OCS2 优化控制工具箱（五）部署训练 汉化

前言

阻碍 MPC 在机器人任务中广泛应用的主要挑战之一是设置最优控制问题的负担。OCS2 提供了多个辅助类，用于定义一些常用模型、成本和约束条件，以减轻这一问题。为此，OCS2 提供了多个第三方软件包接口，如 RobCoGen、CppADCodeGen、Pinocchio 和 HPP-FCL。我们在此重点介绍 Pinocchio 接口。本页讨论的所有软件包都可以在元软件包 ocs2_pinocchio 中找到。

一、从 URDF 到 OCP

1.1 中心模型

多关节浮动基座系统（如足式机器人）可建模为一个未驱动的刚体，该刚体上连接着一组完全驱动的肢体。在对机器人关节有足够控制权的温和假设下，在 MPC 公式中独立考虑中心点动力学作为简化模板模型是合理的。该模型的状态空间包括归一化中心动量、基本坐标和关节位置。输入空间是所有接触扭矩和关节速度的组合。为了捕捉广义坐标变化率对向心动量的影响，该模型使用了向心动量矩阵，并在基础扭转和关节速度之间引入了正确的映射。有关实现的更多详情，请参阅软件包 ocs2_centroidal_model。

1.2 运动学

OCS2 为 URDF 模型中基于 Pinocchio 库的任何已命名坐标系提供了一个运动学接口。该接口为已命名的坐标系列表提供位置、方向误差和速度的一阶模型。OCS2 提供两个接口： 基于分析偏差的 PinocchioEndEffectorKinematics 接口和基于自动微分的 PinocchioEndEffectorKinematicsCppAd 接口。当需要使用 OCS2 的缓存功能时，通常使用前者；否则，应使用 CppAd 变体。

1.3 避免自碰撞

OCS2 库提供了用于定义避免自碰撞约束的辅助类。通过 URDF 模型和用户定义的碰撞体列表可以方便地定义这些约束条件。该列表应是 URDF 模型碰撞体的子集。该列表用于避免在所有碰撞体之间进行碰撞检查，并减少计算开销。碰撞约束计算需要 HPP-FCL 和 Pinocchio 库。有关实现的更多详情，请参阅软件包 ocs2_self_collision。

二、MPC-Net

MPC-Net 是一种模仿学习方法，它使用 MPC 的解决方案来指导策略搜索。其主要思想是通过最小化控制哈密顿来模仿 MPC，同时用参数化策略来表示相应的控制输入。MPC-Net 可用于将模型预测控制器克隆为神经网络策略，其评估速度比 MPC 快得多。因此，MPC-Net 是 MPC 的有用替代品，适用于不需要最精确解决方案的计算要求较高的应用。

多线程数据生成和策略评估与策略训练异步运行。数据生成和策略评估用 C++ 实现，在 CPU 上运行，而策略训练用 Python 实现，在 GPU 上运行。控制哈密顿由线性二次近似表示。因此，训练可以在 GPU 上运行，无需回调在 CPU 上运行的 OCS2 C++ 代码来评估哈密顿方程，而且可以利用 GPU 上的批处理功能。

2.1 机器人

MPC-Net 已在以下机器人示例中实现：

Robot	Recom. CPU Cores	Recom. GPU Memory	RaiSim	Training Time
ballbot	4	2 GB	No	0m 20s
legged_robot	12	8 GB	Yes / No	7m 40s

2.2 设置

确保按照安装页面进行操作。按照所有相关说明进行安装。关于可选依赖项，请务必按照 ONNX Runtime 和虚拟环境的说明进行操作，并可选择设置 RaiSim。

要编译所有 MPC-Net 软件包，请编译元软件包：

```
cd <path_to_catkin_ws>
catkin_build ocs2_mpcnet

# Example:
cd ~/catkin_ws
catkin_build ocs2_mpcnet
```

要构建机器人专用软件包，请将 <robot> 替换为机器人名称：

```
cd <path_to_catkin_ws>
catkin_build ocs2_<robot>_mpcnet

# Example:
cd ~/catkin_ws
catkin_build ocs2_ballbot_mpcnet
```

2.3 训练

要训练 MPC-Net 策略，请运行

```
cd <path_to_ocs2_repo>/ocs2_mpcnet/ocs2_<robot>_mpcnet/python/ocs2_<robot>_mpcnet
source <path_to_catkin_ws>/devel/setup.bash
source <path_to_venvs>/mpcnet/bin/activate
python3 train.py

# Example:
cd ~/git/ocs2/ocs2_mpcnet/ocs2_ballbot_mpcnet/python/ocs2_ballbot_mpcnet
source ~/catkin_ws/devel/setup.bash
source ~/venvs/mpcnet/bin/activate
python3 train.py
```

要使用 Tensorboard 监控训练进度，请运行

```
cd <path_to_ocs2_repo>/ocs2_mpcnet/ocs2_<robot>_mpcnet/python/ocs2_<robot>_mpcnet
source <path_to_venvs>/mpcnet/bin/activate
tensorboard --logdir=runs

# Example:
cd ~/git/ocs2/ocs2_mpcnet/ocs2_ballbot_mpcnet/python/ocs2_ballbot_mpcnet
source ~/venvs/mpcnet/bin/activate
tensorboard --logdir=runs
```

如果使用 RaiSim，您可以使用 RaiSim Unity 对数据生成和策略评估滚动进行可视化，RaiSim 的 raisimUnity 文件夹中提供了预构建的可执行文件。例如，在 Linux 上运行

```
<path_to_raisimLib_repo>/raisimUnity/linux/raisimUnity.x86_64

# Example:
~/git/raisimLib/raisimUnity/linux/raisimUnity.x86_64
```

2.4 部署

要部署存储在机器人专用软件包策略文件夹中的默认策略，请运行

```
cd <path_to_catkin_ws>
source devel/setup.bash
roslaunch ocs2_<robot>_mpcnet <robot>_mpcnet.launch

# Example:
```

```
cd ~/catkin_ws
source devel/setup.bash
roslaunch ocs2_ballbot_mpcnet ballbot_mpcnet.launch
```

要部署存储在机器人专用软件包的 python/ocs2_<robot>_mpcnet/runs 文件夹中的新策略，请将 <path> 替换为最终策略的绝对文件路径，然后运行：

```
cd <path_to_catkin_ws>
source devel/setup.bash
roslaunch ocs2_<robot>_mpcnet <robot>_mpcnet.launch policyFile:=<path>

# Example:
cd ~/catkin_ws
source devel/setup.bash
roslaunch ocs2_ballbot_mpcnet ballbot_mpcnet.launch
policyFile:='/home/user/git/ocs2/ocs2_mpcnet/ocs2_ballbot_mpcnet/python/ocs2_ba
llbot_mpcnet/runs/2022-04-01_12-00-00_ballbot_description/final_policy.onnx'
```

2.5 如何设置新机器人

为新机器人设置 MPC-Net 相对容易，因为 ocs2_mpcnet_core 软件包负责数据生成和策略评估，并实现重要的学习组件，如内存、策略和损失函数。

本节假定您已经拥有机器人专用 MPC 实现的软件包：

1. ocs2_<robot>： 提供机器人专用 MPC 实现库。
2. ocs2_<robot>_ros： 用 ROS 封装 MPC 实现，以定义 ROS 节点。
3. ocs2_<robot>_raisim：（可选）机器人专用 MPC 实现与 RaiSim 之间的接口。

对于实际的 ocs2_<robot>_mpcnet 软件包，请遵循现有机器人专用 MPC-Net 软件包的结构。必须实现的最重要类/文件有

- <Robot>MpcnetDefinition： 定义如何将 OCS2 状态变量转换为策略观测值，以及如何将策略动作转换为 OCS2 控制输入。
- <Robot>MpcnetInterface： 提供 C++ 和 Python 之间的接口，允许交换数据和策略。
- <robot>.yaml： 存储配置参数。
- mpcnet.py： 为 MPC-Net 训练添加机器人专用方法，例如实现机器人应执行的任务。
- train.py： 启动主训练脚本。

2.6 已知问题处理

在处理机器人的模型预测控制（MPC）网络时，确实可能会遇到由于stiff的不等式约束导致的汉密尔顿量（Hamiltonian）及其梯度变得非常大的问题。这种情况可能会阻碍学习过程，导致策略无法学到有用的东西。为了解决这个问题，可以在机器人的MPC-Net的YAML配置文件中启用梯度裁剪（gradient clipping）功能，并调整梯度裁剪值。梯度裁剪是一种常用的技术，用于防止在训练深度神经网络时出现的梯度爆炸问题。通过限制梯度的最大值，可以帮助维持训练过程的稳定性，并确保模型可以有效地学习。

以下是如何在MPC-Net的YAML配置文件中启用梯度裁剪并调整其值的一般步骤：

1. 打开机器人的MPC-Net配置YAML文件。
2. 寻找与梯度裁剪相关的配置项。这通常会在配置文件的“training”或“optimizer”部分下。
3. 启用梯度裁剪功能，这通常可以通过设置一个键值对来完成，例如：

```
gradient_clipping: true
```

1. 调整梯度裁剪值。这可以通过设置一个键值对来指定梯度的最大允许值，例如：

```
max_gradient_norm: 1.0
```

1. 上述示例中，`1.0`是梯度的最大范数。根据具体情况，您可能需要调整这个值以达到最佳效果。
2. 保存配置文件并重新启动训练过程，以应用更改。 请注意，梯度裁剪值的调整可能需要一些实验来找到最佳值。您可能需要尝试不同的值，并通过观察训练过程中的损失函数和学习行为来评估其效果。 总之，通过在MPC-Net的YAML配置文件中启用梯度裁剪并适当调整梯度裁剪值，可以帮助解决由于严格的不等式约束导致的学习障碍，从而提高机器人学习过程的效率和效果。

参考文献

This part of the toolbox has been developed based on the following publications:

1. Jan Carius, Farbod Farshidian, and Marco Hutter. Mpc-net: a first principles guided policy search. *IEEE Robotics and Automation Letters*, 5(2):2897–2904, 2020.
2. Alexander Reske, Jan Carius, Yuntao Ma, Farbod Farshidian, and Marco Hutter. Imitation learning from mpc for quadrupedal multi-gait control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.