Policy Gradients with Function Approximation

Actor-Critic Methods

Reading

- Reinforcement Learning
  - http://www.incompleteideas.net/book/the-book-2nd.html
  - Chapter 13.4-13.7
- Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems* 12 (1999).
- David Silver lecture on Policy Gradients
  - https://www.youtube.com/watch?v=KHZVXao4qXs&t=3s

Overview

- REINFORCE algorithm can work well in some settings but it has to wait for returns at the end of the episode
- Suffers from similar issues as Monte Carlo methods
  - Large variance
  - Slow convergence
- Essentially does not use the Bellman equation
- We will discuss a similar progression of algorithms as in value-based methods
  - Add function approximation
  - Add bootstrapping (actor-critic methods)

REINFORCE algorithm, cont'd

- Final form for the gradient is

$$\nabla v_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=k}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\, G_k \middle| S_k = s\right]$$

- Once we have the gradient, update weights as usual

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \nabla v_{\pi_{\boldsymbol{\theta}}}(s)$$

  – This is similar to the Monte Carlo learning method where we wait until the end of the episode to observe G_t

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)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Issues with REINFORCE

- Can you spot any issues with this iteration?

$$\nabla v_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=k}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\, G_k \middle| S_k = s\right]$$

  - How important is the magnitude of G_k?
  - Turns out quite a bit – tasks have greatly varying returns
  - Especially problematic if *good* runs have zero returns
    - Gradient is 0!
- Vanilla REINFORCE has very large variance depending on G_k
- How do we address this issue?
  - Need to somehow normalize the returns

REINFORCE with Baseline

- Can add an arbitrary baseline $b(s)$ to compare to the action value for each state

$$\nabla v_\pi(s_0) = \mathbb{E}_\pi\left[\sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\,(G_1 - b)|S_t = s_0\right]$$

- Similar to the update in Q-learning
- Expectation remains the same as long as b is not a function of the action a
  - Why?

$$\nabla_{\boldsymbol{\theta}} v_\pi(s_0) = \nabla_{\boldsymbol{\theta}} \mathbb{E}[G_1 - b|S_1 = s_0]$$

  - Since $\nabla_{\boldsymbol{\theta}} b = 0$ when b is not a function of a

REINFORCE with Baseline: minimize variance

- Can add an arbitrary baseline b

$$\nabla v_\pi(s_0) = \mathbb{E}_\pi\left[\sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\,(G_1 - b)|S_t = s_0\right]$$

- Can pick b to minimize variance
  - Recall $Var[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2$
  - The variance of the gradient update (in 1D) is

$$\mathbb{E}_\pi\left[\left(\sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\,(G_1 - b)\right)^2\right] - \left(\mathbb{E}_\pi\left[\sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\,(G_1 - b)\right]\right)^2$$

  - Note that the 2nd term is not affected by the value of b
    - Goes away when taking the gradient w.r.t. b
  - Dropped the conditioning to simplify notation

REINFORCE with Baseline: minimize variance, cont'd

- Differentiating w.r.t. b

$$\frac{dVar}{db} = \frac{d}{db}\mathbb{E}_\pi\left[\left((G_1 - b)\sum_{t=1}^{T}\nabla \log\big(\pi(A_t|S_t)\big)\right)^2\right]$$
$$= \frac{d}{db}\left[\mathbb{E}_\pi\left[g^2 G_1^2\right] - b2\mathbb{E}_\pi\left[g^2 G_1\right] + b^2\mathbb{E}_\pi\left[g^2\right]\right]$$
$$= -2\mathbb{E}_\pi\left[g^2 G_1\right] + 2b\mathbb{E}_\pi\left[g^2\right]$$

  - where $g \coloneqq \sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)$

- Setting it equal to 0 and solving for b, we get

$$b = \frac{\mathbb{E}_\pi\left[g^2 G_1\right]}{\mathbb{E}_\pi\left[g^2\right]}$$

  - Will reduce the algorithm's sensitivity to large variance of G_t
  - Issues?
    - Estimating expectations may be hard

REINFORCE with Baseline: running state value estimate

- Can add an arbitrary baseline b

$$\nabla v_\pi(s_0) = \mathbb{E}_\pi\left[\sum_{t=1}^{T} \nabla \log\big(\pi(A_t|S_t)\big)\,(G_1 - b)|S_t = s_0\right]$$

- What else can we do?
  - Can pick b to be a running estimate of the current state value
  - Can have a parameterized $\hat{v}_{\boldsymbol{w}}(s)$ estimator
    - Pick $\boldsymbol{w}$ to minimize a loss, e.g.,

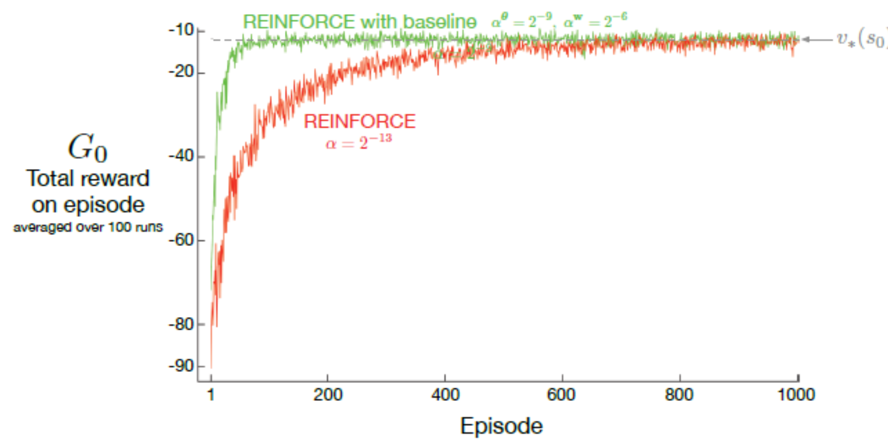
$$\big(G_t - \hat{v}_{\boldsymbol{w}}(S_t)\big)^2$$

    - Can perform gradient descent (with chain rule) after each iteration

$$\boldsymbol{w}' = \boldsymbol{w} + \alpha_{\boldsymbol{w}} 2\big(G_t - \hat{v}_{\boldsymbol{w}}(S_t)\big)\nabla_{\boldsymbol{w}}\hat{v}_{\boldsymbol{w}}(S_t)$$

REINFORCE with Baseline, cont'd

- REINFORCE with state value estimates as baseline
- Lower variance means much faster convergence



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)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Policy gradients with function approximation

- REINFORCE with baseline tries to estimate each state's value
  - Greatly reduces variance if done well
- But we still need to wait for returns
- What else can we do?

$$\nabla v_\pi(s_0) = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

  - We can try to approximate the q function!
  - Then use the approximation $\hat{q}$ in the policy gradient
- What is a potential issue with that approach?
  - Unclear if the true policy gradient is still followed
  - Unclear if it converges (and what it converges to)

Policy gradients with function approximation, cont'd

- What else can we do?

$$\nabla v_\pi(s_0) = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

  - We can try to approximate the q function![1]
- Suppose we use an approximation $f_{\boldsymbol{w}}$ of q_π
  - How do we train $f_{\boldsymbol{w}}$?
  - One option is to use least squares as usual

$$\left(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\right)^2$$

  - As usual, we don't know the true q values
    - Can use G_t instead – will learn the same $\boldsymbol{w}$ in expectation

[1]Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems* 12 (1999).

Policy gradients with function approximation, cont'd

- Suppose we try to minimize least squares

$$\left(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\right)^2$$

- We can use the same algorithm as REINFORCE with baseline
  - except now we use the other form of the policy gradient
  - For each step of an episode: $t = 0,1,\ldots,T$
    - $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    - $\delta_t = G_t - f_{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{w}' = \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta_t \nabla f_{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \nabla \pi(A_t|S_t) f_{\boldsymbol{w}}(S_t, A_t)$
- Can you spot any issues?
  - Algorithm may be very noisy depending on quality of $f_{\boldsymbol{w}}$
  - May not ever converge

Compatible Approximations

- Recall the policy gradient

$$\nabla v_\pi(s_0) = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

- Suppose we use an approximation $f_{\boldsymbol{w}}$ of $q_\pi(s,a)$
  - What property would $f_{\boldsymbol{w}}$ have ideally?

$$\nabla v_\pi(s_0) = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) f_{\boldsymbol{w}}(s,a)$$

  - We follow the correct gradient when we update $\boldsymbol{\theta}$
  - This is known as a compatible approximation

Compatible Approximation Property

- Suppose we train $f_{\boldsymbol{w}}$ until convergence, i.e.,

$$\nabla_{\boldsymbol{w}}\big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big)^2 = 0$$

- i.e.,

$$\big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big)\nabla_{\boldsymbol{w}} f(s,a) = 0$$

- This means that $f_{\boldsymbol{w}}$ is the least squares estimator of q_π
  - Thus, $f_{\boldsymbol{w}}$ is un unbiased estimator of q_π, i.e.,

$$\mathbb{E}_{d_\pi}\big[\big(q_\pi(S,A) - f_{\boldsymbol{w}}(S,A)\big)\nabla_{\boldsymbol{w}} f(S,A)\big] = 0$$

- How do we expand that expected value?

$$\mathbb{E}_{d_\pi}\big[\big(q_\pi(S,A) - f_{\boldsymbol{w}}(S,A)\big)\nabla_{\boldsymbol{w}} f(S,A)\big] =$$

$$= \sum_{a,s} \mathbb{P}_{d_\pi}[S = s, A = a]\,\big(q_\pi(S,A) - f_{\boldsymbol{w}}(S,A)\big)\nabla_{\boldsymbol{w}} f(S,A)$$

$$= \sum_{s} d_\pi(s) \sum_{a} \pi(a|s)\big(q_\pi(S,A) - f_{\boldsymbol{w}}(S,A)\big)\nabla_{\boldsymbol{w}} f(S,A)$$

Compatible Approximation Property, cont'd

- Suppose we train $f_{\boldsymbol{w}}$ until convergence, i.e.,

$$\sum_s d_\pi(s) \sum_a \pi(a|s)\big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big)\nabla_{\boldsymbol{w}} f(s,a) = 0$$

- Suppose that $f_{\boldsymbol{w}}$ satisfies the following equation

$$\nabla_{\boldsymbol{w}} f(s,a) = \frac{1}{\pi(a|s)} \nabla_{\boldsymbol{\theta}} \pi(s,a)$$

  – A bit of a hacky assumption but makes the math work

  – Sutton/Tsitsiklis conjecture it may actually be the only case that guarantees convergence

- Makes the least-squares gradient

$$\sum_s d_\pi(s) \sum_a \big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big)\nabla_{\boldsymbol{\theta}} \pi(s,a) = 0$$

Compatible Approximation Property, cont'd

- Suppose that $f_{\boldsymbol{w}}$ satisfies the following equation

$$\nabla_{\boldsymbol{w}} f(s,a) = \frac{1}{\pi(a|s)} \nabla_{\boldsymbol{\theta}} \pi(s,a)$$

- Makes the least-squares gradient

$$\sum_{s} d_\pi(s) \sum_{a} \big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big) \nabla_{\boldsymbol{\theta}} \pi(s,a) = 0$$

- What does this look like?
  - Policy gradient, plus a term!
  - Moving the extra term to the right, we get

$$\sum_{s} d_\pi(s) \sum_{a} \nabla\pi(a|s) q_\pi(s,a) = \sum_{s} d_\pi(s) \sum_{a} \nabla\pi(a|s) f_{\boldsymbol{w}}(s,a)$$

- So finally,

$$\nabla v_\pi(s_0) = \sum_{s} d_\pi(s) \sum_{a} \nabla\pi(a|s) f_{\boldsymbol{w}}(s,a)$$

Compatible Approximation Property, cont'd

- Suppose that $f_{\boldsymbol{w}}$ satisfies the following equation

$$\nabla_{\boldsymbol{w}} f(s,a) = \frac{1}{\pi(a|s)} \nabla_{\boldsymbol{\theta}} \pi(s,a)$$

- Makes the least-squares gradient

$$\sum_s d_\pi(s) \sum_a \big(q_\pi(s,a) - f_{\boldsymbol{w}}(s,a)\big) \nabla_{\boldsymbol{\theta}} \pi(s,a) = 0$$

- So finally,

$$\nabla v_\pi(s_0) = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) f_{\boldsymbol{w}}(s,a)$$

- A "compatible" approximation points the gradient in the same direction as the true q function!

Compatible Approximation Example

- Suppose policy is the softmax policy as before

$$\pi(a|s;\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^T \boldsymbol{x}(s,a)}}{\sum_{a'} e^{\boldsymbol{\theta}^T \boldsymbol{x}(s,a')}}$$

- What is $\nabla\pi(a|s;\boldsymbol{\theta})$?
  - The derivative of the sigmoid is $\sigma'(x) = \sigma(x)\big(1-\sigma(x)\big)$
  - The derivative of the softmax is the same:

$$\nabla\pi(a|s;\boldsymbol{\theta}) = \boldsymbol{x}(s,a)\pi(a|s;\boldsymbol{\theta})\big(1-\pi(a|s;\boldsymbol{\theta})\big)$$

- Recall a compatible approximation is

$$\nabla_{\boldsymbol{w}} f(s,a) = \frac{1}{\pi(a|s)}\nabla_{\boldsymbol{\theta}}\pi(s,a) = \boldsymbol{x}(s,a)\big(1-\pi(a|s;\boldsymbol{\theta})\big)$$

- One option for f is a linear function:

$$f_{\boldsymbol{w}}(s,a) = \boldsymbol{w}^T\boldsymbol{x}(s,a) - \boldsymbol{w}^T\boldsymbol{x}(s,a)\pi(a|s;\boldsymbol{\theta})$$

Compatible Approximation Example, cont'd

- Recall a compatible approximation is

$$\nabla_{\boldsymbol{w}} f(s,a) = \frac{1}{\pi(a|s)} \nabla_{\boldsymbol{\theta}} \pi(s,a) = \boldsymbol{x}(s,a)\big(1 - \pi(a|s;\boldsymbol{\theta})\big)$$

- One option for f is a linear function:

$$f_{\boldsymbol{w}}(s,a) = \boldsymbol{w}^T \boldsymbol{x}(s,a) - \boldsymbol{w}^T \boldsymbol{x}(s,a)\pi(a|s;\boldsymbol{\theta})$$

- Effectively, we can only prove convergence for linear approximations
  - Linear approximation can be arbitrarily bad if the true q function is very non-linear
  - May need to trade convergence guarantees for better approximators and hope for the best
    - Will need to look at non-linear approximations (wink, wink)

Improving REINFORCE with baseline

- Recall the REINFORCE with baseline policy gradient

$$\nabla v_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=t}^{T} \nabla \log\big(\pi(A_k|S_k)\big)\,(G_t - \hat{v}(S_t))|S_t = s\right]$$

- Similar to MC methods, need to wait for returns
  - Both slow and high-variance
- How can we address it? (What did we do in the MC case?)
  - Use a TD-like approach!
- Instead of using only the current estimate $\hat{v}(S_t)$, use a bootstrapped estimate of G_t:

$$R_t + \gamma\hat{v}(S_{t+1}) - \hat{v}(S_t)$$

Actor-Critic Methods

- The TD-like policy gradient is now

$$\nabla v_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=t}^{T} \nabla \log\big(\pi(A_k|S_k)\big)\,(R_t + \gamma\hat{v}(S_{t+1}) - \hat{v}(S_t))|S_t = s\right]$$

- Just like in TD vs. MC, the above usually converges much faster
- This modification is called the actor-critic approach
  - The function approximating v is called the *critic*
    - Can also have a critic estimate the action-value q instead of v
  - The policy is called the *actor*

Actor-critic, cont'd

- Similar to Q-learning, actor-critic adds a bias
  - but reduces the variance
  - and is consistent (i.e., bias goes to 0 with more data)
- Typically, the critic is trained in parallel with the actor
  - How?

Training the critic

- Typically, the critic is trained in parallel with the actor
- Can train the critic to minimize squared error, as usual

$$\left(q(S_t, A_t) - Q^{\boldsymbol{w}}(S_t, A_t)\right)^2$$

  – where the critic $Q^{\boldsymbol{w}}$ is parameterized by weights $\boldsymbol{w}$
- Of course, we don't have the labels, so we bootstrap them
  – We use labels $y = R_{t+1} + \gamma Q^{\boldsymbol{w}}(S_{t+1}, A_{t+1})$
- Finally, minimize squared error using standard gradient descent

$$\boldsymbol{w}' = \boldsymbol{w} + \alpha_{\boldsymbol{w}} 2(R_{t+1} + \gamma Q^{\boldsymbol{w}}(S_{t+1}, A_{t+1}) - Q^{\boldsymbol{w}}(S_t, A_t))\nabla_{\boldsymbol{w}} Q^{\boldsymbol{w}}(S_t, A_t)$$

  – To calculate, need a tuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$

Actor-critic, cont'd

- In summary, suppose the critic $Q^{\boldsymbol{w}}$ is parameterized by weights $\boldsymbol{w}$ and the actor $\pi_{\boldsymbol{\theta}}$ is parameterized by $\boldsymbol{\theta}$
  - After observing a tuple $(S_t, A_t, R_t, S_{t+1}, A_{t+1})$:
    - $\delta_t = R_t + \gamma Q^{\boldsymbol{w}}(S_{t+1}, A_{t+1}) - Q^{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{w}' = \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta_t \nabla_{\boldsymbol{w}} Q^{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \delta_t \nabla \log\big(\pi_{\boldsymbol{\theta}}(A_t | S_t)\big)$
  - We have separate learning rates for the critic and actor, $\alpha_{\boldsymbol{w}}$ and $\alpha_{\boldsymbol{\theta}}$, respectively
  - Factor of 2 removed since it is incorporated into $\alpha_{\boldsymbol{w}}$
- Note that this is an on-policy approach (why?)
  - Need to wait for action A_{t+1} from current policy

On-Policy Actor-Critic Algorithm

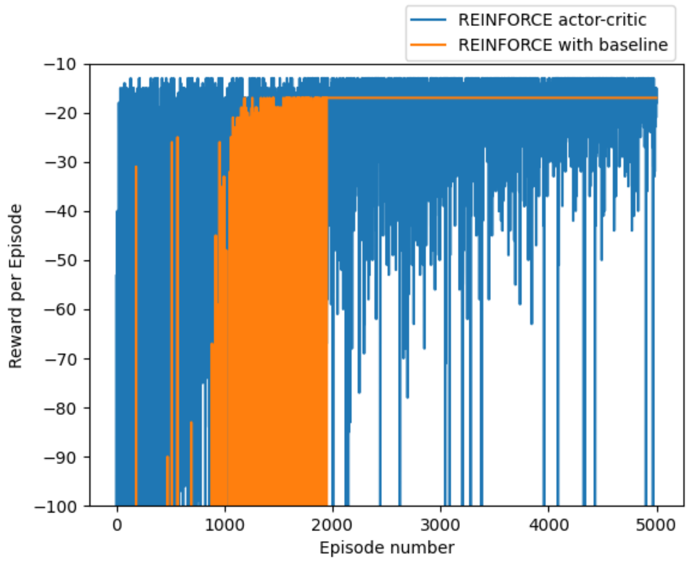
- Ideally, estimate the policy gradient over multiple episodes
  - It's an expectation over trajectories
  - One point is unbiased but has high variance
- As usual, cannot prove convergence for most cases

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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Comparison between REINFORCE algorithms

- Compare REINFORCE with baseline vs REINFORCE actor-critic on cliff environment
  - Use a simple Monte Carlo to estimate each state's value
    $$\hat{v}'(s) = \hat{v}(s) + \alpha\big(G - \hat{v}(s)\big)$$
    - use $\hat{v}$ both as baseline and as critic
  - Actor is a simple softmax policy
- REINFORCE with baseline converges very slowly
- Actor-critic has lower variance but it has a bias
  - Bias slowly converges to 0
  - Also finds optimal policy
  - Could be better with better critic



Extending Actor-Critic to Multi-Step Returns

- Can extend the actor-critic method to multi-step returns, similar to TD(n)
  - How?
  - Instead of collecting one-step reward R_t, collect n-step return $G_{t:t+n} = R_t + \cdots + \gamma^{n-1} R_{t+n-1}$
  - Use return in policy gradient theorem:
    - $\delta_t = G_{t:t+n} + \gamma^n Q^{\boldsymbol{w}}(S_{t+n}, A_{t+n}) - Q^{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{w}' = \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta_t \nabla_{\boldsymbol{w}} Q^{\boldsymbol{w}}(S_t, A_t)$
    - $\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \delta_t \nabla \log\left(\pi_{\boldsymbol{\theta}}(A_t | S_t)\right)$