Controller learning

Martin Adam, Yigit Kayabasi, František Nesveda

Task

- Use data from several demonstrations to learn autonomous maneuvers
- Progress from simple maneuvers to more complex ones
- User changes the high-level activity, drone reacts

Input

1. Flight model
    - F: State x Command -> State

2. Desired trajectory/desired state (stationary hover,...)
    - Controlled by user (using a keyboard?)

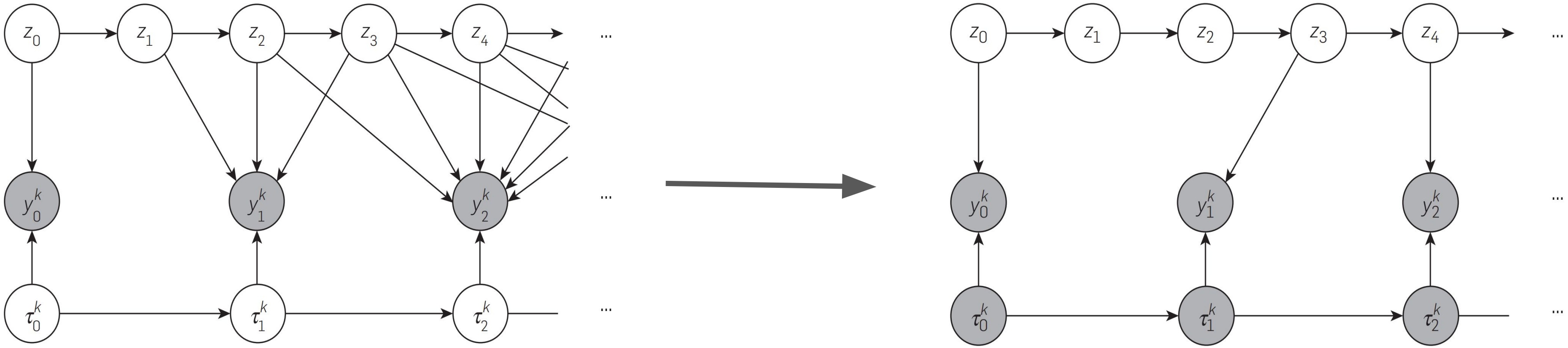
Output

- Command (or a sequence of commands) to be sent to the drone using the YADrone application

- Maximizing the probability of the drone being in the desired state after executing it

Activities

1. Static
   - The status can(?) be described manually, reward would be the least deviation
2. Dynamic (following desired trajectory, …)
   - Multiple trajectory demonstrations ⇒ each a bit different in length, timing, etc.
   - Possibility to ad known trajectory properties



Reinforcement Learning

- States, Actions, Dynamics model, H = Horizon, s(0) = the initial state, Reward function

- The policy $\pi = (\mu_0, \mu_1, \cdots, \mu_H)$
  Policy consists of mappings μ : states --> actions , one map for every t.

- Finding the optimal policy is the goal for making the controller.

Inverse Reinforcement Learning (Apprenticeship Learning)

- Assumes that an expert demonstrates the ideal behaviour and tries to mimic that efficiently. Reward function is correlated strongly with the distance* from desired trajectory of the demonstration.
- Requires no given reward function.
- Learns from a time series of both states and actions to
  - First to approximate the dynamics model around the trajectory
  - Second to derive a reward function
- IRL iteratively changes reward weights that results in policies that brings us closer* to the demonstration(desired trajectory).

LQR control problem

In the Linear Quadratic Regularization problem(A special class of MDPs):

State space is the time series $s(t) \in S$ and the actions $u(t) \in$ set of Actions

Dynamics model given by:
$s(t + 1) = A(t) * s(t) + B(t) * u(t)$

The reward for being in state $s(t)$ and taking action $u(t)$ is given by:
$-s(t)^T *Q(t) *s(t) - u(t)^T *R(t) * u(t)$

- Q and R are positive semi definite matrices which parameterize the reward function.

This standard formulation assumes $s(t) = 0$ for all t is the best trajectory, but with the extension $e(t) = s(t) - s^*(t)$ where $s^*(t)$ for $t=1,2..H$ is the desired trajectory.

DDP (Differential Dynamic Programming)

DDP approximates a solution to the MDP by iterating:

1. Linearly approximating the dynamics and quadratically approximating the reward function when applying the current policy around the desired trajectory.
2. Current Policy ← Compute optimal policy for the LQR

Other Challenges

1. Becoming familiar with the YADrone framework
2. Discovering the range of maneuvers that the drone can do
3. Not destroying the drone in the process :)

References

[Apprenticeship Learning for Helicopter Control]

[Autonomous inverted helicopter flight via reinforcement learning]

[Learning for Control from Multiple Demonstrations]