u}_0) > Q(\bar{x}_0,\ \pi^\star(\bar{x}_0))$。因此，精确 Q-损失会惩罚 $\bar{u}_0$ 偏离 $\pi^\star(\bar{x}_0)$ 的任何情况。

考察式 (7) 时，可以发现它与行动者-评论家算法 [20] 之间存在一个有趣的联系：行动者是策略 π，由一个称为评论家的参数化价值函数 Q进行评判。在这一框架中，由最优控制问题推导出的 Q 函数可以视为评论家。从这一视角来看，式 (8) 的梯度与确定性策略梯度 [21]直接相关。

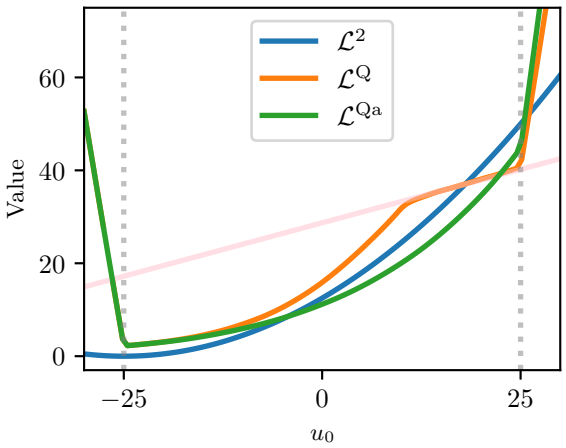

图 1. 对于给定的初始状态 $\bar{x}_0 = (0.8, 0, \pi/4, 0)$ 和 $u_0^\star(\bar{x}_0) = -25$，比较第 IV 节所述示例中的损失。粉色曲线对应于 $\mathcal{L}^{\mathrm{Q}}$ 在 $\bar{u}_0 = 15$处的梯度。

B. 精确 Q-损失的讨论

精确 Q-损失的计算复杂度高于标准损失。为了更好地了解计算代价，可以比较行为克隆和精确 Q-损失的梯度。对于行为克隆损失 $\mathcal{L}^2$，其梯度为

$$
\nabla_w \mathcal{L}^2(w) = \mathbb{E}_{x \sim \mathcal{D}} \left[\nabla_w \pi(x; w)(\pi(x; w) - \pi^\star(x)) \right]. \tag{10}
$$

比较两种损失的梯度，首先注意到，二者都需要对 π 进行一次前向和反向传播，以计算 $\pi(x; w)$ 以及策略 $\nabla_w \pi(x; w)$ 的梯度。精确 Q损失 (8) 还需要梯度$\nabla_u Q(x, u)\big|_{u=\pi(x_i;w)}$，它取决于 x 和 u。因此，每得到一个新样本 x，以及每个新的控制 u，我们都需要求解最优控制问题 (6)。请注意，求解该最优控制问题时，我们通过乘子得到梯度（参见引理 1）。

由于我们可能会针对任意一对（x，$\pi(x; w)$) 计算损失 (7)，因此需要通过引入松弛变量来软化约束，以确保问题 (6) 的可行性，如问题 (1) 所示。

最后，我们指出，根据 (6) 的性质，函数 Q 可能是分段非线性且非凸的。由于 u_0 的非凸性，在给定点 ($\bar{x}_0$，$\bar{u}_0$) 处由 Q 的梯度得到的切平面可能并非 $Q(\bar{x}_0$, $u_0^\star$) 的下界，图 1 给出了一个例子。这可能会减缓损失 (7) 的最小化过程。

在下一节中，我们通过提出其凸近似，来解决精确 Q-损失的计算复杂度问题。

C. Gauss-Newton Q-损失

我们提出了一种简化的损失，旨在减轻与 $\mathcal{L}^{\mathrm{Q}}$ 相关的计算负担，以及由 Q 的非凸性所产生的可能误导性梯度。该损失利用了 $\mathcal{L}^{\mathrm{Q}}$ 的最优控制结构，但

它围绕最优解为 Q-函数 OCP (6) 构建了一个二次规划近似。

假设 *1*：考虑 (1a) 中的函数 L 和 E，它们分别是如下形式的向量值函数 $\|\bar{L}\|^2$的平方，其中 $\bar{L}$：$\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$，$\|\bar{E}\|^2$ 满足 $\bar{E}$：$\mathbb{R}^{n_x} \to \mathbb{R}^{n_e}$。这类代价函数在最优控制中十分常见，因为它们跟踪代价；更重要的是，它们允许采用高斯-牛顿 Hessian 近似 [22]。

给定一个样本 $\bar{x}_0$ 和第一个控制量 $\bar{u}_0$，我们求解 (1)，并将其解记为 $\zeta = (\tilde{x}_0, \tilde{u}_0, \ldots, \tilde{u}_{N-1}, \tilde{x}_N)$，并将其用作问题 (6) 的二次近似的线性化点，如下所示

$$
\begin{aligned}
Q_{\mathrm{a}}(\bar{x}_0, \bar{u}_0) \coloneqq \min_{\substack{x_0,u_0,\ldots,\\ u_{N-1},x_N}} \quad & \sum_{k=0}^{N-1} \|L_{\mathrm{L}}(x_k, u_k; \tilde{x}_k, \tilde{u}_k)\|^2 + \|E_{\mathrm{L}}(x_N; \tilde{x}_N)\|^2 \\
\text{s.t.} \quad & x_0 - \bar{x}_0 = 0, \\
& u_0 - \bar{u}_0 = 0, \\
& x_{k+1} - f_{\mathrm{L}}(x_k, u_k; \tilde{x}_k, \tilde{u}_k) = 0, \\
& h_{\mathrm{L}}(x_k, u_k; \tilde{x}_k, \tilde{u}_k) \le 0,\ k = 1, \ldots, N-1, \\
& r_{\mathrm{L}}(x_N; \tilde{x}_N) \le 0,
\end{aligned}
\tag{11}
$$

其中 $L_{\mathrm{L}}(x_k, u_k; \tilde{x}_k, \tilde{u}_k)$ 定义为 L 在 $(\tilde{x}_k, \tilde{u}_k)$ 处的一阶泰勒级数，如下所示

$$
\begin{aligned}
L_{\mathrm{L}}(x_k, u_k; \tilde{x}_k, \tilde{u}_k) = {} & L(\tilde{x}_k, \tilde{u}_k) \\
& + \nabla_{x,u} L(\tilde{x}_k, \tilde{u}_k)^\top \left(\begin{bmatrix} x \\ u \end{bmatrix} - \begin{bmatrix} \tilde{x}_k \\ \tilde{u}_k \end{bmatrix} \right).
\end{aligned}
\tag{12}
$$

函数 $f_{\mathrm{L}}, h_{\mathrm{L}}$ 以相同方式定义；而对于$E_{\mathrm{L}}, r_{\mathrm{L}}$，线性化在 $\tilde{x}_N$ 处进行，且仅针对 x。函数 Q_{a} 现在由一个凸分段二次函数描述。

引理 *3*：如果我们使用 (1) 的最优解作为线性化点，即设 $(\bar{x}_0, \bar{u}_0) \coloneqq (\bar{x}_0, \pi^\star(\bar{x}_0))$，那么对于每个 x, u，Gauss-Newton Q-损失都是一个凸距离函数，且$Q_{\mathrm{a}}(x, u) \ge Q_{\mathrm{a}}(x, \pi^\star(x))$。

我们可以将近似 Q 损失定义为

$$
\mathcal{L}^{Q_{\mathrm{a}}}(w) \coloneqq \mathbb{E}_{x \sim \mathcal{D}} \left[Q_{\mathrm{a}}(x, \pi(x; w)) \right], \tag{13}
$$

其梯度 $\nabla \mathcal{L}^{Q_{\mathrm{a}}}(w)$为

$$
\nabla \mathcal{L}^{Q_{\mathrm{a}}}(w) = \mathbb{E}_{x \sim \mathcal{D}} \left[\nabla_w \pi(x; w)\ \nabla_u Q_{\mathrm{a}}(x, u; \zeta)|_{u=\pi(x;w)} \right]. \tag{14}
$$

与 (8) 相比，函数 $\nabla \mathcal{L}^{Q_{\mathrm{a}}}$ 需要二次规划问题的梯度，即 $\nabla_u Q_{\mathrm{a}}$，而这部分的计算成本较低。

IV. 数值示例

我们以小车倒立摆为例，展示所提损失相较于标准 ℓ_2 损失的有效性。系统如图 2 所示。任务是控制系统，使摆杆保持直立，小车保持在轨道中心。

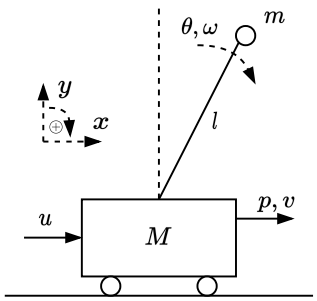

图 2. 小车倒立摆示意图

忽略摩擦力后，系统的动力学由以下方程定义

$$
\begin{bmatrix} \dot{p}(t) \\ \dot{v}(t) \\ \dot{\theta}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} v \\ \frac{-ml\sin(\theta)\dot{\theta}^2 + mg\cos\theta\sin(\theta) + u}{M + m - m\cos^2(\theta)} \\ \omega \\ \frac{-ml\cos(\theta)\sin(\theta)\dot{\theta}^2 + (M+m)g\sin(\theta) + u\cos(\theta)}{l \cdot (M + m - m\cos^2(\theta))} \end{bmatrix}, \tag{15}
$$

其中 $l = 0.8$ (m) $m = 0.1$ (kg)、$M = 1$ (kg)、$g = 9.81$ (m/s^2)。系统状态为 $x(t) = (p(t)\ v(t)\ \theta(t)\ \omega(t)) \in \mathbb{R}^4$，控制为 $u(t) \in \mathbb{R}$。我们假设系统状态完全可观测。

我们采用包含 $N = 20$ 个射击区间、每个区间长度为 $\Delta t = 0.05(s)$ 的多重射击法，并使用 RK4 积分器对式 (15) 进行离散化，从而得到离散时间最优控制问题

$$
\begin{aligned}
\min_{\substack{x_0,u_0,\ldots,\\ u_{N-1},x_N}} \quad & \frac{1}{N} \left(\sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + x_N^\top P x_N \right) \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, \ldots, N-1, \\
& \underline{x} \le x_k \le \bar{x} \qquad k = 0, \ldots, N, \\
& \underline{u} \le u_k \le \bar{u} \qquad k = 0, \ldots, N-1, \\
& x_0 = \bar{x}_0.
\end{aligned}
\tag{16}
$$

其时域长度为 $\bar{x} = (2.4, \frac{\pi}{3}, 2)$ $x = -\bar{x}$，且 $\bar{u} = 25$，$u = -\bar{u}$。代价函数中的权重矩阵为$S = \mathrm{diag}(0.25, 0.025, 0.25, 0.025)$，$R = 0.0025$，而 P对应于系统在 $\bar{x} = (0, 0, 0, 0)$ $\bar{u} = 0$ 处线性化后所得的离散代数黎卡提方程的解。为保证训练过程中的可行性，对 (x, u) 上的盒约束通过松弛变量进行软化，并在代价函数中对其加权惩罚：路径约束对应的权重为 $Z = \Delta t \cdot (50, 5, 50, 5, 500)$ $z = \Delta t \cdot (0.5, 0.05, 0.5, 0.05, 5000)$，终端约束对应的权重为 $Z_e = (50, 5, 50, 5)$ $z_e = (0.5, 0.05, 0.5, 0.05)$。

根据（6）和（11）修改最优控制问题表述（16），即可分别得到Q函数最优控制问题和近似Q函数最优控制问题。

最优控制问题的建模与求解通过 acados [23] 的 Python接口完成。为了加快与最优控制问题求解器的交互，我们使用了已编译的求解器对象 Cython接口。所有计算

均仅使用一个中央处理器线程，在一台运行 Linux Ubuntu 20.04 的服务器上进行。该服务器配备 Intel Xeon E5-2687W @3.1 GHz、16 核和 32 GB 随机存取存储器。

A. 训练设置

为近似 MPC策略，我们使用带有 ReLU 激活函数的前馈神经网络作为隐藏层。此外，在最后一个线性层之后，我们应用 tanh 激活函数，以限制策略的输出，从而满足倒立摆最优控制问题表述 (16) 中控制量的盒约束。我们使用 Adam 优化器 [24]，通过小批量随机梯度下降来优化神经网络。

接下来一个重要的决策是：我们要模仿哪些状态，或者更具体地说，要针对哪种状态分布 $\mathcal{D}$最小化模仿损失。为此，我们首先从 $\mathcal{U}(\alpha \cdot x,\ \alpha \cdot \bar{x})$ 中均匀采样一个初始状态，满足$\alpha = 0.3$。我们持续采样并丢弃样本，直到抽取的初始状态能够生成无需放松约束的最优开环轨迹。然后，从该初始状态出发，按照 [25] 中的描述，使用 Dagger 算法进行滚动执行。滚动执行时，我们不只遵循专家模型预测控制策略，而是使用一种混合策略，随机采用专家模型预测控制策略或当前学习型策略 $\pi(x; w)$ 给出的控制量。这样做是为了生成与应用最终学习型策略时所遇到的状态分布更相符的样本。具体而言，我们对神经网络进行 2000 次更新，并每更新 20 次就通过执行 50 步滚动来收集额外样本。在训练过程中，专家策略与学习型策略之间的混合系数从 1 线性递减至 0：系数为 1 时仅使用专家策略，系数为 0 时仅使用学习型策略。

B. 评估

在评估中，我们使用 10 个不同的随机种子分别训练每种算法。对于初始状态，我们选择从包含较容易初始状态的均匀分布 $\alpha = 0.2$ 中采样、从与训练期间相同的分布 $\alpha = 0.3$ 中采样，或选择较难的初始状态 $\alpha = 0.4$。这样做是为了观察较容易和较难初始状态下的性能差异，并检验不同损失的泛化能力。我们为所有算法和随机种子生成一个固定的测试数据集，其中包含 2000 个初始状态，采样过程与训练期间相同。对于每个初始状态，我们随后使用每个学习型策略进行 50 步滚动执行。

我们使用两项指标评估最终策略的表现：（1）平均滚动成本：将一次滚动执行中的所有阶段成本和松弛变量成本相加，再对所有滚动轨迹和随机种子取平均。（2）平均违反率，即违反约束的滚动轨迹占所有滚动轨迹的比例，并对所有随机种子取平均。此外，当策略无法使系统稳定时，我们仅考虑 0.99 或 0.9 分位数，以提高估计对异常值的稳健性。

表 I 损失性能比较

Loss	α	分位数	平均成本	违反率
$\mathcal{L}^2$	0.2	0.99	0.866 ± 0.013	0.054 ± 0.042
	0.3	0.99	2.267 ± 0.250	0.225 ± 0.064
	0.4	0.90	3.229 ± 0.385	0.299 ± 0.064
$\mathcal{L}^{\mathrm{Q}}$	0.2	0.99	0.919 ± 0.093	0.005 ± 0.012
	0.3	0.99	2.308 ± 0.209	0.090 ± 0.049
	0.4	0.90	3.267 ± 0.365	0.147 ± 0.059
$\mathcal{L}^{\mathrm{Q_a}}$	0.2	0.99	0.884 ± 0.019	0.002 ± 0.001
	0.3	0.99	2.179 ± 0.081	0.088 ± 0.024
	0.4	0.90	3.076 ± 0.150	0.154 ± 0.025
$\pi^\star$ 模型预测控制	0.2	0.99	0.795 ± 0.833	–
	0.3	0.99	1.888 ± 2.027	–
	0.4	0.90	2.604 ± 2.373	–

C. 超参数

为便于更好地比较，我们通过网格搜索进行超参数搜索，搜索范围包括网络深度$\{1, 2, 3\}$、网络宽度 $\{64, 128, 256\}$ 和学习率 $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$。我们采用固定的批大小 32。Gauss-Newton Q-损失和精确 Q-损失使用相同的超参数。搜索目标是上一节所述同一初始状态数据集上的平均滚动成本，但仅使用 3 个随机种子。

D. 结果

下面我们给出实验结果。在表 I 中，我们比较了不同算法的性能。就平均滚动成本而言，精确 Q-损失的表现略逊于行为克隆，而在较难的初始状态数据集 $\alpha = 0.4$ 上，Gauss-Newton Q-损失的表现明显更好。然而，对于较难的样例，与原始 MPC 策略之间的差距也明显更大。一种解释是，对于较难的样例，最优控制策略的非光滑性会增强，使得网络更难以逼近 MPC，行为克隆尤其如此。

就平均违反率而言，我们看到，在 $\mathcal{L}^2$ 的情况下，精确 Q-损失和 Gauss-Newton Q-损失的表现明显优于行为克隆。这可归因于精确 Q-损失和 Gauss-Newton Q-损失包含约束满足项，且不依赖替代损失。

在图 3 中，我们比较了针对一个随机种子训练的神经网络在示例滚动轨迹上，使用不同算法学习到的策略的控制效果。可以看到，原始模型预测控制呈现出非常不平滑的控制信号，而行为克隆损失 $\mathcal{L}^2$ 试图对此进行模仿。对应于精确 Q-损失和 Gauss-Newton Q-损失的策略则呈现出更平滑的控制信号，且与原始模型预测控制的差异更大。这可以解释为，所提损失直接优化最优控制问题的（近似）形式，因此会自行权衡可行性与最优性。

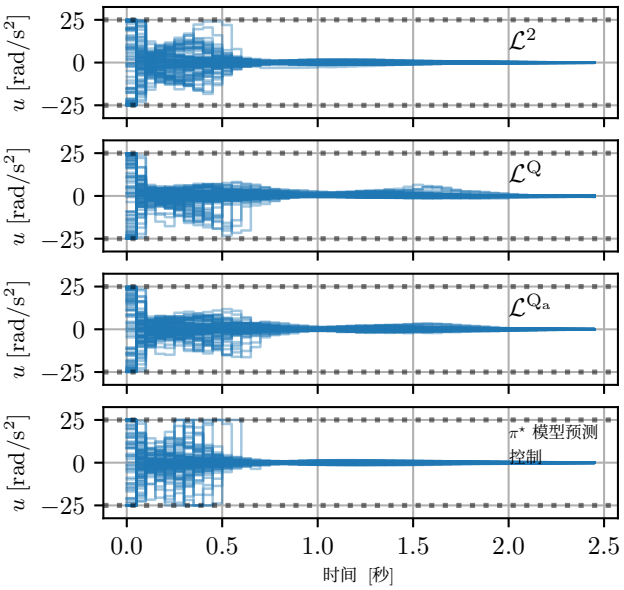

图3. 一个网络的 $\alpha = 0.3$ 代表性100条滚动轨迹。

表II 平均梯度计算速度

Loss	$\mathcal{L}^2$	$\mathcal{L}^{\mathrm{Q}}$	$\mathcal{L}^{\mathrm{Q_a}}$
速度（批次/秒）	151.78	4.12	15.23

在表II中，我们比较了计算一个批次（批大小为 32）的梯度所需的平均时间，共进行 300 次迭代。

V. 结论

本文提出了一种基于底层最优控制问题（OCP）的新型损失，用于从模型预测控制（MPC）中进行模仿学习。该损失使学习型策略能够直接最小化 OCP 的性能目标并满足约束。与标准损失相比，评估 Q损失时，每个新样本都需要求解一个可能为非线性且非凸的优化问题，因此计算量较大。为缓解这一问题，我们提出了另一种 Q损失，它基于相关 OCP 的高斯－牛顿近似，因此评估时只需解一个凸二次规划。最后，我们在一个受约束非线性系统的控制任务中，将使用 Q损失训练的策略与行为克隆进行了比较。在该示例中，基于 Q损失的策略显著减少了约束违反，同时实现了相当的闭环代价。未来，我们计划在更复杂的示例中测试这些损失，并将其与强化学习相结合。

参考文献

[1] J. B. Rawlings、D. Q. Mayne 和 M. M. Diehl，模型预测控制：理论、计算与设计，第2版。Nob Hill，2017年。[2] T. Osa、J. Pajarinen、G. Neumann、J. A. Bagnell、P. Abbeel、J. Peters 等，“模仿学习的算法视角”，*Foundations and Trends® in Robotics*，第7卷，第1–2期，第1–179页，2018年。[3] A. Bemporad、F. Borrelli 和 M. Morari，“基于受约束线性规划的滚动时域控制的显式解”，载于*IEEE*决策与控制会议（*CDC*）论文集，澳大利亚悉尼，1999年。

[4] A. Bemporad、M. Morari、V. Dua 和 E. N. Pistikopoulos，“受约束系统的显式线性二次调节器”，Automatica，第38卷，第3–20页，2002年。[5] T. A. Johansen，“受约束非线性系统的近似显式滚动时域控制”，Automatica，第40卷，第2期，第293–300页，2004年。[6] T. Parisini 和 R. Zoppoli，“非线性系统的滚动时域调节器及其神经网络近似”，Automatica，第31卷，第10期，第1443–1451页，1995年。[7] B. M. ˚Akesson 和 H. T. Toivonen，“神经网络模型预测控制器”，Journal of Process Control，第16卷，第9期，第937–946页，2006年。[8] S. Lucia、D. Navarro、B. Karg、H. Sarnago 和 O. Lucia，“用于谐振功率变换器的基于深度学习的模型预测控制”，IEEE工业信息学汇刊，第17卷，第1期，第409–420页，2020年。[9] J. Drgoˇna、A. Tuor、E. Skomski、S. Vasisht 和 D. Vrabie，“面向建筑物的深度学习显式可微预测控制律”，IFAC-PapersOnLine，第54卷，第6期，第14–19页，2021年。[10] J. Nubert、J. K¨ohler、V. Berenz、F. Allg¨ower 和 S. Trimpe，“机器人机械臂上的安全快速跟踪：鲁棒模型预测控制与神经网络控制”，IEEE机器人与自动化快报，第5卷，第2期，第3050–3057页，2020年。[11] S. Chen、K. Saulnier、N. Atanasov、D. D. Lee、V. Kumar、G. J. Pappas 和 M. Morari，“使用受约束神经网络近似显式模型预测控制”，载于2018年美国控制会议（ACC），IEEE，2018年，第1520–1527页。[12] M. Hertneck、J. K¨ohler、S. Trimpe 和 F. Allg¨ower，“带保证的近似模型预测控制器学习”，IEEE控制系统快报，第2卷，第3期，第543–548页，2018年。[13] R. K. Cosner、Y. Yue 和 A. D. Ames，“利用控制屏障函数实现带安全保证的端到端模仿学习”，载于2022年IEEE第61届决策与控制会议（CDC），IEEE，2022年，第5316–5322页。[14] I. Mordatch 和 E. Todorov，“结合函数逼近与轨迹优化的优势”，载于Robotics: Science and Systems，第4卷，2014年，第23页。[15] J. Carius、F. Farshidian 和 M. Hutter，“MPC-Net：由第一性原理引导的策略搜索”，IEEE机器人与自动化快报，第5卷，第2期，第2897–2904页，2020年。[16] A. Reske、J. Carius、Y. Ma、F. Farshidian 和 M. Hutter，“用于四足机器人多步态控制的基于模型预测控制的模仿学习”，载于2021年IEEE国际机器人与自动化会议（ICRA），IEEE，2021年，第5014–5020页。[17] R. H. Byrd、J. Nocedal 和 R. A. Waltz，“非线性规划的精确罚函数方法的引导”，Optimization Methods and Software，第23卷，第2期，第197–213页，2008年。[18] P. J. Huber，“位置参数的鲁棒估计”，The Annals of Mathematical Statistics，第35卷，第1期，第73–101页，1964年。[19] D. Bertsekas，《动态规划与最优控制：第I卷》。Athena Scientific，2012年，第1卷，第3版。[20] A. G. Barto、R. S. Sutton 和 C. W. Anderson，“能够解决复杂学习控制问题的类神经元自适应元件”，IEEE系统、人类与控制论汇刊，第5期，第846页，1983年，第834页。[21] D. Silver、G. Lever、N. Heess、T. Degris、D. Wierstra 和 M. Riedmiller，“确定性策略梯度算法”，载于国际机器学习会议，PMLR，2014年，第387–395页。[22] R. Verschueren、N. van Duijkeren、R. Quirynen 和 M. Diehl，“利用直接最优控制中的凸性：一种序列凸二次规划方法”，载于IEEE决策与控制会议（CDC）论文集，2016年。[23] R. Verschueren、G. Frison、D. Kouzoupis、J. Frey、N. van Duijkeren、A. Zanelli、B. Novoselnik、T. Albin、R. Quirynen 和 M. Diehl，“acados——用于快速嵌入式最优控制的模块化开源框架”，Mathematical Programming Computation，2021年10月。[24] D. P. Kingma 和 J. Ba，“Adam：一种随机优化方法”，arXiv预印本 arXiv:1412.6980，2014年。[25] S. Ross、G. Gordon 和 D. Bagnell，“将模仿学习与结构化预测归约为无悔在线学习”，载于第十四届国际人工智能与统计会议论文集，JMLR研讨会与会议论文集，2011年，第627–635页。