UPPSALA
UNIVERSITET

Reinforcement Learning

A little bit of control theory, and introduction to the project

Per Mattsson

2022

Department of Information Technology

Introduction

UPPSALA UNIVERSITET

- We have seen model-free prediction and control for finite state and action spaces.
- Next lecture: Function approximation.

Automatic control/control theory:

- A well-established research field for solving control problems in continuous state and action spaces.
- While RL is getting better at these types of problems, it still has a hard time solving many problems.

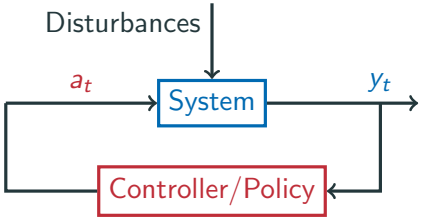
A little bit on feedback and control theory

What is automatic control?

"Making a system behave the way we want"

- History dates back to at least later half of 19th century.
- Used "everywhere": Industrial machines, cars, airplanes, cell phones, medicine, wastewater treatment, nuclear plants, robots.
- Well-established theory.

Feedback control



- In automatic control a model is often used to design a controller/policy.
- A real world system is always affected by disturbances.
- **Feedback can be used to:**
  - React to disturbances.
  - Stabilize an unstable system.
  - Do not need exact model of the system.
- **In RL:** We use feedback from observed state ($a_t = \pi(s_t)$).

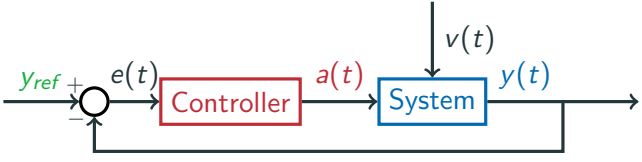
PID – The most famous controller!

- **Goal:** The output $y(t)$ should follow the reference signal y_{ref}.
- That is: Make the control error $e(t) = y_{ref} - y(t)$ as small as possible!
- **The PID-controller:**

$$a(t) = a_0 + \underbrace{K_1 e(t)}_{\text{P}} + \underbrace{K_2 \int_0^t e(t)dt}_{I} + \underbrace{K_3 \frac{de(t)}{dt}}_{\text{D}}.$$

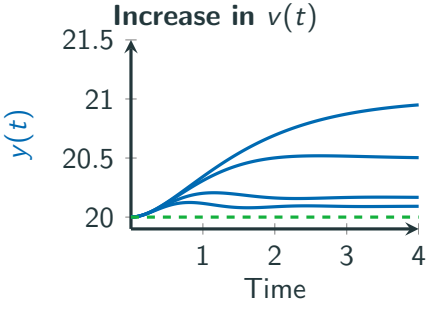
- Only three parameters to tune! (Adaptive PID tuning is RL?)
- **Deep-RL:** Many hyperparameters to choose before you can even start training!
- PID still often works very well, and is without a doubt the most common controller in industry!

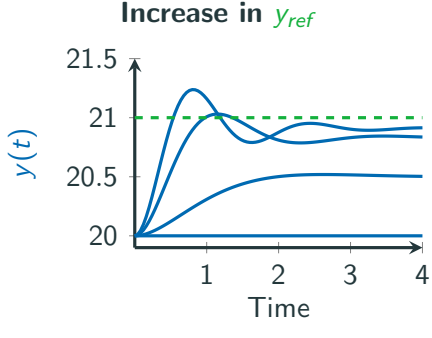
Example: Temperature control



- $y(t)$ – Temperature in the room.
- y_{ref} – Temperature we want.
- $a(t)$ – Radiator power.
- $e(t)$ – Control error, $e(t) = y_{ref} - y(t)$.
- $v(t)$ – Disturbance (e.g. outdoor temperature).

$$a(t) = a_0 + K_1(y_{ref} - y(t)) = a_0 + K_1 e(t).$$

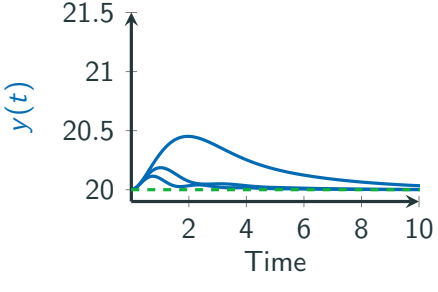




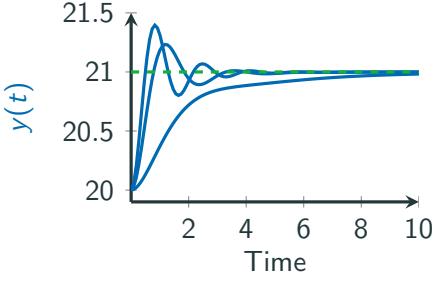
$$K_1 = 0 K_1 = 1 K_1 = 5 K_1 = 10$$

$$a(t) = a_0 + K_1 e(t) + K_2 \int_0^t e(t)dt.$$

Increase in $v(t)$



Increase in y_{ref}



$$K_1 = 0 K_1 = 1 K_1 = 5 K_1 = 10, K_2 = 0.5 K_1$$

The linear quadratic controller

- Many real-world system can be well described as a linear system:

$$s_{t+1} = As_t + Ba_t + w_t$$

  where w_t are white noise disturbances.
- Typical policy: $a = \pi(s) = -Ls$.
- It is common to consider a cost (negative reward)

$$c_t = k_1 y_t^2 + k_2 a_t^2$$

  and for example minimize the average cost over time.
- A rare case when a continuous MDP gives an analytical solution!
- State-value: $v_\pi(s) = s^\top Ps + \sigma$ where P, σ depends on the system and L.
- Optimal π_*: $\pi_*(s) = -L_* s$ with L_* computed from A and B (Riccati equation).

- If we have a model of the system, we could in each time step try to choose action that maximize

$$\sum_{t=1}^{T} R(s_t, a_t)$$

  subject to $s_0 =$ current state, and that s_t follow the model.
- After PID, maybe the most popular controller in industry.

Examples

- Inverted pendulum.
- RL vs MPC for robot catching a ball in a cup.

Moral of the story:

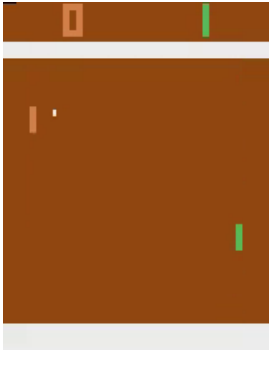
- RL is not magic.
- In many cases there are simpler and more efficient solutions to control problems.
- But there has been great progress in RL during the last decade.

The Project

Learning to play Pong with DQN

- Understand and implement Deep Q-Network (DQN).
- Presented in papers from 2013 and 2015.
- Managed to learn how to play 49 Atari games directly from pixels!

Goal of this project: An agent that learns to play Pong from pixels.



Why is this hard?

UPPSALA UNIVERSITET

Very large state space!

- Will use preprocessing to turn each frame into 84×84 grey scale pixels (256 possible values).
- One frame is not enough for a state!
- In practice not possible to learn Q-values for each state/action-pair individually!

How to solve?

- Estimate approximate value function $Q(s, a; \boldsymbol{w})$.
- using a deep neural network. (Needs new ideas to stabilize learning)

Still hard!

- You will probably notice that DQN is quite sensitive to hyperparameters.
- Will take a few hours of training to learn to play Pong optimally.

Step 0

UPPSALA UNIVERSITET

Join a group!

- 4 people in each group.
- Join a group on Studium. (Or join the group "Help with finding a group")
- Groups have a Studium-page, but you can communicate in other ways too.
- **This week:** Join a group and set up an initial meeting.

Read the papers!

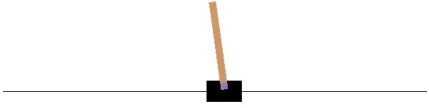
- Links to the two papers that introduce DQN can be found on Studium.

Preparing:

- Install and familiarize yourself with PyTorch.
- If you do not have a supported GPU, then try to use Google Colab.

Step 1 - Implement and test DQN

- We will provide a code skeleton you can start from.
- To test your implementation your goal is to solve `CartPole-v0`.



- **Training** can be completed in a few minutes.
- **Experiment** with different hyperparameters, to see how sensitive it is.
- **Report:** At least plot of the return per episode during the training and discussion about choice of hyperparameters.

Step 2 - Time to learn to play Pong!

UPPSALA UNIVERSITET

- **Extend your code a bit:**
  - Preprocessing (can use `AtariPreprocessing`-wrapper).
  - Stack observations into a state.
  - Use the same policy network architecture as in the paper.
  - *Optional (but good idea):* Restrict agent to only use two actions.
- **Training takes time!**
  - Even with a GPU can take 4-6 hours.
  - Start early!
  - In the group, you can train agents in parallel.
- **Report:** At least plot with return per episode during training. (Should show improvement, even if you do not find optimal policy).
- *Optional:* If you have time, try to train your agent on other Atari games e.g. `Breakout-v0`.

Project report

- Short description of how and why DQN works.
- Results from experiments.
- Discussion.

Deadline: May 29.

Questions: `ruoqi.zhang@it.uu.se`

Summary

UPPSALA UNIVERSITET

- Join group as soon as possible! Set up a first meeting with group.
- Start with preparations as soon as possible.
- Start early! Training Pong takes hours, and you may have to fix errors and re-train!