Project Milestone for CS234 (Winter 2019)
Generalized Locomotion Controllers

Li Quan Khoo [1] [2]

1. Change of project

First, we need to address the reason for the deviation of the final project topic from the initial project proposal. The original proposal was about reinforcement learning (RL) under incomplete information on the board game Stratego. For more detailed information about Stratego, please refer to the initial project proposal, as well as official competition rules (International Stratego Federation).

In short, there are several practical reasons behind this change. Also, after developing a slightly more mature understanding of RL, we decided to tackle a problem that is more relevant, and more broadly applicable.

- Stratego is an incomplete information game. However, under the RL framework, this just means the policy acts on observations rather than states, i.e. $\pi(s, a)$ becomes $\pi(o, a)$. There is only a small (immutable) set of pieces in Stratego, and we think the novelty of the problem is diminished under this perspective. Under the intended representation of the game state, the agent has access to information about its own pieces, the opponent's known and unknown pieces.

- There is a notable lack of recorded Stratego games. AlphaGo was bootstrapped initially by imitating expert play.

- The state-space of Stratego is 10^{115}, compared to chess's 10^{45}. For a one-man project, even if this problem is solvable in the allotted time, handling this complexity should not be the focus of a project of this scope. At least, this was not one of the goals in the original proposal, especially since the class will not be covering MCTS methods for exploring this state space until towards the end.

- Stratego as a game is not interesting in the sense that it doesn't act as any sort of benchmark for RL. For example, Go was novel because of its enormous state space (10^{360}), and Montezuma's Revenge is an important gauge for exploration. We find it difficult to see how the results would inform us about RL other than (possibly) solving the game itself. Even then there is no 'solution' such as in draughts in the traditional sense since the game is stochastic.

- Due to the lack of an existing implementation of Stratego that exposes an interface that allows for automated policy training etc., it is necessary to implement one (or hack one). After reading through competition rules, which is important for understanding the necessary representation of the state of the game (e.g. The state of the game in Go requires knowing the past 7 moves). Our conclusion is that even a simplified implementation is not the best use of time for this project, which should really be spent exploring RL itself.

[1]Computer Science Department [2]Stanford Center for Professional Development. Correspondence to: Li Quan Khoo <lqkhoo@stanford.edu, li.khoo.11@ucl.ac.uk>.

CS234 Reinforcement Learning (Winter 2019), Computer Science Department, Stanford University, Stanford California 94305 USA

2. Background

2.1. Locomotion policies under pure RL

For locomotion tasks trained entirely using RL techniques, we tend to notice several things that could be improved. In particular, training the policy can be highly sample-inefficient due to lack of domain knowledge. There is also a certain unnaturalness or inefficiency of gait due to loose constraints (e.g. flailing arms in a running humanoid), jerkiness of motion due to discontinuous policy or feedback, in continuous state and action spaces.

In the scope of rigid bodies interacting with the environment, locomotion can only happen via surface-to-surface contacts between the agent and the environment. These are temporary degrees of constraints (DOCs) depending on the nature of the contact (e.g. sliding, rolling, fixed), which are otherwise identical to joint articulations between rigid bodies. Although MuJoCo provides a highly simplified model of Newtonian fluids, for the sake of brevity let's set that aside for now.

The fact is we are dealing with a physical system, and for performance reasons we should really be treating it as one. For instance, in (Heess et al., 2017) [video], locomotion behaviours are synthesized from simple reward functions in humanoid models.

The result is certainly impressive in at least two ways: The RL algorithm they used needed to know next to nothing about what it is actually controlling to generate a usable policy, and the learned policy ”generalizes” to different tasks (the terrain and obstacles). However, reinforcement learning is most relevant when we *really* cannot know the dynamics of a system, or the solution is intractable for a control system, e.g. in fluid dynamics (MIT 6.832). For rigid-body systems, we can certainly do a much better job in designing a locomotion controller, and we can start by including physical information into our model. This brings us to control theory.

2.2. Control theory

The field of control theory is specifically developed to deal with dynamical systems in continuous state spaces. A simple example of a linear feedback control system is the PID (proportional-integral-derivative) controller, whose control function (the response) is defined as:

$$\boldsymbol{y}(t) = w_1\boldsymbol{x}(t) + w_2\int_{t_1}^{t_2}\boldsymbol{x}(t')dt' + w_3\frac{d\boldsymbol{x}}{dt}$$

where w terms are tunable scalar parameters, and $\boldsymbol{x}$ is the state vector of the system, e.g. 6 degrees of motion of a rigid body, or temperature etc.

This could be understood as a controller whose response is linear with respect to the actual value of a variable, its instantaneous time delta, as well as moving average (whose job is to correct for a so-called steady-state error). It is notably not optimal for nonlinear systems.

More generally, in optimal control theory, in the special case where system dynamics could be described by a set of *linear* differential equations, and where the cost is described by a *quadratic* function, we have a so-called linear-quadratic (LQ) problem, and the optimal control solution is called the linear-quadratic regulator (LQR), which can be found by solving the system's Ricatti equation. In the related LQG (linear-quadratic-Gaussian) regime where both our readings of $\boldsymbol{x}(t)$ and outputs $\boldsymbol{y}(t)$ are corrupted by additive white Gaussian noise, the optimal solution is comprised of the gains from a Kalman filter on $\boldsymbol{x}$ applied on top of the associated LQR.

2.3. Other control methods

There are other control methods for dynamic processes, such as model predictive control (MPC). These generally require a model of the specific process we are concerned about optimizing, and are thus not generalizable to different tasks. We shall not describe them here.

3. Introduction

The method which we are going to be exploring shall attempt to combine the best of both worlds: the efficient trajectory search from trajectory optimization methods (via LQR or otherwise), and policy tuning via reinforcement learning, which provides the generality over very different trajectories. Emphasis: this is not an original idea. In fact, this proposal is primarily inspired by the following two papers:

- Interactive Control of Diverse Complex Characters with Neural Networks (Mordatch et al., 2015) [video]
- Discovery of Complex Behaviors through Contact-Invariant Optimization (Mordatch et al., 2012b) [video]

Since it is not a requirement of the project to develop something novel, we are treating this as an opportunity to implement a solution which is known to work, as well as taking the opportunity to dig deeper into trajectory optimization and control theory, which we (I) am practically scrambling to understand from scratch, in order to make sense of the literature as well as to push the project forward.

Compared to the original Stratego proposal, this line of inquiry allows time to become more familiar with MuJoCo itself, and provides a stepping stone for further research in locomotion. For instance, the authors have applied a similar method in (Mordatch et al., 2012a) to achieve high-dexterity control of a robotic hand.

We shall use the remaining space in the 3-page limit to describe the most important ideas of the approach, before going into what we expect to be able to do in the coming month, which is the limiting factor for scoping what we are able to do.

Just to clarify that this is a project about reinforcement learning, the method the authors used to produce the final policy is via block-alternating optimization, where they optimize, in turn, the trajectory (by penalizing deviations from the policy), and then the policy (by minimizing the loss w.r.t a set of trajectories, for different tasks). In this sense, we do not initially know what the solution is - trajectory optimization solves each task individually for us - and then the policy is trained to imitate the trajectories generated for *all* tasks at once in order to generalize to unseen states and (and hopefully tasks).

3.1. Additive noise

In (Mordatch et al., 2015), citing other prior work, states that injecting noise during training makes the resulting policy more robust, and noise is added to prevent divergent policies during execution. They add independent Gaussian noise

$\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_\epsilon^2 \boldsymbol{I})$ to all sensory inputs, and showed that if the noise is small enough, the optimal action at nearby noisy states is given by the first-order Taylor approximation $\boldsymbol{a}(\boldsymbol{s}+\epsilon) \approx \boldsymbol{a}(s) + \frac{d\boldsymbol{a}}{d\boldsymbol{s}}\epsilon$.

In addition, they added Gaussian noise $\gamma \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_\gamma^2 \boldsymbol{I})$ to the outputs of their hidden layers, as a network regularizer, in order to make the controller less sensitive to specific values of the hidden units.

The authors emphasized (and showed) the importance of this noise in the robustness of their resulting policies.

3.2. Contact-Invariant Optimization (CIO)

This is the main idea behind the 2012 paper. CIO optimizes over a composite objective function which is linear across four different loss terms. As defined in (Mordatch et al., 2012a):

$$L(\boldsymbol{s}) = L_{CI}(\boldsymbol{s}) + L_{Physics}(\boldsymbol{s}) + L_{Task}(\boldsymbol{s}) + L_{Hint}(\boldsymbol{s})$$

There are variants of it to account for non-planar contacts (Mordatch et al., 2012a), but the most important feature is its decomposability. CIO optimizes over a the contact-invariant (CI) loss function, which is linear in four component losses, and by disabling or enabling different components at different times, it can speed up the process of trajectory search. As defined in (Mordatch et al., 2012a), CI loss has the following definition:

4. Approach

References

Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S., Riedmiller, M., et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

International Stratego Federation. International stratego federation. https://www.kleier.net/isfstratego/rules.html. Accessed: 2019-02-07.

MIT 6.832. Underactuated Robotics Spring 2018 Lecture 20: Reinforcement Learning Part 1.

Mordatch, I., Popović, Z., and Todorov, E. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 137–144. Eurographics Association, 2012a.

Mordatch, I., Todorov, E., and Popović, Z. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012b.

Mordatch, I., Lowrey, K., Andrew, G., Popovic, Z., and Todorov, E. V. Interactive control of diverse complex characters with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3132–3140, 2015.