Intro to deep reinforcement learning

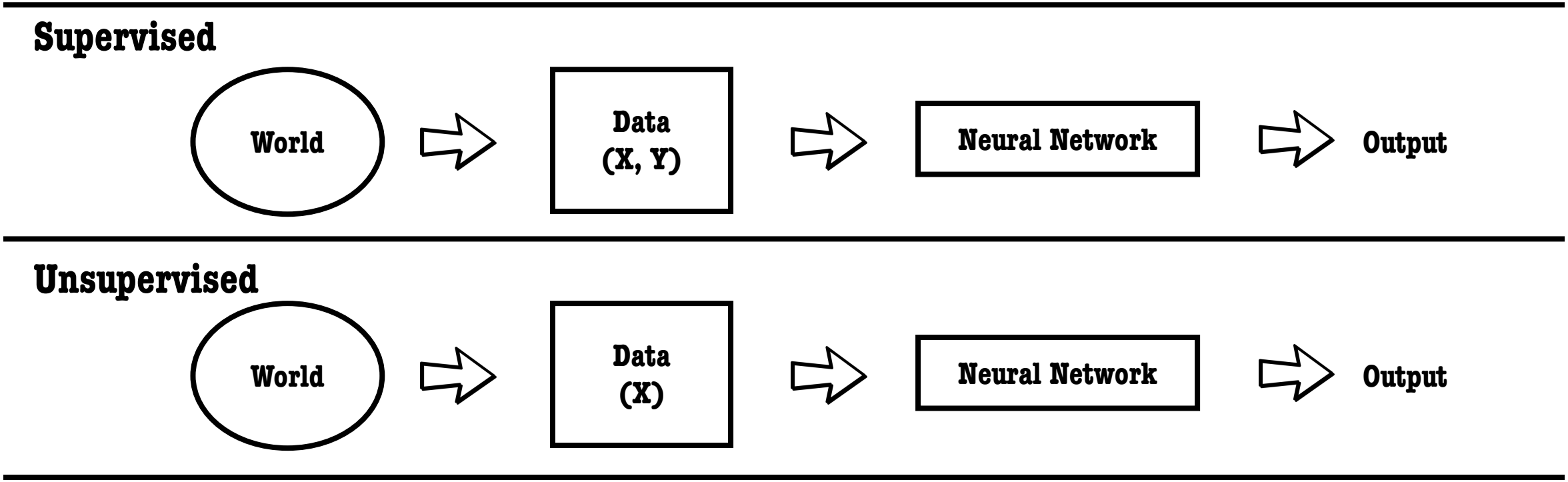
Master in Computer Vision Barcelona
Adriana Romero
adrianars@fb.com

Outline

- Motivation, application and tools
- Problem, terminology and notation
- RL formalism: MDPs/POMDPs, the RL goal & agent, reward, return, value functions, policy
- RL algorithms: overview, value-based, policy gradient
- Wrap Up

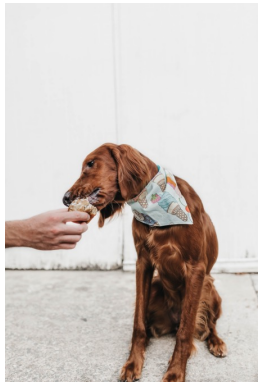
Motivation, applications & tools

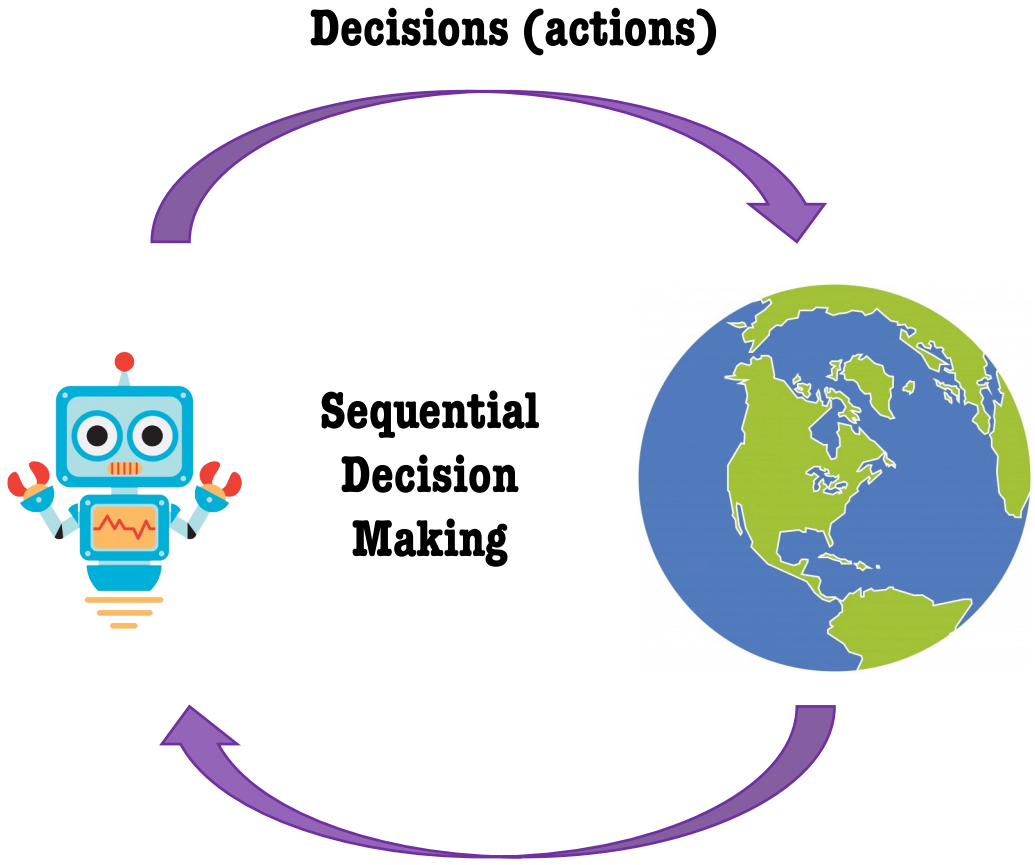
Motivation



Slide courtesy of M. Drozdzal

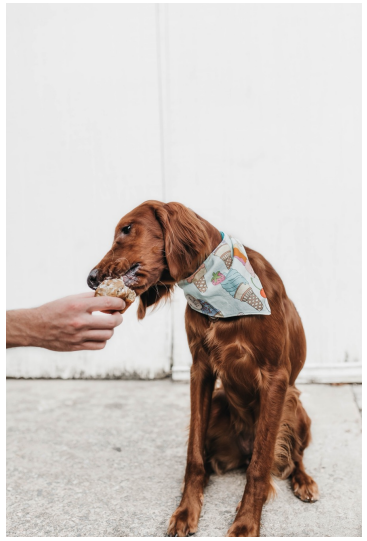
What is RL?



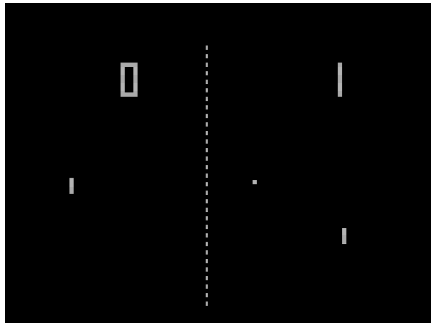


Images from http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf and https://unsplash.com/

Some examples



Action: muscle contraction
Observation: smell, sight
Reward: food



Action: move up/down
Observation: pixels
Reward: scoring

Action: directions (control stick)
Observation: position, orientation
Reward: successful manoeuvres

Images from https://unsplash.com/

Applications of RL

Robotics

Financial trading

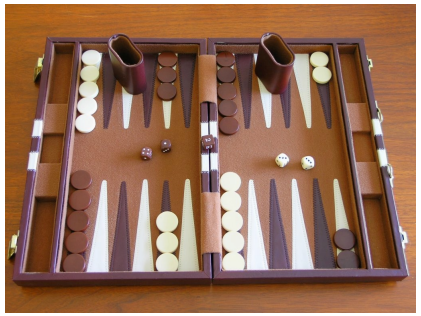
Conversational systems

Games

Healthcare

Images from https://unsplash.com/

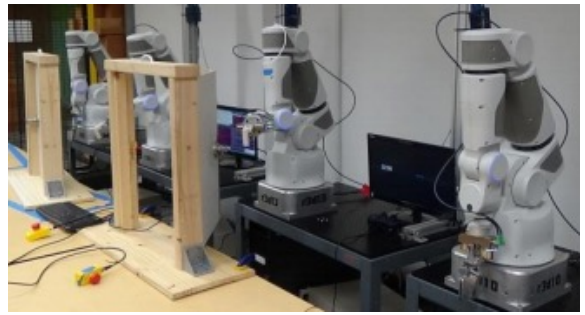
Some successes of RL



Backgammon
[Tesauro, 1995]

Helicopter
[Abbeel et al., 2008]
[Coates et al., 2008]
[Abbeel et al., 2008]

Atari
[Mnih et al., 2013]
[Schulman et al., 2015]
[Mnih et al., 2016]



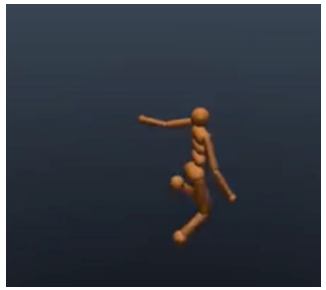
Robots
[Levine et al., 2015]
[Gu et al., 2016]

GO
[Silver et al., 2016]
[Tian et al., 2019]

StarCraft
[DeepMind, 2019]

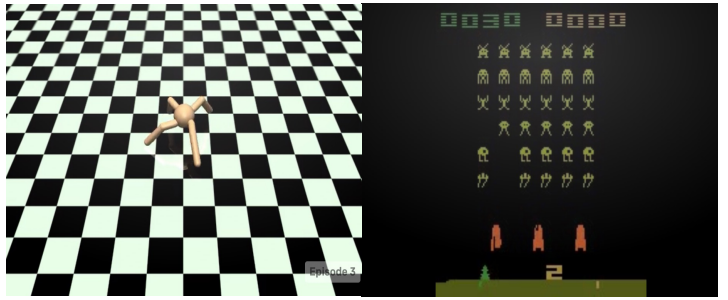
RL tools

MuJoCo

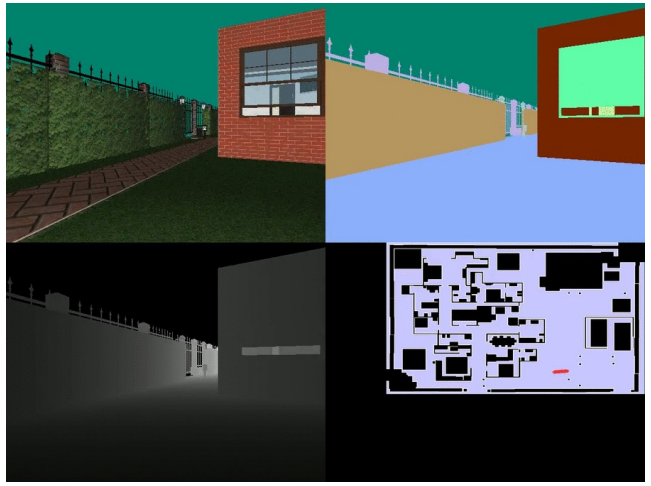


http://www.mujoco.org/index.html

Gym



https://gym.openai.com/



House 3D

https://github.com/facebookresearch/House3D

DeepMind Lab

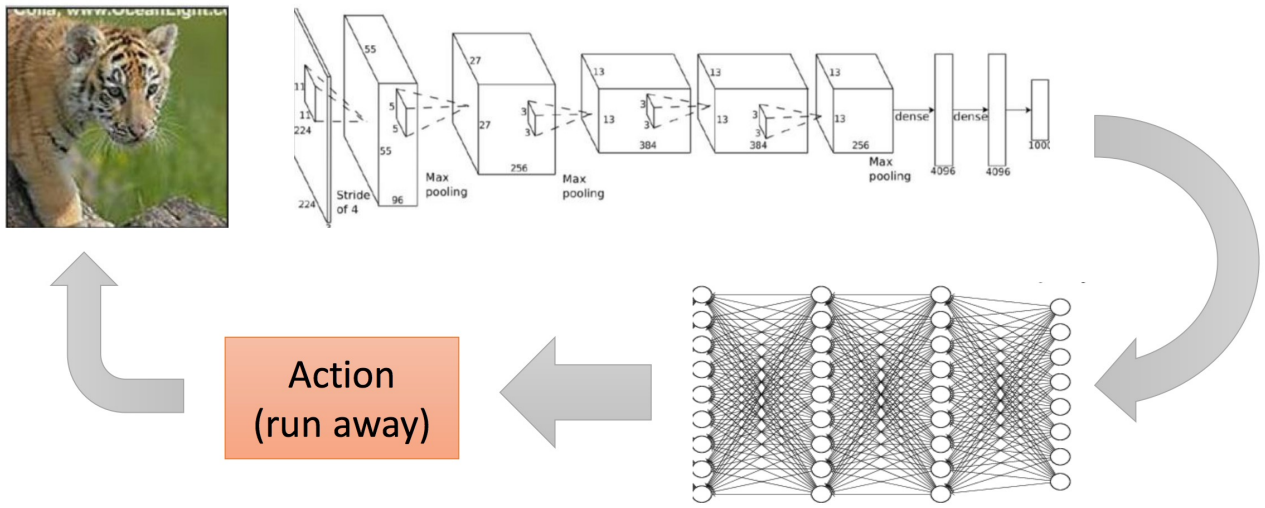
https://github.com/deepmind/lab

Problem, notation & terminology

Deep RL

Current deep learning models allow us to learn multiple levels of representations end-to-end (supervised/unsupervised).

→ **What does end-to-end learning mean for sequential decision making?**



… how can we train those models?

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf

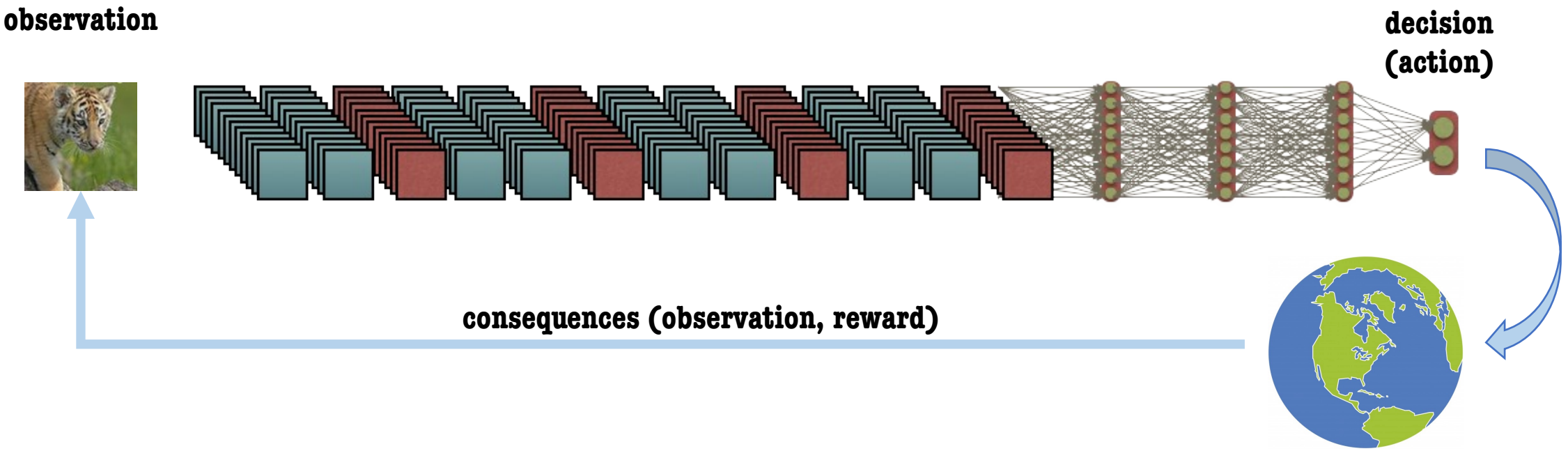
In supervised learning...

input

output (decision)

- We care about single, isolated decisions. Decisions do not affect future decisions.
- We learn with supervisor: signal comes from the ground truth (which tells us whether we gave the correct answer).

But in RL...



- We care about a sequence of decisions (time matters, sequential data, no i.i.d. assumption). Decisions have consequences (they affect the data we subsequently receive).
- We learn by trial/error (attempt to do a task until we get it right). There is no direct supervision (when you see this, do that).
- Feedback is delayed (we realize about good/bad decisions with time). The reward tells us whether or not we are doing well.

Are there other forms of supervision?

- Learning from demonstrations:
  - Copying observed behavior (imitation learning, supervised learning for sequential decision making)
  - Inferring rewards from observed behavior (inverse reinforcement learning)

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf

Imitation learning

[Bojarski et al., 2016]

From http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf

Beyond copying: inferring intent

Warneken and Tomasello

From http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf

Are there other forms of supervision?

- Learning from demonstrations:
  - Copying observed behavior (imitation learning)
  - Inferring rewards from observed behavior (inverse reinforcement learning)
- Learning from observing the world:
  - Learning to predict and use predictions to act.
  - Unsupervised learning (learn about the world)
- Learning from other tasks
  - Transfer learning
  - Meta-learning: learning to learn (use past experience of learning other tasks to learn more quickly).

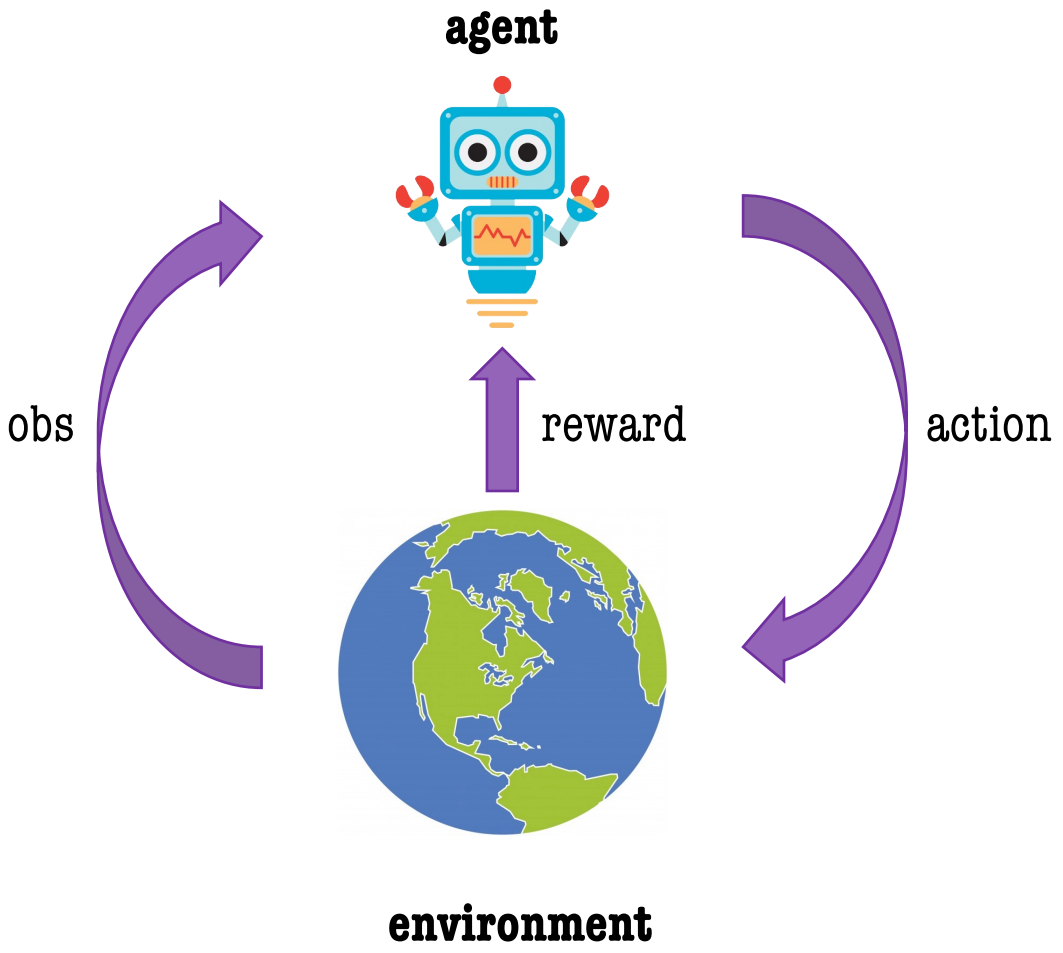
From http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_1_introduction.pdf

RL: the problem

- ✓ **Agent** (our algorithm): at each time step t:
  - Executes an action a_t (what it wants to do)
  - Receives an observation o_t (what it sees)
  - Receives a reward r_t (scalar feedback, how well it is doing)

- ✓ **Environment** (our world): at each time step, receives an action, and sends an observation and a reward). Environment is initially unknown, agent figures things out by itself.

- ✓ **Goal:** Pick sequence of actions to maximize future reward.



Sequential decision making

RL: terminology

- ✓ **History**: experience, time series of $(a_t,\ o_t\ ,\ r_t)$

$$h_t = a_1, o_1, r_1, \ldots, a_t, o_t, r_t$$

Determines what happens next

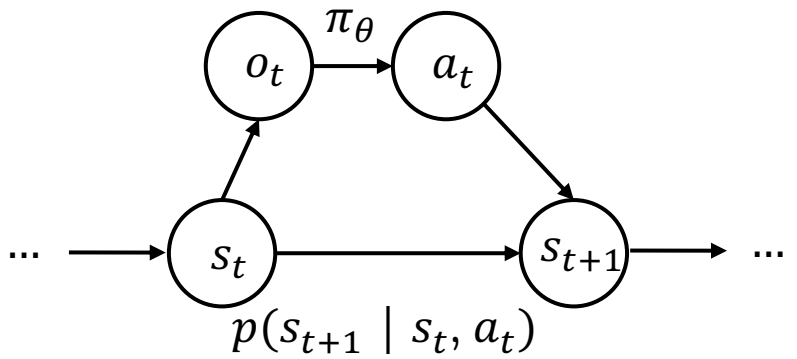
- ✓ **State**: summary of relevant information to determine what happens next

$$s_t = f(h_t)$$

- ✓ **Policy:** how the agent picks actions (behavior)

$$\pi_\theta(a_t \mid o_t) \text{ or } \pi_\theta(a_t \mid s_t)$$

State vs. observation:



- ✓ **Transition:** to predict next state

$$p(s_{t+1} \mid s_t, a_t)$$

Markov property:

"*The future is independent of the past given the present*"

$$p(s_{t+1} \mid s_t) = p(s_{t+1} \mid s_1, \ldots, s_t)$$

Based on http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf

Exploration vs. exploitation

- The RL agent performs trial and error learning.
- It must discover a good policy, from its own experiences without loosing too much reward.

- Online decision-making involves a fundamental choice:
  - ✓ Exploitation: Make the best decision given current information
  - ✓ Exploration: Gather more information

Based on http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf

RL formalism

Markov Decision Processes (MDP)

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma\}$$

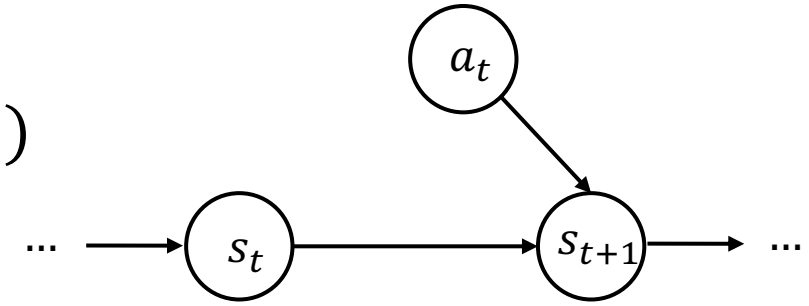
$\mathcal{S}$ - state space

$\mathcal{A}$ - action space

$\mathcal{P}$ - transition operator (3d tensor): $p(s_{t+1} \mid s_t, a_t)$

$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - reward

$\gamma \in [0, 1]$ - discount factor



Follows Markov property.

Partially Observable MDP (POMDP)

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{E}, r, \gamma\}$$

$\mathcal{S}$ - state space

$\mathcal{A}$ - action space

$\mathcal{P}$ - transition operator (3d tensor): $p(s_{t+1} \mid s_t, a_t)$

$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - reward

$\gamma \in [0, 1]$ - discount factor

Partially Observable MDP (POMDP)

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{E}, r, \gamma\}$$

$\mathcal{S}$ - state space

$\mathcal{A}$ - action space

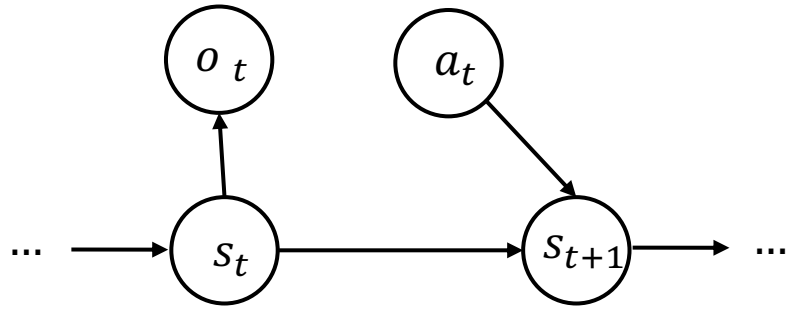
$\mathcal{P}$ - transition operator (3d tensor): $p(s_{t+1} \mid s_t, a_t)$

$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - reward

$\gamma \in [0, 1]$ - discount factor

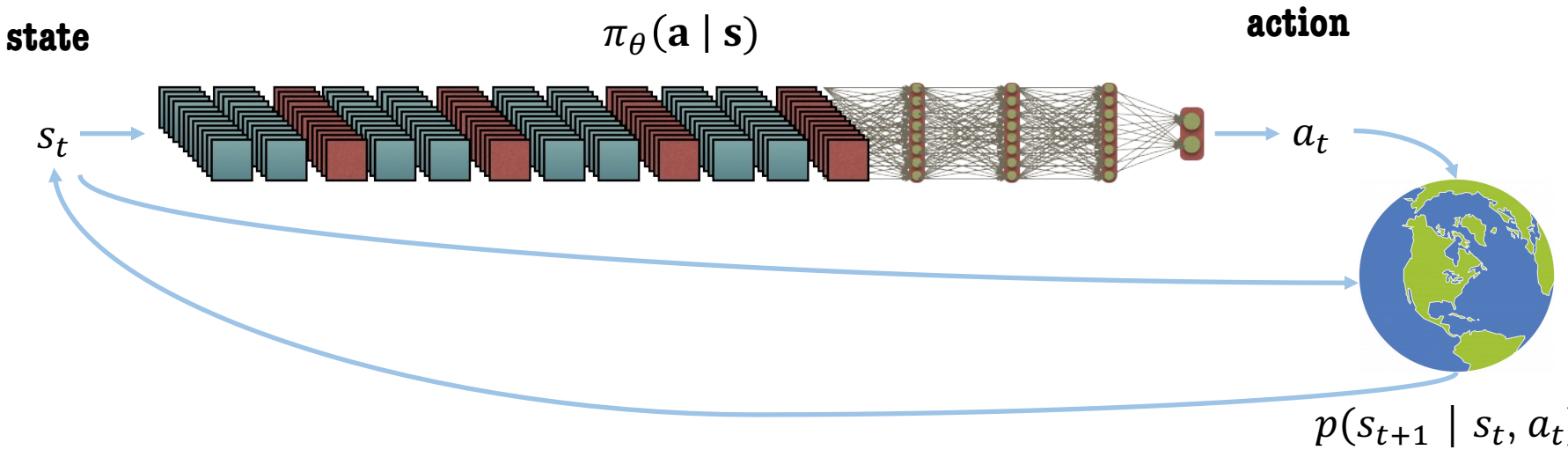
$\mathcal{O}$ - observation space

$\mathcal{E}$ - emission probability $p(o_t \mid s_t)$ / emission function



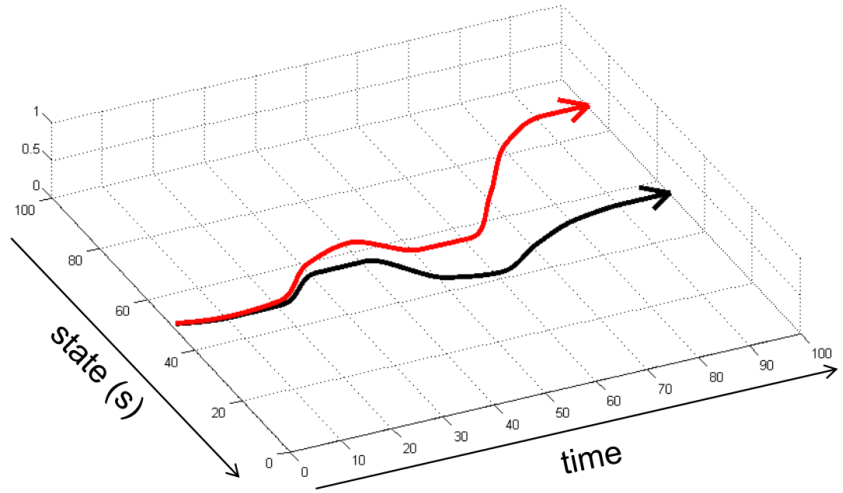
Follows Markov property.

The RL goal (1)



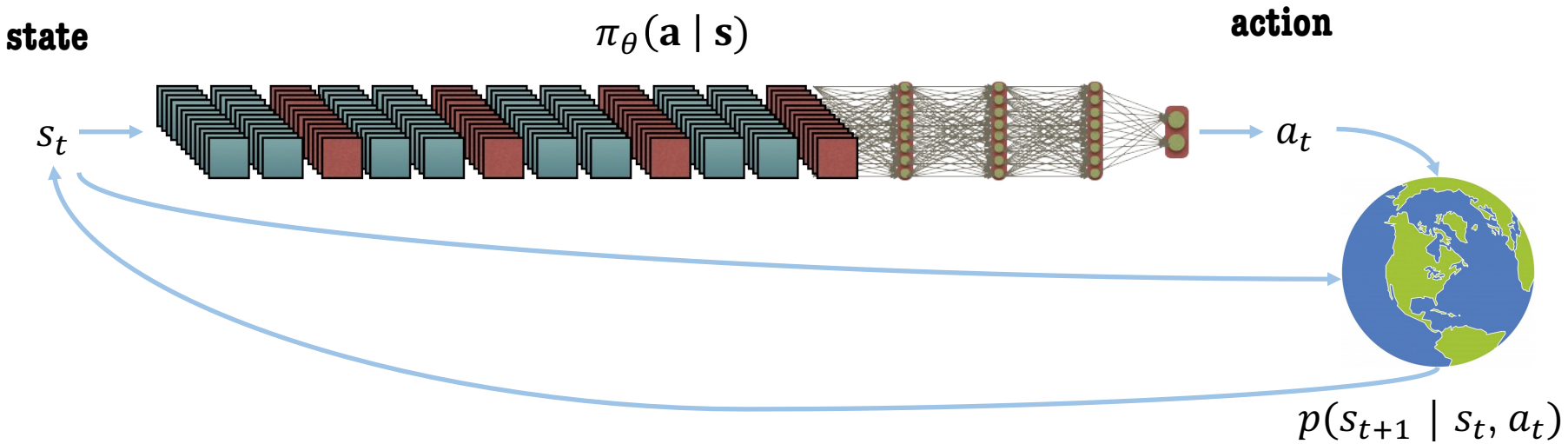
This process gives rise to a distribution of $(\mathbf{s}, \mathbf{a})$ sequences:

$$p_\theta(\underbrace{s_1, a_1, \ldots, s_T, a_T}_{\tau}) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$



Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_3_rl_intro.pdf

The RL goal (2)



This process gives rise to a distribution of $(\mathbf{s}, \mathbf{a})$ sequences:

$$p_\theta(\underbrace{s_1, a_1, \ldots, s_T, a_T}_{\tau}) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

The objective in terms of this distribution is: $\theta^* = \text{argmax}_\theta \, \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t r(s_t, a_t)]$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_3_rl_intro.pdf

Reward vs Return

$$\theta^* = \text{argmax}_\theta \, \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: scalar, which tells us how well we are doing at a given time step.

In RL, we care about the total reward. The return is the total reward, defined as:

$R(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$ - for finite-horizon tasks (e.g. games, mazes)

$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ - for infinite-horizon tasks (e.g. flying, juggling)

discount factor

Why discount?

- γ close to 0 leads to a myopic evaluation
- γ close to 1 leads to a far-sighted evaluation

Value functions

$$\theta^* = \text{argmax}_\theta \, \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

Value functions

$$\theta^* = \text{argmax}_\theta \underbrace{\sum_t \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)]}$$

$$\mathbb{E}_{s_1\sim p\,(s_1)}\,[\mathbb{E}_{a_1\sim\pi\,(a_1|s_1)}[\underbrace{r(s_1,a_1) + \mathbb{E}_{s_2\sim p\,(s_2|s_1,a_1)}[\mathbb{E}_{a_2\sim\pi(a_2|s_2)}[r(s_2,a_2) + \ \ldots|s_2]\mid s_1,a_1]}_{Q(s_1,a_1)}|s_1]]$$

$$\mathbb{E}_{s_1\sim p\,(s_1)}\,[\mathbb{E}_{a_1\sim\pi\,(a_1|s_1)}[Q(s_1,a_1)|s_1]]$$

It is often convenient to know the conditional expectations, as they can be used to improve our policy.

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_3_rl_intro.pdf

Action-value function

Q-function: expected return from taking a_t from s_t

$$Q^{\pi}(s_t, a_t) = \sum_{t=t'}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

The Q-function can be decomposed as:

1. the immediate reward $r(s_t, a_t)$
2. the discounted value of the successor state $\gamma\, Q^{\pi}(s_{t+1}, a_{t+1})$

Law of iterated expectations:
$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$

$$\begin{aligned}
Q^{\pi}(s_t, a_t) &= \mathbb{E}_{\pi_\theta}[\textstyle\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t, a_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, r(s_{t+1}, a_{t+1}) + \gamma^2\, r(s_{t+2}, a_{t+2}) + \ \ldots | s_t, a_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, (r(s_{t+1}, a_{t+1}) + \gamma\, r(s_{t+2}, a_{t+2}) + \ \ldots) | s_t, a_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, (\textstyle\sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1})) | s_t, a_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, Q^{\pi}(s_{t+1}, a_{t+1}) | s_t, a_t]
\end{aligned}$$

Bellman expectation equation

State-value function

V-function: expected return from s_t

$$V^{\pi}(s_t) = \sum_{t=t'}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t]$$

$$V^{\pi}(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^{\pi}(s_t, a_t)]$$

and $\mathbb{E}_{s_1 \sim p(s_1)}[V^{\pi}(s_1)]$ is the RL objective.

The V-function can be decomposed as:

1. the immediate reward $r(s_t, a_t)$
2. the discounted value of the successor state $\gamma\, V^{\pi}(s_{t+1})$

Law of iterated expectations:
$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$

$$\begin{aligned}
V^{\pi}(s_t) &= \mathbb{E}_{\pi_\theta}[\textstyle\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, r(s_{t+1}, a_{t+1}) + \gamma^2\, r(s_{t+2}, a_{t+2}) + \ \ldots \mid s_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, (r(s_{t+1}, a_{t+1}) + \gamma\, r(s_{t+2}, a_{t+2}) + \ \ldots) \mid s_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, (\textstyle\sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1})) | s_t] \\
&= \mathbb{E}_{\pi_\theta}[r(s_t, a_t) + \gamma\, V^{\pi}(s_{t+1}) | s_t]
\end{aligned}$$

Bellman expectation equation

Advantage function

Recall that:

$$Q^{\pi}(s_t, a_t) = \sum_{t=t'}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

$$V^{\pi}(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^{\pi}(s_t, a_t)]$$

To identify good actions: if $Q^{\pi}(s, a) > V^{\pi}(s)$, then a is better than average.

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \rightarrow \text{Advantage function}$$

Tells us how much better/worse is a_t than the average action.

Optimal policy

- ✓ The optimal Q-function is the maximum action-value function over all policies:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

- ✓ The optimal V-function is the maximum state-value function over all policies:

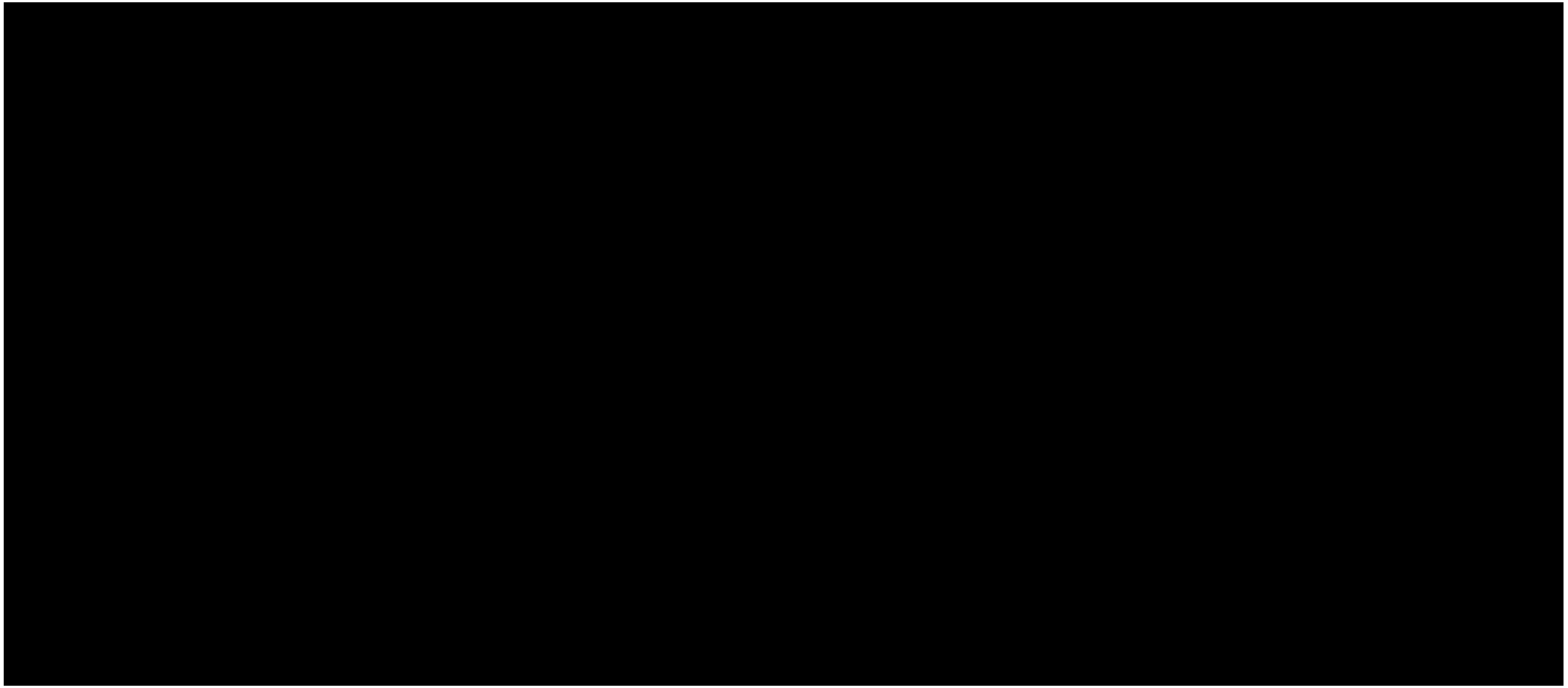
$$V^*(s) = \max_\pi V^\pi(s)$$

- ✓ Comparing policies: $\pi \geq \pi'$ if $V^\pi(s) \geq V^{\pi'}(s), \forall s$.
- ✓ Any policy that achieves $Q^*(s,a)$ or $V^*(s)$ is called optimal policy π^*.
- ✓ All optimal policies achieve the optimal state-value function. All optimal policies achieve the optimal action-value function.

- A policy π' can be improved by maximizing over $Q^{\pi'}(s,a)$: set $\pi(a|s) = 1$ if $a = \text{argmax}_a\, Q^{\pi'}(s,a)$ (this policy is at least as good as π').
- An optimal policy can be found by maximizing over $Q^*(s,a)$: set $\pi^*(a|s) = 1$ if $a = \text{argmax}_a\, Q^*(s,a)$
- To identify good/bad actions: advantage function.

A practical example

How MRI works?

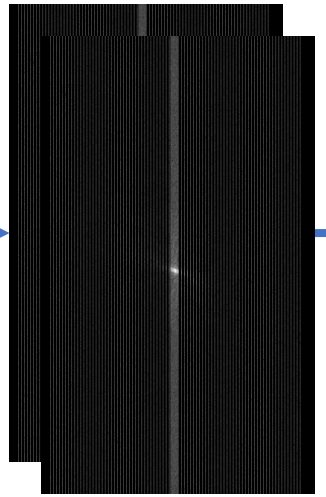


Slide courtesy of M. Drozdzal

Image credit: NYU

Making MRIs faster

measurements



reconstruction

Slide courtesy of M. Drozdzal

Personalized MRI

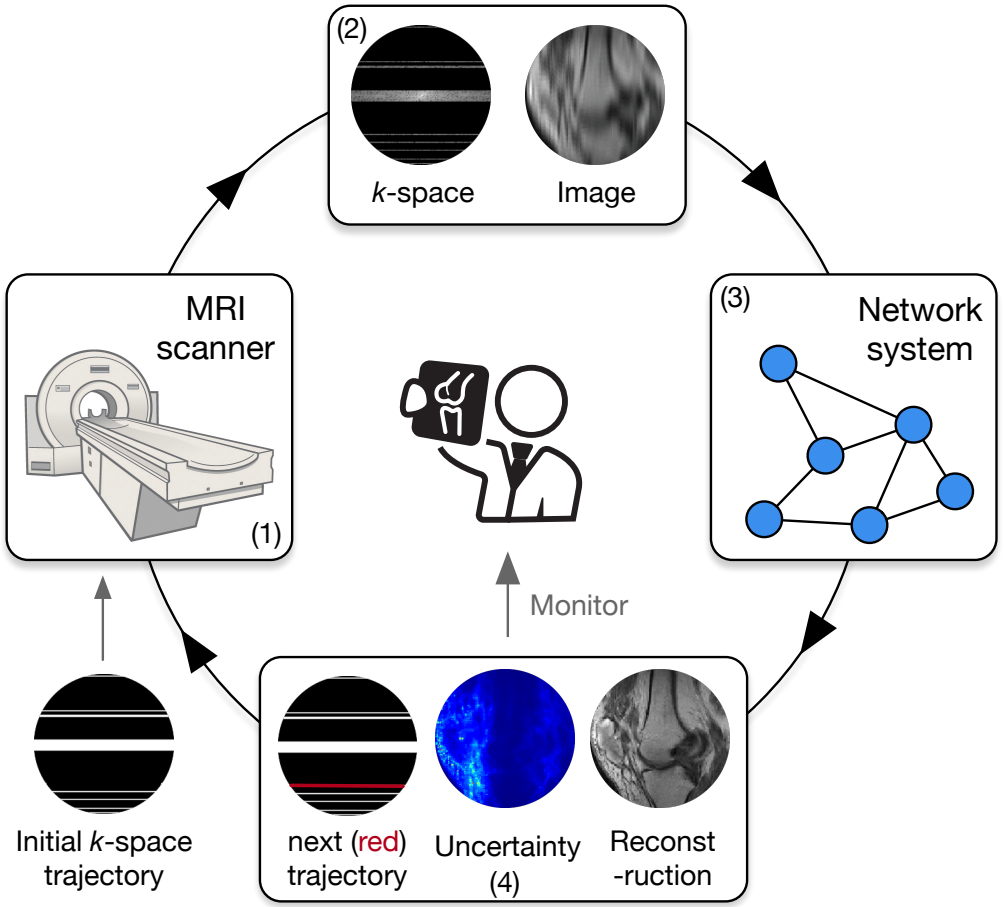
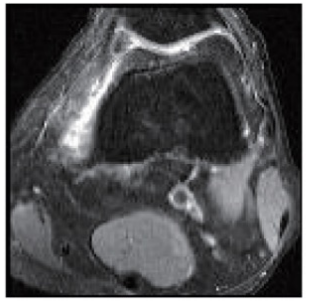


Image credit: Zizhao Zhang

Personalized MRI with RL (1)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.



Personalized MRI with RL (2)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

Personalized MRI with RL (3)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

$\mathcal{P}$ – $\mathbf{M}_{t+1} = \mathbf{M}_t + \mathbf{M}_{a_t}$; note that $\mathbf{x}$ remains unchanged.

Personalized MRI with RL (4)

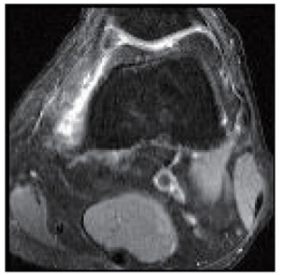
$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

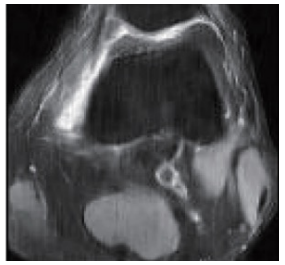
$\mathcal{P}$ – $\mathbf{M}_{t+1} = \mathbf{M}_t + \mathbf{M}_{a_t}$; note that $\mathbf{x}$ remains unchanged.

r – decrease in reconstruction error with respect to the previous reconstruction

$$r(s_t, a_t) = \mathrm{MSE}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}) - \mathrm{MSE}(\hat{\mathbf{x}}_t, \mathbf{x}) .$$



?=



Personalized MRI with RL (5)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

$\mathcal{P}$ – $\mathbf{M}_{t+1} = \mathbf{M}_t + \mathbf{M}_{a_t}$; note that $\mathbf{x}$ remains unchanged.

r – decrease in reconstruction error with respect to the previous reconstruction

$$r(s_t, a_t) = \mathrm{MSE}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}) - \mathrm{MSE}(\hat{\mathbf{x}}_t, \mathbf{x}) .$$

$\gamma \in [0, 1]$ - discount factor.

Personalized MRI with RL (6)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

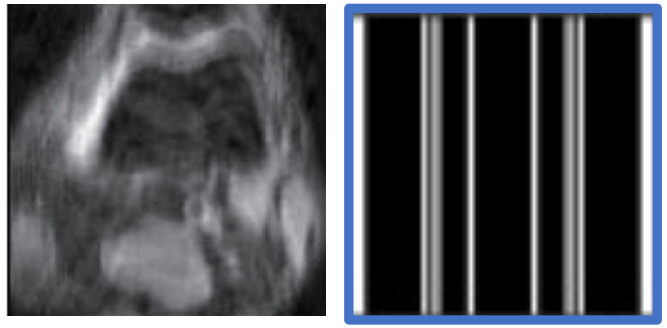
$\mathcal{P}$ – $\mathbf{M}_{t+1} = \mathbf{M}_t + \mathbf{M}_{a_t}$; note that $\mathbf{x}$ remains unchanged.

r – decrease in reconstruction error with respect to the previous reconstruction

$$r(s_t, a_t) = \mathrm{MSE}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}) - \mathrm{MSE}(\hat{\mathbf{x}}_t, \mathbf{x}) .$$

$\gamma \in [0, 1]$ - discount factor.

$\mathcal{O}$ – set of all possible tuples $o_t = < \hat{\mathbf{x}}_t, \mathbf{M}_t >$.



Personalized MRI with RL (7)

$\mathcal{S}$ - set of all possible tuples $s_t = < \mathbf{x}, \mathbf{M}_t >$. $\mathbf{x}$ is the image recovered from all measurements (unobservable), $\mathbf{M}_t$ is the mask that tells us the measurements to take.

$\mathcal{A}$ – the set of all possible a_t measurement indices that can be acquired, $\mathcal{A} = \{1, 2, \dots\}$.

$\mathcal{P}$ – $\mathbf{M}_{t+1} = \mathbf{M}_t + \mathbf{M}_{a_t}$; note that $\mathbf{x}$ remains unchanged.

r – decrease in reconstruction error with respect to the previous reconstruction

$$r(s_t, a_t) = \mathrm{MSE}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}) - \mathrm{MSE}(\hat{\mathbf{x}}_t, \mathbf{x}) .$$

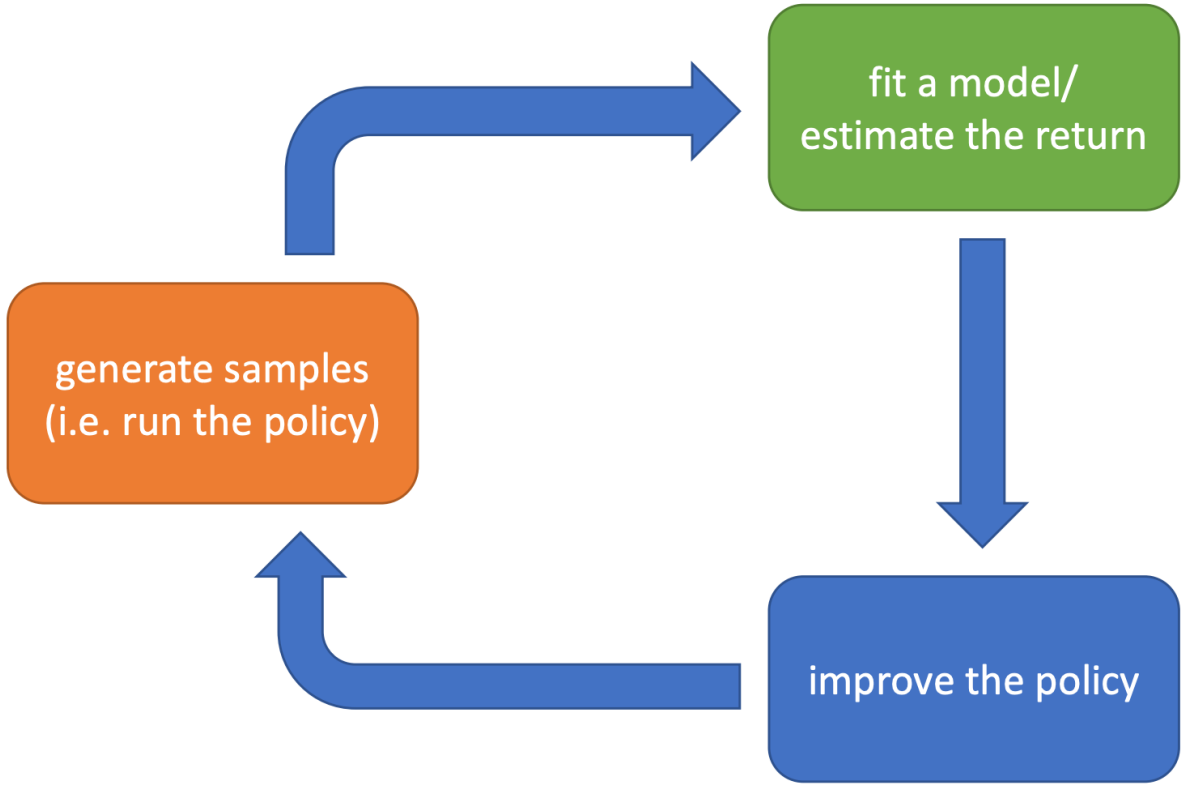
$\gamma \in [0, 1]$ - discount factor.

$\mathcal{O}$ – set of all possible tuples $o_t = < \hat{\mathbf{x}}_t, \mathbf{M}_t >$.

$\mathcal{E}$ - $\hat{\mathbf{x}}_{t+1} = f(\mathbf{x}, \mathbf{M}_{t+1})$, where f is e.g. a neural network.

RL algorithms

Structure of algorithms



From http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_3_rl_intro.pdf

Overview of algorithms

Policy gradients

Directly differentiate the RL objective.

1. Evaluate return.
2. Learn policy: take a gradient step to get a better return.

Value-based

Estimate value function of the optimal policy (no explicit policy).

1. Learn/evaluate V, Q or A (no explicit policy).
2. Improve policy by maximizing over Q.

Actor-critic

Value-based + policy gradient.

1. Learn V, Q or A.
2. Learn policy: take a gradient step to get a better return.

Model-based

1. Learn transition model
2. Use it for planning, improve a policy or learn a value function

Policy gradients (1)

$$\theta^* = \text{argmax}_\theta \underbrace{\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]}_{J(\theta)}$$

1. Evaluating the objective: Monte Carlo approximation; we generate samples and average their returns.

$$J(\theta) \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

2. Policy differentiation:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau) R(\tau) d\tau$$

We would like to compute $\nabla_\theta J(\theta)$.

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

Policy gradients (2)

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau) R(\tau) d\tau$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) R(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) R(\tau)]\end{aligned}$$

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

Recall that: $p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

Policy gradients (2)

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau) R(\tau) d\tau$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) R(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) R(\tau)]\end{aligned}$$

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

Recall that: $p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

So, $\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \log [p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)]$

$= \nabla_\theta [\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)]$

Do not depend on θ

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

Policy gradients (2)

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau) r(\tau) d\tau$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) R(\tau)]\end{aligned}$$

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

Recall that: $p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

$$\begin{aligned}\text{So, } \nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log [\, p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)] \\ &= \nabla_\theta \, [\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)] \\ &= \nabla_\theta \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)\end{aligned}$$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

Policy gradients (2)

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau) r(\tau) d\tau$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) R(\tau)]\end{aligned}$$

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

Recall that: $p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$

$$\begin{aligned}\text{So, } \nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log [\, p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)] \\ &= \nabla_\theta \, [\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)] \\ &= \nabla_\theta \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)\end{aligned}$$

$$\text{And } \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\, (\textstyle\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)) \, (\sum_t r(s_t, a_t))] \approx \frac{1}{N} \sum_i (\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})) \, (\sum_t r(s_{i,t}, a_{i,t}))$$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

Policy gradients (3)

REINFORCE algorithm

1. Sample $\{\tau^i\}$ by running the policy
2. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i (\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})) (\sum_t r(s_{i,t}, a_{i,t}))$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

[Williams et al. 1991, Levine & Koltun 2013, Schulman et al. 2017]

Policy gradients (3)

REINFORCE algorithm

1. Sample $\{\tau^i\}$ by running the policy
2. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i (\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})) (\sum_t r(s_{i,t}, a_{i,t}))$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Policy gradient has high variance:

✓ Reduce variance by exploiting causality: "what we do at time step t' does not affect the reward at time step t if t< t' "

✓ Reduce variance by exploiting baselines: "Increase the probabilities of trajectories that are better than average"

Policy gradient is on-policy, i.e. requires sampling new trajectories at each step (not efficient):

✓ Policy gradient with important sampling

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_4_policy_gradient.pdf

[Williams et al. 1991, Levine & Koltun 2013, Schulman et al. 2017]

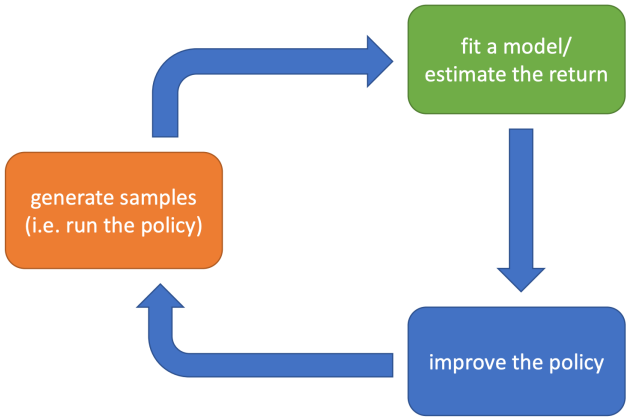
Value-based (1)

Fit Q^π, V^π or A^π and use it as a mechanism for policy improvement.

Policy iteration:

0. Randomly initialize policy.
1. Policy evaluation (value or advantage function):
   → Iteratively apply Bellman expectation equation (V or Q)
2. Policy improvement by acting greedily w.r.t. value or advantage function.

$$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \text{argmax}_a\, Q^\pi(s,a) \\ 0 & \text{otherwise} \end{cases}$$



Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

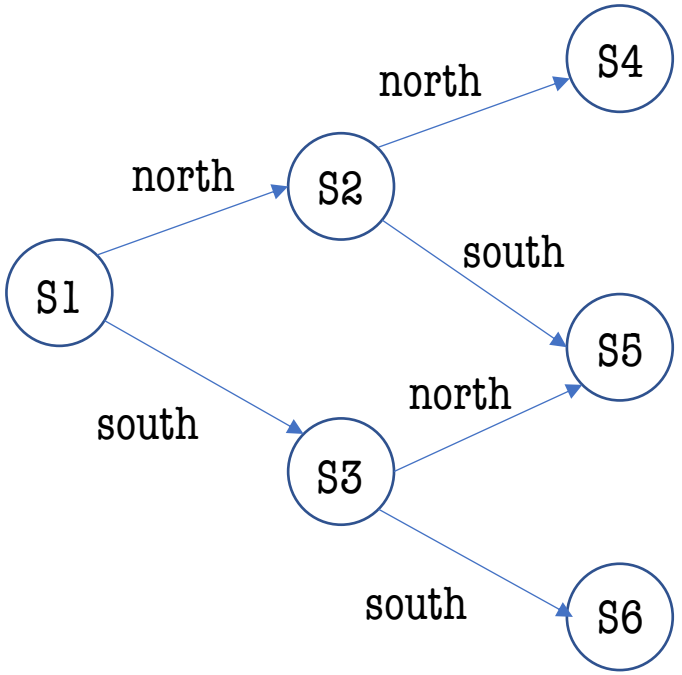
S2. Gold island (+2)

S3. Silver island (+1)

S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)



But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Initial random policy: {N, N, N} (only S1, S2, S3, others are end states)

Based on https://towardsdatascience.com/policy-iteration-in-rl-an-illustration-6d58bdcb87a7

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

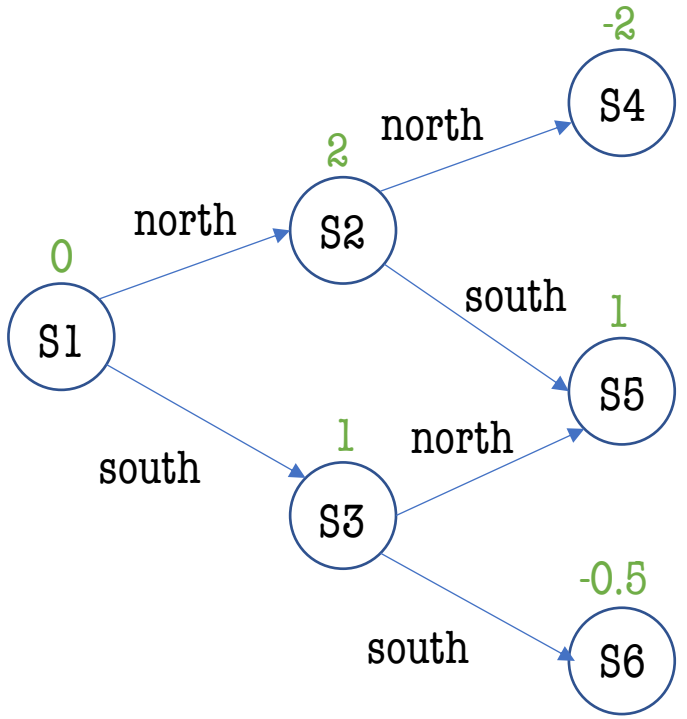
S2. Gold island (+2)

S3. Silver island (+1)

S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)



But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 1.

Policy improvement 1: {N, S, N}

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

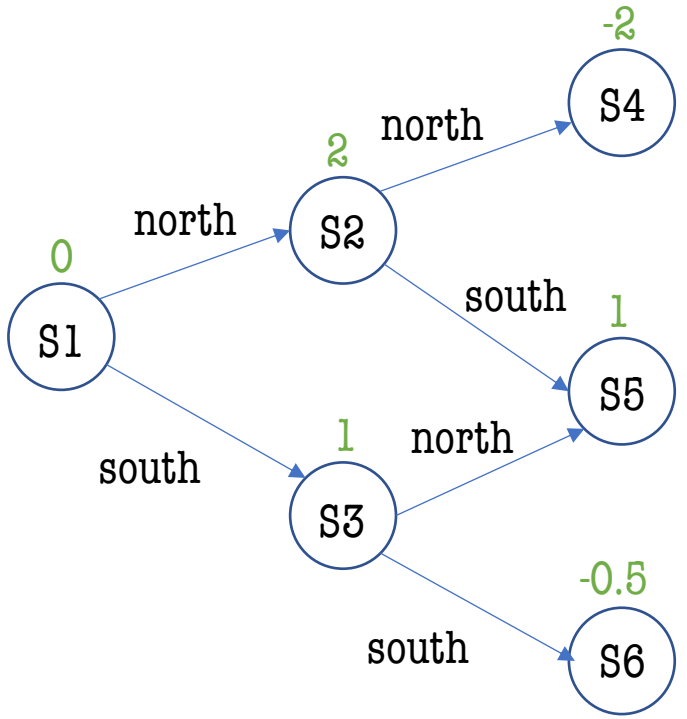
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



S1: 0 + (0.8*2 + 0.2*1) + (0.2*2 + 0.8*1) = 3

Move north Move south

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

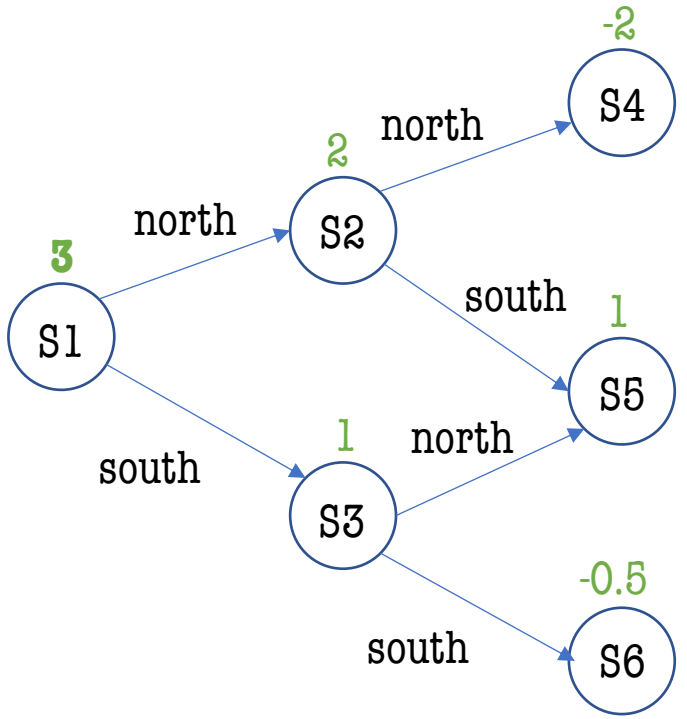
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

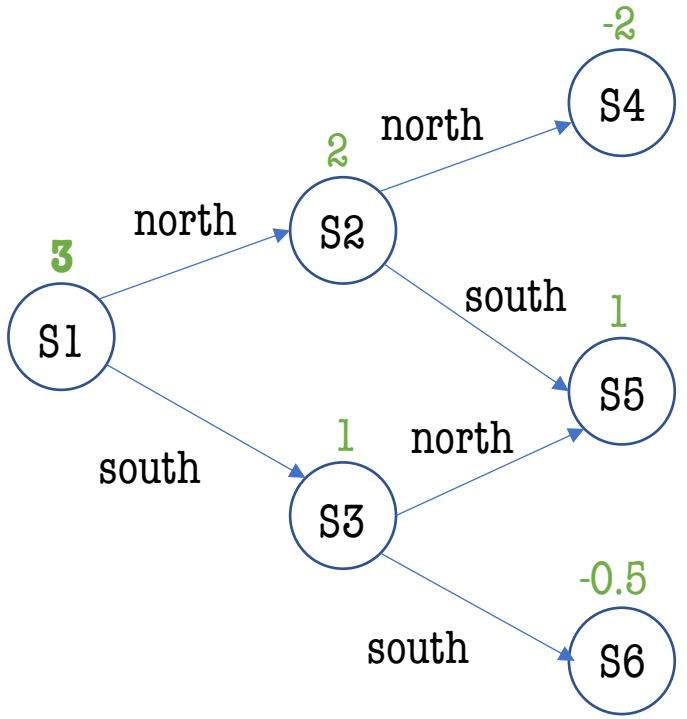
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



S2: 2 + (0.8*(-2) + 0.2*1) + (0.2*(-2) + 0.8*1) = 1

Move north

Move south

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

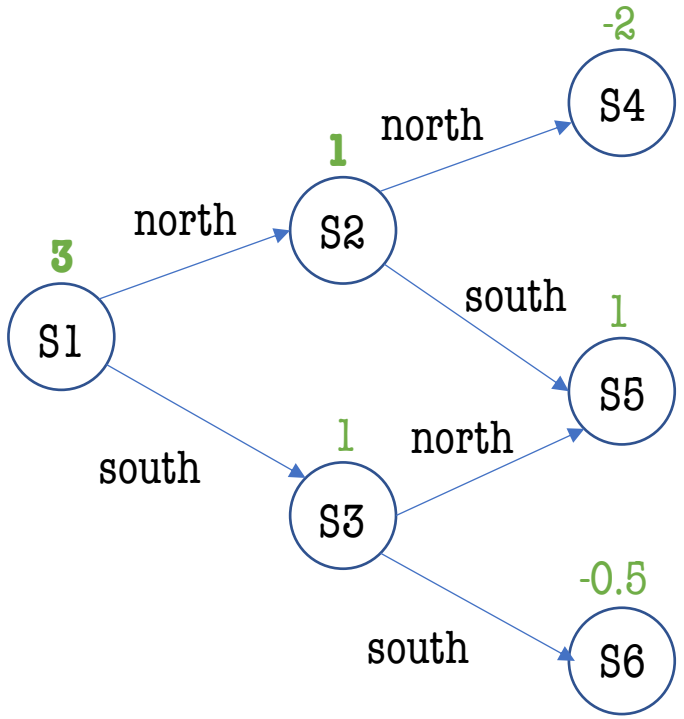
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



S2: 2 + (0.8*(-2) + 0.2*1) + (0.2*(-2) + 0.8*1) = 1

Move north

Move south

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

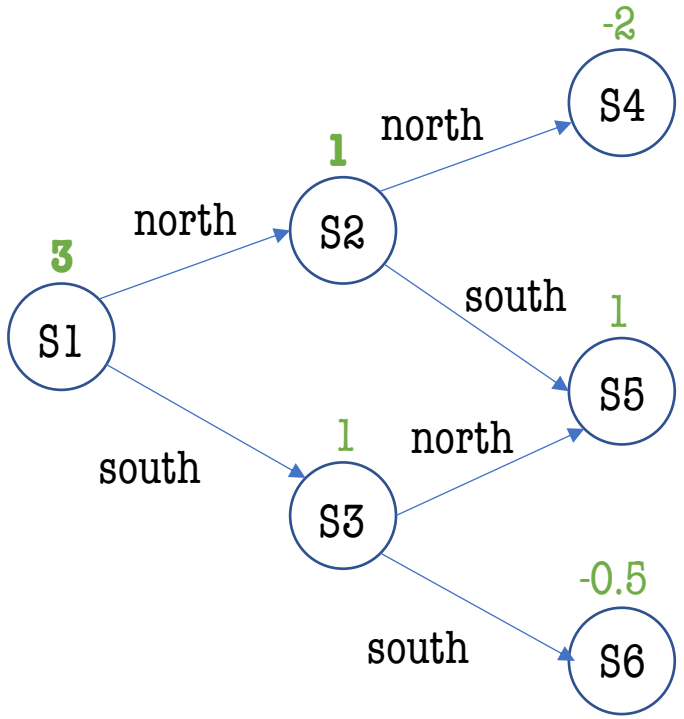
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



S3: 1 + (0.8*1 + 0.2*(-0.5)) + (0.2*1 + 0.8*(-0.5)) = 1.5

Move north

Move south

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

S2. Gold island (+2)

S3. Silver island (+1)

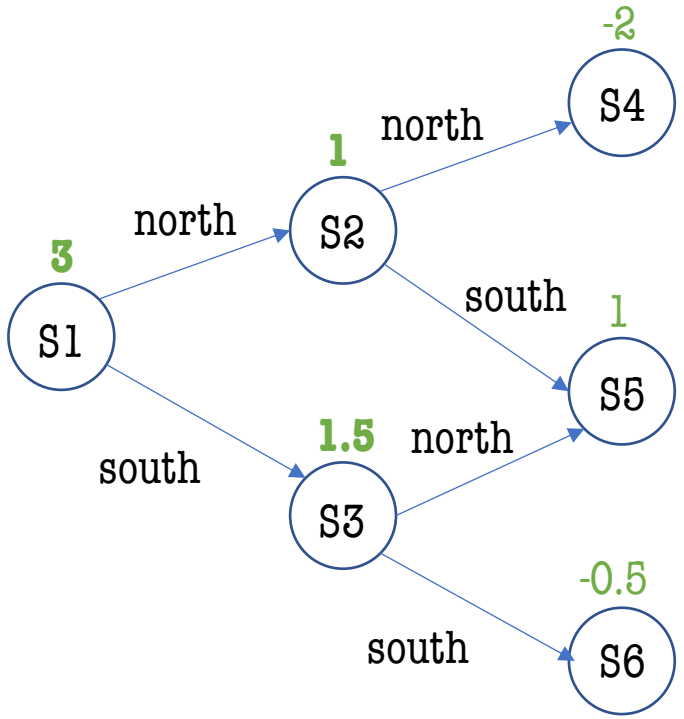
S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)

But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.



S3: 1 + (0.8*1 + 0.2*(-0.5)) + (0.2*1 + 0.8*(-0.5)) = 1.5

Move north Move south

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

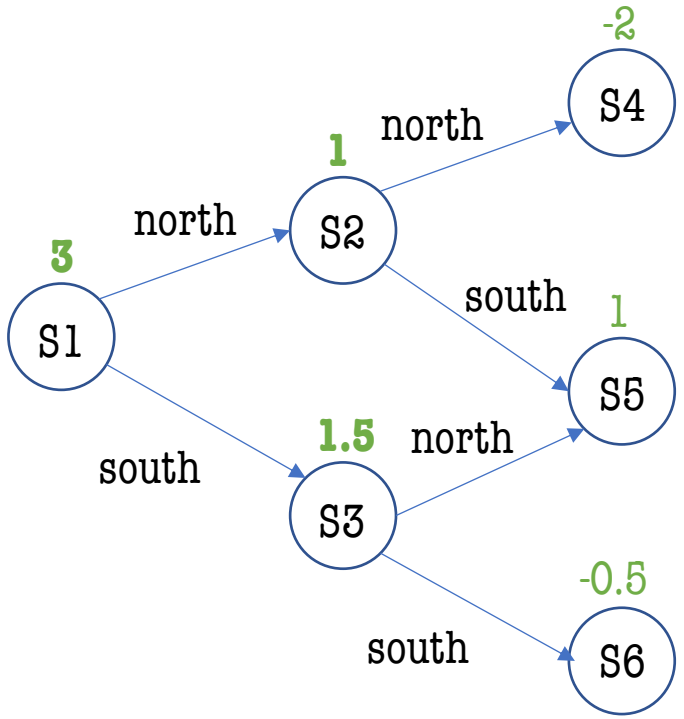
S2. Gold island (+2)

S3. Silver island (+1)

S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)



But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

Policy evaluation step 2.

Policy improvement 2: {S, S, N}

Value-based (2)

Goal: go from start to home with a ship

S1. Start (0)

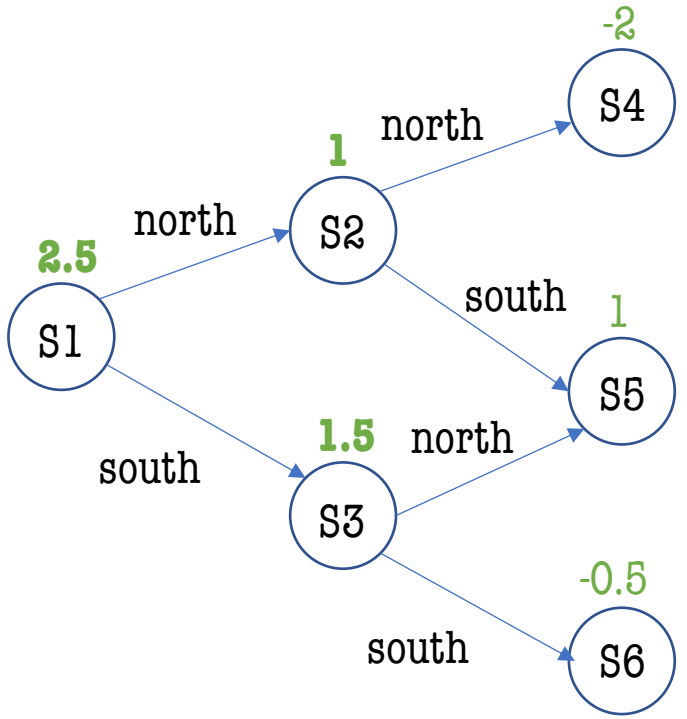
S2. Gold island (+2)

S3. Silver island (+1)

S4. Bermuda triangle (ship is lost forever, -2)

S5. Home (+1)

S6. Prison (crew imprisoned, -0.5)



But... compass is broken – if we attempt to move north, we go north with 0.8 probability and if we attempt to move south, we go south with 0.8 probability.

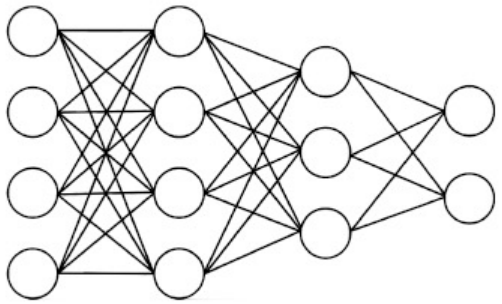
Policy evaluation step 3.

Policy improvement 3: {S, S, N}

What if we have large state and action spaces?

Value-based (2)

Fitted policy iteration: Pose the Q^{π} fitting problem as a supervised regression problem (objective: MSE).

s_t, a_t  $Q_\phi(s_t, a_t)$

Objective:

$$\text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$$

What do we use as target y_i ?

$$y_i \leftarrow r(s_i, a_i) + \gamma \underbrace{\mathbb{E}_{s_i' \sim p(s_i'|s_i, a_i)}[V(s_i')]}_{\approx \max_{a'} Q_\phi(s_i', a_i')}$$

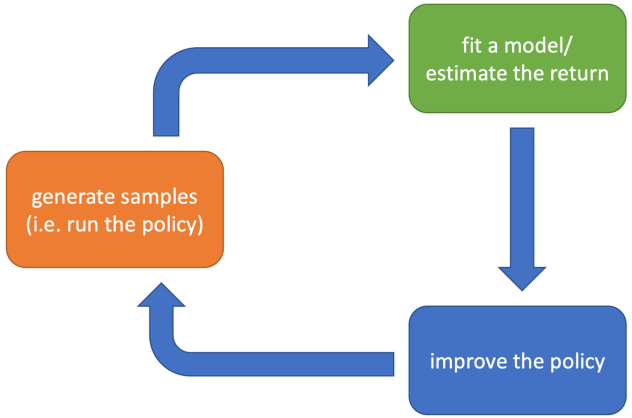
$$\begin{aligned} Q(s_t, a_t) &= \sum_{t=t'}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t] \\ &= r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\mathbb{E}_{a_{t+1} \sim \pi(a_{t+1}|s_{t+1})}[Q(s_{t+1}, a_{t+1}) \mid s_{t+1}]] \\ &= r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V(s_{t+1})] \end{aligned}$$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (3)

Q-learning algorithm

1. Sample $\{(s_i, a_i, s_i', r_i)\}$ by running the policy
2. Set targets $y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$



Then use Q_ϕ as "policy": $\text{argmax}_a Q_\phi(s, a)$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (4)

Q-learning algorithm

1. Sample $\{(s_i, a_i, s_i', r_i)\}$ by running the policy
2. Set targets $y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

Then use Q_ϕ as "policy": $\text{argmax}_a Q_\phi(s, a)$

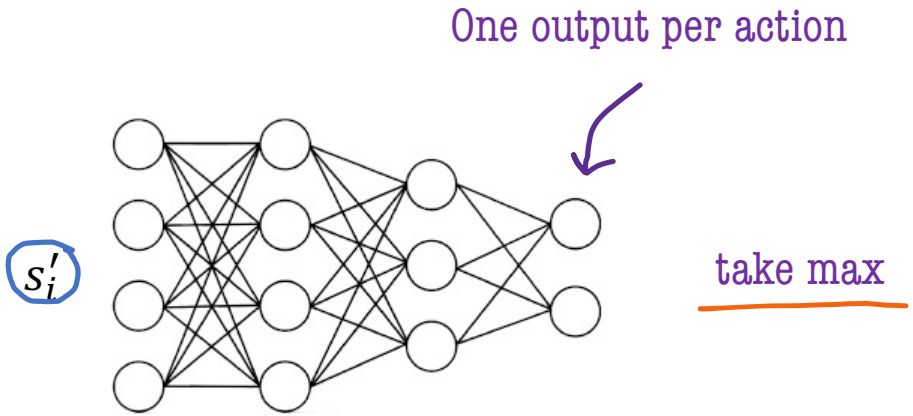
Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (4)

Q-learning algorithm

1. Sample $\{(s_i, a_i, s_i', r_i)\}$ by running the policy
2. Set targets $y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

Then use Q_ϕ as "policy": $\text{argmax}_a Q_\phi(s, a)$

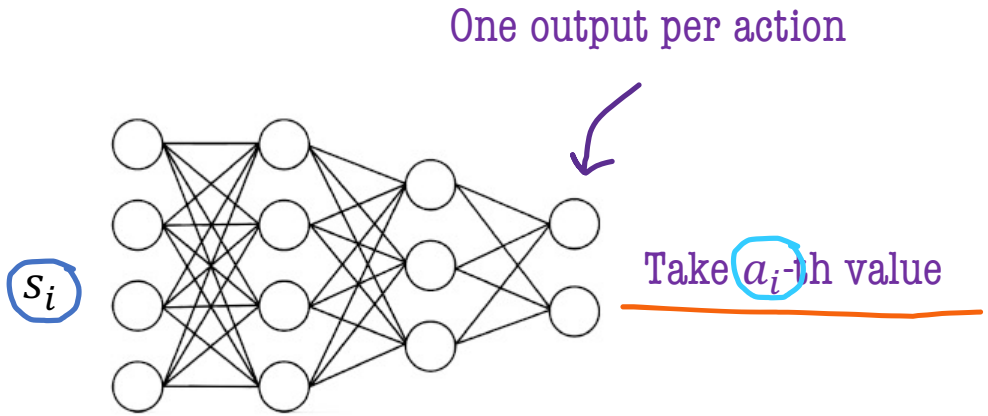


Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (4)

Q-learning algorithm

1. Sample $\{(s_i, a_i, s_i', r_i)\}$ by running the policy
2. Set targets $y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$



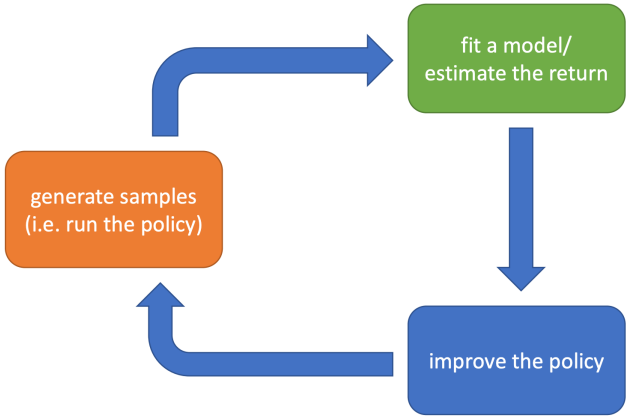
Then use Q_ϕ as "policy": $\text{argmax}_a Q_\phi(s, a)$

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (4)

Q-learning algorithm

1. Sample $\{(s_i, a_i, s_i', r_i)\}$ by running the policy
2. Set targets $y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \text{argmin}_\phi \frac{1}{N} \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$



Then use Q_ϕ as "policy": $\text{argmax}_a Q_\phi(s, a)$

But ...

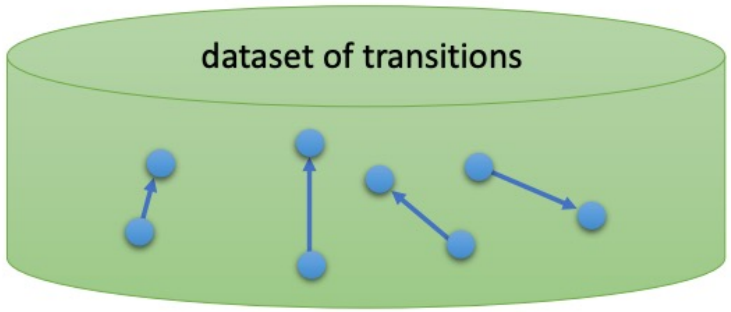
- Correlations between samples
- Non-stationary (moving) targets

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Value-based (5)

DQN

Replay buffer: to mitigate correlation problem



✓ Data in the buffer comes from past experience.

✓ It is important to update the replay buffer periodically with fresh data.

Target Networks: Temporally freeze the parameters of the network to extract the Q targets.

✓ Slow down the pace at which the network parameters (and hence Q targets) move.

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_7_advanced_q_learning.pdf

[Mnih et al., 2013]

Value-based (6)

DQN algorithm

1. Take some action a_i and observe $\{(s_i, a_i, s_i', r_i)\}$, add it to replay buffer β
2. Sample minibatch $\{s_j, a_j, s_j', r_j\}$ from β uniformly
3. Compute $y_j = r_j + \gamma \max_{a_j'} Q_{\phi'}(s_j', a_j')$ [Note: ϕ' are the parameters of the target network]
4. Set $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j)(Q_\phi(s_j, a_j) - y_j)$
5. Update ϕ'

Based on http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_6_value_functions.pdf

Wrap Up

Wrap Up

What is RL?

- Sequential decision making
- RL vs SL
- RL applications
- RL tools

Terminology

- Agent
- Environment
- State/observation
- Action
- Reward
- Transition function
- Policy

RL formalism

- MDP/POMDP
- The RL objective
- Reward/return/value functions (V,Q,A)
- Optimal policy

RL algorithms

- Policy gradients (learn policy, evaluate return)
  - REINFORCE
  - Reducing variance
- Value-based
  - Policy iteration (eval value fct)
  - Q-learning (learn value fct)
  - DQN (replay buffer, target networks)

References

References

ADDITIONAL MATERIALS:

S. Levine - Deep RL course: http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html
D. Silver - RL course: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
Sutton & Barto - Reinforcement learning: an introduction (http://incompleteideas.net/book/the-book-2nd.html)
Francois-Lavet et al. – An introduction to deep RL (https://arxiv.org/abs/1811.12560)

PAPERS:

[Williams, 1991] Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm". (1991)
[Coates et al., 2008] A. Coates, P. Abbeel, A. Y. Ng. "Learning for Control from Multiple Demonstrations". ICML (2008).
[Abbeel et al., 2008] P. Abbeel, A. Coates, T. Hunter, A. Y. Ng. "Autonomous Autorotation of an RC Helicopter". ISR (2008)
[Abbeel et al., 2010] P. Abbeel, A. Coates, A. Y. Ng. "Autonomous Helicopter Aerobatics through Apprenticeship Learning". IJRR (2010).
[Levine & Koltun, 2013] S. Levine, V. Koltun. "Guided policy search: deep RL with importance sampled policy gradient." (2013)

References

[Mnih et al., 2014] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, et al. “Playing Atari with Deep Reinforcement Learning”. (2013).

[Levine et al., 2015] S. Levine*, C. Finn*, T. Darrell, P. Abbeel. “End-to-end training of deep visuomotor policies”. (2015).

[Schulman et al., 2016] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. “Trust Region Policy Optimization”. (2015).

[Mnih et al. 2016] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, et al. “Asynchronous methods for deep reinforcement learning”. (2016).

[Silver et al., 2016] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, et al. “Mastering the game of Go with deep neural networks and tree search”. Nature (2016).

[Gu et al., 2016] S. Gu*, E. Holly*, T. Lillicrap, S. Levine. “Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates”. (2016).

[Mnih et al., 2016] Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu. “Asynchronous methods for deep reinforcement learning: A3C -- parallel online actor-critic”. (2016)

[Schulman et al., 2016] Schulman, Moritz, Jordan, Abbeel. “High-dimensional continuous control using generalized advantage estimation: batch-mode actor-critic with blended Monte Carlo and function approximator returns.” (2016)

[Bojarski et al. 2016] M. Bojarsiki, D. Del Testa, D. Dworakowski, B. Firner, et al. “End to end learning of self-driving cars” (2016)

References

[Schulman et al., 2017] Schulman, Wolski, Dhariwal, Radford, Klimov. “Proximal policy optimization algorithms: deep RL with importance sampled policy gradient”. (2017)

[Gu et al., 2017] Gu, Lillicrap, Ghahramani, Turner. “Q-Prop: sample-efficient policygradient with an off-policy critic: policy gradient with Q-function control variate”. (2017)

[Das et al, 2017] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, D. Batra. “Emboddied Question Answering.” (2017)

[Wu et al., 2018] Y. Wu, Y. Wu, G. Gkioxari, Y. Tian. “Building Generalizable Agents with a Realistic and Rich 3D Environment.” (2018)

Intro to deep reinforcement learning

Master in Computer Vision Barcelona
Adriana Romero
adrianars@fb.com