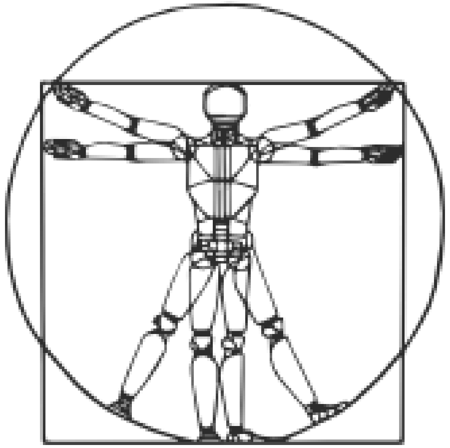
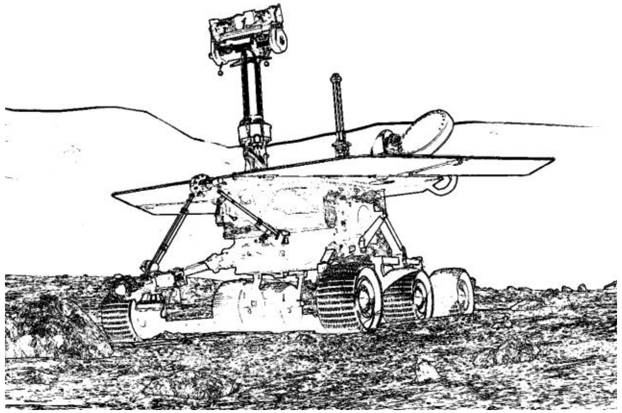




COMP417
Introduction to Robotics and Intelligent Systems

Lecture 23: Intro to Reinforcement Learning

Florian Shkurti

Computer Science Ph.D. student

florian@cim.mcgill.ca

Today’s slides

- Borrow heavily from Pieter Abbeel’s and John Schulman’s presentations.

Useful resources for studying RL

- Rich Sutton's and Andrew Barto's textbook "Reinforcement Learning, An Introduction", 2nd edition. Preprint available for free http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf

- David Silver's RL course https://www.youtube.com/watch?v=2pWv7GOvuf0 (mostly focuses on discrete state and action spaces, but very much applicable to robotics)

- John Schulman's 4-5hr long (in total) summer school videos https://www.youtube.com/watch?v=aUrX-rP_ss4
  focusing mostly on policy gradients.

- Joelle Pineau's, Doina Precup's and Pieter Abbeel's talks at UdeM's deep learning summer school: http://videolectures.net/deeplearning2016_pineau_reinforcement_learning/
  http://videolectures.net/deeplearning2016_pineau_advanced_topics/
  http://videolectures.net/deeplearning2016_abbeel_deep_reinforcement/
  http://videolectures.net/deeplearning2016_precup_advanced_topics/

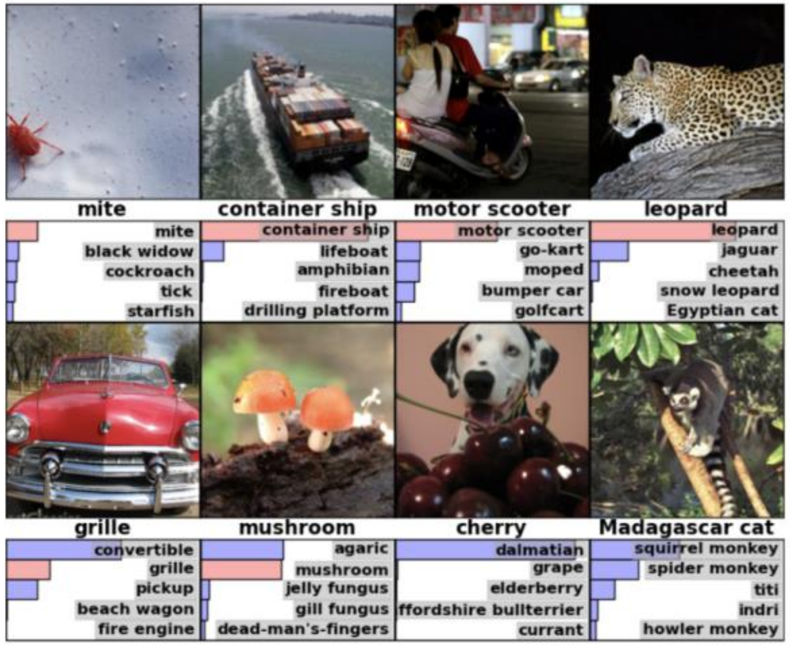
- Doina Precup's COMP767: Reinforcement Learning course
  http://www.cs.mcgill.ca/~dprecup/courses/RL/lectures.html

Learning Paradigms

- Supervised Learning
  - Have access to labeled training data (x, y) where x ~ p(x)
  - Learn a function f : X → Y from training data such that f(x’) correctly predicts y’ for unseen pairs (x’, y’) where x’ ~ p(x)

Learning Paradigms

- Supervised Learning
  - Have access to labeled training data (x, y) where x ~ p(x)
  - Learn a function f : X → Y from training data such that f(x') correctly predicts y' for unseen pairs (x', y') where x' ~ p(x)



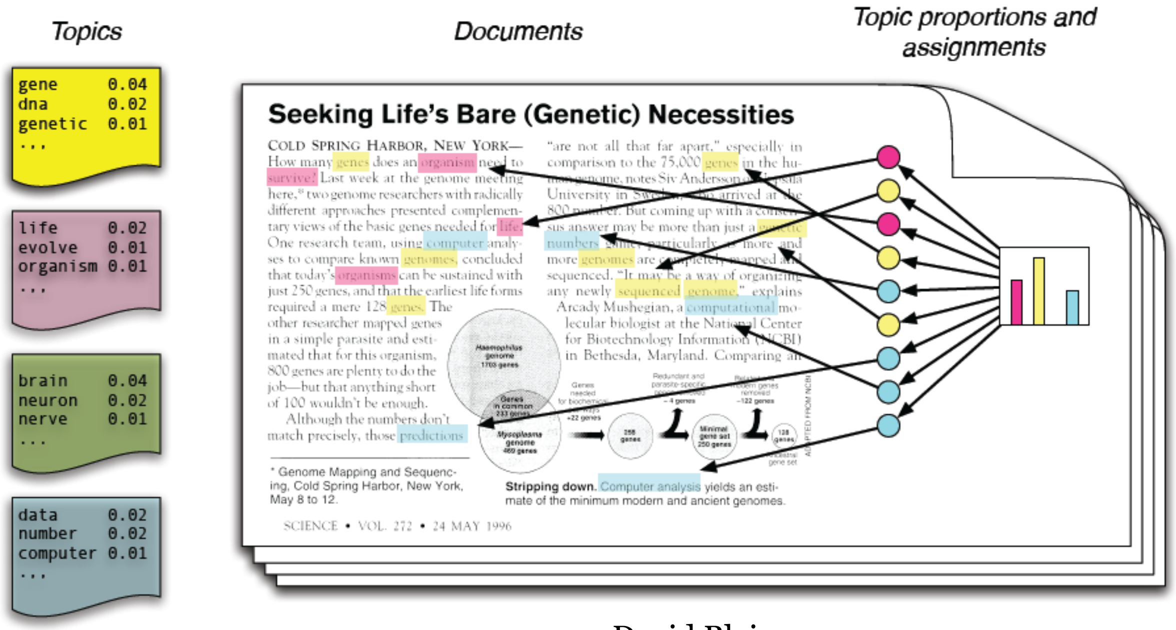
ImageNet Challenge(s)

1000 object classes
1.2M training examples
100K test examples

- classification
- localization
- scene parsing
- detection in video

Learning Paradigms

- Unsupervised Learning
  - Have access to unlabeled data x
  - Learn “hidden” structure of the data. For example:
    - learn distribution of data p(x)   (density estimation)
    - learn factors explaining the data
    - cluster analysis
    - dimensionality reduction



David Blei, 2011

Learning Paradigms

- Semi-supervised Learning
  - Have access to many unlabeled data x, and only a few labeled data x’

Learning Paradigms

- Imitation Learning
  - Expert/teacher/demonstrator shows good actions/controllers
  - Agent needs to optimize its actions without veering too far off the expert’s trajectories.

Learning Paradigms

- Imitation Learning
  - Expert/teacher/demonstrator shows good actions/controllers
  - Agent needs to optimize its actions without veering too far off the expert’s trajectories.

- Transfer Learning
  - Policies learnt in one domain can be morphed and applied to another domain.
  - E.g. learn policies in a simulator and apply them in the real-world after some refinement/transformation

Learning Paradigms

- Reinforcement Learning
  - Agent interacts with environment by applying actions
  - Environment responds by rewarding or punishing the agent
  - Agent has to optimize actions in order to maximize expected cumulative rewards (minimize punishment) in the future.

Different than supervised learning because the agent’s actions potentially change the distribution of the data being observed. Also, in RL, time-consecutive observations can be dependent.

Learning Paradigms

- Reinforcement Learning + Game Theory
  - Agent interacts with environment by applying actions.
  - Other individual agents with their own objectives apply actions.
  - Each agent receives reward or punishment.

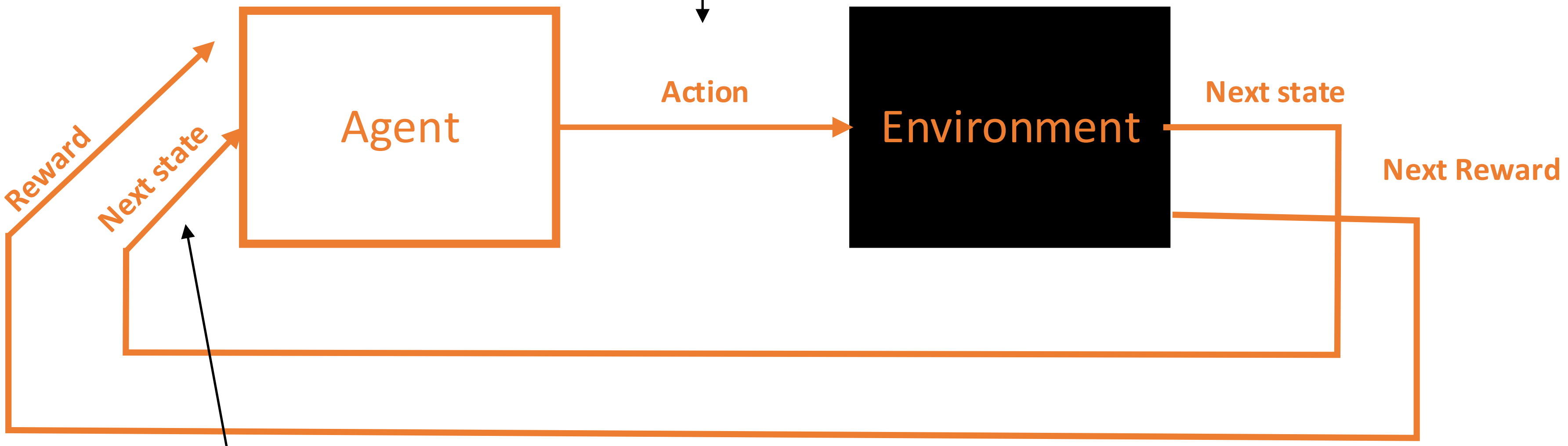
  - Agent has to optimize actions in order to maximize expected cumulative rewards (minimize punishment) in the future, and take other agents' actions into account.

Reinforcement Learning

Policy

$p_\theta(u_t|x_t)$

Model-free RL: reward and dynamics are unknown
Model-based RL: reward and dynamics are known



Agent's state is observed directly at each time step.

Reinforcement Learning

Today's class

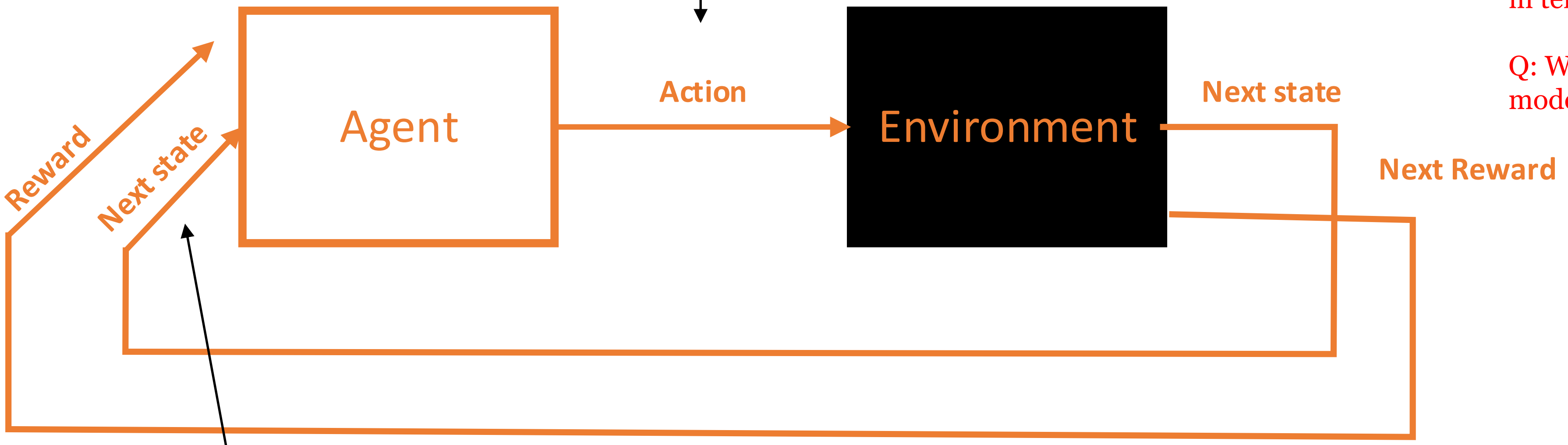
Model-free RL: reward and dynamics are unknown
Model-based RL: reward and dynamics are known

Most useful for robotics
in terms of efficient learning.

Q: What happens when the model is wrong/imperfect?

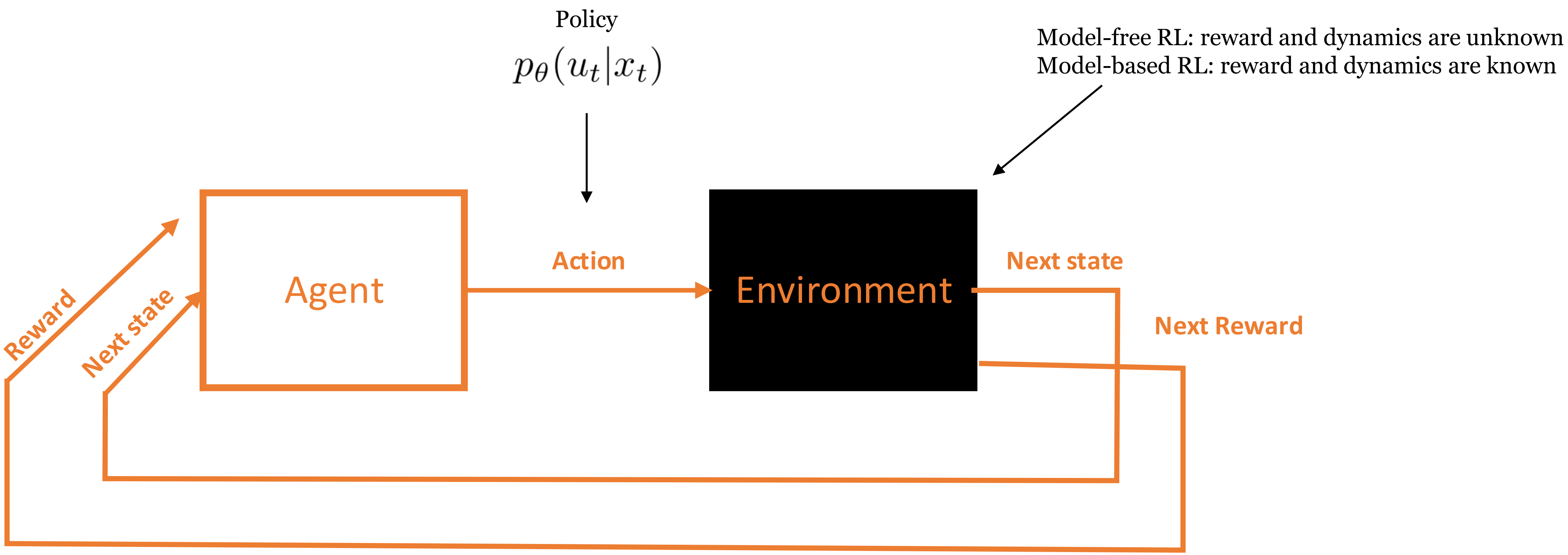
Policy

$p_\theta(u_t|x_t)$



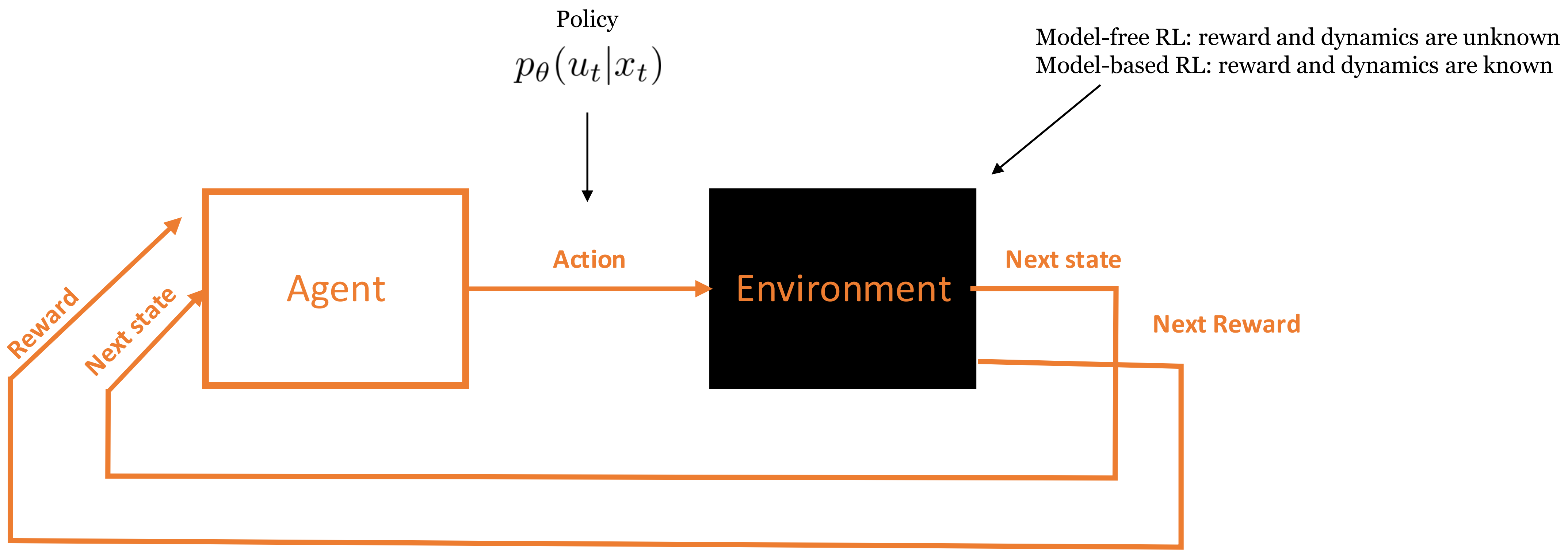
Agent's state is observed directly
at each time step.

Reinforcement Learning



One of the main problems in RL: **temporal credit assignment**
Which action was responsible for high reward? Rewards may be delayed or sparse.

Reinforcement Learning



Another main problem in RL: **exploration vs. exploitation tradeoff**
Should the agent select its currently most promising action or should it randomly search (explore) for a better alternative?

Markov Decision Processes (MDP)

Frameworks for optimal decision-making under uncertainty

Discrete states x_t
Discrete actions u_t

Uncertain dynamics $p(x_t|x_{t-1}, u_{t-1})$
State is directly observed at each step, without uncertainty

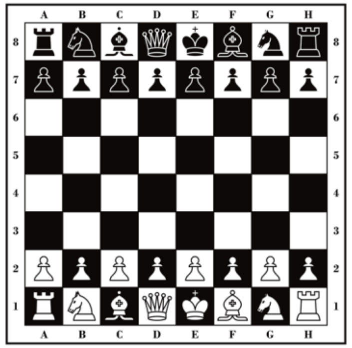
Agent receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition.

Goal: find a policy $p(u_t|x_t) = \pi^*(u_t|x_t)$ that maximizes cumulative expected reward

Time horizon for optimization.
Can be infinite.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \; \mathbb{E}[\sum_{t=0}^{H} \gamma^t R_t \mid \pi]$$

Discount factor in [0,1) specifies
later rewards are less valued than
earlier ones.



Markov Decision Processes (MDP)

Frameworks for optimal decision-making under uncertainty

Discrete states x_t
Discrete actions u_t

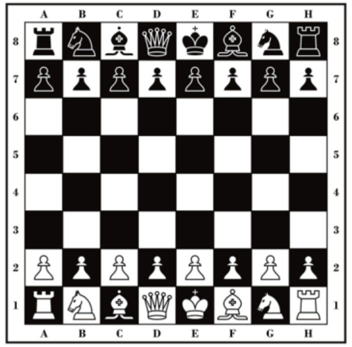
Uncertain dynamics $p(x_t|x_{t-1}, u_{t-1})$
State is directly observed at each step, without uncertainty

Agent receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition.

Goal: find a policy $p(u_t|x_t) = \pi^*(u_t|x_t)$ that maximizes cumulative expected reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}[\sum_{t=0}^{H} \gamma^t R_t \mid \pi]$$

With respect to future states



Partially-Observable Markov Decision Processes (POMDP)

Frameworks for optimal decision-making under uncertainty

Like MDP, but state is not directly observed. Instead, observation model $p(z_t|x_t)$

Agent knows action-observation history $h_t = (u_0, z_0, u_1, z_1, ..., u_t, z_t)$

Maintains and updates belief $b_t(x_t|h_t)$ using Bayes' filter

Receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition, but when planning it can only weigh the reward by the probability of being at a state (since it does not observe it), so we define

$$r_t(h_t, u_t) = \sum_{x_t} b_t(x_t|h_t) R_t$$

Partially-Observable Markov Decision Processes (POMDP)

Frameworks for optimal decision-making under uncertainty

Like MDP, but state is not directly observed. Instead, observation model $p(z_t|x_t)$

Agent knows action-observation history $h_t = (u_0, z_0, u_1, z_1, ..., u_t, z_t)$

Maintains and updates belief $b_t(x_t|h_t)$ using Bayes' filter

Receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition, but when planning it can only weigh the reward by the probability of being at a state (since it does not observe it), so we define

$$r_t(h_t, u_t) = \sum_{x_t} b_t(x_t|h_t) R_t$$

Goal: find a policy $p(u_t|h_t) = \pi^*(u_t|h_t)$ that maximizes cumulative expected reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \; \mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t \mid \pi, b_0]$$

Partially-Observable Markov Decision Processes (POMDP)

Frameworks for optimal decision-making under uncertainty

Like MDP, but state is not directly observed. Instead, observation model $p(z_t|x_t)$

Agent knows action-observation history $h_t = (u_0, z_0, u_1, z_1, ..., u_t, z_t)$

Maintains and updates belief $b_t(x_t|h_t)$ using Bayes' filter

Receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition, but when planning it can only weigh the reward by the probability of being at a state (since it does not observe it), so we define

$$r_t(h_t, u_t) = \sum_{x_t} b_t(x_t|h_t) R_t$$

Goal: find a policy $p(u_t|h_t) = \pi^*(u_t|h_t)$ that maximizes cumulative expected reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t \mid \pi, b_0]$$

With respect to future states and future observations

Deisenroth, Fox, Rasmussen, 2011

Breakout, Atari

DQN, Volodymir Mnih et al., 2013

Sergey Levine, Chelsea Finn, Trevor Darrell, Pieter Abbeel, 2015

Today: two algorithms

- Cross-entropy method (gradient-free)
- Policy gradient

- Idea: optimize expected reward over policy parameters θ

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \; \mathbb{E}[\sum_{t=0}^{H} \gamma^t R_t \mid \pi_\theta]$$

without using the dynamics model.

Cross-Entropy Method

- Model components of $\theta \in \mathbb{R}^d$ as Gaussians
- Initialize $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$
- For iteration 1,2,...
  - Collect n samples of the policy parameters $\theta_i \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$

Cross-Entropy Method

- Model components of $\theta \in \mathbb{R}^d$ as Gaussians
- Initialize $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$
- For iteration 1,2,…
  - Collect n samples of the policy parameters $\theta_i \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$
  - Evaluate the policy for each of those parameters $\theta_i \to \mathcal{R}_i = \sum_{t=0}^{H} \gamma^t R_t$
  - Select the top performing p% of parameters (e.g. p=20)

Cross-Entropy Method

- Model components of $\theta \in \mathbb{R}^d$ as Gaussians
- Initialize $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$
- For iteration 1,2,…
  - Collect n samples of the policy parameters $\theta_i \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$
  - Evaluate the policy for each of those parameters $\theta_i \rightarrow \mathcal{R}_i = \sum_{t=0}^{H} \gamma^t R_t$
  - Select the top performing p% of parameters (e.g. p=20)
  - Fit new Gaussian to top performing parameters and update μ, σ

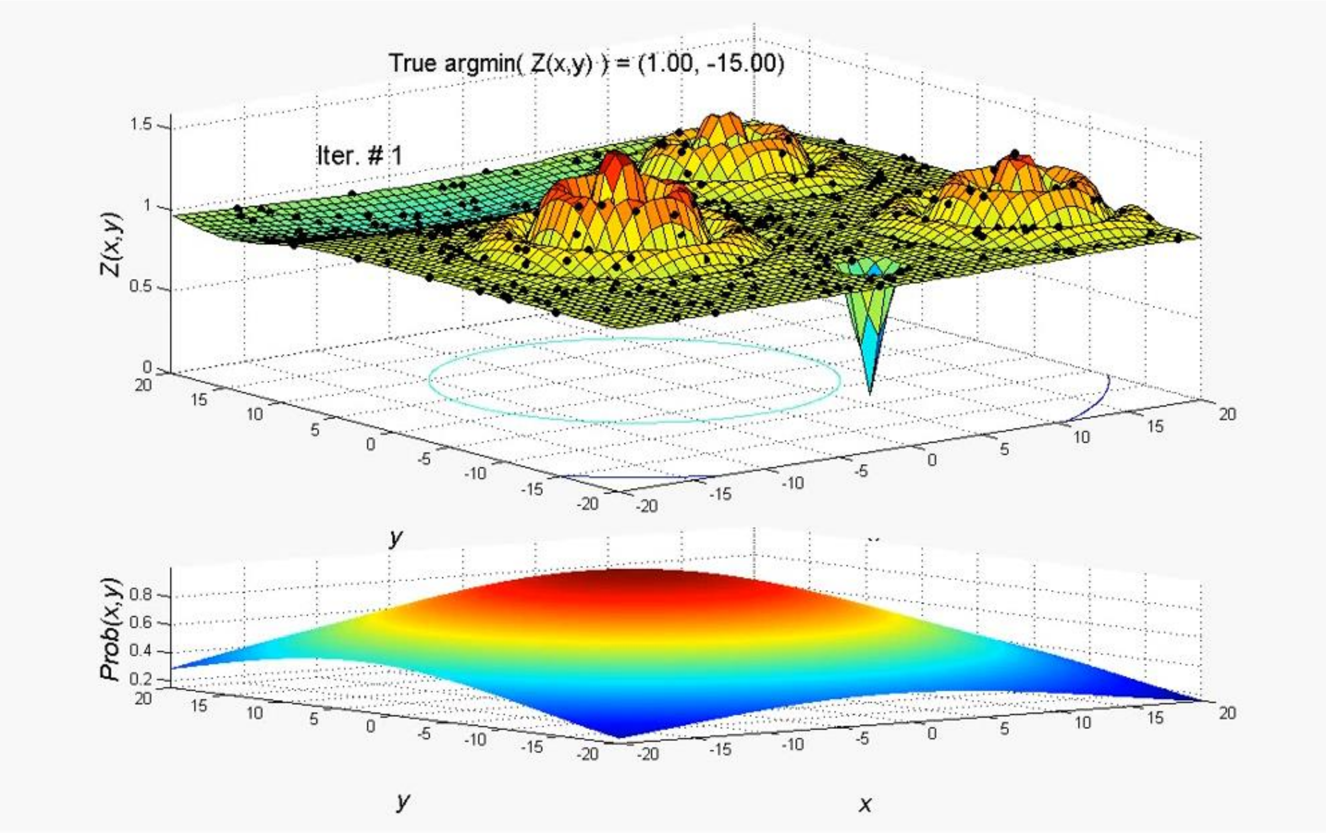
- Return final μ

Cross-Entropy Method

- Model components of $\theta \in \mathbb{R}^d$ as Gaussians
- Initialize $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$
- For iteration 1,2,...
  - Collect n samples of the policy parameters $\theta_i \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$
  - Evaluate the policy for each of those parameters $\theta_i \rightarrow \mathcal{R}_i = \sum_{t=0}^{H} \gamma^t R_t$
  - Select the top performing p% of parameters (e.g. p=20)
  - Fit new Gaussian to top performing parameters and update μ, σ

- Return final μ

No gradients were used


True argmin(Z(x,y)) = (1.00, -15.00)
Iter. # 1
Z(x,y)
y
x
Prob(x,y)
y
x

(Score Function) Policy Gradient

- Consider the trajectory as a random variable

$$\tau = (x_0, u_0, r_0, x_1, u_1, ..., x_{H-1}, u_{H-1}, r_{H-1}, x_H)$$

(Score Function) Policy Gradient

- Consider the trajectory as a random variable

$$\tau = (x_0, u_0, r_0, x_1, u_1, ..., x_{H-1}, u_{H-1}, r_{H-1}, x_H)$$

- What is the probability of such a trajectory given a policy?

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t)\, p(x_{t+1}|x_t, u_t)$$

(Score Function) Policy Gradient

- Consider the trajectory as a random variable

$$\tau = (x_0, u_0, r_0, x_1, u_1, ..., x_{H-1}, u_{H-1}, r_{H-1}, x_H)$$

- What is the probability of such a trajectory given a policy?

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t)\, p(x_{t+1}|x_t, u_t)$$

- Overload reward notation to include trajectories: $R(\tau) = \sum_{t=0}^{H-1} R(x_t, u_t, x_{t+1})$

(Score Function) Policy Gradient

- Consider the trajectory as a random variable

$$\tau = (x_0, u_0, r_0, x_1, u_1, ..., x_{H-1}, u_{H-1}, r_{H-1}, x_H)$$

- What is the probability of such a trajectory given a policy?

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t)\, p(x_{t+1}|x_t, u_t)$$

- Overload reward notation to include trajectories: $R(\tau) = \sum_{t=0}^{H-1} R(x_t, u_t, x_{t+1})$

- Objective function to maximize:

$$J(\theta) = \mathbb{E}[R(\tau)|\pi_\theta] = \sum_\tau R(\tau) p(\tau|\theta)$$

(Score Function) Policy Gradient

- To maximize $J(\theta) = \mathbb{E}[R(\tau)|\pi_\theta] = \sum_\tau R(\tau)p(\tau|\theta)$ we take its derivative:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau R(\tau)p(\tau|\theta)$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_\tau R(\tau) p(\tau|\theta) \\
&= \sum_\tau R(\tau) \nabla_\theta\, p(\tau|\theta)
\end{aligned}
$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_\tau R(\tau)\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \nabla_\theta\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \frac{p(\tau|\theta)}{p(\tau|\theta)} \nabla_\theta\, p(\tau|\theta)
\end{aligned}
$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_\tau R(\tau)\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \nabla_\theta\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \frac{p(\tau|\theta)}{p(\tau|\theta)} \nabla_\theta\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, p(\tau|\theta) \frac{\nabla_\theta p(\tau|\theta)}{p(\tau|\theta)} \\
&= \sum_\tau R(\tau)\, p(\tau|\theta)\, \boxed{\nabla_\theta \log p(\tau|\theta)}
\end{aligned}
$$

Score Function

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_\tau R(\tau)\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \nabla_\theta\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, \frac{p(\tau|\theta)}{p(\tau|\theta)} \nabla_\theta\, p(\tau|\theta) \\
&= \sum_\tau R(\tau)\, p(\tau|\theta) \frac{\nabla_\theta p(\tau|\theta)}{p(\tau|\theta)} \\
&= \sum_\tau R(\tau)\, p(\tau|\theta)\, \nabla_\theta \log p(\tau|\theta) \\
&= \mathbb{E}_{\tau\sim p(\tau|\theta)} \left[R(\tau)\, \nabla_\theta \log p(\tau|\theta)\right]
\end{aligned}$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative $\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \; \nabla_\theta \log p(\tau|\theta) \right]$

- We can simplify the score function further. Recall that

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t) \; p(x_{t+1}|x_t, u_t)$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative $\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \; \nabla_\theta \log p(\tau|\theta) \right]$

- We can simplify the score function further. Recall that

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t) \; p(x_{t+1}|x_t, u_t)$$

$$\log p(\tau|\theta) = \log p(x_0) + \sum_{t=0}^{H-1} \log \pi_\theta(u_t|x_t) + \log p(x_{t+1}|x_t, u_t)$$

(Score Function) Policy Gradient

- To maximize $J(\theta)$ we take its derivative $\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \, \nabla_\theta \log p(\tau|\theta) \right]$

- We can simplify the score function further. Recall that

$$p(\tau|\theta) = p(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t) \, p(x_{t+1}|x_t, u_t)$$

$$\log p(\tau|\theta) = \log p(x_0) + \sum_{t=0}^{H-1} \log \pi_\theta(u_t|x_t) + \log p(x_{t+1}|x_t, u_t)$$

$$\nabla_\theta \log p(\tau|\theta) = \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t)$$

The derivative of the objective function does not depend on the dynamics!

(Score Function) Policy Gradient Algorithm

- To maximize $J(\theta)$ we take its derivative

$$\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau)\ \nabla_\theta \log p(\tau|\theta)\right]$$ where $$\nabla_\theta \log p(\tau|\theta) = \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t)$$

Algorithm:

- Generate m trajectories τ_i using policy $\pi_\theta(u_t|x_t)$
- Compute $\nabla_\theta \log p(\tau_i|\theta)$ for each trajectory
- Return empirical average $\frac{1}{m}\sum_{i=1}^{m} \left[R(\tau_i)\ \nabla_\theta \log p(\tau_i|\theta)\right]$

(Score Function) Policy Gradient Algorithm

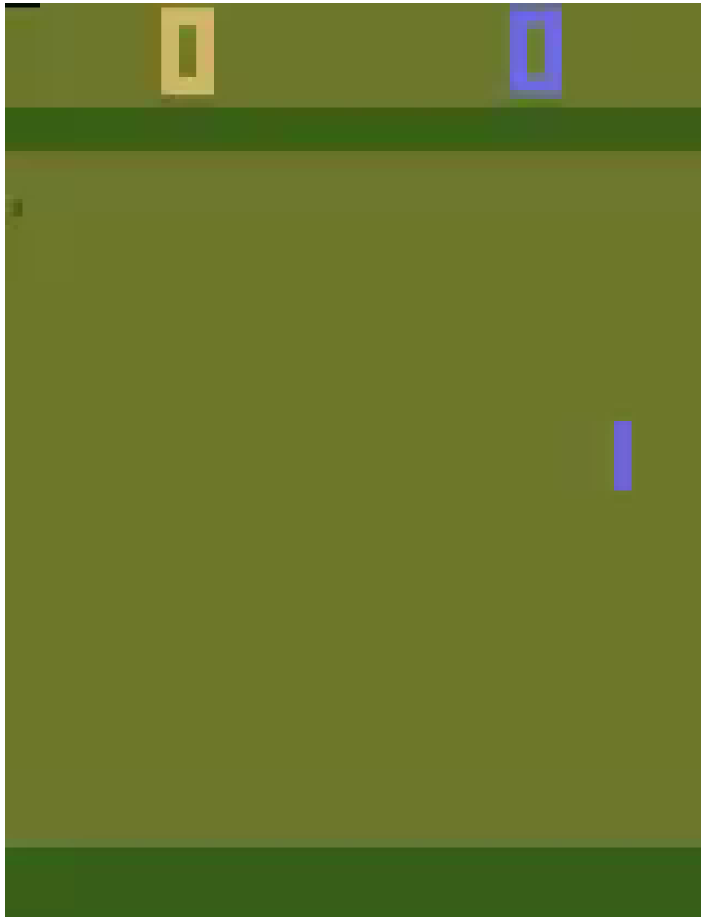
- To maximize $J(\theta)$ we take its derivative

$$\nabla_\theta J(\theta) \quad = \quad \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \, \nabla_\theta \log p(\tau|\theta) \right]$$ where $$\nabla_\theta \log p(\tau|\theta) = \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t)$$

Algorithm:

- Generate m trajectories τ_i by sampling actions from policy $\pi_\theta(u_t|x_t)$
- Compute $\nabla_\theta \log p(\tau_i|\theta)$ for each trajectory
- Return empirical average $\frac{1}{m} \sum_{i=1}^{m} \left[R(\tau_i) \, \nabla_\theta \log p(\tau_i|\theta) \right]$

This algorithm ends up having high variance. Additional tricks are needed to make it work in practice, but they are outside the scope of this lecture.


0
0