CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Robotics: Introduction to AI in robotics

Vladimír Petrík

vladimir.petrik@cvut.cz

08.01.2024

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T$ to minimize cost function J

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{T-1}}{\arg\min} \; J(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T, \boldsymbol{u}_0, \dots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1}}{\arg\min} \; J(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system
- Cost function:
  - $J = \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + l_T(\boldsymbol{x}_T)$
  - l is cost function at time t
  - l_T is terminal cost function
  - T is time horizon

Optimal control - Model Predictive Control

- Find optimal control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_T$ to minimize cost function J
  - $\boldsymbol{u}^* = \underset{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{T-1}}{\arg\min}\ J(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T, \boldsymbol{u}_0, \dots, \boldsymbol{u}_T)$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
  - $\boldsymbol{x}_t$ is state of the system at time t
  - $\boldsymbol{u}$ is control (torque, velocity, ...)
  - $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is dynamics/simulation of the system
- Cost function:
  - $J = \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + l_T(\boldsymbol{x}_T)$
  - l is cost function at time t
  - l_T is terminal cost function
  - T is time horizon
- Use numerical optimization to solve the minimization problem
  - dynamics (f) and costs (l, l_T) needs to be differentiable

MPC in practical application

- Robot controlled at 100Hz

MPC in practical application

- Robot controlled at 100Hz
- For each control step, MPC is solved
  - find sequence of control that optimize cost function
  - fixed time horizon (e.g. 0.5 s)

MPC in practical application

- Robot controlled at 100Hz
- For each control step, MPC is solved
  - find sequence of control that optimize cost function
  - fixed time horizon (e.g. 0.5 s)
- Apply first control from the sequence

MPC in practical application

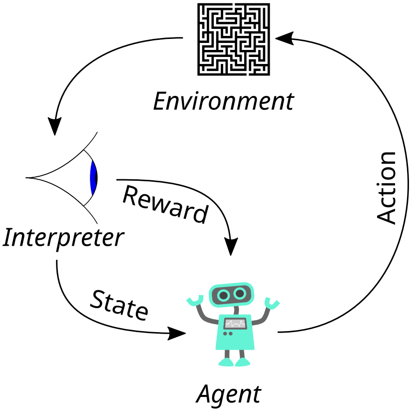
- Robot controlled at 100Hz
- For each control step, MPC is solved
  - find sequence of control that optimize cost function
  - fixed time horizon (e.g. 0.5 s)
- Apply first control from the sequence
- Repeat
- Why not applying all controls from the sequence?

MPC in practical application

- Robot controlled at 100Hz
- For each control step, MPC is solved
  - find sequence of control that optimize cost function
  - fixed time horizon (e.g. 0.5 s)
- Apply first control from the sequence
- Repeat
- Why not applying all controls from the sequence?
- What if we do not have gradient of dynamics/costs?

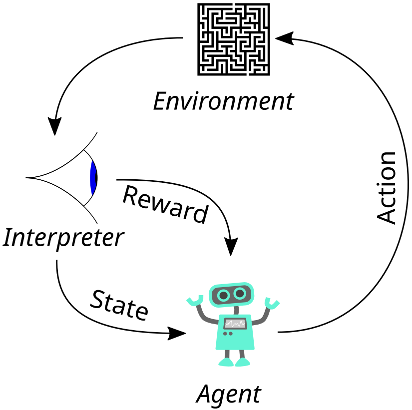
Reinforcement learning

- Modeled as Markov Decision Process



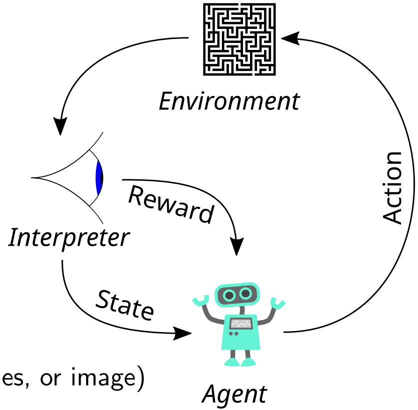
Reinforcement learning

- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state

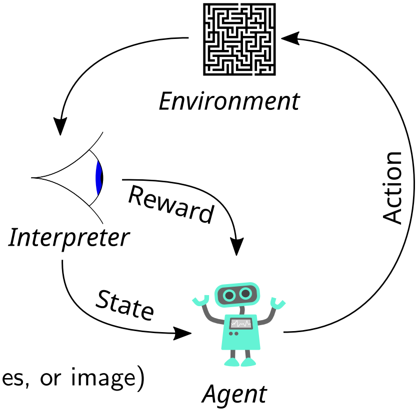


Reinforcement learning

- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ

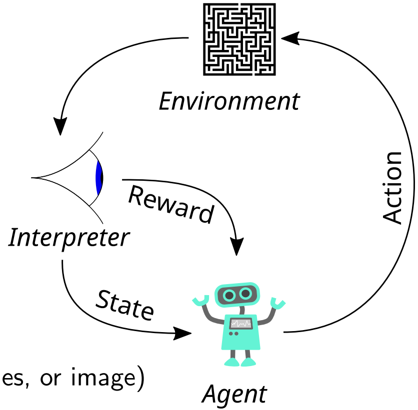


Reinforcement learning



- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ
- Instantaneous reward: $r(\boldsymbol{s}, \boldsymbol{a})$
- Expected return of the policy: $R = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{s}_t)} \left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$ s.t. $\boldsymbol{s}_{t+1} \sim f(\boldsymbol{s}_t, \boldsymbol{a}_t)$

Reinforcement learning



- Modeled as Markov Decision Process
- Agent interacts with environment
- Agent receives reward for each action/state
- Goal is to find policy that maximizes reward in time
- Stochastic policy: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
  - $\boldsymbol{a}$ is action (e.g. torque)
  - $\boldsymbol{s}$ is state of the system (e.g. joint angles and velocities, or image)
  - π_θ is policy parameterized by θ
- Instantaneous reward: $r(\boldsymbol{s}, \boldsymbol{a})$
- Expected return of the policy: $R = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{s}_t)} \left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$ s.t. $\boldsymbol{s}_{t+1} \sim f(\boldsymbol{s}_t, \boldsymbol{a}_t)$
- Goal: $\arg\max_\theta R$
- Compare to MPC: $\arg\min_{\boldsymbol{u}_1,\dots,\boldsymbol{u}_T} J$ s.t. $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$
  - expectation over trajectories τ sampled by following policy π_θ

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?

Policy gradient

- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?
- Can we apply millions of trajectories to real robot?

Policy gradient

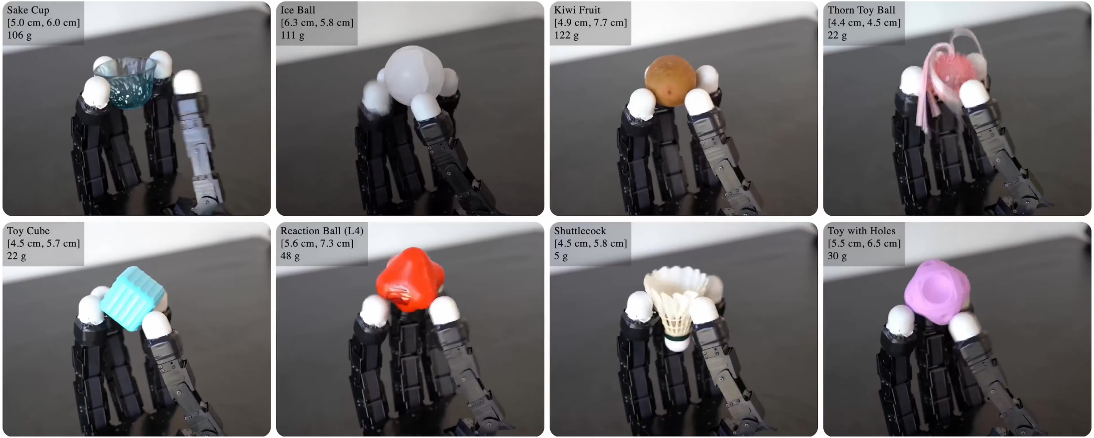
- Policy π_θ is parameterized by θ
- Is used to sample action $\boldsymbol{a}$ given state $\boldsymbol{s}$: $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{s})$
- Gradient descent algorithm: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta R(\pi_\theta)$
  - θ parameterizes policy π_θ
  - α is learning rate
  - $\nabla_\theta R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(\boldsymbol{s}_t) r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$
  - expectation over trajectories τ sampled by following policy π_θ
  - in practise expectation is approximated by sampling a lot of trajectories (millions)
  - why we need stochastic policy?
- Can we apply millions of trajectories to real robot?
- We need fast and accurate simulation of robots
  - Gazebo
  - NVIDIA Isaac Sim

Example of RL

Example of RL

Example of RL

Example of RL



Reward shaping

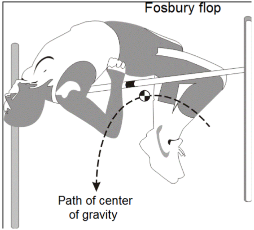
- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time

Reward shaping

- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time
- Reward shaping
  - add additional reward to the original reward
  - additional reward is designed to guide learning and avoid local minima
  - engineering work
- Is there a better solution?

Reward shaping

- Finding solution to RL problem is hard
  - sparse reward
  - local minima
  - long training time
- Reward shaping
  - add additional reward to the original reward
  - additional reward is designed to guide learning and avoid local minima
  - engineering work
- Is there a better solution? Learning from demonstration.
- Example from high-jump (Fosbury flop - 1968 gold medal)



Offline reinforcement learning - Learning from demonstration

- Collect data from real robot guided by the operator
- Pre-Train policy on collected data
- Optionally, fine-tune policy in simulation/ on real robot
- How to pre-train policy?

Offline reinforcement learning - Learning from demonstration

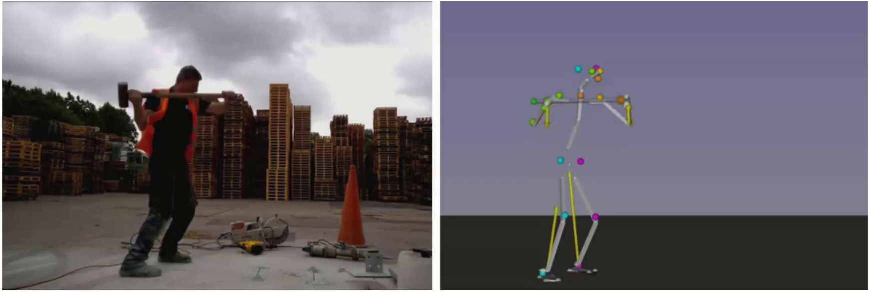
- Collect data from real robot guided by the operator
- Pre-Train policy on collected data
- Optionally, fine-tune policy in simulation/ on real robot
- How to pre-train policy?
  - behavior cloning - supervised learning
  - $\arg\min_{\theta} \sum_{i=1}^{N} (\pi_\theta(s_i) - a_i)^2$
  - diffusion policy - supervised learning

Learning from video

- Instructional videos are widely available on YouTube
- Can we learn from them?

Learning from video

- Instructional videos are widely available on YouTube
- Can we learn from them?
- Depends on the task/video, e.g. if human is visible
  - we can extract human pose from video
  - we can extract the manipulated object pose
  - we can extract interaction forces



Learning tool manipulation from instructional video

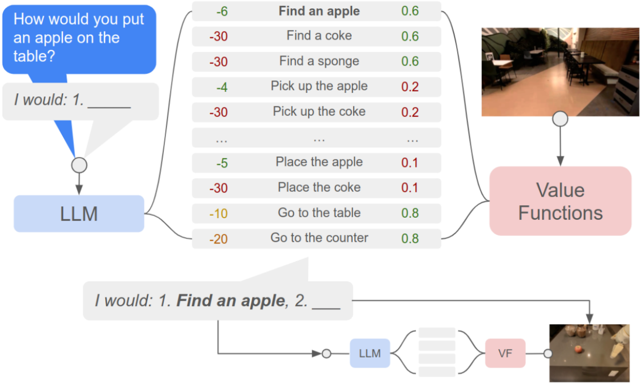
Learning to Use Tools by Watching Videos

Input: instructional video from YouTube

Output: tool manipulation skill transferred to a robot

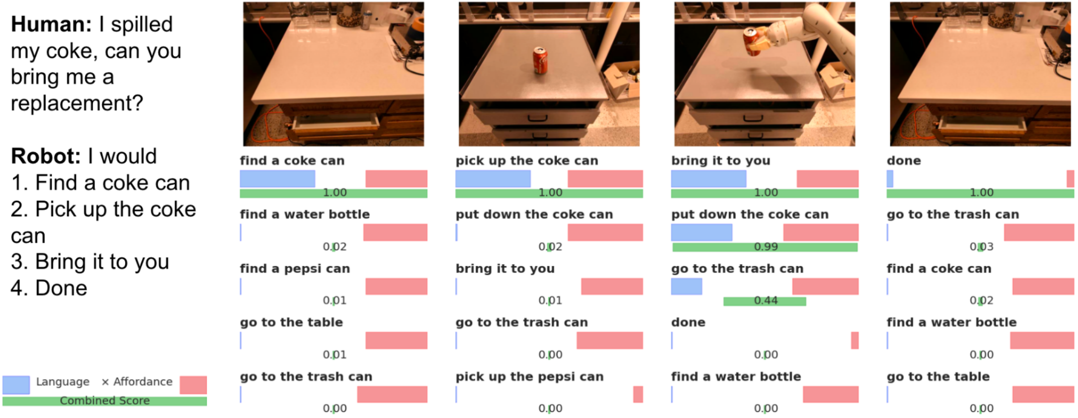
Large language models for robot learning - SayCan [1]

- Combine LLM plan with learned (RL) set of skills
  - LLM generates the global plan (prompt engineering needed)
  - Ask LLM, how much is the skill contributing to the plan
  - Ask skill, how likely it is to success

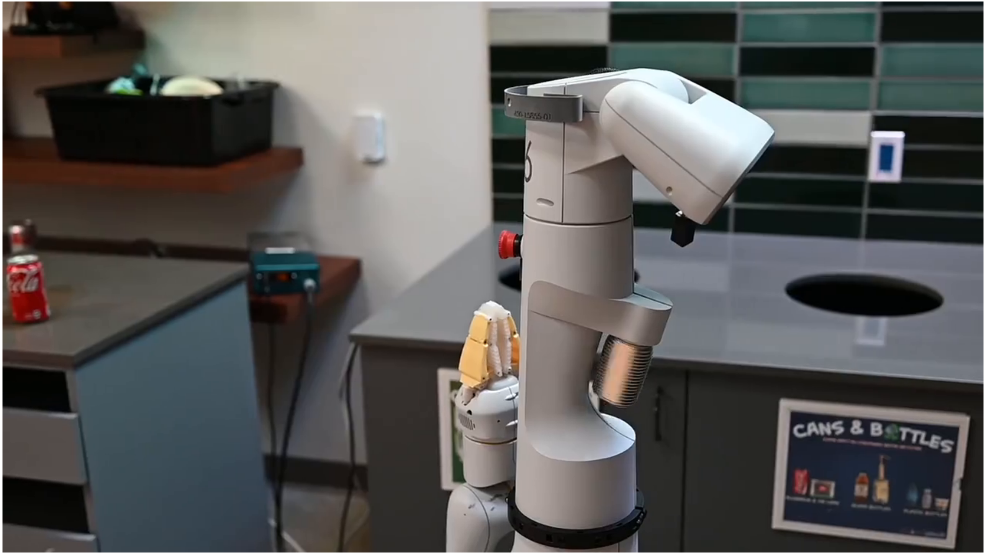


[1] https://say-can.github.io/

SayCan example



SayCan example



Final work

- Reservation system in BRUTE
  - reserve robot
  - reserve teacher (V. Smutný, P. Krsek, V. Petrík)
- Upload your report and code 24 hours before demonstration
- You need to demonstrate your work before signing up to exam

Exam

- CIIRC B670/B671 from 8AM
- Theoretical questions
  - what is computed by forward dynamics
  - how to efficiently compute inverse of rotation matrix

Exam

- CIIRC B670/B671 from 8AM
- Theoretical questions
  - what is computed by forward dynamics
  - how to efficiently compute inverse of rotation matrix
- Computation with coordinate frames
  - express vector in coordinate frame A if you know its coordinates in coordinate frame B

Exam

- CIIRC B670/B671 from 8AM
- Theoretical questions
  - what is computed by forward dynamics
  - how to efficiently compute inverse of rotation matrix
- Computation with coordinate frames
  - express vector in coordinate frame A if you know its coordinates in coordinate frame B
- Computation of manipulator kinematics
  - forward kinematics
  - inverse kinematics
  - Jacobian