CMPT 983

Grounded Natural Language Understanding

March 11, 2021

Instruction Following

(review of RL)

How to Train Your Agent

(A Crash Course in Sequential Decision Making with Deep Nets)

Stefan Lee – OSU, CS539 – Fall 2019

Today

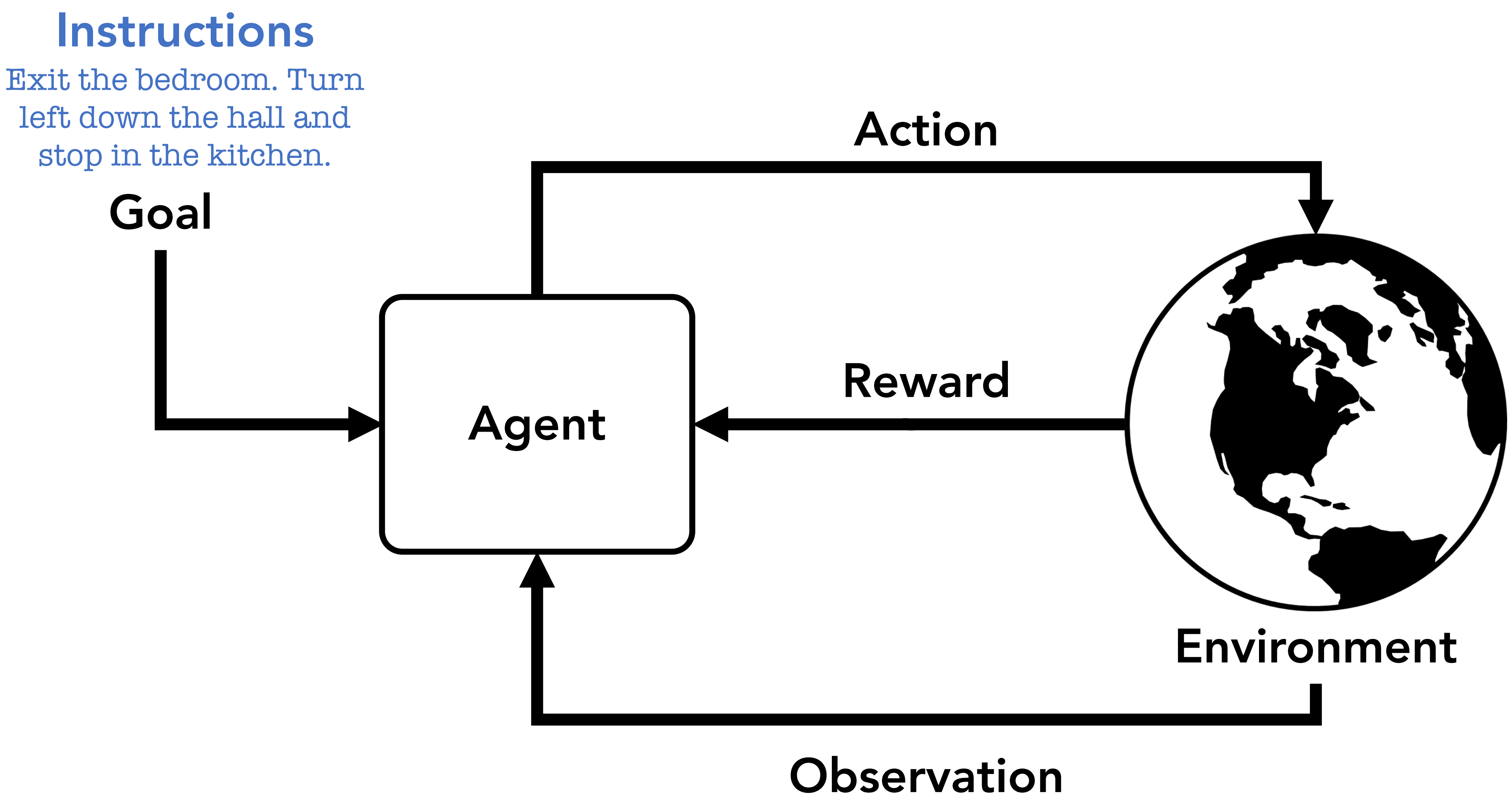
- Intro and Notation

- Imitation Learning
  - Behavior Cloning
  - Direct Policy Learning
  - Sketch of Inverse Reinforcement Learning

- Reinforcement Learning
  - Policy-based (REINFORCE, Actor-Critic)
  - Value-based (Q-Learning)
  - Model-based

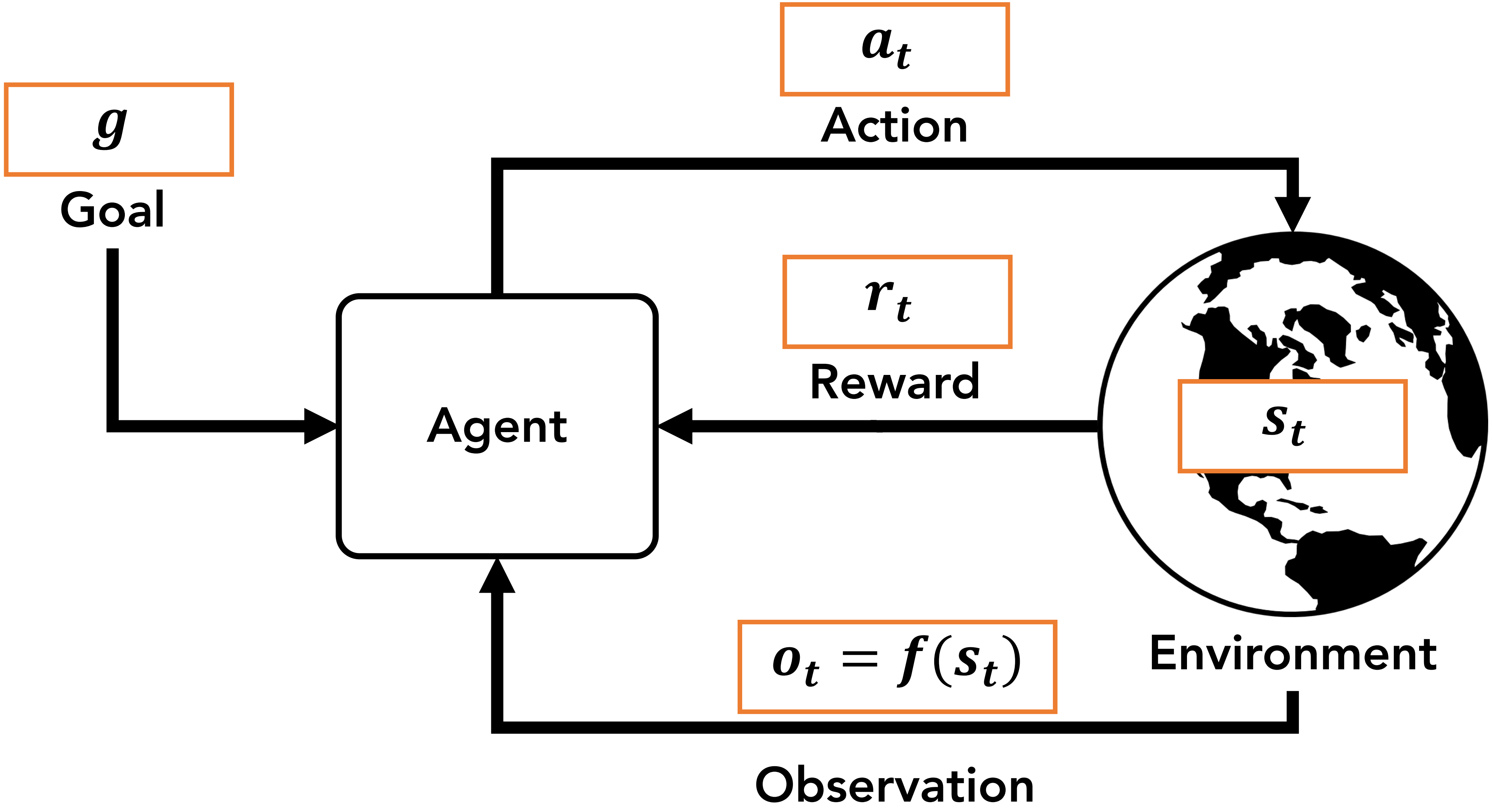
Intro and Notation

A General Embodied Agent



Slide Credit: Stefan Lee

A General Embodied Agent



Slide Credit: Stefan Lee

Some Notation

Markov Decision Process (MDP)

Defined as $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

- $\mathcal{S}$ Set of possible **states**
- $\mathcal{A}$ Set of possible **actions**
- $\mathrm{R}: \mathcal{S} \times \mathcal{A} \rightarrow \Omega_{\mathbb{R}}$ Distribution of **reward** given state-action pair
- $\mathbb{P}: \mathcal{S} \times \mathcal{A} \rightarrow \Omega_{\mathcal{S}}$ **Transition** function – distribution over next states
- γ **Discount** factor

state, action, reward
at each time step

Life looks like $(s_0, a_0, r_0, \; s_1, a_1, r_1, \ldots)$

where $s_{t+1} \sim \mathbb{P}(s_{t+1} | s_t, a_t)$ and $r_t \sim R(r_t | s_t, a_t)$

Slide Credit: Stefan Lee

Some Notation

Markov Decision Process (MDP)

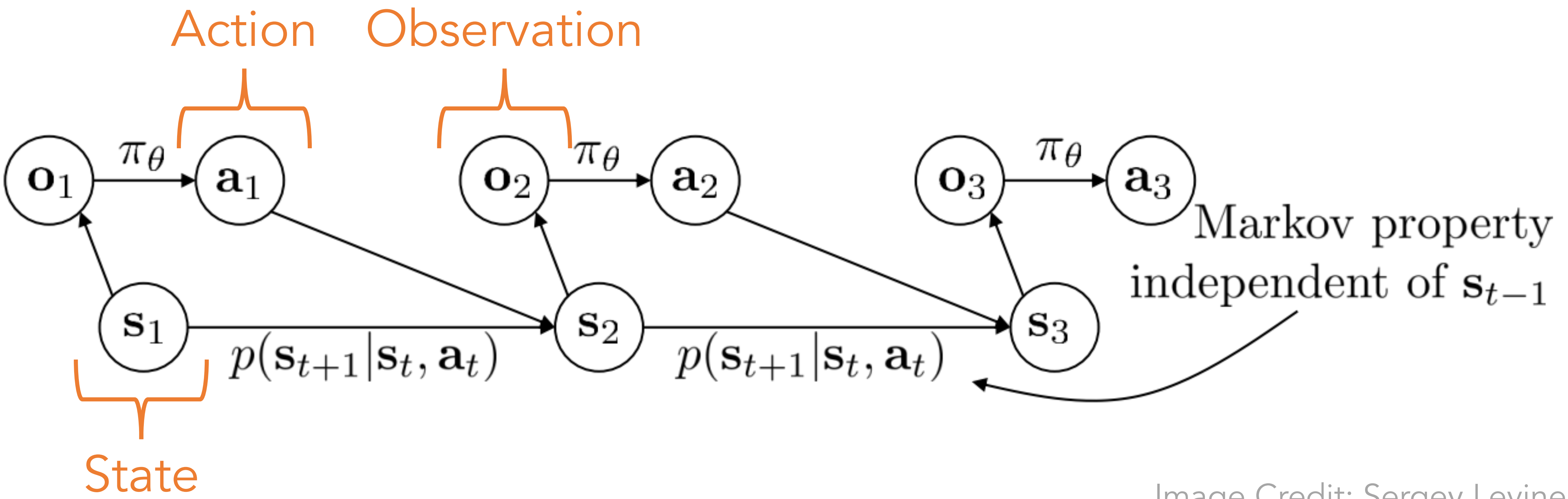


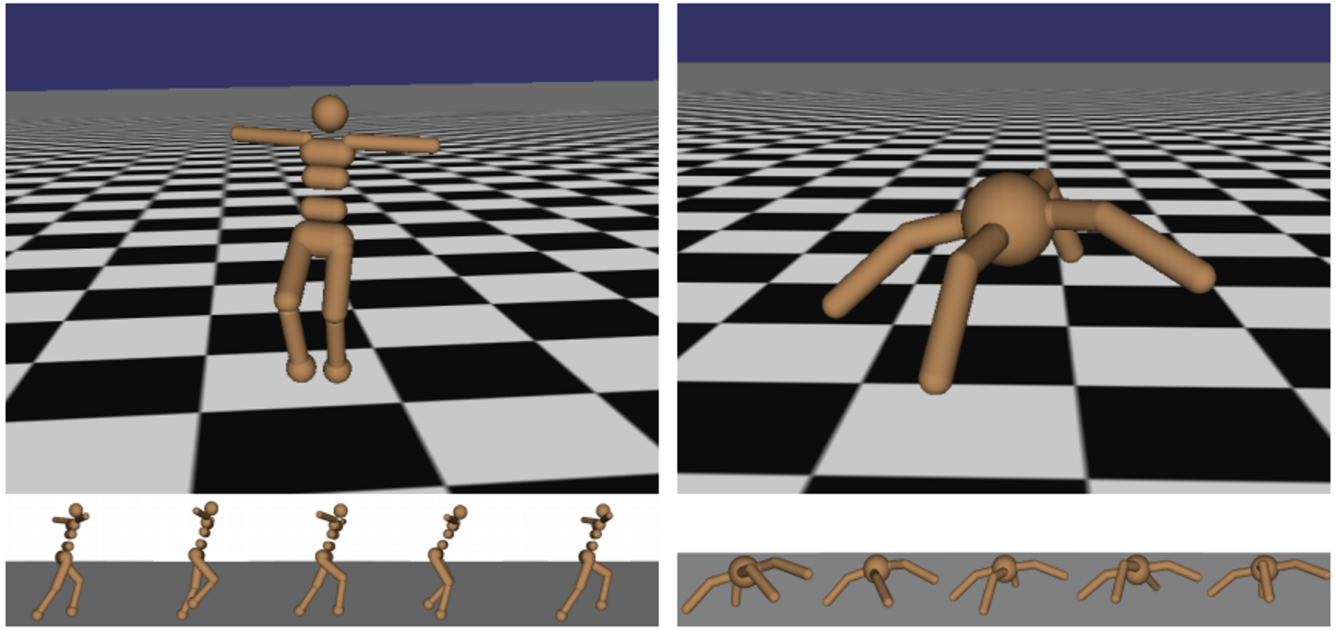
Image Credit: Sergey Levine

POMDP: Partially observed MDP

- Often we don’t know what the states are!

Examples MDPs

Robot Locomotion



Figures copyright John Schulman et al., 2016. Reproduced with permission.

Make the robot move forward

State Space: Angle and position of the joints
Action Space: Torques applied on joints
Reward Function: 1 at each time step upright + forward movement

Examples MDPs

Atari Games

Complete the game with the highest score

Observation Space: Raw pixel inputs of the game state
Action Space: Game controls e.g. Left, Right, Up, Down
Reward Function: Score increase/decrease at each time step

Examples MDPs

PointGoal Visual Navigation

Navigate to the specified point

Observation Space: Raw pixel inputs
Action Space: Forward, Turn Left, Turn Right
Reward Function: Distance increase/decrease per time step
+ "It hurts to be alive" penalty

Slide Credit: Stefan Lee

Some Notation

Common Terminology and Definitions

Policy – How should the agent act?

- Stochastic policy $\pi: \mathcal{S} \rightarrow \Omega_{\mathcal{A}}$ $a_t \sim \pi(s_t)$
- Deterministic policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ $a_t = \pi(s_t)$
- Optimal policy $\boldsymbol{\pi}^* = \boldsymbol{argmax}_{\boldsymbol{\pi}} \ \mathbb{E}_{\boldsymbol{s}_0}[\boldsymbol{V}_{\boldsymbol{\pi}}(\boldsymbol{s}_o)]$

Value – How good is each state? Or state-action pair?

- (State) Value Function $V_\pi(s_t) = \mathbb{E}_{\pi,\mathbb{P}}\left[\sum_{i=t}^{\infty} \gamma^{i-1} r_i\right]$
- Q Function $Q_\pi(s_t, a) = R(s_t, a) + \gamma \mathbb{E}_{s_{t+1}}[V_\pi(s_{t+1})]$
- Advantage $A_\pi(s_t, a) = Q_\pi(s_t, a) - V_\pi(s_t)$

Expected discounted return

How much better is taking the action *a* than the average?

Slide Credit: Stefan Lee

Some Notation

Common Terminology and Definitions

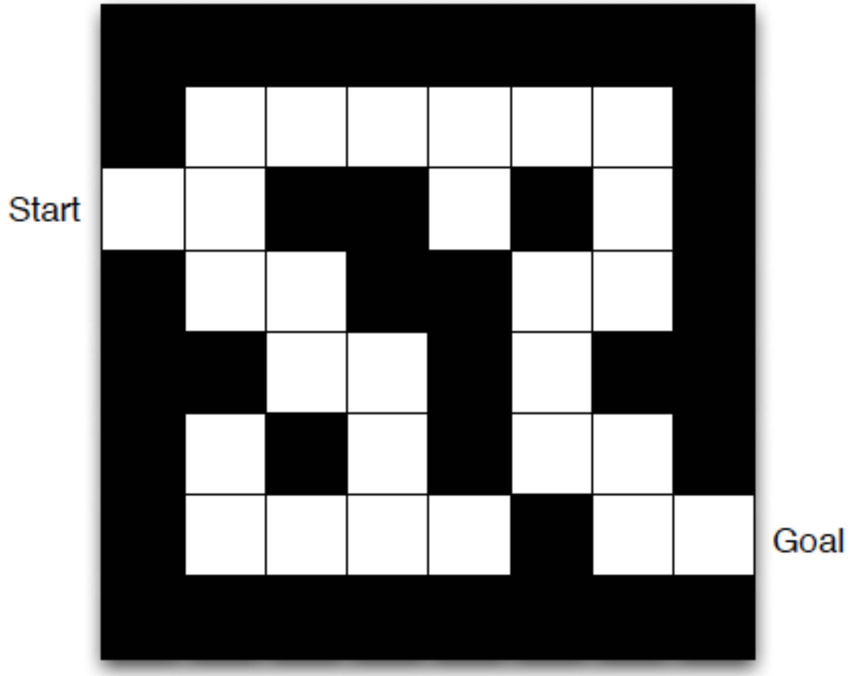
Model – What will happen when the agent acts?

- Learn to mimic the transition function $M\colon \mathcal{S} \times \mathcal{A} \rightarrow \Omega_{\mathcal{S}}$

Rollout – What happens if we let the policy act for a while?

- Trajectory $\tau = (s_t, a_t, s_{t+1}, a_{t+1}, \ldots)$
- $\boldsymbol{\tau} \sim \prod \mathbb{P}(s_{t+1}|s_t, a_t)\pi(a_t|s_t)\, P(s_o)$ often written $\tau \sim \pi$
- Can also consider states visited by policy: $P(s|\pi)$ or $\mathrm{s} \sim \pi$

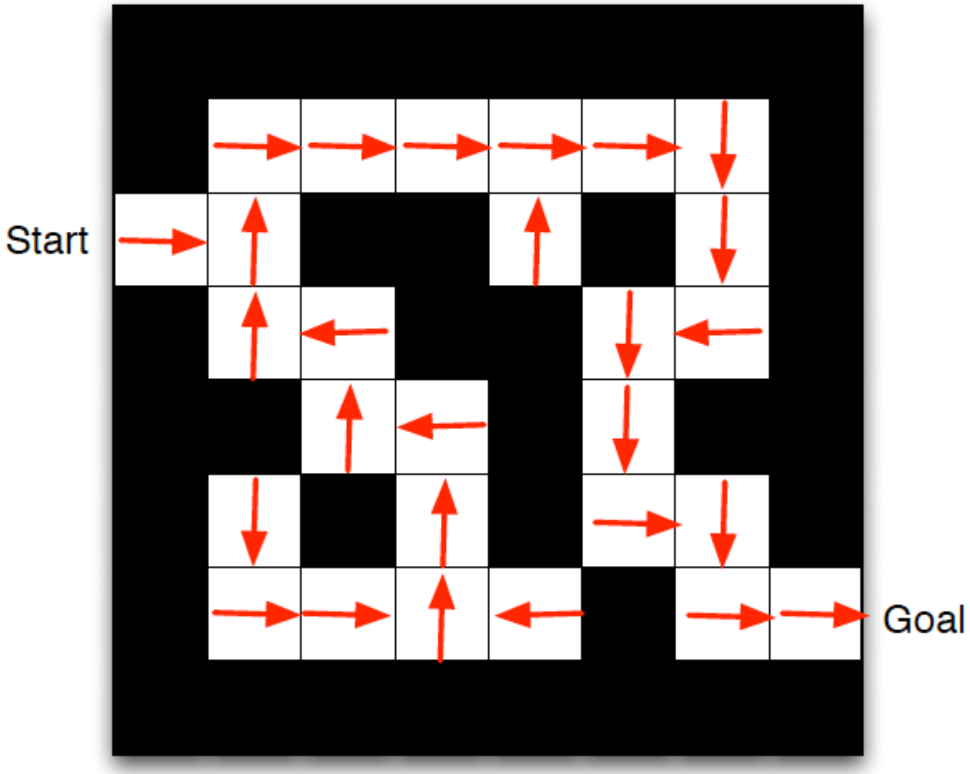
Some Notation



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Some Notation

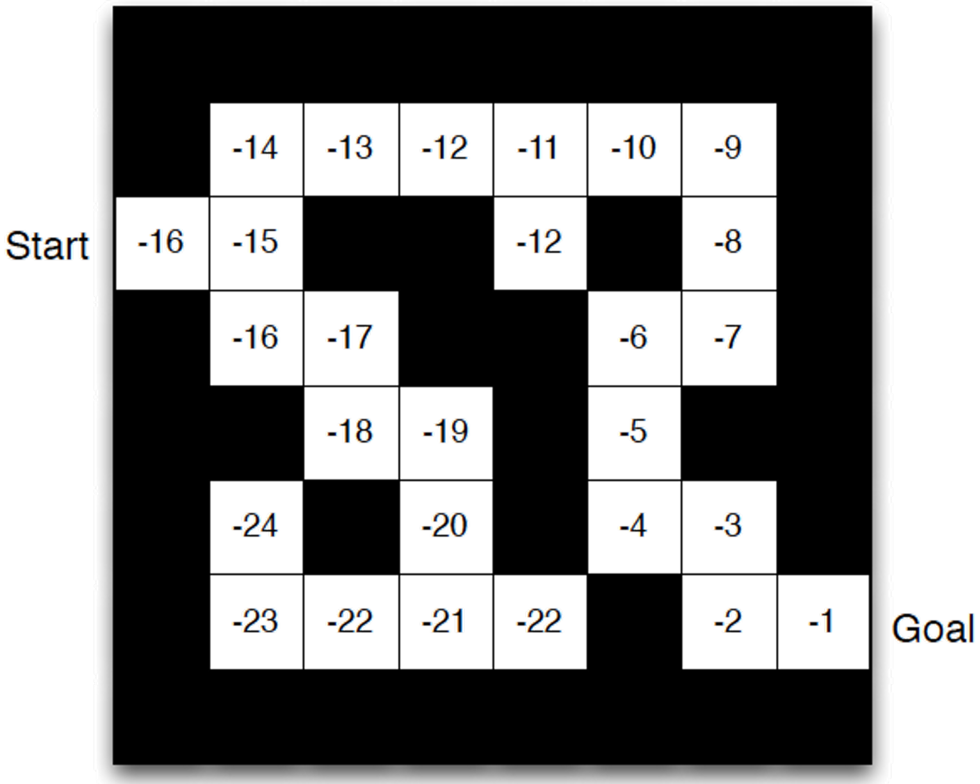
Policy



- Arrows represent policy $\pi(s)$ for each state s

Some Notation

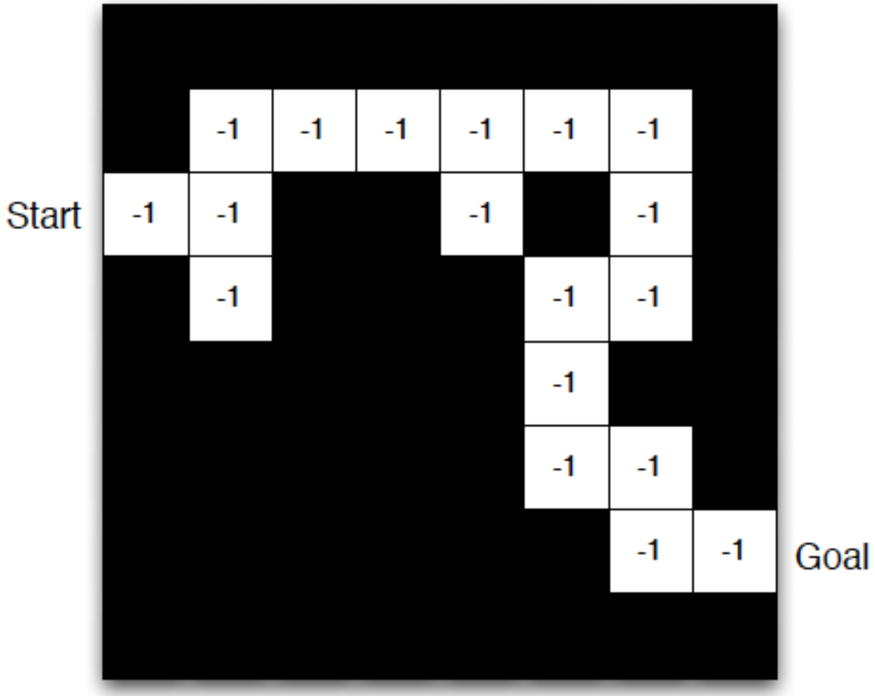
Value



- Numbers represent value $v_{\pi}(s)$ of each state s

Some Notation

Model



- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $\mathcal{P}_{ss'}^a$
- Numbers represent immediate reward $\mathcal{R}_s^a$ from each state s (same for all a)

Getting a Handle on These Definitions

1. If we have a policy π and know the true $Q_\pi(s, a)$ -- can we derive a new policy π' that is as good or better than π?

Recall that $Q_\pi(s, a)$ is the expected reward of taking action a in state s

Set $\pi(a'|s) = 1$ if $a' = argmax\ Q_\pi(s, a)$

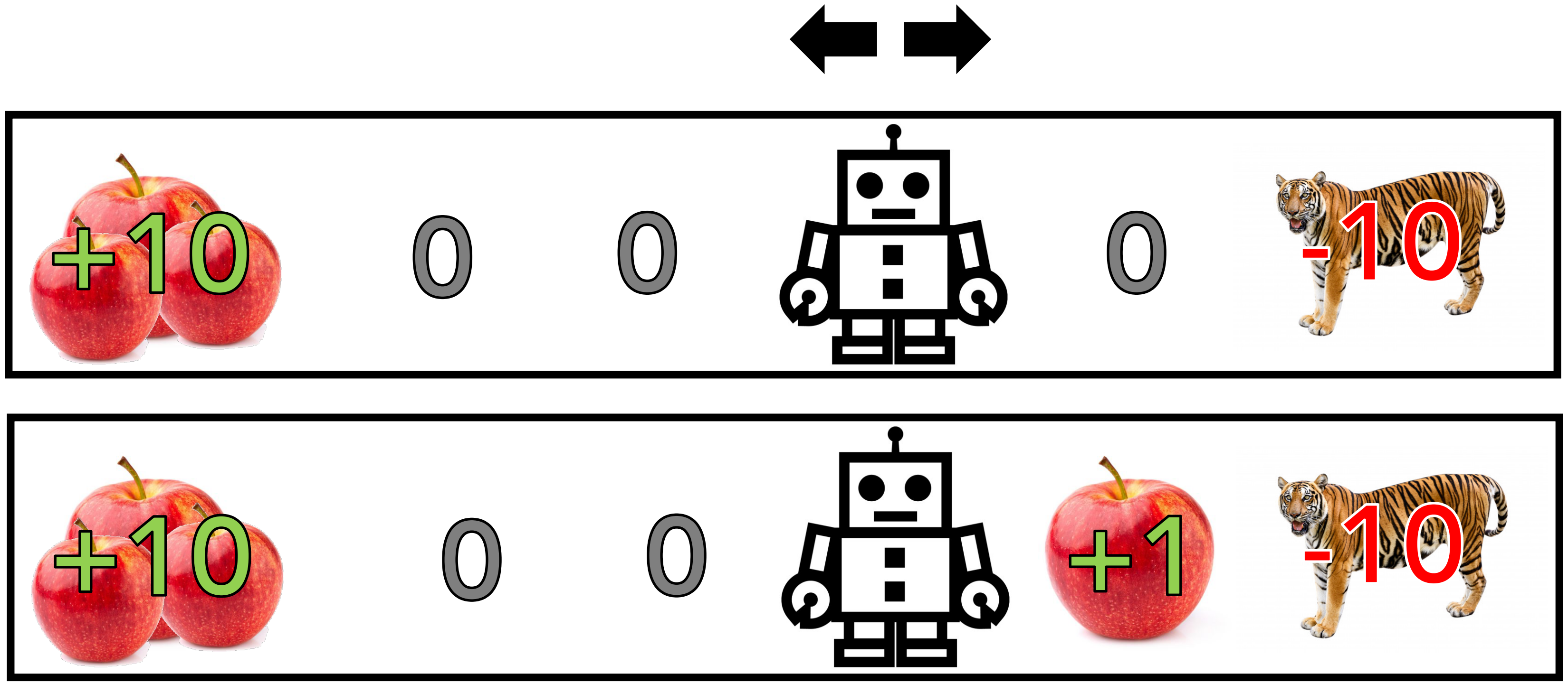
2. Fill in a simple algorithm to improve a policy:

Increase the probability $\pi(a|s)$ if ___ $r(s, a) > 0$? ___.

Decrease the probability $\pi(a|s)$ if ___ $r(s, a) < 0$? ___.

Slide Credit: Sergey Levine

Getting a Handle on These Definitions



Doesn't matter if you got one apple if you got eaten by a tiger

Immediate reward is not a particularly useful signal in many task settings.

Getting a Handle on These Definitions

1. If we have a policy π and know the true $Q_\pi(s, a)$ -- can we derive a new policy π' that is as good or better than π?

Recall that $Q_\pi(s, a)$ is the expected reward of taking action a in state s

Set $\pi(a'|s) = 1$ if a' $= argmax\ Q_\pi(s, a)$

2. Fill in a simple algorithm to improve a policy:

Increase the probability $\pi(a|s)$ if $\underline{\quad Q_\pi(s, a) > V_\pi(s) \quad}$.

Decrease the probability $\pi(a|s)$ if $\underline{\quad Q_\pi(s, a) < V_\pi(s) \quad}$.

Recall that $V_\pi(s)$ is the expected reward of following π from state s

Getting a Handle on These Definitions

1. If we have a policy π and know the true $Q_\pi(s,a)$ -- can we derive a new policy π' that is as good or better than π?

Recall that $Q_\pi(s,a)$ is the expected reward of taking action a in state s

Set $\pi(a'|s) = 1$ if $a' = argmax\, Q_\pi(s,a)$

2. Fill in a simple algorithm to improve a policy:

Increase the probability $\pi(a|s)$ if $\underline{\quad A_\pi(s,a) > 0 \quad}$.

Decrease the probability $\pi(a|s)$ if $\underline{\quad A_\pi(s,a) < 0 \quad}$.

Recall $A_\pi(s_t,a) = Q_\pi(s_t,a) - V_\pi(s_t)$

Slide Credit: Sergey Levine

Getting a Handle on These Definitions

3. Given an accurate deterministic world model $s_{t+1} = M(s_t, a_t)$ and value function $V_\pi(s_t)$, how should an agent act in state s_t?

For each possible action a,
  Compute $V_\pi(s_{t+1})$ for $s_{t+1} = M(s_t, a)$
Select action with highest value.

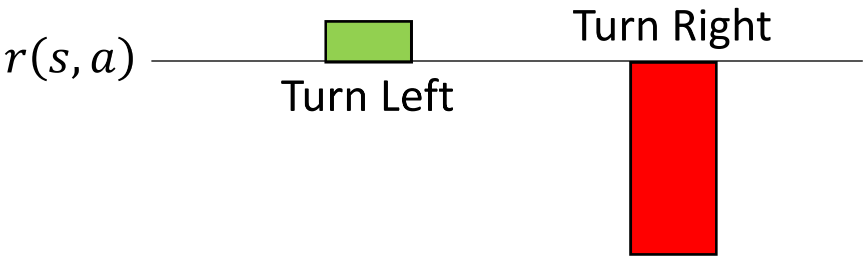
Relies on having a model of the transition probabilities

Slide Credit: Sergey Levine

Markov Decision Processes

Challenges of Markov Decision Processes

Reward is very often discontinuous



$r(s,a)$ – not smooth

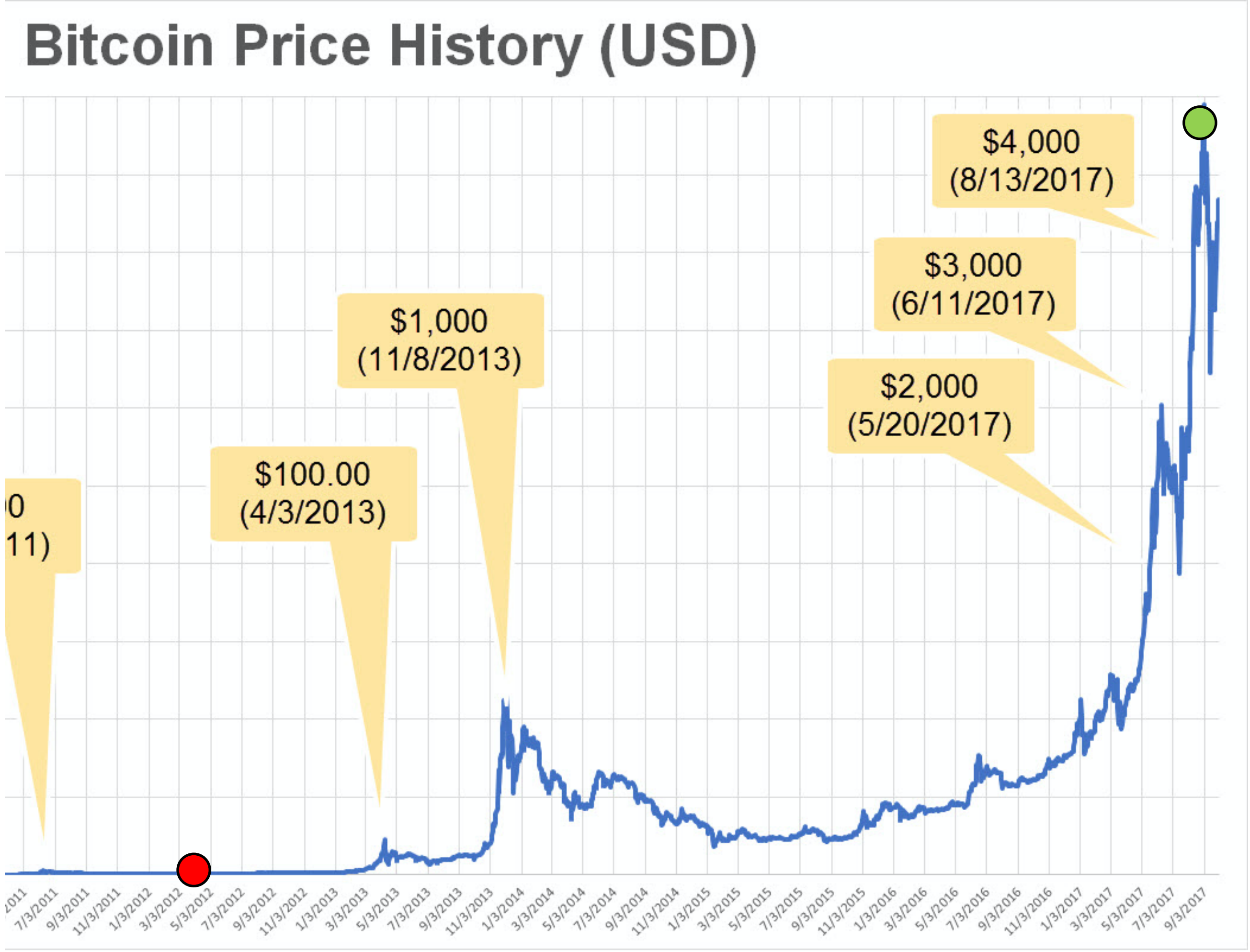
$\pi_\theta(a = turn\,right) = \theta$

$\mathbb{E}_{\pi_\theta}[r(s,a)]$ – smooth in θ

Slide Credit: Stefan Lee

Markov Decision Processes

Challenges of Markov Decision Processes



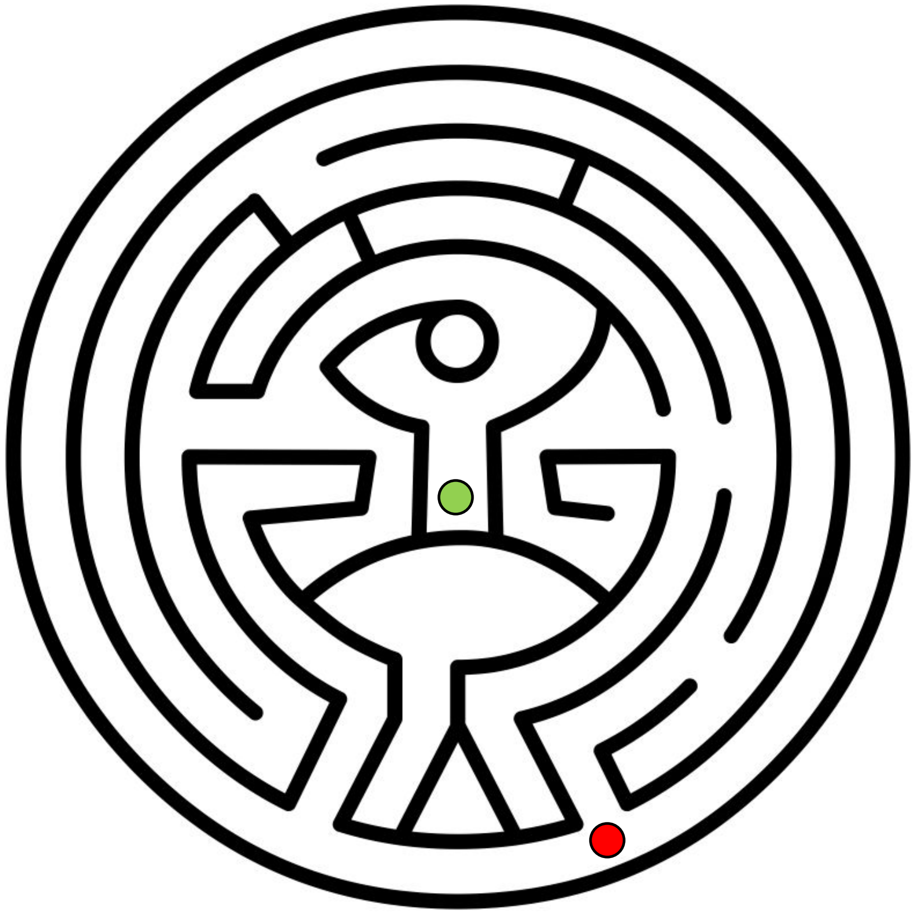
Reward is often sparse and delayed

Taking an action at time t
Doesn't pay off till time t+k

Slide Credit: Stefan Lee

Markov Decision Processes

Challenges of Markov Decision Processes



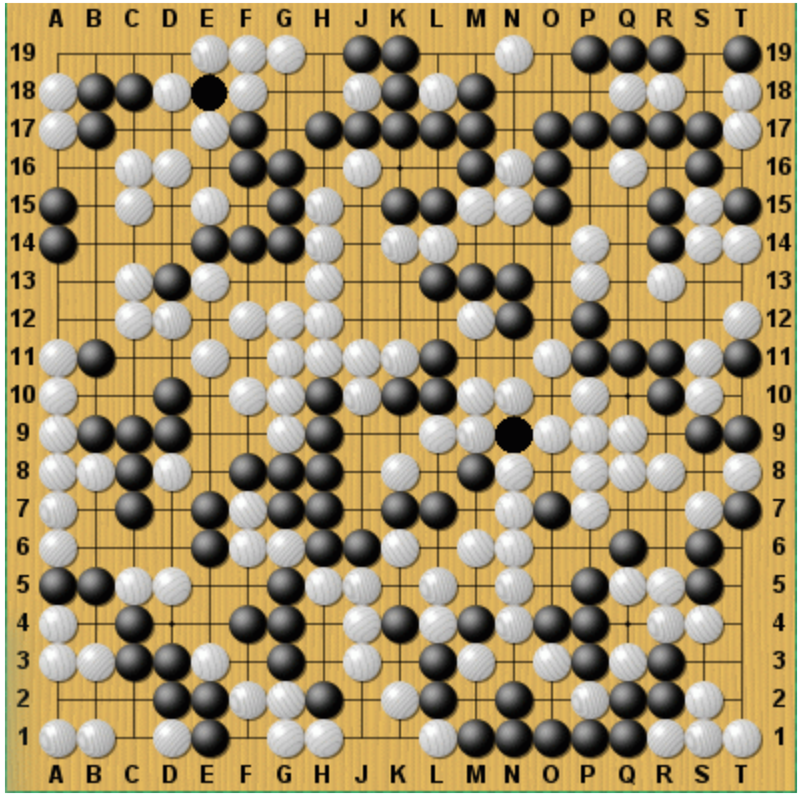
Reward is often sparse and delayed

Taking an action at time t
Doesn't pay off till time t+k

Markov Decision Processes

Challenges of Markov Decision Processes

2.08×10^{170} Legal Board Configurations

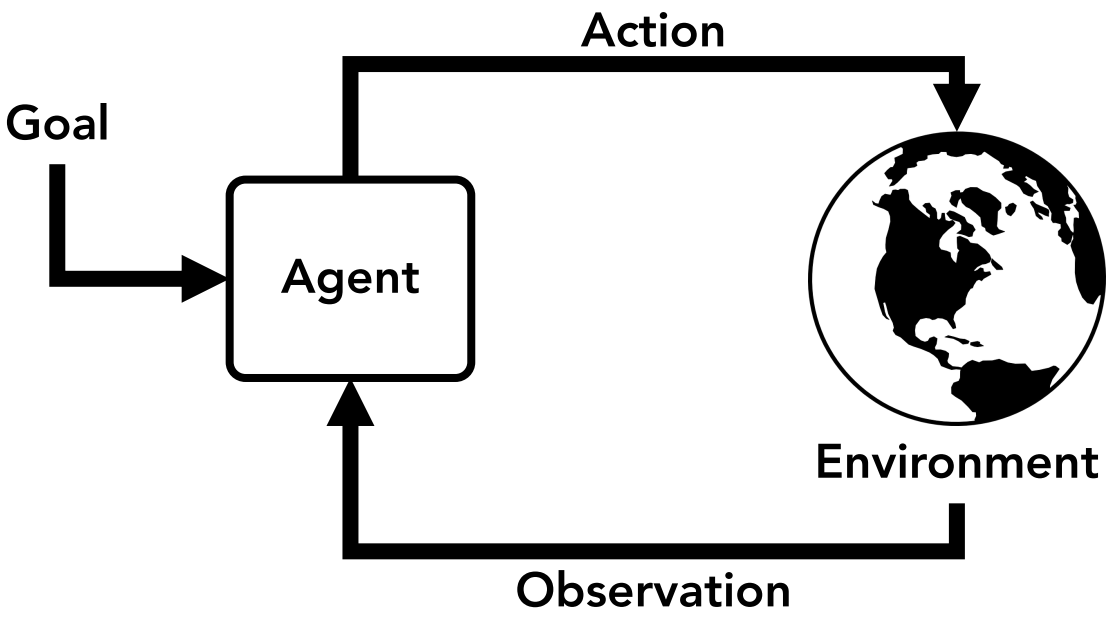


State and action spaces can be huge (or infinite)

A General Embodied Agent

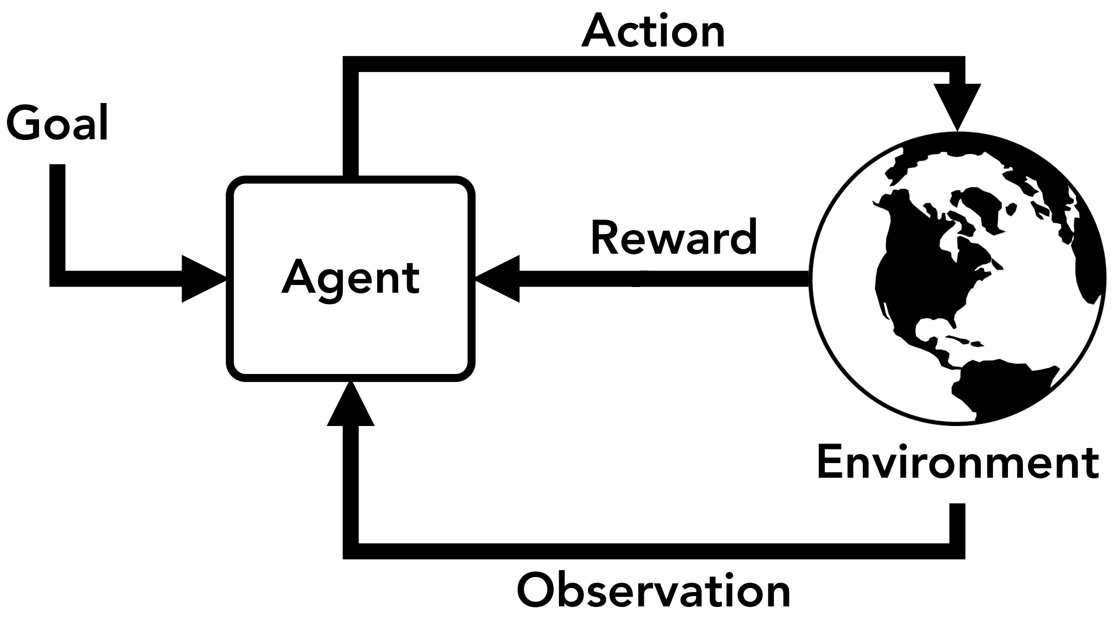
Imitation Learning

- Have expert demonstrations (possibly interactive)



Reinforcement Learning

- Environment provides feedback
- No examples of optimal policy



Slide Credit: Stefan Lee

Some Notation

Common Terminology and Definitions (for categorizing RL algorithms)

Model-free vs Model-based RL

- Do you know the world model of how actions affect state?

$$M\colon \mathcal{S} \times \mathcal{A} \to \Omega_{\mathcal{S}}$$

On-policy vs Off-policy

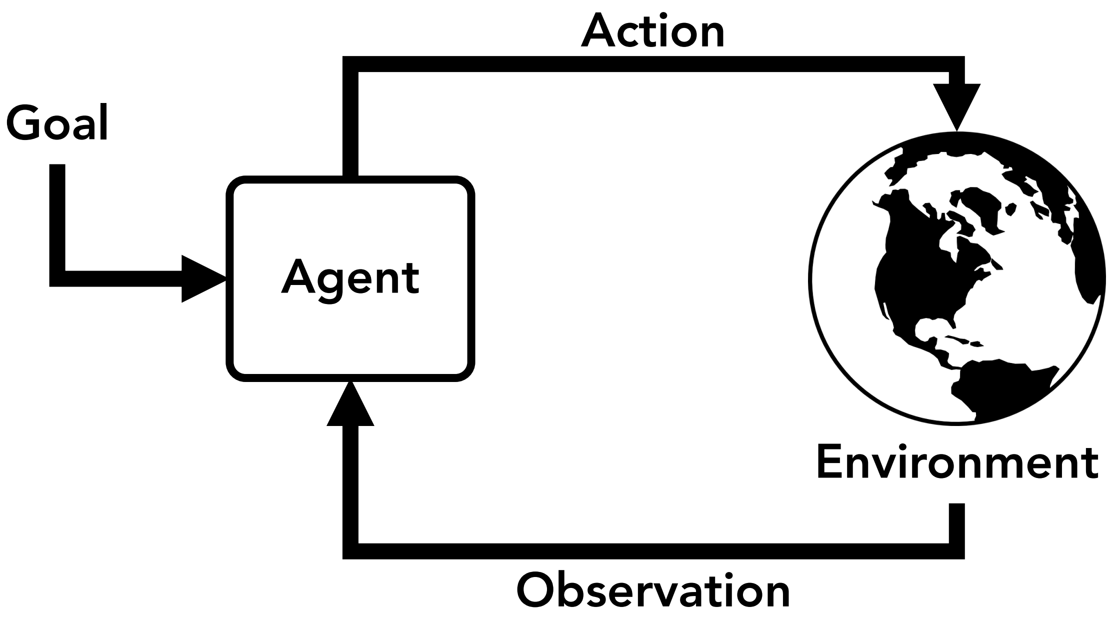
- On-policy: Use samples from the target policy for training
- Off-policy: Train on a distribution of trajectories

(set of interaction sequences / episodes) that comes from a different policy than the target policy

A General Embodied Agent

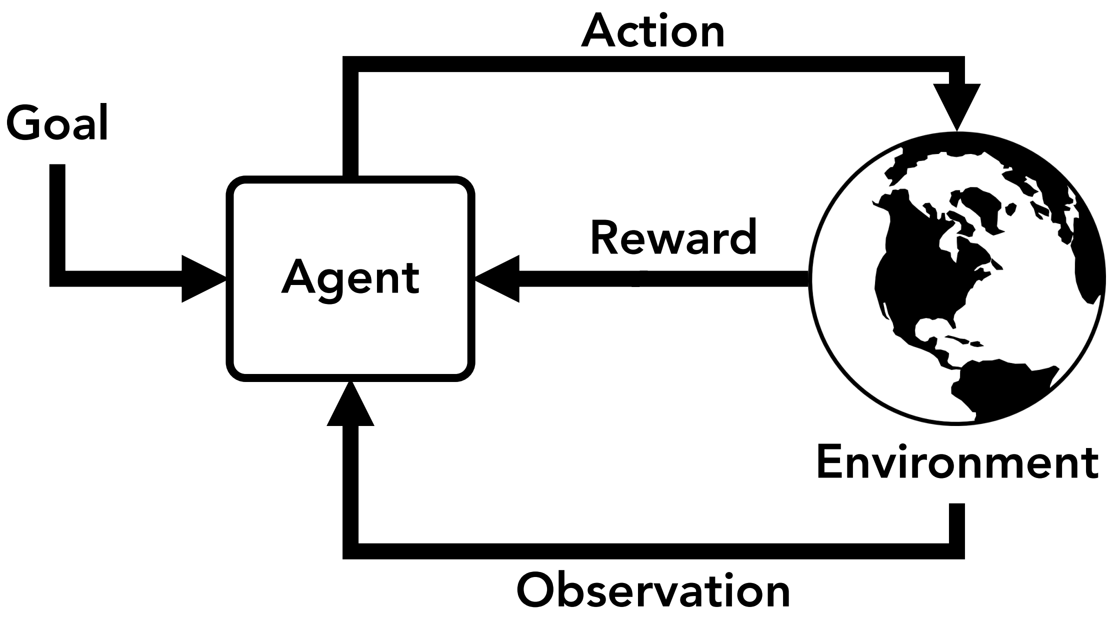
Imitation Learning

- Have expert demonstrations (possibly interactive)



Reinforcement Learning

- Environment provides feedback
- No examples of optimal policy



Slide Credit: Stefan Lee

Imitation Learning

Imitation Learning

Imitation Learning

- Assume access to an expert demonstrator $\boldsymbol{\pi}^*$ at some point or another and to varying levels of interactivity **Does not assume reward function is given!**

Imitation Learning

Imitation Learning

- Assume access to an expert demonstrator $\boldsymbol{\pi}^*$ at some point or another and to varying levels of interactivity
- Does not assume reward function is given!

- **Behavior Cloning / Inverse Reinforcement Learning**
  - Given dataset of expert trajectories $D = \{\ (s_0, a_0, s_1, a_1, \dots, s_T, a_T)_i\ \}_{i=1}^{N}$

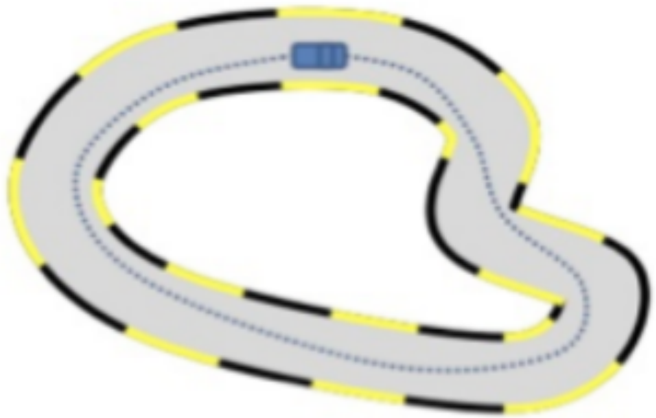
- **Direct Policy Learning / Interactive Expert**
  - Assume queryable expert $\boldsymbol{\pi}^*$ during training

Imitation Learning
Behavior Cloning

Imitation Learning – Behavior Cloning

Example #1: Racing Game

(Super Tux Kart)

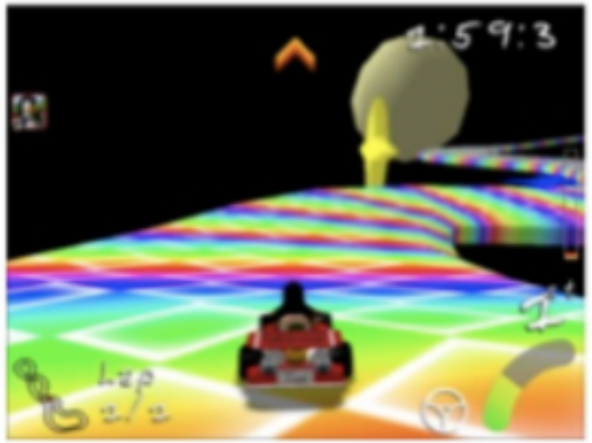


s = game screen

a = turning angle

Training set: D={$\boldsymbol{\tau}$:=($\mathbf{s}$,$\mathbf{a}$)} from π^*

- $\mathbf{s}$ = sequence of s
- $\mathbf{a}$ = sequence of a



Goal: learn $\pi_\theta(s) \rightarrow a$

Images from Stephane Ross

Slide Credit: Yisong Yue

Imitation Learning – Behavior Cloning

Behavior Cloning

- Given dataset of trajectories $D = \{ (s_0, a_0, s_1, a_1, \dots, s_T, a_T)_i \}_{i=1}^{N}$ from an expert demonstration policy π^*
- Break things down to individual state-action pairs s_t, a_t and directly train a policy $\hat{a}_t = \pi(s_t)$ using supervised learning:

$$\theta^* = argmin_\theta \mathbb{E}_{s \sim \pi^*}[\ L(\pi_\theta(s), \pi^*(s))\ ]$$

$$\theta^* = argmin_\theta \sum_i L(\pi_\theta(a_t|s_t), \pi^*(a_t|s_t))$$

- **Interpretations:**
  - Assuming perfect imitation so far, learn to continue imitating perfectly
  - Minimize 1-step deviation for states the expert visits

Imitation Learning – Behavior Cloning

Data Distribution Mis-match

	Supervised Learning	Behavior Cloning
Train	$(\boldsymbol{x}, \boldsymbol{y}) \sim \boldsymbol{D}$	$(\boldsymbol{s}, \boldsymbol{a}) \sim \boldsymbol{\pi}^*$
Test	$(\boldsymbol{x}, \boldsymbol{y}) \sim \boldsymbol{D}$	$(\boldsymbol{s}, \boldsymbol{a}) \sim \boldsymbol{\pi_\theta}$

Distributions of states the agent will encounter during test may differ from training!

Slide Credit: Katerina Fragkiadaki

Imitation Learning – Behavior Cloning

Behavior Cloning: Use set of demonstrations as targets for a supervised learning task while minimizing 1-step error

- Strengths:
  - Dead simple. Seriously. It is just supervised learning.
  - Works well when minimizing 1-step deviation is sufficient.

- Weaknesses:
  - Compounding errors.
  - Data distribution mis-match.

Slide Credit: Stefan Lee

Imitation Learning – Behavior Cloning

Data Distribution Mis-match

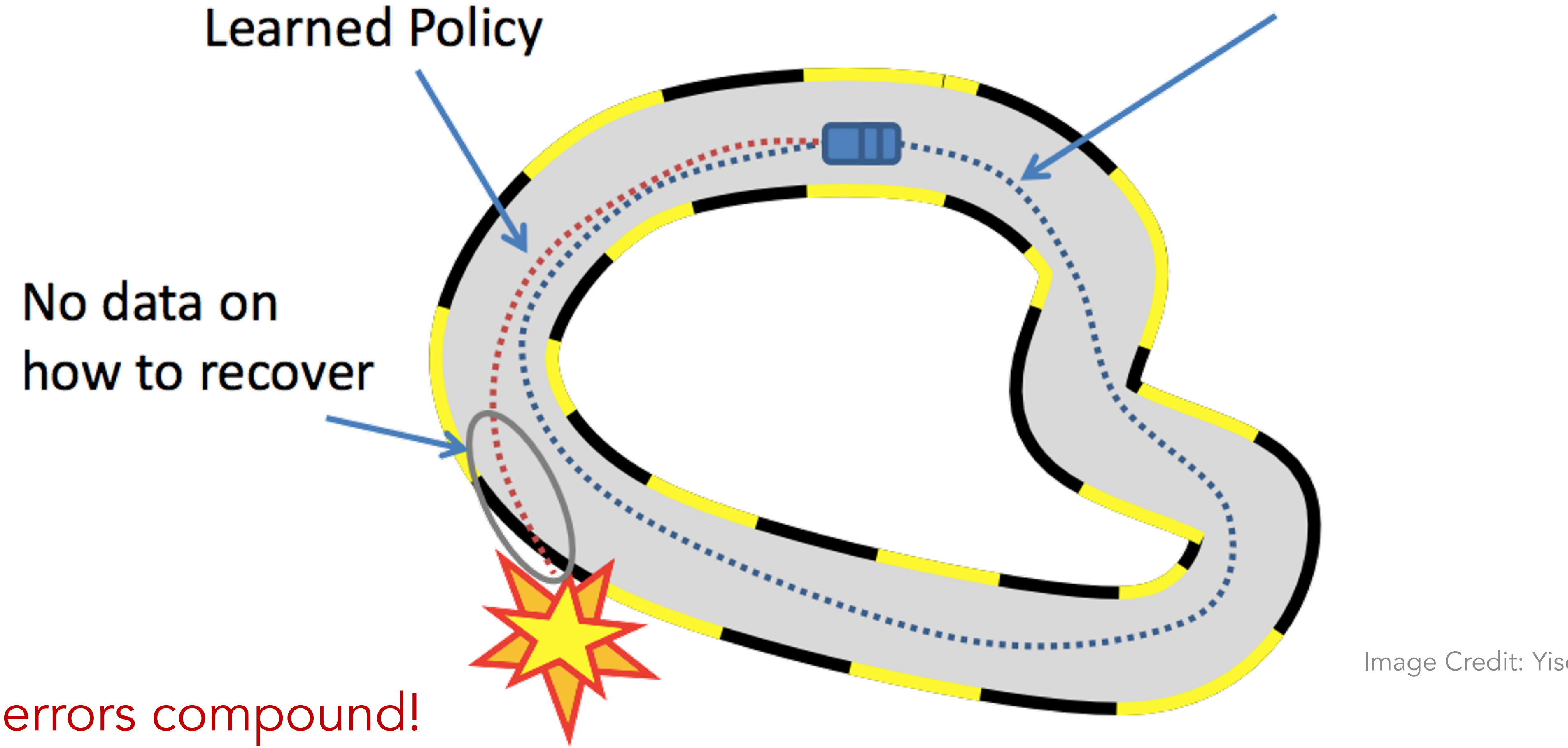


Image Credit: Yisong Yue

Worse: errors compound!

Suppose π_θ achieves an error rate of ϵ for states induced by π^*, then over a T length trajectory the expected number of errors is

$$E_\pi[mistakes] = O(T^2\epsilon)$$

Imitation Learning – Behavior Cloning

Behavior Cloning: Use set of demonstrations as targets for a supervised learning task while minimizing 1-step error

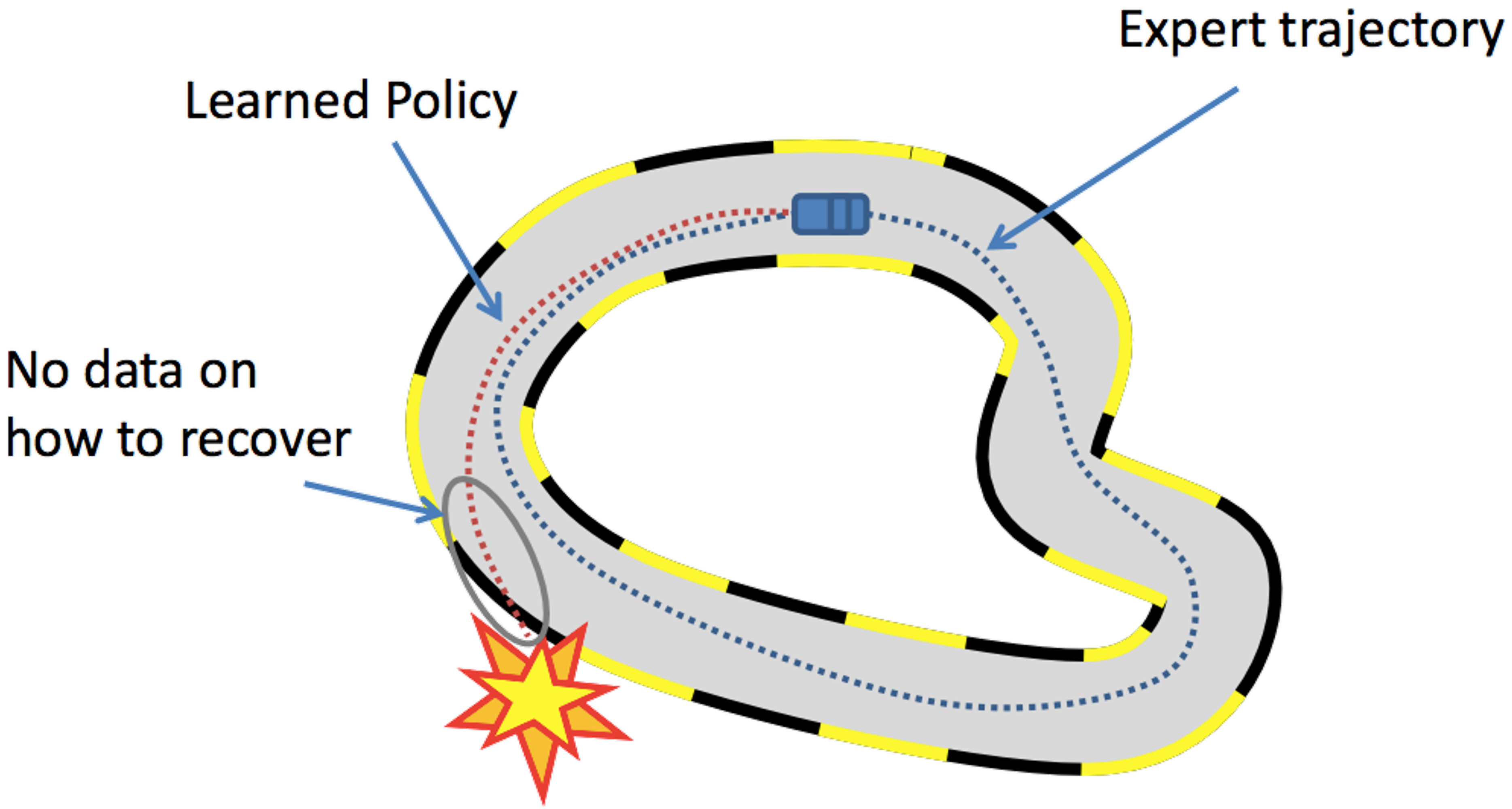
- **Strengths:**
  - Dead simple. Seriously. It is just supervised learning.
  - Works well when minimizing 1-step deviation is sufficient.
- **Weaknesses:**
  - Compounding errors.
  - Data distribution mis-match.
- **When to use this?**
  - When the state space is well-covered by the demonstrator.
  - When recovering from 1-step deviations is easy.
  - To pre-train before doing a full RL approach.

Slide Credit: Stefan Lee

Imitation Learning
Direct Policy Learning

Imitation Learning – Direct Policy Learning

Data Distribution Mis-match



Imitation Learning – Interactive Direct Policy Learning

Why is this a problem for Behavior Cloning?

$$\theta^* = argmin_\theta \mathbb{E}_{\boldsymbol{s\sim\pi^*}}[\ L(\pi_\theta(s), \pi^*(s))\ ]$$

Train a policy that behaves the same in states the demonstrations visit.

What if we had a demonstration policy we could query?

$$\theta^* = argmin_\theta \mathbb{E}_{\boldsymbol{s\sim\pi_\theta}}[\ L(\pi_\theta(s), \pi^*(s))\ ]$$

Removes state mis-match, but requires us to evaluate $\pi^*(s)$ for arbitrary states.

Imitation Learning – Interactive Direct Policy Learning

$$\theta^* = argmin_\theta \mathbb{E}_{s\sim\pi_\theta}[\ L(\pi_\theta(s), \pi^*(s))\ ]$$

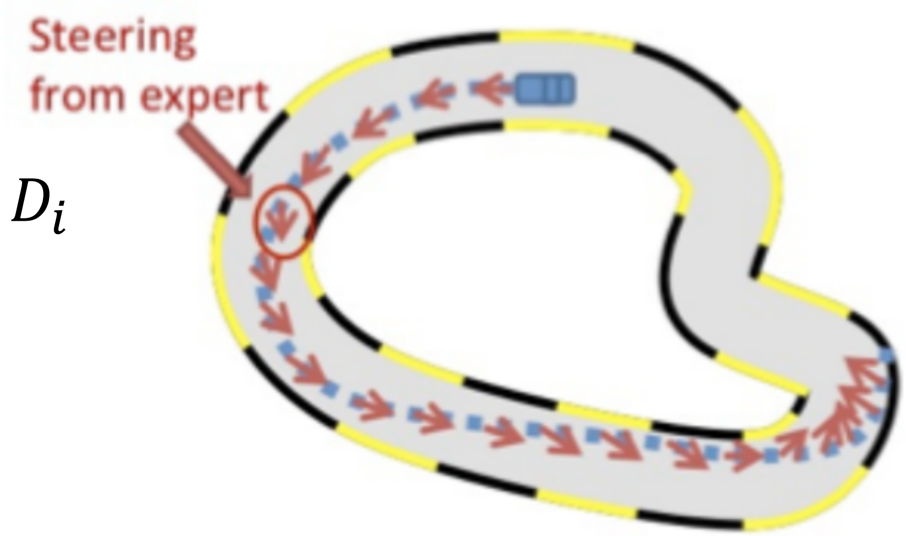
A Naïve Algorithm

Estimate policy π_θ parameters

Train a policy π_θ^i using behavior cloning D_i

Estimate state space $\mathbf{s} \sim \boldsymbol{\pi_\theta}$ and collect demonstrations

Rollout π_θ^i to generate a set of states, query π^* to generate a new dataset D_{i+1}



Not guaranteed to converge / might oscillate.

Imitation Learning – Interactive Direct Policy Learning

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Collect Data

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

Slide Credit: Stefan Lee

Imitation Learning – Interactive Direct Policy Learning

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
Sample T-step trajectories using π_i.
Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Behavior Cloning

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

Slide Credit: Stefan Lee

Imitation Learning – Interactive Direct Policy Learning

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Not an actual convex combination.
Expert chooses controls with probability β_i

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

Slide Credit: Stefan Lee

Imitation Learning – Interactive Direct Policy Learning

DAGGER is a Dataset Aggregation based approach.

Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.

Alternative approaches do Policy Aggregation

$$\pi^i = (1-\alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} \hat{\pi}^{*j}.$$

SMILe from Efficient Reductions for Imitation Learning, 2010
SEARN from Search-based Structured Prediction, 2009

Imitation Learning – Interactive Direct Policy Learning

Interactive Direct Policy Learning

Iteratively perform behavior cloning and then query an expert demonstrator to label newly entered states.

- **When to use this?**
  - When querying the expert is cheap!
    - Why not just use that expert? Some cases "expert actions" are easy to compute, but their relation to the observed state may not be.
  - When executing a possibly bad policy is safe.

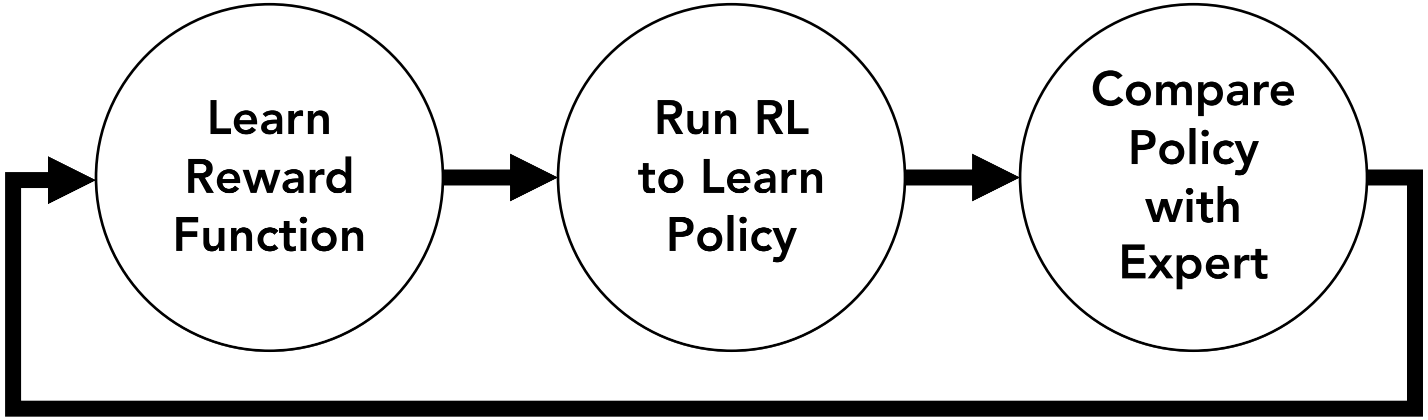
Imitation Learning
Inverse Reinforcement Learning

Imitation Learning – Inverse Reinforcement Learning

Inverse Reinforcement Learning

Given dataset of trajectories $D = \{ (s_0, a_0, s_1, a_1, \dots, s_T, a_T)_i \}_{i=1}^{N}$ from an expert policy π^*, find a reward function $r(s, a)$ such that:

$$\pi^* = argmax_\pi \, \mathbb{E}_{s,a \sim \pi_\theta}[r(s, a)]$$

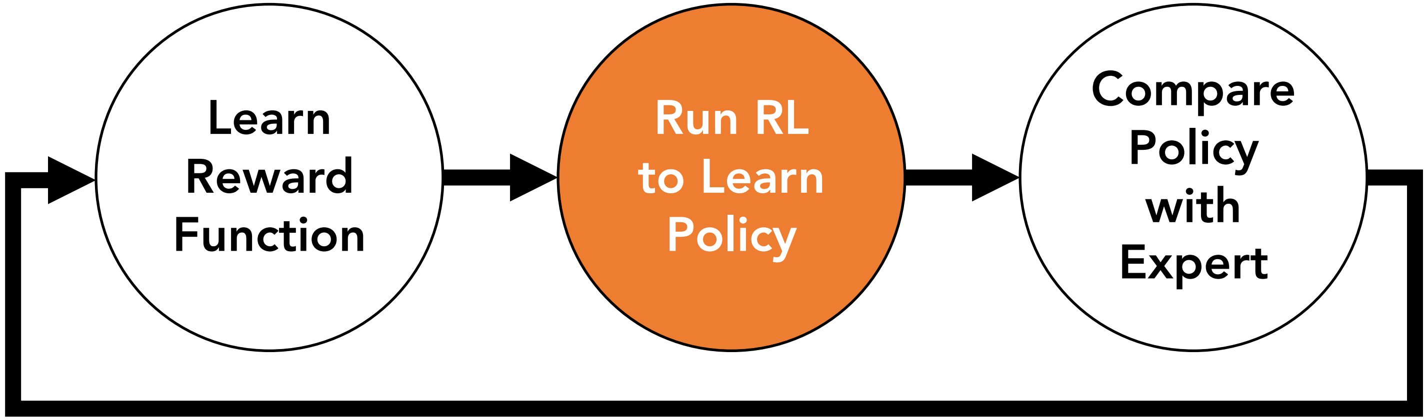


Imitation Learning – Inverse Reinforcement Learning

Inverse Reinforcement Learning

Given dataset of trajectories $D = \{ (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)_i \}_{i=1}^{N}$ from an expert policy π^*, find a reward function $r(s, a)$ such that:

$$\pi^* = argmax_\pi \, \mathbb{E}_{s,a \sim \pi_\theta}[r(s, a)]$$



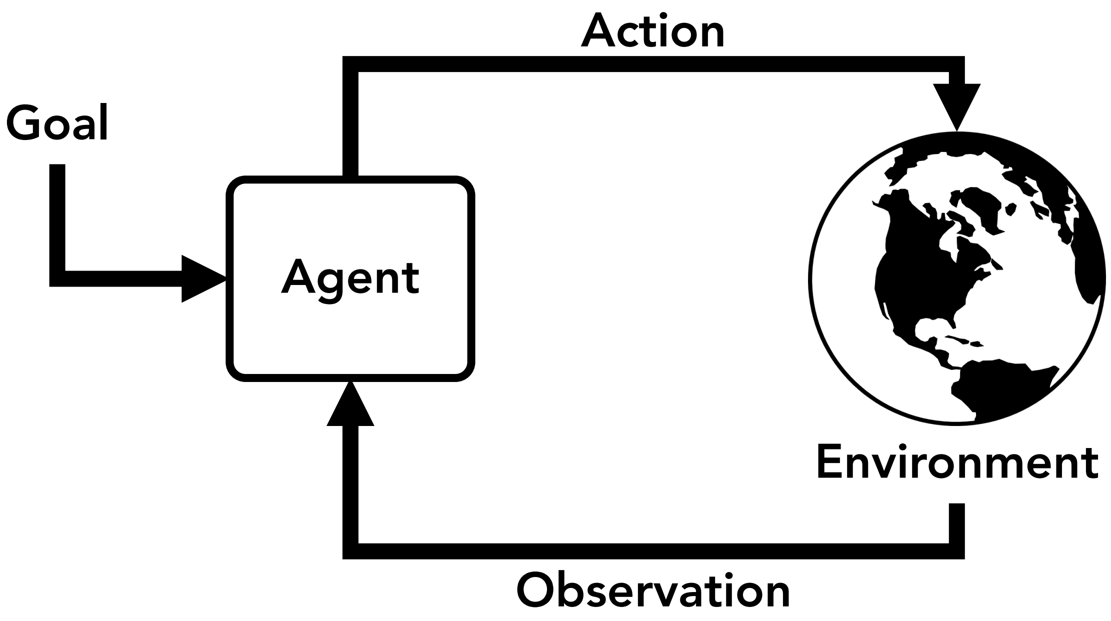
Slide Credit: Stefan Lee

Reinforcement Learning

A General Embodied Agent

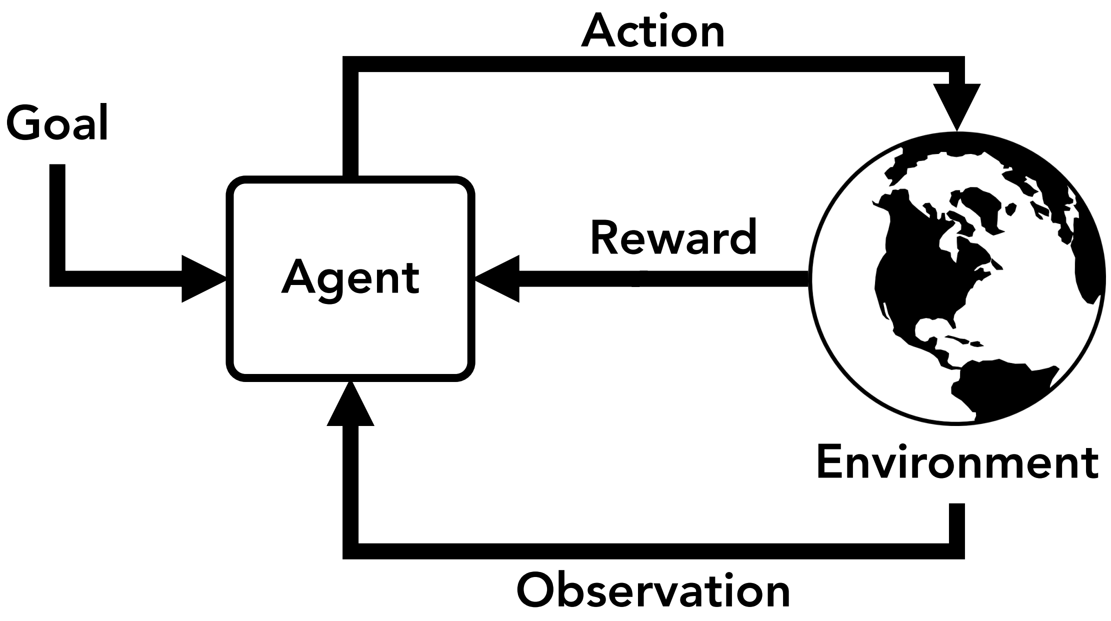
Imitation Learning

- Have expert demonstrations (possibly interactive)



Reinforcement Learning

- Environment provides feedback
- No examples of optimal policy



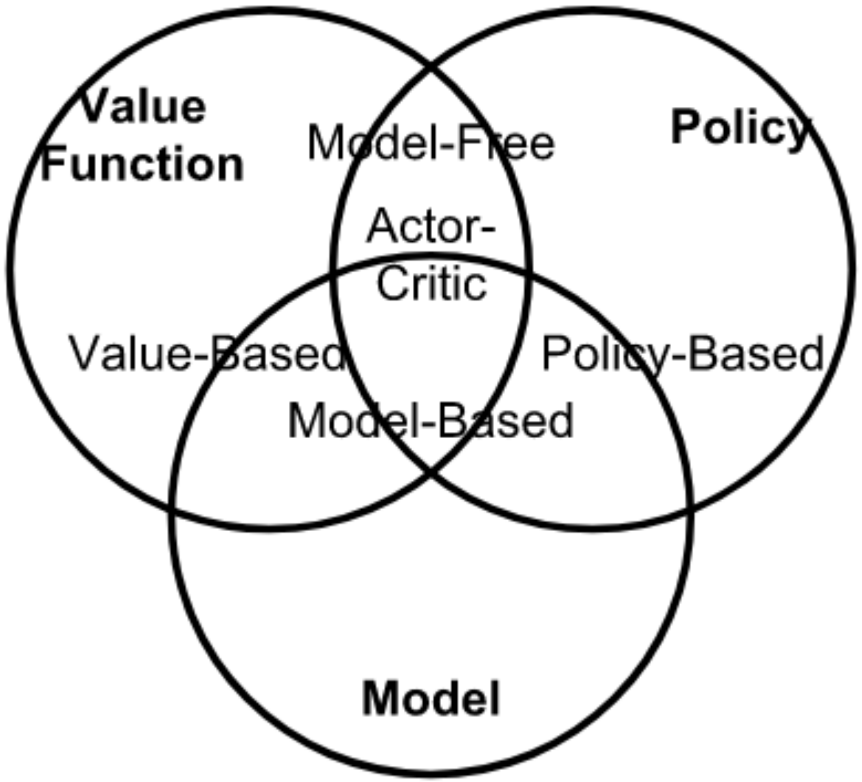
Reinforcement Learning

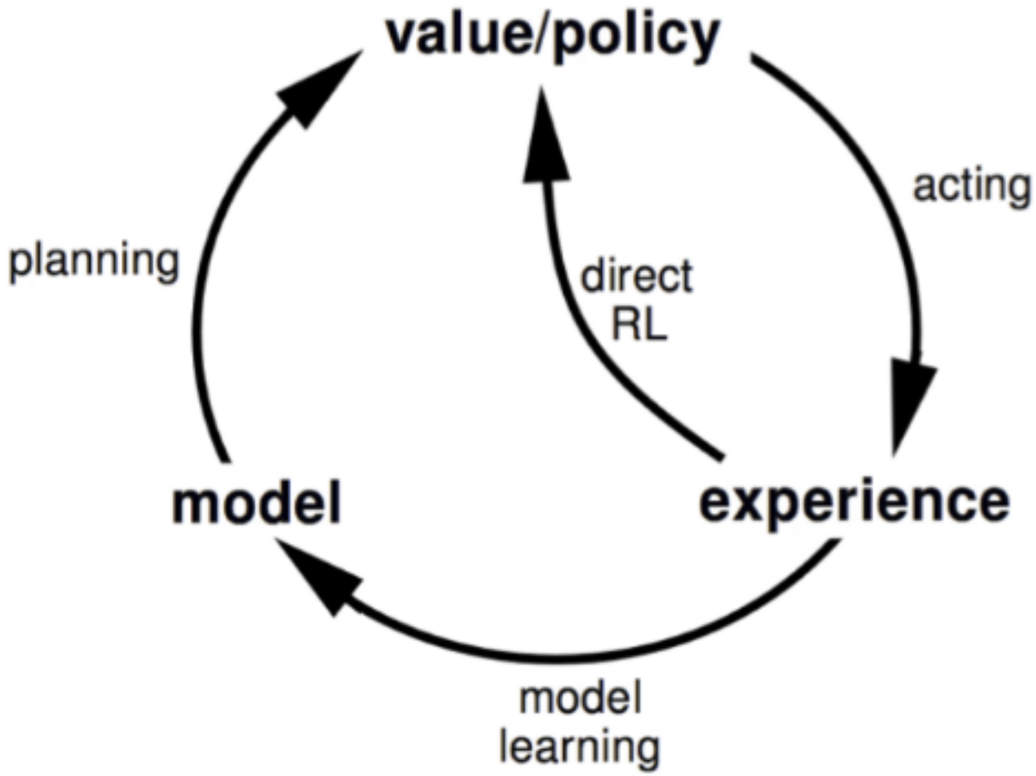
Approaches to Reinforcement Learning

- **Policy-based RL**
  - Search directly for the optimal policy π^*
- **Value-based RL**
  - Estimate the optimal action-value function $Q^*(s, a)$
    - Under some fixed policy (e.g. epsilon-greedy)
- **Model-based RL**
  - Build a model of the world
    - State transition, reward probabilities
  - Plan (e.g. by look-ahead) using model

Taxonomy

Model-Free RL: Don't know how our action will affect the state





Model-Based RL: Need to build a model of how our action will affect the state

Figure Credit: David Silver

Reinforcement Learning

Deep Reinforcement Learning

- **Policy-based RL**
  - Learn a policy network $\pi(s;\theta^*) \approx \pi^*(s)$ parameterized by θ

- **Value-based RL**
  - Learn a network $Q(s,a;\theta^*) \approx Q^*(s,a)$ parameterized by θ
    - Under some fixed policy (e.g. epsilon-greedy)

- **Model-based RL**
  - Learn a transition function $M(s;\theta^*) \approx \mathbb{P}(s)$
  - Plan (e.g. by look-ahead) using model

Slide Credit: Dhruv Batra

Policy-Based RL

REINFORCE

Policy-Based Reinforcement Learning

Goal: Learn a policy network $\pi(s;\theta^*)$ such that:

$$\theta^* = \text{argmax}_\theta \, \mathbb{E}_{\pi_\theta,\mathbb{P}}\left[\sum_i \gamma^{i-1} r(s_i, a_i)\right]$$

$$\underbrace{\mathbb{E}_{\pi_\theta,\mathbb{P}}\left[\sum_i \gamma^{i-1} r(s_i, a_i)\right]}_{J(\theta)}$$

How to optimize θ to maximize $J(\theta)$?

Gradient Ascent! $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

Slide Credit: Dhruv Batra

Policy Gradient Methods

Model and optimize the policy directly

Let's write $$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$$

$$= \int_\tau p(\tau;\theta) r(\tau)\, d\tau$$

Where $r(\tau)$ is the reward of trajectory $\tau = (\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1, \mathrm{a}_1, \mathrm{s}_2, \ldots)$

$$p(\tau;\theta) = \prod p(s_{t+1}|s_t, a_t)\, \pi_\theta(a_t, s_t)$$

Slide Credit: Dhruv Batra

Policy Gradient Methods

Expected rewards of policy $\pi(\cdot \mid \cdot\,; \theta)$:

$$J(\theta) = \int_{\tau} p(\tau; \theta) r(\tau) d\tau$$

Let's differentiate with respect to θ:

$$\nabla_\theta J(\theta) = \int_{\tau} \underbrace{\nabla_\theta p(\tau; \theta) r(\tau) d\tau}_{\text{Intractable}}$$

REINFORCE Algorithm (Williams, 1992)

Let's differentiate with respect to θ: $\nabla_\theta J(\theta) = \int_\tau \nabla_\theta p(\tau;\theta)\, r(\tau)\ d\tau$

A useful identity/trick:

$$\underbrace{\frac{p(\tau;\theta)}{p(\tau;\theta)}}_{1} \nabla_\theta p(\tau;\theta) = p(\tau;\theta)\,\frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta)\,\nabla_\theta \log p(\tau;\theta)$$

REINFORCE Algorithm (Williams, 1992)

Let's differentiate with respect to θ:

$$\nabla_\theta J(\theta) = \int_\tau \nabla_\theta p(\tau;\theta)\, r(\tau)\ d\tau$$

$$\nabla_\theta J(\theta) = \int_\tau p(\tau;\theta)\ \nabla_\theta \log p(\tau;\theta)\, r(\tau)\, d\tau$$

REINFORCE Algorithm (Williams, 1992)

Let's differentiate with respect to θ: $$\nabla_\theta J(\theta) = \int_\tau \nabla_\theta p(\tau;\theta)\, r(\tau)\ d\tau$$

$$\nabla_\theta J(\theta) = \int_\tau p(\tau;\theta)\ \nabla_\theta \log p(\tau;\theta)\, r(\tau)\, d\tau$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)\ \nabla_\theta \log p(\tau;\theta)]$$

REINFORCE Algorithm (Williams, 1992)

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)]$$

Transformed the **gradient of an expectation** into the **expectation of a gradient**

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)\ \nabla_\theta \log p(\tau;\theta)]$$

Slide Credit: Stefan Lee

REINFORCE Algorithm (Williams, 1992)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)\ \nabla_\theta \log p(\tau;\theta)]$$

Computing $\nabla_\theta \log p(\tau;\theta)$:

$$p(\tau;\theta) = \prod p(s_{t+1}|s_t, a_t)\,\pi_\theta(a_t, s_t)$$

$$\log p(\tau;\theta) = \sum \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t, s_t)$$

does not depend on θ

➡ $$\nabla_\theta \log p(\tau;\theta) = \sum \nabla_\theta \log \pi_\theta(a_t, s_t)$$

No model needed! Model-free RL.

Slide Credit: Stefan Lee

REINFORCE Algorithm (Williams, 1992)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{s_i, a_i \in \tau} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau)\right]$$

Monte Carlo Approximation:

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$$

sample some trajectories

Slide Credit: Stefan Lee

REINFORCE Algorithm (Williams, 1992)

Intuition:

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$$

- If trajectory reward is positive, push up the probabilities of the action
- If trajectory reward is negative, push down the probabilities of the action

All actions in trajectory move in same direction based on reward?!?

I know it seems too simple but it averages out.

REINFORCE Algorithm (Williams, 1992)

REINFORCE Algorithm (Williams, 1992)

While not converged:

1. Perform rollout to collect trajectory $\tau = (s_0, \mathrm{a}_0, s_1, \mathrm{a}_1, s_2, \dots)$ and reward $\mathrm{r}(\tau)$
2. Compute gradient estimate $\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$
3. Update policy parameters $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

REINFORCE In Action

Pong from Pixels

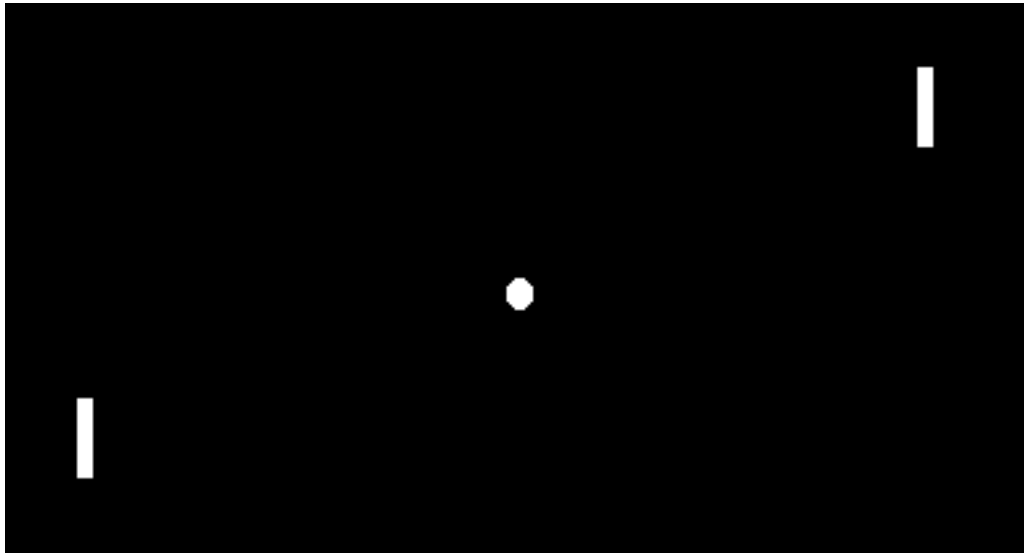


Image Credit: http://karpathy.github.io/2016/05/31/rl/

REINFORCE In Action

Pong from Pixels

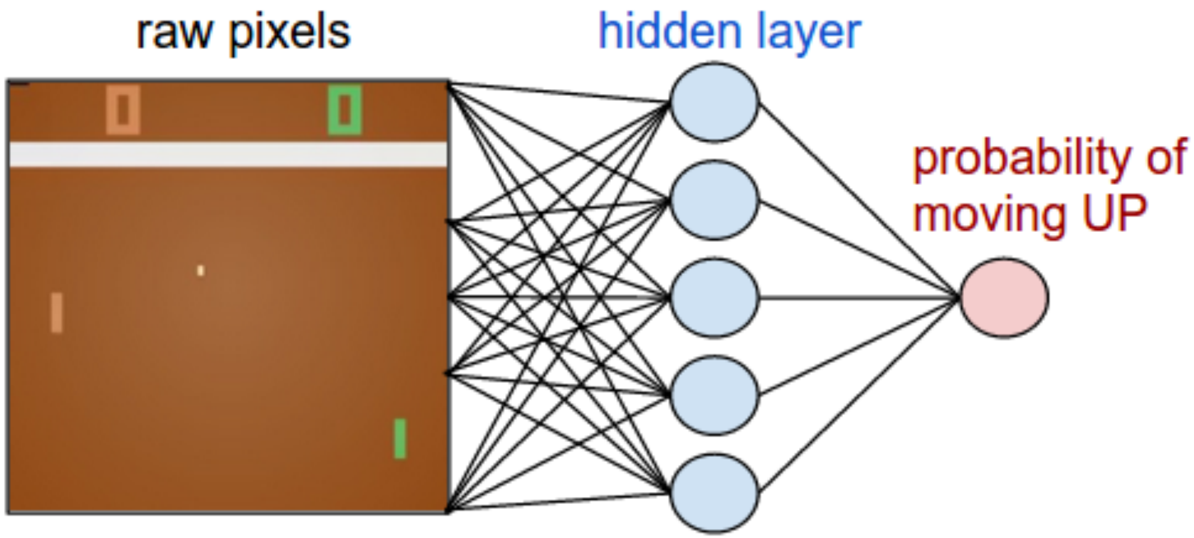


Image Credit: http://karpathy.github.io/2016/05/31/rl/

REINFORCE In Action

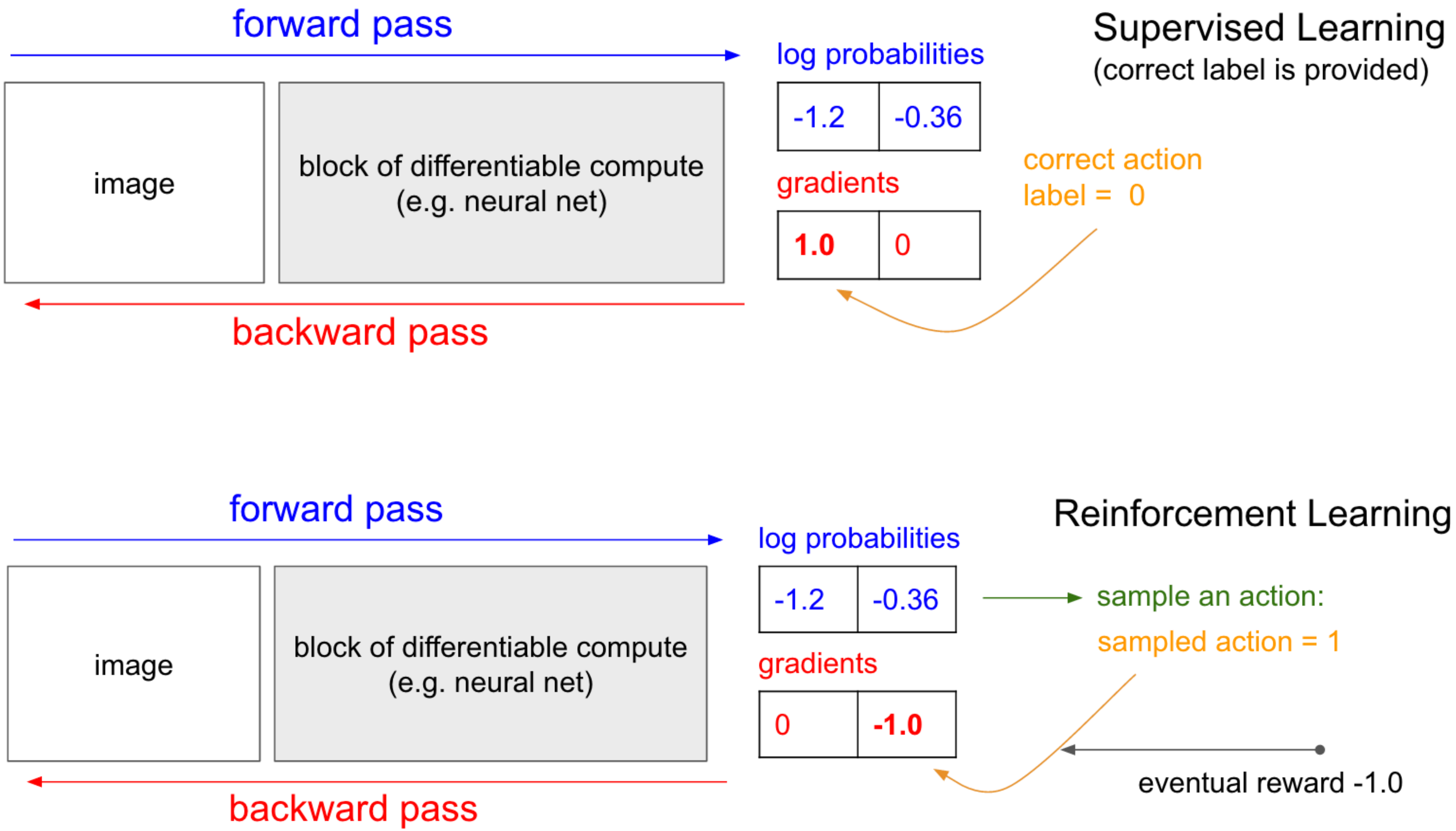


Image Credit: http://karpathy.github.io/2016/05/31/rl/

What's wrong with policy gradients?

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$$

High variance! Trajectories are long samples.
Rewards are often sparse and for the whole trajectory.

Slide Credit: Stefan Lee

Reducing Variance

Causality:

Policy at time t' can't affect rewards at time $t < t'$

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$$

⬇

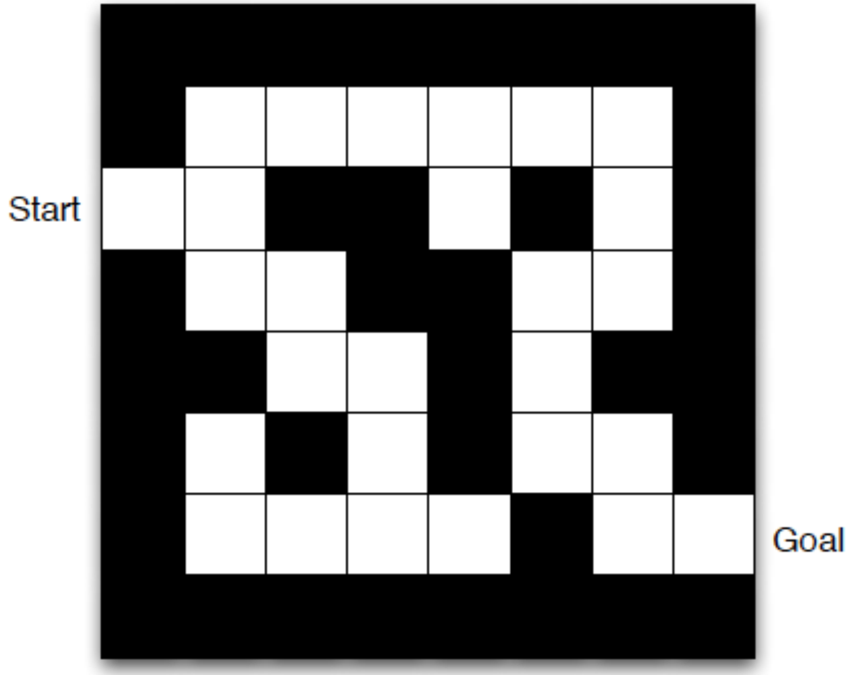
$$\nabla_\theta J'(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta) \left(\sum_{t=i} \gamma^{i-t} r(s_t, a_t) \right)$$

Slide Credit: Sergey Levine

Reducing Variance

Baselines:

What happens if the reward of "good samples" is negative?



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Reducing Variance

Baselines:

What if the variance in reward is huge?

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$$

-0.01

1000

Reducing Variance

Baselines:

Baseline

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta) \, [r(\tau_n) - b]$$

Average reward: $b = \frac{1}{N} \sum r(\tau)$

Are we allowed to do this? Still solving the same problem?

$$E[\nabla_\theta \log \pi_\theta(\tau) b] = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) b \, d\tau = \int \nabla_\theta \pi_\theta(\tau) b \, d\tau = b \nabla_\theta \int \pi_\theta(\tau) d\tau = b \nabla_\theta 1 = 0$$

Unbiased in Expectation!

Slide Credit: Sergey Levine

Policy Gradient Methods

REINFORCE Recap:

Simple algorithm that formalizes the notion ”repeat actions that lead to high rewards, avoid actions that lead to low rewards”. The approach is model-free. Big problems are with variance of the estimate, applying causality and baselines can help.

- **When to use this?**
  - When reward functions are well-defined and simulation is cheap.
  - If you don’t have time to implement the next thing we will talk about.

Policy-Based RL

Actor-Critic Methods

REINFORCE Algorithm (Williams, 1992)

REINFORCE Algorithm

(Williams, 1992)

While not converged:

1. Perform rollout to collect trajectory $\tau = (s_0, a_0, s_1, a_1, s_2, \dots)$ and reward $r(\tau)$
2. Compute gradient estimate $\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, r(\tau_n)$
3. Update policy parameters $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

Slide Credit: Stefan Lee

Actor-Critic Methods

Causality:

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta) \underbrace{\left(\sum_{t=i} \gamma^{i-t} r(s_t, a_t) \right)}_{\text{"reward from here"}}$$

Better estimate of expected rewards from a state-action pair?

$$Q_\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[V_\pi(s_{t+1})]$$

Actor-Critic Methods

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta) \left(Q_\pi(s_t, a_t) \right)$$

What about a baseline?

$$V_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{i=t} \gamma^{i-t} \, r(s_i, a_i) \right]$$

Advantage: $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$

How much better than average is this action?

Slide Credit: Stefan Lee

Actor-Critic Methods

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta) \left(A_\pi(s_t, a_t) \right)$$

$$A_\pi(s_t, a_t) = \overbrace{r(s_t, a_t) + \gamma V_\pi(s_{t+1})}^{Q_\pi(s_t, a_t)} - V_\pi(s_t)$$

$$A_\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[V_\pi(s_{t+1})] - V_\pi(s_t)$$

Just need to estimate the value function!
Let's throw a neural network at it!

Slide Credit: Stefan Lee

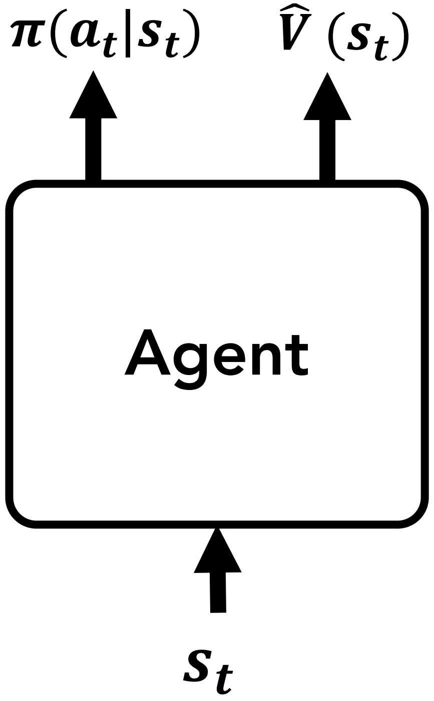
Actor-Critic Methods

Actor: models the policy (what to do)

Critic: estimate how good the state (or state-action) is by estimating the value (or Q) function

Agent predicts both action probabilities and value estimates

Actor-Critic Methods



Gradient step for parameters with respect to policy parameters:

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i|s_i;\theta) \left(\hat{A}_\pi(s_t, a_t)\right)$$

How to train the value estimator?

$$\mathcal{L}_V(\theta) = \left\|\hat{V}(s_t) - V_\pi^*(s_t)\right\|^2$$

$$V_\pi^*(s_t) = \mathbb{E}_\pi\left[\sum_{i=t} \gamma^{i-t}\, r(s_i, a_i)\right] \approx \sum_n \sum_{i=t} \gamma^{i-t}\, r(s_i, a_i)$$

Actor-Critic Methods

Gradient step for parameters with respect to policy parameters:

$$\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i|s_i;\theta)\left(\hat{A}_\pi(s_t, a_t)\right)$$

How to train the value estimator?

$$\mathcal{L}_V(\theta) = \left\| \hat{V}(s_t) - \sum_{i=t} \gamma^{i-t} r(s_i, a_i) \right\|^2$$

$$V_\pi^*(s_t) = \mathbb{E}_\pi\left[\sum_{i=t} \gamma^{i-t} r(s_i, a_i)\right] \approx \sum_n \sum_{i=t} \gamma^{i-t} r(s_i, a_i)$$

Just supervised regression with data $\{\, s_t,\ \sum_{i=t} \gamma^{i-t} r(s_i, a_i) \,\}$

Slide Credit: Stefan Lee

Actor-Critic Methods

Advantage Actor-Critic (A2C) Algorithm

While not converged:

1. Perform rollout to collect trajectory $\tau = (s_0, a_0, s_1, a_1, s_2, \ldots)$ and reward $r(\tau)$
2. Fit $\hat{V}_\pi(s)$ to sampled rewards
3. Evaluate $\hat{A}_\pi(s_t, a_t) = r(s_t, a_t) + \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t)$
4. Compute gradient estimate $\nabla_\theta J(\theta) \approx \sum_n \sum_{s_i, a_i \in \tau_n} \nabla_\theta \log \pi(a_i | s_i; \theta)\, \hat{A}_\pi(s_t, a_t)$
5. Update policy parameters $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

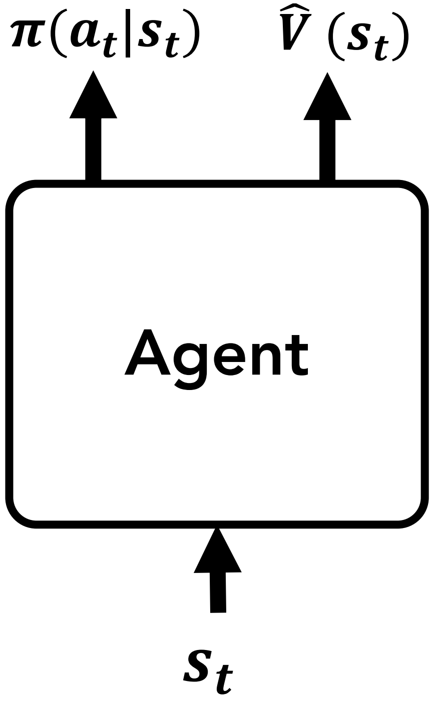
Actor-Critic Methods – Bootstrap Targets / Temporal Differences

$$\mathcal{L}_V(\theta) = \left\|\hat{V}(s_t) - V_\pi^*(s_t)\right\|^2$$

$$V_\pi^*(s_t) = \mathbb{E}_\pi\left[\sum_{i=t} \gamma^{i-t} r(s_i, a_i)\right] \approx \sum_n \sum_{i=t} \gamma^{i-t} r(s_i, a_i)$$

$$\mathcal{L}_V(\theta) = \left\|\hat{V}(s_t) - \underbrace{\sum_{i=t} \gamma^{i-t} r(s_i, a_i)}_{\text{high variance!}}\right\|^2$$

Actor-Critic Methods – Bootstrap Targets / Temporal Differences



Bootstrap target:

$$\mathcal{L}_V(\theta) = \left\|\hat{V}(s_t) - V_\pi^*(s_t)\right\|^2$$

$$V_\pi^*(s_t) = \mathbb{E}_\pi[r(s_t, a_t) + \gamma V_\pi^*(s_{t+1})]$$

Monte Carlo $\approx r(s_t, a_t) + \gamma V_\pi^*(s_{t+1})$

Value Estimate $\approx \underbrace{r(s_t, a_t) + \gamma \hat{V}_\pi(s_{t+1})}$

"biased bootstrap estimate"

This is a temporal difference target. Biased but lower variance.

Slide Credit: Stefan Lee

Actor-Critic Methods – Bootstrap Targets / Temporal Differences

Bootstrap target:

$$\mathcal{L}_V(\theta) = \left\| \hat{V}(s_t) - \left[r(s_t, a_t) + \gamma \hat{V}_\pi(s_{t+1}) \right] \right\|^2$$

$$V_\pi^*(s_t) = \mathbb{E}_\pi[r(s_t, a_t) + \gamma V_\pi^*(s_{t+1})]$$
$$\approx \underbrace{r(s_t, a_t) + \gamma \hat{V}_\pi(s_{t+1})}_{\text{"biased bootstrap estimate"}}$$

"biased bootstrap estimate"

Just supervised regression with data $\{ \boldsymbol{s_t}, \boldsymbol{r(s_i, a_i) + \gamma \hat{V}_\pi(s_{i+1})} \}$

Slide Credit: Stefan Lee

Policy Gradient Methods

Actor-Critic Recap:

Policy gradient method that **trades off variance for bias** in gradient estimates by using a simultaneously learned value function. With clever choices of baselines, we repeat actions that are better than average and avoid those that are worse (through advantage estimate).

- **When to use this?**
  - When reward functions are well-defined and simulation is cheap.
  - Any time you are doing policy gradients, might as well do this.

Slide Credit: Stefan Lee

Value-Based RL

Deep Q-Learning

Value-Based RL – Deep Q-Learning

Earlier today….

1. If we have a policy π and know the true $Q_\pi(s, a)$ -- can we derive a new policy π' that is as good or better than π?

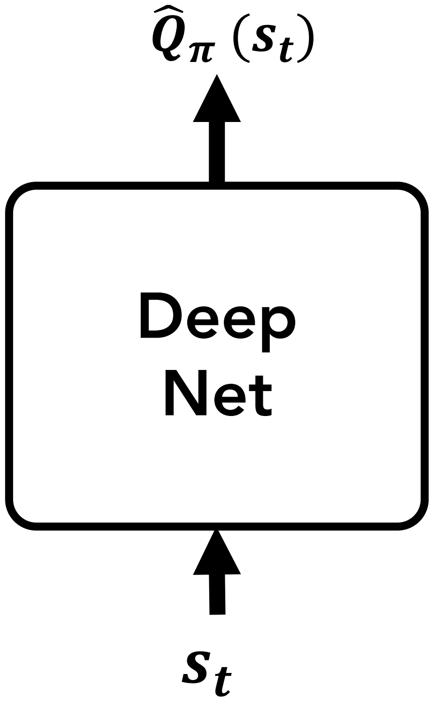
Recall that $Q_\pi(s, a)$ is the expected reward of taking action a in state s

Set $\pi(a'|s) = 1$ if a$' = argmax\hat{Q}_\pi(s, a)$

If we can estimate $\mathbf{Q_\pi(s, a)}$ why do we even need an explicit policy?

Value-Based RL – Deep Q-Learning

If we can estimate $\mathbf{Q}_{\pi}^{*}(\mathbf{s},\mathbf{a})$ why do we even need an explicit policy?



$$\mathcal{L}_Q(\theta) = \left\| \hat{Q}_{\pi}(s_t, a_t) - Q_{\pi}^{*}(s_t, a_t) \right\|^2$$

$$Q_{\pi}^{*}(s_t) = r(s_t, a_t) + \gamma V_{\pi}^{*}(s_{t+1})$$

Recall our implicit policy is a' $= argmax\, Q_{\pi}(s,a)$

$$V_{\pi}^{*}(s_t) = \mathbb{E}_{\pi}[Q_{\pi}^{*}(s_t, a_t)] = \max_{a} Q_{\pi}^{*}(s_t, a_t)$$

$$\mathcal{L}_Q(\theta) \approx \left\| \hat{Q}_{\pi}(s_t, a_t) - \left[r(s_t, a_t) + \max_{a} \hat{Q}_{\pi}(s_{t+1}, a_{t+1}) \right] \right\|^2$$

Slide Credit: Stefan Lee

Value-Based RL – Deep Q-Learning

Simplest DQN Algorithm

While not converged:

1. Perform rollout to collect trajectory $\tau = (\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1, \mathrm{a}_1, \mathrm{s}_2, \ldots)$ and reward $\mathrm{r}(\tau)$
2. **Compute loss from samples** $\mathcal{L}_Q(\theta) \approx \left\| \hat{Q}_\pi(s_t, a_t) - \left[r(s_t, a_t) + \max_a \widehat{Q}_\pi(s_{t+1}, a_{t+1}) \right] \right\|^2$
3. Update Q-network parameters $\theta' = \theta + \alpha \nabla_\theta \mathcal{L}_Q(\theta)$ with gradient decent

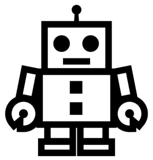
Value-Based RL – Deep Q-Learning

Weaknesses in our Simple DQN:

Limited exploration:

1. Perform rollout to collect trajectory $\tau = (\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1, \mathrm{a}_1, \mathrm{s}_2, \ldots)$ and reward $\mathrm{r}(\tau)$

Recall our implicit policy π_θ is a′ $= argmax\ Q_{\pi_\theta}(s, a)$

10	0	0	0		1	1	1	1	1

$$\mathcal{L}_Q(\theta) \approx \left\| \hat{Q}_\pi(s_t, a_t) - \left[r(s_t, a_t) + \max_a \hat{Q}_\pi\ (s_{t+1}, a_{t+1}) \right] \right\|^2$$

Slide Credit: Stefan Lee

Value-Based RL – Deep Q-Learning

ϵ-greedy policy:

$$\pi_\epsilon(s_t) = \begin{cases} \mathrm{argmax_a}\ \hat{Q}_{\pi_\theta}(\mathrm{s_t}, \mathrm{a}) & \text{with probability } \epsilon \\ \sim \text{uniform over A} & \text{with probabiliy } 1 - \epsilon \end{cases}$$

Value-Based RL – Deep Q-Learning

Off-policy DQN:

While not converged:

1. Rollout $\boldsymbol{\pi_\epsilon}$ to collect trajectory $\tau = (\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1, \mathrm{a}_1, \mathrm{s}_2, \dots)$ and reward $\mathrm{r}(\tau)$
2. Compute loss from samples $\mathcal{L}_Q(\theta) \approx \left\| \hat{\boldsymbol{Q}}_{\boldsymbol{\pi_\theta}}(\boldsymbol{s_t}, \boldsymbol{a_t}) - \left[r(s_t, a_t) + \max_a \hat{\boldsymbol{Q}}_{\boldsymbol{\pi_\theta}}(\boldsymbol{s_{t+1}}, \boldsymbol{a_{t+1}}) \right] \right\|^2$
3. Update Q-network parameters $\theta' = \theta + \alpha \nabla_\theta \mathcal{L}_Q(\theta)$ with gradient decent

Exploration policy is π_ϵ, but we evaluate based on π_θ

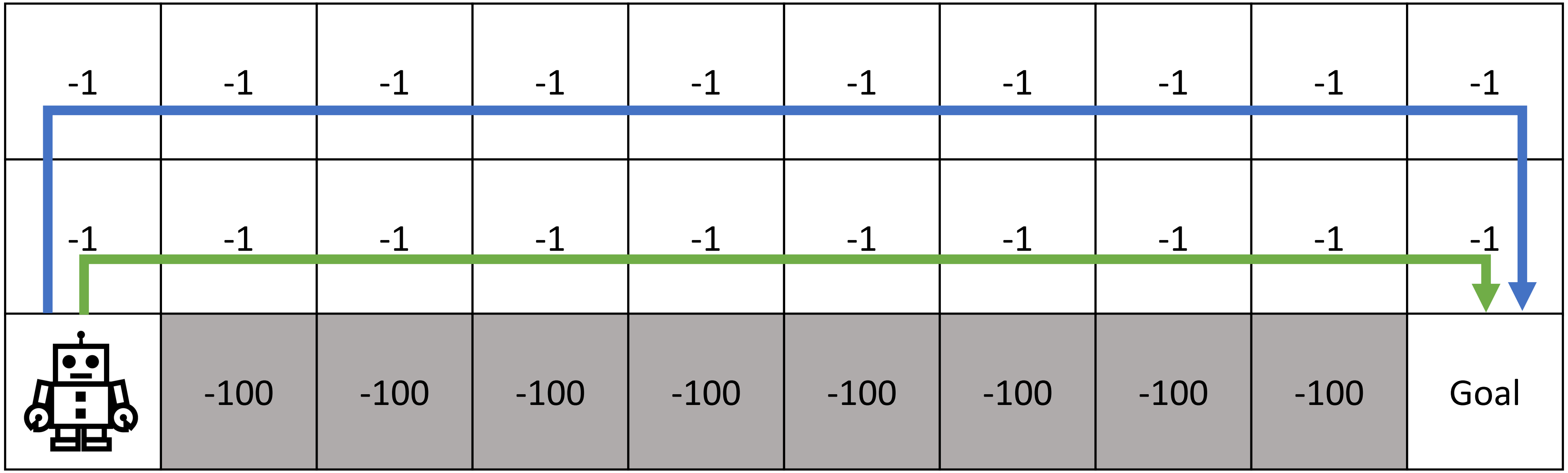
On-policy: Exploration policy == evaluation policy

Off-policy: Exploration policy != evaluation policy

Value-Based RL – Deep Q-Learning

ϵ-greedy vs. argmax:

Consider the cliff-walking game:



Value-Based RL – Deep Q-Learning

Weaknesses in our Simple DQN:

Examples in a trajectory are correlated

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots)$$

Gradient updates are highly correlated and can lead to oscillation.

Value-Based RL – Deep Q-Learning

Experience Replay:

Idea: collect a buffer of trajectories and then randomly sample transitions to perform our update

$$Replay\ Buffer = \left\{ \left(s_t^j, a_t^j, r_t^j, s_{t+1}^j \right) \right\}_j$$

$$batch \sim Replay\ Buffer$$

Value-Based RL – Deep Q-Learning

Off-policy DQN with Experience Replay:

While not converged:

1. Rollout π_ϵ to collect trajectory $\tau = (s_0, a_0, s_1, a_1, s_2, \dots)$ and reward $\mathrm{r}(\tau)$
2. Store transitions $(s_\mathrm{t}, a_\mathrm{t}, \mathrm{r}_\mathrm{t}, s_{\mathrm{t}+1})$ in replay buffer B
3. Sample N transitions from B and compute

$$\mathcal{L}_Q(\theta) \approx \sum_{n=1}^{N} \left\| \hat{Q}_{\pi_\theta}(s_t^n, a_t^n) - \left[r(s_t^n, a_t^n) + \max_a \hat{Q}_{\pi_\theta}(s_{t+1}^n, a_{t+1}^n) \right] \right\|^2$$

4. Update Q-network parameters $\theta' = \theta + \alpha \nabla_\theta \mathcal{L}_Q(\theta)$ with gradient decent

Tends to be more sample efficient than policy-gradient methods because transitions are valid targets forever.

Slide Credit: Stefan Lee

Value-Based RL – Deep Q-Learning

Weaknesses in our Simple DQN:

Chasing a non-stationary target:

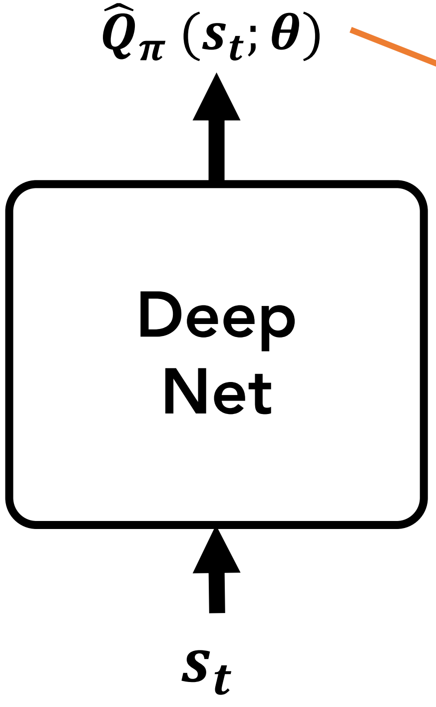
$$\mathcal{L}_Q(\theta) \approx \sum_{n=1}^{N} \left\| \boldsymbol{\hat{Q}_{\pi_\theta}(s_t^n, a_t^n)} - \left[r(s_t^n, a_t^n) + \max_a \boldsymbol{\hat{Q}_{\pi_\theta}(s_{t+1}^n, a_{t+1}^n)} \right] \right\|^2$$

Same network producing both.
Each update changes the targets.

Value-Based RL – Deep Q-Learning

Target networks:

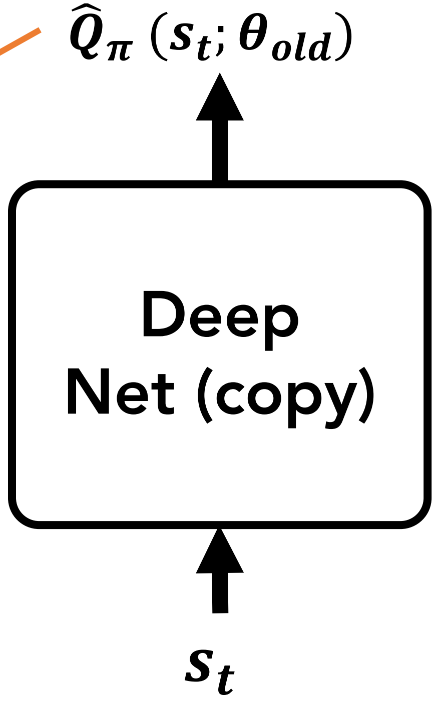
Idea: Keep an old version of parameters around to estimate targets



$$\mathcal{L}_Q(\theta) \approx \sum_{n=1}^{N} \left\| \hat{Q}_{\pi_\theta}(s_t^n, a_t^n) - \left[r(s_t^n, a_t^n) + \max_a \hat{Q}_{\pi_\theta}(s_{t+1}^n, a_{t+1}^n) \right] \right\|^2$$

Re-copy the target network weights periodically.

Fixed Target Network



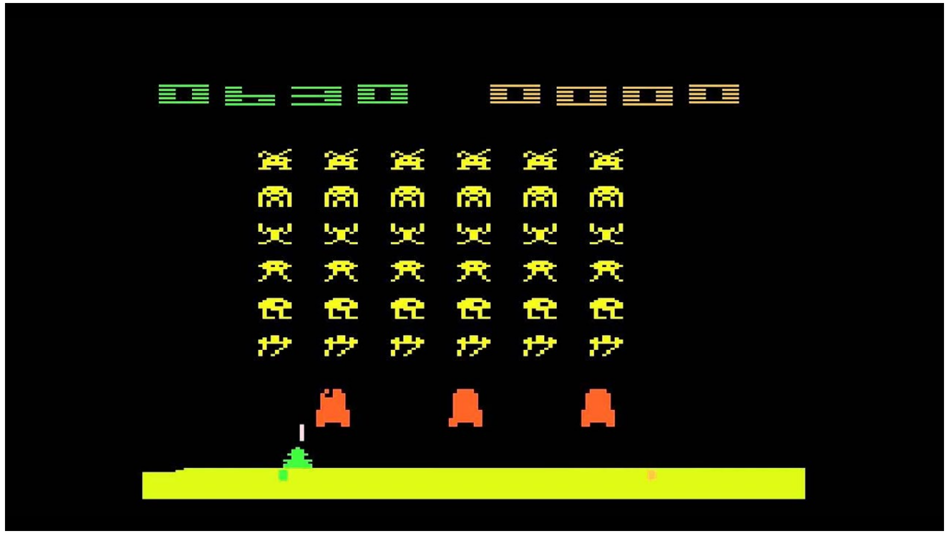
Slide Credit: Stefan Lee

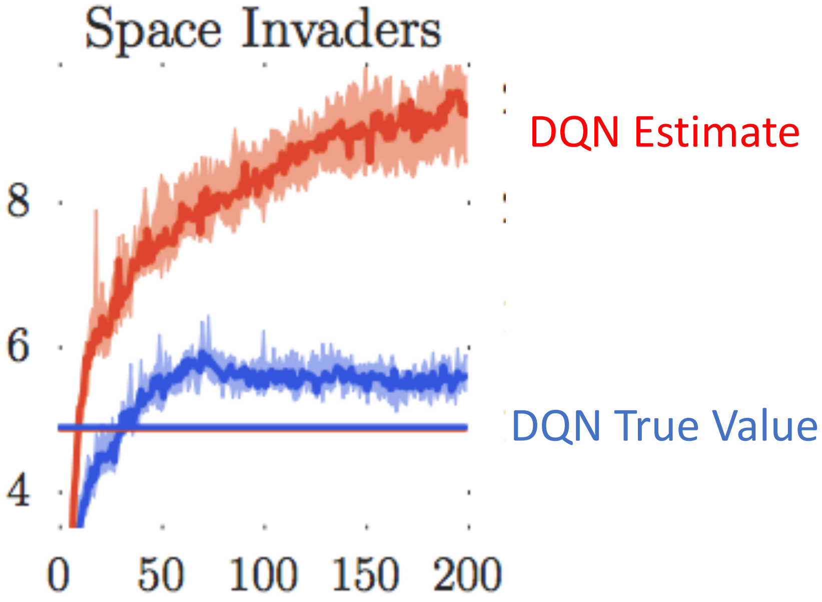
Value-Based RL – Double Q-Learning

Weaknesses in DQN:

Tends to overestimates action values:

$$\mathcal{L}_Q(\theta) \approx \left\| \hat{Q}_\pi(s_t, a_t) - \left[r(s_t, a_t) + \max_a Q^*_\pi(s_{t+1}, a_{t+1}) \right] \right\|^2$$





Value-Based RL – Deep Q-Learning

Deep Q-Learning:

Assume an implicit greedy policy and just learn its action-value function. The approach is model-free and fairly general but does require some tricks to overcome a few problems during training.

Bonus Concept: Experience replay to reduce correlation in examples is broadly applicable.

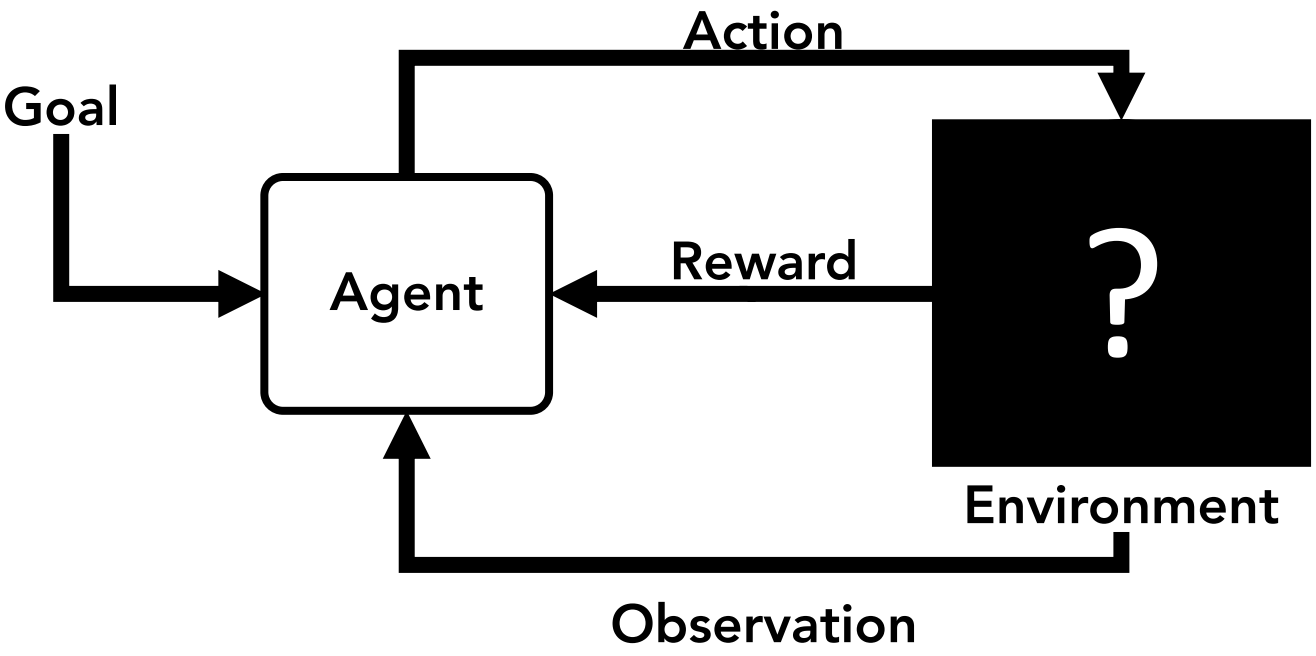
- **When to use this?**
  - When reward functions are well-defined but rollouts are more expensive.
  - Worried about sample efficiency and have strong exploration policy.
  - … don't mind trying to get it stable … which is a challenge sometimes

Slide Credit: Stefan Lee

Model-based RL

Model-based RL

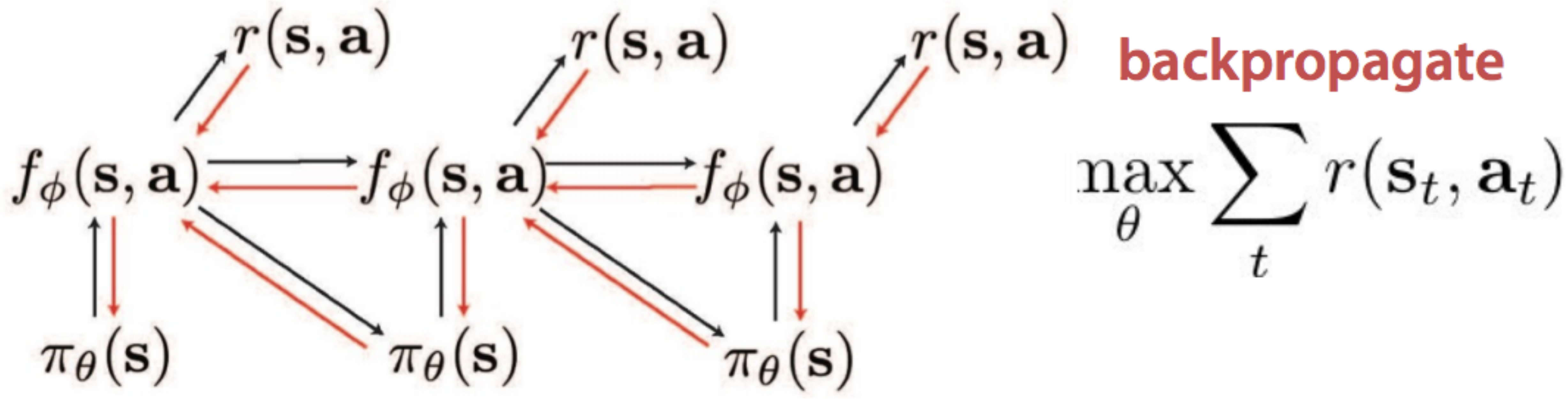
All the previous RL methods we discussed are model-free algorithms:



If we can estimate a model $s_{t+1} \sim f(s_t, a_t)$, can we be more efficient?

Model-based RL

If we can estimate a model $s_{t+1} \sim f(\mathbf{s}_t, \mathbf{a}_t)$, can we be more efficient?



Dynamics are reward independent – changing reward function isn't a problem!

Slide Credit: Sergey Levine

Model-based RL

Model-based Control Algorithm

1. Run base policy (possibly random or human) to collect samples
2. Fit a model $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ using least squares or some other loss
3. Backprop through $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ to optimize policy parameters

As with Behavior Cloning, we may suffer from state distribution mis-match.

Slide Credit: Sergey Levine

Model-based RL

Model-based Control Algorithm (Iterative)

Run base policy (possibly random or human) to collect samples

While not converged:
1. Fit a model $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ using least squares or some other loss
2. Backprop through $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ to optimize policy parameters
3. Run this backprop derived policy and add samples to training set

Luckily, the world is an oracle with respect to the model! No need to worry about querying an expert.

Slide Credit: Sergey Levine

Model-based RL

Model Predictive Control Algorithm (Iterative)

Run base policy (possibly random or human) to collect samples

While not converged:

1. Fit a model $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ using least squares or some other loss
    1. Backprop through $\boldsymbol{f}(\mathbf{s_t}, \mathbf{a_t})$ to optimize policy parameters
    2. Run this backprop derived policy and add samples to training set
    3. Take a step with this policy then refit based on current state

A bit expensive to run this optimization every step.

Value-Based RL – Deep Q-Learning

Model-Based Reinforcement Learning from Pixels
with Structured Latent Variable Models 2019

https://bair.berkeley.edu/blog/2019/05/20/solar/

Slide Credit: Stefan Lee

Value-Based RL – Deep Q-Learning

Learning Latent Dynamics for
Planning from Pixels 2019

https://planetrl.github.io

Slide Credit: Stefan Lee

Model-based RL:

Model-based Recap:

Learn the dynamics model and then optimize for long-term rewards through it (aka plan!). Very sample efficient and can be self-supervised. Some initial work does it directly in pixel space (including goal specification).

- **When to use this?**
  - When dynamics are unknown (e.g. physical systems) but modelable
  - Very worried about sample efficiency (e.g. robotics)
  - Want to transfer to different goals

Slide Credit: Stefan Lee

Next time

- Paper presentations (3/15)
  - Mapping Instructions and Visual Observations to Actions with Reinforcement Learning (Atmika)
  - Learning Interpretable Spatial Operations in a Rich 3D Blocks World (Discussion)
- Thursday (3/18): Instruction following – VLN