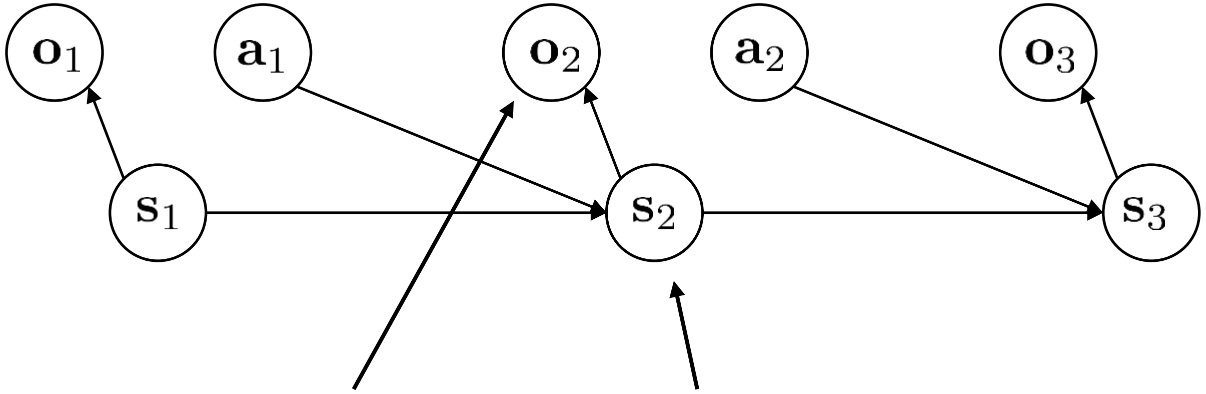
RL with Sequence Models

CS 285

Instructor: Sergey Levine
UC Berkeley

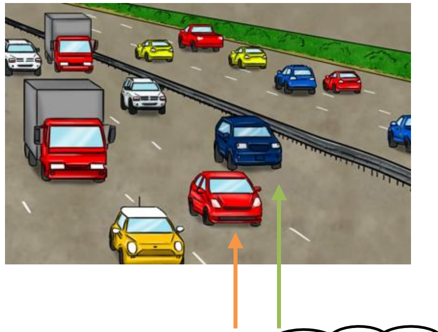
Beyond MDPs

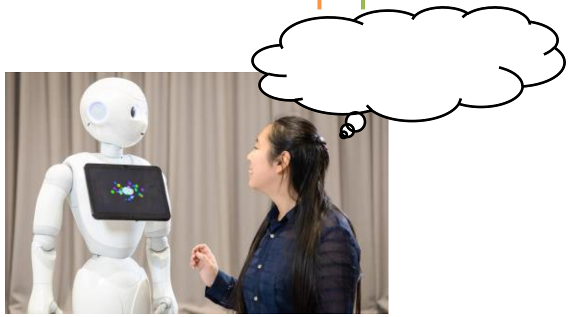
most real-world problems are like this!



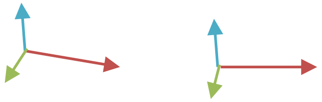
doesn’t obey the Markov property

not known

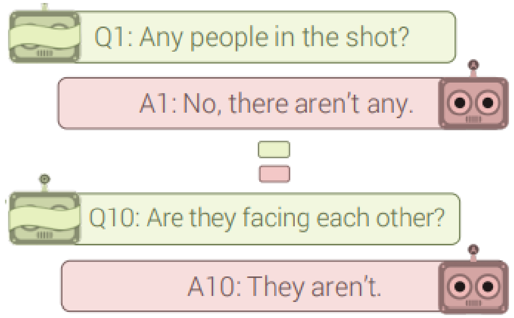




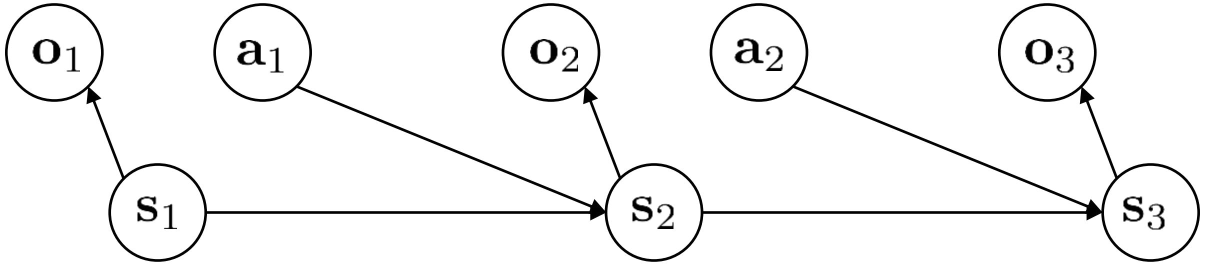
$\mathbf{o}_t$ – observation



$\mathbf{s}_t$ – state

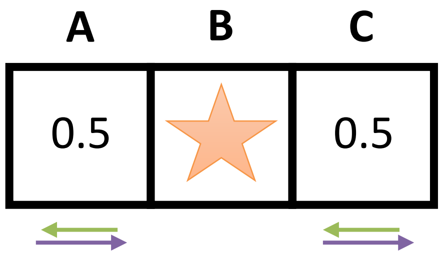


Partially observed MDPs can be weird



Example 1: information-gathering actions

Example 2: stochastic optimal policies



Which methods handle partial observability?

Policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})A(\mathbf{s}_{i,t},\mathbf{a}_{i,t}) \xrightarrow{??} \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})A(\mathbf{o}_{i,t},\mathbf{a}_{i,t})$$

Value-based methods

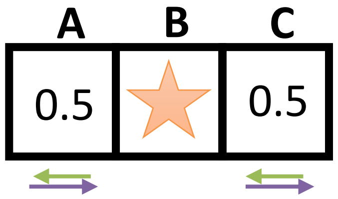
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') \xrightarrow{??} Q(\mathbf{o},\mathbf{a}) \leftarrow r(\mathbf{o},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{o}',\mathbf{a}')$$

Model-based RL methods

$$\hat{p}(\mathbf{s}'|\mathbf{s},\mathbf{a}) \xrightarrow{??} \hat{p}(\mathbf{o}'|\mathbf{o},\mathbf{a})$$

Trick question:
what does "handle" mean?

$$\pi_\theta(\mathbf{a}|\mathbf{o})$$



handle = find best policy in policy class

Which methods handle partial observability?

Policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})A(\mathbf{s}_{i,t},\mathbf{a}_{i,t}) \quad \xrightarrow{??} \quad \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})A(\mathbf{o}_{i,t},\mathbf{a}_{i,t})$$

this is OK (no Markov property!)

this takes some care

Key point: advantage is a function of the state $\mathbf{s}_t$

it does *not* depend on $\mathbf{s}_{t-1}$

that's why it's OK to use $r_t + \gamma \hat{V}(\mathbf{s}_{t+1}) - \hat{V}(\mathbf{s}_t)$

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

it is *not* OK to train $\hat{V}(\mathbf{o}_t)$

past observations **do** matter for this value

Which methods handle partial observability?

Policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})A(\mathbf{s}_{i,t},\mathbf{a}_{i,t}) \xrightarrow{??} \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})A(\mathbf{o}_{i,t},\mathbf{a}_{i,t})$$

this is OK (no Markov property!)

this takes some care

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{o}_{i,t},\mathbf{a}_{i,t})\right)$$

✓

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})(r_{i,t} + \gamma \hat{V}(\mathbf{o}_{i,t+1}) - \hat{V}(\mathbf{o}_{i,t}))$$

✗

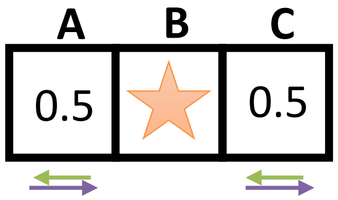
Pop quiz: $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{o}_{i,t})\left(\sum_{t'=t}^{T} r(\mathbf{o}_{i,t'},\mathbf{a}_{i,t'})\right)$$

Which methods handle partial observability?

Value-based methods

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') \xrightarrow{??} Q(\mathbf{o},\mathbf{a}) \leftarrow r(\mathbf{o},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{o}',\mathbf{a}')$$

Value-based methods do not work without the Markov property



Remember:

Key point: advantage is a function of the state $\mathbf{s}_t$

it does *not* depend on $\mathbf{s}_{t-1}$

that's why it's OK to use $r_t + \gamma \hat{V}(\mathbf{s}_{t+1}) - \hat{V}(\mathbf{s}_t)$

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

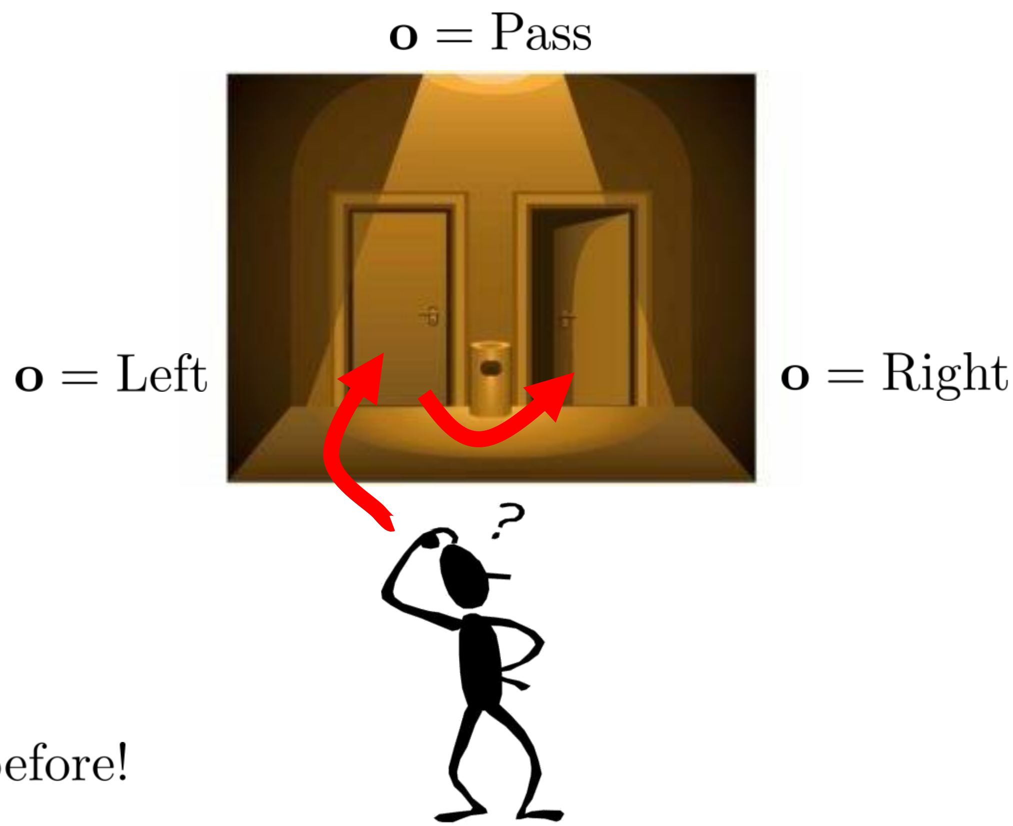
it is *not* OK to train $\hat{V}(\mathbf{o}_t)$

past observations **do** matter for this value

Which methods handle partial observability?

Model-based RL methods $\hat{p}(\mathbf{s}'|\mathbf{s},\mathbf{a}) \xrightarrow{??} \hat{p}(\mathbf{o}'|\mathbf{o},\mathbf{a})$

Example showing why this is such a bad idea:



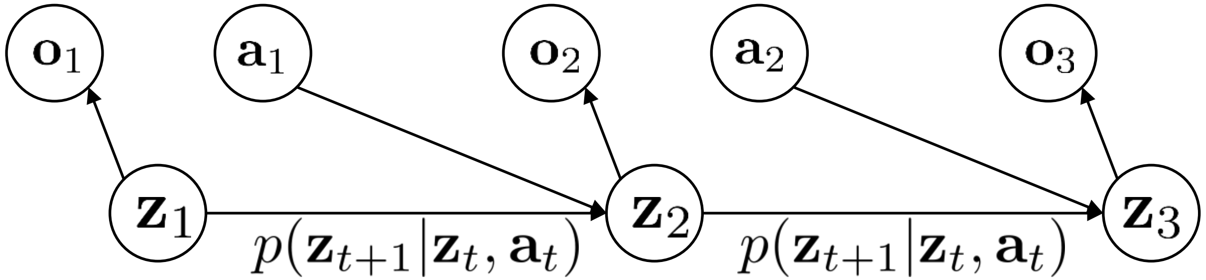
$p(\mathbf{o}' = \text{Pass}|\mathbf{o} = \text{Left}, \mathbf{a} = \text{Open}) = 0.5$

50% probability to open *each time* you try

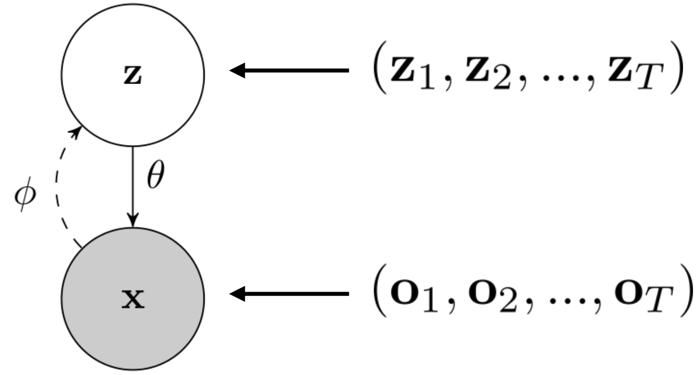
Just keep trying!

$p(\mathbf{o}' = \text{Pass}|\mathbf{o} = \text{Left}, \mathbf{a} = \text{Open}) = 0$ if it didn't open before!

State space models



Can we *learn* a Markovian state space?



$\mathcal{N}(0, \mathbf{I})$ learned

$$p(\mathbf{z}) = p(\mathbf{z}_1) \prod_t p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$$

$$p_\theta(\mathbf{o}|\mathbf{z}) = \prod_t p(\mathbf{o}_t|\mathbf{z}_t)$$

$$q_\phi(\mathbf{z}|\mathbf{o}) = \prod_t q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t})$$

This can work quite well!

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$

$$Q(\mathbf{o}, \mathbf{a}) \leftarrow r(\mathbf{o}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{o}', \mathbf{a}')$$

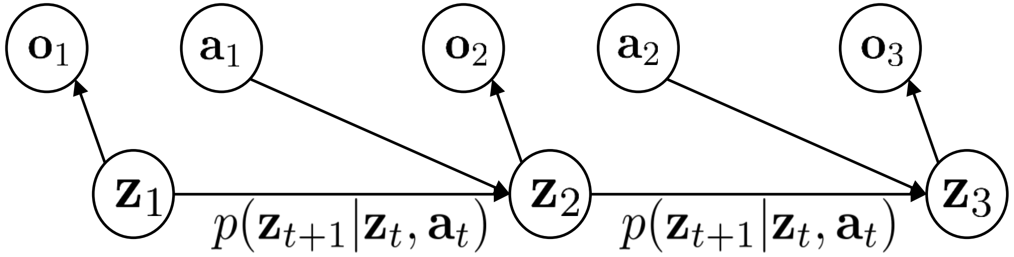
$$Q(\mathbf{z}, \mathbf{a}) \leftarrow r(\mathbf{z}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{z}', \mathbf{a}')$$

Why might this not be enough?

Prediction can be *hard*

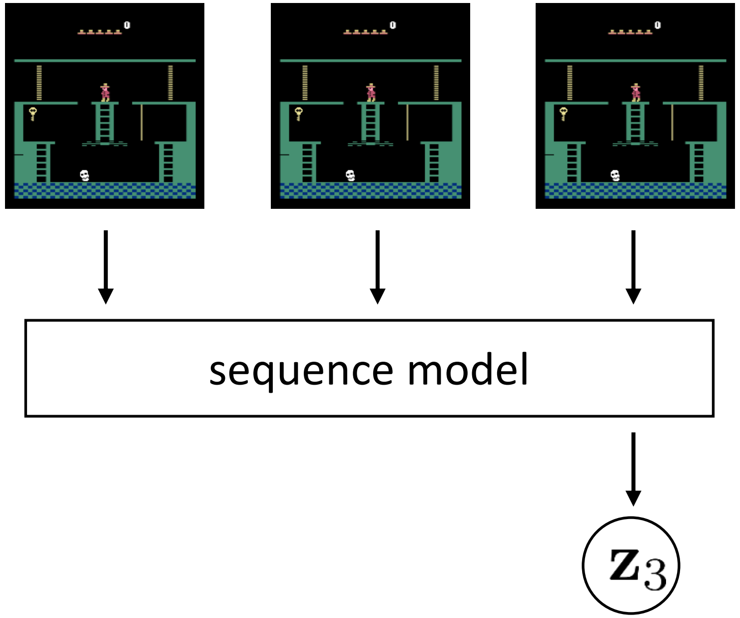
Maybe we don't need good prediction to get high rewards

History states



$$q_\phi(\mathbf{z}|\mathbf{o}) = \prod_t q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t})$$

state is inferred from a *history*



state is a *function* of history

can we just give the history to our value function?

$$\mathbf{s}_t = (\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t)$$

Does that work?

Does that obey the Markov property?

tells *nothing* we didn't know from $\mathbf{s}_t$!

$$\mathbf{s}_{t+1} \perp \mathbf{s}_{t-1} | \mathbf{s}_t$$

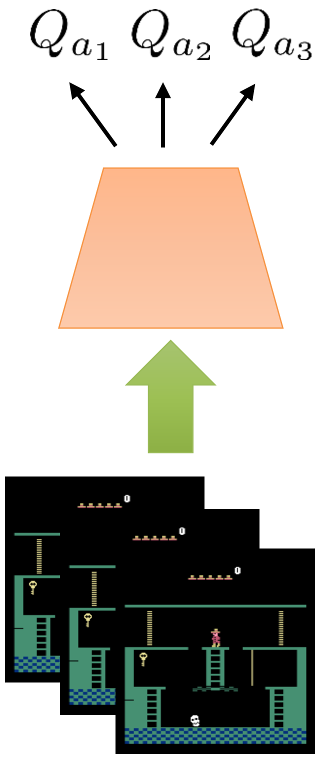
$(\mathbf{o}_1, ..., \mathbf{o}_{t-1})$ $(\mathbf{o}_1, ..., \mathbf{o}_t)$

$$Q(\mathbf{o}_1, ..., \mathbf{o}_t, \mathbf{a}) \leftarrow r(\mathbf{o}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{o}_1, ..., \mathbf{o}_{t+1}, \mathbf{a}')$$

Model architectures

$$Q(\mathbf{o}_1, ..., \mathbf{o}_t, \mathbf{a}) \leftarrow r(\mathbf{o}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{o}_1, ..., \mathbf{o}_{t+1}, \mathbf{a}')$$

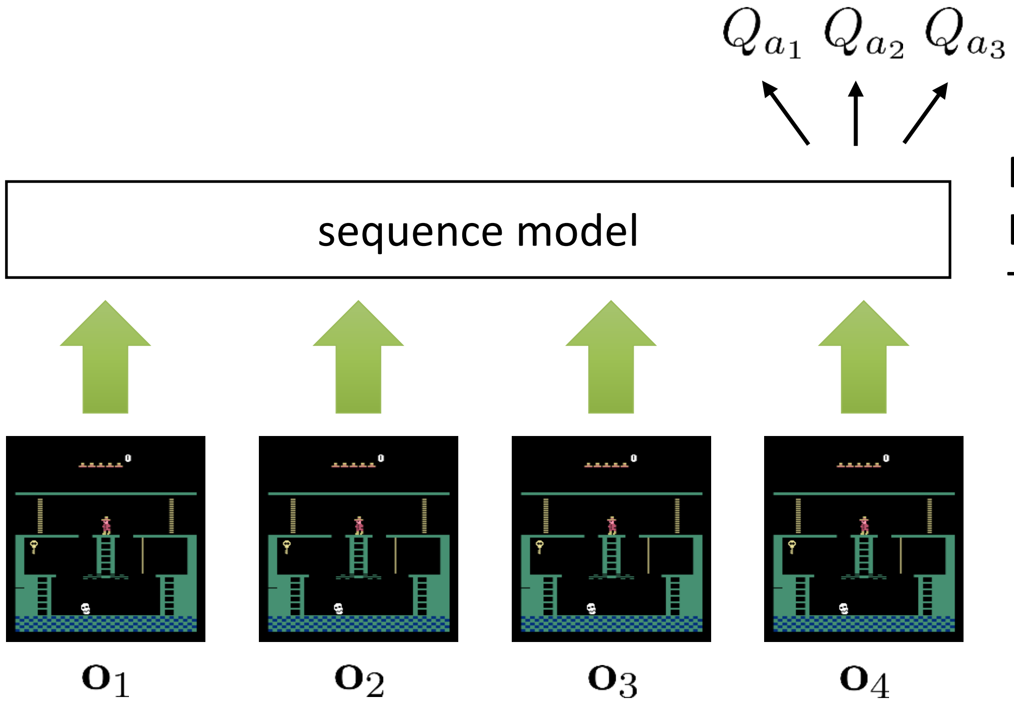
how to represent this?



fixed (short) history

Is that bad?

Sometimes...



RNN
LSTM
Transformer

A practical detail…

Standard deep Q-learning:

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

Deep Q-learning with history states:

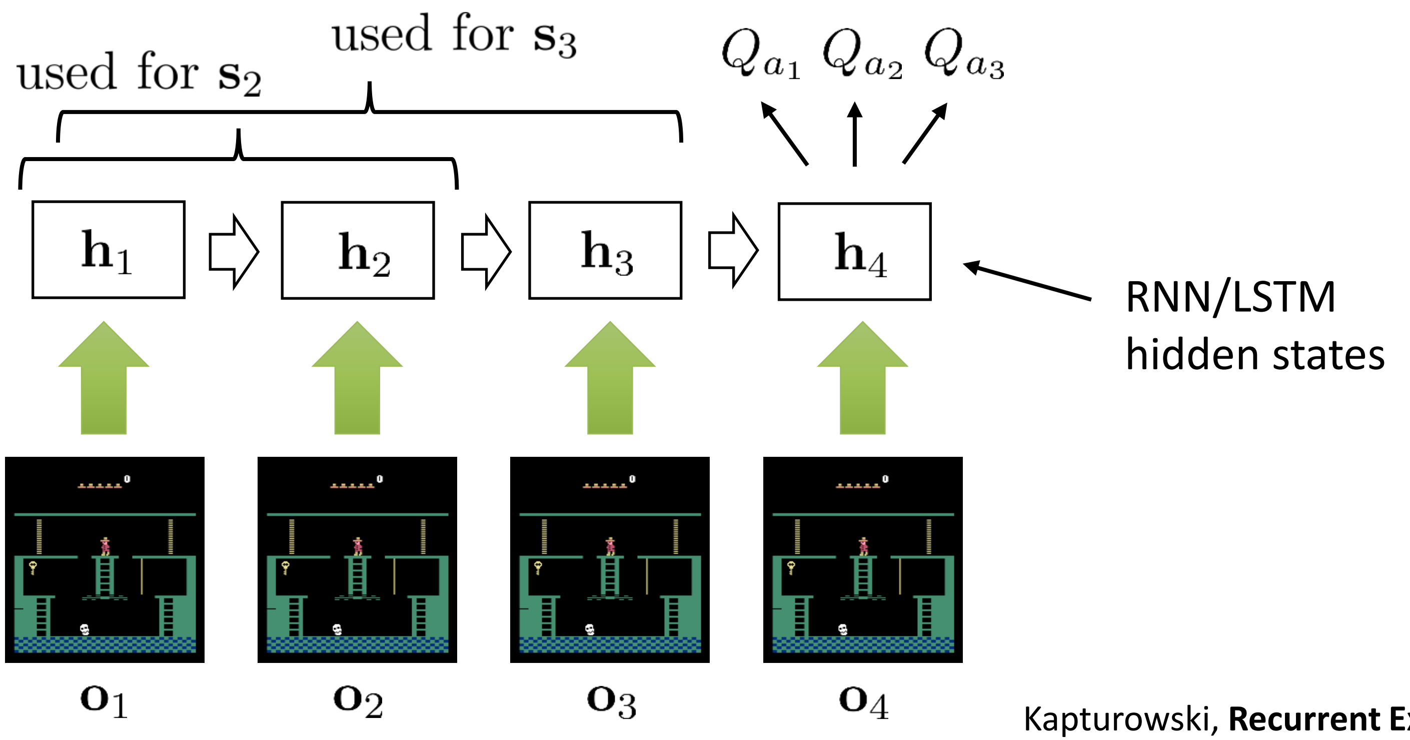
1. Collect $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$, get history by cat'ing $\mathbf{o}_1, ..., \mathbf{o}_{t-1}$, add to $\mathcal{R}$
2. Sample batch $\{(\mathbf{o}_{1,i}, ..., \mathbf{o}_{t,i}, \mathbf{a}_{t,i}, \mathbf{o}_{1,i}, ..., \mathbf{o}_{t+1,i})\}_{i=1}^n$ from $\mathcal{R}$
3. Update Q-function on batch

Super expensive

A practical detail…

Deep Q-learning with history states:

1. Collect $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$, get history by cat'ing $\mathbf{o}_1, \ldots, \mathbf{o}_{t-1}$, add to $\mathcal{R}$
2. Sample batch $\{(\mathbf{o}_{1,i}, \ldots, \mathbf{o}_{t,i}, \mathbf{a}_{t,i}, \mathbf{o}_{1,i}, \ldots, \mathbf{o}_{t+1,i})\}_{i=1}^{n}$ from $\mathcal{R}$
3. Update Q-function on batch



can we reuse $\mathbf{h}_t$?

Key idea: store $\mathbf{h}_t$ in $\mathcal{R}$

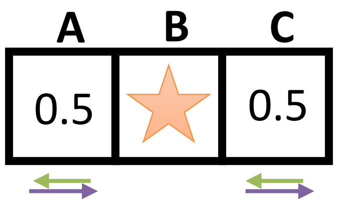
details a little subtle, see paper

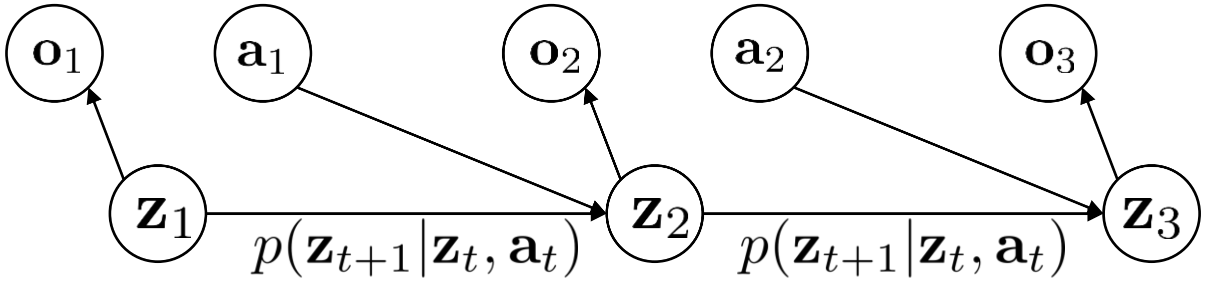
not clear how to do w/ transformer

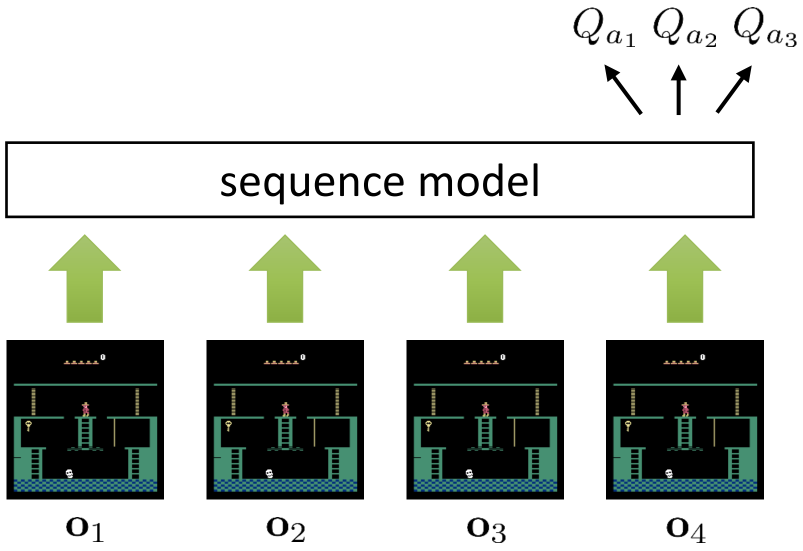
Kapturowski, **Recurrent Experience Replay in Distributed Reinforcement Learning**, ICLR 2019.

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

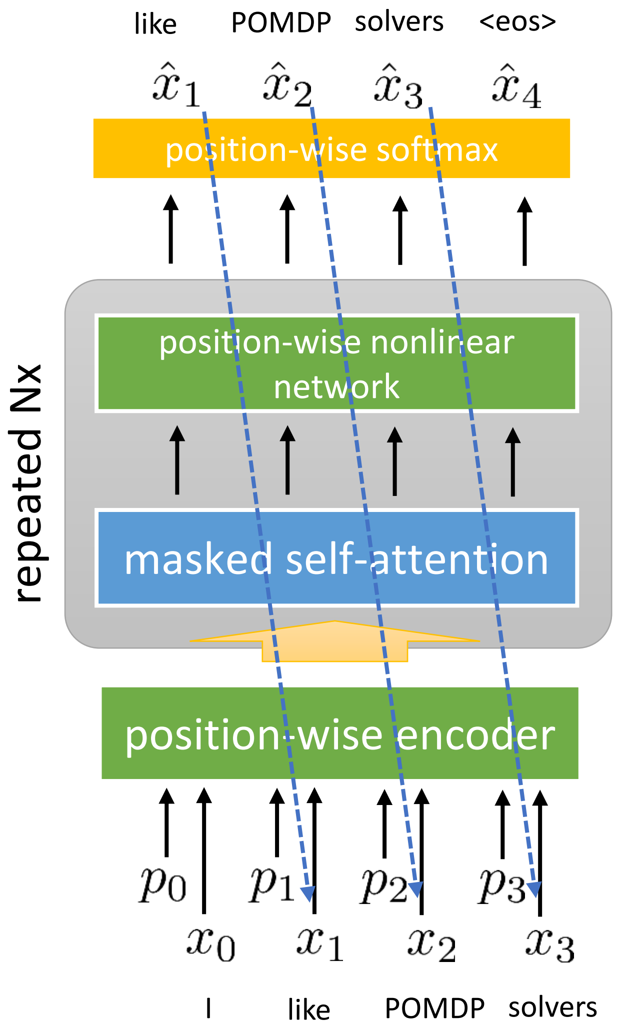




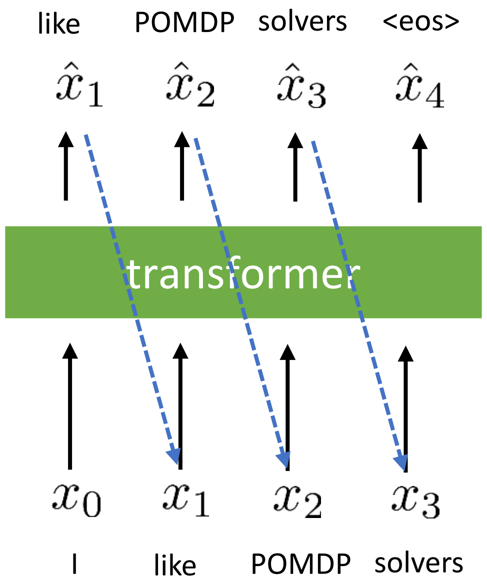


RL and language models

Language models



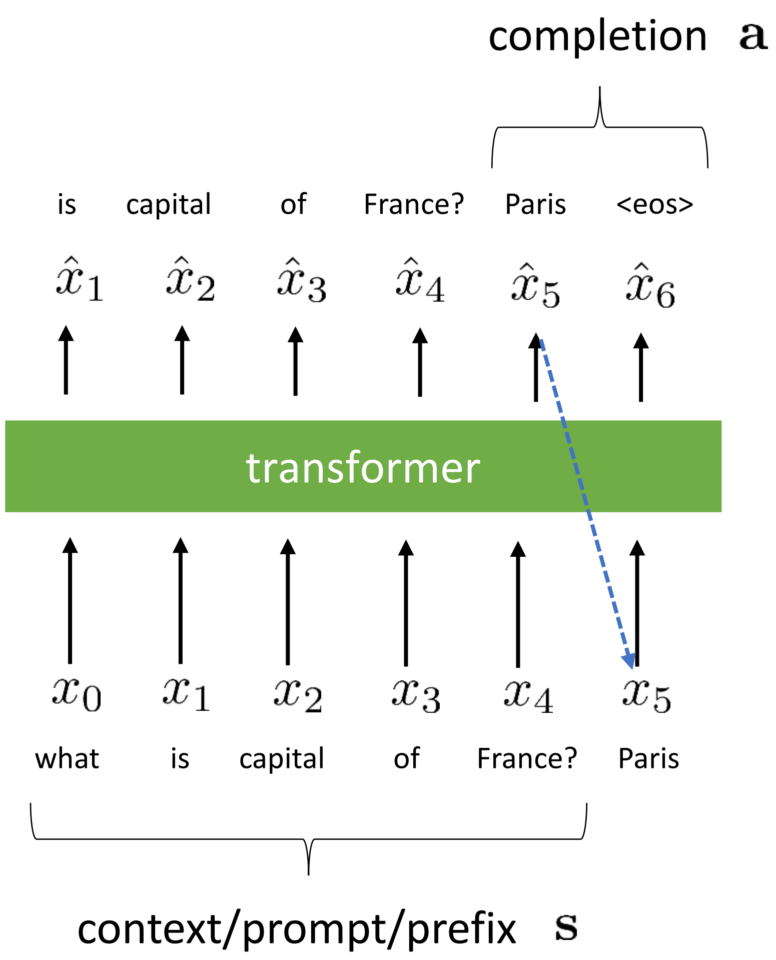
let's simplify



- 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



$\pi_\theta(\mathbf{a}|\mathbf{s})$

$$p(\mathbf{a}|\mathbf{s}) = p(x_5|\underbrace{x_{1:4}}_{\text{prompt}})p(x_6|\underbrace{x_{1:4}, x_5}_{\text{prompt}})$$

$$E_{\pi_\theta(\mathbf{a}|\mathbf{s})}[r(\mathbf{s}, \mathbf{a})]$$

Basic one step RL problem

Language models and policy gradients

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

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

samples from $\pi_\theta(\mathbf{a}|\mathbf{s})$

REINFORCE-style estimator

$$\approx \frac{1}{N}\sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}) r(\mathbf{s},\mathbf{a}_i)$$

samples from $\bar{\pi}(\mathbf{a}|\mathbf{s})$

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

$$\approx \frac{1}{N}\sum_i \frac{\pi_\theta(\mathbf{a}_i|\mathbf{s})}{\bar{\pi}(\mathbf{a}_i|\mathbf{s})} \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}) r(\mathbf{s},\mathbf{a}_i)$$

Why might we prefer this?

Language models and policy gradients

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

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

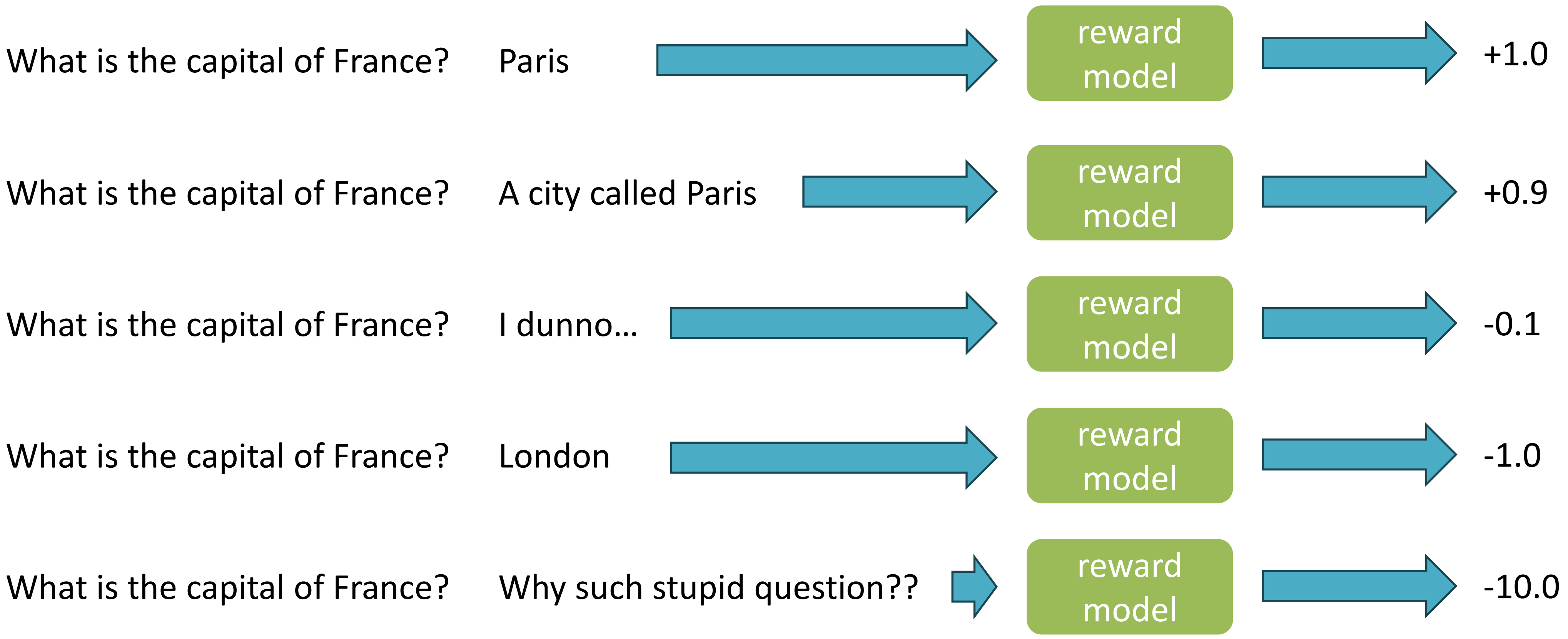
1. sample batch $\mathcal{B} = \{\mathbf{a}_i\}$, $\mathbf{a}_i \sim \pi_\theta(\mathbf{a}|\mathbf{s})$
2. evaluate $r(\mathbf{s},\mathbf{a}_i)$ for each $\mathbf{a}_i \in \mathcal{B}$
3. $\bar{\pi} \leftarrow \pi_\theta$
4. sample *minibatch* $\mathcal{M} \subset \mathcal{B}$
5. $\theta \leftarrow \theta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

what is this?

repeat K times

Learned rewards

What if $r(\mathbf{s}, \mathbf{a})$ is itself a neural network?



RL from human feedback

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

What is the capital of France? Paris → +1.0

How do people know these numbers?

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)

…

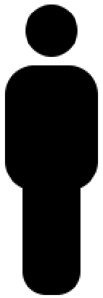
→ supervised learning → reward model

Rewards from preferences

label

A: Paris
$\mathbf{a}_1$

What is the capital of France?
$\mathbf{s}$



I prefer A

B: Why such stupid question??
$\mathbf{a}_2$

Prediction: given $(\mathbf{s}, \mathbf{a}_1, \mathbf{a}_2)$, how likely is a person to prefer $\mathbf{a}_1$ over $\mathbf{a}_2$?

maximize likelihood
w.r.t. params ψ of r

$$p(\mathbf{a}_1 \text{ preferred over } \mathbf{a}_2) = \frac{\exp(r_\psi(\mathbf{s}, \mathbf{a}_1))}{\exp(r_\psi(\mathbf{s}, \mathbf{a}_1)) + \exp(r_\psi(\mathbf{s}, \mathbf{a}_2))}$$

Overall method

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

Ziegler et al. **Fine-Tuning Language Models from Human Preferences**. 2019.

Ouyang et al. **Training language models to follow instructions with human feedback**. 2019.

Some issues…

- Human preferences are expensive

This is **model-based RL**!

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

Many iterations of RL (including generating new samples from policy) per each iteration of preference gathering

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

So what's the problem?

- "Overoptimization"

original supervised policy

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

- Reward model needs to be very good

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)

$$\sum_i \frac{\pi_\theta(\mathbf{a}_i|\mathbf{s})}{\bar{\pi}(\mathbf{a}_i|\mathbf{s})} \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}) r(\mathbf{s}, \mathbf{a}_i)$$

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

Why such stupid question?? → reward model → -10.0

- 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

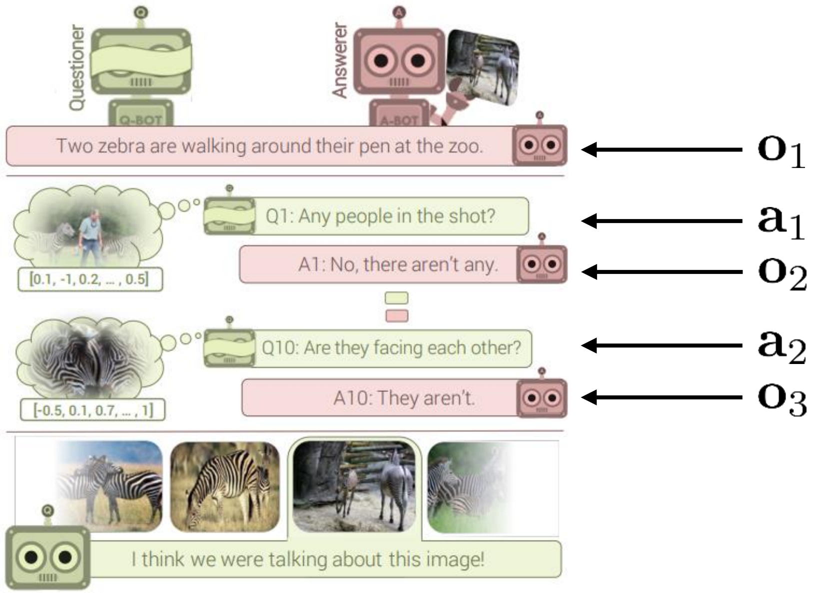
$$\frac{\exp(r_\psi(\mathbf{s}, \mathbf{a}_1))}{\exp(r_\psi(\mathbf{s}, \mathbf{a}_1)) + \exp(r_\psi(\mathbf{s}, \mathbf{a}_2))}$$

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

$$E_{\pi_\theta(\mathbf{a}|\mathbf{s})}[r(\mathbf{s}, \mathbf{a}) + \beta \log \pi_\beta(\mathbf{a}|\mathbf{s}) - \beta \log \pi_\theta(\mathbf{a}|\mathbf{s})]$$

Multi-step RL and language models

Multi-step RL with language models



Das et al. **Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning**. 2017.

action: what the bot says

observation: what the human says

state: the history $\mathbf{s}_3 = \{\mathbf{o}_1, \mathbf{a}_1, \mathbf{o}_2, \mathbf{a}_2, \mathbf{o}_3\}$

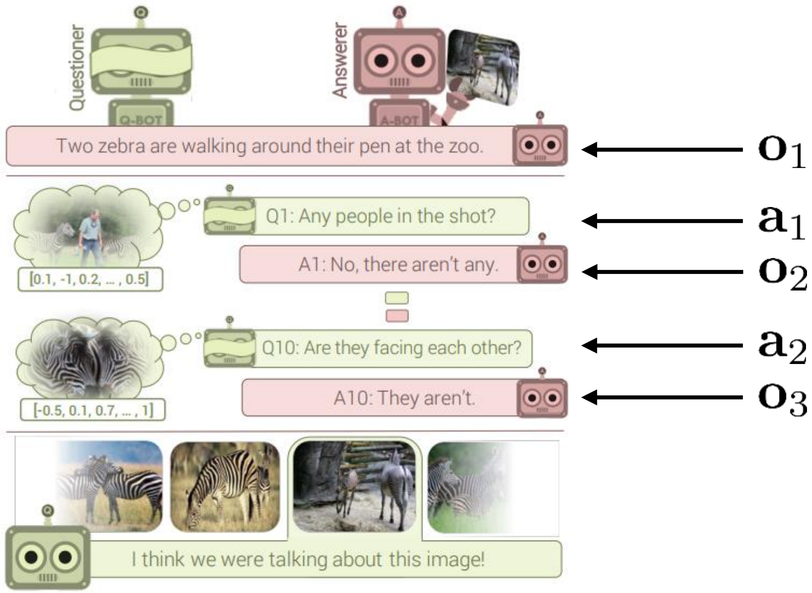
reward: dialogue outcome

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

This is **not** RLHF

<u>RLHF</u>	<u>sequential decision making</u>
learn from human preferences	learn from **dialogue outcome**
episode = single answer	episode = whole dialogue

Multi-step RL with language models



Das et al. **Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning**. 2017.

action: what the bot says

observation: what the human says

state: the history $\mathbf{s}_3 = \{\mathbf{o}_1, \mathbf{a}_1, \mathbf{o}_2, \mathbf{a}_2, \mathbf{o}_3\}$

reward: dialogue outcome

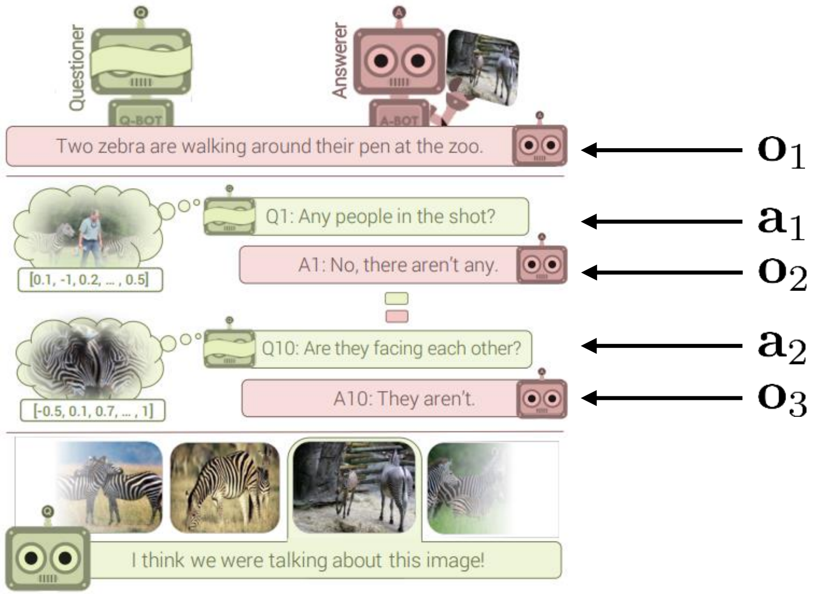
How to train?

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

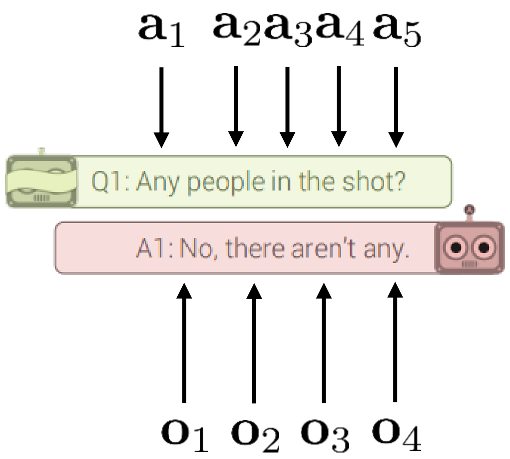


$$\mathbf{s}_3 = \{\mathbf{o}_1, \mathbf{a}_1, \mathbf{o}_2, \mathbf{a}_2, \mathbf{o}_3\}$$

+ natural choice, short horizons

- huge action space

Per-token time step



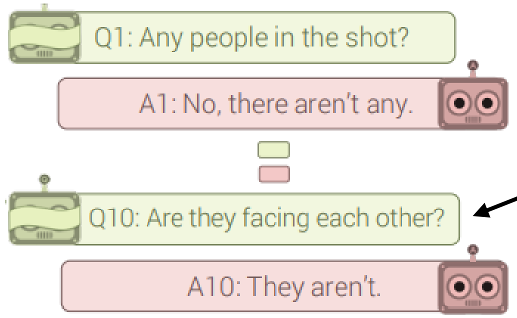
+ simple discrete actions

- very long horizons

Value-based RL with language models

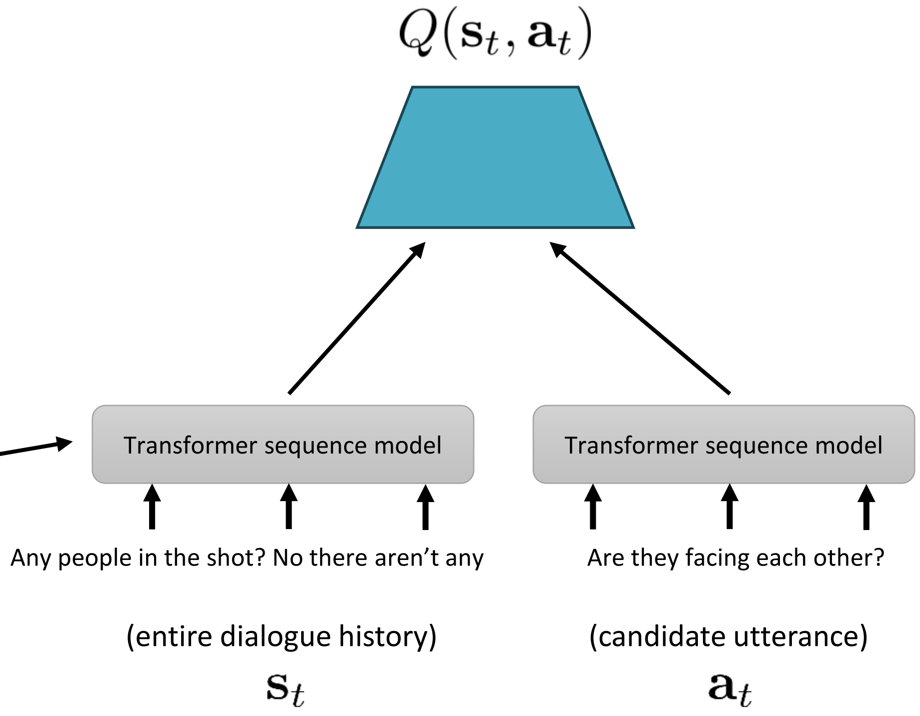
With per-utterance time steps

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$



pretend
we're here

could use pretrained
language model, BERT, etc.

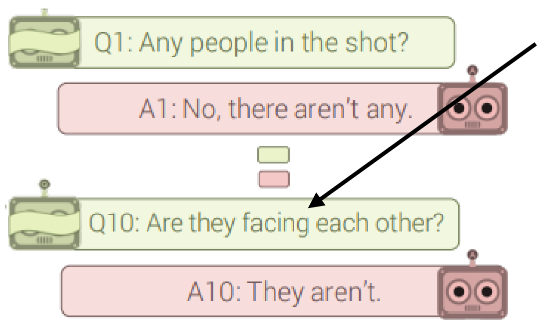


Value-based RL with language models

With per-token time steps

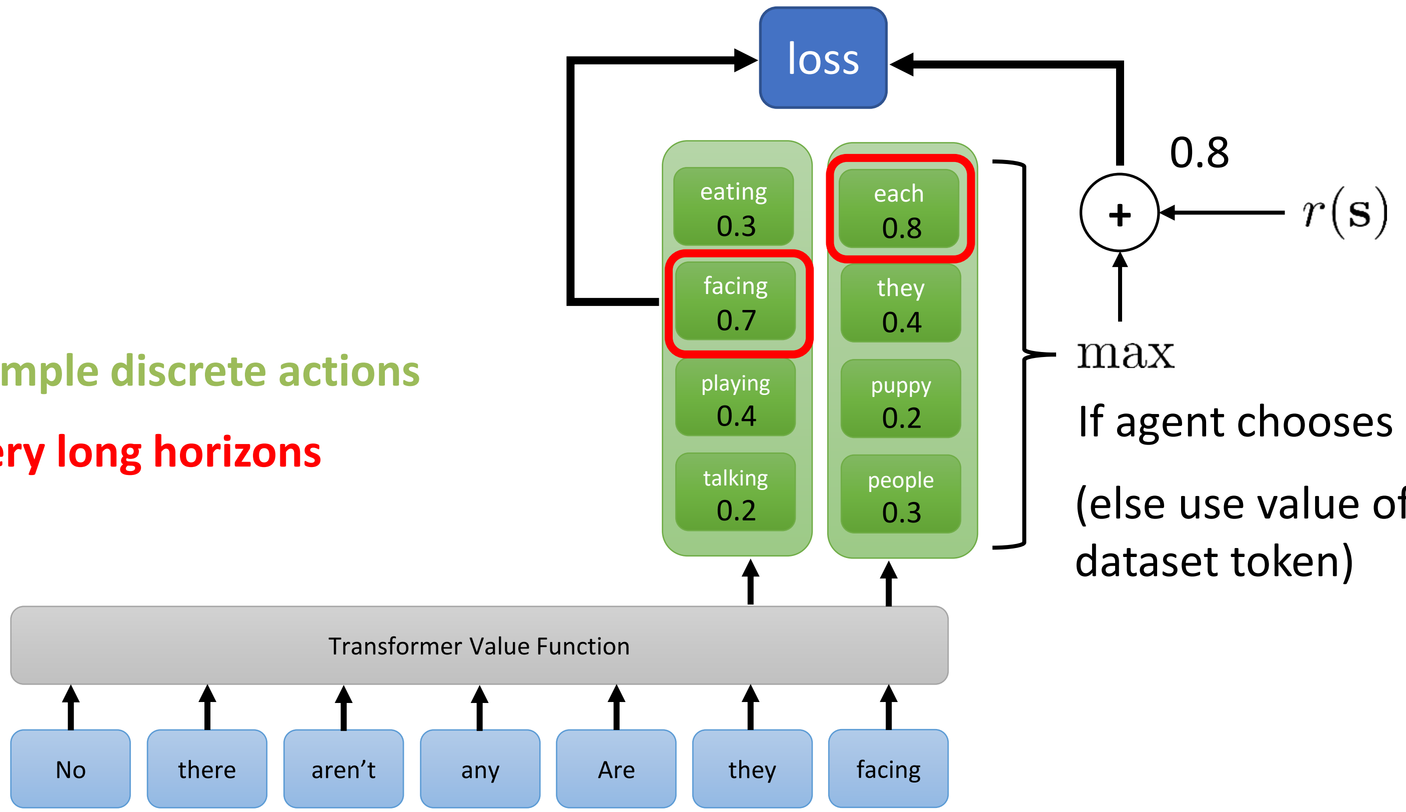
$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$$

pretend
we're here



+ simple discrete actions

- very long horizons



If agent chooses next token

(else use value of dataset token)

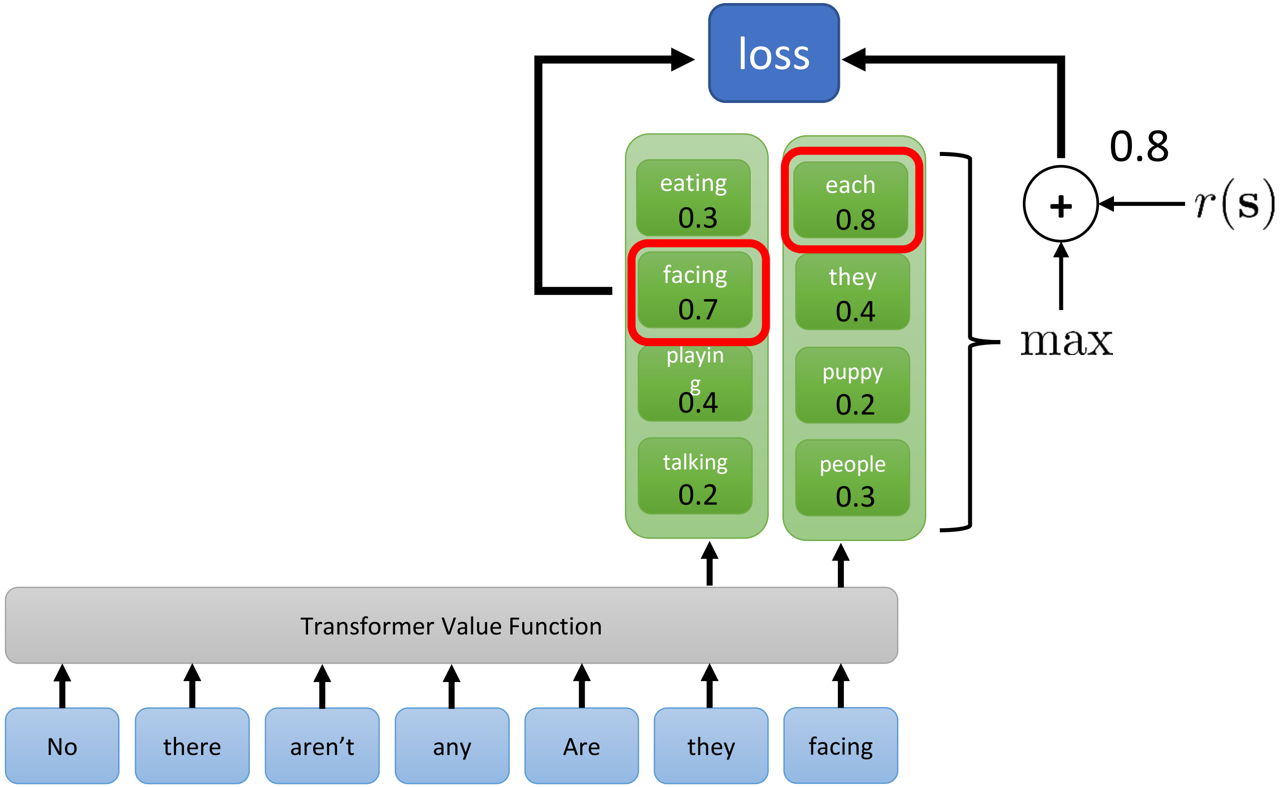
Putting it all together

- 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

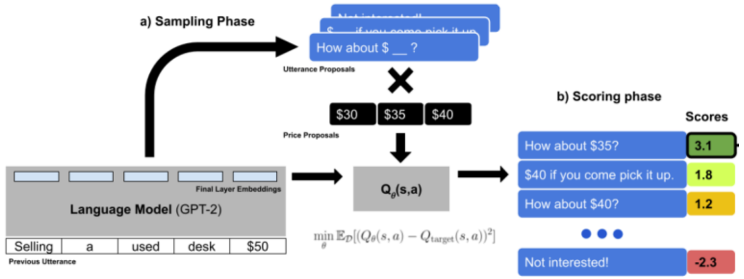
Human-Centric Dialog Training via Offline Reinforcement Learning

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

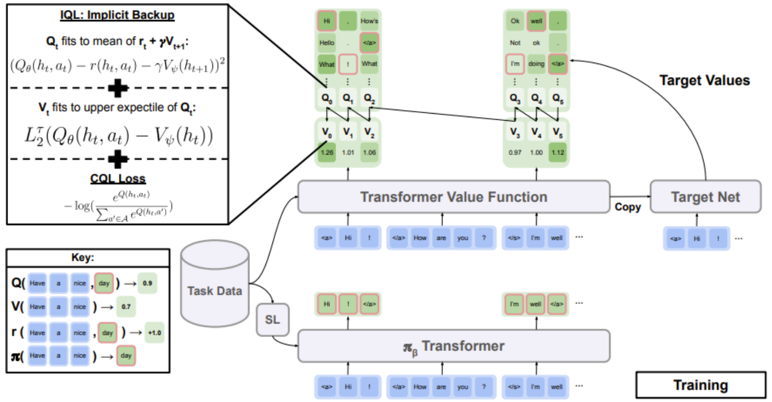
Verma et al. 2022



- 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