535514: Reinforcement Learning

Lecture 6 — Policy Gradient

Ping-Chun Hsieh

March 11, 2024

Some Historical Accounts on Policy Gradient

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

RONALD J. WILLIAMS rjw@corwin.ccs.northeastern.edu
College of Computer Science, 161 CN, Northeastern University, 360 Huntington Ave., Boston, MA 02115

Abstract. This article presents a general class of associative reinforcement learning algorithms for connectionist networks containing stochastic units. These algorithms, called REINFORCE algorithms, are shown to make weight adjustments in a direction that lies along the gradient of expected reinforcement in both immediate-reinforcement tasks and certain limited forms of delayed-reinforcement tasks, and they do this without explicitly computing gradient estimates or even storing information from which such estimates could be computed. Specific examples of such algorithms are presented, some of which bear a close relationship to certain existing algorithms while others are novel but potentially interesting in their own right. Also given are results that show how such algorithms can be naturally integrated with backpropagation. We close with a brief discussion of a number of additional issues surrounding the use of such algorithms, including what is known about their limiting behaviors as well as further considerations that might be used to help develop similar but potentially more powerful reinforcement learning algorithms.

A seminal paper on PG published in *Machine Learning* journal in 1992 (citation > 10000)

Ronald Williams (Northeastern University)

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \quad (2)$$

† To whom correspondence should be addressed.

Another well-known paper by him and Jeff Hinton in 1986 (citation > 36000)

Jeff Hinton

Historical Accounts on PG

"Policy Gradient" Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Abstract

Function approximation is essential to reinforcement learning, but the standard approach of approximating a value function and determining a policy from it has so far proven theoretically intractable. In this paper we explore an alternative approach in which the policy is explicitly represented by its own function approximator, independent of the value function, and is updated according to the gradient of expected reward with respect to the policy parameters. Williams's REINFORCE method and actor–critic methods are examples of this approach. Our main new result is to show that the gradient can be written in a form suitable for estimation from experience aided by an approximate action-value or advantage function. Using this result, we prove for the first time that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy.

Richard Sutton
(U of Alberta & DeepMind)

Satinder Singh
(UMich & DeepMind)

A seminal paper on PG in NIPS 1999

Policy Optimization Framework:

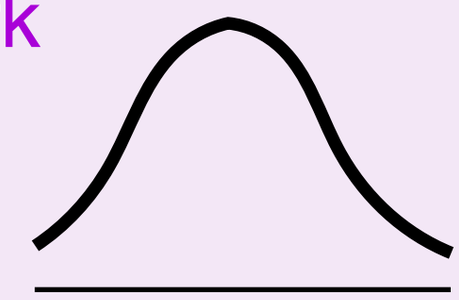
$E[G_t | S_t = S_0]$

- Consider a parametric policy class: $\{\pi_\theta | \theta \in \Theta \subset \mathbb{R}^d\}$
- View RL as an optimization problem: $\max_\theta V^{\pi_\theta}(\mu) := \mathbb{E}_{s_0 \sim \mu}[V^{\pi_\theta}(s_0)]$

Example:

1. Parametrize the policy by a neural network

s → $\pi_\theta(\cdot)$ →



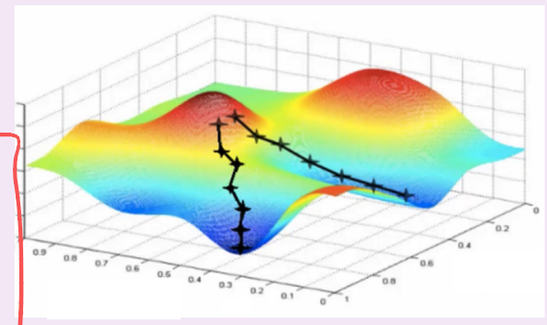
Neural network

A deterministic action or an action distribution

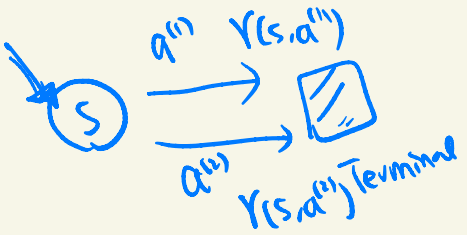
2. Update π_θ iteratively, e.g., by gradient ascent (GA)

$$\theta \leftarrow \theta + \alpha \nabla_\theta V^{\pi_\theta}(\mu)$$

$\frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta_{s,a}}$



- Question: Why using $V^\pi(\mu)$ as the objective (instead of each individual $V^\pi(s)$)?



$$\theta = [\theta_{s,a^{(1)}}, \theta_{s,a^{(2)}}]$$

$$V^{\pi_\theta}(\mu) = \pi_\theta(a^{(1)}|s) \cdot r(s,a^{(1)}) + \pi_\theta(a^{(2)}|s) \cdot r(s,a^{(2)})$$

$$\nabla_\theta V^{\pi_\theta}(\mu) = \begin{bmatrix} \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta_{s,a^{(1)}}} \\ \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta_{s,a^{(2)}}} \end{bmatrix}$$

3 Challenges in Policy Optimization

(C1) How to parametrize the policy?

(C2) How to solve the optimization problem in a model-free manner?

1. Policy gradient (PG)
2. Model-free prediction (aka model-free policy evaluation)

(C3) Any theoretical guarantee of PG?

Typical Examples of Parametric Policies

Discrete action space

- Softmax policies:

$$\pi_\theta(a \mid s) = \frac{\exp(\theta_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}$$

- Linear softmax policies:

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^T \phi_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta^T \phi_{s,a'})}$$

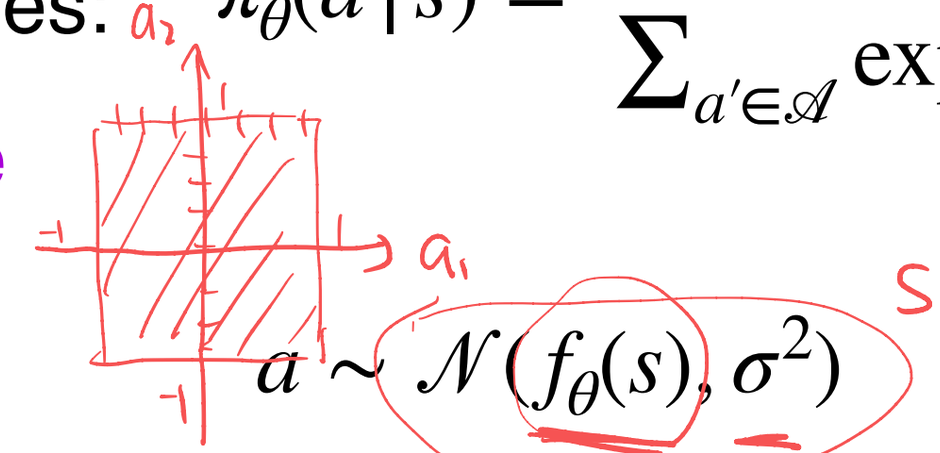
- Neural softmax policies:

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a' \in \mathcal{A}} \exp(f_\theta(s,a'))}$$

Continuous action space

- Gaussian policies:

$$a \sim \mathcal{N}(f_\theta(s), \sigma^2)$$

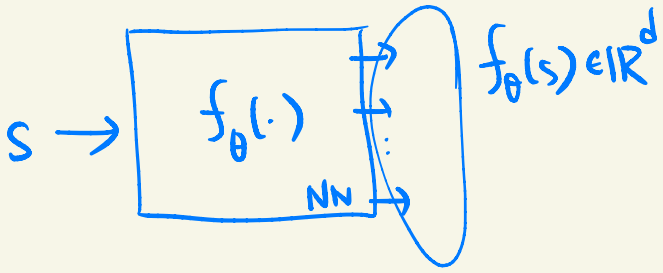


Questions: Are these parametric policies general enough?

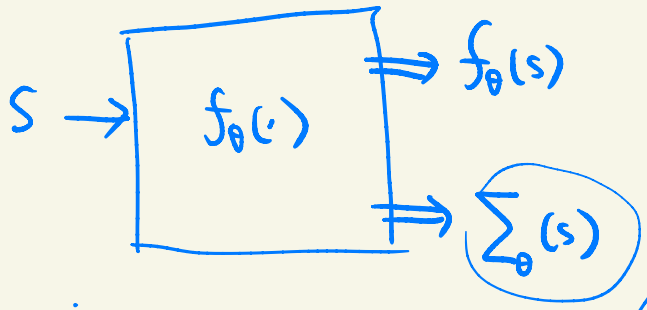
(We will come back to this momentarily)

- Gaussian policies ($a \in \mathbb{R}^d$)

$$a \sim \mathcal{N}(f_\theta(s), \Sigma)$$

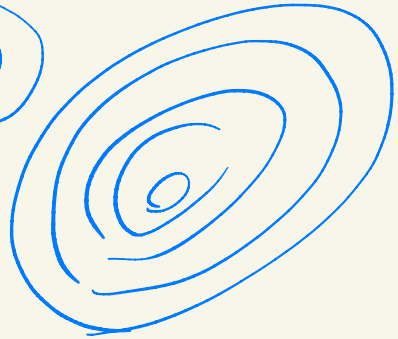


e.g. $\Sigma = \sigma^2 I_d = \begin{bmatrix} \sigma^2 & & 0 \\ & \ddots & \\ 0 & & \sigma^2 \end{bmatrix}$ (d)



$$\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ & & \ddots \\ 0 & & \sigma_d^2 \end{bmatrix}$$

Normalizing Flows



How About Direct Tabular Parametrization?

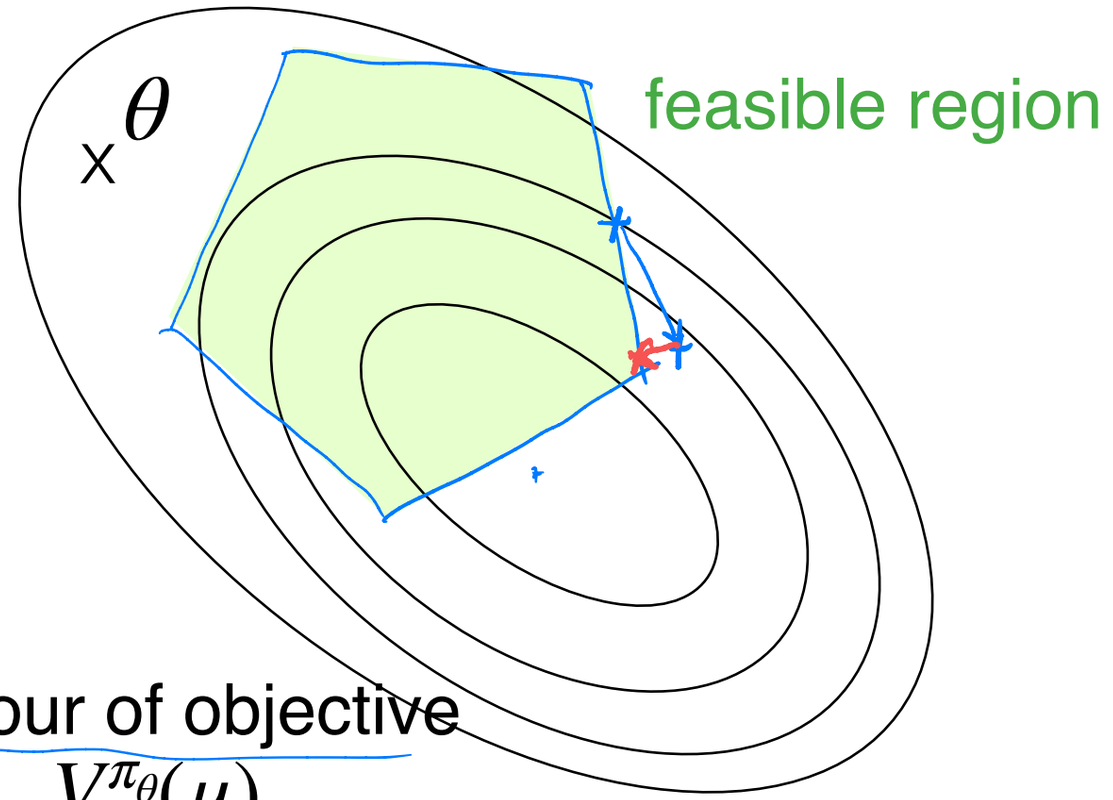
- Question: Can we do "direct parametrization", i.e., $\pi_\theta(a \mid s) = \theta_{s,a}$?

$\sum_{a \in A} \theta_{s,a} = 1$, $\theta_{s,a} \geq 0$ for all s, a

- Possible, but now we face constrained policy optimization

$$\max_{\theta \in \Theta} V^{\pi_\theta}(\mu)$$

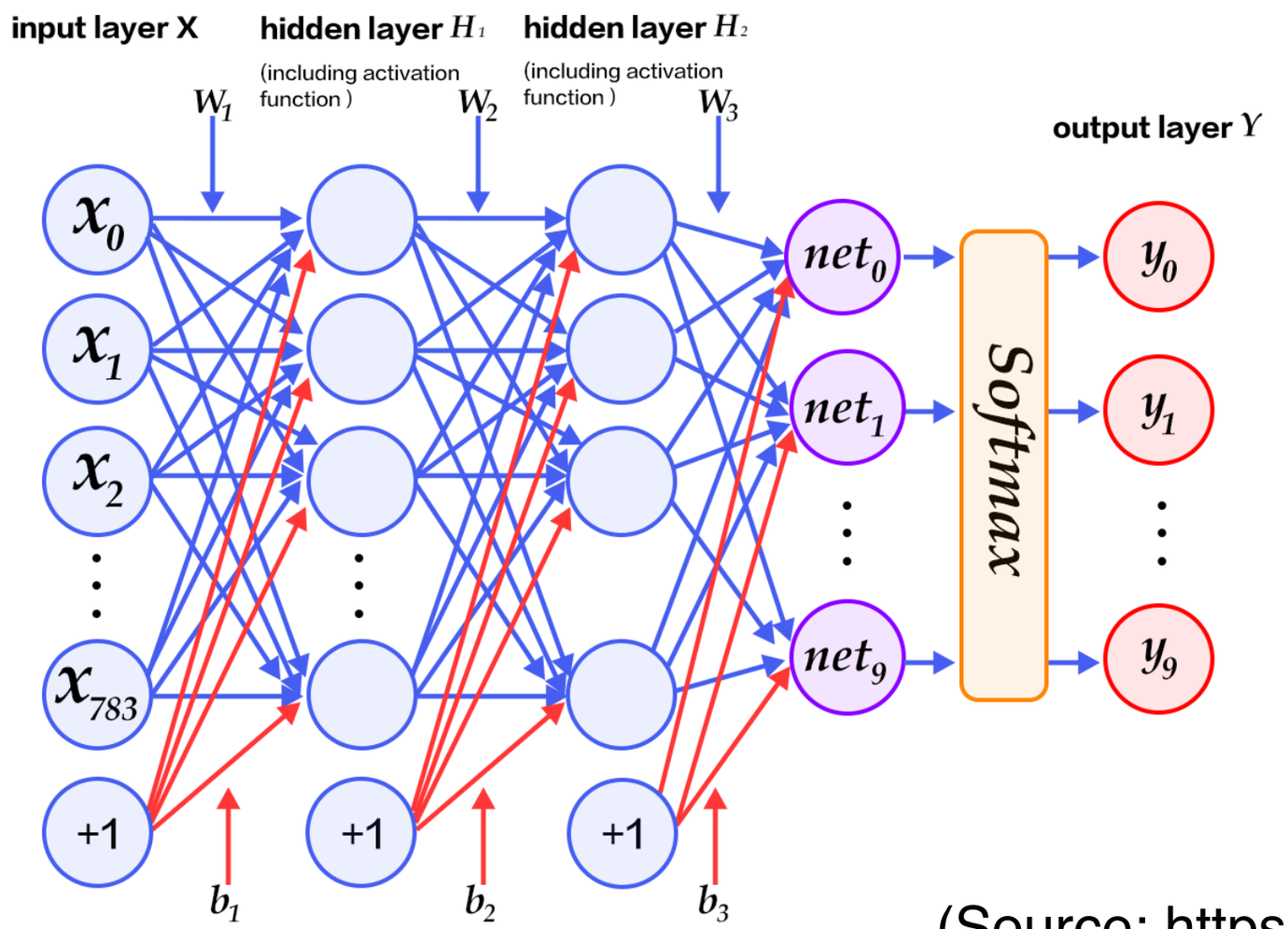
subject to $\sum_a \theta_{s,a} = 1, \forall s$



- In this case, *projection* is needed to ensure a valid probability distribution (often computationally heavy)

Neural Softmax Policies

- Neural softmax policies: $$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a' \in \mathcal{A}} \exp(f_\theta(s,a'))}$$
  - θ denotes all the weights and biases of a neural net



(Source: https://files.ifi.uzh.ch/ddis/)

Policy Gradient (PG)

Review: Some Useful Notations

- Question: To apply GA, what do we need?
- Policy gradient! $\nabla_\theta V^{\pi_\theta}(\mu)$ (μ: distribution of initial state)

Some useful notations

- 1. Sample return $G(\tau)$ along a trajectory $\tau = (s_0, a_0, r_1, \cdots)$

$$G(\tau) := \sum_{t=0}^{\infty} \gamma^t \cdot r_{t+1}(\tau)$$

- 2. Value function written in $G(\tau)$

$$V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}[G(\tau)] = \sum_\tau G(\tau) P_\mu^{\pi_\theta}(\tau)$$

Sources of randomness: ① State transition ② Reward ③ Policies

Review: Some Useful Notations (Cont.)

- 3. Discounted state visitation distribution (aka "occupancy measure")

$$d_{s_0}^{\pi}(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid s_0, \pi)$$

$$d_{\mu}^{\pi}(s) := \mathbb{E}_{s_0\sim\mu}\left[d_{s_0}^{\pi}(s)\right]$$

$$\sum_{s\in S} d_{s_0}^{\pi}(s) = (1-\gamma)\sum_{s\in S}\left(\sum_{t=0}^{\infty}\gamma^t P(s_t=s\mid s_0,\pi)\right)$$

$$= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \sum_{s\in S} P(s_t=s\mid s_0,\pi)$$

$$\sum_{s\in S} d_{s_0}^{\pi}(s) = 1$$

- Question: What's the purpose of $(1-\gamma)$ in the above?

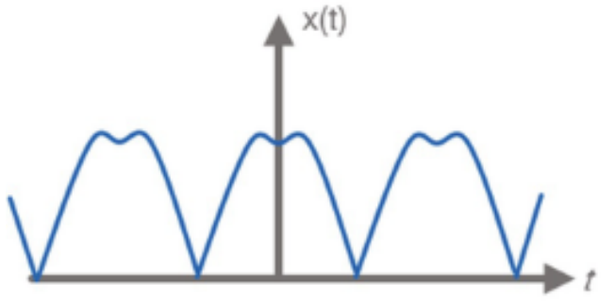
"Normalization factor"

$$= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t = 1 \qquad \frac{1}{1-\gamma}$$

State Visitation Distribution: An Analogy

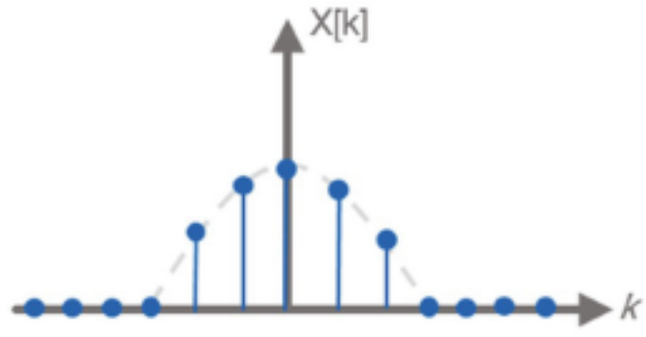
Fourier Series

Time domain



≡

Frequency domain



RL Objective Function

Time domain

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t \,\middle|\, s_0 \sim \mu; \pi_\theta\right] = V^{\pi_\theta}(\mu)$$

=

Distribution domain

$$\mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi(\cdot|s)}\left[R(s,a)\right]$$

(We will use this form several times in the sequel)

The Policy Gradient: Inherent Difficulty

- Issue: Computing $\nabla_\theta V^{\pi_\theta}(\mu)$ might seem highly non-trivial…
- Recall: $V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}[G(\tau)] = \sum_\tau G(\tau) P_\mu^{\pi_\theta}(\tau)$

- $V^{\pi_\theta}(\mu)$ is affected by θ in two ways:

1. θ determines $\pi_\theta(\cdot \mid s)$
2. The actions taken by π_θ affect the <u>state visitation</u> accordingly

Expressions of The Policy Gradient

- **Expressions of Policy Gradient (aka Policy Gradient Theorem)**:

(P1) Total reward: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$

(P2) REINFORCE: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$

(P3) Q-value and discounted state visitation:

$$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}\left[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a \mid s)\right]$$

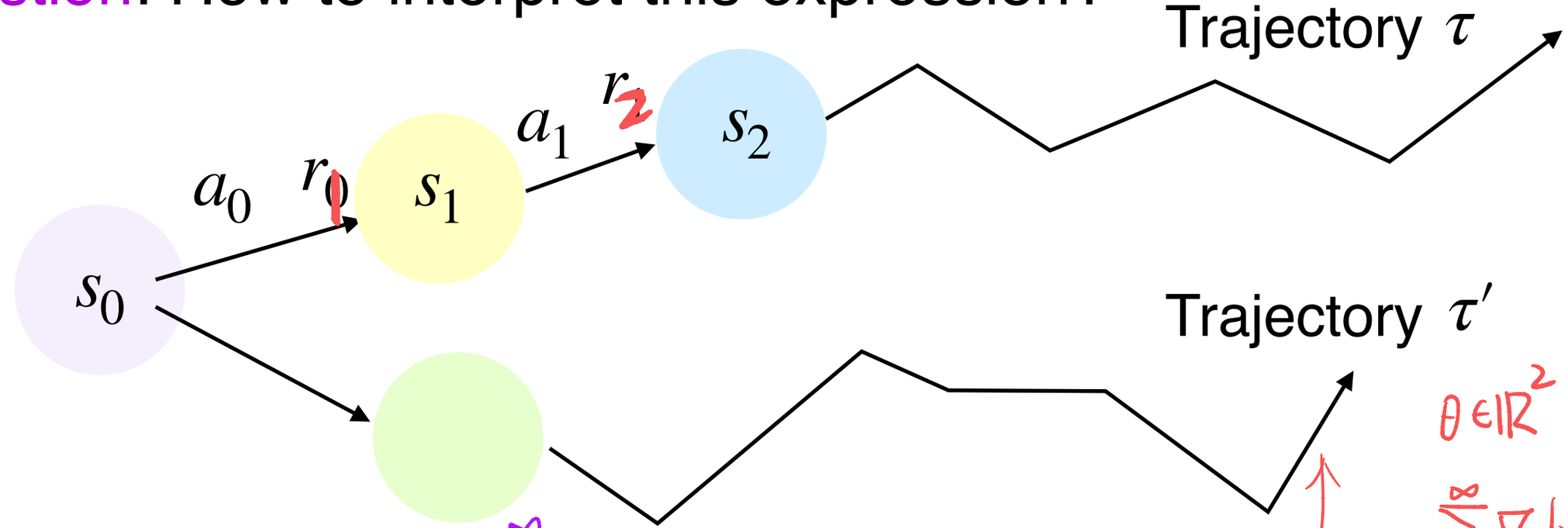
- Question: Why do we care about various expressions?
- Each expression gives us an RL algorithm!

[AJK, Chapter 11]

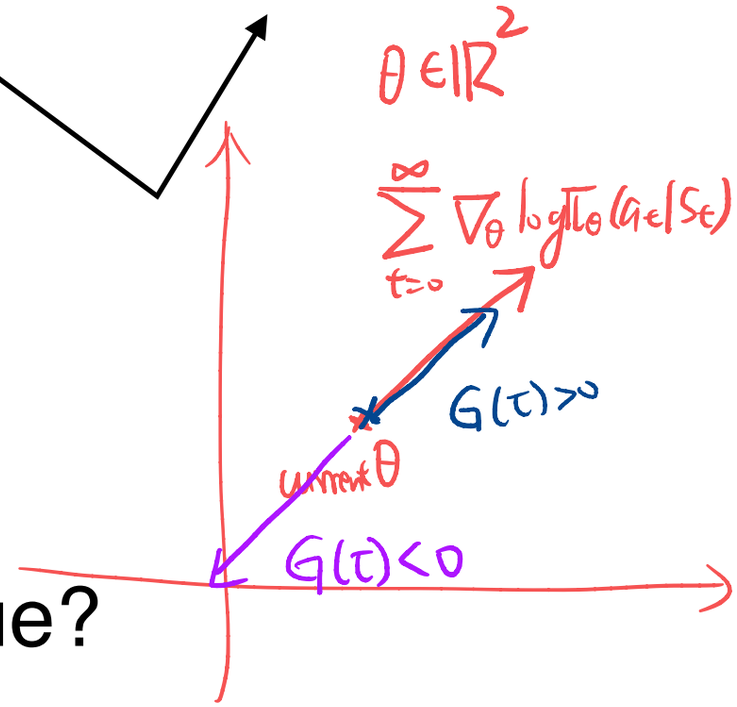
(P1) Total Reward Policy Gradient

- Want to show: $$\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

- Question: How to interpret this expression?



1. $G(\tau) > 0$: Follow $\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$
2. $G(\tau) < 0$: Follow $-\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$



- Question: Can you find out any potential issue?

(P1) Total Reward Policy Gradient (Cont.)

- Want to show: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$

- Recall: $V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}[G(\tau)] = \sum_\tau G(\tau) P_\mu^{\pi_\theta}(\tau)$

$$\nabla_\theta V^{\pi_\theta}(\mu) = \sum_\tau G(\tau) \nabla_\theta P_\mu^{\pi_\theta}(\tau)$$

chain rule

$$= \sum_\tau G(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \nabla_\theta \log P_\mu^{\pi_\theta}(\tau) \right)$$

the probability of observing trajectory τ

$$= \sum_\tau G(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \nabla_\theta \log \left(\mu(s_0) \pi_\theta(a_0 \mid s_0) P(s_1 \mid s_0, a_0) \pi_\theta(a_1 \mid s_1) \cdots \right) \right)$$

$$= \sum_\tau G(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right)$$

$$= \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

$G(\tau)$ does not depend on θ

$\nabla_\theta \log P_\mu^{\pi_\theta}(\tau) = \frac{1}{P_\mu^{\pi_\theta}(\tau)} \cdot \nabla_\theta P_\mu^{\pi_\theta}(\tau)$

$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3 \ldots)$

(Not a rigorous proof)

Technical Subtlety in the Proof of (P1):

- For τ of infinite length, $\nabla_\theta P_\mu^{\pi_\theta}(\tau)$ needs to be handled more carefully
- Idea: Consider "truncated trajectories" and take the limit

1. Without loss of generality, assume $R(s,a) \in [0,1]$

Define $\tau^{(k)}$ to be the truncated version of τ up to step k

Define $V_k^{\pi_\theta}(\mu) = \sum_{\tau^{(k)}} G(\tau^{(k)}) P_\mu^{\pi_\theta}(\tau^{(k)})$

2. Then, $\lim_{k\to\infty} V_k^{\pi_\theta}(\mu) = V^{\pi_\theta}(\mu)$ (by Monotone convergence theorem)

3. Next, we need to verify that $\lim_{k\to\infty} \nabla_\theta V_k^{\pi_\theta}(\mu) = \nabla_\theta V^{\pi_\theta}(\mu)$ by showing "uniform convergence" of $\nabla_\theta V_k^{\pi_\theta}(\mu)$ [Rudin 1976, Thm 7.17]

(Note that it is not always valid to interchange ∇ and $\lim$)

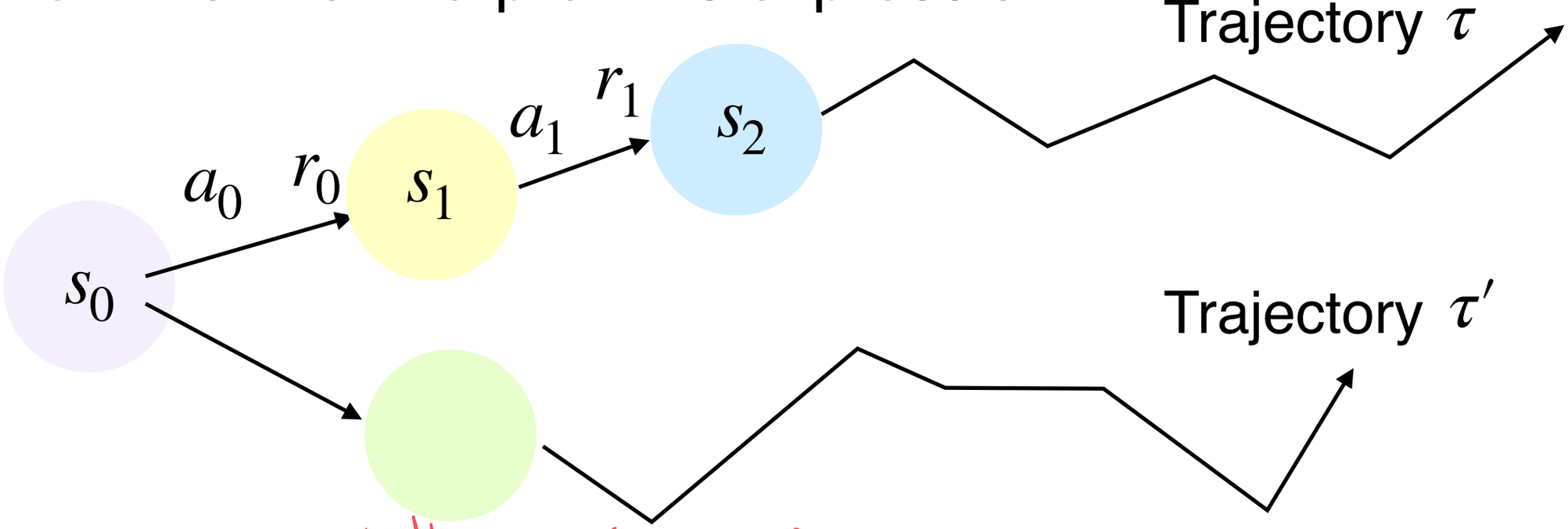
(P2) REINFORCE Policy Gradient

$E[G_t | s_t, a_t, \pi_\theta]$

- Want to show: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)\right]$

  follow/ not follow update direction

- Question: How to interpret this expression?



1. $Q(s, a) > 0$: Follow $\nabla_\theta \log \pi_\theta(a_t | s_t)$
2. $Q(s, a) < 0$: Follow $-\nabla_\theta \log \pi_\theta(a_t | s_t)$

"Relative importance matters"

- Question: Can you find out any potential issue?

(P2) REINFORCE Policy Gradient

- Want to show: $$\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$
- Idea: prove by induction
- Recall: $$V^{\pi_\theta}(s_0) = \sum_{a_0 \in \mathcal{A}} \pi_\theta(a_0 \mid s_0) Q^{\pi_\theta}(s_0, a_0)$$

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_{a_0} \nabla_\theta \left(\pi_\theta(a_0 \mid s_0) Q^{\pi_\theta}(s_0, a_0)\right)$$

$$= \underbrace{\sum_{a_0} \nabla_\theta \left(\pi_\theta(a_0 \mid s_0)\right) Q^{\pi_\theta}(s_0, a_0)}_{(1)} + \underbrace{\sum_{a_0} \pi_\theta(a_0 \mid s_0) \nabla \left(Q^{\pi_\theta}(s_0, a_0)\right)}_{(2)}$$

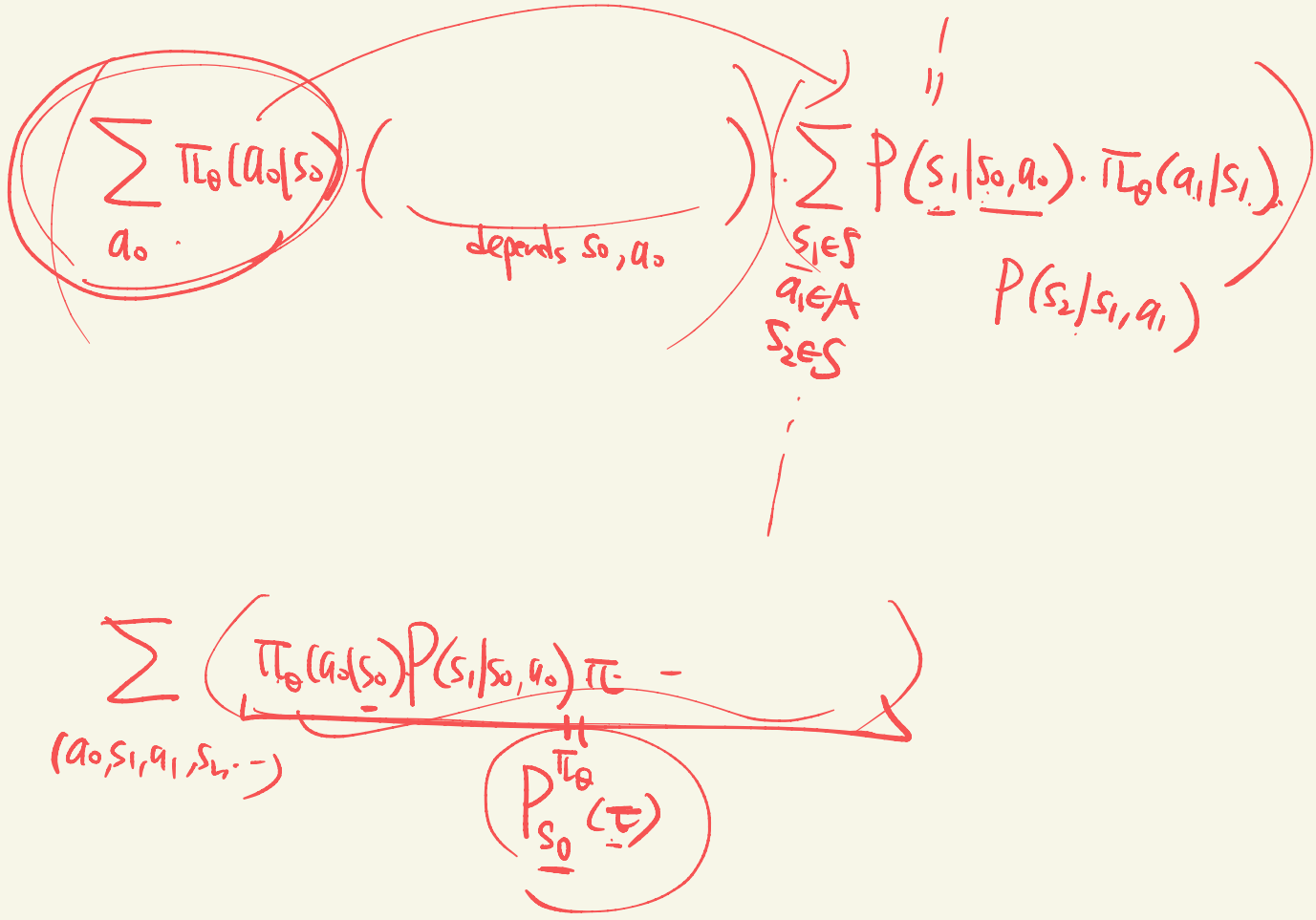
chain rule

$$(1) = \sum_{a_0} \pi_\theta(a_0 \mid s_0) \cdot \nabla_\theta \left(\log \pi_\theta(a_0 \mid s_0)\right) \cdot Q^{\pi_\theta}(s_0, a_0)$$

$$= \mathbb{E}_{\tau \sim P_{s_0}^{\pi_\theta}}\left[\nabla_\theta \left(\log \pi_\theta(a_0 \mid s_0)\right) \cdot Q^{\pi_\theta}(s_0, a_0)\right]$$

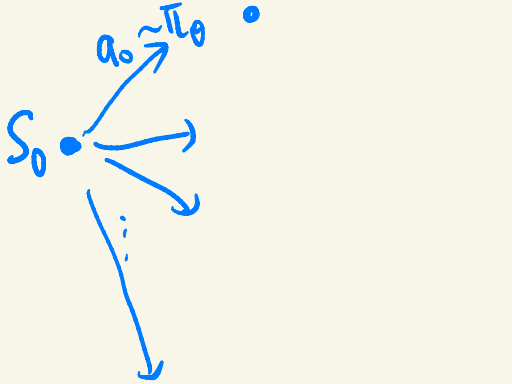
$$\nabla_\theta \log \pi_\theta(a_0 \mid s_0) = \frac{1}{\pi_\theta(a_0 \mid s_0)} \nabla_\theta \pi_\theta(a_0 \mid s_0)$$

$$(2) = \sum_{\tau : (s_0, a_0) \in \tau} P_\mu^{\pi_\theta}(\tau)$$


$\sum_{a_0} \pi_\theta(a_0|s_0)$
depends s_0, a_0
$\sum_{\substack{s_1 \in S \\ a_1 \in A \\ s_2 \in S}} P(s_1|s_0,a_0) \cdot \pi_\theta(a_1|s_1)$
$P(s_2|s_1,a_1)$
$\sum_{(a_0,s_1,a_1,s_2,\dots)} \pi_\theta(a_0|s_0) P(s_1|s_0,a_0) \pi \cdots$
$P_{s_0}^{\pi_\theta}(\tau)$

$$E_{\tau \sim p^{\pi_\theta}_{s_0}}[\ \ldots\ ]$$

trajectory that starts from s_0 π_θ



(P2) REINFORCE Policy Gradient (Cont.)

$$
\begin{aligned}
(2) &= \sum_{a_0} \pi_\theta(a_0 \mid s_0) \nabla \left(Q^{\pi_\theta}(s_0, a_0) \right) \\
&= \sum_{a_0} \pi_\theta(a_0 \mid s_0) \nabla_\theta \left(R(s_0, a_0) + \gamma \sum_{s_1} P(s_1 \mid s_0, a_0) \cdot V^{\pi_\theta}(s_1) \right) \\
&= \sum_{a_0} \pi_\theta(a_0 \mid s_0) \left(\gamma \sum_{s_1} P(s_1 \mid s_0, a_0) \cdot \nabla_\theta V^{\pi_\theta}(s_1) \right) \\
&= \mathbb{E}_{\tau \sim P^{\pi_\theta}_{s_0}} \left[\gamma \nabla_\theta V^{\pi_\theta}(s_1) \right]
\end{aligned}
$$

Bellman equation

By combining (1) and (2):

$$
\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\tau \sim P^{\pi_\theta}_{s_0}} \left[\nabla_\theta \left(\log \pi_\theta(a_0 \mid s_0) \right) \cdot Q^{\pi_\theta}(s_0, a_0) \right] + \mathbb{E}_{\tau \sim P^{\pi_\theta}_{s_0}} \left[\gamma \nabla_\theta V^{\pi_\theta}(s_1) \right]
$$

(1) (2)

This is a recursion!

(P2) REINFORCE Policy Gradient (Cont.)

Given the recursion:

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\tau \sim P_{s_0}^{\pi_\theta}}\Big[\nabla_\theta\big(\log \pi_\theta(a_0 \,|\, s_0)\big) \cdot Q^{\pi_\theta}(s_0, a_0)\Big] + \mathbb{E}_{\tau \sim P_{s_0}^{\pi_\theta}}\Big[\gamma \nabla_\theta V^{\pi_\theta}(s_1)\Big]$$

- We can verify that:

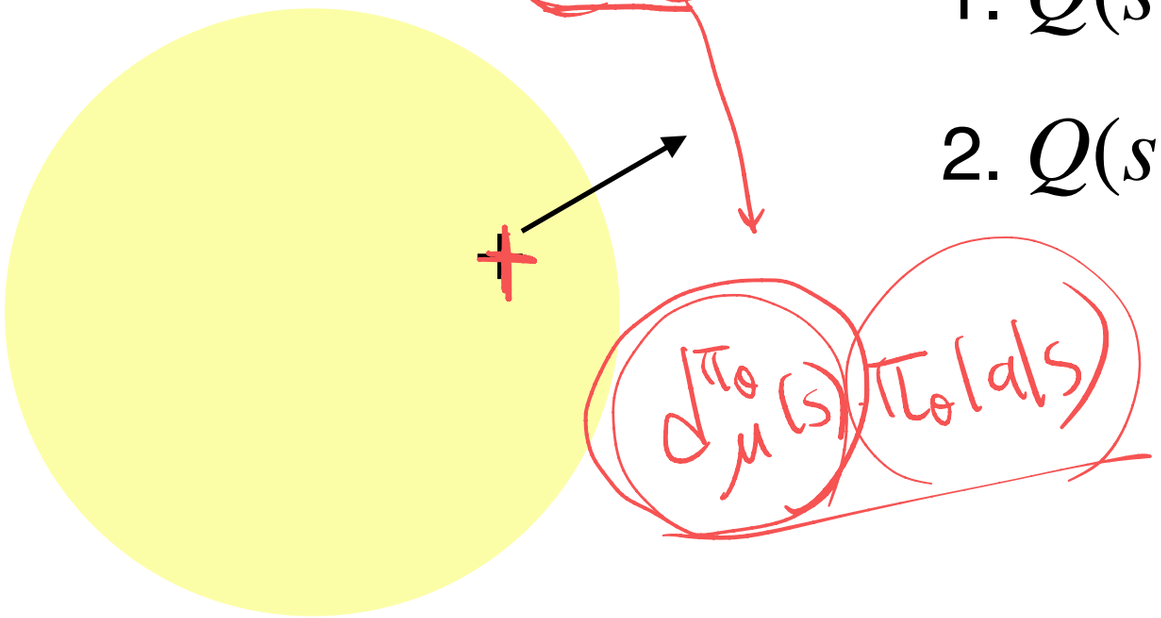
$$\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_{\mu}^{\pi_\theta}}\Big[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t)\, \nabla_\theta \log \pi_\theta(a_t \,|\, s_t)\Big]$$

(P3) Q-Value and Discounted State Visitation

- Want: $\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s\sim d_\mu^{\pi_\theta}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \left[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a \mid s) \right]$

- Question: How to interpret this expression?

Sample space of (s,a)



1. $Q(s,a) > 0$:

2. $Q(s,a) < 0$:

- Question: Why is this expression useful?

Sampling (s,a) is easier than sampling trajectories.

(P3) Q-Value and Discounted State Visitation (Cont.)

- Want: $\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a\,|\,s)\right]$

- Proof idea: Use the following lemma and (P2)

- **Lemma (From Trajectories to Visitation)**:

  For any function $f(s,a) : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$, we have

$$\mathbb{E}_{\tau\sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty}\gamma^t f(s_t,a_t)\right] = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta},a\sim\pi_\theta(\cdot|s)}\left[f(s,a)\right]$$

- Proof: Expand RHS and recollect terms to get LHS (HW1)

Computing $\nabla_\theta \log \pi_\theta(a \mid s)$: Linear Softmax Policy

- (P1)-(P3) all involve $\nabla_\theta \log \pi_\theta(a \mid s)$
- $\nabla_\theta \log \pi_\theta(a \mid s)$ is often called the score function
- Example: Linear softmax policy
  - Feature vector for each state-action pair $\phi(s, a)$
  - Probability of action is proportional to exponentiated weight

$$\pi_\theta(a \mid s) = \frac{e^{\phi(s,a)^T \theta}}{\sum_{a' \in \mathcal{A}} e^{\phi(s,a')^T \theta}}$$

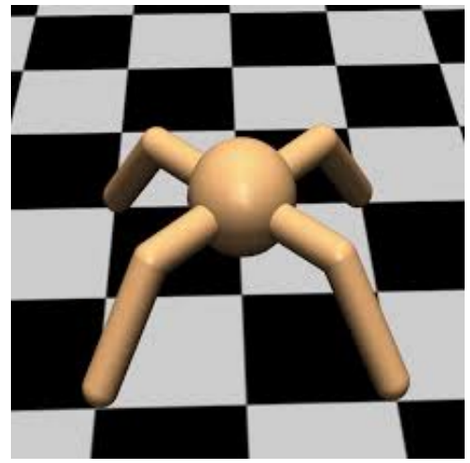
  - The score function is

$$\nabla_\theta \log \pi_\theta(a \mid s) = \phi(s, a) - \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}[\phi(s, a)]$$

Computing $\nabla_\theta \log \pi_\theta(a \mid s)$: Gaussian Policy

- Example: Gaussian policies for continuous actions
  - $a \sim \mathcal{N}(\mu_\theta(s), \sigma^2)$
  - Mean is a linear combination of state features $\mu_\theta(s) = \phi(s)^T \theta$
  - Variance may be fixed σ^2 (or can also be parametrized)
  - The score function is

$$\nabla_\theta \log \pi_\theta(a \mid s) = \frac{(a - \mu_\theta(s))\phi(s)}{\sigma^2}$$



- Question: How about an NN-based Gaussian policy?

(P1) Total reward: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\Big[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \,|\, s_t)\Big]$

(P2) REINFORCE: $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\Big[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \,|\, s_t)\Big]$

(P3) Q-value and discounted state visitation:

$$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)}\Big[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a \,|\, s)\Big]$$

Next Question: Is it easy to compute PG?

(P1)-(P3) all involve a highly complex expectation!

(This issue is quite common in many ML problems!)

Issues With (Exact) PG

- (Exact) PG Update (Use (P1) as an Example)

$$\theta_{k+1} = \theta_k + \eta_k \cdot \nabla_\theta V^{\pi_\theta}(\mu)|_{\theta=\theta_k}$$

$$= \theta_k + \eta_k \cdot \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

- Question: Any issues?

1. Distribution / statistics of τ is unknown (i.e., model-free setting)

2. Expectation usually involves multi-dimensional integral, which is computationally expensive

“Stochastic” Policy Gradient

- Idea: Use sampling to estimate expectation

$$\theta_{k+1} = \theta_k + \eta_k \cdot \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \qquad \text{(Exact PG)}$$

$$\theta_{k+1} = \theta_k + \eta_k \cdot \underbrace{G(\tau') \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a'_t \mid s'_t)}_{g(\theta_k, \tau')} \qquad \text{(Stochastic PG)}$$

where $\tau' \equiv (s'_0, a'_0, r'_1, s'_1, a'_1, r'_2 \cdots)$

- $g(\theta_k, \tau')$ is an **unbiased** estimate of exact PG $\nabla_\theta V^{\pi_\theta}(\mu)|_{\theta=\theta_k}$
- An unbiased estimate can be constructed from 1 or multiple trajectories

More Generally: Stochastic Gradient Descent (SGD) for Stochastic Optimization

Stochastic Optimization:

$$\theta^* = \arg\max_{\theta \in \Theta} \; F(\theta), \text{where } F(\theta) := \mathbb{E}_{\xi}[f(\theta; \xi)]$$

where ξ is the randomness (possibly unknown) in our problem

$$\theta_{k+1} = \theta_k - \eta_k \cdot \mathbb{E}[\nabla_\theta f(\theta_k; \xi_k)] \quad \Rightarrow \quad \theta_{k+1} = \theta_k - \eta_k \cdot g(\theta_k; \xi_k)$$

(GD) (SGD)

- $g(\theta_k; \xi_k)$ can be constructed from 1 or multiple samples (mini-batch)
- Advantage: SGD has a low computational cost in each iteration

[Robbins & Monro, 1951]

SGD: A Special Case of “Stochastic Approximation”

A STOCHASTIC APPROXIMATION METHOD[1]

By Herbert Robbins and Sutton Monro

University of North Carolina

1. Summary. Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making successive experiments at levels $x_1, x_2, \cdots$ in such a way that x_n will tend to θ in probability.

2. Introduction. Let $M(x)$ be a given function and α a given constant such that the equation

$$M(x) = \alpha \tag{1}$$

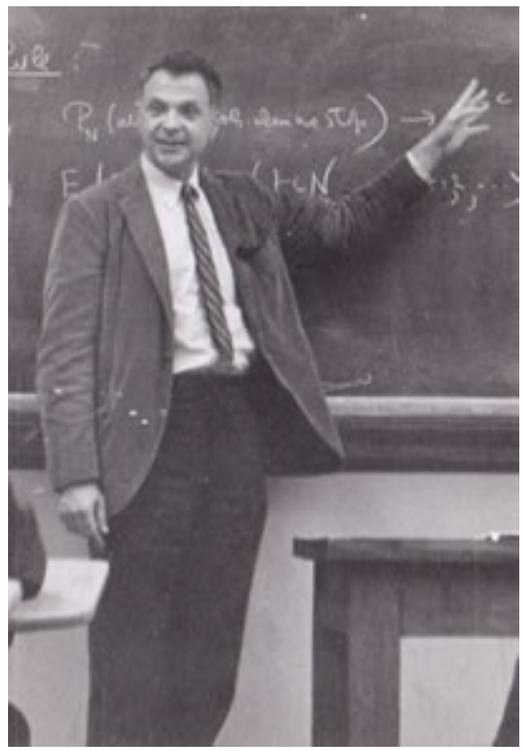
has a unique root $x = \theta$. There are many methods for determining the value of θ by successive approximation. With any such method we begin by choosing one or more values $x_1, \cdots, x_r$ more or less arbitrarily, and then successively obtain new values x_n as certain functions of the previously obtained $x_1, \cdots, x_{n-1}$, the values $M(x_1), \cdots, M(x_{n-1})$, and possibly those of the derivatives $M'(x_1), \cdots, M'(x_{n-1})$, etc. If

$$\lim_{n\to\infty} x_n = \theta, \tag{2}$$

irrespective of the arbitrary initial values $x_1, \cdots, x_r$, then the method is effective for the particular function $M(x)$ and value α. The speed of the convergence in (2) and the ease with which the x_n can be computed determine the practical utility of the method.

We consider a stochastic generalization of the above problem in which the nature of the function $M(x)$ is unknown to the experimenter. Instead, we suppose that to each value x corresponds a random variable $Y = Y(x)$ with distribution function $Pr[Y(x) \leq y] = H(y \mid x)$, such that

$$M(x) = \int_{-\infty}^{\infty} y \, dH(y \mid x) \tag{3}$$



Herbert Robbins Sutton Monro

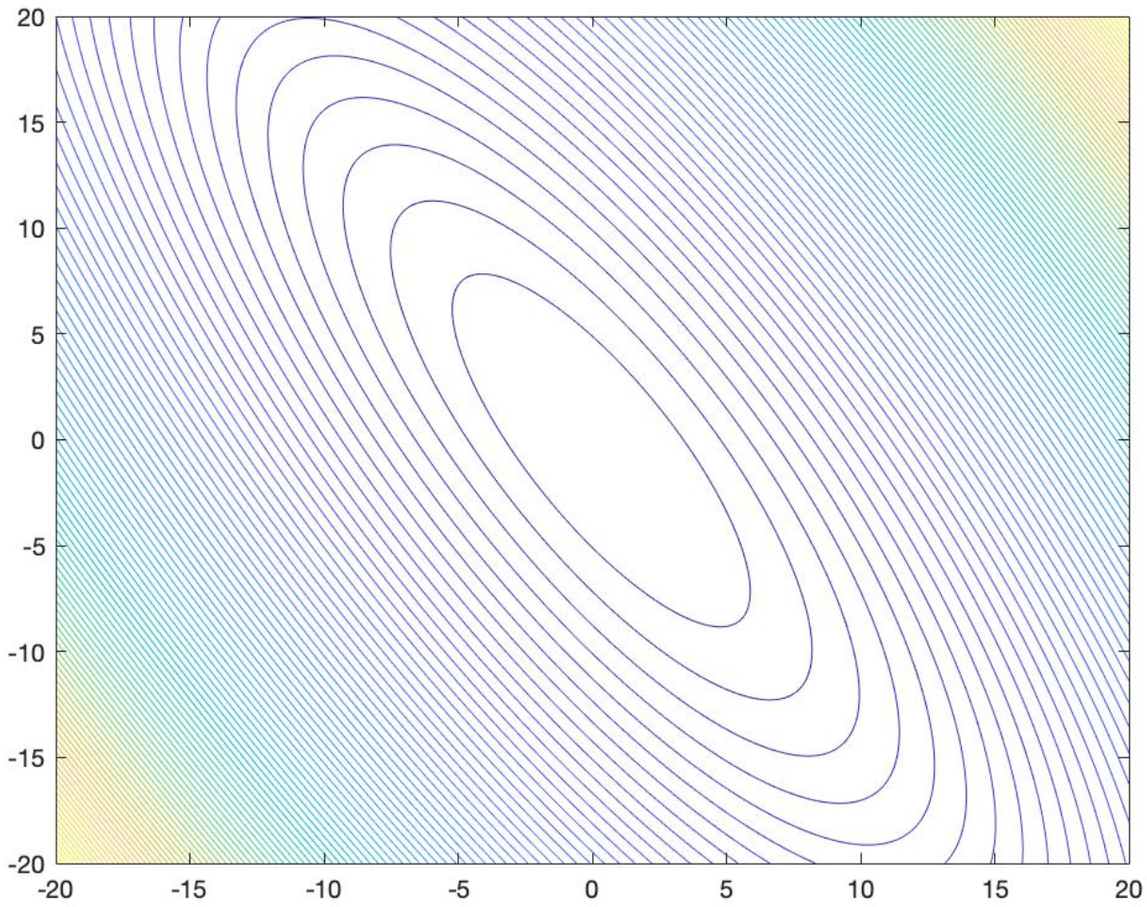
“Stochastic approximation” is the core of many RL methods, e.g., Q-learning and TD learning

A seminal paper on stochastic approximation in 1951

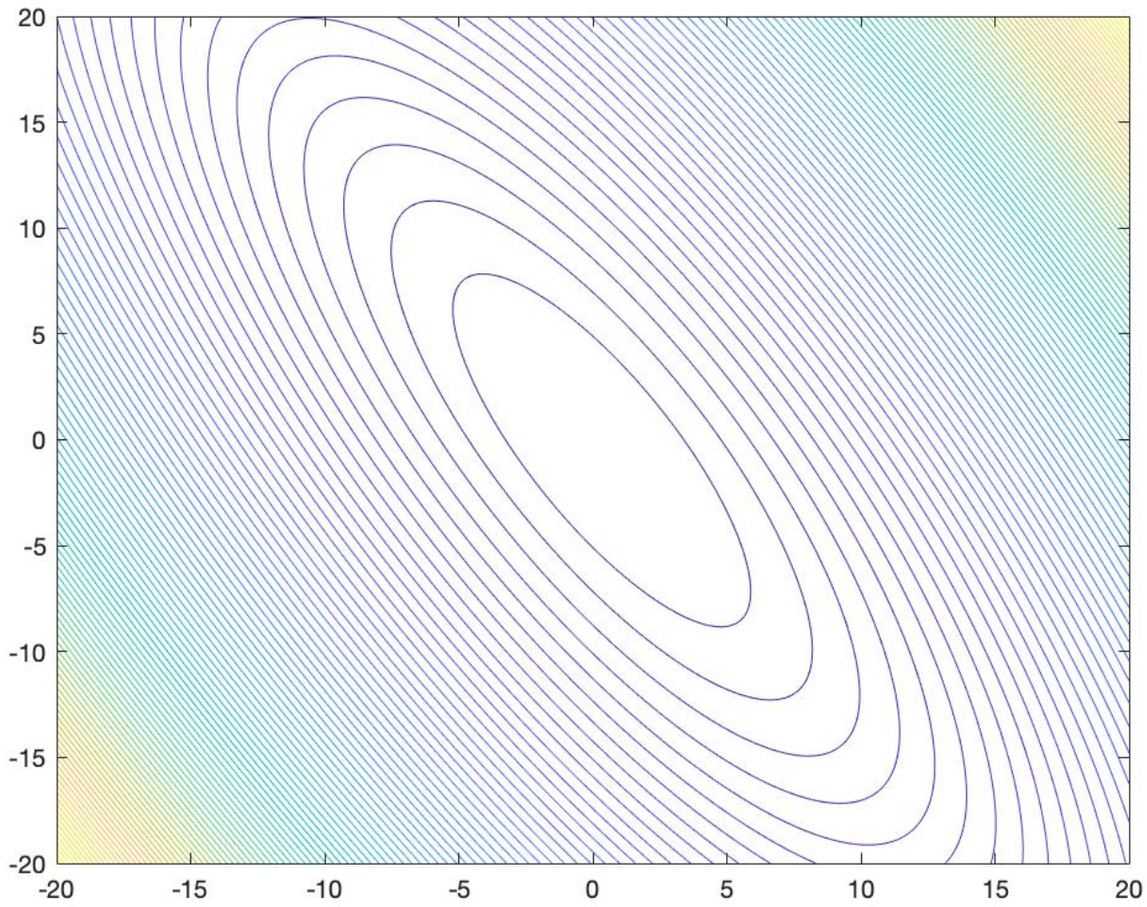
(Cited for more than 11000 times)

Visualization: SGD vs GD

- SGD usually exhibits more “random” behavior than GD



GD



SGD

Let’s combine SGD and PG!

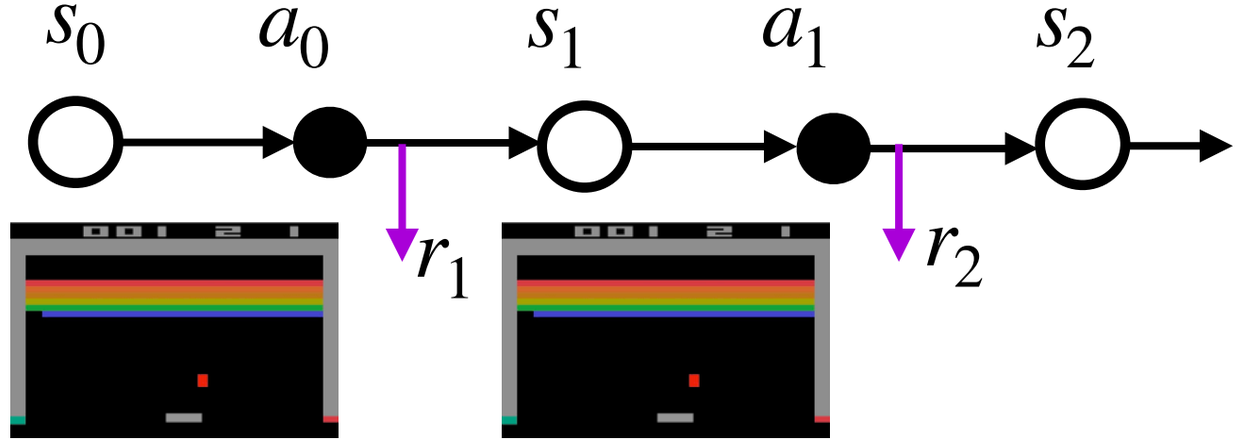
Combining SGD and Policy Gradient

- Recall (P2): $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\Big[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big]$

- In each iteration, we draw a trajectory $\tau = (s_0, a_0, r_1, s_1, a_1 \cdots)$ under π_θ and μ, and then construct:

$$G_t(\tau) := \sum_{m=t}^{\infty} \gamma^m r_{m+1}$$

$$\hat{\nabla}_\tau := \sum_{t=0}^{\infty} \gamma^t G_t(\tau)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$



- Question: Is $\hat{\nabla}_\tau$ an **unbiased** estimate of $\nabla_\theta V^{\pi_\theta}(\mu)$?

- (P2): $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$
- Show that $\hat{\nabla}_\tau$ is an **unbiased** estimate of $\nabla_\theta V^{\pi_\theta}(\mu)$

$$G_t(\tau) := \sum_{m=t}^{\infty} \gamma^m r_{m+1}$$

$$\hat{\nabla}_\tau := \sum_{t=0}^{\infty} \gamma^t G_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

The REINFORCE Algorithm (Formally)

- REINFORCE algorithm (aka *Monte Carlo policy gradient*)

Step 1: Initialize θ_0 and step size η

Step 2: Sample a trajectory $\tau \sim P_{\mu}^{\pi_\theta}$ and make the update as

$$\theta_{k+1} = \theta_k + \eta \cdot \nabla_\tau$$

$$= \theta_k + \eta \Big(\sum_{t=0}^{\infty} \gamma^t G_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Big)$$

(Repeat Step 2 until termination)

William, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” ML 1992

Can we design RL algorithms by using (P1) and (P3)?

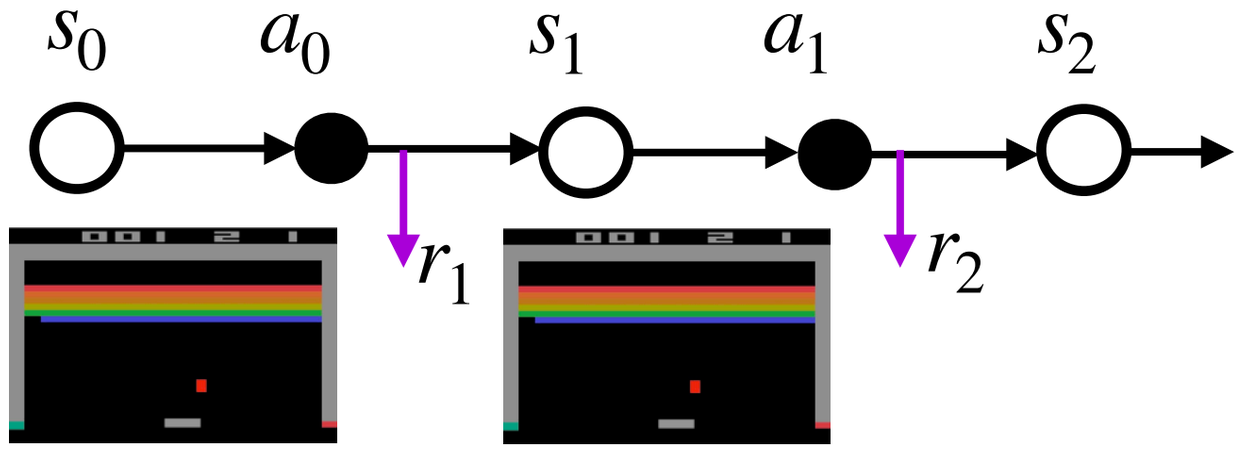
An Alternative Monte-Carlo Policy Gradient Algorithm

- Recall (P1): $\nabla_\theta V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[G(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$

Step 1: In each iteration k, draw a trajectory $\tau = (s_0, a_0, r_1, s_1, a_1 \cdots)$ under π_θ and μ, and then construct:

$$G(\tau) := \sum_{t=0}^{\infty} \gamma^t r_t$$

$$\bar{\nabla}_\tau := R(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$



Step 2: Apply $\theta_{k+1} = \theta_k + \eta \cdot \bar{\nabla}_\tau$

- Question: Is $\bar{\nabla}_\tau$ an unbiased estimate of $\nabla_\theta V^{\pi_\theta}(\mu)$?
- Question: Any difference from REINFORCE?

How About Using (P3)?

- Recall (P3): $\nabla_\theta V^{\pi_\theta}(\mu) = \dfrac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\,\nabla_\theta \log \pi_\theta(a\,|\,s)\right]$

Step 1: In each iteration k, draw a batch B of n state-action pairs from $d_\mu^{\pi_\theta}$ and π_θ and construct

$$\tilde{\nabla}_\tau := \frac{1}{1-\gamma}\cdot\left(\frac{1}{n}\sum_{(s,a)\in B} Q^{\pi_\theta}(s,a)\,\nabla_\theta \log \pi_\theta(a\,|\,s)\right)$$

Step 2: Apply $\theta_{k+1} = \theta_k + \eta\cdot\tilde{\nabla}_\tau$

- Question: What does "draw samples from $d_\mu^{\pi_\theta}$" really mean?

  Are the samples i.i.d.?

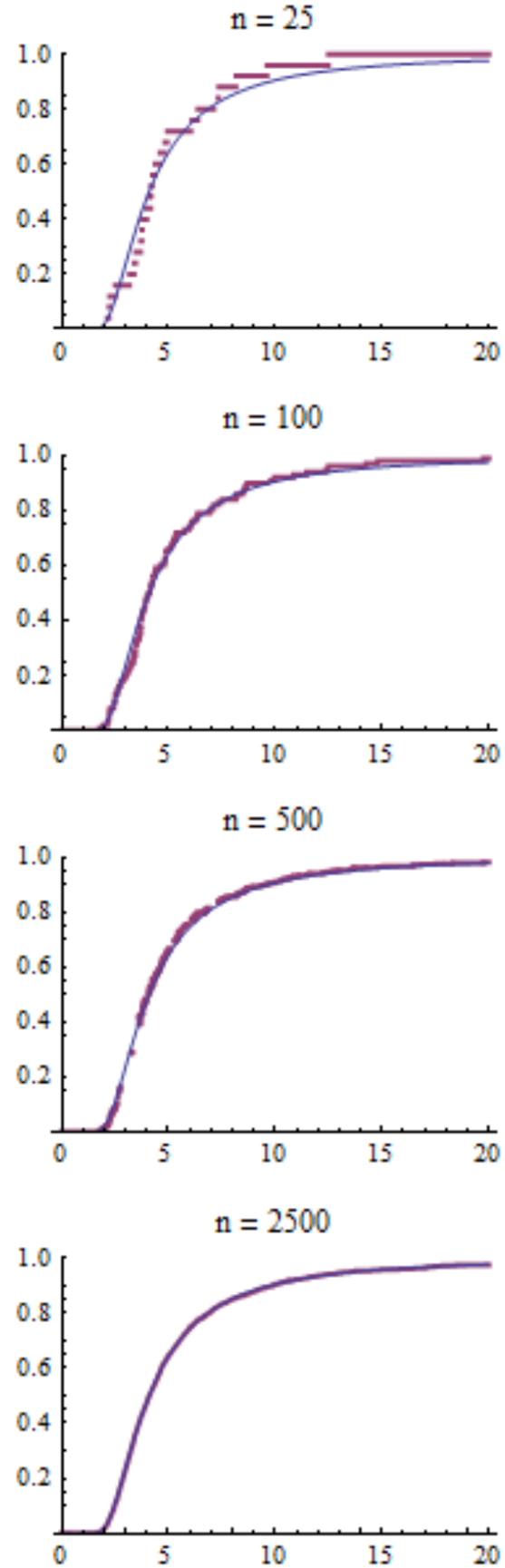
An Untold Secret in RL Community…

- $\tilde{\nabla}_\tau$ for (P3) is actually NOT an unbiased estimator of the true PG!

- But for large n, $\tilde{\nabla}_\tau$ can still nicely approximate the true PG (Why?)

A Fundamental Property:

Empirical distribution uniformly approximates the true distribution!



https://www.wolfram.com/mathematica/new-in-8/probability-and-statistics-solvers-and-properties/demonstrate-the-glivenko-cantelli-theorem.html

Gilvenko-Cantelli Theorem (Formally)

Empirical distribution uniformly approximates the true distribution

- Let $\{X_n, n \geq 1\}$ be a sequence of i.i.d. random variables with a common CDF F
- Define empirical CDF as $\hat{F}_n(x) := \frac{1}{n} \sum_{i=1}^{n} I\{X_i \leq x\}$
- Define $D_n := \sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)|$

- **Gilvenko-Cantelli Theorem:**

$$D_n \to 0, \text{ as } n \to \infty$$

- This result could be extended to Markov chains

Stute and Schmann, “A General Gilvenko-Cantelli Theorem for Stationary Sequences of Random Observations,” 1980

Yet Another Untold Secret in RL Community…

- RL people usually ignore the effect of γ on $d_\mu^{\pi_\theta}$ (which is not theoretically justified)

Is the Policy Gradient a Gradient?

Chris Nota
College of Information and Computer Sciences
University of Massachusetts Amherst
cnota@cs.umass.edu

Philip S. Thomas
College of Information and Computer Sciences
University of Massachusetts Amherst
pthomas@cs.umass.edu

ABSTRACT

The policy gradient theorem describes the gradient of the expected discounted return with respect to an agent's policy parameters. However, most policy gradient methods drop the discount factor from the state distribution and therefore do not optimize the discounted objective. What do they optimize instead? This has been an open question for several years, and this lack of theoretical clarity has lead to an abundance of misstatements in the literature. We answer this question by proving that the update direction approximated by most methods is not the gradient of any function. Further, we argue that algorithms that follow this direction are not guaranteed to converge to a "reasonable" fixed point by constructing a counterexample wherein the fixed point is globally *pessimal* with respect to both the discounted and undiscounted objectives. We motivate this work by surveying the literature and showing that there remains a widespread misunderstanding regarding discounted policy gradient methods, with errors present even in highly-cited papers published at top conferences.

is pessimal, regardless of whether the discounted or undiscounted objective is considered.

The analysis in this paper applies to nearly all state-of-the-art policy gradient methods. In Section 6, we review all of the policy gradient algorithms included in the popular `stable-baselines` repository [9] and their associated papers, including A2C/A3C [13], ACER [28], ACKTR [30], DDPG [11], PPO [18], TD3 [6], TRPO [16], and SAC [8]. We motivate this choice in Section 6, but we note that all of these papers were published at top conferences[1] and have received hundreds or thousands of citations. We found that all of the implementations of the algorithms used the "incorrect" policy gradient that we discuss in this paper. While this is a valid algorithmic choice if properly acknowledged, we found that only *one* of the eight papers acknowledged this choice, while three of the papers made erroneous claims regarding the discounted policy gradient and others made claims that were misleading. The purpose of identifying these errors is not to criticize the authors or the algorithms, but to draw attention to the fact that confusion regarding the behavior of policy gradient algorithm exists at the very core of the RL community and has gone largely unnoticed by reviewers.

Philip Thomas

Chris Nota and Philip Thomas, "Is the Policy Gradient a Gradient?," AAMAS 2020