10703 Deep Reinforcement Learning

Policy Gradient Methods

Tom Mitchell

October 1, 2018

Reading: Barto & Sutton, Chapter 13

Used Materials

- Much of the material and slides for this lecture were taken from Chapter 13 of Barto & Sutton textbook.

- Some slides are borrowed from Ruslan Salakhutdinov, who in turn borrowed from Rich Sutton's RL class and David Silver's Deep RL tutorial

Policy-Based Reinforcement Learning

- So far we approximated the value or action-value function using parameters θ (e.g. neural networks)

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s, a) \approx Q^\pi(s, a)$$

- A policy was generated directly from the value function e.g. using ε-greedy

- In this lecture we will directly parameterize the policy

$$\pi_\theta(s, a) = \mathbb{P}\left[a \mid s, \theta\right]$$

- We will focus again on model-free reinforcement learning

Policy-Based Reinforcement Learning

- So far we approximated the value or action-value function using parameters θ (e.g. neural networks)

$$V_\theta(s) \approx V^\pi(s)$$

$$Q_\theta(s, a) \approx Q^\pi(s, a)$$

- A polic .g. using ε-greedy

Sometimes I will also use the notation:

$$\pi(A_t | S_t, \boldsymbol{\theta})$$

- In this l

$$\pi_\theta(s, a) = \mathbb{P}[a \mid s, \theta]$$

- We will focus again on model-free reinforcement learning

Typical Parameterized Differentiable Policy

- Softmax:

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}},$$

where $h(s,a,\theta)$ is any function of s, a with params θ

e.g., linear function of features x(s,a) you make up

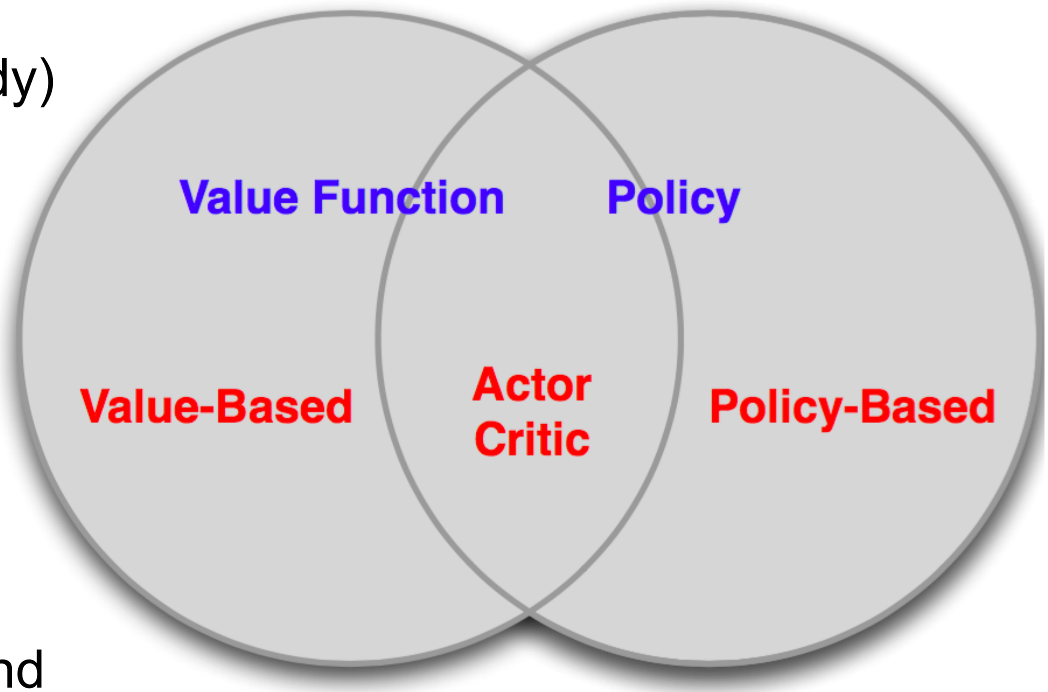
$$h(s,a,\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}(s,a)$$

$$\mathbf{x}(s,a) \in \mathbb{R}^d$$

e.g., $h(s,a,\theta)$ is output of trained neural net

Value-Based and Policy-Based RL

- Value Based
  - Learn a Value Function
  - Implicit policy (e.g. ε-greedy)
- Policy Based
  - Learn a Policy directly
- Actor-Critic
  - Learn a Value Function, and
  - Learn a Policy



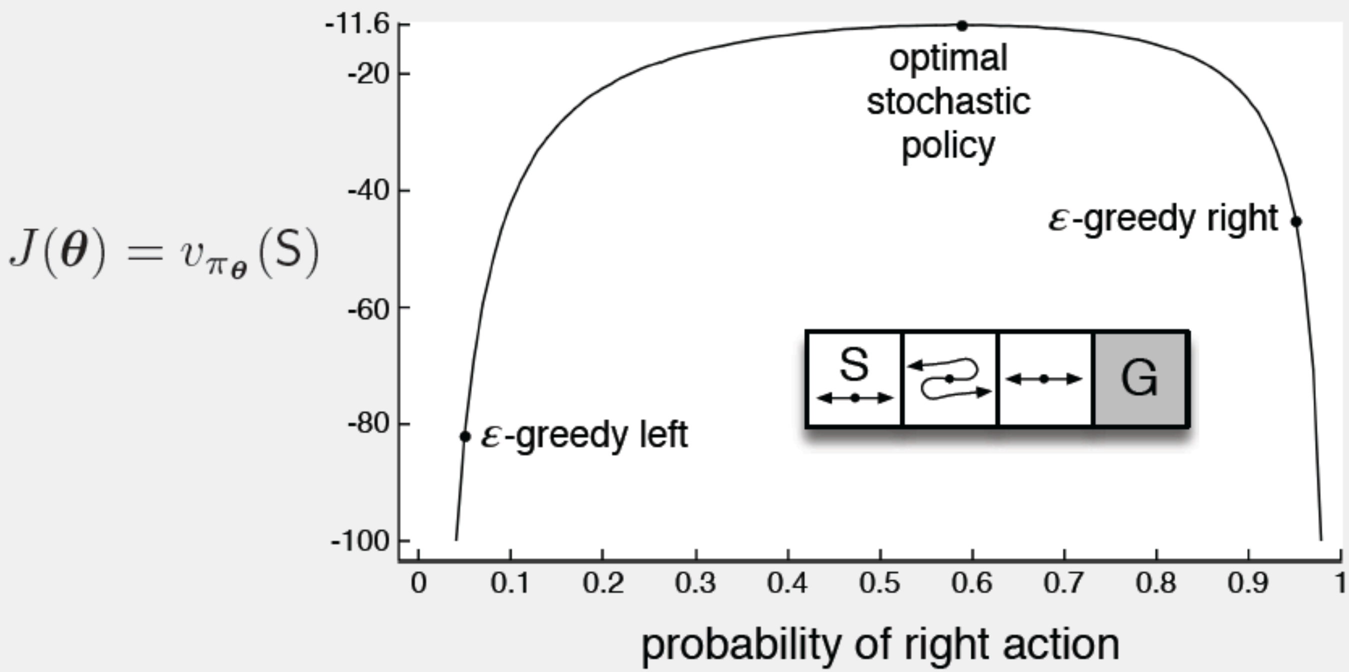
Advantages of Policy-Based RL

- Advantages
  - Better convergence properties
  - Effective in high-dimensional, even continuous action spaces
  - Can learn stochastic policies

- Disadvantages
  - Typically converge to a local rather than global optimum

Example: Why use non-deterministic policy?

Consider the small corridor gridworld shown inset in the graph below. The reward is -1 per step, as usual. In each of the three nonterminal states there are only two actions, right and left. These actions have their usual consequences in the first and third states (left causes no movement in the first state), but in the second state they are reversed, so that right moves to the left and left moves to the right. The problem is difficult because all the states appear identical under the function approximation. In particular, we define $\mathbf{x}(s, \mathsf{right}) = [1, 0]^\top$ and $\mathbf{x}(s, \mathsf{left}) = [0, 1]^\top$, for all s. An action-value method with ε-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \varepsilon/2$ on all steps or choosing left with the same high probability on all time steps. If $\varepsilon = 0.1$, then these two policies achieve a value (at the start state) of less than -44 and -82, respectively, as shown in the graph. A method can do significantly better if it can learn a specific probability with which to select right. The best probability is about 0.59, which achieves a value of about -11.6.



What Policy Learning Objective?

- Goal: given policy $\pi_\theta(s,a)$ with parameters θ, wish to find best θ
  - define "best θ" as $\arg\max_\theta J(\theta)$ for some $J(\theta)$
- In episodic environments we can optimize the value of start state s_1

$$J_1(\theta) = V^{\pi_\theta}(s_1)$$

Remember: Episode of experience under policy π: $S_1, A_1, R_2, ..., S_k \sim \pi$

What Policy Learning Objective?

- Goal: given policy $\pi_\theta(s,a)$ with parameters θ, wish to find best θ
  - define "best θ" as $\text{argmax}_\theta\, J(\theta)$ for some $J(\theta)$
- In episodic environments we can optimize the value of start state s_1

$$J_1(\theta) = V^{\pi_\theta}(s_1)$$

- In continuing environments we can optimize the average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or the average immediate reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathcal{R}_s^a$$

where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

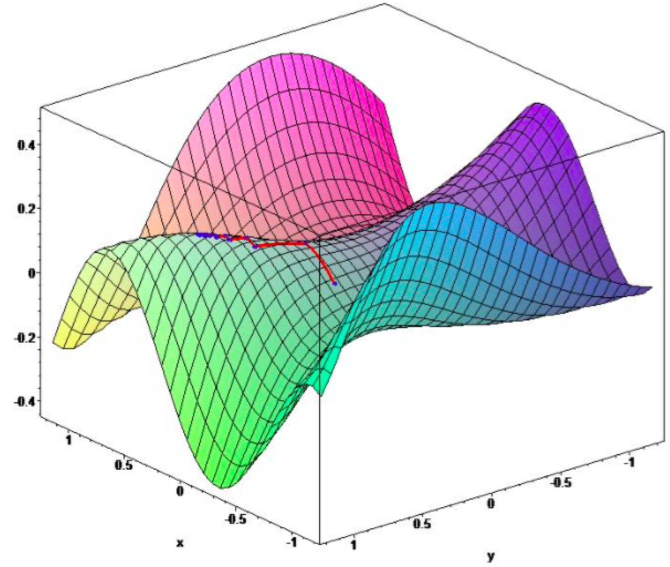
Policy Optimization

- Policy based reinforcement learning is an optimization problem
  - Find θ that maximizes J(θ)
- Some approaches do not use gradient
  - Hill climbing
  - Genetic algorithms
- Greater efficiency often possible using gradient
  - Gradient descent
  - Conjugate gradient
  - Quasi-Newton
- We focus on gradient ascent, many extensions possible
- And on methods that exploit sequential structure

Gradient of Policy Objective

- Let J(θ) be any policy objective function

- Policy gradient algorithms search for a local maximum in J(θ) by ascending the gradient of the policy, w.r.t. parameters θ



$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

α is a step-size parameter (learning rate)

is the policy gradient

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

Computing Gradients By Finite Differences

- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension k in [1, n]
  - Estimate k^{th} partial derivative of objective function w.r.t. θ
  - By perturbing θ by small amount ε in k^{th} dimension

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k^{th} component, 0 elsewhere

- Uses n evaluations to compute policy gradient in n dimensions
- Simple, inefficient – but general purpose!
- Works for arbitrary policies, even if policy is not differentiable

How do we find an expression for $\nabla J(\boldsymbol{\theta})$?

Consider episodic case: $J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$

Problem in calculating $\nabla J(\boldsymbol{\theta})$
doesn't a change to θ alter both:

- action chosen by π_θ in each state s
- distribution of states we'll encounter

Remember: Episode of experience under policy π: $S_1, A_1, R_2, ..., S_k \sim \pi$

How do we find an expression for $\nabla J(\boldsymbol{\theta})$?

Consider episodic case: $J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$

Problem in calculating $\nabla J(\boldsymbol{\theta})$
doesn't a change to θ alter both:

- action chosen by π_θ in each state s
- distribution of states we'll encounter

Good news: policy gradient theorem:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

where $\mu(s)$ is probability distribution over states

Proof of the Policy Gradient Theorem (episodic case)

With just elementary calculus and re-arranging of terms, we can prove the policy gradient theorem from first principles. To keep the notation simple, we leave it implicit in all cases that π is a function of $\boldsymbol{\theta}$, and all gradients are also implicitly with respect to $\boldsymbol{\theta}$. First note that the gradient of the state-value function can be written in terms of the action-value function as

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} [\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'')] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

SGD Approach to Optimizing J(θ) : Approach 1

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right].$$

We could stop here and instantiate our stochastic gradient-ascent

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta}),$$

SGD Approach to Optimizing J(θ) : Approach 2

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \\ &= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right].\end{aligned}$$

SGD Approach to Optimizing J(θ) : Approach 2

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right].$$

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right]$$
$$= \mathbb{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \qquad \text{(replacing } a \text{ by the sample } A_t \sim \pi\text{)}$$
$$= \mathbb{E}_\pi\left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right], \qquad \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t)\text{)}$$

SGD Approach to Optimizing J(θ) : Approach 2

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right].$$

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right]$$
$$= \mathbb{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \qquad \text{(replacing } a \text{ by the sample } A_t \sim \pi\text{)}$$
$$= \mathbb{E}_\pi\left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right], \qquad \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t)\text{)}$$

$$\boxed{\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}}$$

REINFORCE algorithm

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

$$\boxed{\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}}$$

Note $$\frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} = \nabla \ln \pi(A_t|S_t, \theta)$$

because $$\frac{d \ln x}{dx} = \frac{1}{x}$$

Typical Parameterized Differentiable Policy

- Softmax:

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}},$$

where $h(s,a,\theta)$ is any function of s, a with params θ

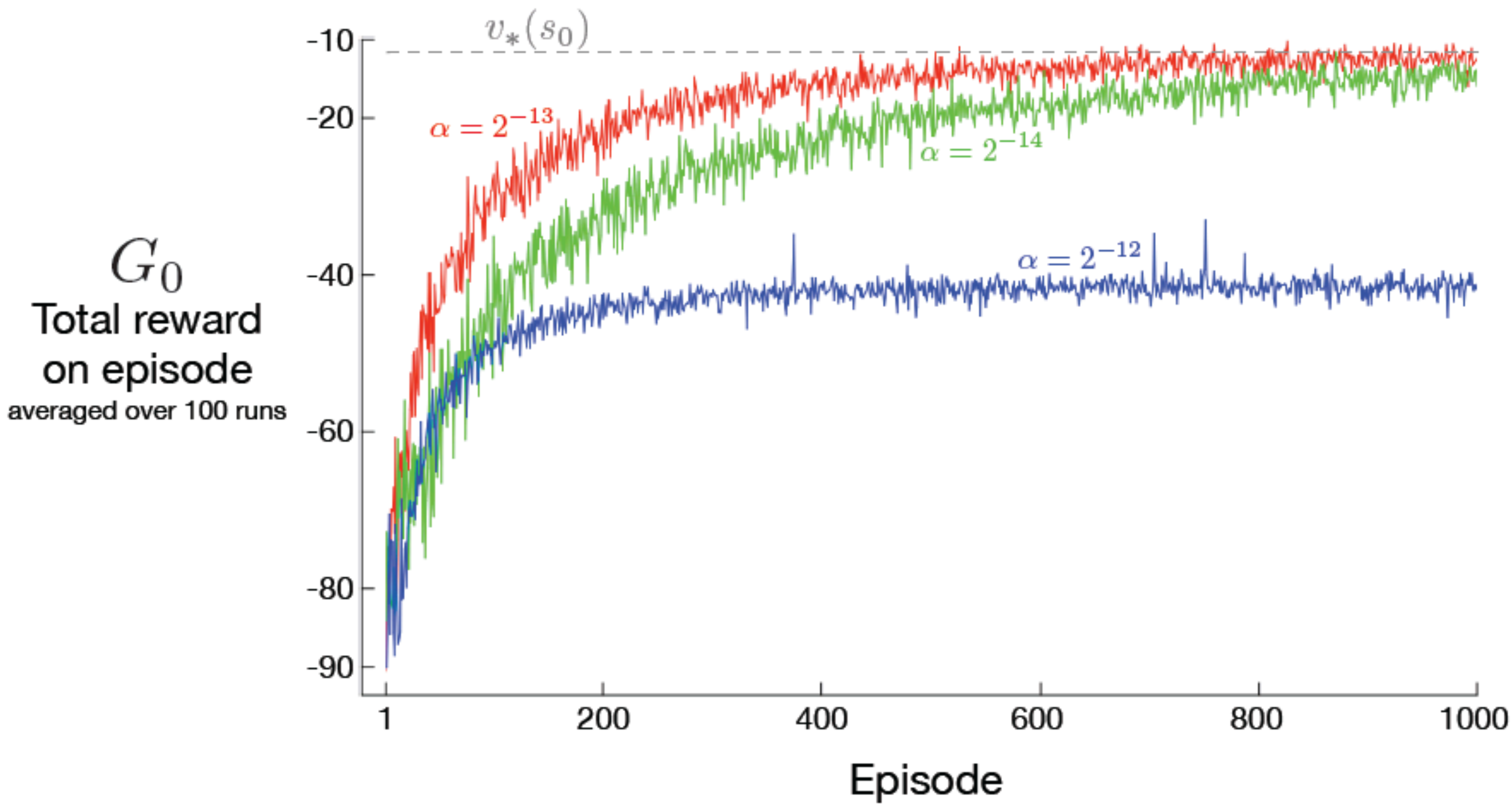
e.g., linear function of features x(s,a) you make up

$$h(s,a,\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}(s,a)$$

$$\mathbf{x}(s,a) \in \mathbb{R}^d$$

e.g., $h(s,a,\theta)$ is output of trained neural net

REINFORCE algorithm on Short Corridor World



Good news:

- REINFORCE converges to local optimum under usual SGD assumptions
- because $E_{\pi}[G_t] = Q(S_t,A_t)$

But variance is high

- recall high variance of Monte Carlo sampling

Good news:

- REINFORCE converges to local optimum under usual SGD assumptions
- because $E_{\pi}[G_t] = Q(S_t,A_t)$

But variance is high

- recall high variance of Monte Carlo sampling

Adding a baseline to REINFORCE Algorithm

replace $$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

by $$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s) \Big) \nabla \pi(a|s,\boldsymbol{\theta})$$

for some fixed function b(s) that captures prior for s

Note the equation is still valid because

$$\sum_a b(s) \nabla \pi(a|s,\boldsymbol{\theta}) \;=\; b(s) \nabla \sum_a \pi(a|s,\boldsymbol{\theta}) \;=\; b(s) \nabla 1 \;=\; 0$$

Result: $$\boxed{\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big(G_t - b(S_t) \Big) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}}$$

Adding a baseline to REINFORCE Algorithm

replacing $$\boxed{\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}}$$

by $$\boxed{\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big(G_t - b(S_t)\Big) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}}$$

for a good $b(S_t)$ reduces variance in training target

one typical b(S) is a learned value function

$b(S_t) = \hat{v}(S_t, w)$

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

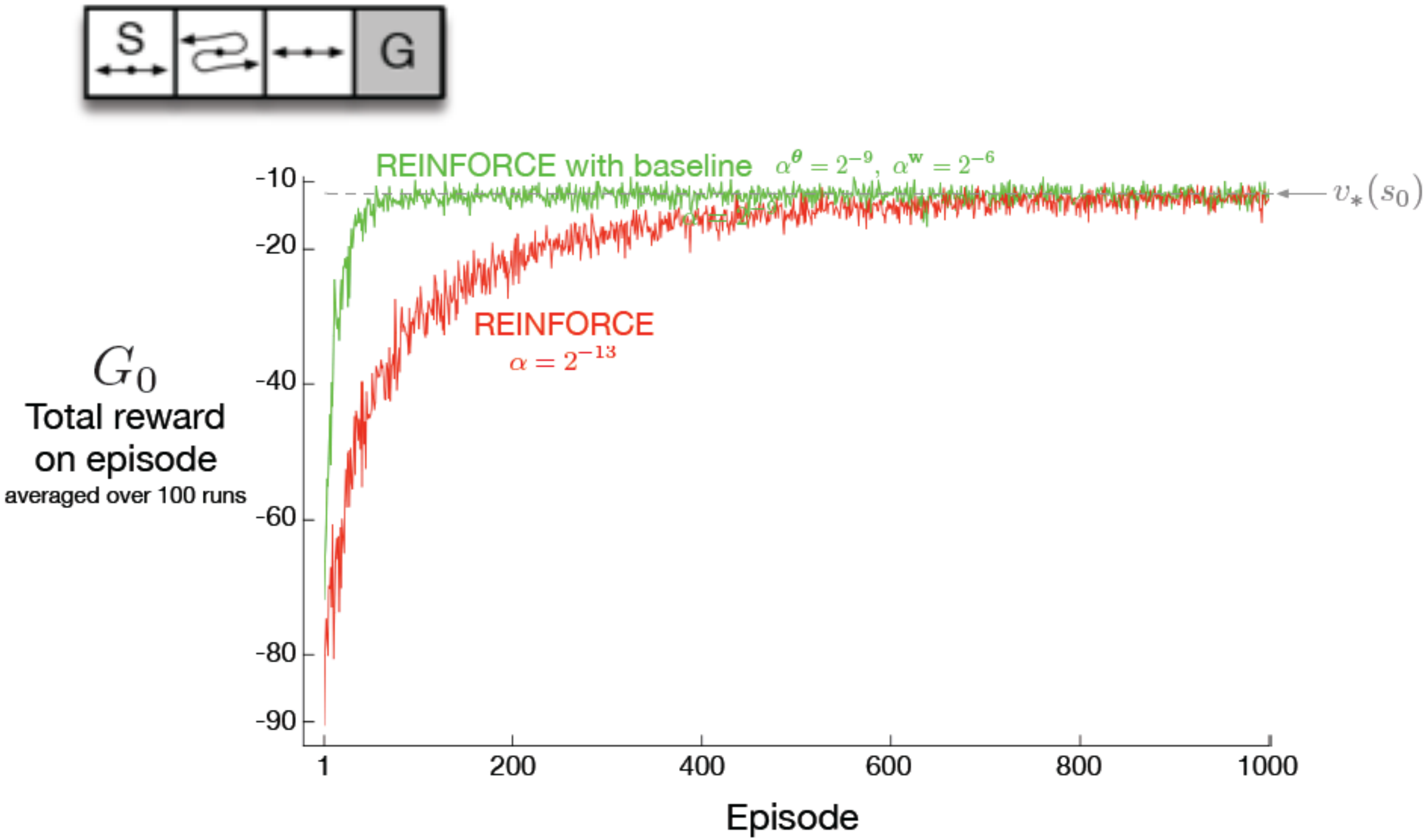


Figure 13.2: Adding a baseline to REINFORCE can make it learn much faster, a

Good news:

- REINFORCE converges to local optimum under usual SGD assumptions
- because $E_\pi[G_t] = Q(S_t,A_t)$

But variance is high

- recall high variance of Monte Carlo sampling

Actor-Critic Model

- learn both Q and π
- use Q to generate target values, instead of G

One step actor-critic model:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha\Big(G_{t:t+1} - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\
&= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\
&= \boldsymbol{\theta}_t + \alpha\,\delta_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}.
\end{aligned}$$

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$