Imagination and Curiosity

John Canny

Spring 2018

Lecture 22 of CS194/294-129: Designing, Visualizing and Understanding Deep Neural Networks

Some slides borrowed from S. Levine et al. "Deep Reinforcement Learning"

Last Time: Q Functions

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=1}^{T} \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)}_{\text{"reward to go"}}$$

$\hat{Q}_{i,t}$

Define an expected value estimator Q:

Last Time: Q Functions

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=1}^{T} \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)}_{\text{"reward to go"}\ \hat{Q}_{i,t}}$$

Define an expected value estimator Q:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta} \left[\gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t \right] \quad : \text{true } \textit{expected} \text{ reward-to-go}$$

Last Time: Q Functions

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=1}^{T}\gamma^{t'-t} r(\mathbf{s}_{i,t'},\mathbf{a}_{i,t'})\right)}_{\text{“reward to go”}\ \hat{Q}_{i,t}}$$

Define an expected value estimator Q:

$$Q(\mathbf{s}_t,\mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[\gamma^{t'-t} r(\mathbf{s}_{t'},\mathbf{a}_{t'})|\mathbf{s}_t,\mathbf{a}_t\right] \quad : \text{true } \textit{expected} \text{ reward-to-go}$$

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

Last Time: Advantage functions

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[\gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

: true *expected* reward-to-go
total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

Value function $V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from $\mathbf{s}_t$

Advantage function $A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$: how much better $\mathbf{a}_t$ is

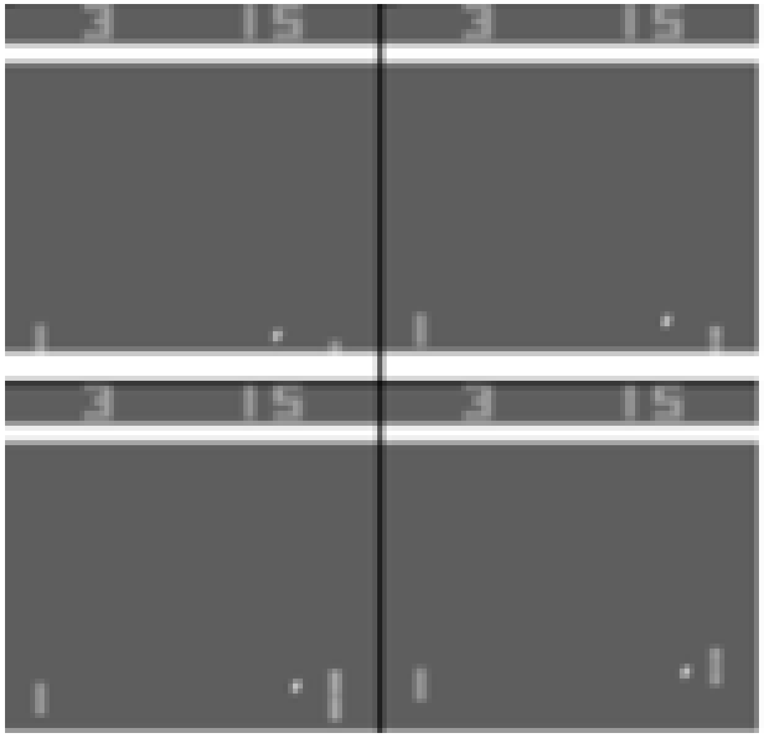
Advantage functions are typically ***sparse*** functions of state.

i.e. for many environments, most states have action advantages = 0.

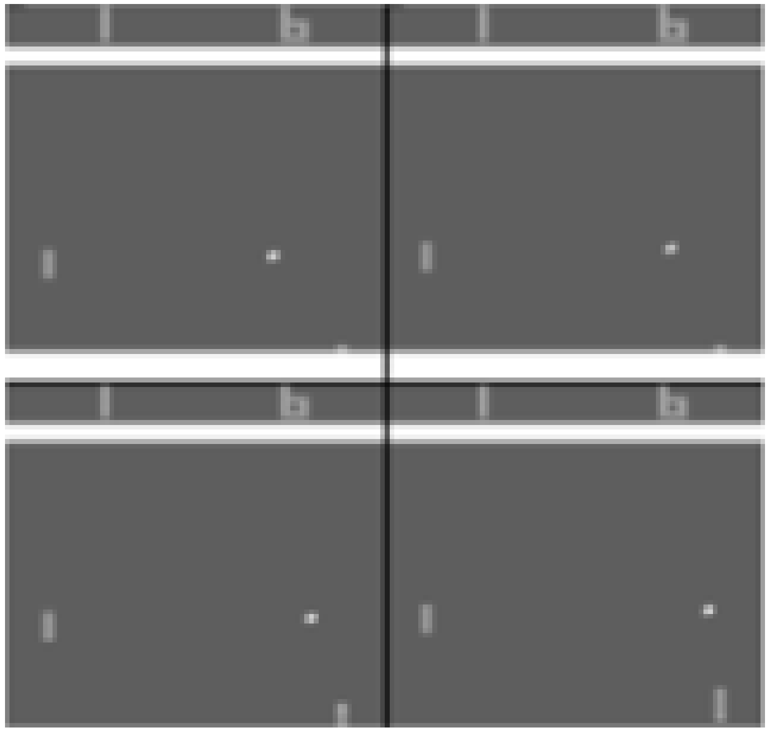
Advantage function Sparseness

Atari Pong, player on the right:

High advantage for some action:



Advantages close to zero
(action doesn’t affect reward)
true for most states:



Advantage functions address the ***temporal credit assignment problem***. Advantage focuses updates on actions that have high advantage, and ignores others:

Advantage function Sparseness

Advantage functions address the ***temporal credit assignment problem***. Advantage focuses updates on actions that have high advantage, and ignores others.

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

Actor-critic gradient: $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i | \mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$

For the sequence $(s_0, a_0, s_1, a_1, s_2, a_2, s_3, a_3, s_4, a_4, s_5, a_5, \ldots)$
Advantages: 0, 0, 0, 1.2, 0, 0

Backpropagated gradient

Gradients only backpropagate to actions with non-zero advantage, i.e. only actions which affect the reward are reinforced.

Last Time: Actor-critic algorithms

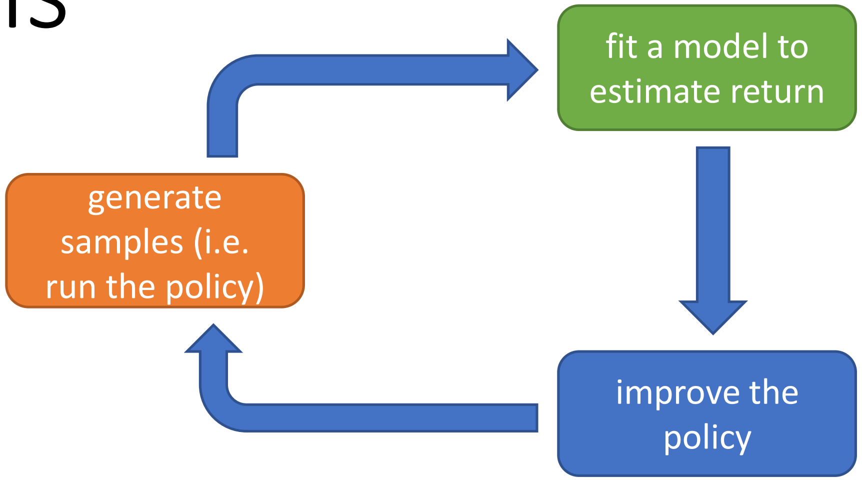
batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

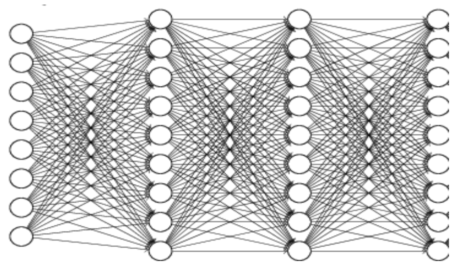
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}^\pi_\phi$ using target $r + \gamma \hat{V}^\pi_\phi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}^\pi_\phi(\mathbf{s}') - \hat{V}^\pi_\phi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

fit $\hat{V}^\pi$



$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$\mathbf{s}$



$\hat{V}^\pi(\mathbf{s})$

parameters ϕ

Last Time: "Classic" deep Q-learning algorithm (DQN)

"classic" deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

$$\left.\begin{array}{l}\text{2.}\\ \text{3.}\\ \text{4.}\end{array}\right\} K = 1$$

Mnih et al. '13

Last Time: "Classic" deep Q-learning algorithm (DQN)

Save to and sample from a replay buffer

"classic" deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

$K = 1$

Mnih et al. '13

Last Time: "Classic" deep Q-learning algorithm (DQN)

"classic" deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

$K = 1$

Mnih et al. '13

Last Time: “Classic” deep Q-learning algorithm (DQN)

“classic” deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

$K = 1$

Use a target network, updated periodically

Mnih et al. ‘13

Last Time: "Classic" deep Q-learning algorithm (DQN)

"classic" deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

$K = 1$

Mnih et al. '13

Course Logistics

- Guest lectures next week by Chelsea Finn and Andrew Critch.

- Please try to attend in person.

- The course survey is on Wednesday, you can get that done as well...

Course Logistics

M 4/23	RL Capstone: Learning to Learn. **Chelsea Finn Guest Lecture.**		Assignment 4 due 11pm (pdf to Gradescope and zipfile to Bcourses)
W 4/25	Risks. **Andrew Critch Guest Lecture.**		
Su 4/29			Final Project Presentation due 11pm
M 4/30	Final project presentations I	4-5:30pm in 306 Soda Hall	Final Project Presentation
W 5/2	Final project presentations II	12:30-2:30pm in 306 Soda Hall	Final Project Presentation
Th 5/3	Final project poster session	3-4:30pm in Soda 5th floor atrium	Final Project Poster due
F 5/11			Final Project Report due

This Time: Curiosity and Imagination

This Time: Curiosity and Imagination

- Exploration vs. Exploitation

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
    - Optimistic exploration

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
  - Optimistic exploration
  - Posterior sampling

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
  - Optimistic exploration
  - Posterior sampling
  - Curiosity-driven exploration

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
  - Optimistic exploration
  - Posterior sampling
  - Curiosity-driven exploration

- Model-free vs. model-based methods

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
  - Optimistic exploration
  - Posterior sampling
  - Curiosity-driven exploration

- Model-free vs. model-based methods
  - Visual prediction

This Time: Curiosity and Imagination

- Exploration vs. Exploitation
- Exploration Methods:
  - Optimistic exploration
  - Posterior sampling
  - Curiosity-driven exploration

- Model-free vs. model-based methods
  - Visual prediction
  - An imagination architecture

What’s the problem?

What's the problem?

this is easy (mostly)

What's the problem?

this is easy (mostly)

this is impossible

What's the problem?

this is easy (mostly)

Why?

this is impossible

Montezuma’s revenge

Montezuma's revenge

- Getting key = reward

Montezuma’s revenge

- Getting key = reward
- Opening door = reward

Montezuma’s revenge

- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)

Montezuma’s revenge

- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)
- Finishing the game only weakly correlates with rewarding events

Montezuma's revenge

- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)
- Finishing the game only weakly correlates with rewarding events
- We know what to do because we **understand** what these sprites mean!

Exploration and exploitation examples

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement
  - Exploration: show a different random advertisement

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement
  - Exploration: show a different random advertisement
- Oil drilling

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement
  - Exploration: show a different random advertisement
- Oil drilling
  - Exploitation: drill at the best known location

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement
  - Exploration: show a different random advertisement
- Oil drilling
  - Exploitation: drill at the best known location
  - Exploration: drill at a new location

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Classes of exploration methods in deep RL

Classes of exploration methods in deep RL

- Optimistic exploration:

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses

- Thompson sampling style algorithms:

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies
  - sample and act according to sample

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies
  - sample and act according to sample
- Information gain style algorithms

Classes of exploration methods in deep RL

- Optimistic exploration:
  - new state = good state
  - requires estimating state visitation frequencies or novelty
  - typically realized by means of exploration bonuses
- Thompson sampling style algorithms:
  - learn distribution over Q-functions or policies
  - sample and act according to sample
- Information gain style algorithms
  - reason about information gain from visiting new states

Optimistic exploration in RL

UCB (Upper Confidence Bound) methods for MDPs

Define $N(s)$ as the number of times we have visited state s,
or $N(s,a)$ as the number of times we performed action a in state s.

Add an intrinsic reward of bonus for visiting rarely-visited states:

use $r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

bonus that decreases with $N(\mathbf{s})$, e.g. $\mathcal{B}\big(N(s)\big) = \sqrt{\frac{2 \ln T}{N(s)}}$

use $r^+(\mathbf{s},\mathbf{a})$ instead of $r(\mathbf{s},\mathbf{a})$ with any model-free algorithm

+ simple addition to any RL algorithm

- need to tune bonus weight

The trouble with counts

use $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

The trouble with counts

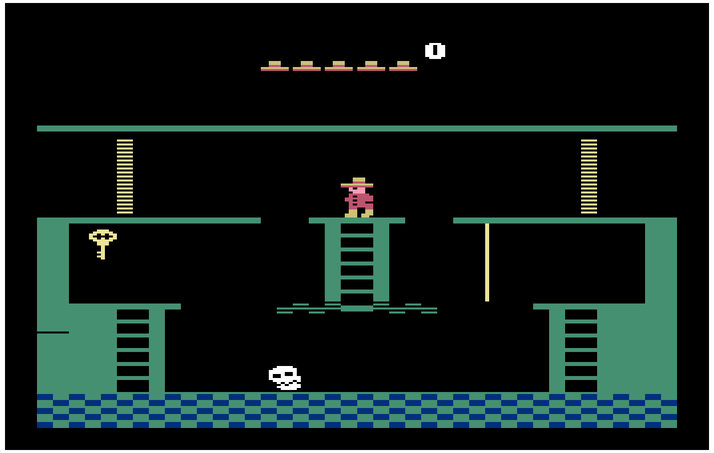
$$\text{use } r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

But wait… what's a count?

The trouble with counts

$$\text{use } r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

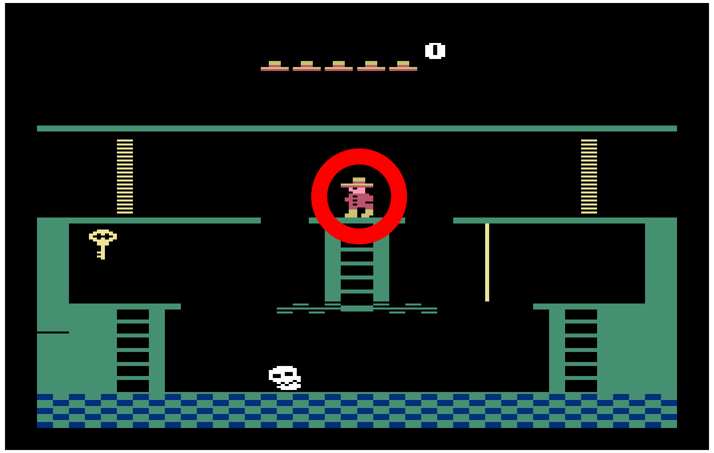
But wait… what's a count?



The trouble with counts

$$\text{use } r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

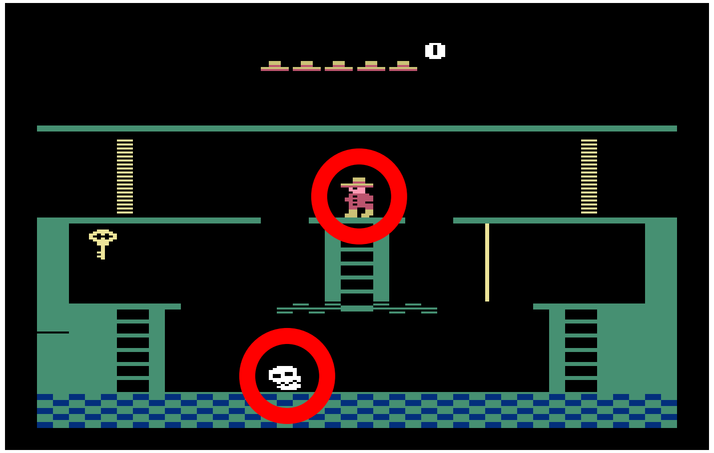
But wait… what's a count?



The trouble with counts

use $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

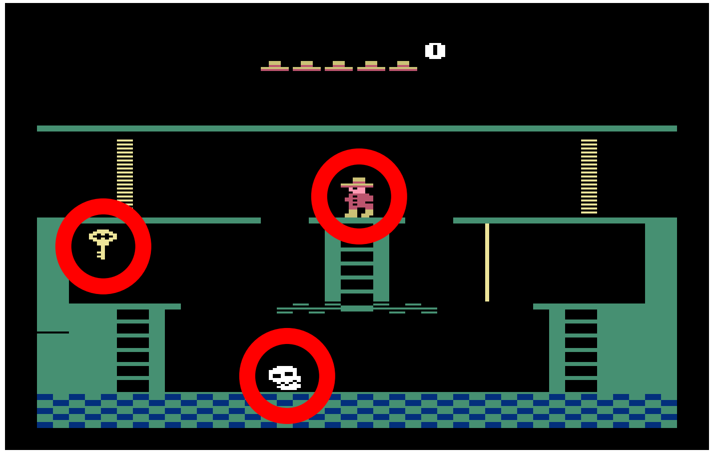
But wait… what's a count?



The trouble with counts

$$\text{use } r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

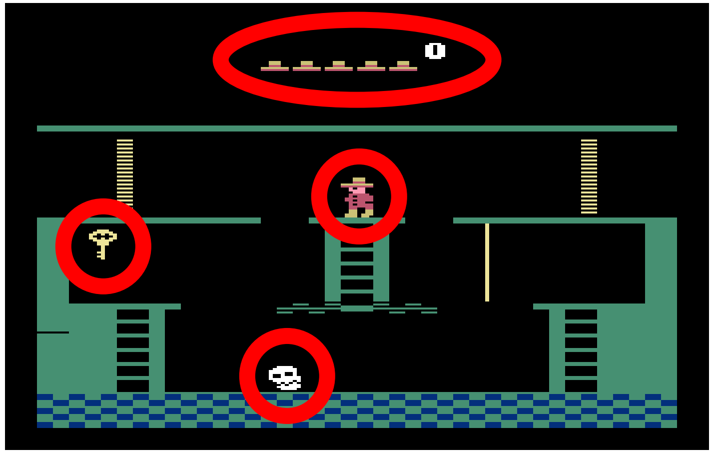
But wait… what's a count?



The trouble with counts

$$\text{use } r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

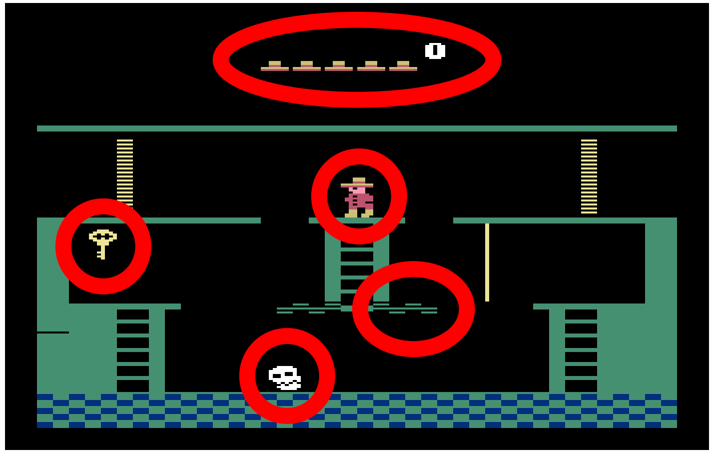
But wait… what's a count?



The trouble with counts

use $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

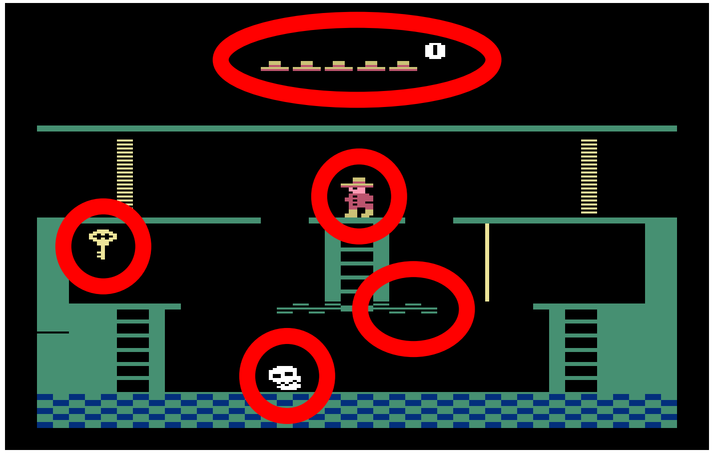
But wait… what's a count?

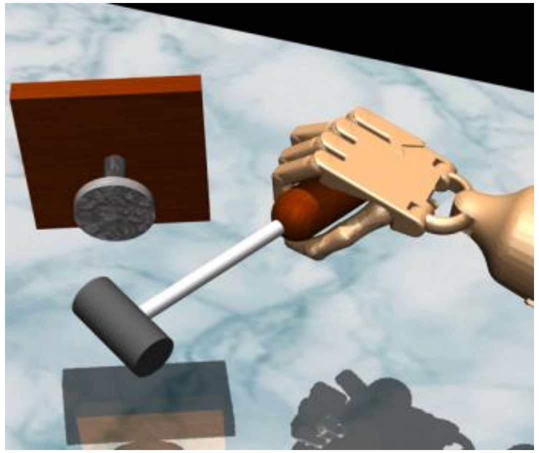


The trouble with counts

$$\text{use } r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$$

But wait… what's a count?

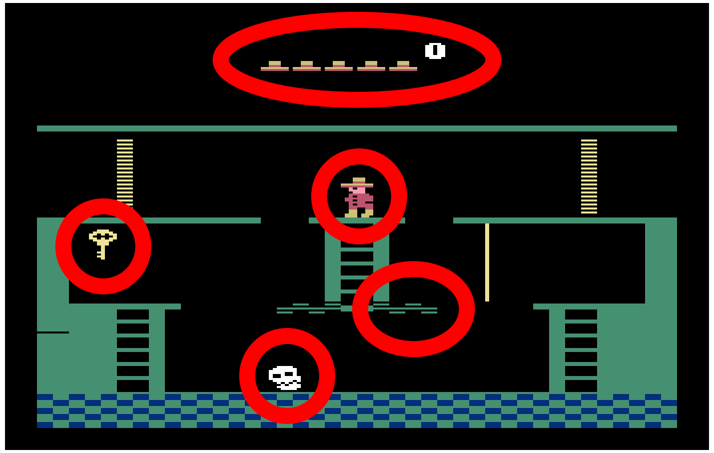


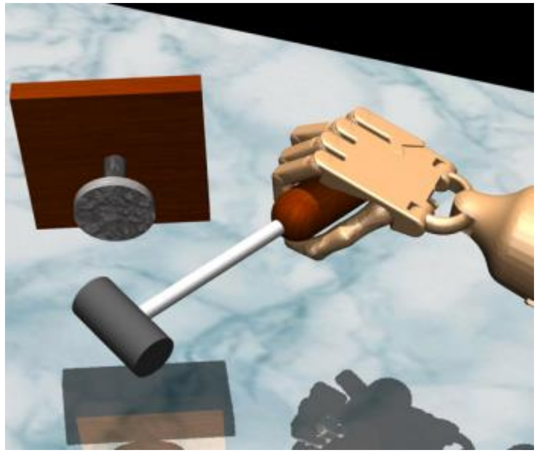


The trouble with counts

use $r^+(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

But wait… what's a count?



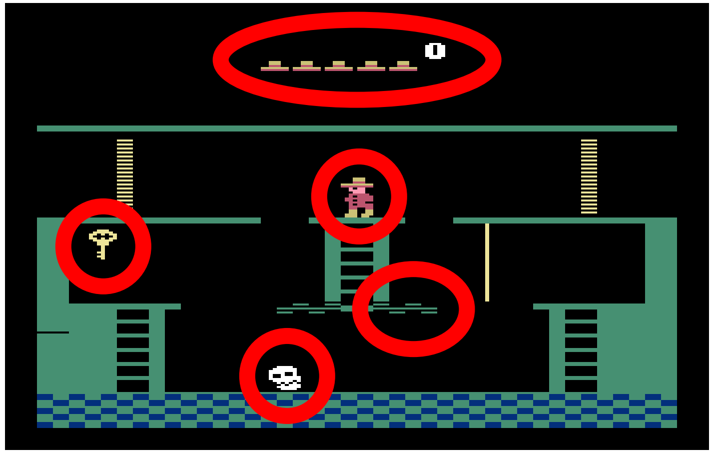


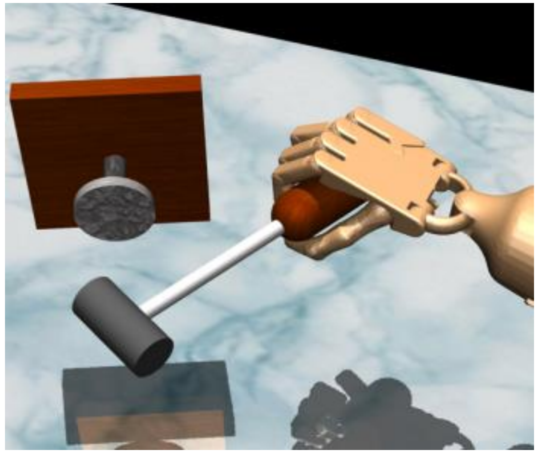
Uh oh… we never see the same thing twice!

The trouble with counts

use $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$

But wait… what's a count?

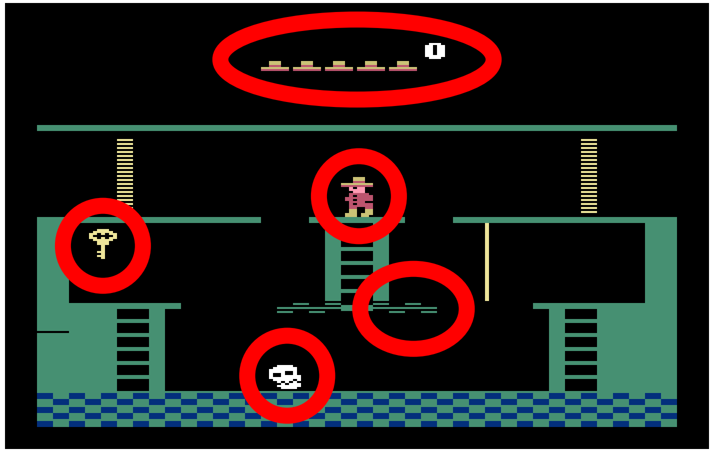


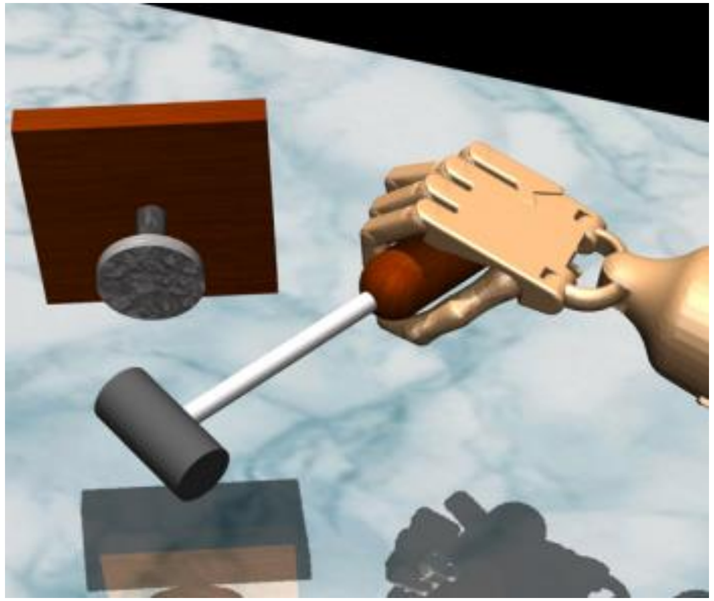


Uh oh… we never see the same thing twice!

But some states are more similar than others

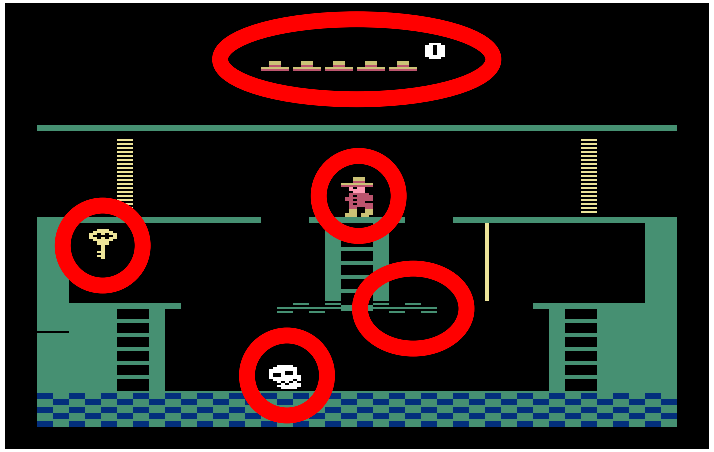
Fitting generative models

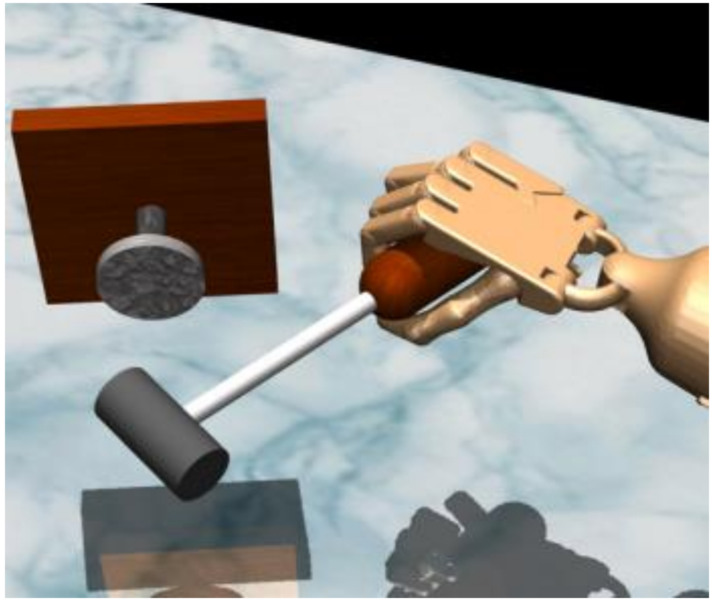




idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

Fitting generative models



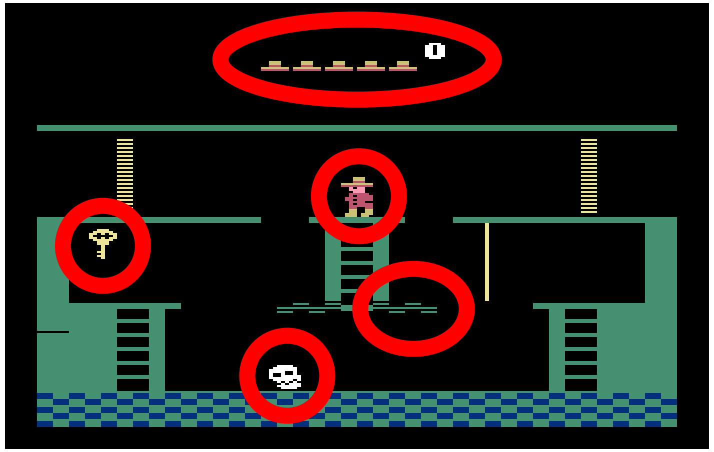


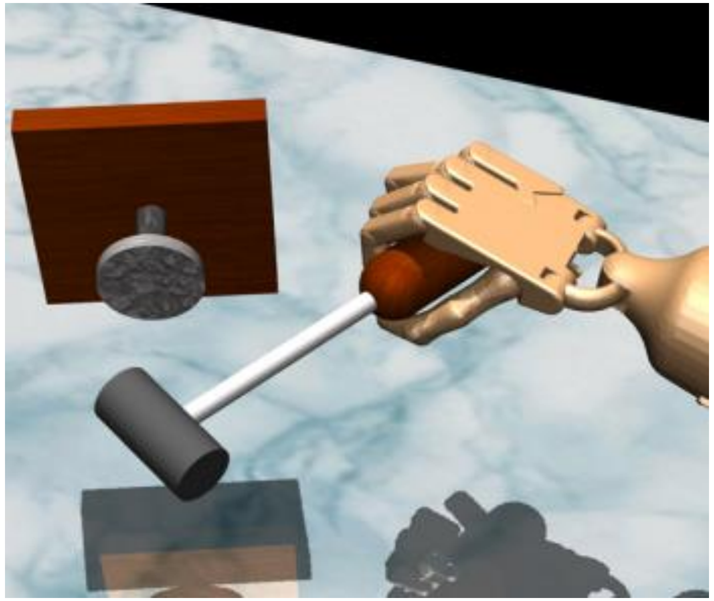
idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$

if $\mathbf{s}$ is similar to previously seen states

Fitting generative models





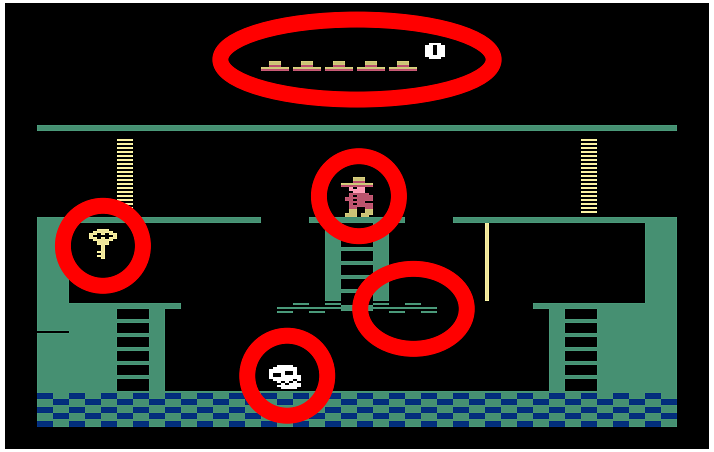
idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

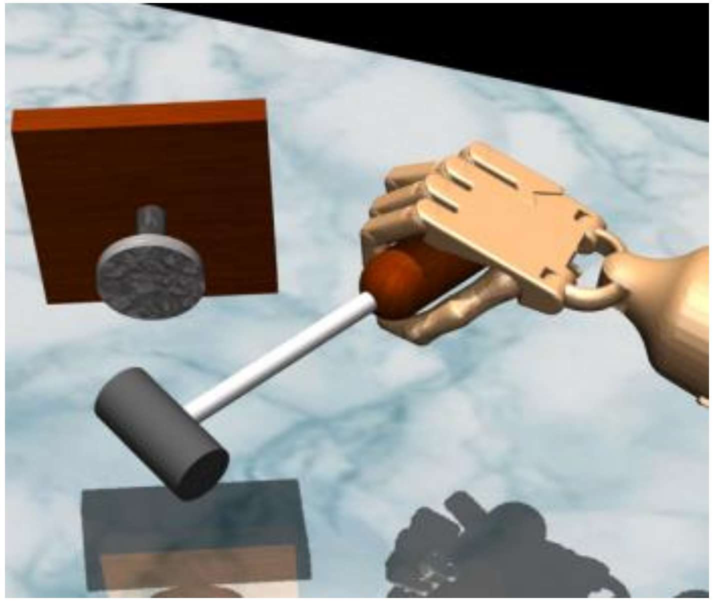
$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$

if $\mathbf{s}$ is similar to previously seen states

can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?

Fitting generative models





idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

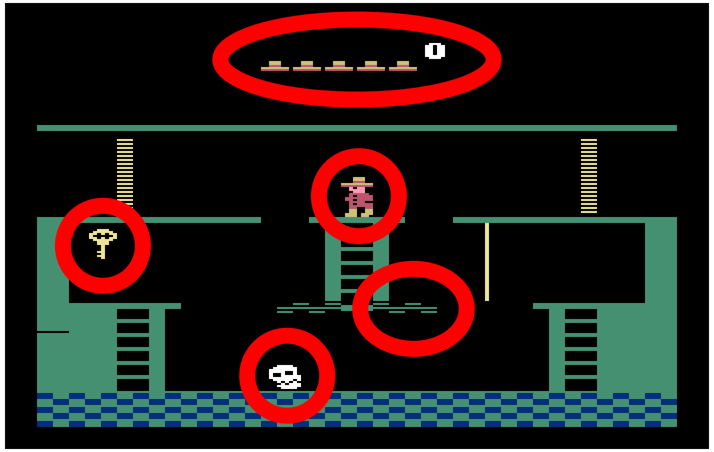
$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$
if $\mathbf{s}$ is similar to previously seen states

can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?

if we have small MDPs
the true probability is:

$$P(\mathbf{s}) = \frac{N(\mathbf{s})}{n}$$

Fitting generative models

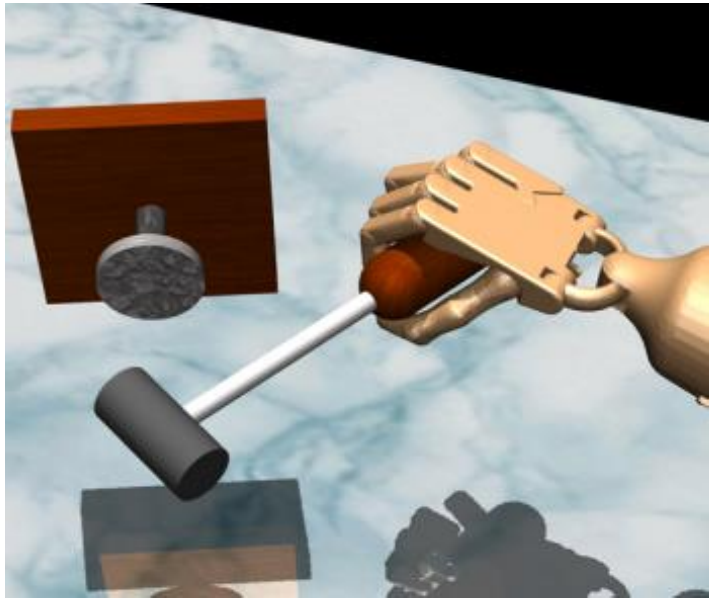


idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$

if $\mathbf{s}$ is similar to previously seen states

can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?



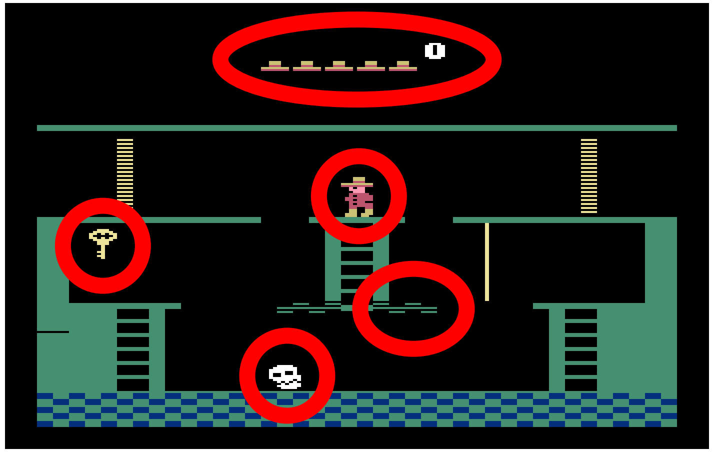
if we have small MDPs

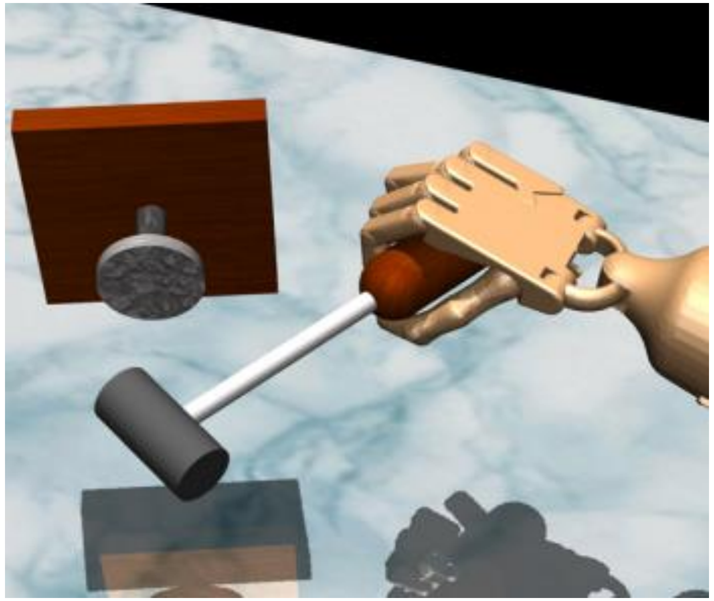
the true probability is:

$$P(\mathbf{s}) = \frac{N(\mathbf{s})}{n}$$

probability/density

Fitting generative models





idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$

if $\mathbf{s}$ is similar to previously seen states

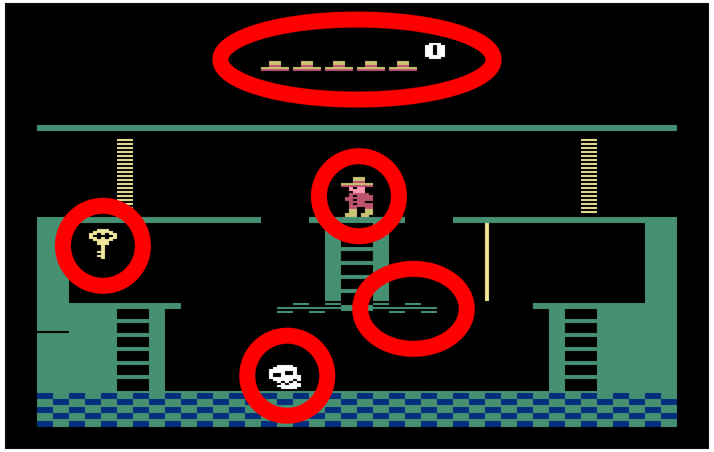
can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?

if we have small MDPs

the true probability is:

$$\underbrace{P(\mathbf{s})}_{\text{probability/density}} = \frac{\overbrace{N(\mathbf{s})}^{\text{count}}}{n}$$

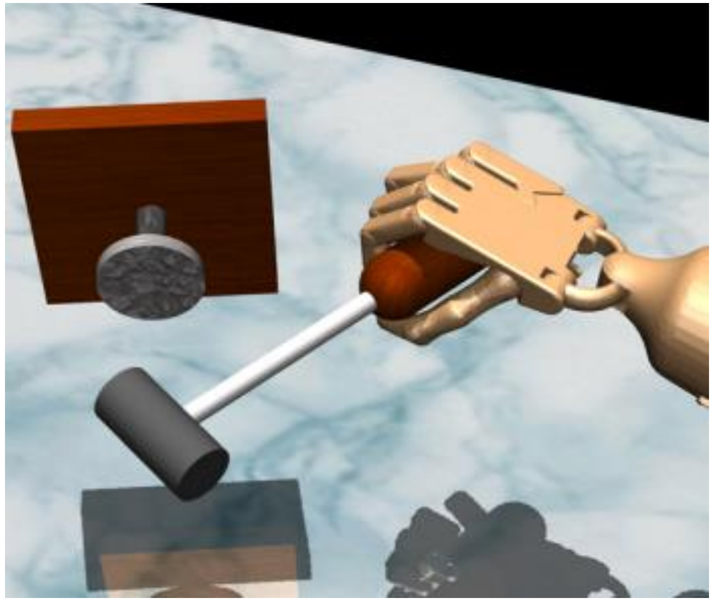
Fitting generative models



idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$
if $\mathbf{s}$ is similar to previously seen states

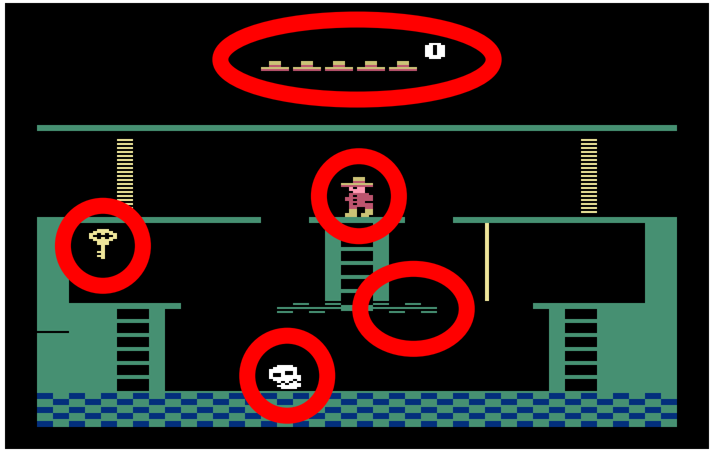
can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?



if we have small MDPs
the true probability is:

$$\underbrace{P(\mathbf{s})}_{\text{probability/density}} = \frac{\overbrace{N(\mathbf{s})}^{\text{count}}}{\underbrace{n}_{\text{total states visited}}}$$

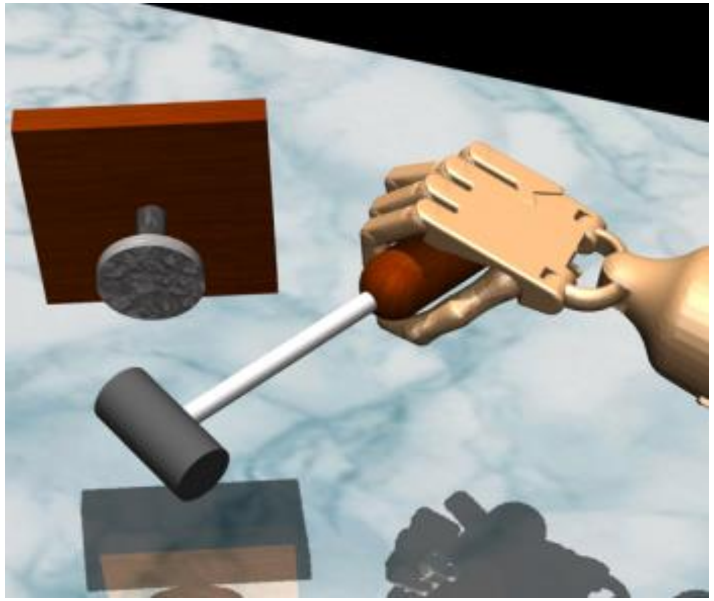
Fitting generative models



idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$
if $\mathbf{s}$ is similar to previously seen states

can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?



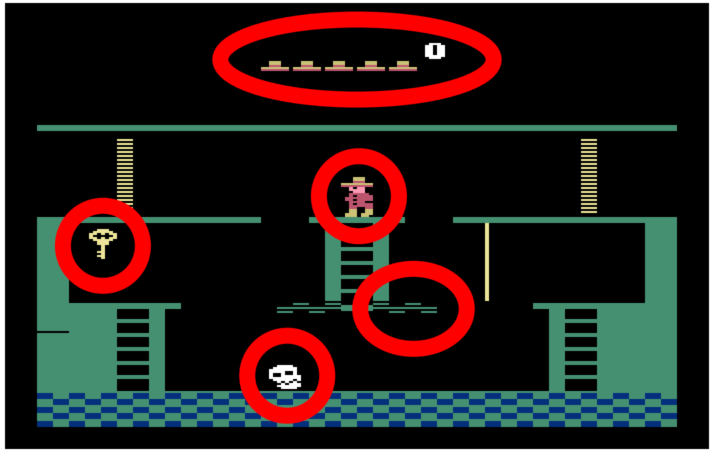
if we have small MDPs
the true probability is:

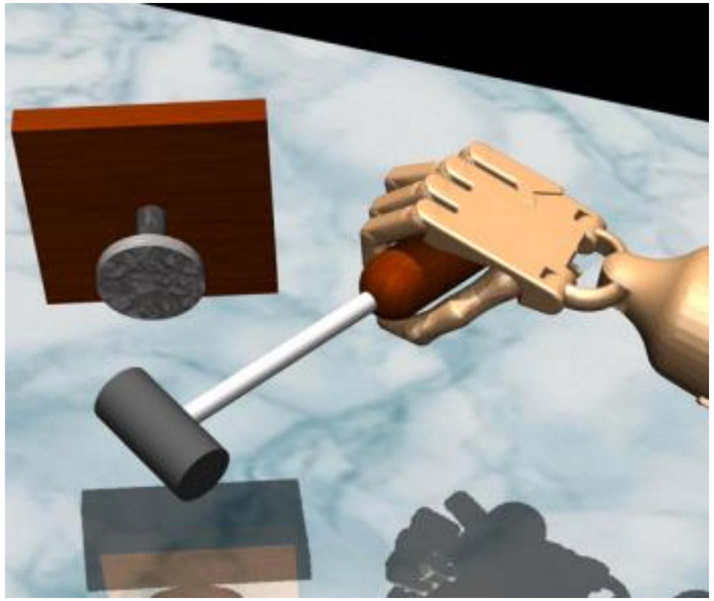
$$\underbrace{P(\mathbf{s})}_{\text{probability/density}} = \frac{\overbrace{N(\mathbf{s})}^{\text{count}}}{\underbrace{n}_{\text{total states visited}}}$$

after we see $\mathbf{s}$, we have:

$$P'(\mathbf{s}) = \frac{N(\mathbf{s}) + 1}{n + 1}$$

Fitting generative models





idea: fit a density model $p_\theta(\mathbf{s})$ (or $p_\theta(\mathbf{s}, \mathbf{a})$)

$p_\theta(\mathbf{s})$ might be high even for a new $\mathbf{s}$

if $\mathbf{s}$ is similar to previously seen states

can we use $p_\theta(\mathbf{s})$ to get a "pseudo-count"?

if we have small MDPs

the true probability is:

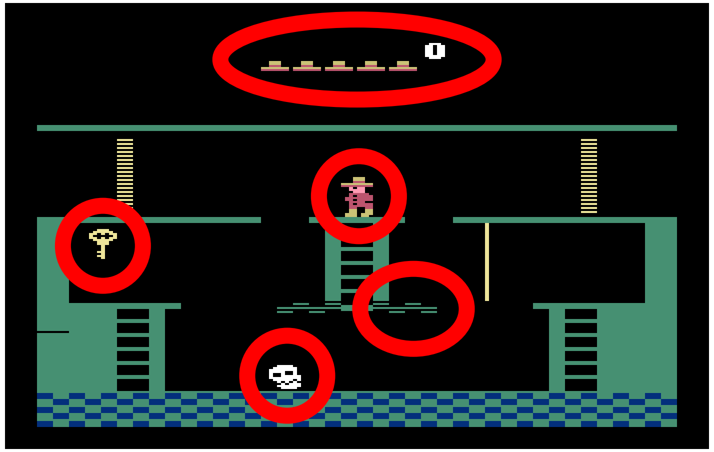
$$\underbrace{P(\mathbf{s})}_{\text{probability/density}} = \frac{\overbrace{N(\mathbf{s})}^{\text{count}}}{\underbrace{n}_{\text{total states visited}}}$$

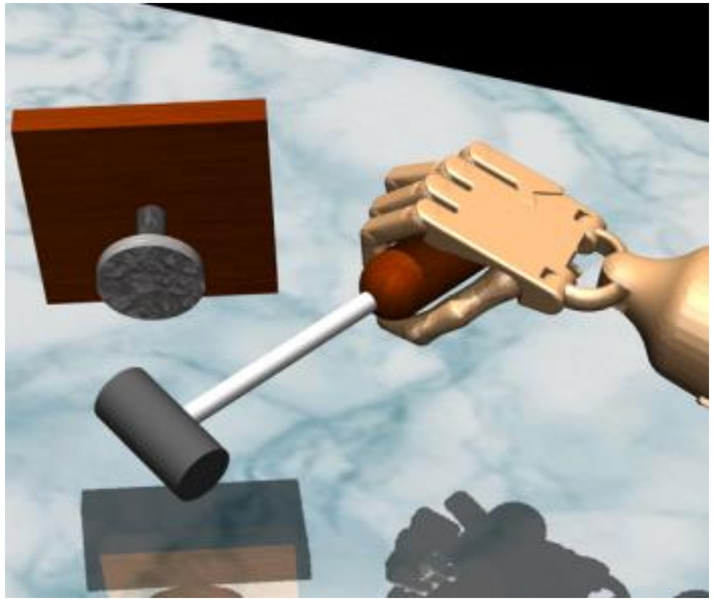
after we see $\mathbf{s}$, we have:

$$P'(\mathbf{s}) = \frac{N(\mathbf{s}) + 1}{n + 1}$$

can we get $p_\theta(\mathbf{s})$ and $p_{\theta'}(\mathbf{s})$ to obey these equations?

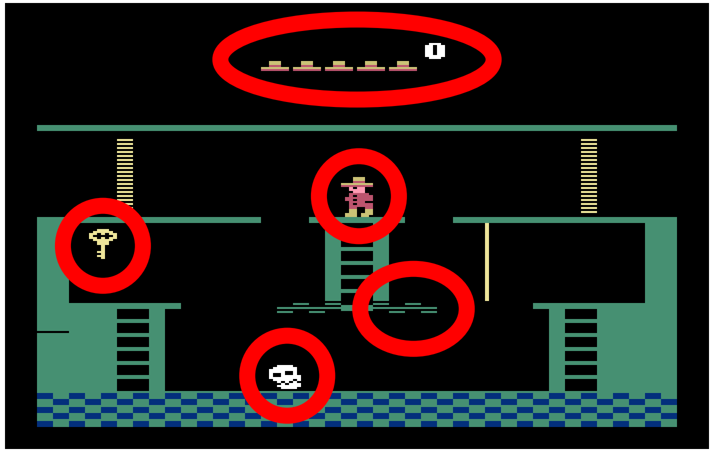
Exploring with pseudo-counts

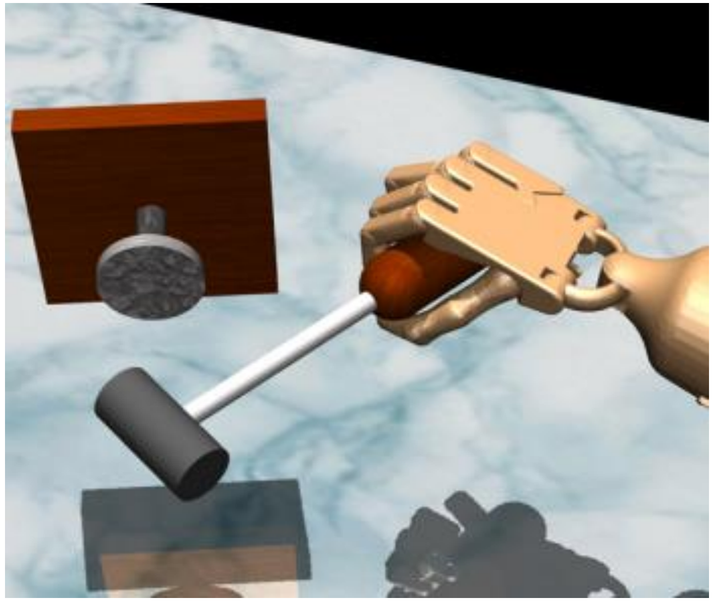




Bellemare et al. “Unifying Count-Based Exploration…”

Exploring with pseudo-counts

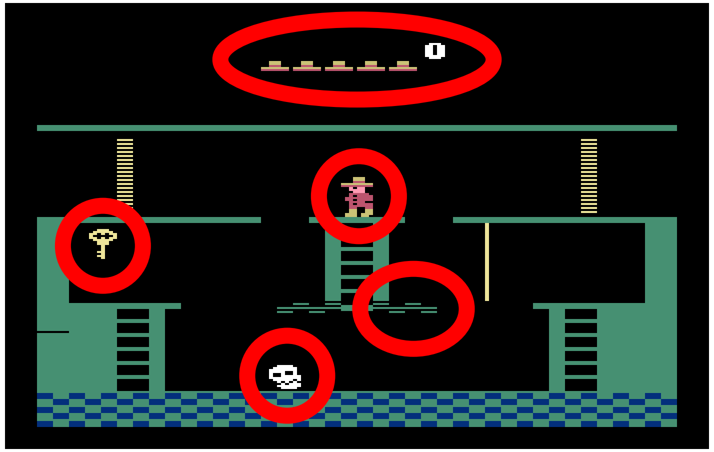


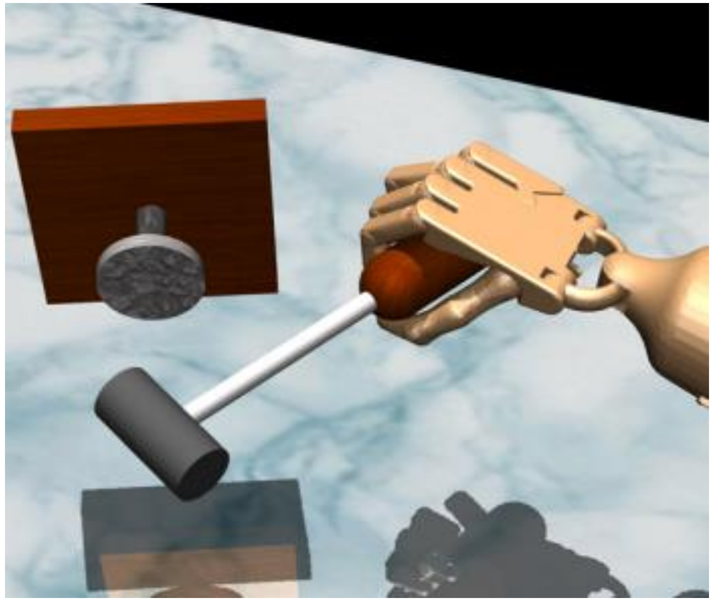


fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts



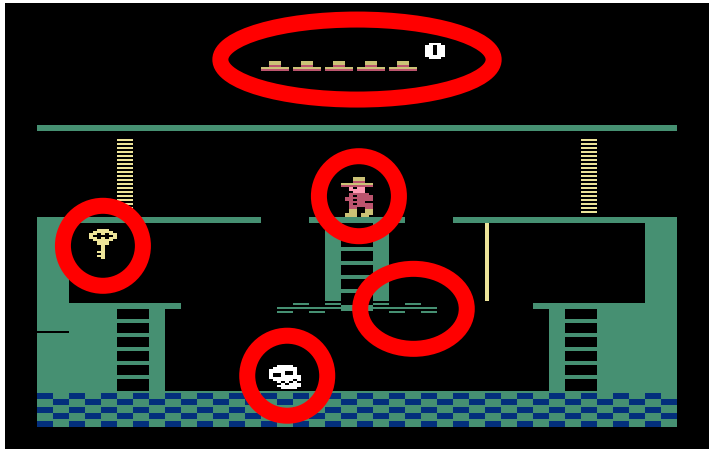


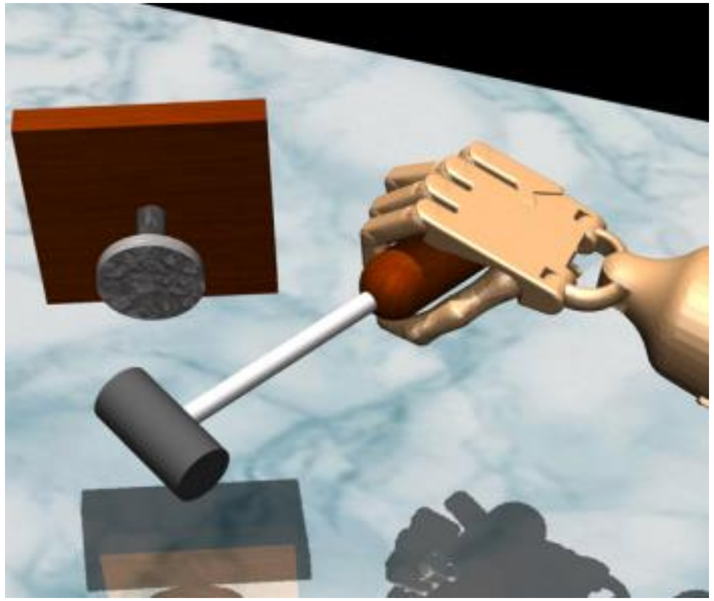
fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

Bellemare et al. “Unifying Count-Based Exploration…”

Exploring with pseudo-counts





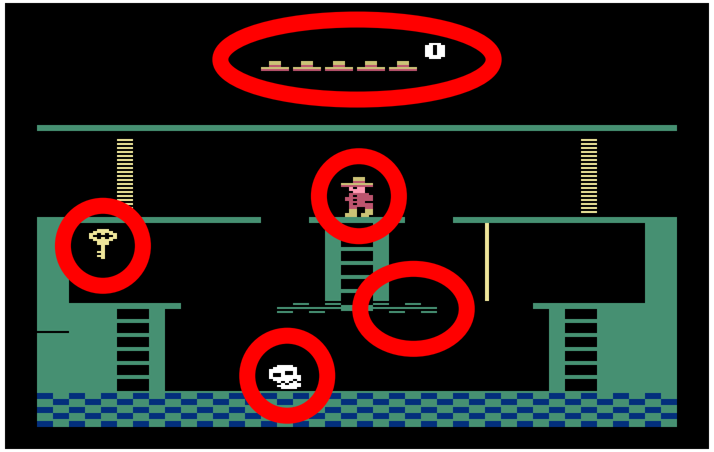
fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

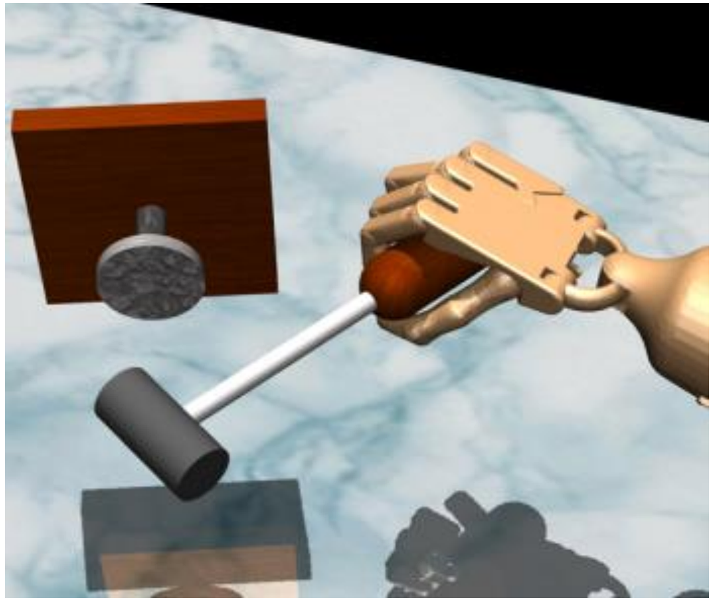
take a step i and observe $\mathbf{s}_i$

fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

Bellemare et al. “Unifying Count-Based Exploration…”

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

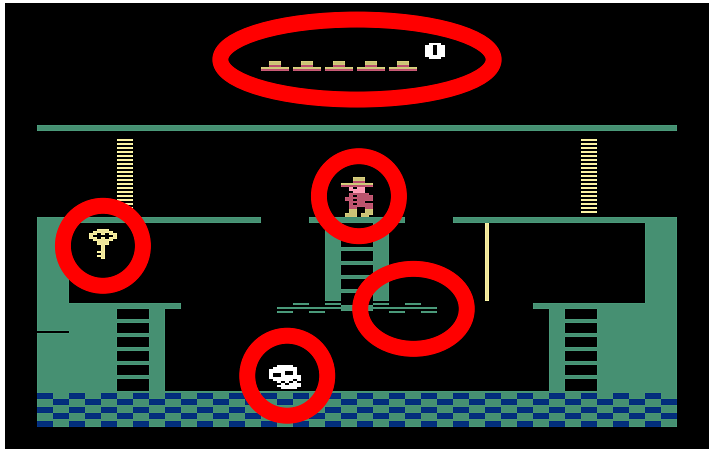
take a step i and observe $\mathbf{s}_i$

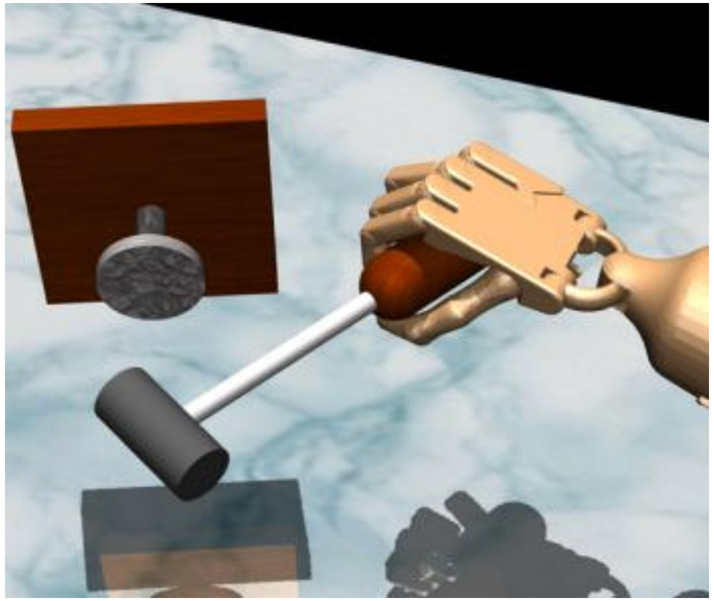
fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

Bellemare et al. “Unifying Count-Based Exploration…”

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

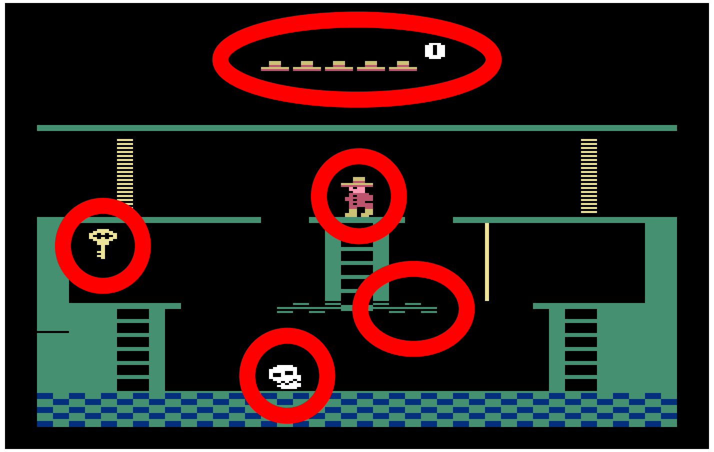
fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

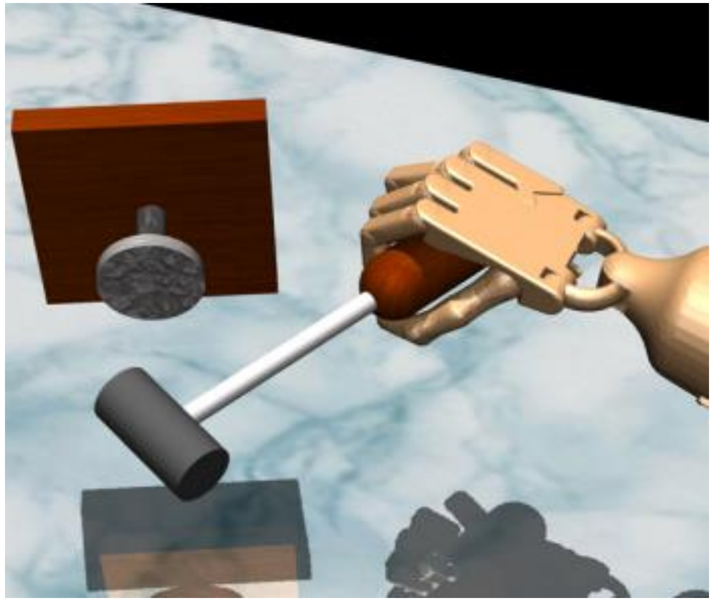
use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

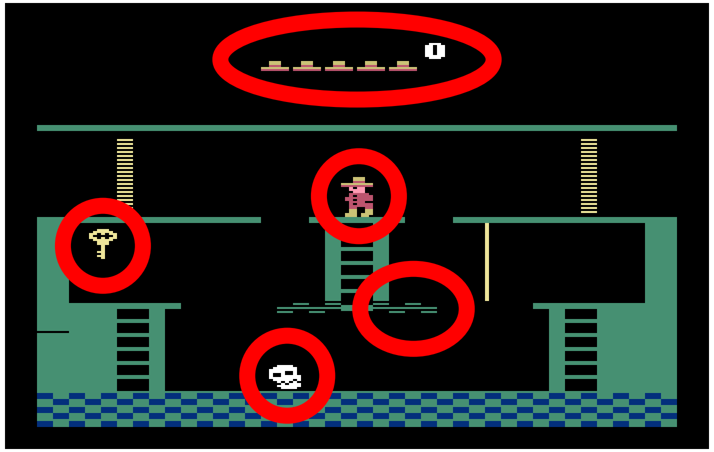
fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

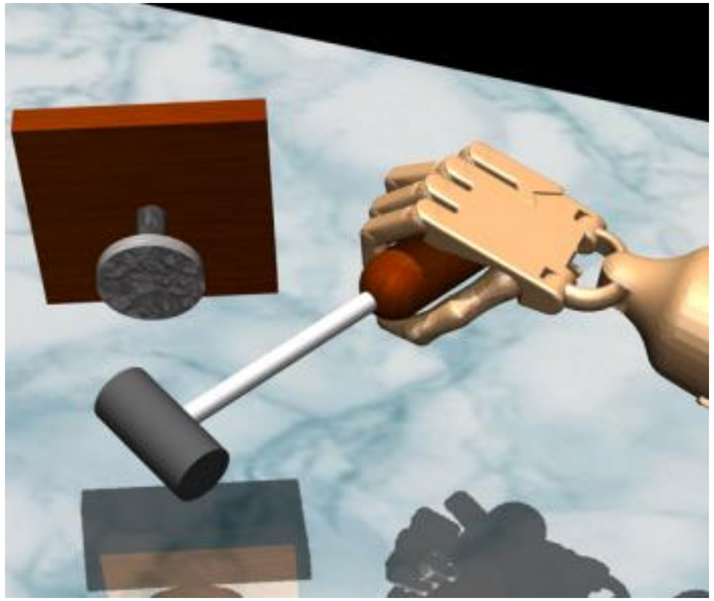
use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ ⟵ "pseudo-count"

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

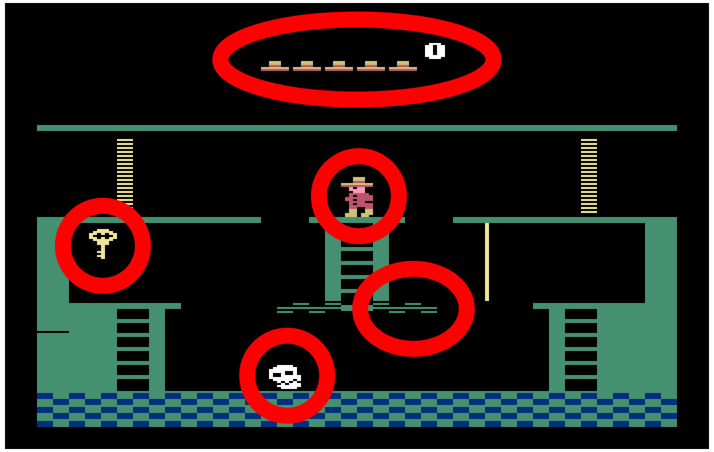
fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

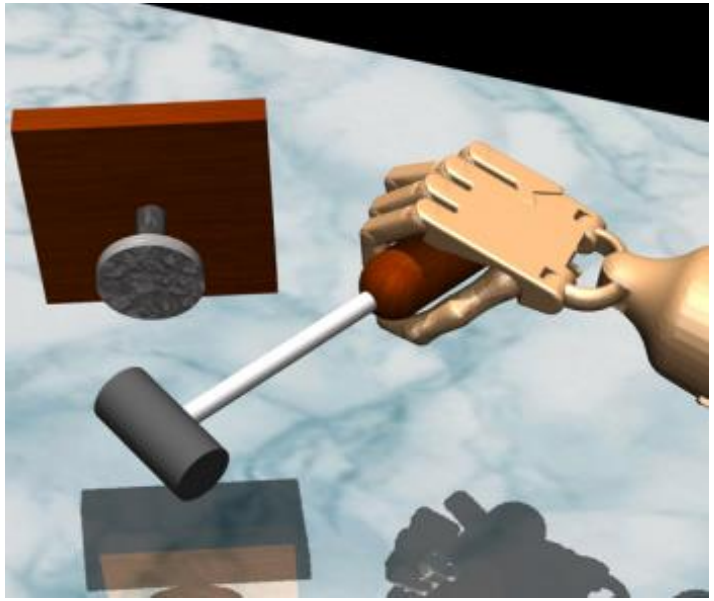
use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ "pseudo-count"

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

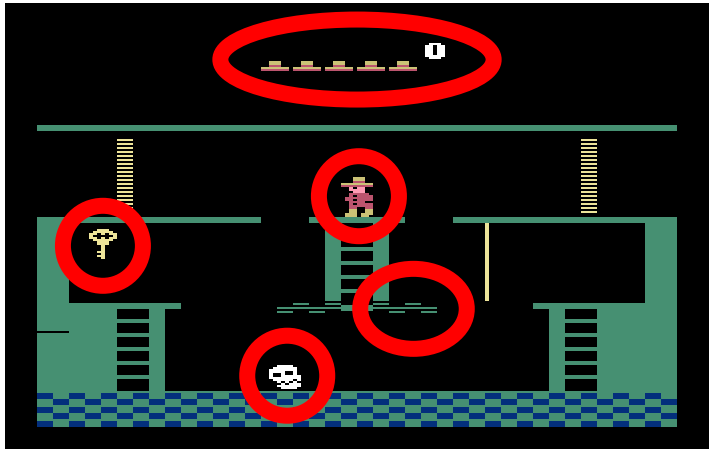
use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

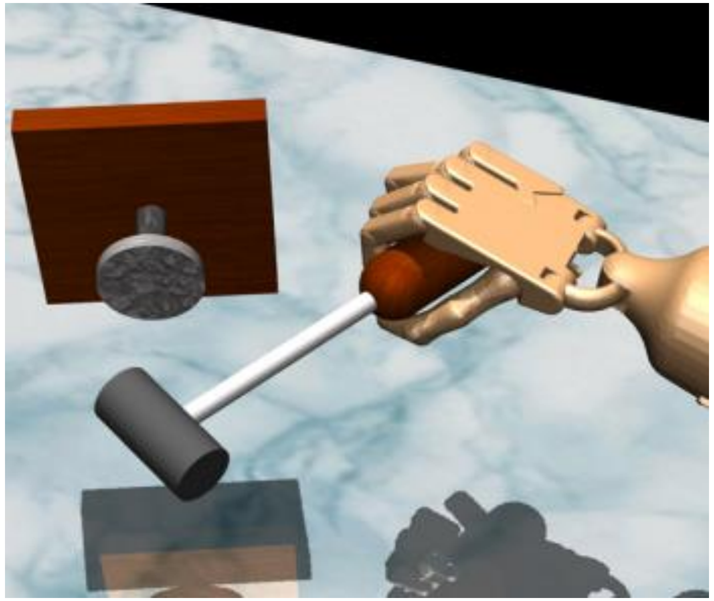
set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ "pseudo-count"

how to get $\hat{N}(\mathbf{s})$? use the equations

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

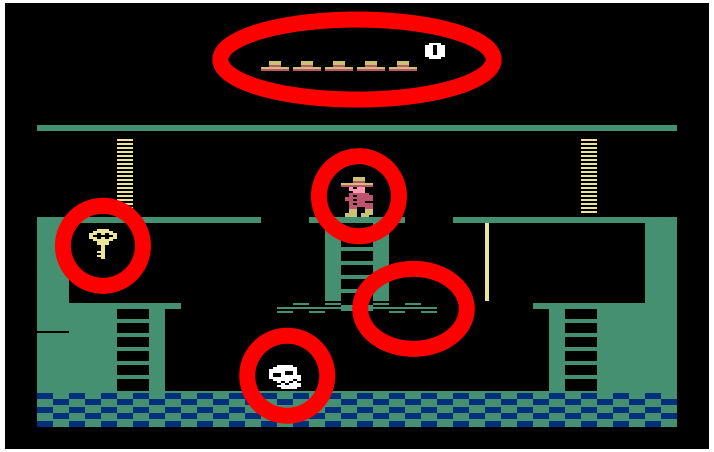
set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ ← "pseudo-count"

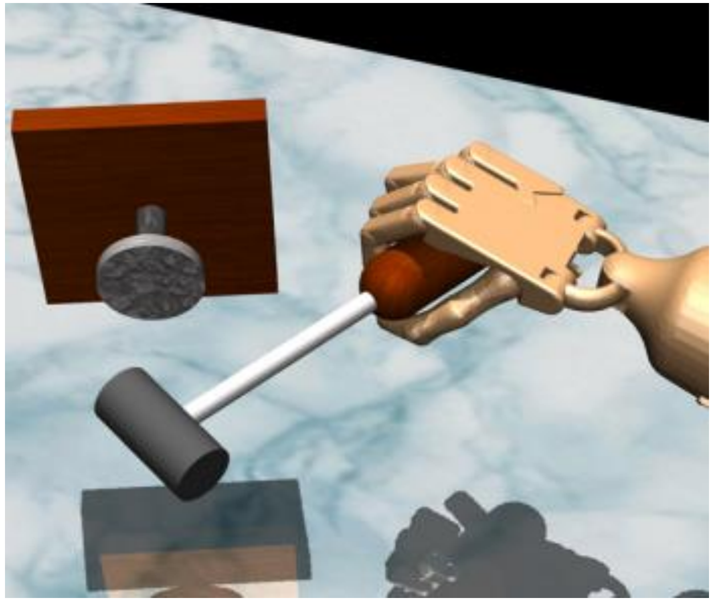
how to get $\hat{N}(\mathbf{s})$? use the equations

$$p_\theta(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i)}{\hat{n}} \qquad p_{\theta'}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i) + 1}{\hat{n} + 1}$$

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ "pseudo-count"

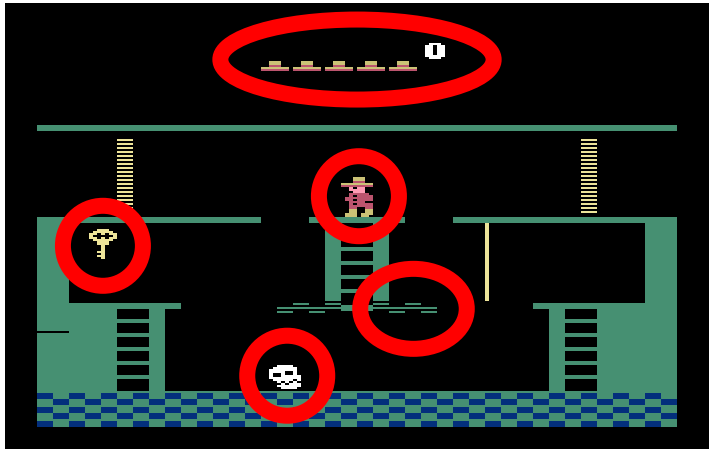
how to get $\hat{N}(\mathbf{s})$? use the equations

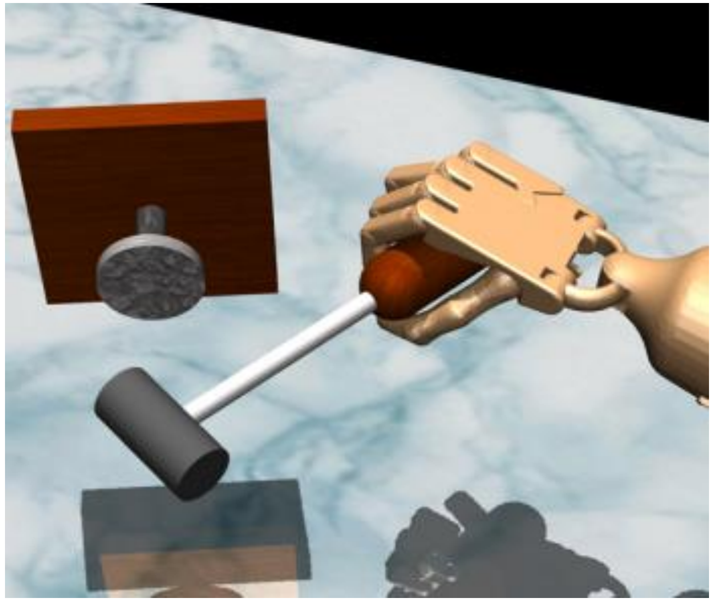
$$p_\theta(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i)}{\hat{n}} \qquad p_{\theta'}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i) + 1}{\hat{n} + 1}$$

two equations and two unknowns!

Bellemare et al. "Unifying Count-Based Exploration…"

Exploring with pseudo-counts





fit model $p_\theta(\mathbf{s})$ to all states $\mathcal{D}$ seen so far

take a step i and observe $\mathbf{s}_i$

fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$

use $p_\theta(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$

set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ ← "pseudo-count"

how to get $\hat{N}(\mathbf{s})$? use the equations

$$p_\theta(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i)}{\hat{n}} \qquad\qquad p_{\theta'}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i) + 1}{\hat{n} + 1}$$

two equations and two unknowns!

$$\hat{N}(\mathbf{s}_i) = \hat{n} p_\theta(\mathbf{s}_i) \qquad\qquad \hat{n} = \frac{1 - p_{\theta'}(\mathbf{s}_i)}{p_{\theta'}(\mathbf{s}_i) - p_\theta(\mathbf{s}_i)} p_\theta(\mathbf{s}_i)$$

Bellemare et al. "Unifying Count-Based Exploration…"

What kind of bonus to use?

What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

UCB:

$$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2 \ln n}{N(\mathbf{s})}}$$

What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

UCB: $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2\ln n}{N(\mathbf{s})}}$$

MBIE-EB (Strehl & Littman, 2008): $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{1}{N(\mathbf{s})}}$$

What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

UCB: $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2\ln n}{N(\mathbf{s})}}$$

MBIE-EB (Strehl & Littman, 2008): $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{1}{N(\mathbf{s})}}$$

BEB (Kolter & Ng, 2009): $$\mathcal{B}(N(\mathbf{s})) = \frac{1}{N(\mathbf{s})}$$

What kind of bonus to use?

Lots of functions in the literature, inspired by optimal methods for bandits or small MDPs

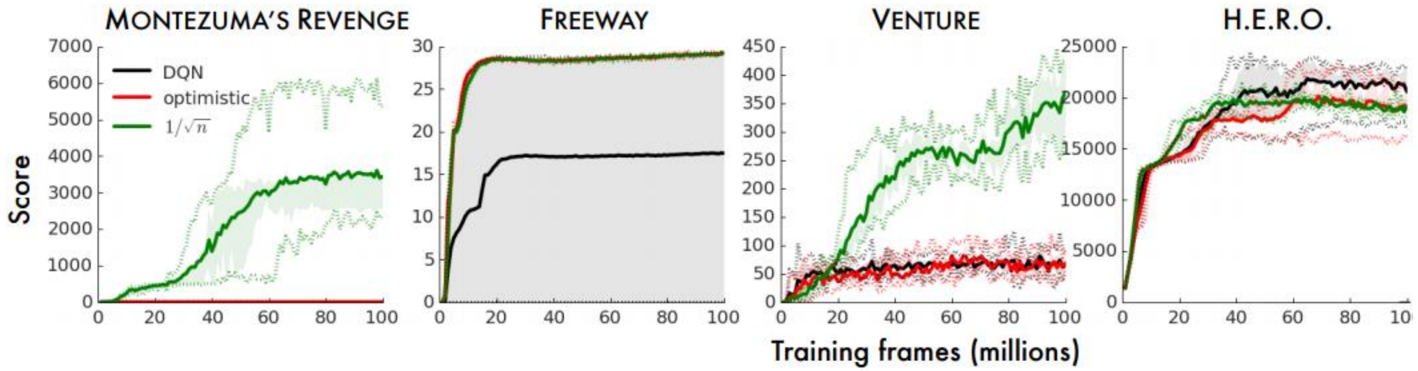
UCB: $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2 \ln n}{N(\mathbf{s})}}$$

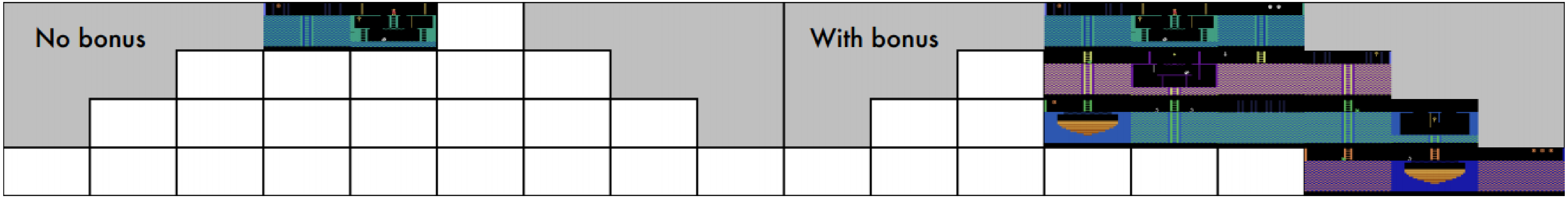
MBIE-EB (Strehl & Littman, 2008): $$\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{1}{N(\mathbf{s})}}$$

this is the one used by Bellemare et al. '16

BEB (Kolter & Ng, 2009): $$\mathcal{B}(N(\mathbf{s})) = \frac{1}{N(\mathbf{s})}$$

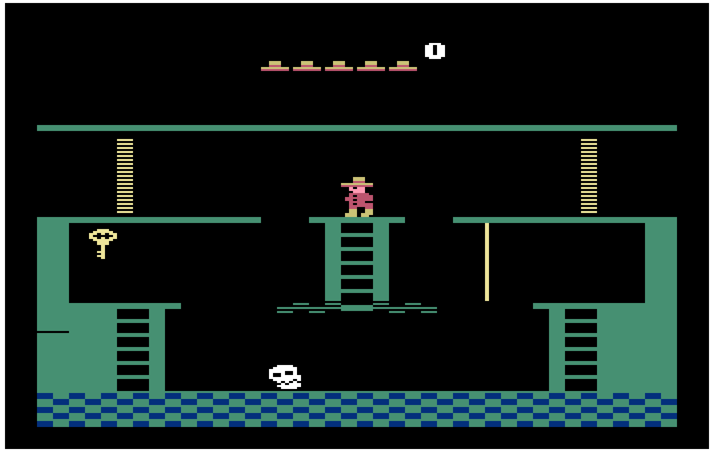
Does it work?

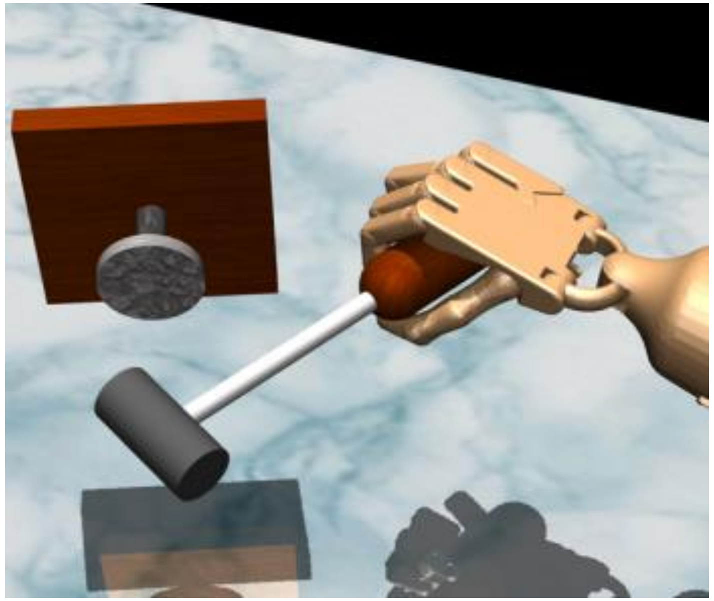




Bellemare et al. "Unifying Count-Based Exploration…"

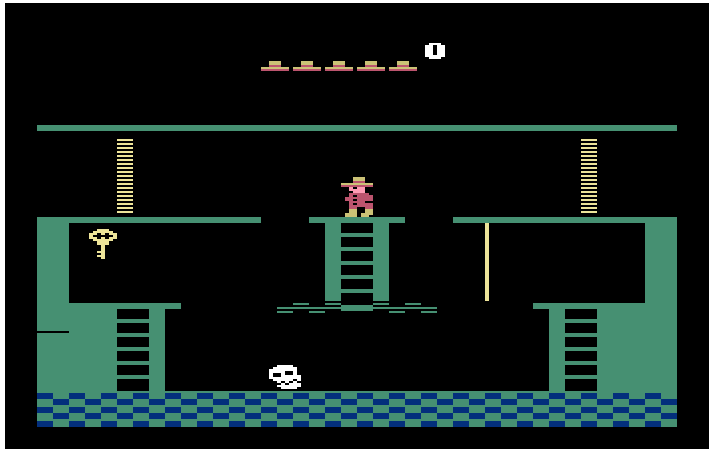
What kind of model to use?

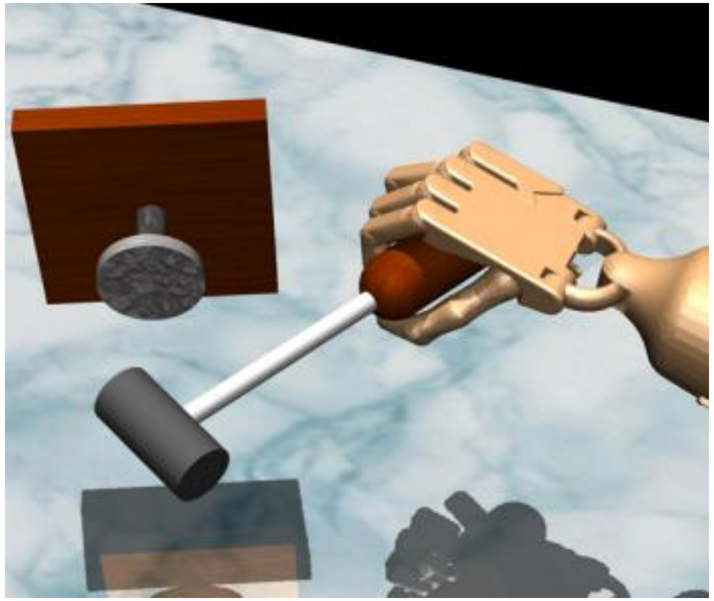




$$p_\theta(\mathbf{s})$$

What kind of model to use?

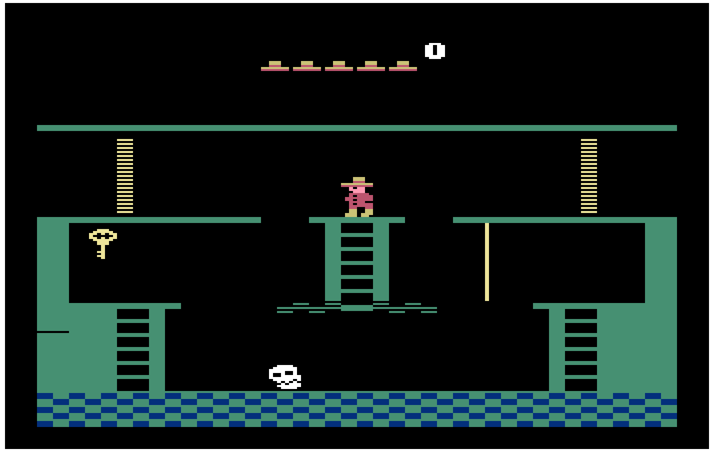


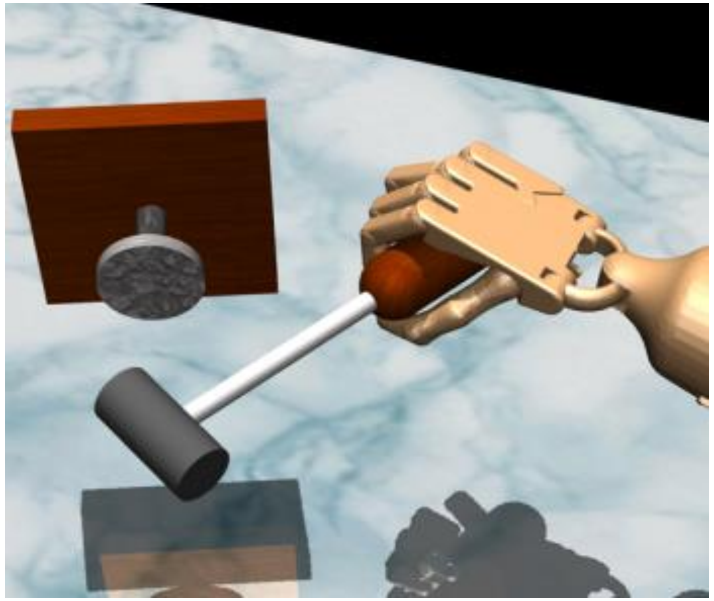


$$p_\theta(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

What kind of model to use?



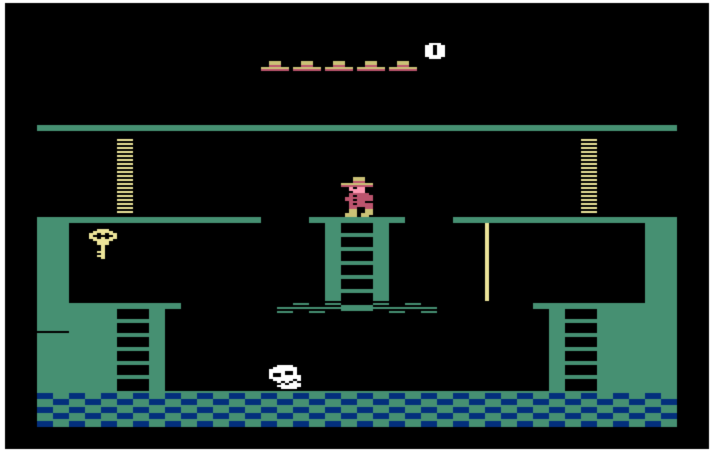


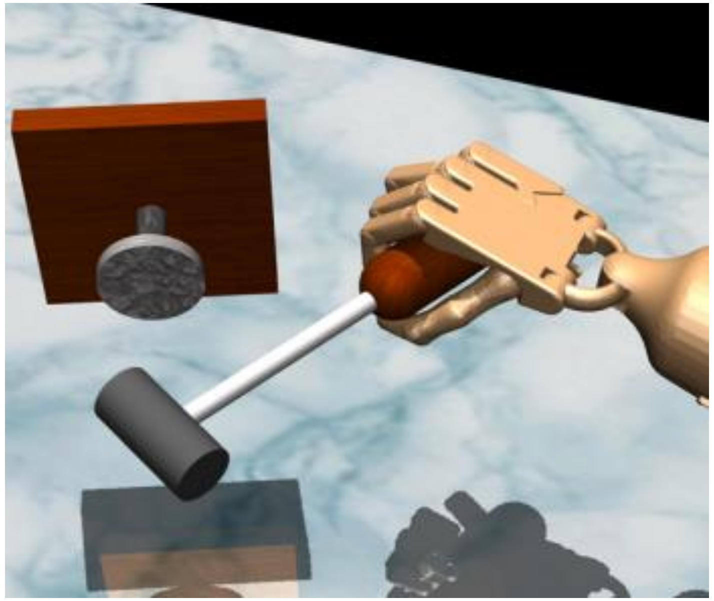
$$p_\theta(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

opposite considerations from many popular generative models in the literature (e.g., GANs)

What kind of model to use?





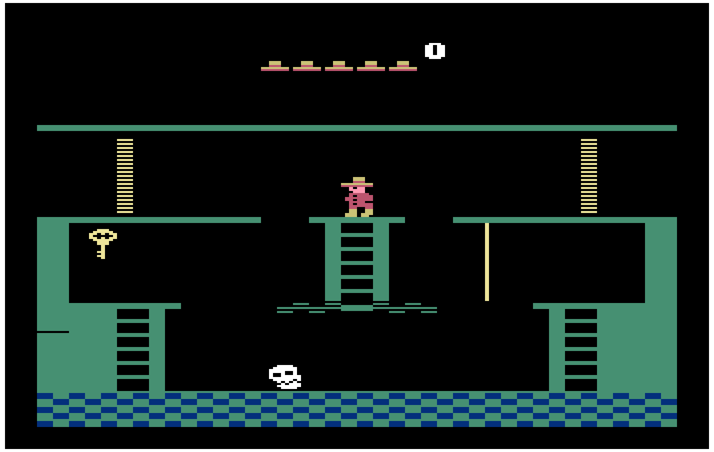
$$p_\theta(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

opposite considerations from many popular generative models in the literature (e.g., GANs)

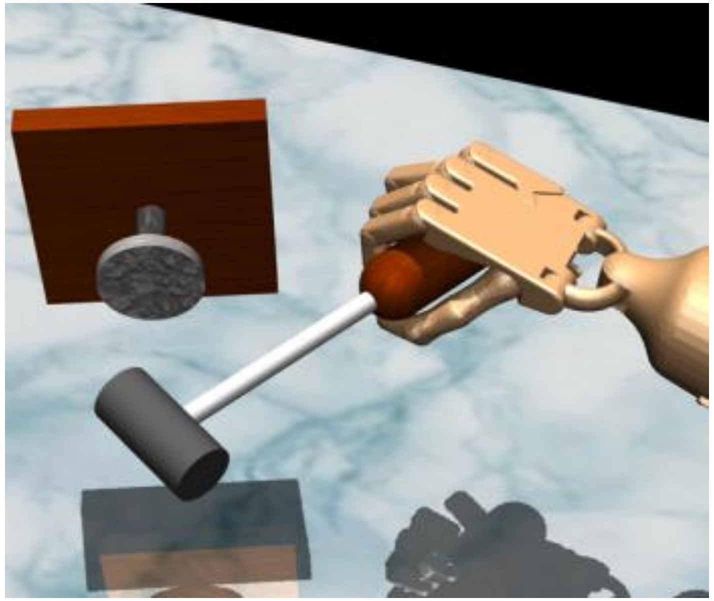
Bellemare et al.: "CTS" model: condition each pixel on its top-left neighborhood

What kind of model to use?



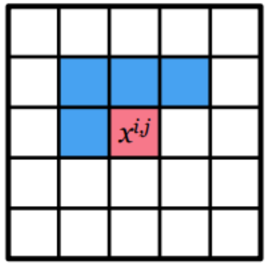
$$p_\theta(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

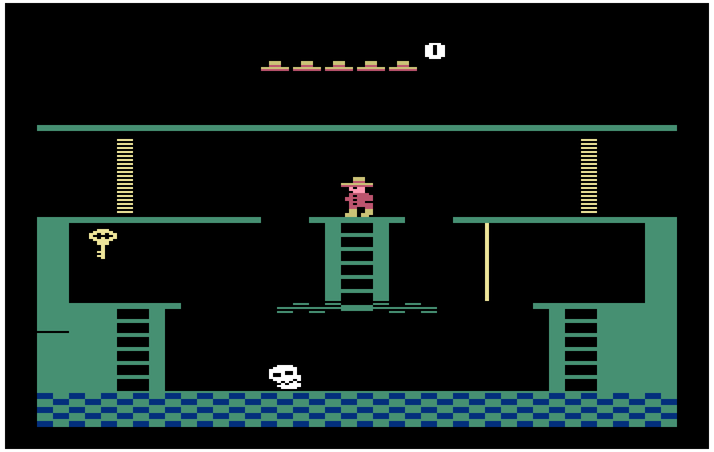


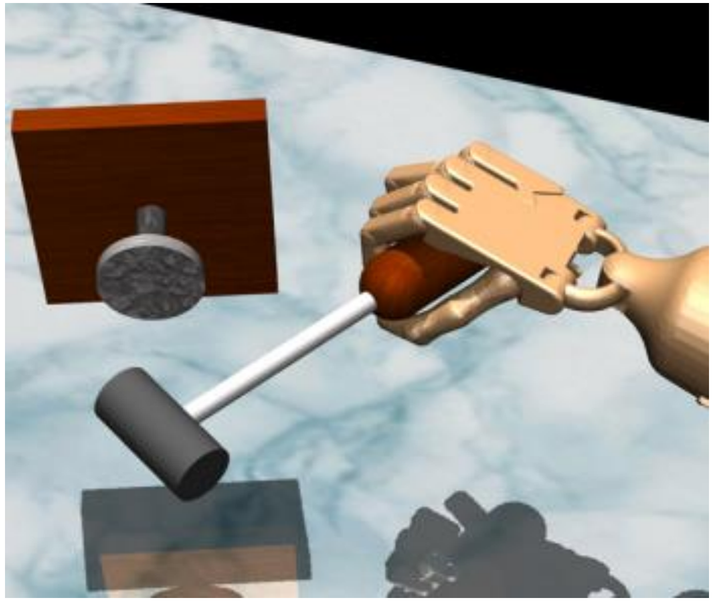
opposite considerations from many popular generative models in the literature (e.g., GANs)

Bellemare et al.: "CTS" model: condition each pixel on its top-left neighborhood



What kind of model to use?



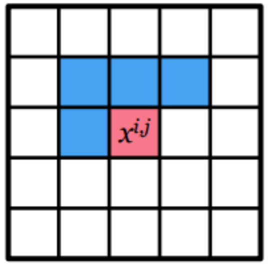


$$p_\theta(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

opposite considerations from many popular generative models in the literature (e.g., GANs)

Bellemare et al.: "CTS" model: condition each pixel on its top-left neighborhood



Other models: stochastic neural networks, compression length, EX2

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$$

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

Osband et al. "Deep Exploration via Bootstrapped DQN"

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

Osband et al. “Deep Exploration via Bootstrapped DQN”

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

1. sample Q-function Q from $p(Q)$

Osband et al. "Deep Exploration via Bootstrapped DQN"

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

1. sample Q-function Q from $p(Q)$
2. act according to Q for one episode

Osband et al. "Deep Exploration via Bootstrapped DQN"

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

1. sample Q-function Q from $p(Q)$
2. act according to Q for one episode
3. update $p(Q)$

Osband et al. “Deep Exploration via Bootstrapped DQN”

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

1. sample Q-function Q from $p(Q)$
2. act according to Q for one episode
3. update $p(Q)$

Osband et al. "Deep Exploration via Bootstrapped DQN"

Posterior sampling in deep RL

Represent explicitly our uncertainty in the model parameters θ

Then sample from it (Thompson sampling):

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$

How do we represent the distribution?

$$a = \arg\max_a E_{\theta_a}[r(a)]$$

A simple and very general approach is to compute an ensemble of models, and then sample from it:

1. sample Q-function Q from $p(Q)$
2. act according to Q for one episode
3. update $p(Q)$

since Q-learning is off-policy, we don't care which Q-function was used to collect data

Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

Osband et al. “Deep Exploration via Bootstrapped DQN”

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \ldots, \mathcal{D}_N$

Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model f_{θ_i} on $\mathcal{D}_i$

Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model f_{θ_i} on $\mathcal{D}_i$

to sample from $p(\theta)$, sample $i \in [1, \dots, N]$ and use f_{θ_i}

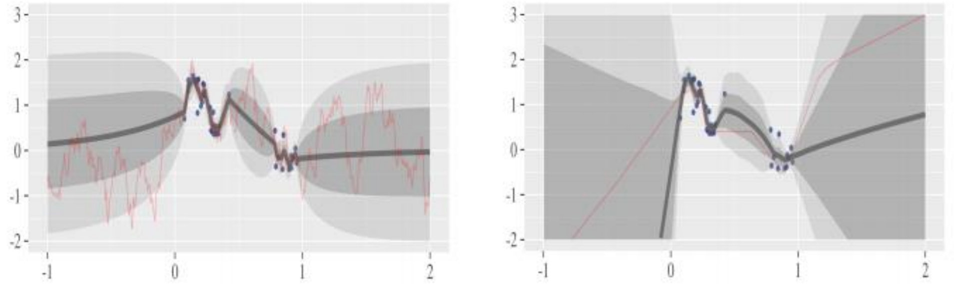
Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model f_{θ_i} on $\mathcal{D}_i$

to sample from $p(\theta)$, sample $i \in [1, \dots, N]$ and use f_{θ_i}



(b) Gaussian process posterior (c) Bootstrapped neural nets

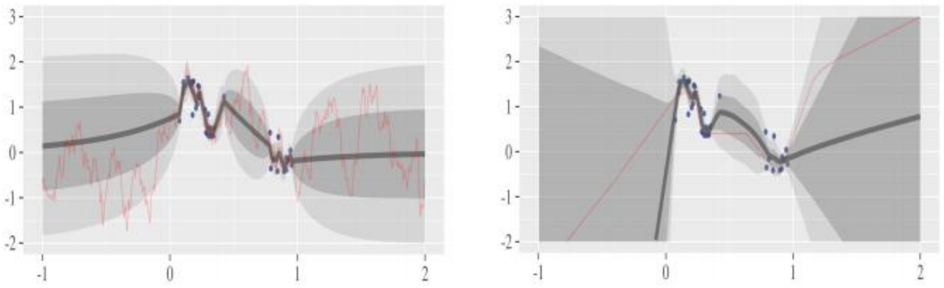
Osband et al. “Deep Exploration via Bootstrapped DQN”

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model f_{θ_i} on $\mathcal{D}_i$

to sample from $p(\theta)$, sample $i \in [1, \dots, N]$ and use f_{θ_i}



(b) Gaussian process posterior (c) Bootstrapped neural nets

training N big neural nets is expensive, can we avoid it?

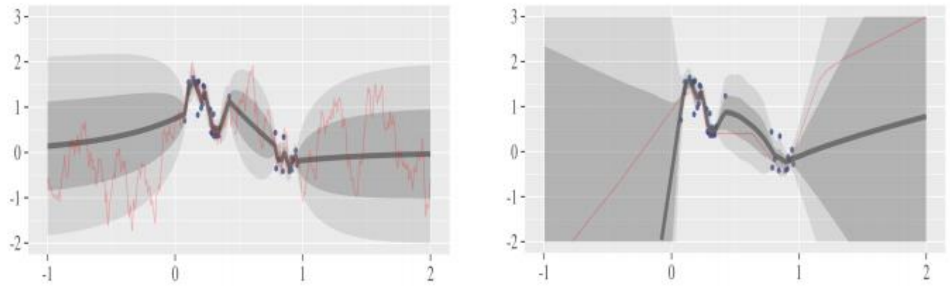
Osband et al. "Deep Exploration via Bootstrapped DQN"

Bootstrap

given a dataset $\mathcal{D}$, resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

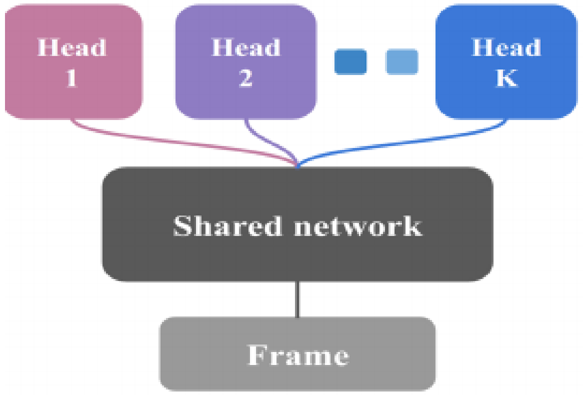
train each model f_{θ_i} on $\mathcal{D}_i$

to sample from $p(\theta)$, sample $i \in [1, \dots, N]$ and use f_{θ_i}



(b) Gaussian process posterior (c) Bootstrapped neural nets

training N big neural nets is expensive, can we avoid it?

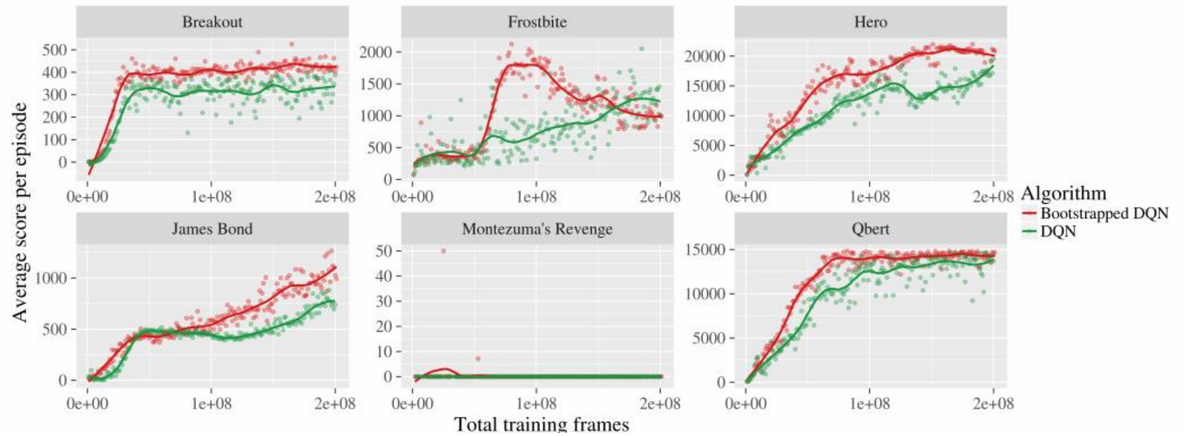


Osband et al. "Deep Exploration via Bootstrapped DQN"

Why does this work?

Exploring with random actions (e.g., epsilon-greedy): random walk pattern, in general $\Omega(N^2)$ steps to visit N states.

Exploring with random Q-functions: commit to a randomized but internally consistent strategy ***for an entire episode***

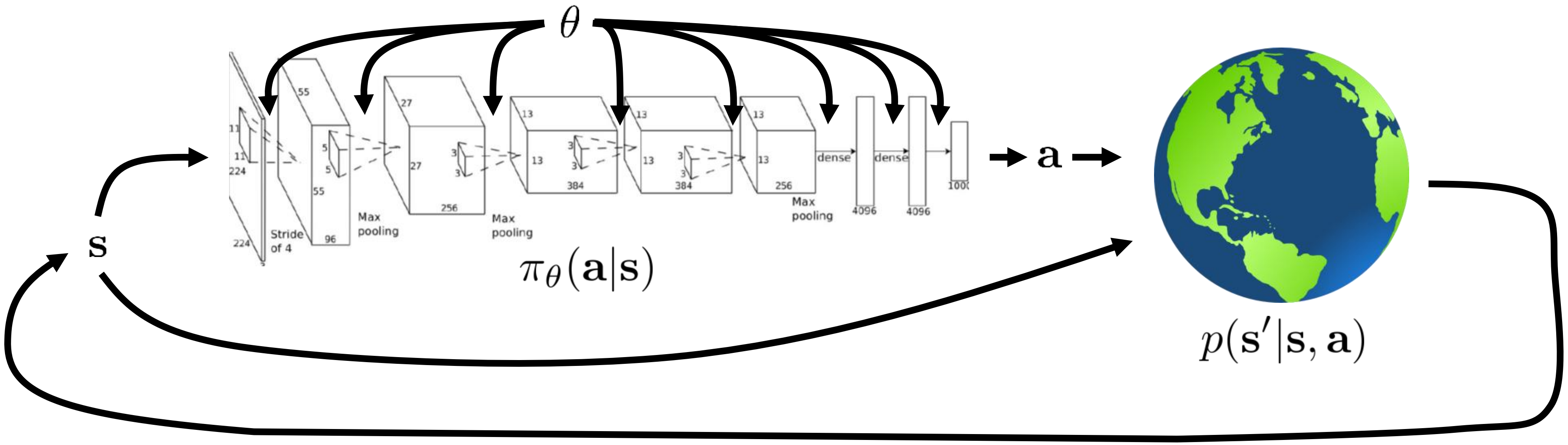


+ no change to original reward function

- very good bonuses often do better

Osband et al. "Deep Exploration via Bootstrapped DQN"

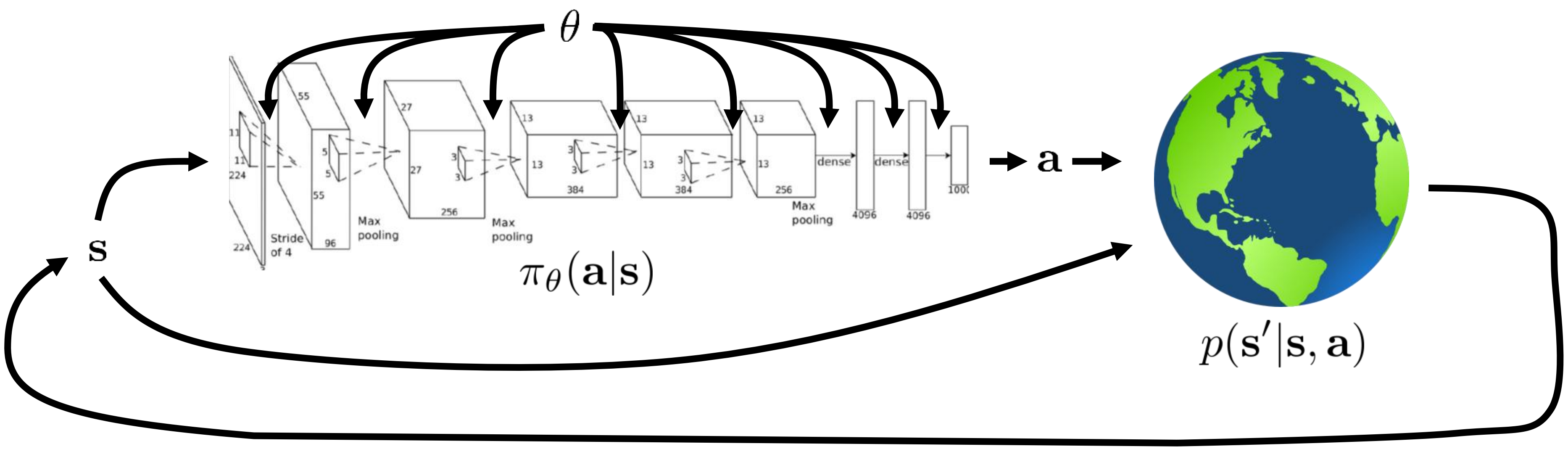
Recap: model-free reinforcement learning



$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

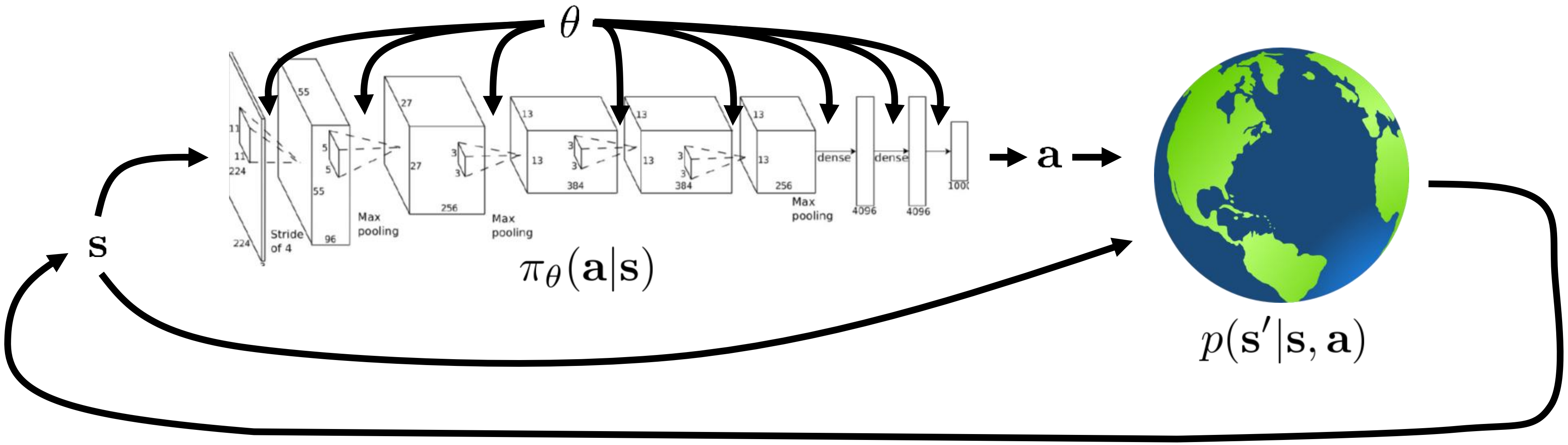
Recap: model-free reinforcement learning



$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}$$

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Recap: model-free reinforcement learning

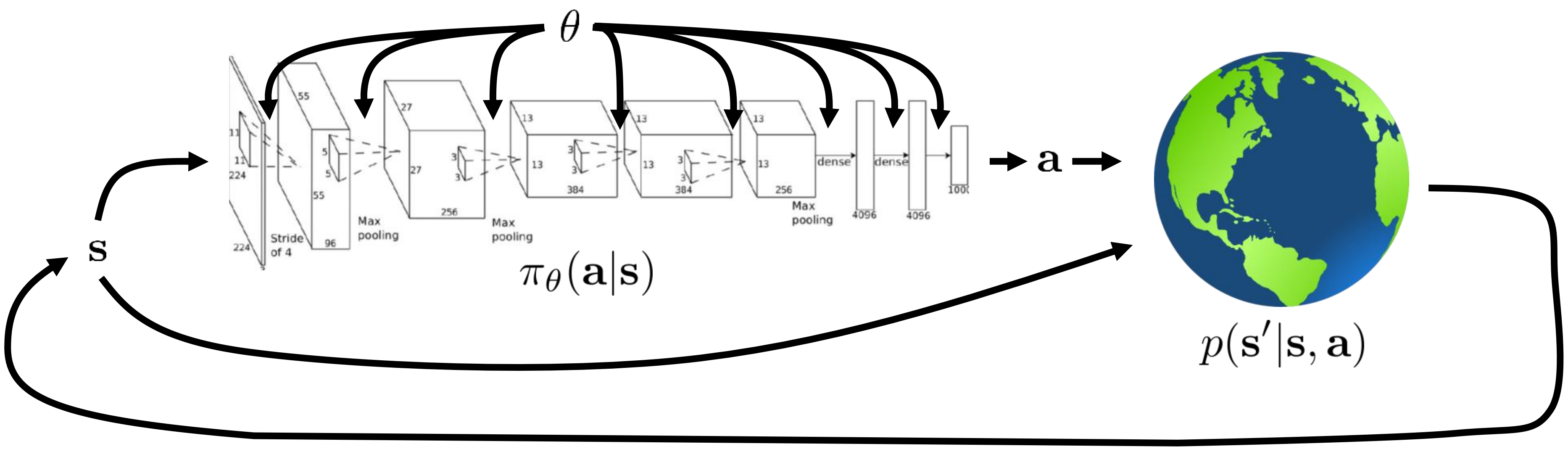


$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\dots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}$$

assume this is unknown

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Recap: model-free reinforcement learning



$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}$$

assume this is unknown

don't even attempt to learn it

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

What if we knew the transition dynamics?

What if we knew the transition dynamics?

- Often we do know the dynamics

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)
  3. Simulated environments (e.g., simulated robots, video games)

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)
  3. Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)
  3. Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics
  1. System identification – fit unknown parameters of a known model

What if we knew the transition dynamics?

- Often we do know the dynamics
    1. Games (e.g., Go)
    2. Easily modeled systems (e.g., navigating a car)
    3. Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics
    1. System identification – fit unknown parameters of a known model
    2. Learning – fit a general-purpose model to observed transition data

What if we knew the transition dynamics?

- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)
  3. Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics
  1. System identification – fit unknown parameters of a known model
  2. Learning – fit a general-purpose model to observed transition data

Does knowing the dynamics make things easier?

What if we knew the transition dynamics?

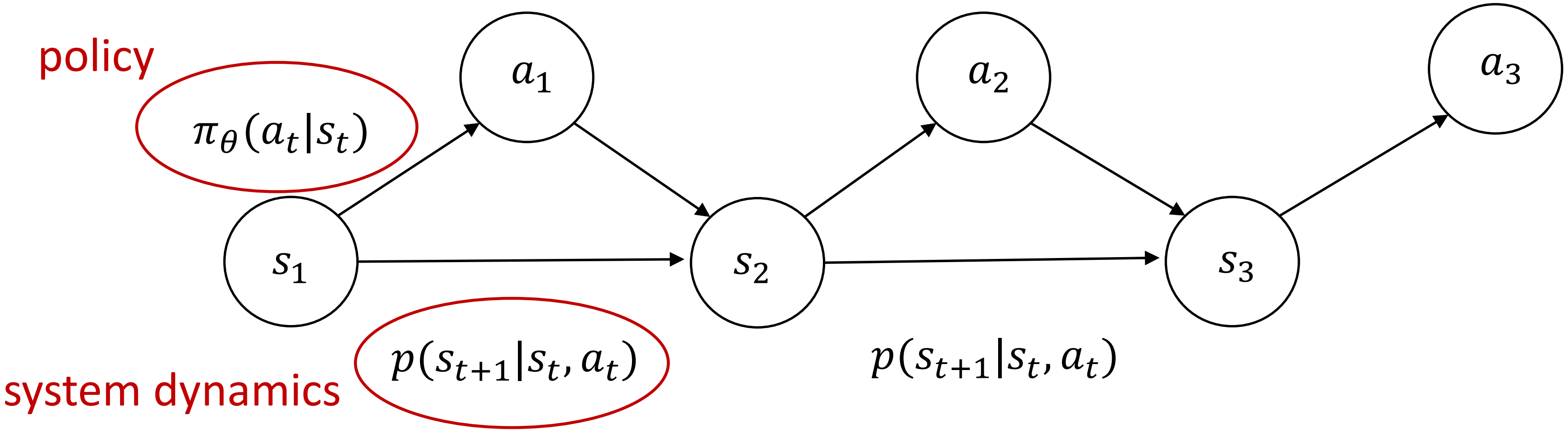
- Often we do know the dynamics
  1. Games (e.g., Go)
  2. Easily modeled systems (e.g., navigating a car)
  3. Simulated environments (e.g., simulated robots, video games)
- Often we can learn the dynamics
  1. System identification – fit unknown parameters of a known model
  2. Learning – fit a general-purpose model to observed transition data

Does knowing the dynamics make things easier?

Often, yes!

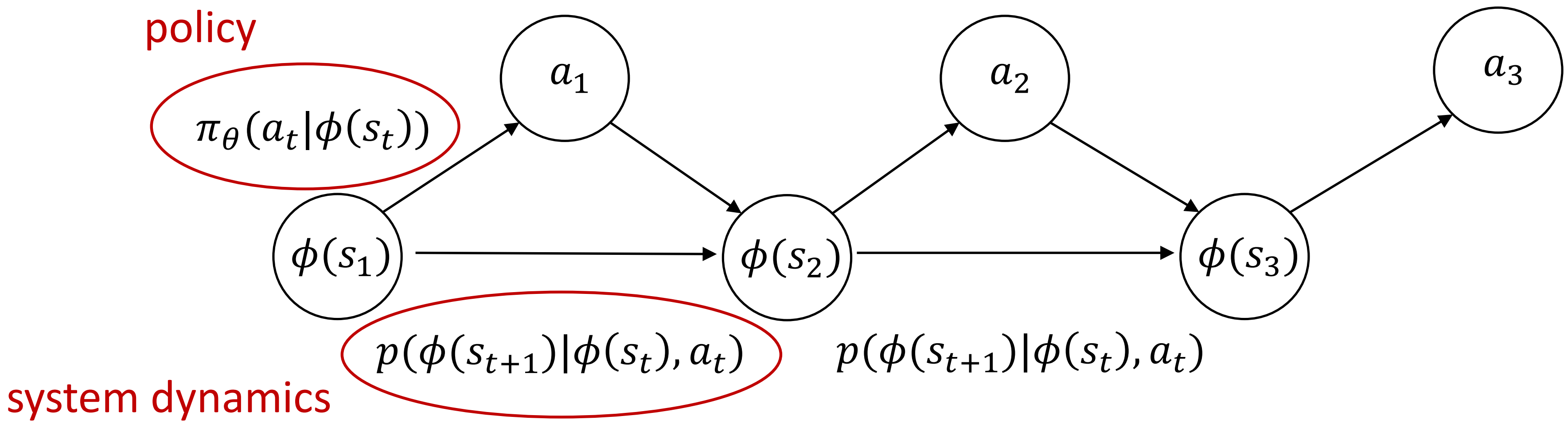
Model-based reinforcement learning

1. Model-based reinforcement learning: learn the transition dynamics, then figure out how to choose actions
2. Advantages:
    1. Avoid making expensive real-world or simulator actions
    2. Possibly differentiate through the dynamics to optimize action choice
3. We then have an optimal control problem



Simplifying Model-based reinforcement learning

1. Computing a complete environment model can be very expensive (has to generate images for vision-based policies)
2. Do we really need the full state?
3. Is there a simplified function of the state $\phi(s)$ that's sufficient for RL?
4. Should be sufficient to predict next state and for policy to choose next action.



Curiosity-driven Exploration by Self-supervised Prediction

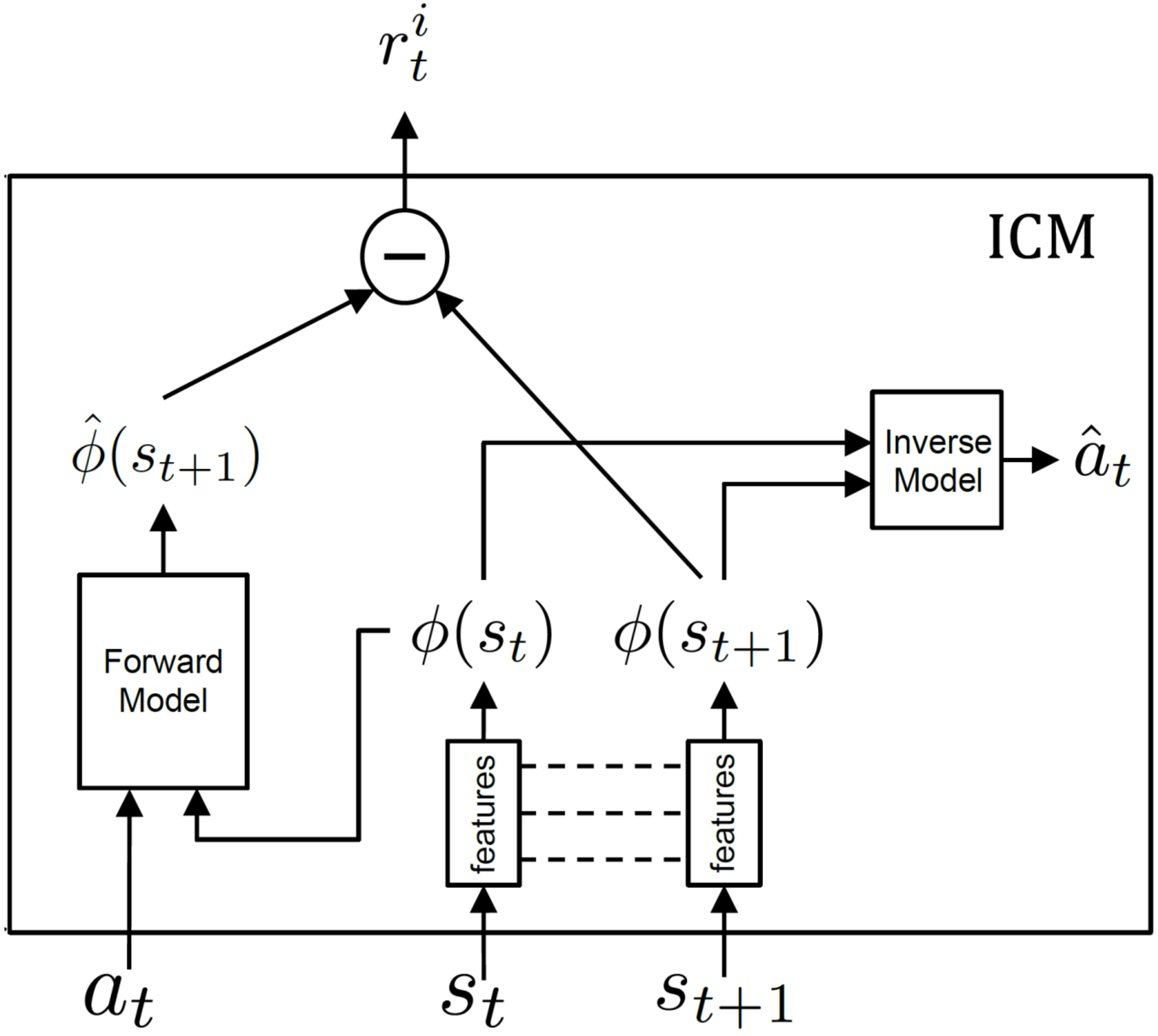
Pathak et al. 2017

1. Computes a simplified environment model $\phi(s_t)$
2. Uses error in the model's prediction to highlight states that need further exploration.

Intrinsic Curiosity Model (ICM)

Pathak et al. 2017

1. Uses an inverse model to ensure $\phi(s_t)$ is sufficient for action selection.
2. Estimates a forward model for next state.
3. The forward model error is the curiosity signal.

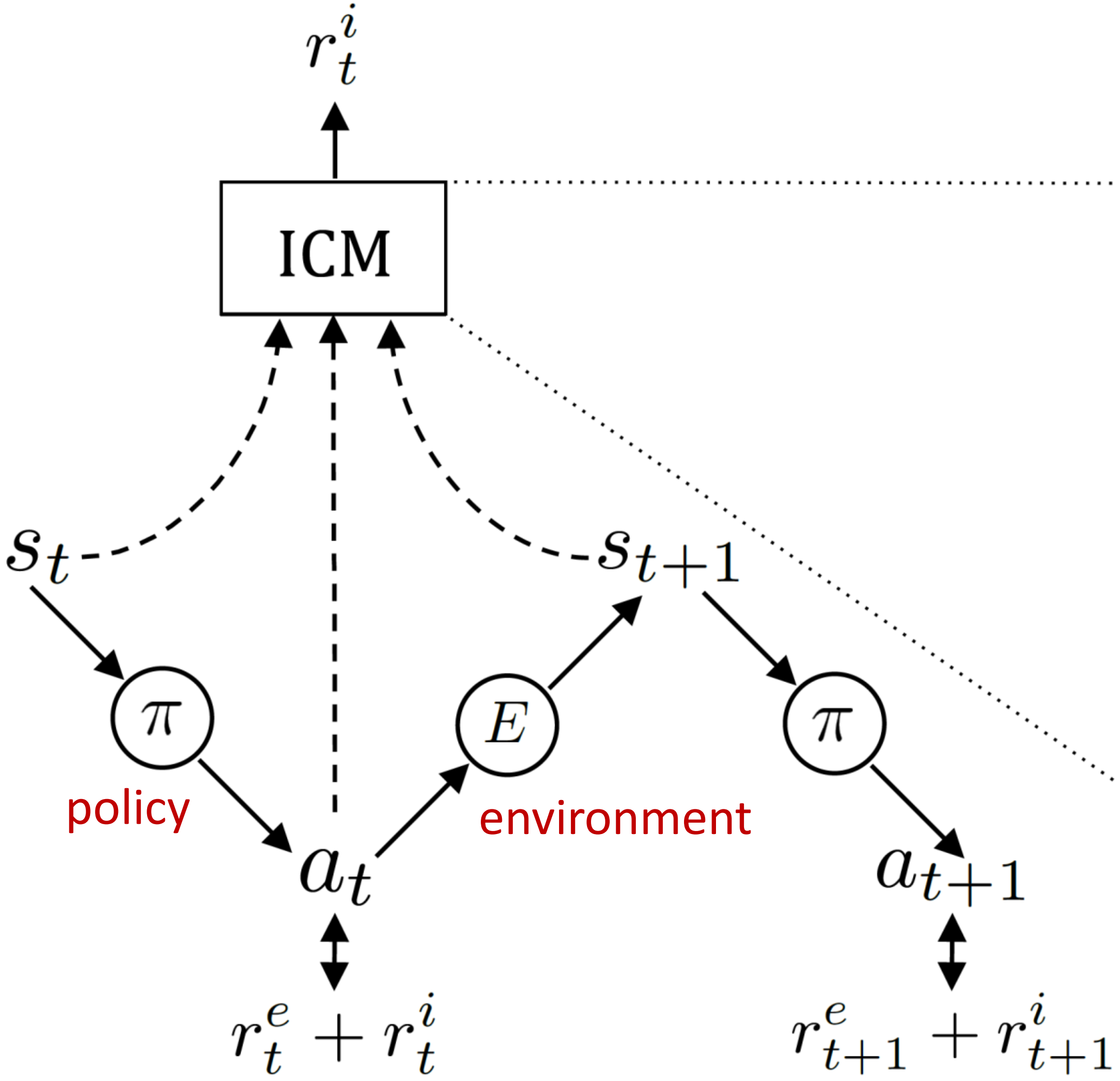


Curiosity-Driven Exploration

Pathak et al. 2017

1. The ICM provides an intrinsic reward signal r_t^i.

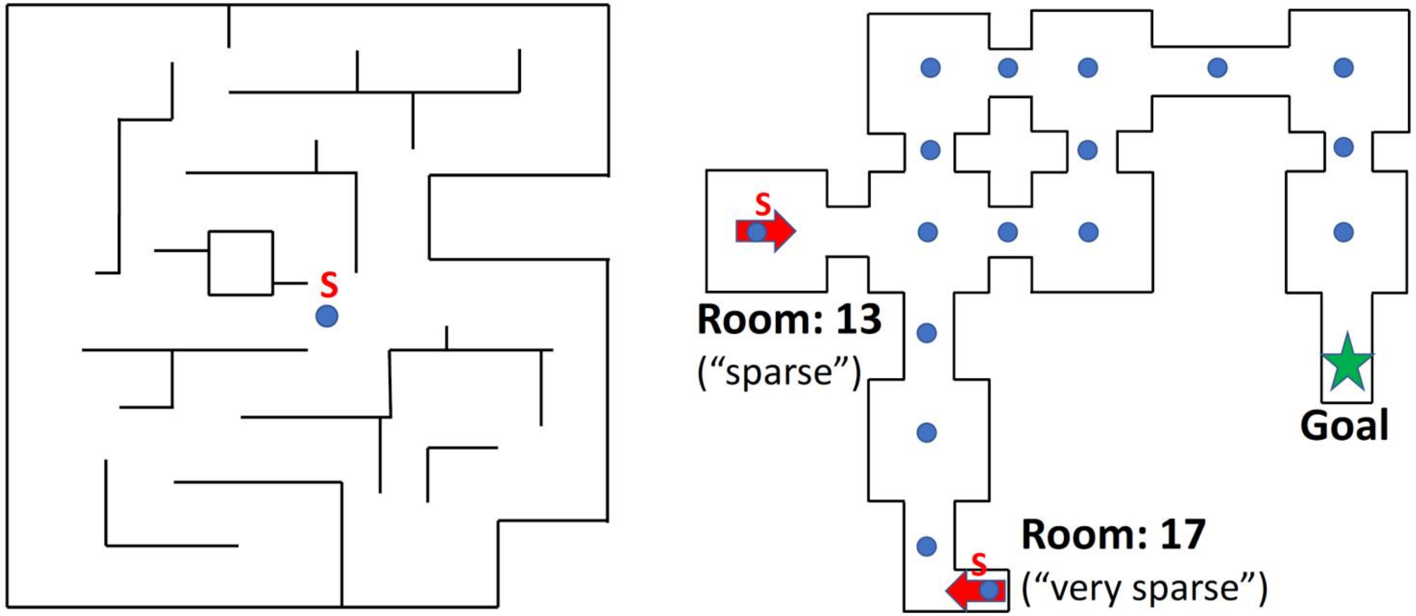
2. The policy can be trained by any RL method using the combined reward.



Curiosity-Driven Exploration

Pathak et al. 2017

Environments: Vizdoom

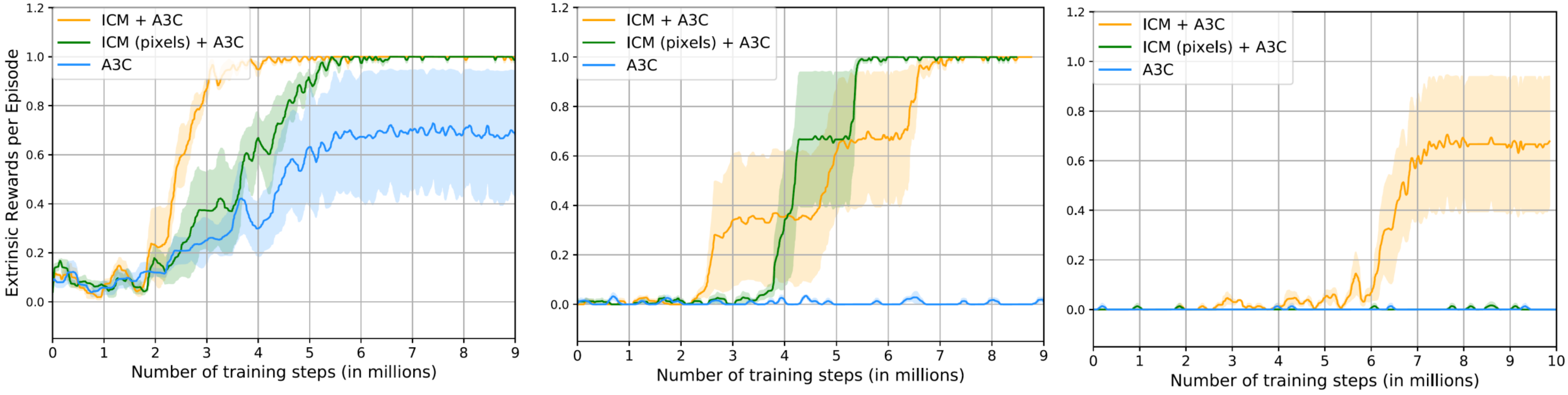


(a) Train Map Scenario (b) Test Map Scenario

Model is trained initially without goal reward (left). Agent is randomly started on one of the blue dots in the "dense" case (right).

Curiosity-Driven Exploration

Pathak et al. 2017



(a) "dense reward" setting

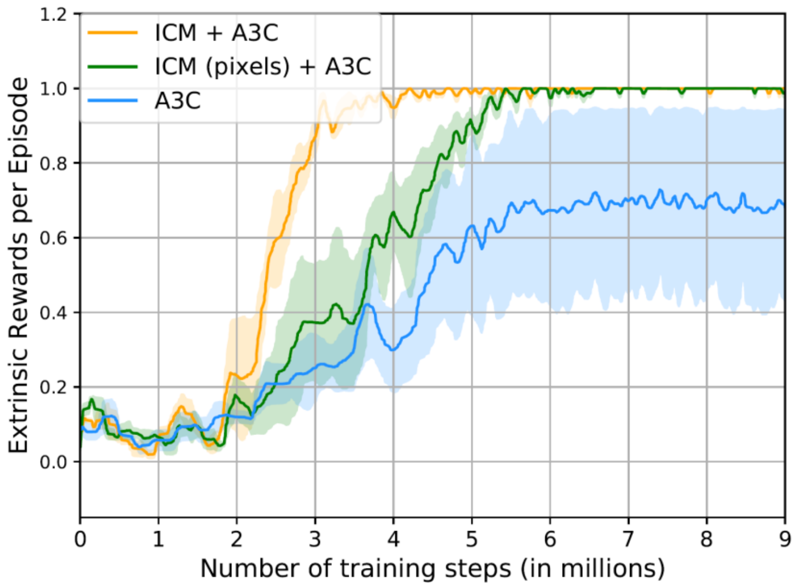
(b) "sparse reward" setting

(c) "very sparse reward" setting

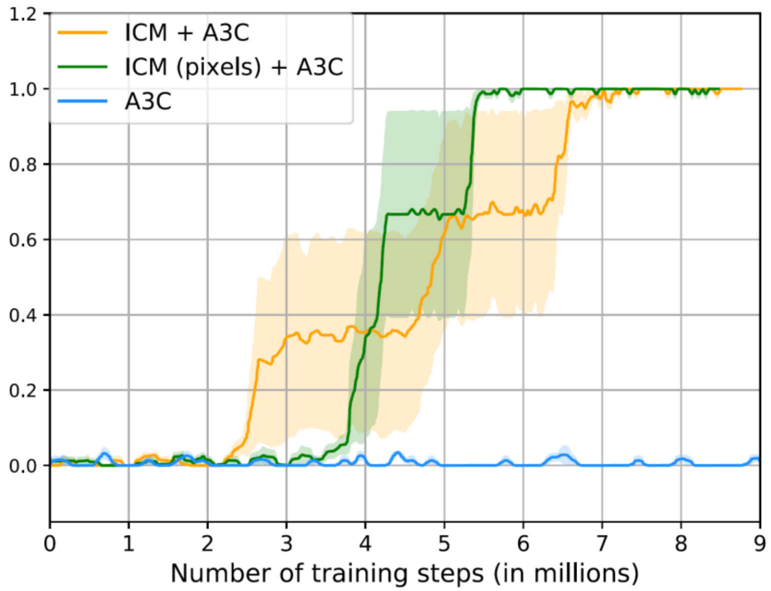
Curiosity-Driven Exploration

Pathak et al. 2017

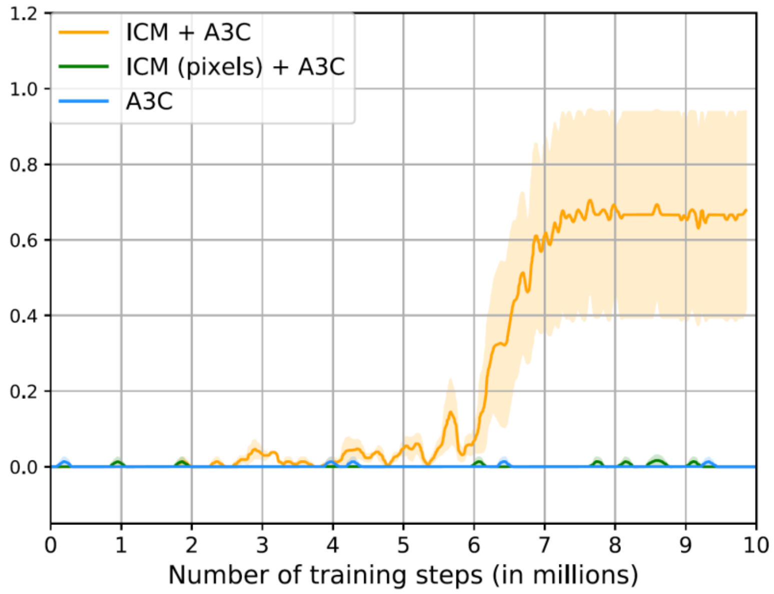
Relative performance improves as reward sparseness increases.



(a) "dense reward" setting

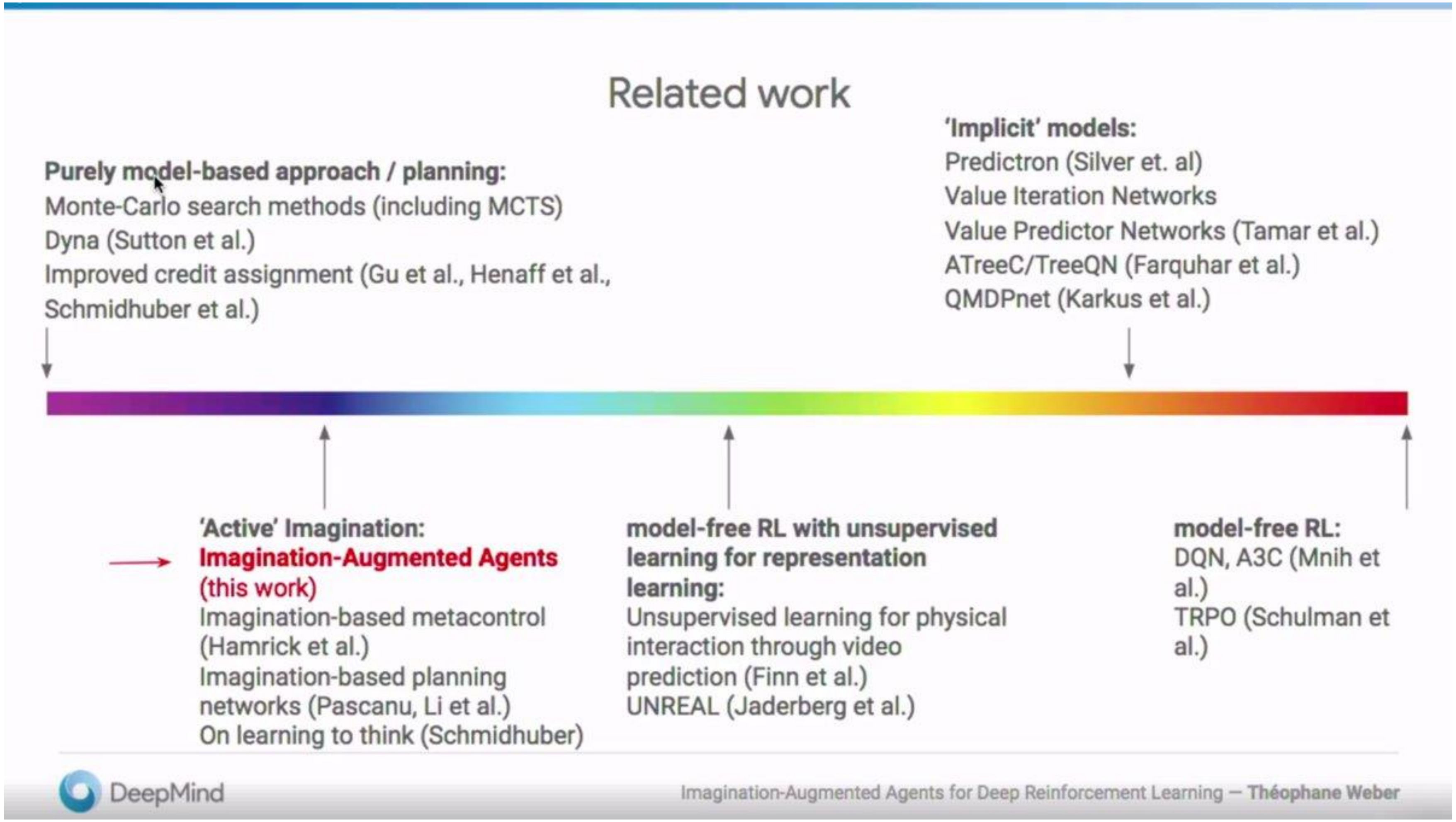


(b) "sparse reward" setting



(c) "very sparse reward" setting

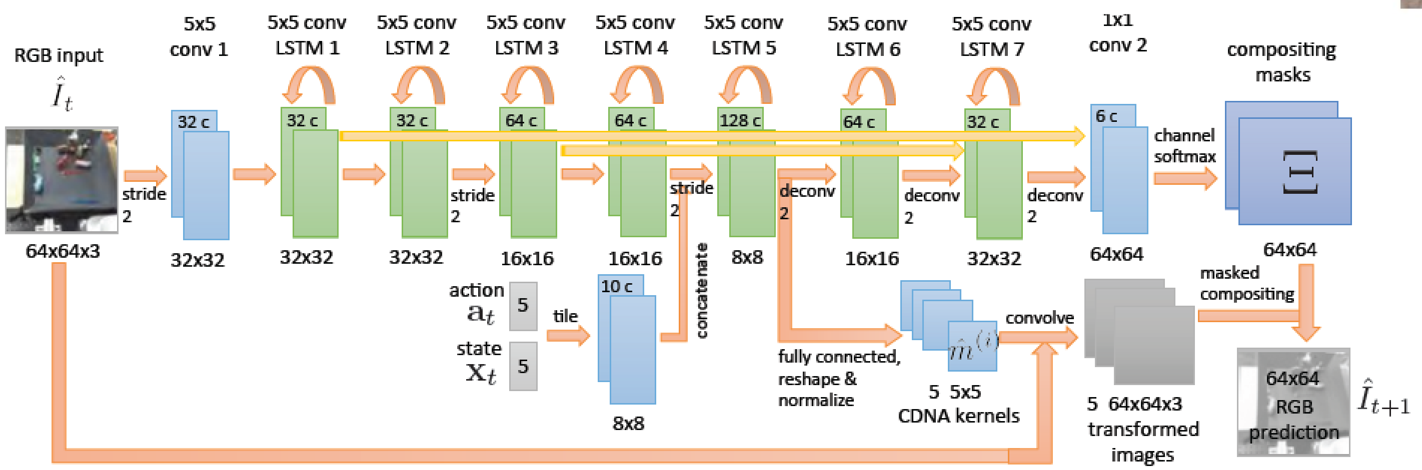
Model-Free vs Model-Based (Theophane Weber)

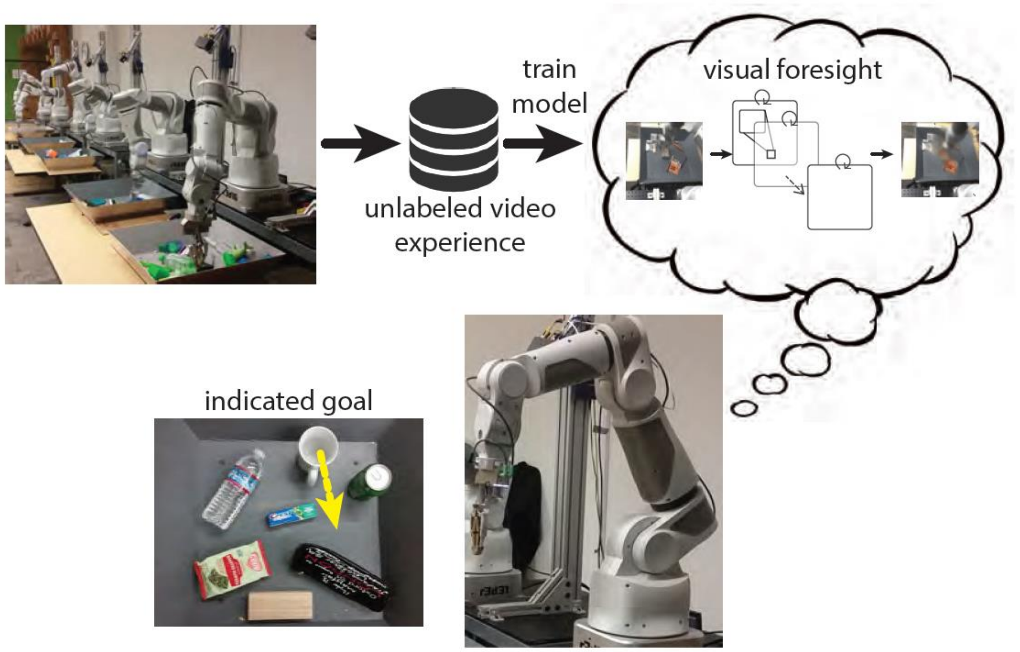


Model-Free vs Model-Based

Video predictive models, e.g.
"Deep Visual Foresight for Planning Robot Motion" by Finn et al.

Pixel-level recurrent image prediction:





Computes a flow (motion) map across the image.

Optimizes action choice using CEM (Cross-Entropy Method).

Aside: Stochastic optimization

Aside: Stochastic optimization

abstract away optimal control/planning:

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} J(\mathbf{a}_1, \ldots, \mathbf{a}_T)$$

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

$$\mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

$$\mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

simplest method: guess & check

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

$$\mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

simplest method: guess & check

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}} \qquad \mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

simplest method: guess & check

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Aside: Stochastic optimization

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

$$\mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

simplest method: guess & check "random shooting method"

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

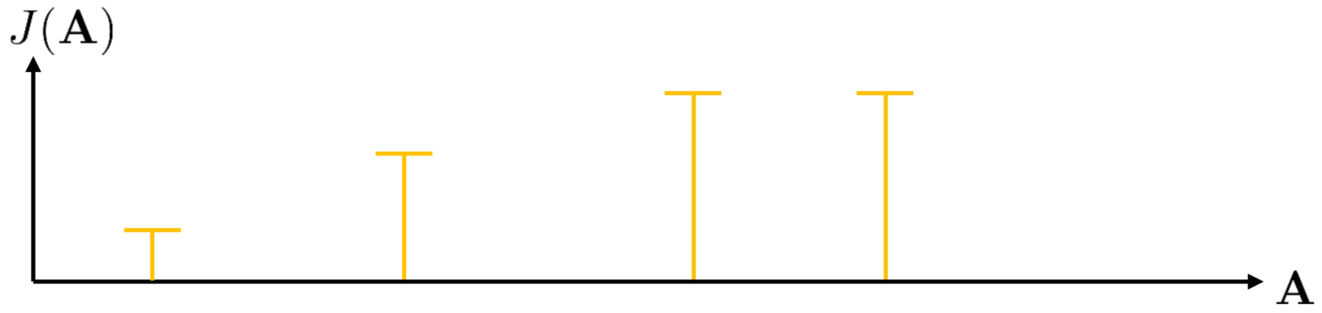
can we do better?

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \dots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

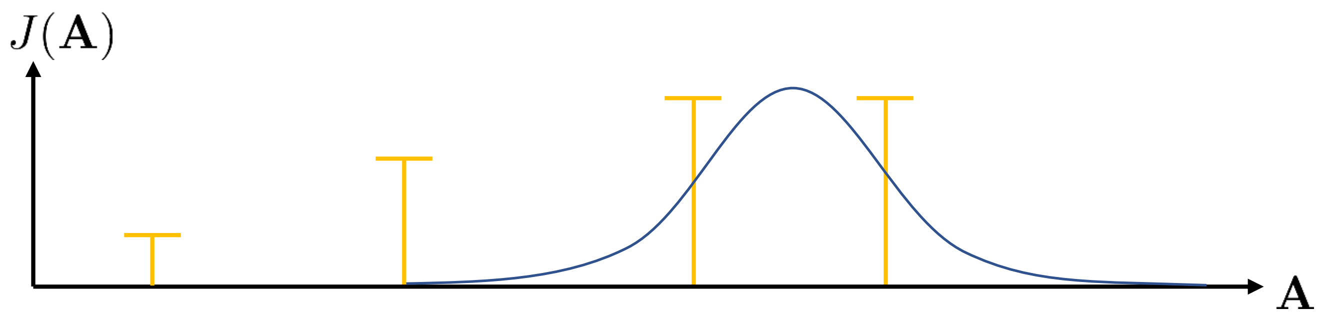


Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

   can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

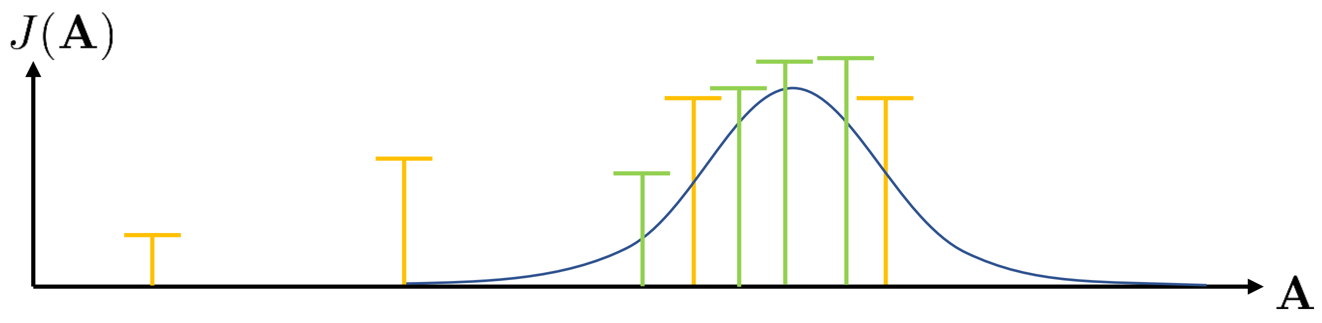


Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

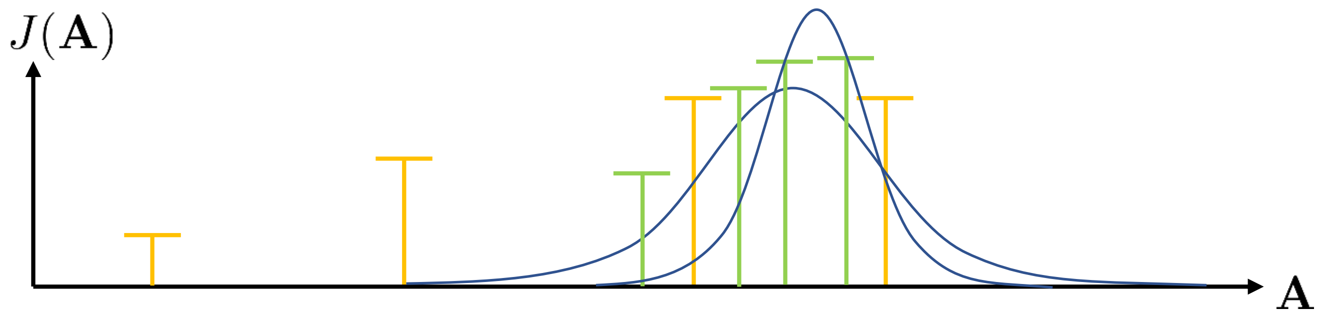


Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

can we do better?

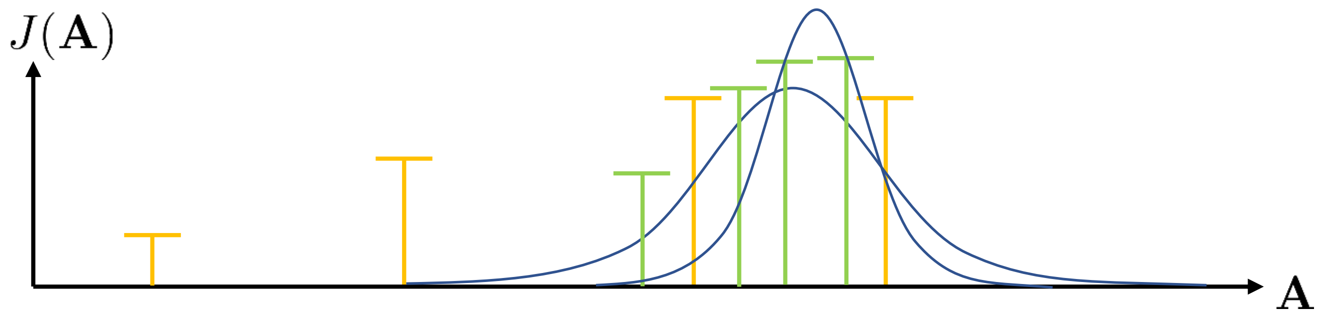


Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



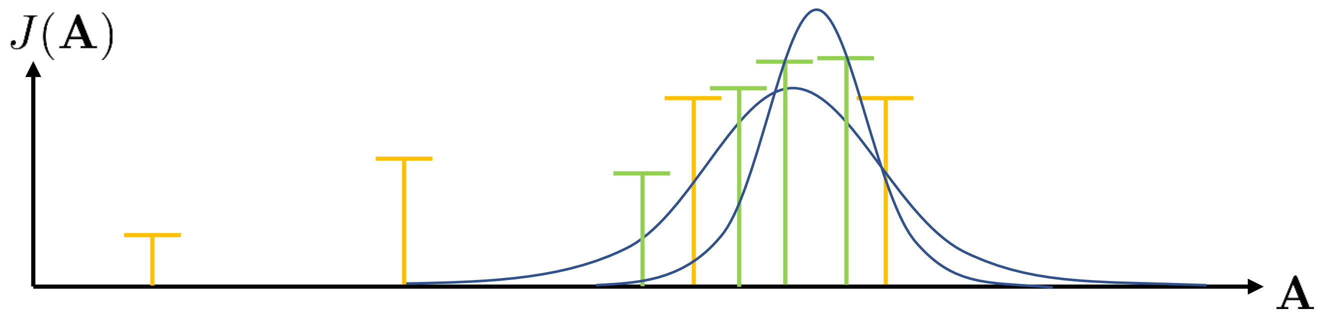
cross-entropy method with continuous-valued inputs:

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



cross-entropy method with continuous-valued inputs:

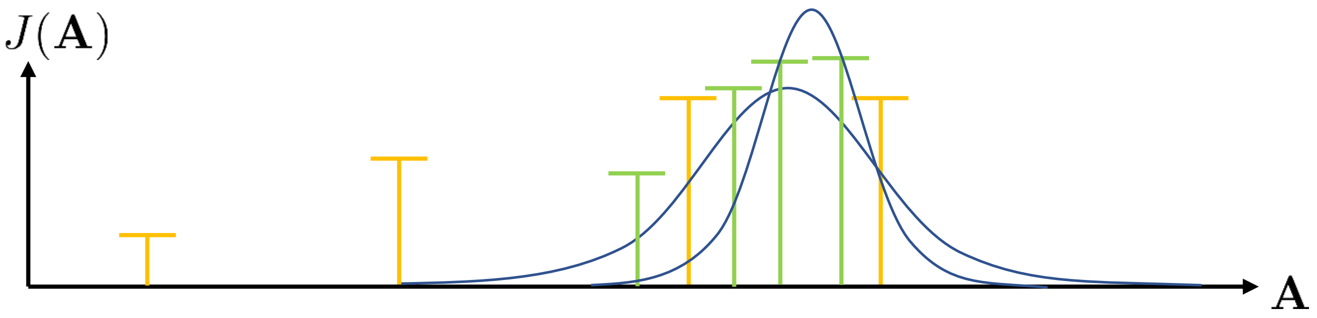
1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



cross-entropy method with continuous-valued inputs:

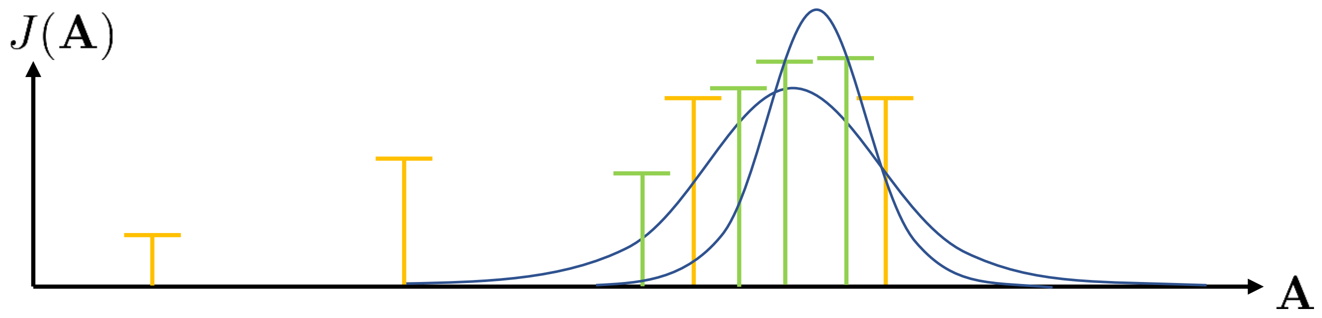
1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \ldots, J(\mathbf{A}_N)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \dots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



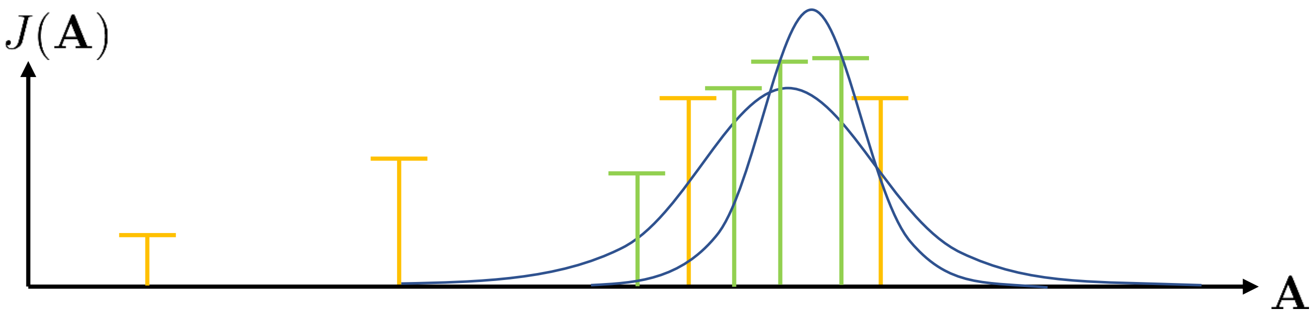
cross-entropy method with continuous-valued inputs:

1. sample $\mathbf{A}_1, \dots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \dots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \dots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

can we do better?



cross-entropy method with continuous-valued inputs:

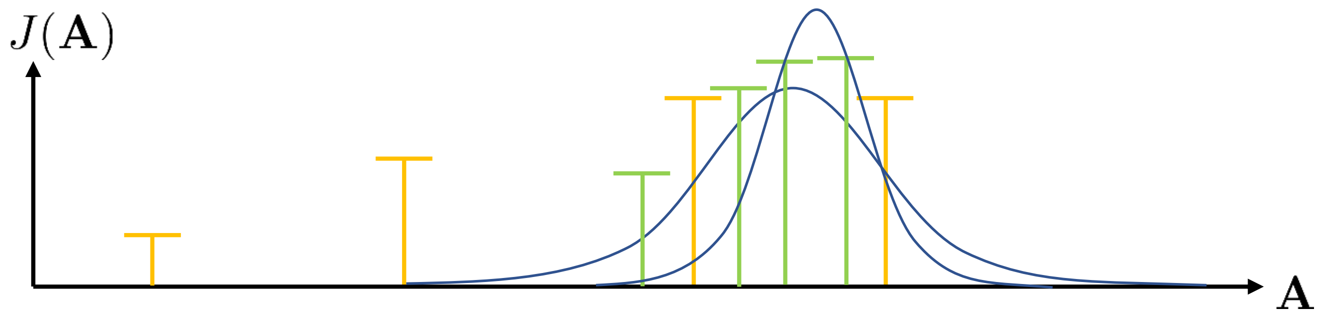
1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \ldots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \dots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



cross-entropy method with continuous-valued inputs:

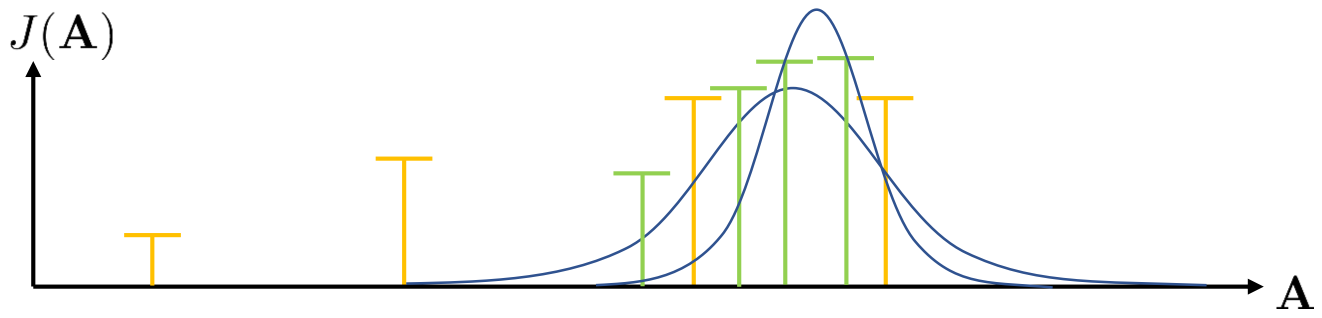
1. sample $\mathbf{A}_1, \dots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \dots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \dots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \dots, \mathbf{A}_{i_M}$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)

can we do better?

2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$



typically use Gaussian distribution

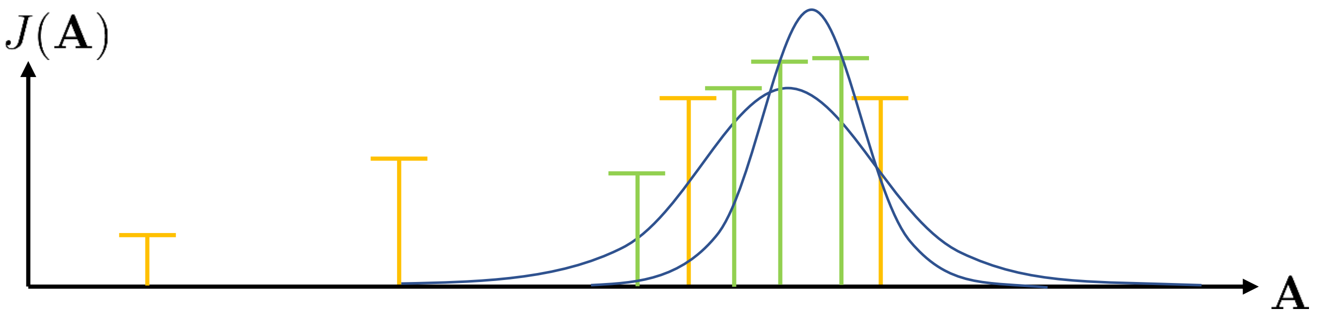
cross-entropy method with continuous-valued inputs:

1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \ldots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \dots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

can we do better?



typically use Gaussian distribution

see also: CMA-ES (sort of like CEM with momentum)

cross-entropy method with continuous-valued inputs:

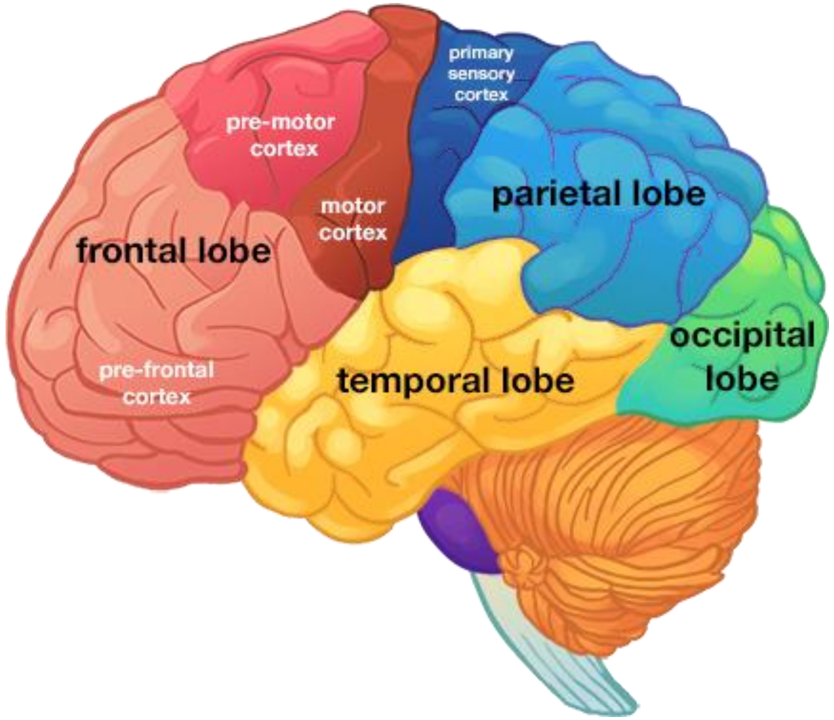
1. sample $\mathbf{A}_1, \dots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \dots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \dots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \dots, \mathbf{A}_{i_M}$

Imagination

s or $\phi(s)$, that is the question…

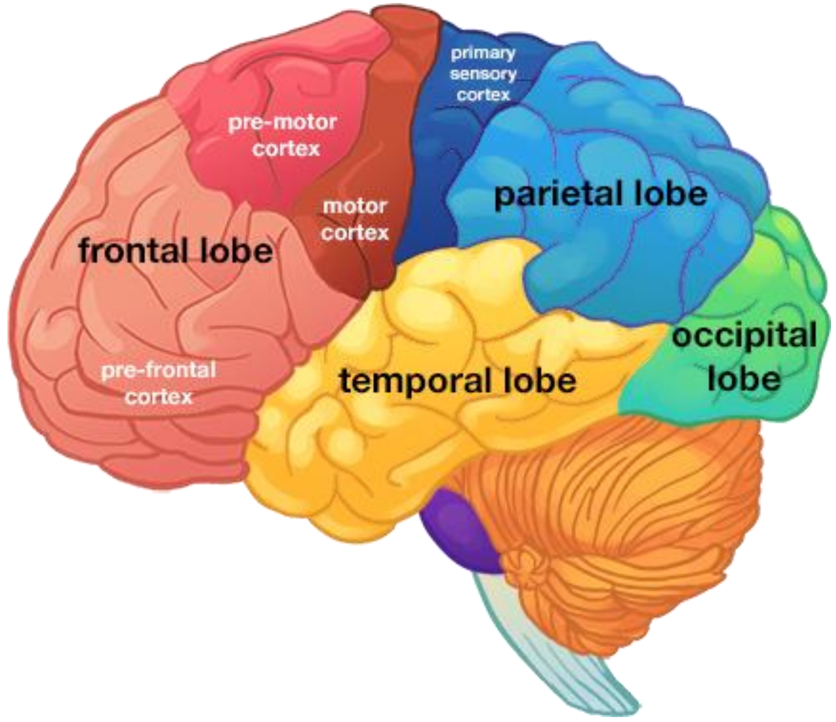
i.e. how important is it to have an accurate facsimile of the input images, vs and embedding tailored for planning?

Visual Imagination



Visual Imagination

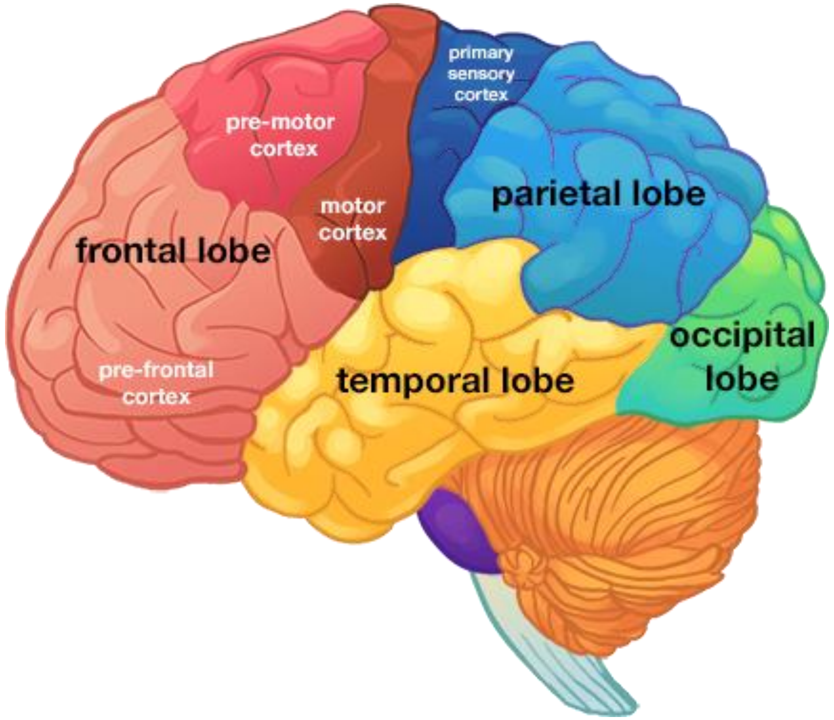
What do people do?



Visual Imagination

What do people do?

We know the occipital/temporal areas are involved (visual and planning).

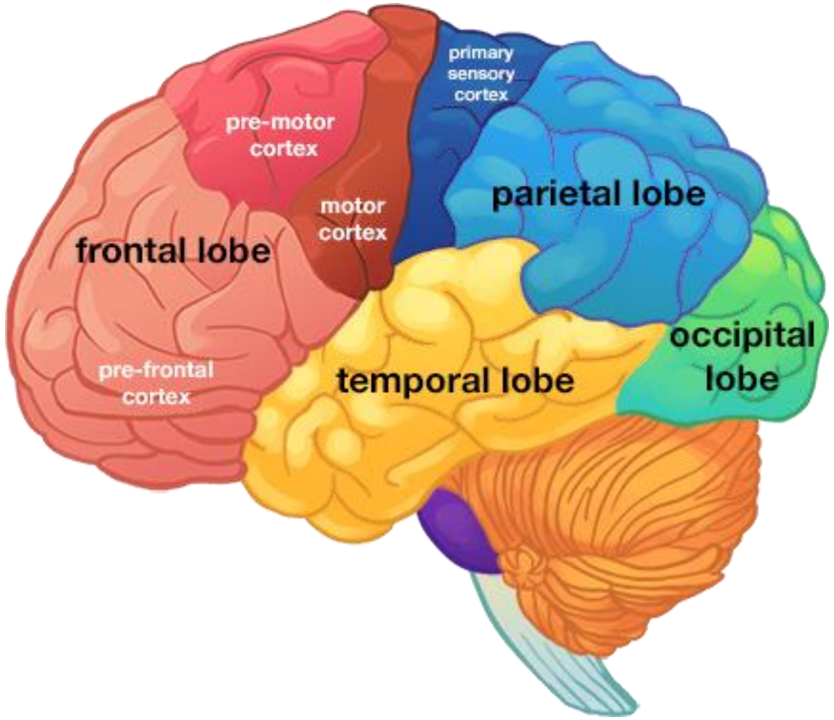


Visual Imagination

What do people do?

We know the occipital/temporal areas are involved (visual and planning).

But the results on primary visual cortex V1 (the pixels) in visual imagination experiments is mixed.



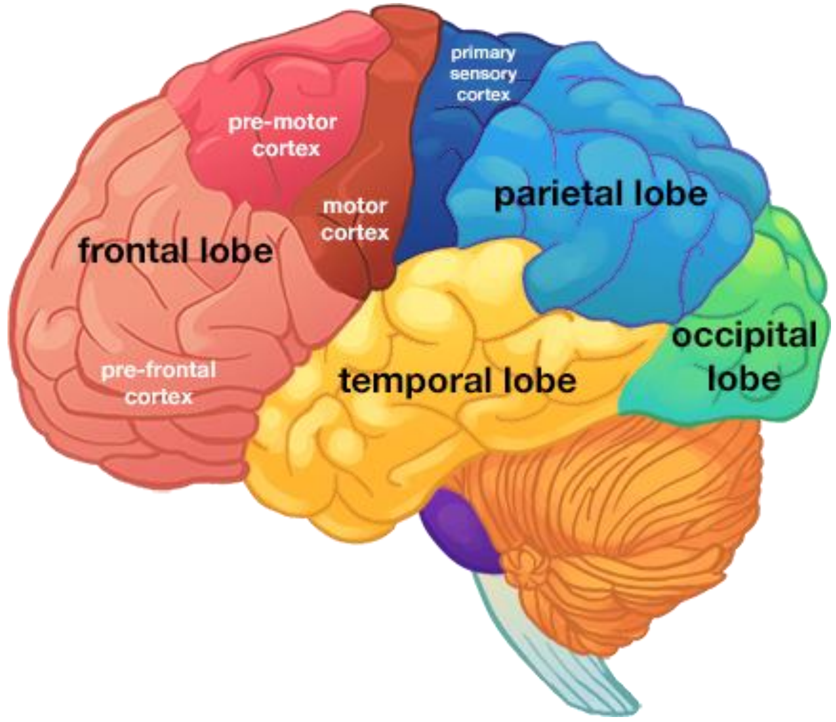
Visual Imagination

What do people do?

We know the occipital/temporal areas are involved (visual and planning).

But the results on primary visual cortex V1 (the pixels) in visual imagination experiments is mixed.

See this recent study:



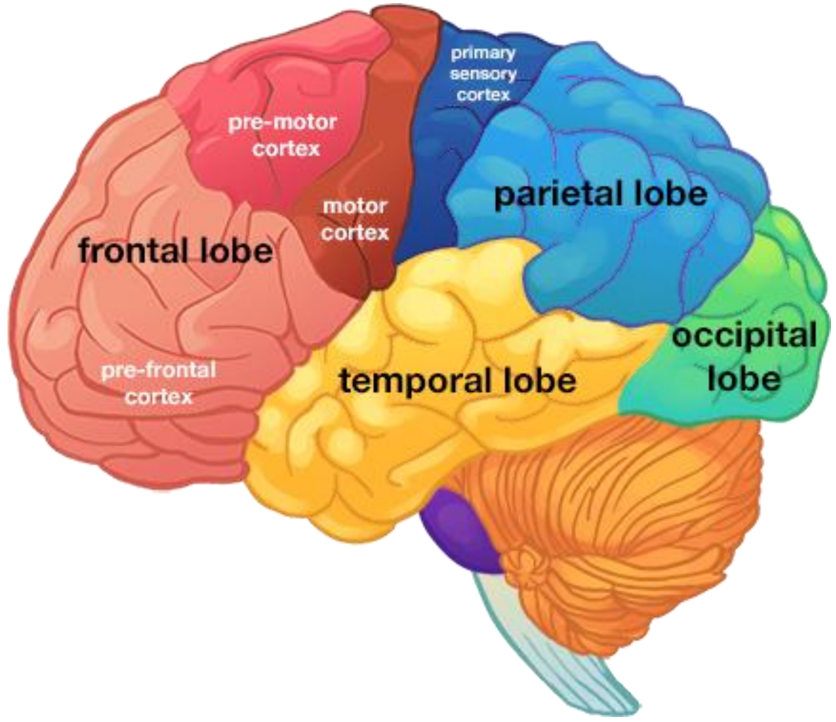
Visual Imagination

What do people do?

We know the occipital/temporal areas are involved (visual and planning).

But the results on primary visual cortex V1 (the pixels) in visual imagination experiments is mixed.

See this recent study:
"Vivid visual mental imagery in the absence of the primary visual cortex" by Bridge et al.



Imagination Architecture

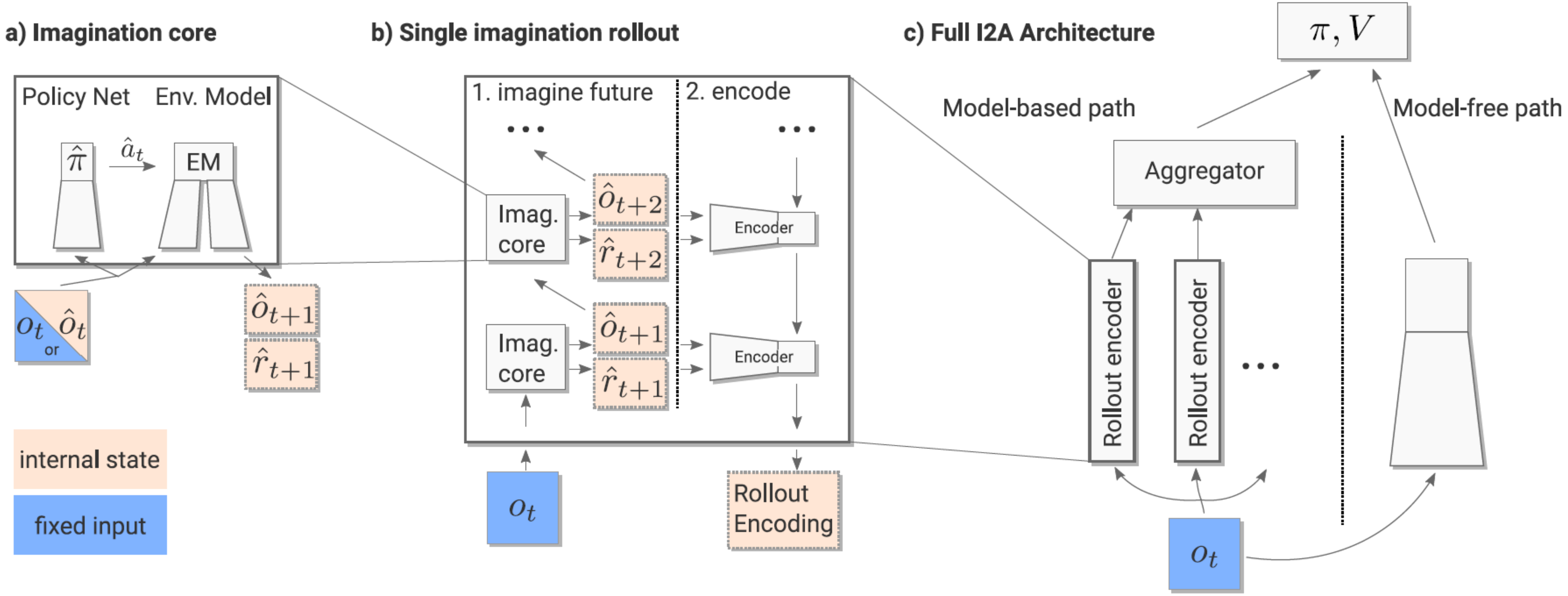
"Imagination-Augmented Agents for Deep Reinforcement Learning" Weber et al

In the mean time we can build it and try it…

Imagination Architecture

"Imagination-Augmented Agents for Deep Reinforcement Learning" Weber et al

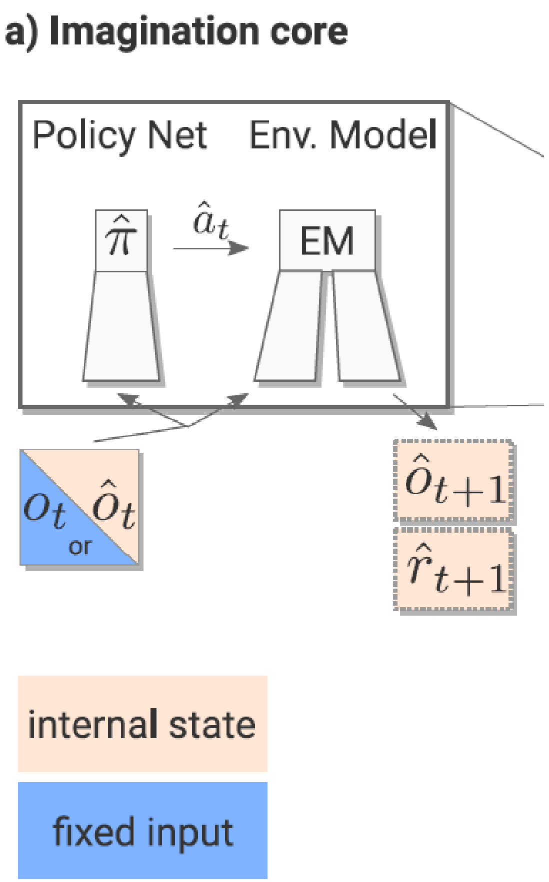
Symbols with $\hat{}$ represent imagined values



Imagination Architecture

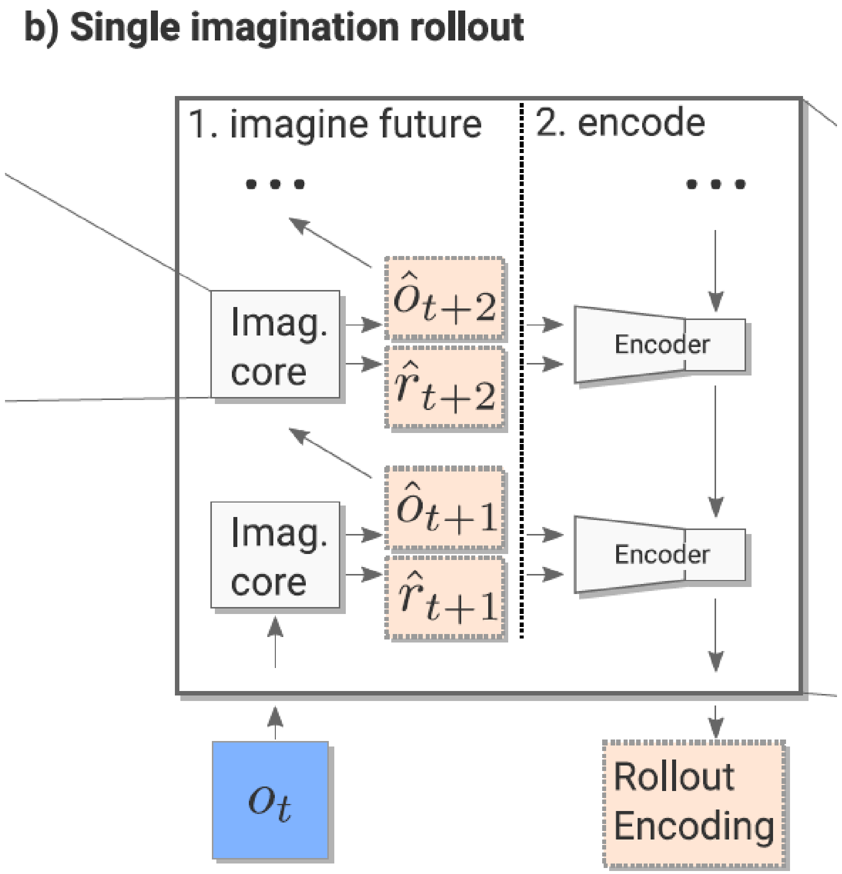
The rollout policy $\hat{\pi}$ generates the action sequence.

Its produced by a standard policy network.



Imagination Architecture

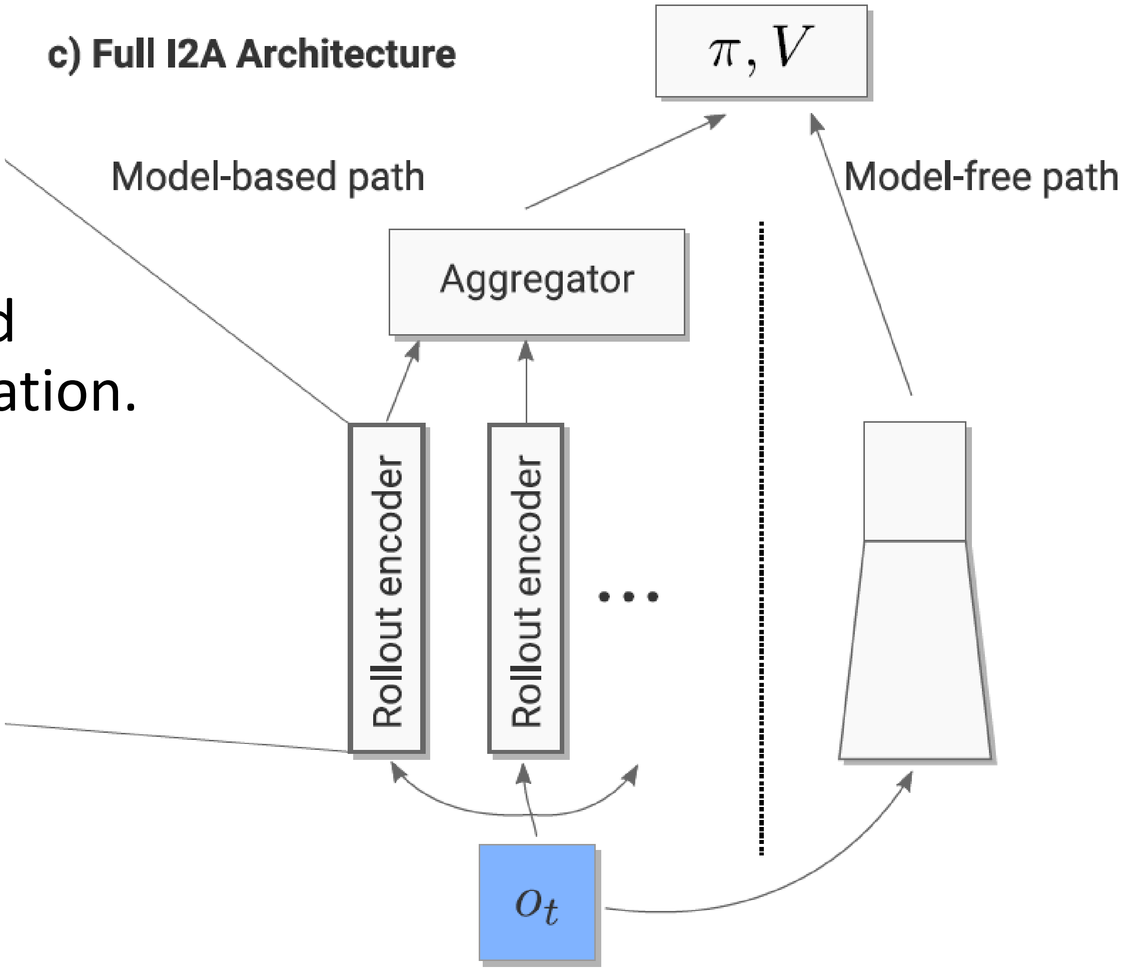
The imagination rollout block recurrently computes a sequence of observations and rewards.



Imagination Architecture

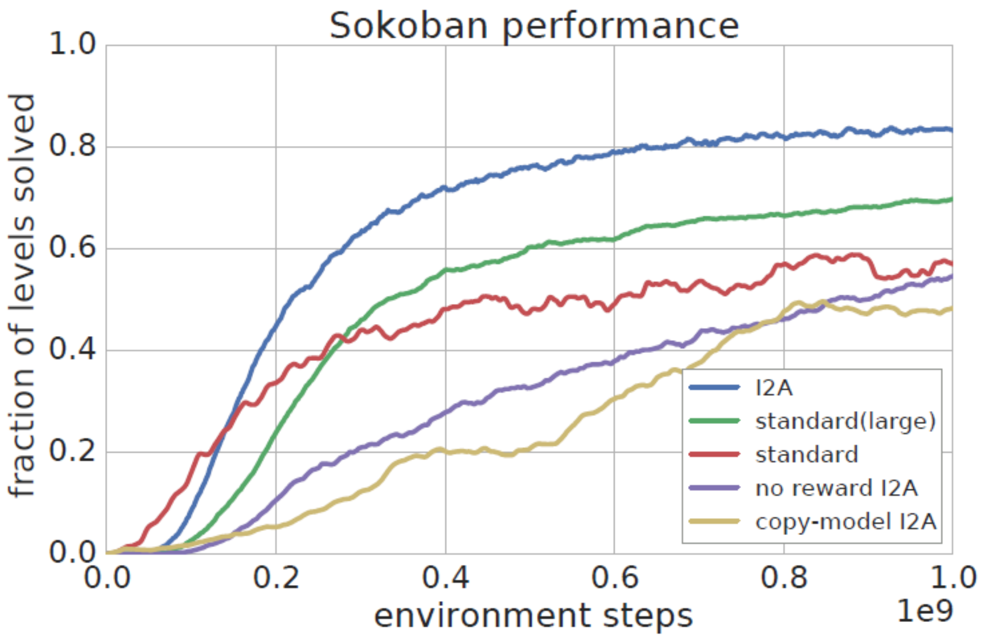
The imagined sequence forms “advice”
to the final policy and value function blocks.

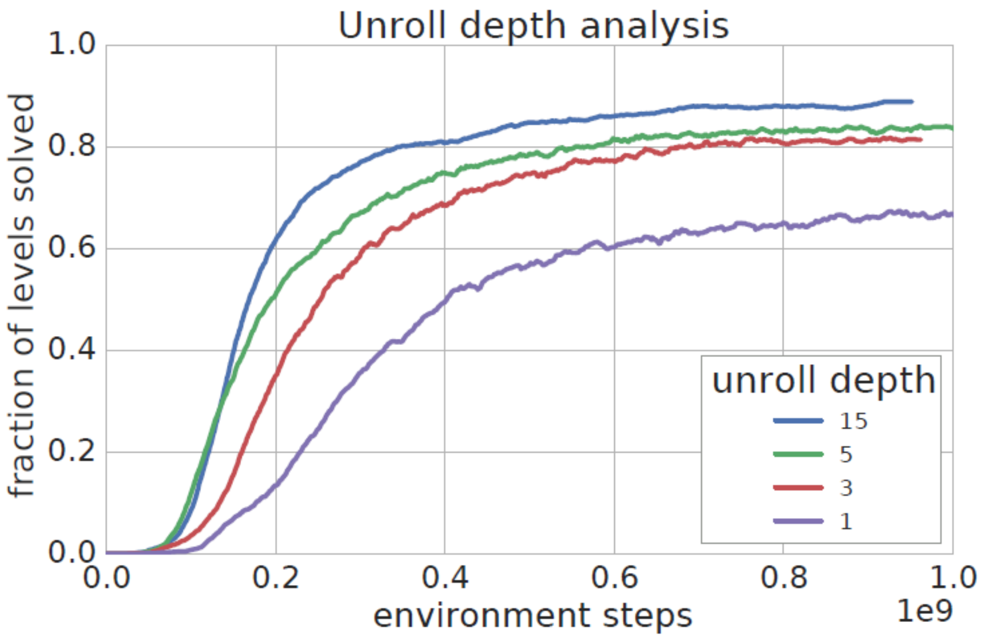
They are both also informed by a feed-forward
network connected to the current real observation.



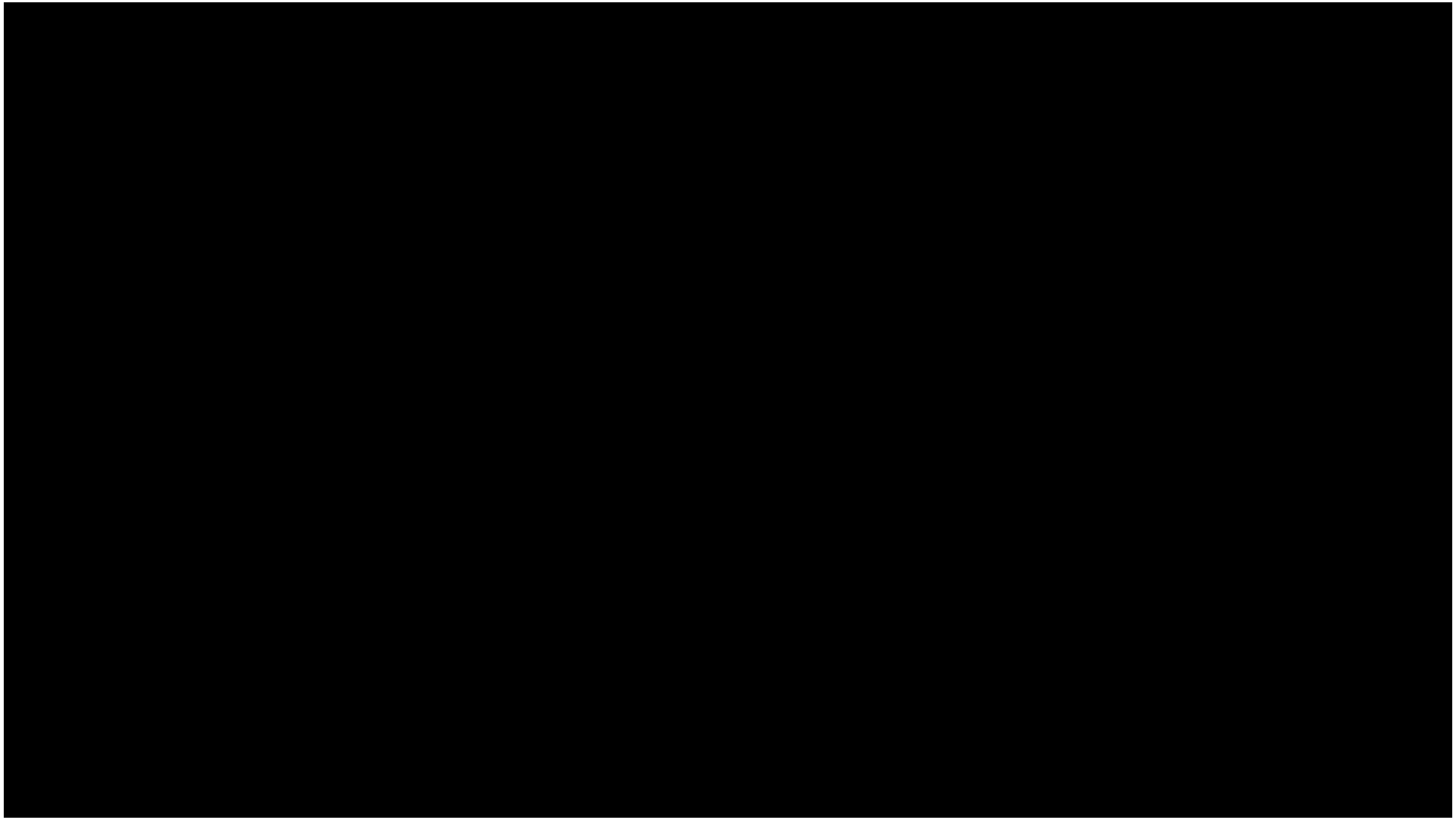
Imagination Architecture

Performance on Sokoban:



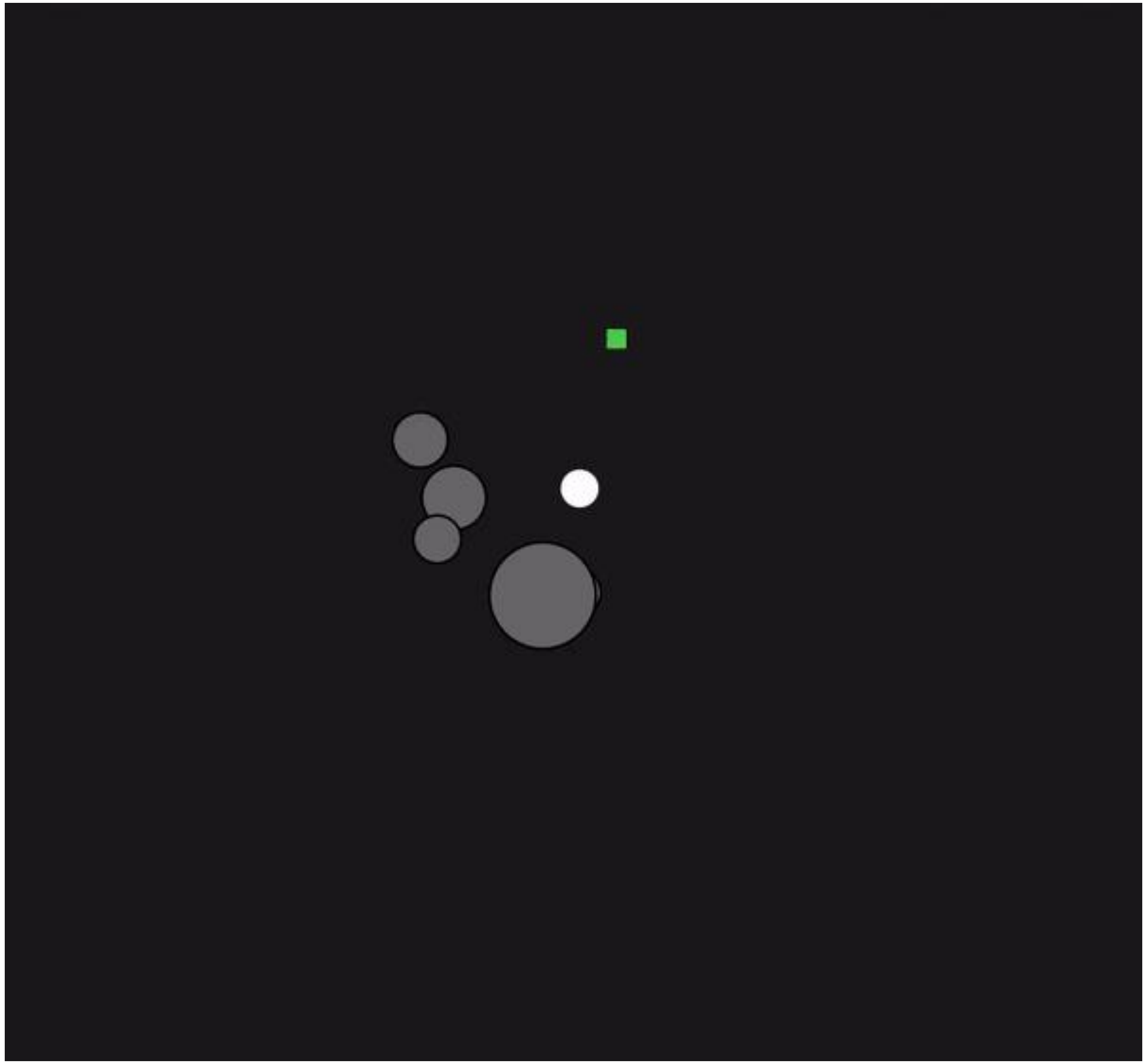


Imagination Architecture



Imagination Architecture

- An agent playing the spaceship game.
- Red lines are executed trajectories.
- Blue and green are imaged trajectories.



Take-aways

- Optimistic exploration methods favor poorly-explored states, assuming their value is as high as it could be. Use counts to estimate uncertainties.
- An elegant way to model θ uncertainty is to train an ensemble of models using bootstrap sampling. Sample from the posterior → choose a model.
- Avoid random walks at all costs! $\Omega(N^2)$ cost to explore N states. Simple methods: epsilon-greedy, entropy regularization and Thompson sampling all do random walks.
- Doing better requires memory (counts, environment models etc.)

Take-aways

- Can plan and anticipate states and rewards with a simplified environment model ($\phi(s)$ instead of s).
- Human visual imagination doesn’t always use pixel-level representation.
- But visual representation is the most powerful, and allows the system to use policies or other networks trained for very different tasks.
- Imagination was used as “advice” in the imagination architecture. Another network needs to learn how to use it (CEM etc.).