Behavior Cloning

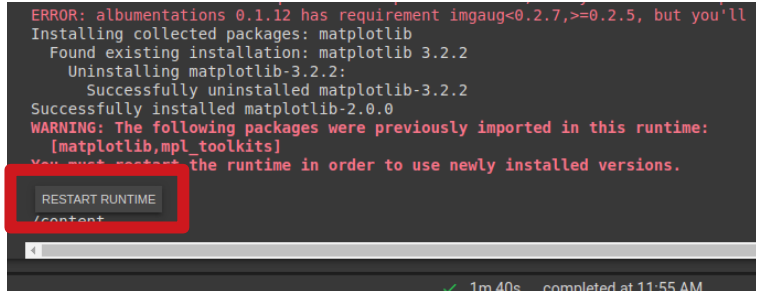
Lee Hong Jung, Jun Ho Woong
RLLAB

Overview

- Markov Decision Process (MDP)
  - Value Iteration: Simple algorithm to solve discrete MDPs
- Behavior Cloning
  - Copy expert behavior via supervised learning
    - ex) Gaussian Process Regression

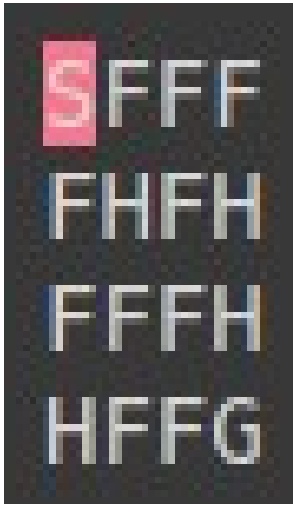
Setup

- Download zipfile from eTL
- Go to “colab.research.google.com”
- Select file “Value_Iteration.ipynb” in the “upload” tab



Markov Decision Process (MDP)

- Set of states
  - (4x4 map) 0, 1, …, 15
- Set of actions (in each state)
  - 0: left, 1: down, 2: right, 3: up
- Transition model
  - Note: In this modified version, we removed randomness!
  - Thus, the transition model is deterministic
    - Ex: For state = 0 and action = 2, next_state = 1
- Reward function
  - If state = goal_state, reward = 1
  - Otherwise, reward = 0



Modified FrozenLake gym environment
S: start state
G: goal state
F: frozen surface, safe
H: hole, AVOID!
https://gym.openai.com/envs/FrozenLake-v0/

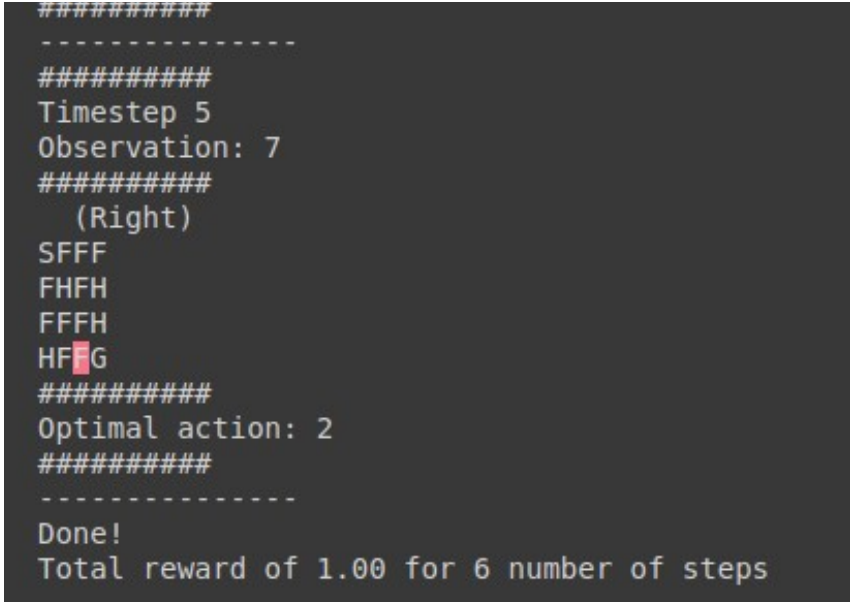
Value Iteration

- **Policy** π: $\mathcal{S} \rightarrow \{a \mid a \in Actions(s), s \in \mathcal{S}\}$ such that $\pi(s) \in Actions(s)$ for all s. $\pi(s)$ is the action recommended by the policy π for state s.
- **Optimal policy** π^*: a policy with the highest expected utility (expected sum of rewards).

Value Iteration Algorithm:

- Repeat until convergence
  - For all state s

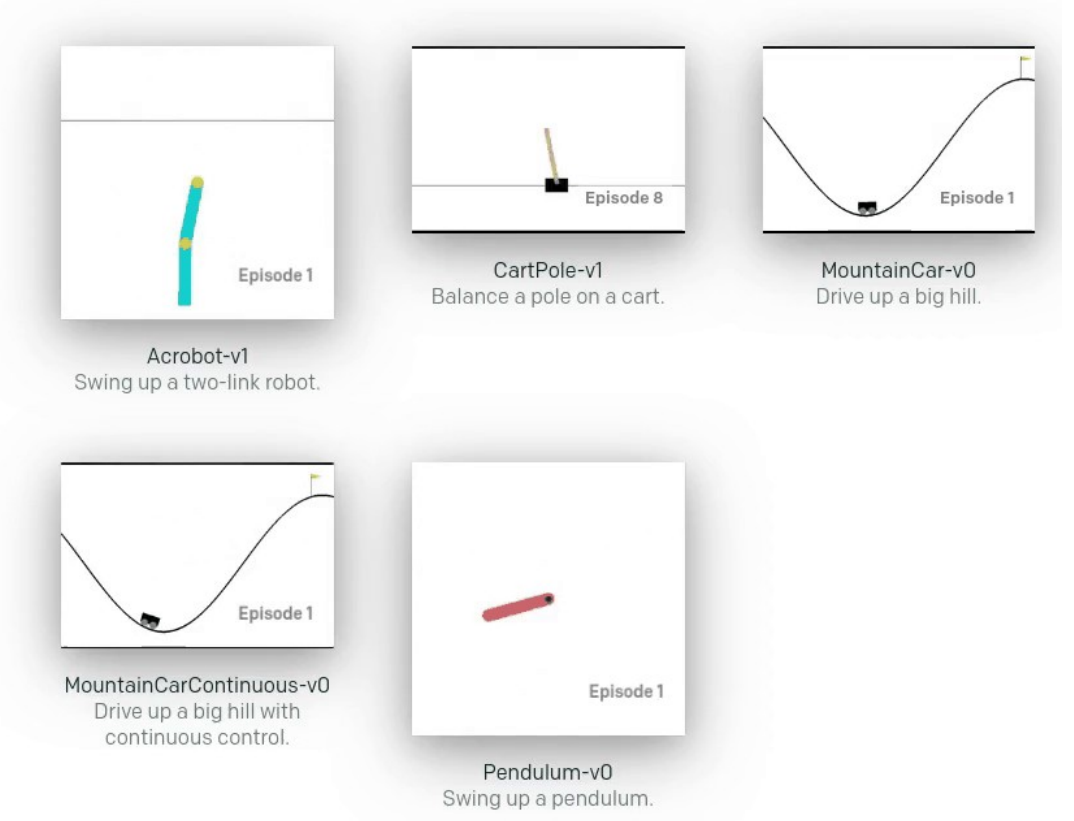
$$V(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$$



Introduction to Behavior Cloning

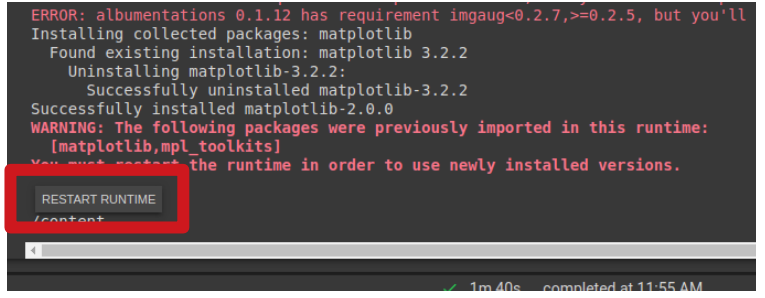
- Behavioral cloning problem
  - given control data sequence {observes, actions}
  - find controller from data (using Supervised Learning method)
  - Today, we will use Gaussian process regression (GPR)

- Gym environment
  - reference : https://gym.openai.com/envs/



Setup

- Download zipfile from eTL
- Go to “colab.research.google.com”
- Select file “Behavior_Cloning.ipynb” in the “upload” tab
- Run first cell
- RESTART RUNTIME
- Rerun first cell



Mountain Car Gym Environment

Observation

Type: Box(2)

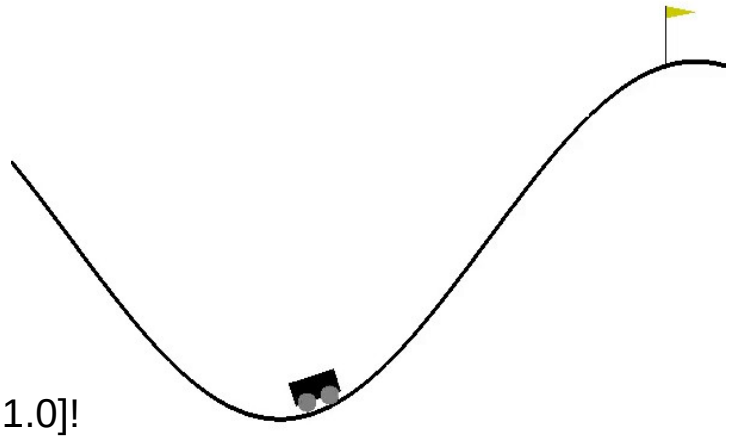
Num	Observation	Min	Max
0	Car Position	-1.2	0.6
1	Car Velocity	-0.07	0.07

Actions

Type: Box(1)

Num	Action
0	Push car to the left (negative value) or to the right (positive value)

Bounded in [-1.0, 1.0]!



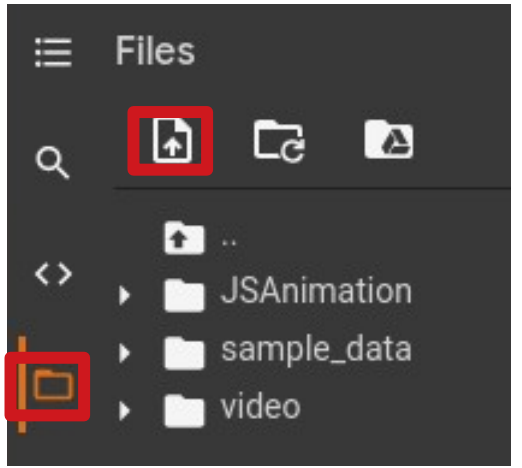
Reward

Reward is 100 for reaching the target of the hill on the right hand side, minus the squared sum of actions from start to goal.

Mountain Car Expert Behavior

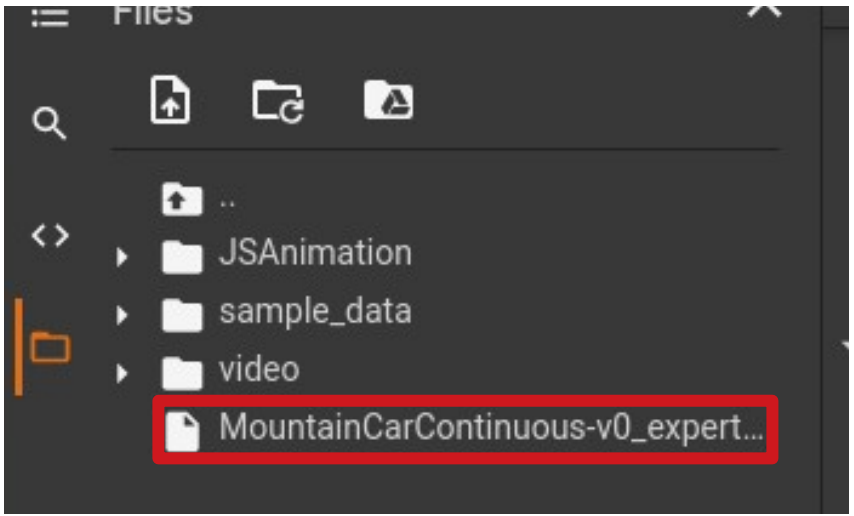
- Upload the pickle file that contains expert behavior!
  - “MountainCarContinuous-v0_expert_demo.pkl”

2. Upload!



1. Click here first!

- Check if uploaded successfully.



Mountain Car Expert Behavior

- Pickle file contains 100 expert demonstrations!
- Each expert demonstration is a dictionary with the following keys:
  - ‘rewards’
  - ‘actions’
  - ‘infos’
  - ‘observes’
- We will clone the expert behavior by using its actions given observations!

```
# Remember to upload "MountainCarContinuous-v0_expert.pkl"!
# We will now visualize how an expert's policy on the environment

env_name = "MountainCarContinuous-v0"

# Load demonstrations
with open('./' + env_name + '_expert_demo.pkl', 'rb') as f:
    demos = pickle.load(f)[0]
demos = shuffle(demos)
print("Number of expert demonstrations: {}".format(len(demos)))
print("Available information on each demonstration", list(demos[0].keys()))
print("Dimensions on components of first demonstration")
print("Rewards: {}".format(demos[0]['rewards'].shape))
print("Actions: {}".format(demos[0]['actions'].shape))
print("Infos: {}".format(len(demos[0]['infos'])))
print("Observes: {}".format(demos[0]['observes'].shape))
# Check expert's performance
exp_ret = np.mean([np.sum(d['rewards']) for d in demos])
print("Expert's Average Cumulative Rewards: {:.3f}".format(exp_ret))
```

```
Number of expert demonstrations: 100
Available information on each demonstration ['rewards', 'actions', 'infos', 'observes']
Dimensions on components of first demonstration
Rewards: (653,)
Actions: (653, 1)
Infos: 653
Observes: (653, 2)
Expert's Average Cumulative Rewards: 92.459
```

Gaussian Process Regression

- Recall, we can implement GPR easily using scikit-learn’s library!
- Normalize the demonstrations’ observations by obtaining their mean and std values!
- Finally, check out how the cloned behavior works on the environment!