Advanced Topics in Deep Reinforcement Learning

MDPs

Theresa Eimer

Recap: What Is An MDP?

- MDP == Markov Decision Process
- Formalization of RL tasks

Recap: What Is An MDP?

- MDP == Markov Decision Process
- Formalization of RL tasks
- Defined as:

$$M = \langle S, A, T, R \rangle \text{ with } T: S \times A \rightarrow S \text{ and } R: S \times A \rightarrow \mathbb{R}$$

Recap: What Is An MDP?

- MDP == Markov Decision Process
- Formalization of RL tasks
- Defined as:

$$M = \langle S, A, T, R \rangle \text{ with } T: S \times A \rightarrow S \text{ and } R: S \times A \rightarrow \mathbb{R}$$

- Additional elements that can be included in this definition:

Recap: What Is An MDP?

- MDP == Markov Decision Process
- Formalization of RL tasks
- Defined as:

  M = <S, A, T, R> with T: SxA -> S and R: SxA -> $\mathbb{R}$

- Additional elements that can be included in this definition:
  - Horizon h
  - Discount factor γ
  - Starting state distribution ϱ

Recap: What Is An MDP?

Important terminology: Markov assumption

Recap: What Is An MDP?

Important terminology: Markov assumption

- We assume that we only need the last state to take actions, not the full history
- In practice, this is not always the case
- Examples for RL environments violating the Markov assumption:

Recap: What Is An MDP?

Important terminology: Markov assumption

- We assume that we only need the last state to take actions, not the full history
- In practice, this is not always the case
- Examples for RL environments violating the Markov assumption:
  - Many gridworlds, e.g. MiniGrid
  - Robotic simulations

MDPs For Modelling The World

The standard MDP formulation is limited:

MDPs For Modelling The World

The standard MDP formulation is limited:

- Perfect information in the state
- Single Task to solve
- Task stays constant during training
- Single agent interacts with the environment

MDPs For Modelling The World

The standard MDP formulation is limited:

- Perfect information in the state
- Single Task to solve
- Task stays constant during training
- Single agent interacts with the environment

Solution: extensions to the basic MDP that meet real-world requirements

MDPs For Modelling The World

The standard MDP formulation is limited:

- Perfect information in the state
- Single Task to solve
- Task stays constant during training
- Single agent interacts with the environment

Solution: extensions to the basic MDP that meet real-world requirements

Partial Observability

Motivation: perfect information in the state is very hard to achieve in practice

POMPDs extend standard MDPs to cover this case:

$M = \langle S, A, T, R, \Omega, O \rangle$ with T: SxA -> S, R: SxA -> $\mathbb{R}$ and $O: S \times A \rightarrow \Omega$

Partial Observability

Motivation: perfect information in the state is very hard to achieve in practice

POMPDs extend standard MDPs to cover this case:

$M = \langle S, A, T, R, \Omega, O \rangle$ with T: SxA -> S, R: SxA -> $\mathbb{R}$ and $O: S \times A \rightarrow \Omega$

Ω is a set of observations

O provides an observation for the goal state s’ of a transition T(s,a) = s’

Partial Observability

Motivation: perfect information in the state is very hard to achieve in practice

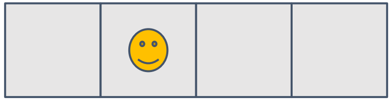
POMPDs extend standard MDPs to cover this case:

$M = \langle S, A, T, R, \Omega, O \rangle$ with $T: S \times A \rightarrow S$, $R: S \times A \rightarrow \mathbb{R}$ and $O: S \times A \rightarrow \Omega$

Ω is a set of observations

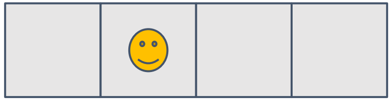
O provides an observation for the goal state s' of a transition T(s,a) = s'

Partial Observability: Example



Agent chooses action “right”

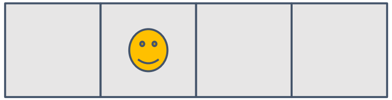
Partial Observability: Example



Agent chooses action “right”

T(1, “right”) = 2 => Agent moves to state 2 on the right

Partial Observability: Example

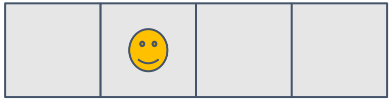


Agent chooses action “right”

T(1, “right”) = 2 => Agent moves to state 2 on the right

O(2, “right”) = “not the goal”

Partial Observability: Example



Agent chooses action “right”

T(1, “right”) = 2 => Agent moves to state 2 on the right

O(2, “right”) = “not the goal”

=> State is 2, observation is “not the goal”

Navigating Partial Observability

- Partial observability can make a task significantly harder to solve
- Important factors in partially observable environments:
  - Memory
  - Representations
- Deep RL methods on their own are often quite good at learning good representations
- Dedicated methods for partial observability exist as well

Partial Observability & Belief-State MDPs

- Solving POMDPs can also be modeled as solving belief-state MDPs
- Idea: a belief-state is a probability distribution over states
- This models the uncertainty we have about a physical system more explicitly
- Solution methods can then try to learn this probability distribution

Value Iteration in POMDPs

Instead of V(s), we are looking for V(b) for each belief state b:

While $\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$:

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S}\, b(s)\, R(s,a) + \gamma\, \Sigma_{o \epsilon O}\, Pr(o|a, b) V_t(b_{a,o})]$$

Value Iteration in POMDPs

Instead of $V(s)$, we are looking for $V(b)$ for each belief state b:

While V is still changing ($\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$):

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S}\, b(s)\, R(s,a) + \gamma\, \Sigma_{o \epsilon O}\, Pr(o|a, b) V_t(b_{a,o})]$$

Value Iteration in POMDPs

Instead of V(s), we are looking for V(b) for each belief state b:

While V is still changing ($\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$):

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S} b(s) R(s,a) + \gamma \Sigma_{o \epsilon O} Pr(o|a, b) V_t(b_{a,o})]$$

Update value of belief b

Value Iteration in POMDPs

Instead of V(s), we are looking for V(b) for each belief state b:

While V is still changing ($\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$):

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S}\, b(s)\, R(s,a) + \gamma\, \Sigma_{o \epsilon O}\, Pr(o|a, b) V_t(b_{a,o})\,]$$

Update value of belief b to value of best action a

Value Iteration in POMDPs

Instead of V(s), we are looking for V(b) for each belief state b:

While V is still changing ($\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$):

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S}\ b(s)\ R(s,a) + \gamma\ \Sigma_{o \epsilon O}\ Pr(o|a, b) V_t(b_{a,o})\ ]$$

Update value of belief b to value of best action a

Expected reward of our belief b

Value Iteration in POMDPs

Instead of V(s), we are looking for V(b) for each belief state b:

While V is still changing ($\sup_b |V_{t+1}(b) - V_t(b)| > \epsilon$):

$$V_{t+1}(b) = \max_{a \epsilon A} [\Sigma_{s \epsilon S}\, b(s)\, R(s,a) + \gamma\, \Sigma_{o \epsilon O}\, Pr(o|a, b) V_t(b_{a,o})\,]$$

Update value of belief b to value of best action a

Expected reward of our belief b

Discounted expected value of follow-up beliefs

POMDPs In Summary

- POMDPs model tasks where we can’t assume perfect information in the state
- Instead: an observation that depends on the state and last action is emitted
- This makes solving POMDPs harder than standard MDPs, but more useful when modelling real-world problems
- Solution methods either try to learn the partial observability implicitly or try to approximate a belief state about the POMDP
- Such a belief state is a probability distribution across the true state of the environment

Beyond POMDPs

- POMDPs are maybe the most common MDP extension, but far from the only one
- In general, MDP extensions either:
  - modify one or more components of the basic MDP
  - add new components
  - define a set of MDPs
- They can be helpful when designing solution algorithms with specific properties

Modelling Generalization: cMDPs

- Contextual MDPs model generalization tasks
- Formally, they are a set of MDPs, each defining an instance of a task:

  $M = \{M_c = \langle S, A, T_c, R_c \rangle \mid c \in I\}$ for context distribution I

How can we use cMDPs when solving generalization tasks?

Modelling Generalization: cMDPs

How can we use cMDPs when solving generalization tasks?

- Developing methods that focus on detecting context

Modelling Generalization: cMDPs

How can we use cMDPs when solving generalization tasks?

- Developing methods that focus on detecting context
- Building architectures that digest state and context separately

Modelling Generalization: cMDPs

How can we use cMDPs when solving generalization tasks?

- Developing methods that focus on detecting context
- Building architectures that digest state and context separately
- Defining distinct test and training distributions

Modelling Generalization: cMDPs

How can we use cMDPs when solving generalization tasks?

- Developing methods that focus on detecting context
- Building architectures that digest state and context separately
- Defining distinct test and training distributions

Use the problem structure to construct inductive biases that help us solve the problem more effectively

Other Examples Of MDP Extensions

- Block MDPs [Du et al. 2019] model observations as stochastic and correlated across a set of MDPs, making the task to learn the latent structure of the observations
- Factored MDPs use random variables to split the MDP into smaller local problems. This can be useful to exploit the structure of the problem, e.g. in multi-agent settings [Guestrin et al, 2001]
- Time-Varying MDPs [Liu & Sukhatme, 2018] aim to move away from the unchanging transition functions of standard MDPs to tasks that can evolve during training.

Variations on RL Tasks

Offline RL

Multi-Agent RL

Continual RL

Variations on RL Tasks

Offline RL	Learning from transition datasets instead of interaction
Multi-Agent RL	Multiple agents learning together
Continual RL	Learning in an ever changing task

Variations on RL Tasks

Offline RL	Standard MDP formulation
Multi-Agent RL	Factored MDPs
Continual RL	Time-varying MDP variations

But: often the exact MDP variation is not formally defined at all

Variations on RL Tasks

Offline RL

Multi-Agent RL

Continual RL

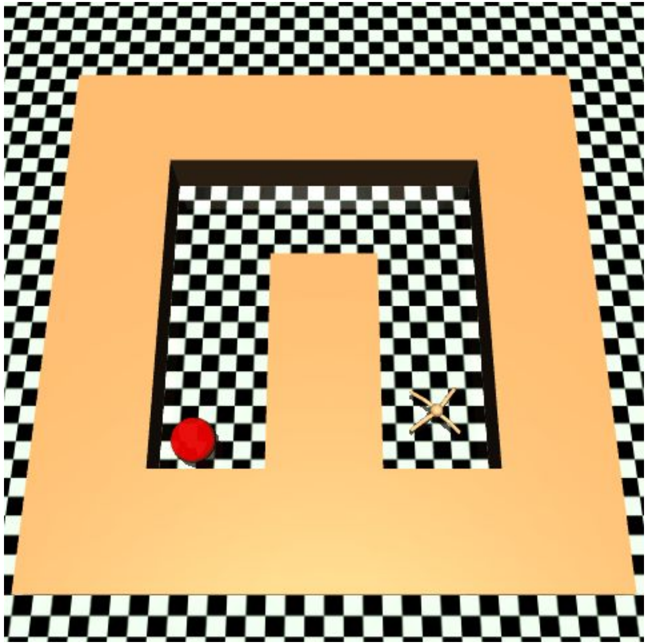
Standard MDP formulation

Factored MDPs

Time-varying MDP variations

Our Task Variation 1: Offline RL

- Task: Navigate ant through U-shaped maze
- 1M interactions of SAC & Q-Iteration
- Observations & goals in state
- 8D action space for 8 ant joints
- Binary reward signal
- Episode doesn’t terminate when reaching the goal!
- Dataset via Minari (with example for torch dataloader)

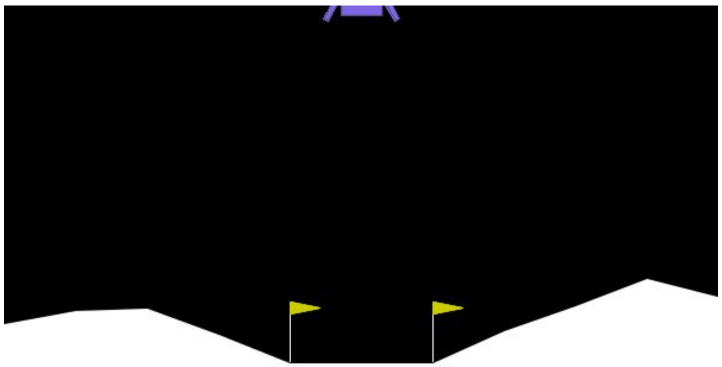


Our Task Variation 2: Multi-Agent RL

- Task: cooperatively controlling hyperparameters of a 2D DAC simulation
- Instances are sigmoid curves in two dimensions
- 100 train and 100 test instances are available
- 3 actions in each dimension
- Reward signal is a product, i.e. both dimensions have to play well together!
- Generally: relatively easy task, the goal is to be as efficient as possible

Our Task Variation 3: Continual RL

- Task: LunarLander with erratic gravity
- Gravity is either randomly re-sampled or flipped every 10.000 steps
- The goal is to still perform well and be able to adapt the policy
- Observations & actions are like in the standard LunarLander you know



My Understanding of MDPs

- ❑ I can formally define an MDP
- ❑ I have an intuition of why MDP extensions are useful
- ❑ I understand the basic concepts of partial observability

- ❑ I can formally define POMDPs
- ❑ I understand what belief states are
- ❑ I can analyze our tasks wrt. their formalism

- ❑ I understand how RL can exploit MDP problem structures
- ❑ I understand how cMDPs work
- ❑ I can imagine how other formalisms extend MDPs

Reminder: Seminar Session

- Before the session: make a PR to the repo adding your slides
- You have a maximum of 2 slides of thoughts
- Since we were very theoretical today, you can talk about whatever you want:
  - Why are we using ants? 5 reasons why dogs would be better
  - Differences between cooperative and competitive multi-agent settings
  - MDP structures are pure theory and therefore probably not useful
  - MDP structures are theory, why isn’t everyone building on them?
  - Reasons why I think zero-shot generalization is a lie
  - Our target tasks are cool and all, but here’s what I’m really interested in solving!
  - …