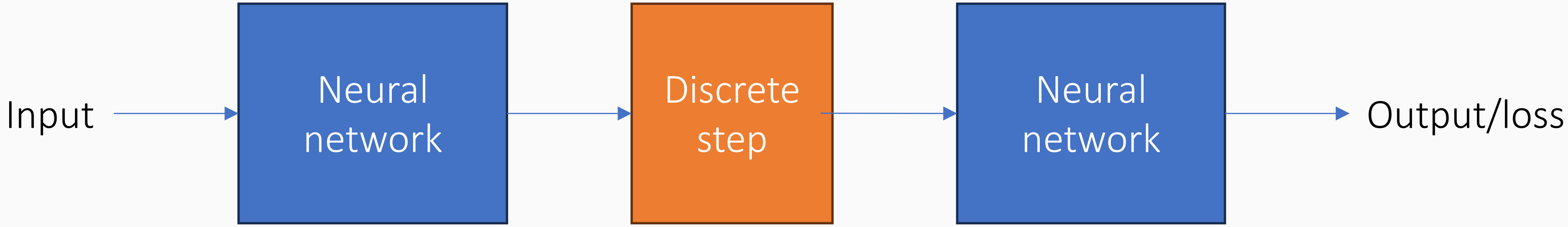
Learning with symbols within neural networks

Neuro-symbolic modeling

THE UNIVERSITY OF UTAH

Neural networks containing discrete elements



Let's see some examples

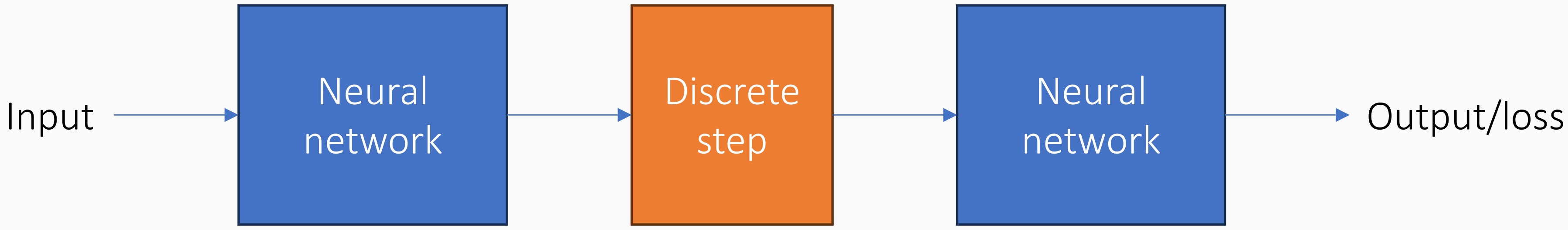
This lecture

- Motivating examples
- The straight-through estimator
- The Gumbel trick
- REINFORCE

(others if time permits)

Not all these approaches are always applicable

Suppose the discrete step is opaque



If we have no idea or control about the inner workings of the discrete step, then we will need to use ideas from reinforcement learning

Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

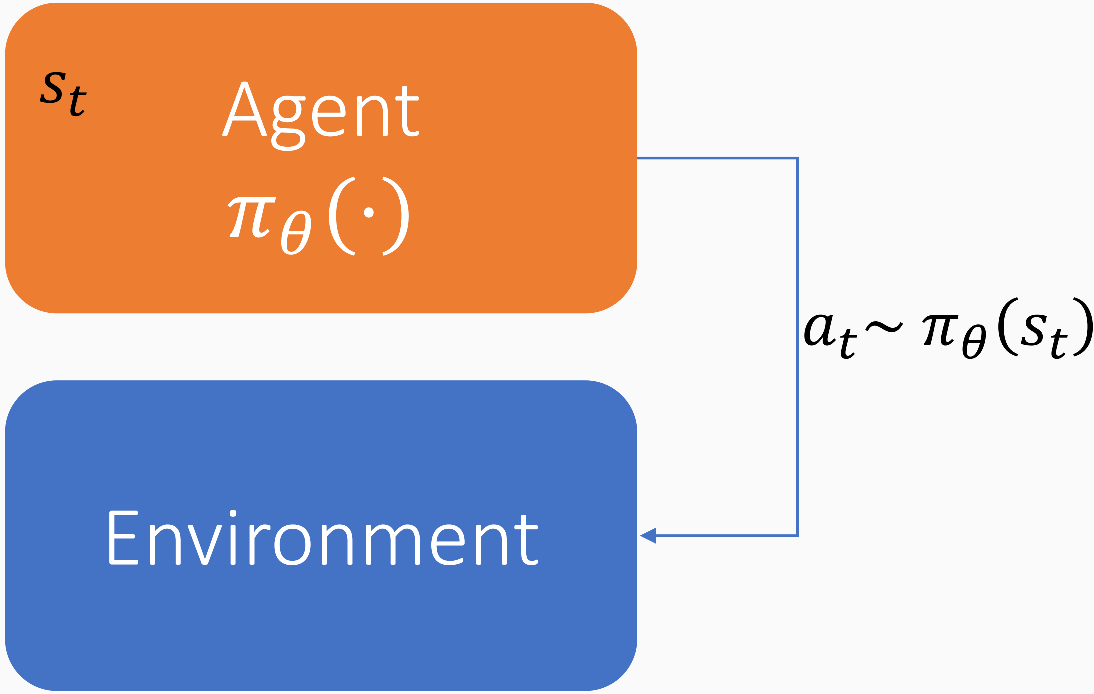
At any step t, the agent exists in a state s_t

Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t, the agent exists in a state s_t

In state s_t at time step t, the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$



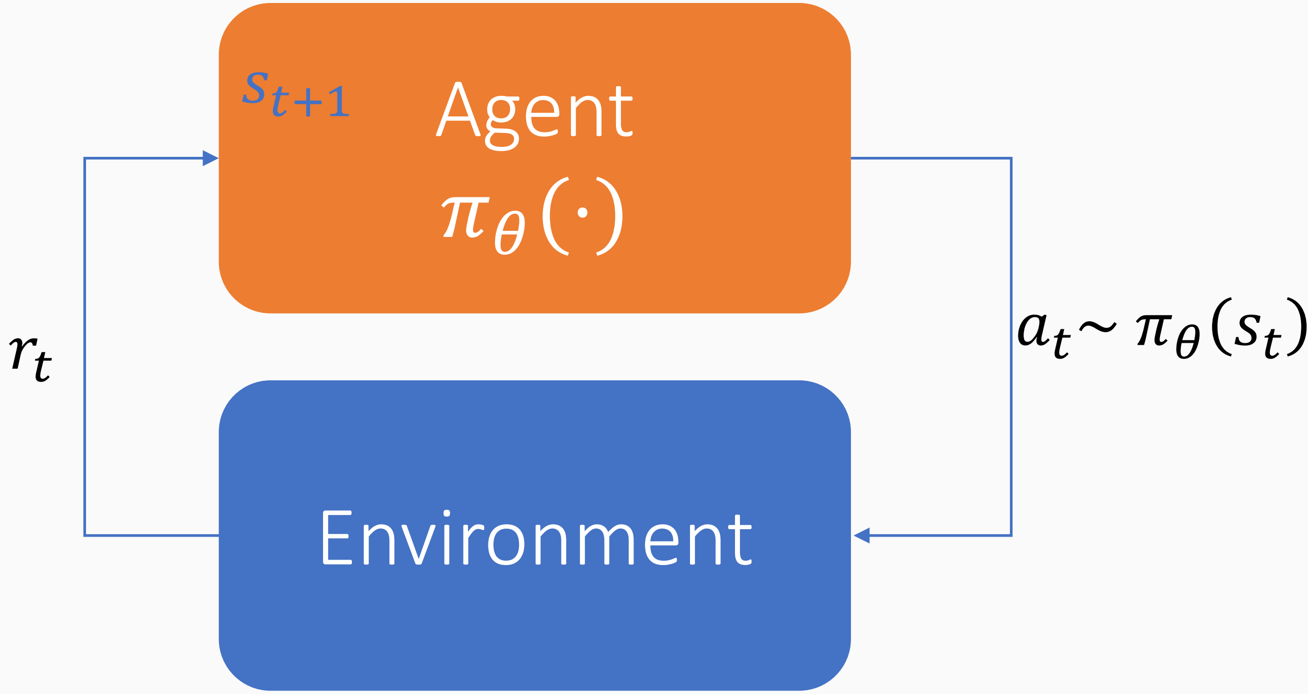
Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t, the agent exists in a state s_t

In state s_t at time step t, the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$

The environment returns a reward r_t and takes the agent to the new state s_{t+1}



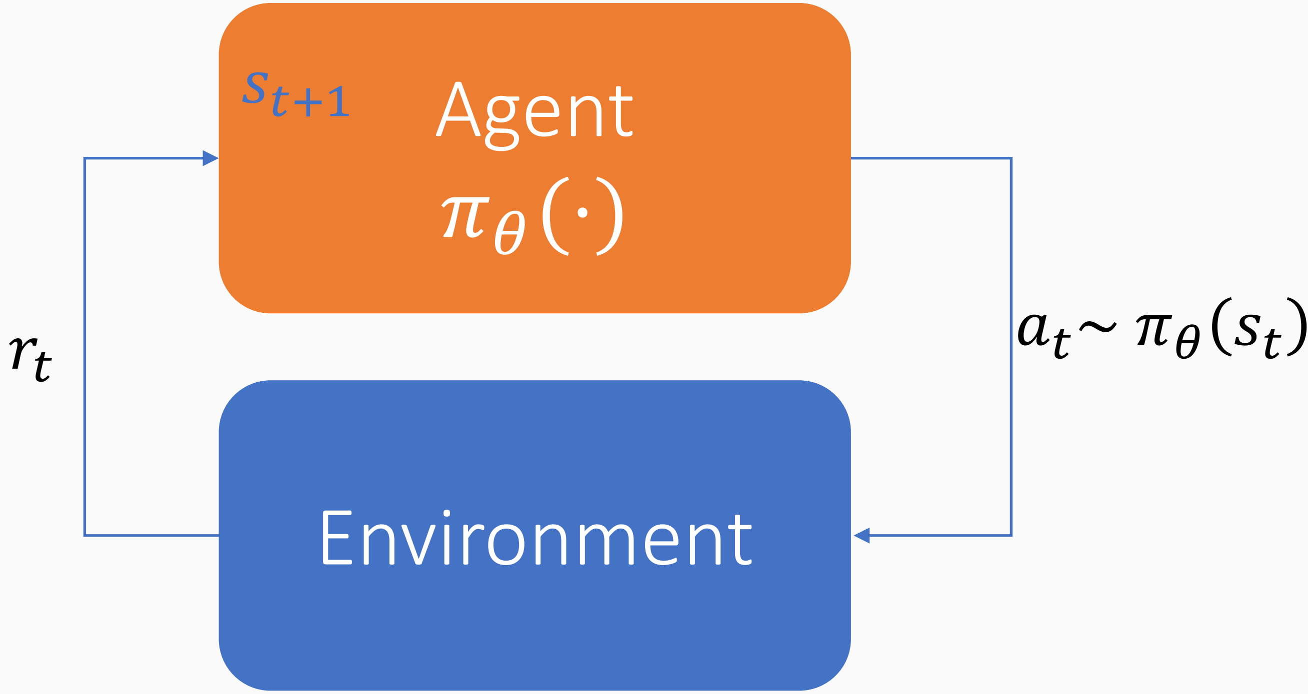
Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t, the agent exists in a state s_t

In state s_t at time step t, the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$

The environment returns a reward r_t and takes the agent to the new state s_{t+1}

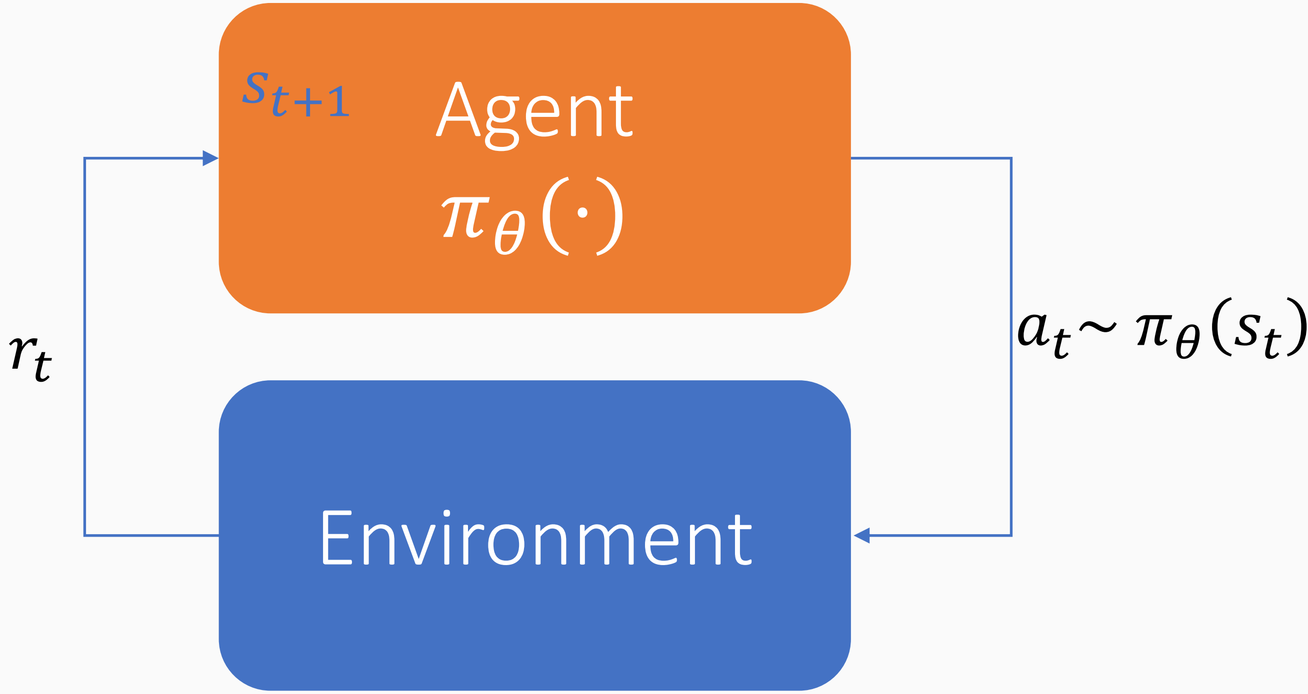


Quite an open-ended learning paradigm

Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

What are some examples of agents and environments?

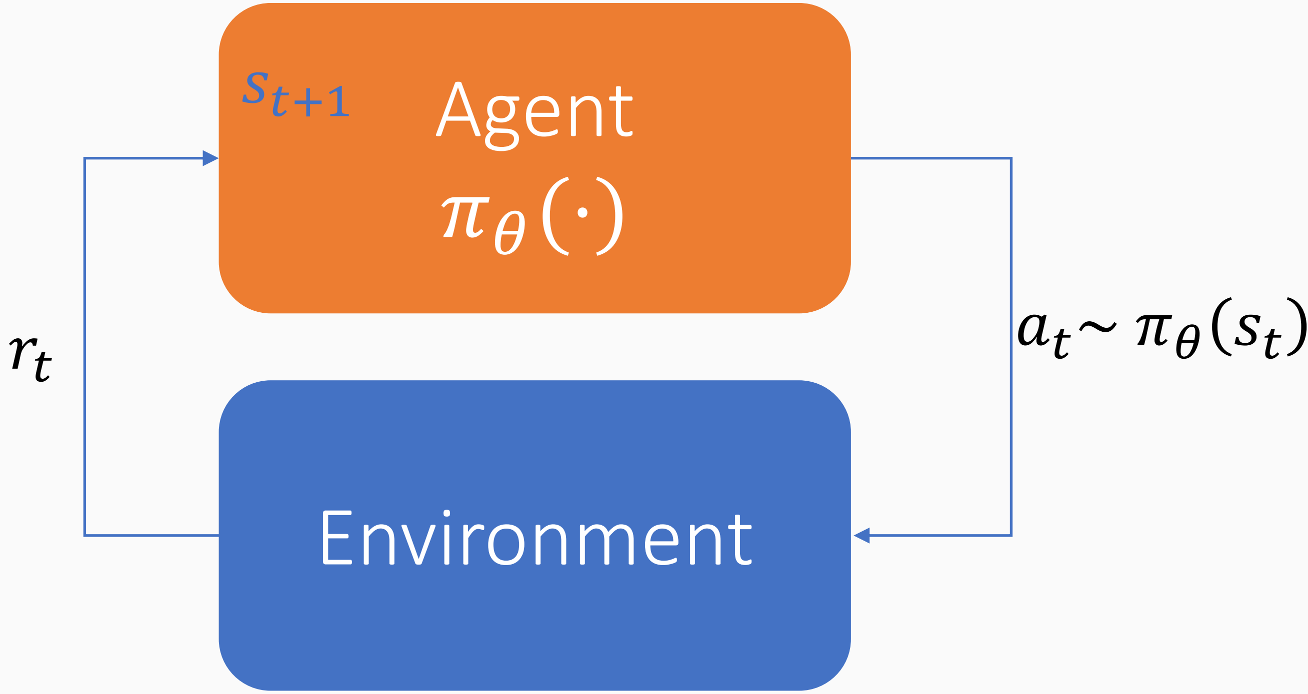


Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

Some examples of agents and environments

Agent: converts a natural language command into a program
Environment: executes the program and returns the output



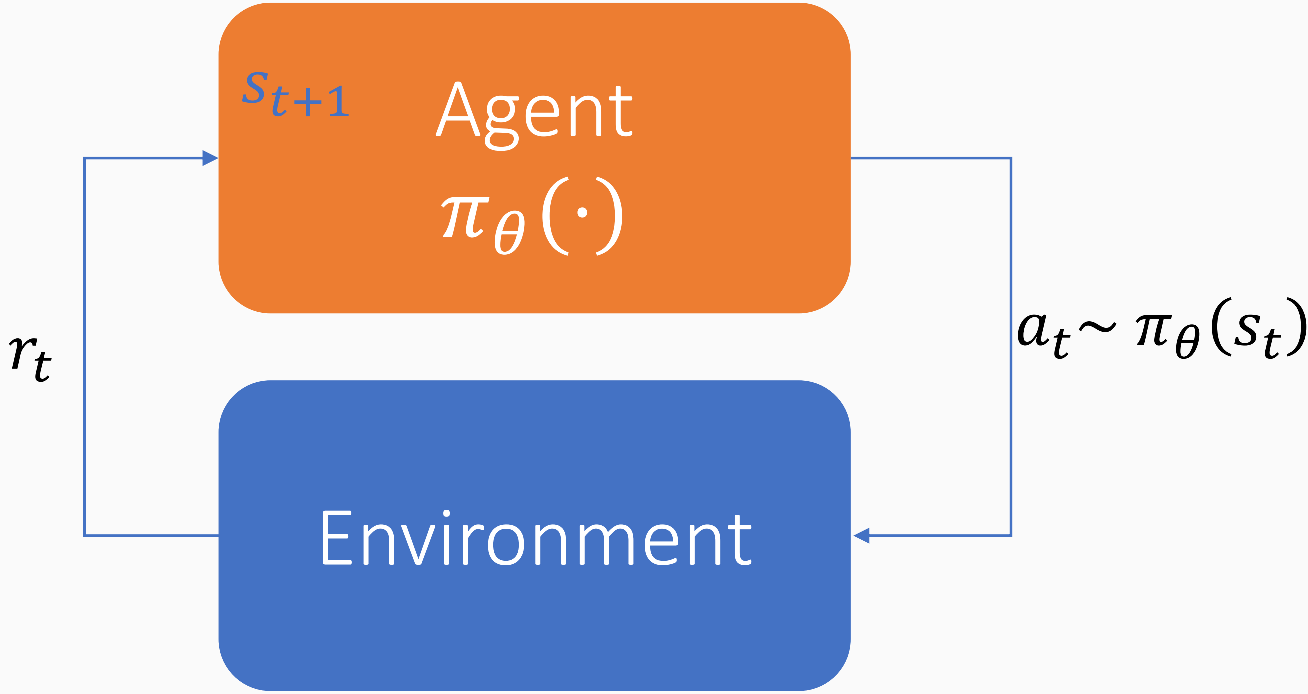
Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

Some examples of agents and environments

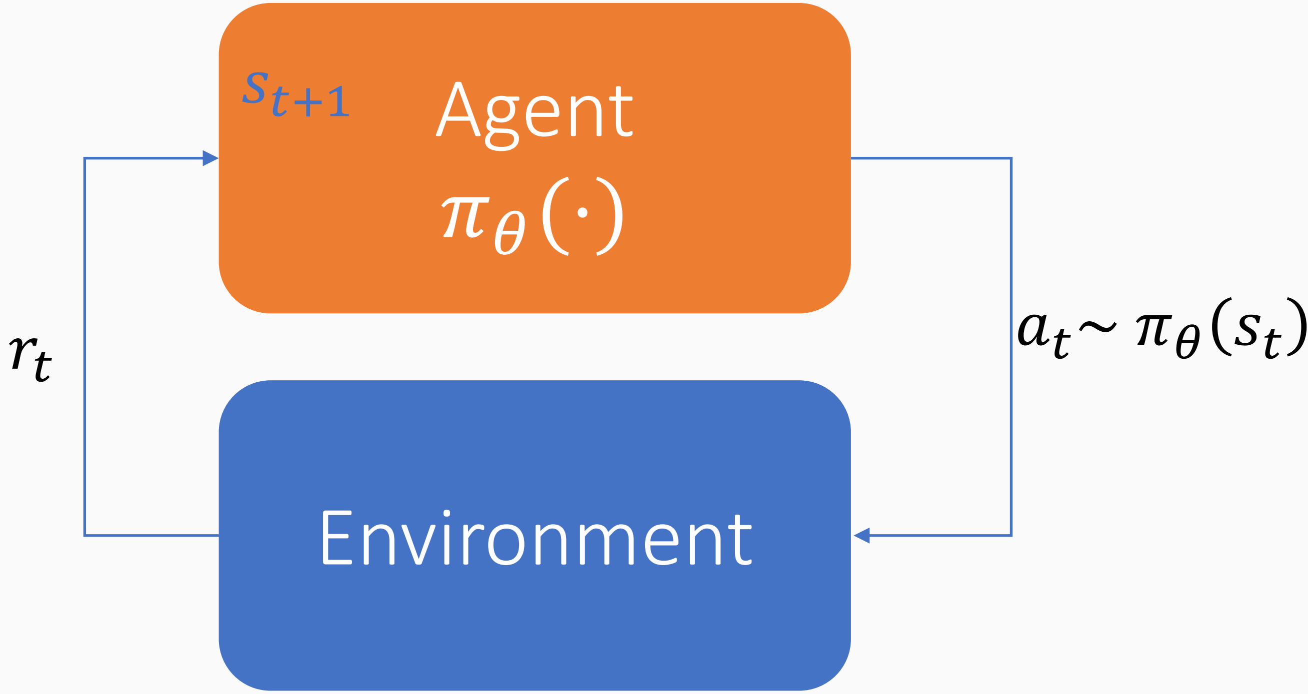
Agent: converts a natural language command into a program
Environment: executes the program and returns the output

Agent: controls a quadcopter
Environment: the physical environment where the quadcopter can navigate (or not)



Reinforcement learning: Some basics

An agent interacts with an environment by taking actions



Some examples of agents and environments

Agent: converts a natural language command into a program
Environment: executes the program and returns the output

Agent: controls a quadcopter
Environment: the physical environment where the quadcopter can navigate (or not)

Agent: Observes a board game and plays the next move
Environment: A game engine with more real or simulated players

Reinforcement learning: Some basics

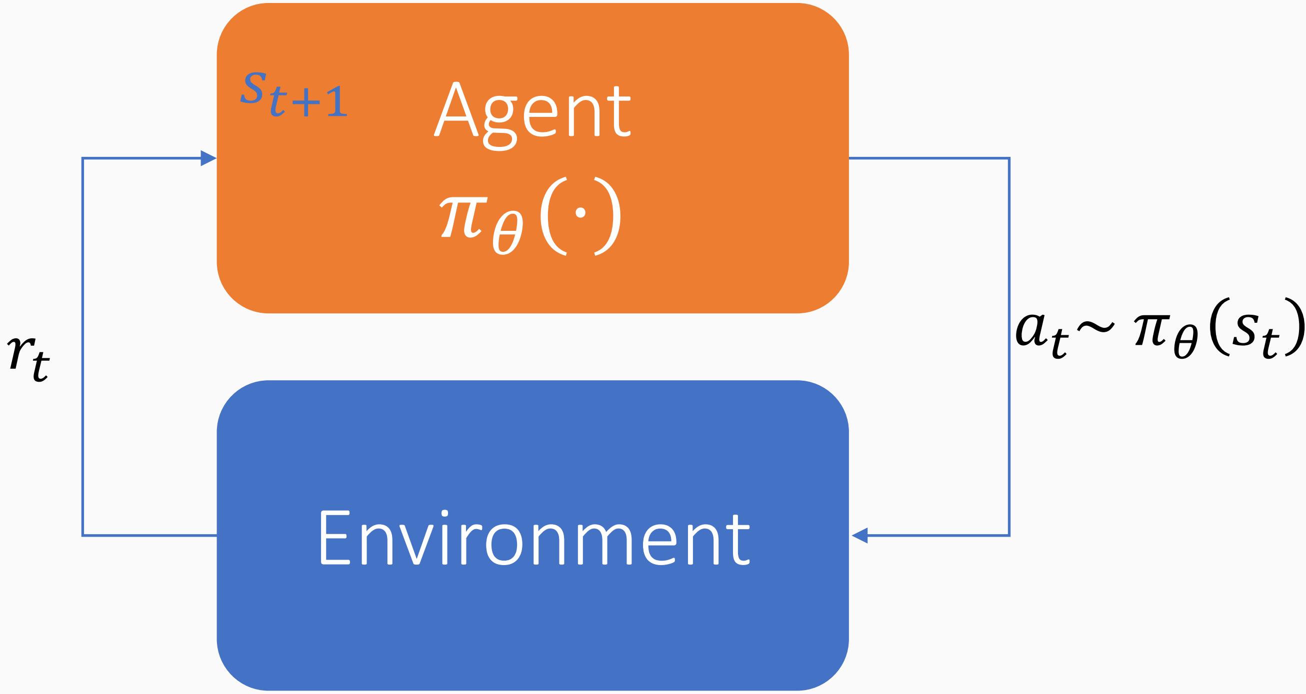
An agent interacts with an environment by taking actions

Some examples of agents and environments

Agent: converts a natural language command into a program
Environment: executes the program and returns the output

Agent: controls a quadcopter
Environment: the physical environment where the quadcopter can navigate (or not)

Agent: Observes a board game and plays the next move
Environment: A game engine with more real or simulated players



What are the states, actions and rewards in each case?

Reinforcement Learning

The field of reinforcement learning (RL) has studied the problem of learning by interacting with an environment for many years now [Williams, 1992; Sutton and Barto, 1998]

Circa 2013: resurgence of interest in RL applied to deep learning, game-playing [Mnih et al., 2013]

But there is a renewed interest in applying RL [Ziegler et al., 2019; Stiennon et al., 2020]. Why?

- RL w/ LMs has commonly been viewed as very hard to get right (still is!)
- RL algorithms that work for large neural models, including language models (e.g. PPO; [Schulman et al., 2017])

Caveats about this lecture

RL is a rich field with a diverse collection of algorithms and ideas

We will look at one popular approach: REINFORCE

The general setting allows for rewards being harvested over multiple steps corresponding to multiple actions. For this lecture, we will assume that there is only one action and one step that gets a reward

Learning to maximize reward

The goal of learning: Discover a policy that allows the agent to accure high rewards

Formally: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s, a)]$$

How do we solve this optimization problem?

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{a \sim p_{\theta}}[R(s,a)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \text{learning rate} \times \text{gradient of the objective}$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{a \sim p_{\theta}}[R(s,a)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{a \sim p_{\theta_t}}[R(s,a)]$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s,a)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \underbrace{\alpha}_{\text{learning rate}} \underbrace{\nabla_{\theta_t} \mathbb{E}_{a \sim p_{\theta_t}}[R(s,a)]}_{\text{gradient of the objective}}$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_\theta}[R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{a \sim p_{\theta_t}}[R(s, a)]$$

But how do we estimate this gradient?

(Why doesn't the usual approach for derivatives not work?)

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_\theta}[R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{a \sim p_{\theta_t}}[R(s, a)]$$

But how do we estimate this gradient?

Let us look at a simple version of policy gradients

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x \sim P}[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x\sim P}\left[f(x)\right] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

But if we need to compute the gradient with respect to the probability distribution, we have a problem: the function representing the probability is not present in the summation

$$\nabla_{\theta_t}\mathbb{E}_{a\sim p_{\theta_t}}[R(s;a)] \approx \nabla_{\theta_t}\frac{1}{n}\sum_{i=1}^{n} R(s;a)$$

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x\sim P}\left[f(x)\right] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

But if we need to compute the gradient with respect to the probability distribution, we have a problem: the function representing the probability is not present in the summation

$$\nabla_{\theta_t}\mathbb{E}_{a\sim p_{\theta_t}}[R(s;a)] \approx \nabla_{\theta_t}\boxed{\frac{1}{n}\sum_{i=1}^{n} R(s;a)}$$

No θ_t in this expression!

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x\sim P}\left[f(x)\right] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

2. The REINFORCE trick [Williams 1992]

$$\frac{\partial}{\partial x} f(x) = f(x)\frac{\partial}{\partial x}\log f(x)$$

Let us reformulate the gradient

$\nabla_{\theta} \mathbb{E}_{a \sim \mathrm{P}_{\theta}}[R(s, a)$

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

$$= \sum_a R(s,a) P_\theta(a \mid s) \nabla_\theta \log P_\theta(a \mid s)$$

The REINFORCE trick

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

$$= \sum_a R(s,a) P_\theta(a \mid s) \nabla_\theta \log P_\theta(a \mid s)$$

The REINFORCE trick

$$= \mathbb{E}_{a\sim P_\theta}[R(s,a) \nabla_\theta \log P_\theta(a \mid s)]$$

Rewrite as an expectation

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

$$= \sum_a R(s,a) P_\theta(a \mid s) \nabla_\theta \log P_\theta(a \mid s)$$

The REINFORCE trick

$$= \mathbb{E}_{a\sim P_\theta}[R(s,a) \nabla_\theta \log P_\theta(a \mid s)]$$

Rewrite as an expectation

$$\approx \frac{1}{n} \sum_{i=1}^{n} R(s,a_i) \nabla_\theta \log P_\theta(a_i \mid s)$$

Approximate with n samples

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a\sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

$$= \sum_a R(s,a) P_\theta(a \mid s) \nabla_\theta \log P_\theta(a \mid s)$$

The REINFORCE trick

$$= \mathbb{E}_{a\sim P_\theta}[R(s,a) \nabla_\theta \log P_\theta(a \mid s)]$$

Rewrite as an expectation

$$\approx \frac{1}{n} \sum_{i=1}^{n} R(s,a_i) \nabla_\theta \log P_\theta(a_i \mid s)$$

Approximate with n samples

How do we compute the derivative of the log probability?

Let us reformulate the gradient

$$\nabla_\theta \mathbb{E}_{a \sim \mathrm{P}_\theta}[R(s,a)] = \nabla_\theta \sum_a R(s,a) P_\theta(a \mid s)$$

Definition of expectation. Also works with integrals, but let's keep things simple

$$= \sum_a R(s,a) \nabla_\theta P_\theta(a \mid s)$$

R does not depend on θ

$$= \sum_a R(s,a) P_\theta(a \mid s) \nabla_\theta \log P_\theta(a \mid s)$$

The REINFORCE trick

$$= \mathbb{E}_{a \sim P_\theta}[R(s,a) \nabla_\theta \log P_\theta(a \mid s)]$$

Rewrite as an expectation

$$\approx \frac{1}{n} \sum_{i=1}^{n} R(s,a_i) \nabla_\theta \log P_\theta(a_i \mid s)$$

Approximate with n samples

How do we compute the derivative of the log probability? Autodiff

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s, a\ )]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{a \sim p_{\theta_t}}[R(s, a)]$$

But how do we estimate this gradient?

Answer: We use a neat trick to estimate the gradient of the expectation

Policy gradient

We want $\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s,a)]$

Gradient ascent with n samples from p_θ:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

Policy gradient

We want $\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s, a)]$

Gradient ascent with n samples from p_θ:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

This is a simplified version

There are many variants of this idea

Policy gradient

We want $\max_{\theta} \mathbb{E}_{a \sim p_\theta}[R(s,a)]$

Gradient ascent with n samples from p_θ:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

Note that we have no restriction on the reward $R(s,a)$. It could be non-differentiable, provided by the environment somehow, or provided by humans.

REINFORCE algorithm

Repeat:

1. Sample n actions $a_1, a_2, \cdots, a_n$ at state s
2. Compute all rewards $\mathrm{R}(a_i, s)$
3. Update parameters as:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

For our purposes, the states could represent examples in our data and actions could be predictions of the discrete part of our neuro-symbolic system

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a)\in D} \log p_{\theta}(a \mid s)$$

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a)\in D} \log p_{\theta}(a \mid s)$$

We could optimize this using stochastic gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \nabla_{\theta} \log p_{\theta}(a \mid s)$$

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a) \in D} \log p_{\theta}(a \mid s)$$

We could optimize this using stochastic gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \nabla_{\theta} \log p_{\theta}(a \mid s)$$

Compare to the update rule from policy gradient:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_{\theta} \log p_{\theta}(a_i \mid s)$$

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a)\in D} \log p_{\theta}(a \mid s)$$

We could optimize this using stochastic gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \nabla_{\theta} \log p_{\theta}(a \mid s)$$

Compare to the update rule from policy gradient:

Similar terms in both cases

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_{\theta} \log p_{\theta}(a_i \mid s)$$

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a)\in D} \log p_{\theta}(a \mid s)$$

We could optimize this using stochastic gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \nabla_{\theta} \log p_{\theta}(a \mid s)$$

For the supervised case, the action a is the “true” action and its probability will be made higher

Compare to the update rule from policy gradient:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_{\theta} \log p_{\theta}(a_i \mid s)$$

Compare to supervised maximum likelihood learning

Suppose we have a dataset of states paired with ground truth actions: $\mathrm{D} = \{(s, a)\}$ whose probability we wish to maximize:

$$\max_{\theta} \sum_{(s,a)\in D} \log p_{\theta}(a \mid s)$$

We could optimize this using stochastic gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \boxed{\nabla_{\theta} \log p_{\theta}(a \mid s)}$$

For the supervised case, the action a is the “true” action and its probability will be made higher

Compare to the update rule from policy gradient:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \boxed{\nabla_{\theta} \log p_{\theta}(a_i \mid s)}$$

We do not know which action is good. So we sample actions and weight the gradients associated with them by the reward

Problems with policy gradient

Does it always work?

- Vanilla policy gradient tends to have high variance in gradient estimates
  - There may be sudden jumps in performance, but training may be slow overall

- In practice: combine with variance reduction techniques
  - Several approaches exist in the literature
  - Let's look at one: *baselines*

Baselines

Our original update:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

Baselines

Our original update:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} R(s, a_i) \nabla_\theta \log p_\theta(a_i \mid s)$$

Modify to:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} [R(s, a_i) - b(s)] \nabla_\theta \log p_\theta(a_i \mid s)$$

Subtract a baseline from the reward

Baselines

Our original update:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n}\sum_{i=1}^{n} R(s, a_i)\nabla_\theta \log p_\theta(a_i \mid s)$$

Modify to:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n}\sum_{i=1}^{n} [R(s, a_i) - b(s)]\nabla_\theta \log p_\theta(a_i \mid s)$$

Subtract a baseline from the reward

- Doing so reduces variance
- But does not introduce any additional bias

In practice: a good choice of $b(s)$ is

$$b(s) = \frac{1}{n}\sum_{i=1}^{n} R(s, a_i)$$

Not theoretically the best, but often good

Introducing a baseline is unbiased

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]$$

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]$$

With a baseline, we have:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)]$$

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]$$

With a baseline, we have:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand it as

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)] - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]$$

With a baseline, we have:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand it as

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)] - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

This works by the linearity of expectations

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \boxed{\mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]}$$

With a baseline, we have:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand it as

$$\boxed{\mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]} - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

The first term is the same as the original expectation $\mathbb{E}_{a\sim p_\theta}[R(s,a)]$

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)] - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

Introducing a baseline is unbiased

Consider the expected reward:

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)]$$

With a baseline, we have:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand it as

$$\mathbb{E}_{a\sim p_\theta}[R(s,a)\nabla_\theta \log p_\theta(a \mid s)] - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

The first term is the same as the original expectation $\mathbb{E}_{a\sim p_\theta}[R(s,a)]$

Let us focus on the second term

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a \sim p_\theta}[b(s) \nabla_\theta \log p_\theta(a \mid s)]$$

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a \sim p_\theta}[b(s) \nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s) p_\theta(a \mid s) \nabla_\theta \log p_\theta(a \mid s)$$

We could have written this as an integral, but let's keep things simple

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s) p_\theta(a \mid s)\, \nabla_\theta \log p_\theta(a \mid s)$$

Apply the REINFORCE trick to simplify:

$$\sum_a b(s) \nabla_\theta p_\theta(a \mid s)$$

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s)p_\theta(a \mid s)\,\nabla_\theta \log p_\theta(a \mid s)$$

Apply the REINFORCE trick to simplify:

$$\sum_a b(s)\nabla_\theta p_\theta(a \mid s) = b(s)\nabla_\theta \sum_a p_\theta(a \mid s)$$

The baseline $b(s)$ doesn't depend on the action a, so we can pull it out of the summation

And the sum of gradients is the gradient of the sum

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a \sim p_\theta}[b(s) \nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s) p_\theta(a \mid s) \nabla_\theta \log p_\theta(a \mid s)$$

Apply the REINFORCE trick to simplify:

$$\sum_a b(s) \nabla_\theta p_\theta(a \mid s) = b(s) \nabla_\theta \boxed{\sum_a p_\theta(a \mid s)} = 0$$

This quantity is equal to 1 because it accumulates the entire support for the probability p_θ. Its derivative is zero

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s) p_\theta(a \mid s)\, \nabla_\theta \log p_\theta(a \mid s)$$

Apply the REINFORCE trick to simplify:

$$\sum_a b(s)\nabla_\theta p_\theta(a \mid s) = b(s)\nabla_\theta \sum_a p_\theta(a \mid s) = 0$$

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)] - \mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

The entire second term vanishes

Introducing a baseline is unbiased (2)

Let us look at the second term

$$\mathbb{E}_{a\sim p_\theta}[b(s)\nabla_\theta \log p_\theta(a \mid s)]$$

We can expand the expectation to write:

$$\sum_a b(s)p_\theta(a \mid s)\, \nabla_\theta \log p_\theta(a \mid s)$$

Apply the REINFORCE trick to simplify:

$$\sum_a b(s)\nabla_\theta p_\theta(a \mid s) = b(s)\nabla_\theta \sum_a p_\theta(a \mid s) = 0$$

What have we done:

$$\mathbb{E}_{a\sim p_\theta}[(R(s,a) - b(s))\nabla_\theta \log p_\theta(a \mid s)] = \mathbb{E}_{a\sim p_\theta}[R(s,a)]$$

REINFORCE algorithm with baselines

Repeat:

1. Sample n actions $a_1, a_2, \cdots, a_n$ at state s
2. Compute all rewards $\mathrm{R}(a_i, s)$
3. Compute baseline $b(s) = \frac{1}{n}\sum_{i=1}^{n} R(s, a_i)$
4. Update parameters as:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n}\sum_{i=1}^{n}[R(s, a_i) - b(s)]\nabla_\theta \log p_\theta(a_i \mid s)$$

For our purposes, the states could represent examples in our data and actions could be predictions of the discrete part of our neuro-symbolic system

REINFORCE: Summary

A useful tool when we have a black box system within a neural network

Caveats:

- Gradients will have high variance
- Variance control methods could help a little, but the gradients are still going to be noisy

In practice: this means that experiments will be tricky

- Use much larger batches than you may be used to
- Learning rates will matter, ADAM may be good
  - There are learning rate adjustment strategies designed for policy gradient too