CS 369: Introduction to Robotics

Prof. Thao Nguyen

Spring 2026

HAVERFORD
COLLEGE

Outline for today

- Robot learning
- Behavioral cloning

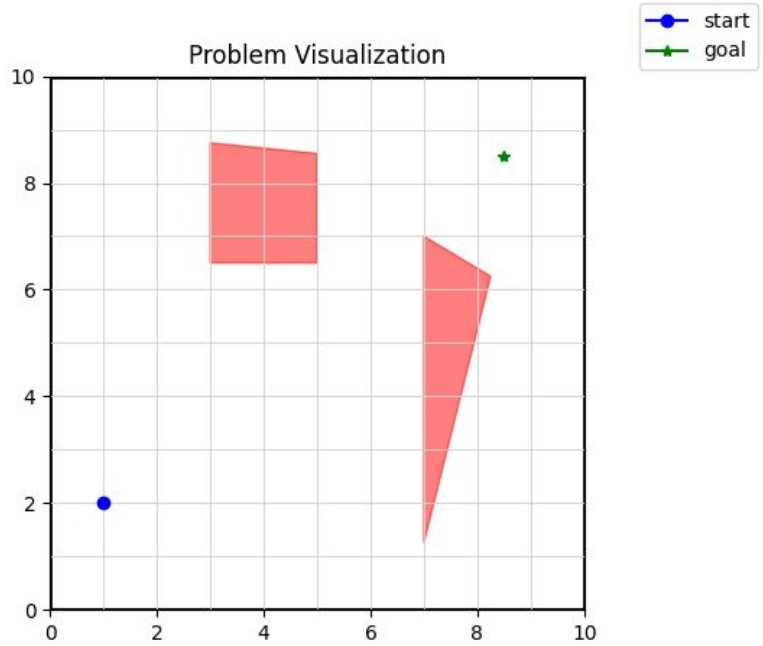
Outline for today

- Robot learning

- Behavioral cloning

Classical robotics

- Compose different systems for perception, control, etc.
- Require models of the environment
- Generalizable, modular, explainable
- Bottlenecked by worst component
- Indirect relationship between components' performance and task performance



Robot learning

- Learn from data
- Do not require environment models
- Directly optimize for task performance
- Less explainable, performance depends on data and ML model

Imitation learning

- Mimic human demonstrations
- Techniques:
  - Behavioral cloning
  - Inverse reinforcement learning

Outline for today

- Robot learning

- Behavioral cloning

Behavioral cloning

- Treats imitation learning as a supervised learning task
- Takes in expert demonstrations: $D = \{(s_i, a_i)\}^N$
- Minimizes loss:
  - Continuous actions: $\frac{1}{N}\sum_{i=1}^{N} \|h_w(s_i) - a_i\|^2$
  - Discrete actions: $\frac{1}{N}\sum_{i=1}^{N} -\log[l(h_w(s_i), a_i)]$

Limitations

- Heavily depends on the diversity and coverage of expert demonstrations
- Limited generalization
- Distributional shift: small prediction errors can lead the agent into states not seen in the training data -> training and test data are not drawn from the same distribution

DAgger

Dataset aggregation: Iterative approach where the agent collects data by acting in the environment and queries the expert for corrections, reducing distributional shift.

```
# Take an initial policy: π_0, Teacher: state -> action,
# Learner: [ (state,action) ] -> policy, GenSystemTrajectory : π -> [state]
def DAGGER(π_0, Teacher, GenSystemTrajectory, Learn):
    D = [], π = π_0
    for i in range(N): # run for N iterations
        D_i = [(state,Teacher(state)) for state in GenSystemTrajectory(π) ]
        D.append(D_i)
        π = Learn(D) #Optionally run any no-regret learner on the D_i
    return π
    # Preferred: instead return the stochastic policy that mixes uniformly between all the
    # policies learned or choose the best single policy on validation over the iterations
```