Imitation Learning

CMPT 419/983

Mo Chen

SFU Computing Science

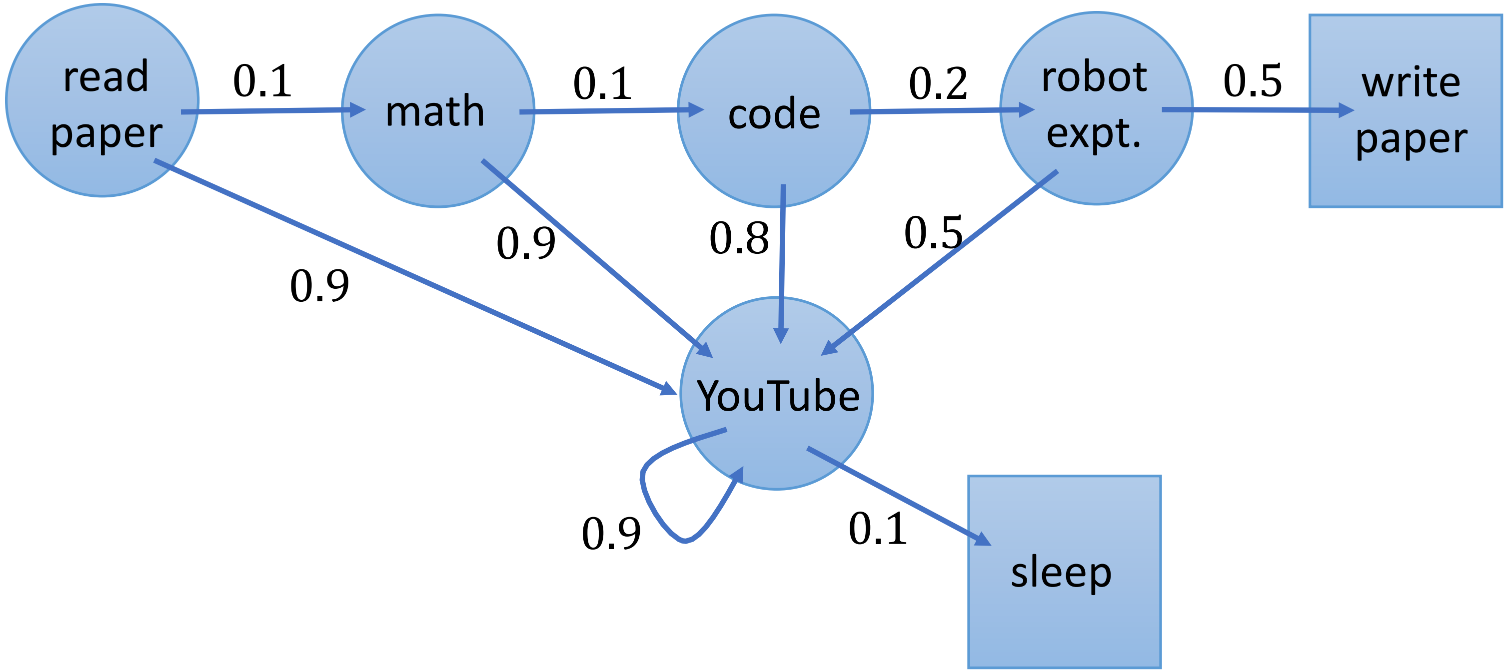
28/10/2019

Outline

- Markov Decision Process
- Imitation Learning

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



State space includes

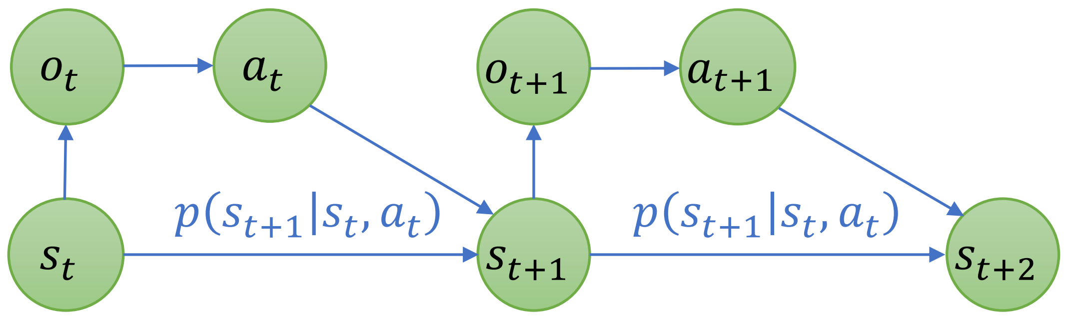
- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.1 & & & 0.9 & & \\ & & 0.1 & & 0.9 & & \\ & & & 0.2 & 0.8 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.9 & & 0.1 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Extensions of Problem Setup

- Partially observability
  - Partially Observable Markov Decision Process (POMDP)
  - State not fully known; instead, act based on observations



  - Policy: $\pi_\theta(a|o)$
- In this class, state s will be synonymous with observation o.

Reinforcement Learning Objective

- Given: an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, and reward function $r(s, a)$

- Objective: Maximize sum of rewards ("return")

$$\underset{\pi_\theta}{\text{maximize}} \sum_{t=0}^{\infty} r(s_t, a_t)$$

Reinforcement Learning Objective

- Given: an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, and reward function $r(s, a)$

- Objective: Maximize discounted sum of rewards ("return")

$$\underset{\pi_\theta}{\text{maximize}} \quad \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

Reinforcement Learning Objective

- Given: an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, and reward function $r(s, a)$

- Objective: Maximize expected discounted sum of rewards ("return")

$$\underset{\pi_\theta}{\text{maximize}}\ \mathbb{E} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

Reinforcement Learning Objective

- Given: an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, and reward function $r(s, a)$

- Objective: Maximize expected discounted sum of rewards ("return")

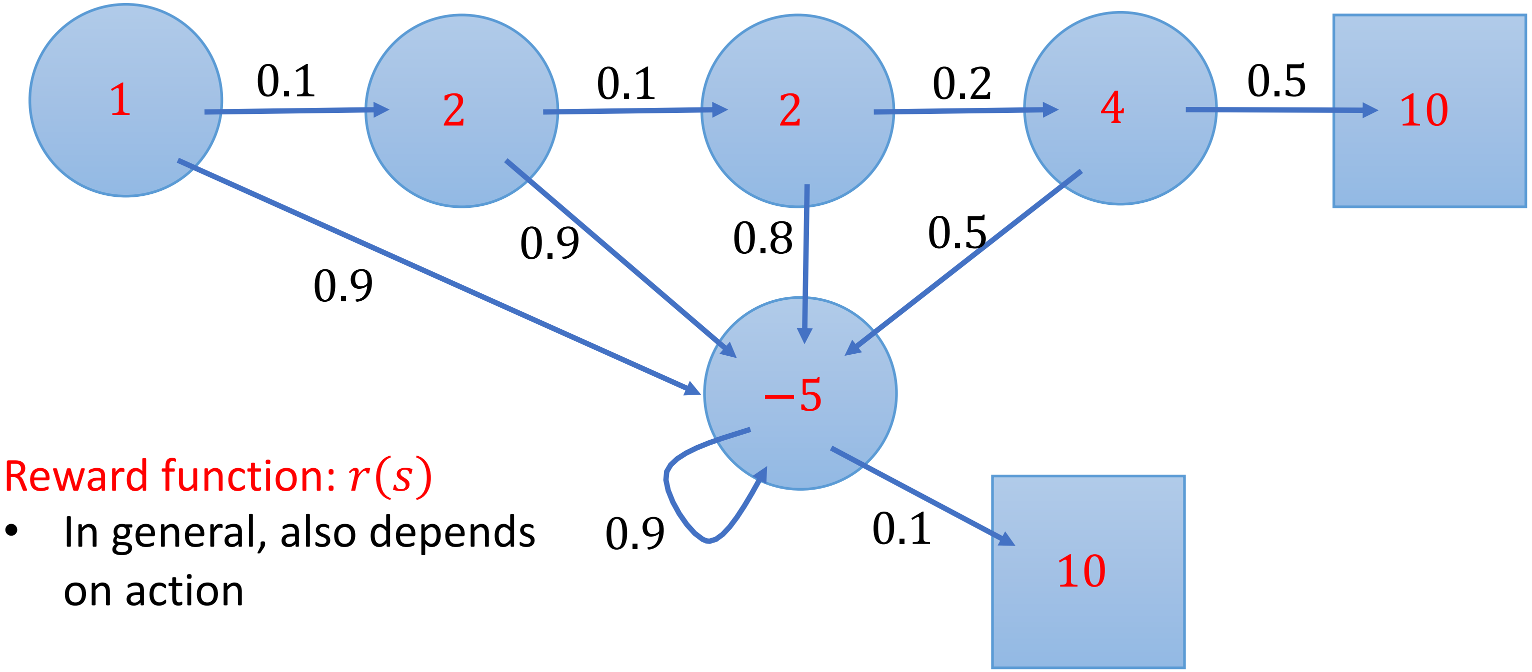
$$\underset{\pi_\theta}{\text{maximize}}\, \mathbb{E} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

  - $\gamma \in (0,1]$: discount factor – larger roughly means "far-sighted"
    - Prioritizes immediate rewards
    - $\gamma < 1$ avoids infinite rewards; $\gamma = 1$ is possible if all sequences are finite

- Constraints: now incorporated into the reward function
  - Only constraint (usually implicit): subject to transition matrix $\mathcal{T}$ (system dynamics)

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



Reward function: $r(s)$

- In general, also depends on action

State space includes

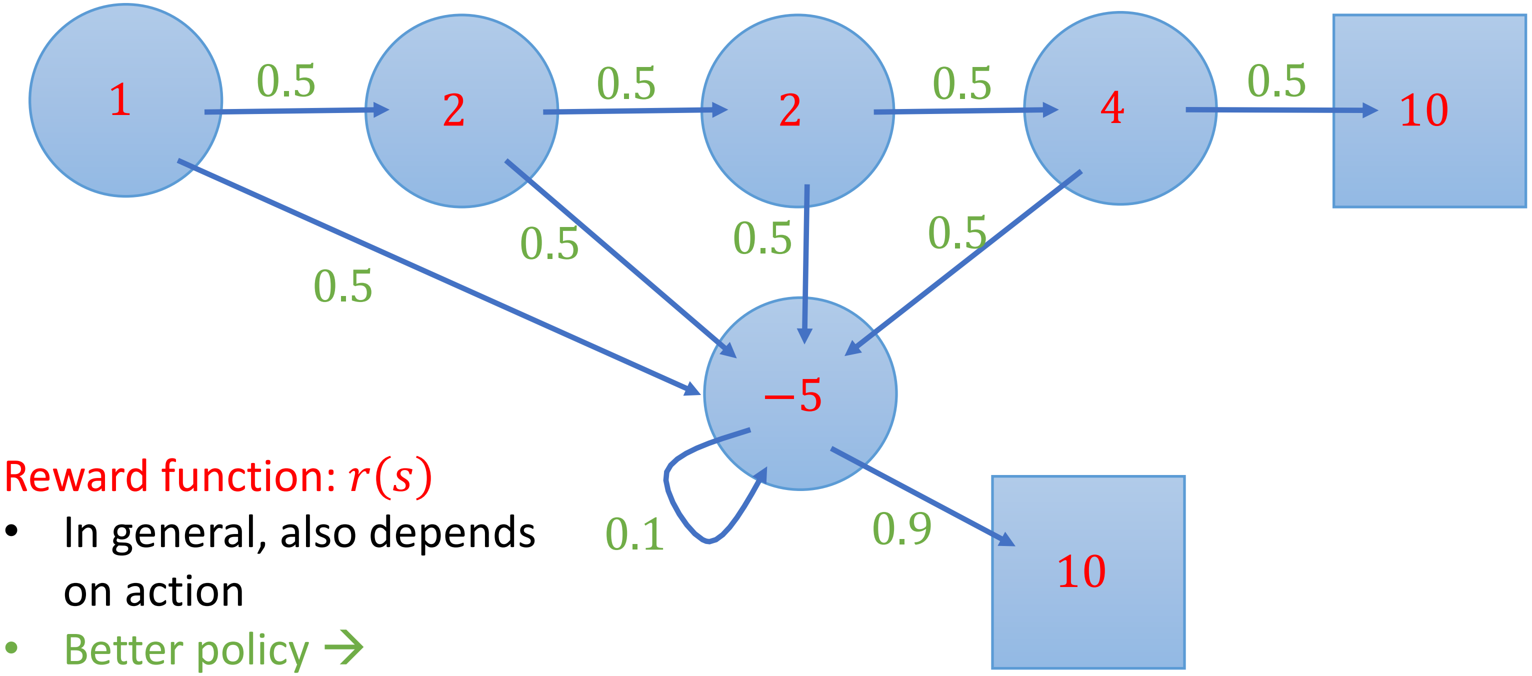
- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.1 & & & 0.9 & & \\ & & 0.1 & & 0.9 & & \\ & & & 0.2 & 0.8 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.9 & & 0.1 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



Reward function: $r(s)$

- In general, also depends on action
- Better policy → different Markov chain → different reward

State space includes

- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.5 & & & 0.5 & & \\ & & 0.5 & & 0.5 & & \\ & & & 0.5 & 0.5 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.1 & & 0.9 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Reinforcement Learning and Optimal Control

- Reinforcement Learning

$$\underset{\pi_\theta}{\text{maximize}}\, \mathbb{E} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

  - Dynamics constraint is implicit
    - And not necessary needed
  - Typically, no other explicit constraints
  - Problem set up captured entirely in the reward
  - Probabilistic

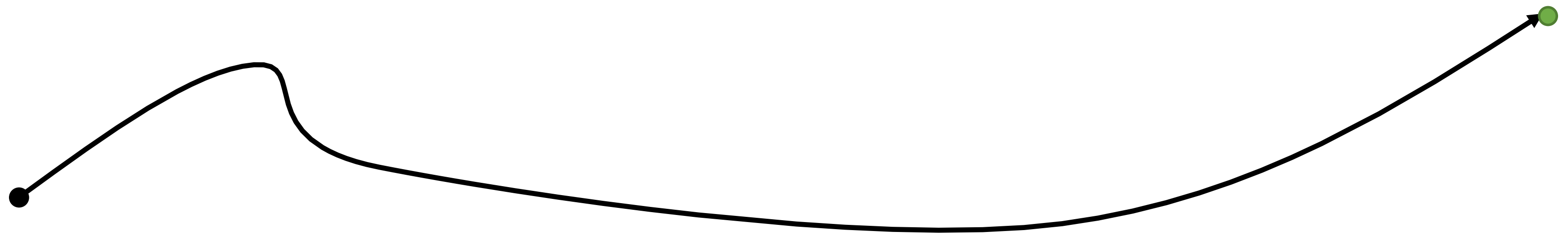
- Optimal control

$$\begin{aligned}
\underset{u(\cdot)}{\text{minimize}}\ & l\big(x(t_f), t_f\big) + \int_0^{t_f} c(x(t), u(t), t)\,dt \\
\text{subject to}\ & \dot{x}(t) = f\big(x(t), u(t)\big) \\
& g\big(x(t), u(t)\big) \geq 0 \\
& x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0
\end{aligned}$$

  - Explicit constraints
  - Can be continuous time
  - Not necessarily probabilistic

Imitation Learning

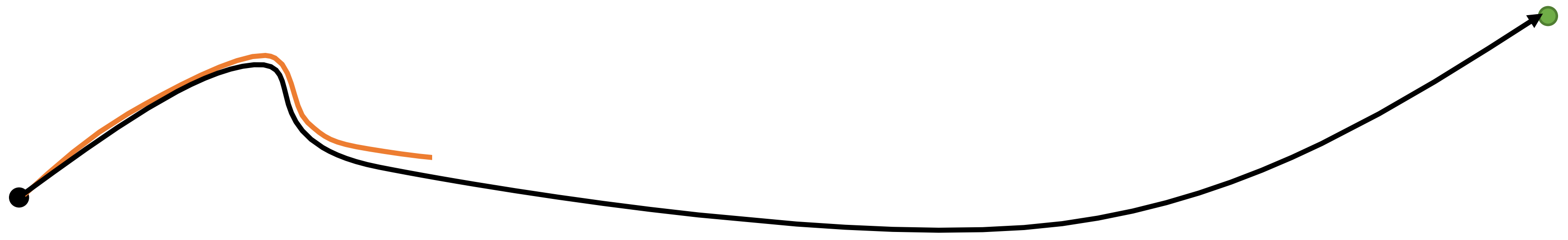
- Collect data through expert demonstration – sequence of states and actions, $\{s_0, a_0, s_1, a_1, \dots, s_{N-1}, a_{N-1}, s_N\}$
    - Note: Expert may not be solving $\underset{\pi}{\text{maximize}}\ \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$



- Learn $\pi_\theta(a_t|s_t)$ from data via regression
    - Minimize $\mathbb{E}[\sum \| a_t - \pi_\theta(a_t|s_t) \|]$

Imitation Learning

- Collect data through expert demonstration – sequence of states and actions, $\{s_0, a_0, s_1, a_1, \dots, s_{N-1}, a_{N-1}, s_N\}$
    - Note: Expert may not be solving $\underset{\pi}{\text{maximize}}\, \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$



- Learn $\pi_\theta(a_t|s_t)$ from data via regression
    - Minimize $\mathbb{E}[\sum \| a_t - \pi_\theta(a_t|s_t) \|]$

Imitation Learning

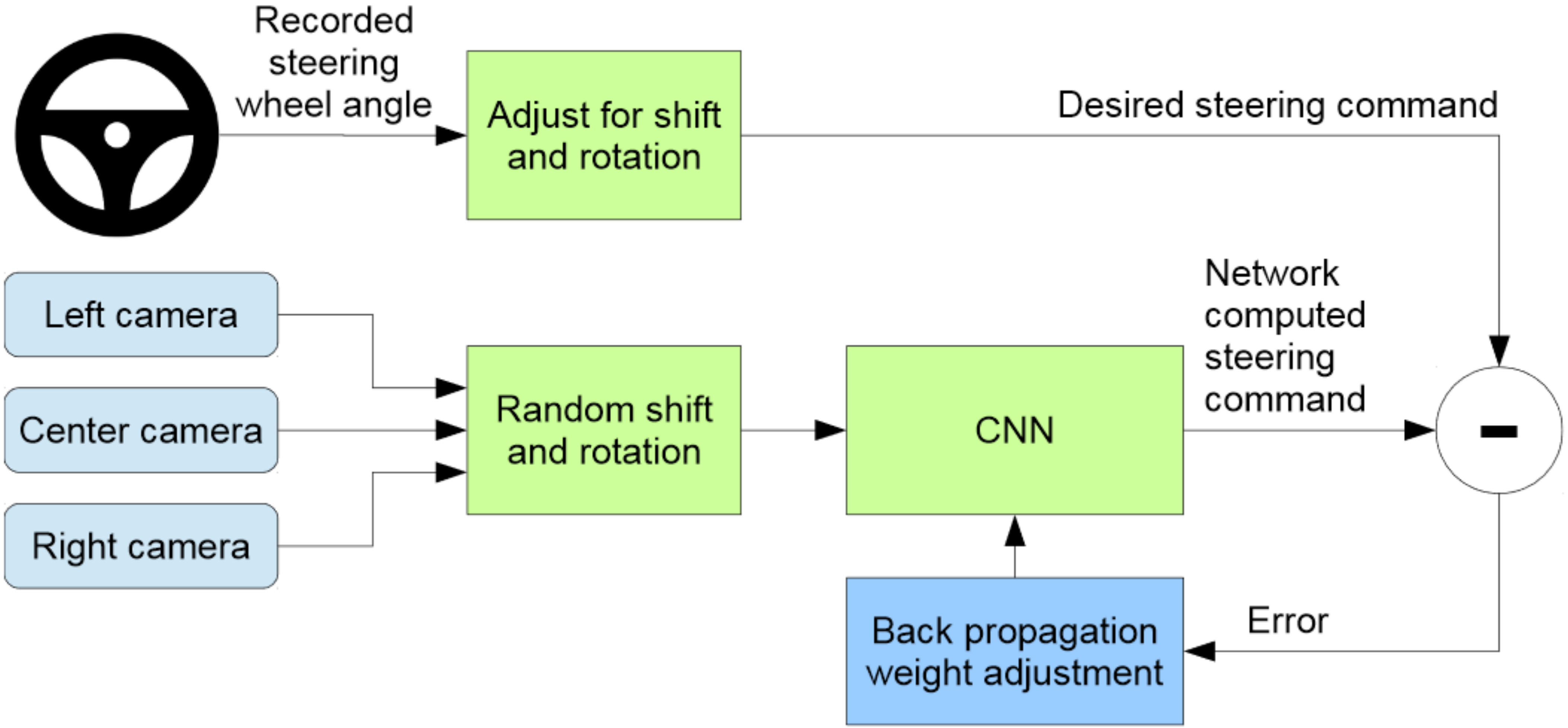
- Collect data through expert demonstration – sequence of states and actions, $\{s_0, a_0, s_1, a_1, \ldots, s_{N-1}, a_{N-1}, s_N\}$
  - Note: Expert may not be solving $\underset{\pi}{\text{maximize}}\, \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$

- Learn $\pi_\theta(a_t|s_t)$ from data via regression
  - Minimize $\mathbb{E}(\sum \|a_t - \pi_\theta(a_t|s_t)\|)$

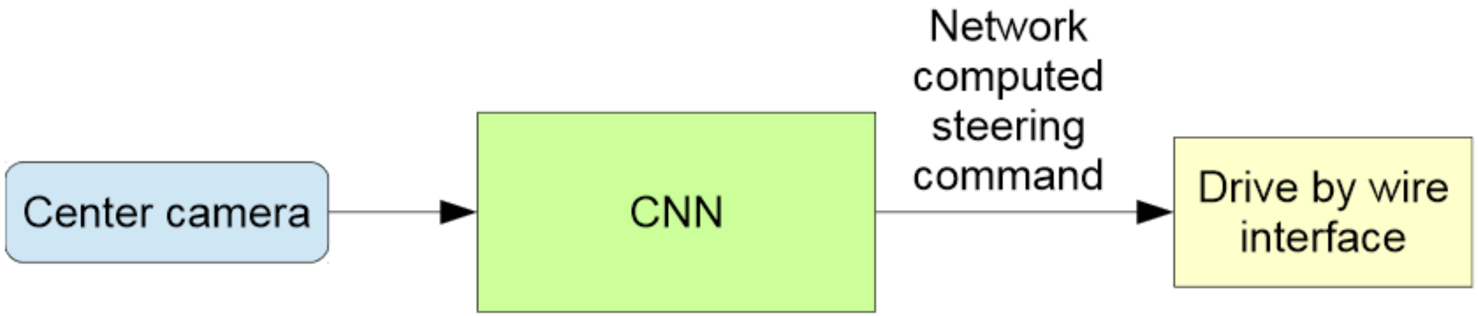
- Usually doesn't work due to "drift": small mistakes add up, and takes the system far from trained states
  - Sometimes, there can be "tricks" to make imitation learning work!

Autonomous Driving Through Imitation

Training:



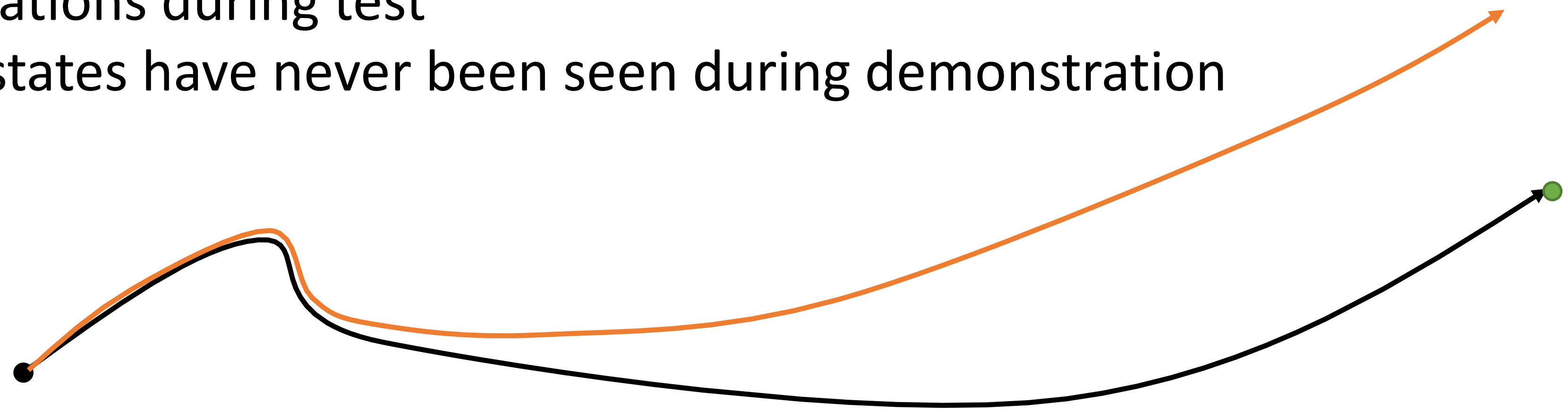
Testing:



Bojarski '16. "End to End Learning for Self-Driving Cars," CVPR 2016

Dataset Aggregation

- Imitation learning drawback:
  - Distribution of observations in training is different from distribution of observations during test
  - Some states have never been seen during demonstration



- How to make the distributions equal?
  - Train perfect policy
  - Change data set → DAgger (Dataset Aggregation)

Dataset Aggregation (DAgger) Algorithm

1. Train policy from some initial data, $\mathcal{D}_i = \{s_0, a_0, s_1, a_1, \ldots, s_{N-1}, a_{N-1}, s_N\}$
2. Run policy to obtain new observations $\{s_{N+1}, s_{N+2}, \ldots, s_{N+M}\}$
   - Note: time indices and states here may not continue from initial data
3. Use humans to label data by providing actions for new observations, $\{a_{N+1}, \ldots, a_{N+M-1}\}$
   - This creates another data set, $\overline{\mathcal{D}}_i = \{s_{N+1}, a_{N+1}, s_{N+2}, a_{N+2} \ldots, a_{N+M-1}, s_{N+M}\}$
4. Combine two datasets, $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \overline{\mathcal{D}}_i$
   - Go back to first step

Challenges

- Non-Markovian behaviour
  - Perhaps augment state/observation space to include some history
  - Use neural networks that implicitly capture time series data: RNNs/LSTMs
- Unnatural data collection
  - Humans are probably not very good at collecting correction data in this manner
- Inconsistencies in human action

Addressing Drift

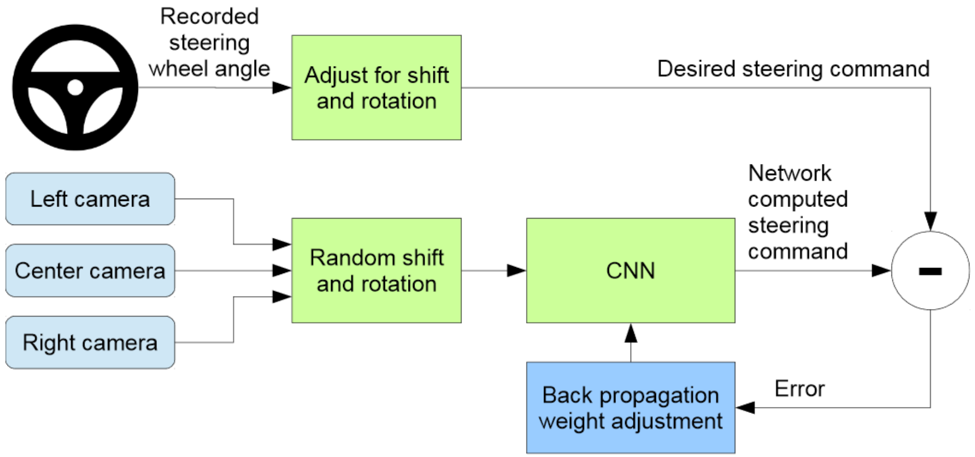
- Main goal: Teach system to correct errors
- Explicitly demonstrate corrections (DAgger)
- During demonstration, add noise to “force” mistakes, and see how humans correct them
- Ask humans to intentionally make mistakes
- Prior knowledge and heuristics
  - Example: Learn from stabilizing controller

Imitation Learning Tricks

- Common neural network architectures
  - LSTM – since we have time-series data
  - CNN – usually in combination with LSTM, if the observations are images

- Simplify action space:
  - Driving example: action space simplified to {left, centre, right}

- Clever data collection
  - Driving example: side cameras



- Inverse reinforcement learning
  - Learn goal, instead of policy, from data
  - Use reinforcement learning to learn to achieve the same goal

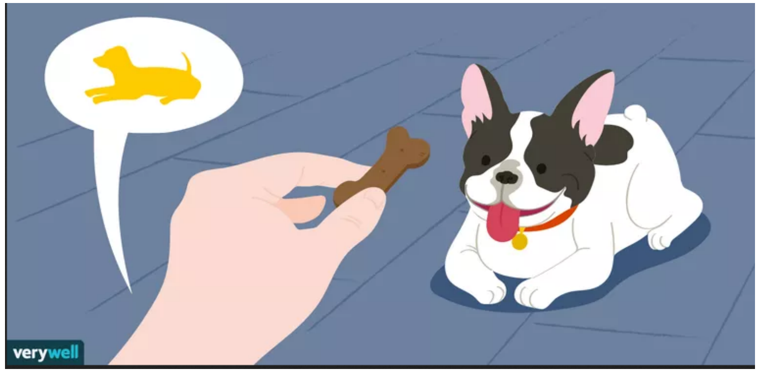
Imitation Learning Drawbacks

- Very small amount of data – challenging for training deep neural networks
- Humans are not very good at providing some kinds of actions
  - Quadrotor motor speed
  - Non-humanoid machines
- Hard to perform better at tasks humans are not very good at

Reinforcement Learning

- Humans can learn without imitation
  - Given goal/task
  - Try an initial strategy
  - See how well the task is performed
  - Adjust strategy next time

- Reinforcement learning agent
  - Given goal/task in the form of reward function $r(s, a)$
  - Start with initial policy $\pi_\theta(a|s)$; execute policy
  - Obtain sum of rewards, $\sum_t r(s_t, a_t)$
  - Improve policy by updating θ, based on rewards



RL vs. Other ML Paradigms

- No supervisor
- Sequential data in time
- Reward feedback is obtained after a long time
    - Many actions combined together will receive reward
    - Actions are dependent on each other
- In robotics: lack of data