DEEP REINFORCEMENT LEARNING

Markus Hofmarcher, Marius-Constantin Dinu
March 2021
LIT AI LAB
Institute for Machine Learning

JOHANNES KEPLER UNIVERSITY LINZ

RECAP: IMITATION LEARNING

Imitation Learning (IL)

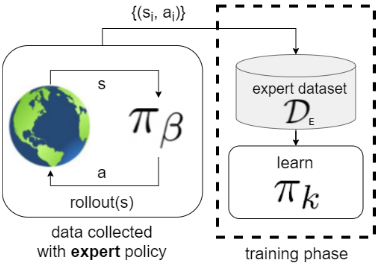
Given : Demonstrations or demonstrator
Goal : Train a policy to mimic demonstrations
Applied : When it is easier to demonstrate the desired behavior and learn a direct policy, rather than defining a reward function

Imitation Learning (IL)

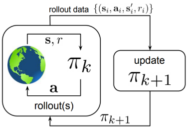
- Powerful way of overcoming problems when naive exploration is not enough, or when rewards are sparse.
- Provide good initial biases or learning initialization for online methods.
- Transfers part of the learning to a Supervised Learning setting.
  - classifier for discrete actions
  - regressor for continuous actions

RECAP: BEHAVIORAL CLONING

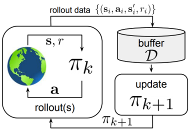
Overview



Behavioral Cloning



On-Policy

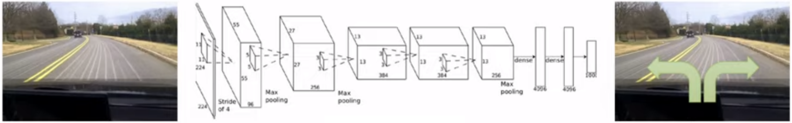


Off-Policy

Notation & Terminology

- State $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, observations $o \in \mathcal{O}$
- State transition $p(s'|s, a)$
- Trajectory τ
- Data set $\mathcal{D} = \{(s_i, a_i)\}_{1 \leqslant i \leqslant N}$ of length N
- Expert policy π^*
- Learned policy π_θ
- State distribution over the expert dataset, $p_{\mathrm{data}}(s)$
- State distribution over the learned policy, $p_{\pi_\theta}(s)$

Behavioral Cloning



Behavioral Cloning

Very Simple. The goal of Behavioral cloning (BC) is to learn a policy that is at least as good as the policy from what the demonstrations were obtained. [Bain and Sammut, 1995]

- The demonstrations must contain state and actions.
- The policy π_θ can be learned directly by Supervised Learning.

Learning objective:

$$\underset{\theta}{\operatorname{argmin}} \; \mathrm{E}_{s \sim p_{\text{data}}} \left[\mathcal{L}(\pi^*(s), \pi_\theta(s)) \right] \tag{1}$$

Behavioral Cloning Loss function

The loss function depends on the action space, for example:

- mean squared error for continuous actions

$$\mathcal{L}_{\mathrm{BC}}(\mathcal{D}, \theta) := \frac{1}{N} \sum_{k=1}^{N} (\pi_\theta(s_k) - a_k)^2 \tag{2}$$

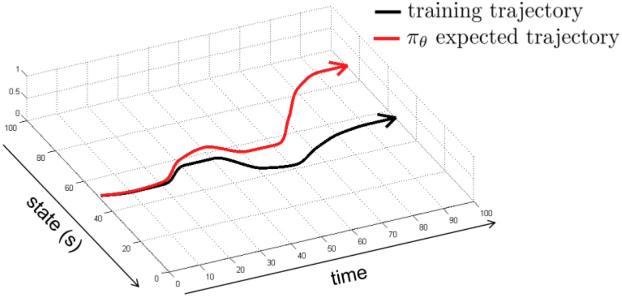
- cross-entropy loss for discrete actions

$$\mathcal{L}_{\mathrm{BC}}(\mathcal{D}, \theta) := -\frac{1}{N} \sum_{k=1}^{N} \log \pi_\theta(a_k | s_k) \tag{3}$$

Distribution Shift and Compounding Error

- BC treats Imitation Learning as a standard Supervised Learning problem.
- In Supervised Learning, we assume that training and test data are independent and identically distributed (i.i.d.).
- **Distribution Shift:** this is not the case in Imitation Learning
  - Training data can be i.i.d (i.e. sampling from a buffer)
  - Test data is not i.i.d! Policy prediction of actions affects future states via the state transition $p(s' \mid s, a)$
- **Compounding errors** accumulate and grow quadratically with time [Ross and Bagnell, 2010]

Compounding Error



Intuitively, the reason this grows quadratically in T is because as soon as π_θ makes a mistake, it could end up in new states that were not visited by π^*, and always incur maximal cost at each step from then on. [Ross and Bagnell, 2010].

INTERACTIVE EXPERT

One Solution: DAgger

- Data Aggregation, **DAgger** [Ross et al., 2011]
- Query expert at any state at training time
  - Explore with π_θ (or start exploring with π^*) and collect states
  - Build $\mathcal{D}_i = (s_i, \pi^*(s_i))$
  - Aggregate dataset $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_n$
  - Estimate π_θ with behavioral cloning on $\mathcal{D}$
  - Repeat until convergence

DAgger Algorithm

Require: Expert policy π^*, β
Initialize $\hat{\pi}_1$ to any policy in Π (i.e. random policy)
for $i = 1;\ \ i < N;\ \ i{+}{+}$ **do**
Let $\pi_i = \begin{cases} \pi^*, & \text{with probability } \beta \\ \hat{\pi}_i, & \text{with probability } (1-\beta) \end{cases}$
Sample T-step trajectories using π_i
Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$, i.e. label states with expert
Aggregate datasets $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
end for
return best $\hat{\pi}_i$ on validation

Problems DAgger doesn't solve

- Non-markovian behaviour
  - In real world, we don't have s_t but o_t.
  - Human actions are conditioned on history $\pi^*(a_t \mid o_t, o_{t-1}, ..., o_0)$
  - Causal confusion
- Multimodal behaviour
  - Output mixture of Gaussians (n modes)
  - Latent variable models (hard to train)
  - Autoregressive discretization (discretizise your actions)

EXERCISE

Exercise

- File: 03_ImitationLearning_Exercise.ipynb
- Objective: Given a dataset with expert trajectories, learn a policy that imitates the expert.
- Tasks:
    1. Implement neural network architecture
    2. Implement the DAgger algorithm
    3. Train and submit an agent

Task 1: Neural Network Architecture

- You are free in the design of your network
- Network input is fixed (no pre-processing other than what is already provided)
- Some hyper-parameters you can choose from:
  - Network architecture (type, number of layers and units, ...)
  - Activation functions
  - Learning rate (you can also implement a schedule)
  - Batch size.

Task 2: Implement DAgger

- You can initialize $\hat{\pi}_1$ randomly or with a previously trained BC policy
- Try different values for β
- You can anneal β during training, i.e.
  - Start with $\beta = 1$ (use only the export to explore)
  - Reduce β during training to let the policy explore
- You could even train (or be) your own expert

Submission

- Submission server: https://apps.ml.jku.at/challenge
- Export your trained agent as ONNX file
- Upload ONNX file to submission server
- Code is provided in 03_ImitationLearning_Exercise
- Submission closes on Friday, April 16th.
- Submit your code, best model and a short report to Moodle
- Moodle submission stays open 24h after the challenge closes
- Please upload a zip file named `k<studentid>.zip` containing:
  - `code.ipynb`
  - `model.onnx`
  - `report.pdf`

Evaluation

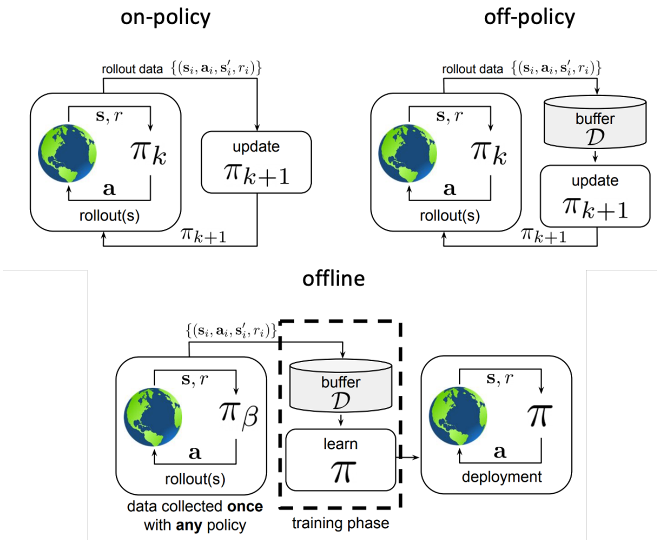
- < 200: 0 pts
- 200: 10 pts
- 300: 12 pts
- 500: 14 pts
- 600: 16 pts
- 700: 18 pts
- 800: 20 pts

BONUS: OFFLINE REINFORCEMENT LEARNING

Offline Reinforcement Learning (ORL)

- Seita's Blog: Offline (Batch) Reinforcement Learning: A Review of Literature and Applications
- Kumar's and Levine's NeurIPS Talk: Offline Reinforcement Learning: From Algorithms to Practical Challenges
- Berkeley Artificial Intelligence Research Blog: Offline Reinforcement Learning: How Conservative Algorithms Can Enable New Applications

ORL Illustration



Why ORL?

- Leverage the collected data
  - data collection is expensive and time-consuming
- Data-driven learning methods
  - similar to supervised learning setting
  - generalize based on existing data
- Avoid exploration in unsafe environments
  - robotics
  - autonomous driving

ORL vs Imitation Learning (IL)

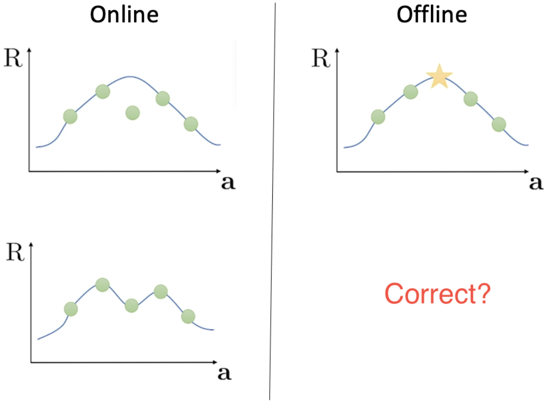
- **IL**
  - Usually only expert data / expert and non-expert data is labeled
  - Does not use the reward
  - Used for good initialization of online methods
  - May use the environment for evaluation
- **ORL**
  - May contain suboptimal data as well as expert data
  - Considers the reward
  - Never interacts with the actual environment
  - Is expected to generalize and perform well also to unseen data regions

ORL Challenges

- No exploration available
  - no solution
- Extrapolation error
- Distributional shift
  - cannot collect data to correct
- Optimizing the objective
  - simply imitating does not work well
  - policy may exploit maximization of the expected return

Off-policy methods fail if the dataset is uncorrelated to the true distribution under the current policy!

Extrapolation Error I

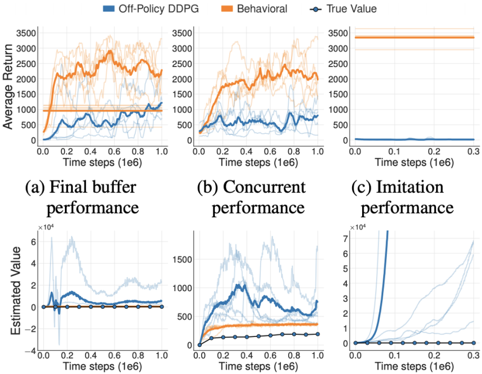


Correct?

Invalid values are used for neural network backups!

Extrapolation Error II

- Batch 1 – Final Buffer
  - DDPG agent for 1M transitions with large exploration
  - store all transitions in buffer
- Batch 2 – Concurrent
  - train off-policy and behavioral policy DDPG agents for 1M transitions concurrently
  - share the same buffer
- Batch 3 – Imitation
  - pre-trained DDPG as an expert to collect 1M transitions



(a) Final buffer performance
(b) Concurrent performance
(c) Imitation performance
(d) Final buffer value estimate
(e) Concurrent value estimate
(f) Imitation value estimate

Off-Policy Deep Reinforcement Learning without Exploration, ICML 2019, Fujimoto et al.

Constraining ORL Methods

- Policy constrained
  - works well for simpler behavioral policy distribution
  - more conservative
- Model constrained
  - data distribution has high coverage
  - when model is easy to learn
- Value-regularized
  - conservative
  - can be combined with policy and mode-based constraints
- Uncertainty-based

Batch-Constrained Deep Q-Learning (BCQ)

Algorithm 1 Batch-Constrained Q-Learning Discrete

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N. Initialize Q-network Q_θ, policy network π_ϕ with random parameters θ and ϕ, and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$ and policy network $\pi_{\phi'}$ with $\phi' \leftarrow \phi$.

for $t = 1$ **to** T **do**

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$

  Get next action: $a_i \leftarrow \text{argmax}_{a'} Q_\theta(s', a')$

  Set $y = r + \gamma \max_{a_i} Q_{\theta'}(s', a_i)$

  $\theta \leftarrow \text{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

  $\phi \leftarrow \text{argmax}_\phi \sum \ell(\pi_\phi(s), a)$

  Update target networks: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$

  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

BONUS: CHALLENGE

Bonus Challenge

- Implement BCQ
- Use the file 03_OfflineRL_BCQ
- Submit an ONNX file of your trained agent
- Submission closes at the end of the semester.
- You need to implement:
  - ☐ Dataloader
  - ☐ BCQ algorithm for discrete action space
  - ☐ Tune hyperparameters

REFERENCES

References I

[Bain and Sammut, 1995] Bain, M. and Sammut, C. (1995).
A framework for behavioural cloning.
In *Machine Intelligence 15*.

[Ross and Bagnell, 2010] Ross, S. and Bagnell, D. (2010).
Efficient reductions for imitation learning.
In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668.

References II

[Ross et al., 2011] Ross, S., Gordon, G., and Bagnell, D. (2011).
A reduction of imitation learning and structured prediction to no-regret online learning.
In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635.

JKU