RL with Sequence Models

CS 285

Instructor: Sergey Levine
UC Berkeley
Beyond MDPs

-most real-world problems are like this!

o_1 \rightarrow o_2 \rightarrow o_3

s_1 \rightarrow s_2 \rightarrow s_3

doesn’t obey the Markov property

not known

o_t – observation

s_t – state
Partially observed MDPs can be weird

**Example 1:** information-gathering actions

**Example 2:** stochastic optimal policies
Which methods handle partial observability?

Policy gradients

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})A(s_{i,t}, a_{i,t}) \]

Value-based methods

\[ Q(s, a) \leftarrow r(s, a) + \gamma \max_{a'} Q(s', a') \]

Model-based RL methods

\[ \hat{p}(s'|s, a) \rightarrow \hat{p}(o'|o, a) \]

Trick question: what does “handle” mean?

\[ \pi_\theta(a'|o) \]

**handle** = find best policy in policy class
Which methods handle partial observability?

Policy gradients

$$
\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A(s_{i,t}, a_{i,t})

\text{??}

\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t}) A(o_{i,t}, a_{i,t})

$$

Key point: advantage is a function of the state $s_t$

it does not depend on $s_{t-1}$

that’s why it’s OK to use $r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$

it is not OK to train $\hat{V}(o_t)$

past observations do matter for this value

every time we see this state, we expect to get this value, regardless of past states

this is OK (no Markov property!)

this takes some care
Which methods handle partial observability?

Policy gradients

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) A(s_{i,t}, a_{i,t}) \]

??

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | o_{i,t}) A(o_{i,t}, a_{i,t}) \]

this is OK (no Markov property!)

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | o_{i,t}) \right) \left( \sum_{t=1}^{T} r(o_{i,t}, a_{i,t}) \right) \]

this takes some care

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | o_{i,t})(r_{i,t} + \gamma \hat{V}(o_{i,t+1}) - \hat{V}(o_{i,t})) \]

Pop quiz: \[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | o_{i,t}) \left( \sum_{t' = t}^{T} r(o_{i,t'}, a_{i,t'}) \right) \]
Which methods handle partial observability?

Value-based methods

\[ Q(s, a) \leftarrow r(s, a) + \gamma \max_{a'} Q(s', a') \quad \Rightarrow \quad Q(o, a) \leftarrow r(o, a) + \gamma \max_{a'} Q(o', a') \]

Remember:

**Key point:** advantage is a function of the state \( s_t \)
- it does not depend on \( s_{t-1} \)
- that's why it's OK to use \( r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \)
- it is not OK to train \( \hat{V}(o_t) \)

Value-based methods do not work without the Markov property

every time we see this state, we expect to get this value, regardless of past states

past observations do matter for this value
Which methods handle partial observability?

Model-based RL methods

\[
\hat{p}(s'|s, a) \quad \rightarrow \quad \hat{p}(o'|o, a)
\]

Example showing why this is such a bad idea:

\[ p(o' = \text{Pass}|o = \text{Left, } a = \text{Open}) = 0.5 \]

50% probability to open each time you try

Just keep trying!

\[ p(o' = \text{Pass}|o = \text{Left, } a = \text{Open}) = 0 \text{ if it didn’t open before!} \]
State space models

Can we learn a Markovian state space?

\[
\begin{align*}
\mathbf{z} & \leftarrow (z_1, z_2, \ldots, z_T) \\
\mathbf{x} & \leftarrow (o_1, o_2, \ldots, o_T)
\end{align*}
\]

\[
p(\mathbf{z}) = p(z_1) \prod_t p(z_{t+1}|z_t, a_t)
\]

\[
p_\theta(\mathbf{o}|\mathbf{z}) = \prod_t p(o_t|z_t)
\]

\[
q_\phi(\mathbf{z}|\mathbf{o}) = \prod_t q_\phi(z_t|o_{1:t})
\]

This can work quite well!

\[
Q(s, a) \leftarrow r(s, a) + \gamma \max_{a'} Q(s', a')
\]

\[
Q(o, a) \leftarrow r(o, a) + \gamma \max_{a'} Q(o', a')
\]

\[
Q(z, a) \leftarrow r(z, a) + \gamma \max_{a'} Q(z', a')
\]

Why might this not be enough?

Prediction can be hard

Maybe we don’t need good prediction to get high rewards
History states

\[
q_\phi(z | o) = \prod_{t} q_\phi(z_t | o_{1:t})
\]

state is inferred from a history

state is a function of history

can we just give the history to our value function?

\[
s_t = (o_1, o_2, \ldots, o_t)
\]

Does that work?

Does that obey the Markov property?

tells nothing we didn’t know from \(s_t\)!

\[
s_{t+1} \perp s_{t-1} | s_t
\]

\[
(o_1, \ldots, o_{t-1}, o_t)
\]

\[
Q(o_1, \ldots, o_t, a) \leftarrow r(o, a) + \gamma \max_{a'} Q(o_1, \ldots, o_{t+1}, a')
\]
Model architectures

\[ Q(o_1, \ldots, o_t, a) \leftarrow r(o, a) + \gamma \max_{a'} Q(o_1, \ldots, o_{t+1}, a') \]

how to represent this?

\[ Q_{a_1} \quad Q_{a_2} \quad Q_{a_3} \]

fixed (short) history

Is that bad?

Sometimes...

sequence model

RNN
LSTM
Transformer

\[ o_1 \quad o_2 \quad o_3 \quad o_4 \]
A practical detail...

Standard deep Q-learning:
1. Collect transition \((s, a, s')\), add to \(\mathcal{R}\)
2. Sample batch \(\{(s_i, a_i, s'_i)\}_{i=1}^{n}\) from \(\mathcal{R}\)
3. Update Q-function on batch

Deep Q-learning with history states:
1. Collect \((o_t, a_t, o_{t+1})\), get history by cat’ing \(o_1, ..., o_{t-1}\), add to \(\mathcal{R}\)
2. Sample batch \(\{(o_{1,i}, ..., o_{t,i}, a_{t,i}, o_{1,i}, ..., o_{t+1,i})\}_{i=1}^{n}\) from \(\mathcal{R}\)
3. Update Q-function on batch

Super expensive
A practical detail...

Deep Q-learning with history states:

1. Collect \((o_t, a_t, o_{t+1})\), get history by cat’ing \(o_1, ..., o_{t-1}\), add to \(\mathcal{R}\)
2. Sample batch \(\{(o_{1,i}, ..., o_{t,i}, a_{t,i}, o_{1,i}, ..., o_{t+1,i})\}_{i=1}^m\) from \(\mathcal{R}\)
3. Update Q-function on batch

\[\text{can we reuse } h_t?\]

**Key idea**: store \(h_t\) in \(\mathcal{R}\)

details a little subtle, see paper

not clear how to do w/ transformer

Recap & overview

➢ POMDPs are *weird*

➢ Some methods “just do it”
  ▪ But most efficient ones don’t, because they require value functions
  ▪ Even those that do only get the best *memoryless* policy

➢ We could *learn* a Markovian state space with models

➢ We could also just use *history states*, which just means using a sequence model to read in observation histories
RL and language models
Language models

➢ Language models are typically trained with supervised learning

➢ But we can also train them with RL if what we want is to maximize some reward function, rather than just represent the data distribution
  ▪ Why?

➢ Some questions:
  ▪ What is the (PO)MDP?
  ▪ What is the reward?
  ▪ What algorithm to use?

We have a few choices to make!
A basic formulation

\[
\begin{align*}
\pi_\theta(a|s) &= p(x_5|x_{1:4})p(x_6|x_{1:4}, x_5) \\
E_{\pi_\theta(a|s)}[r(s, a)]
\end{align*}
\]

Basic one step RL problem

context/prompt/prefix $s$

transformer

\[
\begin{align*}
\hat{x}_1 & \quad \hat{x}_2 & \quad \hat{x}_3 & \quad \hat{x}_4 & \quad \hat{x}_5 & \quad \hat{x}_6 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow
\end{align*}
\]

\[
\begin{align*}
x_0 & \quad x_1 & \quad x_2 & \quad x_3 & \quad x_4 & \quad x_5 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow
\end{align*}
\]

is capital of France? Paris <eos>

what is capital of France? Paris
Language models and policy gradients

\[ \nabla_\theta \log \pi_\theta(a|s) = \nabla_\theta \log p(x_5|x_{1:4}) + \nabla_\theta \log p(x_6|x_{1:4}, x_5) \]

\[ \nabla_\theta E_{\pi_\theta(a|s)}[r(s, a)] = E_{\pi_\theta(a|s)}[\nabla_\theta \log \pi_\theta(a|s)r(s, a)] \]

- **REINFORCE-style estimator**
  \[ \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a_i|s)r(s, a_i) \]
  samples from \( \pi_\theta(a|s) \)

- **importance-weighted estimator (e.g., PPO)**
  \[ \approx \frac{1}{N} \sum_i \frac{\pi_\theta(a_i|s)}{\bar{\pi}(a_i|s)} \nabla_\theta \log \pi_\theta(a_i|s)r(s, a_i) \]
  samples from \( \bar{\pi}(a|s) \)

Why might we prefer this?
Language models and policy gradients

\[
\nabla_{\theta} E_{\pi_\theta(a|s)}[r(s,a)] \approx \frac{1}{N} \sum_i \frac{\pi_\theta(a_i|s)}{\bar{\pi}(a_i|s)} \nabla_{\theta} \log \pi_\theta(a_i|s) r(s,a_i) \\
\hat{\nabla}(\theta, \bar{\pi}, \{a_i\})
\]

1. sample batch \( B = \{a_i\}, a_i \sim \pi_\theta(a|s) \)
2. evaluate \( r(s,a_i) \) for each \( a_i \in B \)
3. \( \bar{\pi} \leftarrow \pi_\theta \)
4. sample minibatch \( M \subset B \)
5. \( \theta \leftarrow \theta + \alpha \hat{\nabla}(\theta, \bar{\pi}, M) \)

repeat K times

importance-weighted estimator (e.g., PPO)
Learned rewards

What if $r(s,a)$ is itself a neural network?

What is the capital of France?  Paris  **reward model**  +1.0
What is the capital of France?  A city called Paris  **reward model**  +0.9
What is the capital of France?  I dunno...  **reward model**  -0.1
What is the capital of France?  London  **reward model**  -1.0
What is the capital of France?  Why such stupid question??  **reward model**  -10.0
RL from human feedback

How do we train the reward model $r_{\psi}(s, a)$?

What is the capital of France?  Paris  +1.0

What is the capital of France?  Why such stupid question??  -10.0

(“What is the capital of France? Paris”, +1.0)
(“What is the capital of France? Why such stupid question??”, -10.0)
...

How do people know these numbers?

supervised learning  reward model
Rewards from preferences

What is the capital of France?

A: Paris

B: Why such stupid question??

Prediction: given \((s, a_1, a_2)\), how likely is a person to prefer \(a_1\) over \(a_2\)?

\[
p(a_1 \text{ preferred over } a_2) = \frac{\exp(r_\psi(s, a_1))}{\exp(r_\psi(s, a_1)) + \exp(r_\psi(s, a_2))}
\]

maximize likelihood w.r.t. params \(\psi\) of \(r\)
Overall method

1. Run supervised training (or finetuning) to get initial $\pi_\theta(a|s)$
2. For each $s$ sample $K$ answers $a_k \sim \pi(a|s)$, add to dataset $D = \{(s_i, a_{i,1}, ..., a_{i,K})\}$
3. Get humans to label which $a_{i,k}$ they prefer for each $s_i$
4. Train $r_\psi$ using labeled dataset $D$
5. Update $\pi_\theta$ using RL with reward $r_\psi(s, a)$

Some issues...

- Human preferences are expensive

- Most preference data comes from the initial supervised-trained model, each iteration of RL typically adds a smaller set of preferences

- "Overoptimization"

- Reward model needs to be very good

\[ E_{\pi_\theta(a|s)}[r(s,a)] - \beta D_{KL}(\pi_\theta||\pi_\beta) = E_{\pi_\theta(a|s)}[r(s,a) + \beta \log \pi_\beta(a|s) - \beta \log \pi_\theta(a|s)]. \]

- This is model-based RL!

- Offline (model-based) RL if we only collect preferences once

- So what’s the problem?

- Reward model is typically itself a large transformer
Recap & overview

➢ We can train language models with policy gradients
   ➢ It’s a bandit problem (for now)

➢ We can use a reward model
   ➢ Typically this needs to be learned!

➢ We can learn the reward model from human preferences
   ➢ This can be more convenient than direct supervision
   ➢ This ends up being (technically) a model-based RL algorithm
   ➢ Potentially an offline model-based RL algorithm

➢ Details to take care of
   ➢ Minimize human labeling
   ➢ Overoptimization
   ➢ Use powerful reward models

\[
\sum_i \frac{\pi_\theta(a_i|s)}{\bar{\pi}(a_i|s)} \nabla_\theta \log \pi_\theta(a_i|s) r(s, a_i)
\]

Why such stupid question??

\[
\frac{\exp(r_\psi(s, a_1))}{\exp(r_\psi(s, a_1)) + \exp(r_\psi(s, a_2))}
\]

\[
E_{\pi_\theta(a|s)}[r(s, a) + \beta \log \pi_\beta(a|s) - \beta \log \pi_\theta(a|s)]
\]
Multi-step RL and language models
Multi-step RL with language models

**action**: what the bot says

**observation**: what the human says

**state**: the history $s_3 = \{o_1, a_1, o_2, a_2, o_3\}$

**reward**: dialogue outcome

- Dialogue systems
- Assistant chat bots
- Tool use (e.g., using command line tools)
- Playing games


---

This is not RLHF

---

**RLHF**

- learn from human preferences
- episode = single answer

**sequential decision making**

- learn from dialogue outcome
- episode = whole dialogue
Multi-step RL with language models

- **action**: what the bot says
- **observation**: what the human says
- **state**: the history \( s_3 = \{o_1, a_1, o_2, a_2, o_3\} \)
- **reward**: dialogue outcome

How to train?

- **Policy gradients**: requires samples from human could work, but expensive
- **Value-based methods**: could learn offline from data!
What is a time step

Per-utterance time step

\[ s_3 = \{o_1, a_1, o_2, a_2, o_3\} \]

+ natural choice, short horizons
- huge action space

Per-token time step

+ simple discrete actions
- very long horizons
Value-based RL with language models

With per-utterance time steps

\[
Q(s, a) \leftarrow r(s) + \gamma \max_{a'} Q(s', a')
\]

Q1: Any people in the shot?
A1: No, there aren’t any.

Q10: Are they facing each other?
A10: They aren’t.

pretend we’re here

could use pretrained language model, BERT, etc.
Value-based RL with language models

With per-token time steps

\[ Q(s, a) \leftarrow r(s) + \gamma \max_{a'} Q(s', a') \]

+ simple discrete actions
- very long horizons

If agent chooses next token
(else use value of dataset token)

Transformer Value Function

No there aren’t any Are they facing
Putting it all together

- Usual value-based details apply
  - Target network
  - Replay buffer
  - Double-Q trick
  - Etc.

- Can be used with either online or offline RL

- But value-based methods particularly useful in the offline setting

- That means that we need to take care of the details!
  - Handling distributional shift
  - Policy constraint: KL-divergence on actor
  - CQL-style penalty
  - IQL-style backup
  - No single best answer (yet)
Some examples

Human-Centric Dialog Training via Offline Reinforcement Learning
Jaques et al. 2020

• Actor-critic + policy constraint (KL divergence)
• Reward from human user sentiment
• Time step = utterance

CHAI: A CHatbot AI for Task-Oriented Dialogue with Offline Reinforcement Learning
Verma et al. 2022

• Q-function + CQL
• Reward from task (Craigslist negotiation)
• Time step = utterance

Offline RL for Natural Language Generation with Implicit Language Q Learning
Snell et al. 2022

• Q-function with IQL + CQL
• Policy extraction with BC actor
• Reward from task (visual dialogue)
• Time step = token
Recap & overview

- Multi-step language interactions (e.g., dialogue) are a POMDP
  - Can be defined as per-utterance or per-token

- In principle, any RL method can be used (with history states)

- In practice, we might really want an offline RL method
  - But not necessarily (e.g., text games, tools)

- Value-based methods treat either utterances or tokens as actions, build Q-functions with history states

- Same details & tricks as regular (offline) value-based methods apply