A Differentiable Physics Engine for Deep Learning in Robotics

Reminders

- Model based reinforcement learning vs optimal control
  - Quite similar, but optimal control assumes knowledge of state transitions, while the model based reinforcement learning tries to learn the transitions
  - When the transitions are learned, different optimization methods can be employed
- Usual methods for solving these problems are:
  1. Direct shooting - plug x' from system dynamics, and only optimize over u (e.g. LQR)
  2. Collocation methods - optimize over both x and u with constraints (e.g. numerical optimization using dual gradient descent)
- Both methods need $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial u}, \frac{\partial c}{\partial x}, \frac{\partial c}{\partial u}$ - where f is the system dynamics and c is the cost
  - This is usually approximated, and the higher the order - the better

Overview

- Engine for differentiating control parameters through system models
  - Previously finite differences and approximations used
- In control systems, robot is usally treated as a non-differentiable black box
  - In other words, we treat the robot as the environment, and given
- Mujoco also calculates gradients through the model, using finite difference - not as stable nor efficient
- Some implementation concerns (e.g. branching) that had to be taking into account for efficient GPU computations

Key ingredients

- The system allows us to efficiently optimize the control policy, by backpropagating the **analytic** cost gradients directly to control parameters

Comments

- What is the gain from using this, compared to Mujoco? What is the difference to Mujoco? We get access to the gradients?
  - Not only do we (apparently) get access to the gradients, the gradients are analytically computed
- How is this helpful for the model based approach? When doing reinforcement learning, we assume the model class, and try to learn it's parameters. What do these actual model derivatives give us? Some only applicable for control?
- How is this different from e.g. optimal control?
  - This seems like something that could be used with iLQG, etc, but the authors decided to use it with a custom policy
- What was preventing us to compute this earlier?
  - Earlier we had discrete numerical gradients, which were not stable enough
  - There were attempts, but only on smaller problems (mentioned in the paper)
- Lack of implementation details - hopefully the code will be published soon
- Not clear comparison to other methods - compares gradient free method, with this method, but does not argue about similarities, and potential uses with gradient based methods
- No discussion about uncertainty in the models