Deep Reinforcement Learning

2022-23 Second Semester, M.Tech (AIML)

Session #13:

Policy Gradients - REINFORCE, Actor-Critic algorithms

Instructors :

1. Prof. S. P. Vimal (vimalsp@wilp.bits-pilani.ac.in),
2. Prof. Sangeetha Viswanathan (sangeetha.viswanathan@pilani.bits-pilani.ac.in)

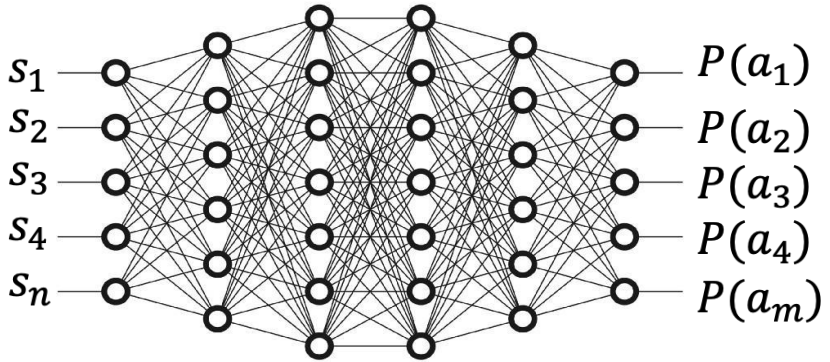
Agenda for the classes

- ➔ Introduction
- ➔ Policy gradients
- ➔ REINFORCE algorithm
- ➔ Actor-critic methods
- ➔ REINFORCE - example

Acknowledgements: Some of the slides were adopted *with permission* from the course CSCE-689 (Texas A&M University) by Prof. Guni Sharon

Notation

- The policy is a parametrized function: $\pi_\theta(a|s)$
  - For policy gradient we need a continues, differentiable policy… (Softmax activation can be useful for discreate action spaces)
  - $\pi(a|s)$ is assumed to be differentiable with respect to θ
  - E.g., a DNN were θ is the set of weights and biases
- $J(\theta)$ is a scalar policy performance measure (expected sum of discounted rewards) with respect to the policy params



Improving the policy

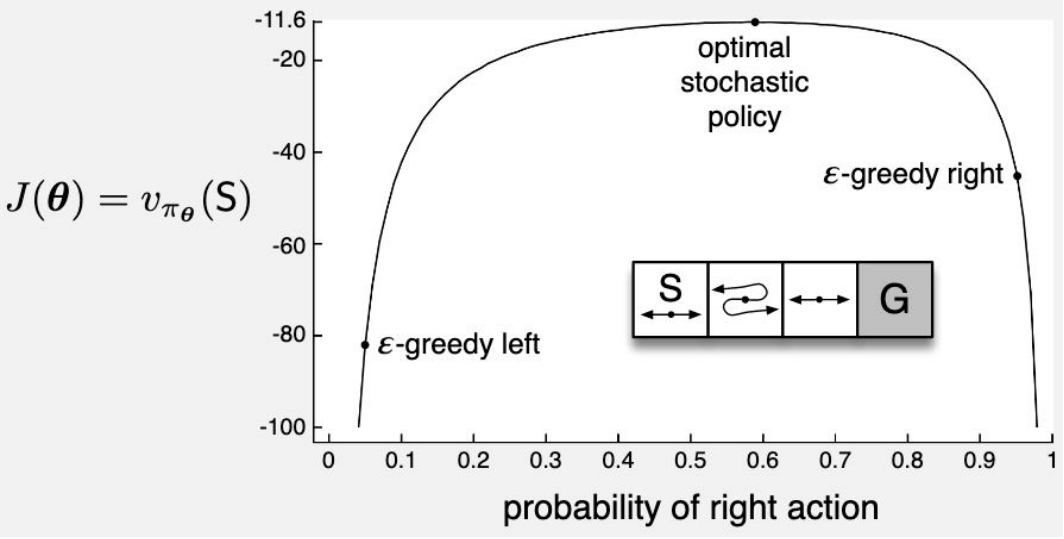
- SGD: $\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)}$
- Where $\widehat{\nabla J(\theta_t)}$ is a stochastic estimate, whose expectation approximates the gradient of the performance measure
- All methods that follow this general schema we call policy gradient methods
  - Might also learn an approximate value function for normalization (baseline)
- Methods that use approximations to both policy and value functions for computing the policy's gradient are called actor–critic methods (more on this later)

Advantages of PG

- The policy convergence over time as opposed to an epsilon-greedy value-based approach
- Naturally applies to continues action space as opposed to a Q learning approach
- In many domains the policy is a simpler function to approximate
  - Though this is not always the case
- Choice of policy parameterization is sometimes a good way of injecting prior knowledge
  - E.g., in phase assignment by a traffic controller
- Can converge on stochastic optimal policies, as opposed to value-based approaches
  - Useful in games with imperfect information where the optimal play is often to do two different things with specific probabilities, e.g., bluffing in Poker

Stochastic policy

- Reward = -1 per step
- $\mathcal{A} = \{left,\ right\}$
- Left goes left and right goes right except in the middle state where they are reversed
- States are identical from the policy's perspective
- $\pi^* = [0.41 \quad 0.59]$



Evaluate the gradient in performance

- $\widehat{\nabla_\theta J(\theta)} = ?$
- J depends on both the action selections and the distribution of states in which those selections are made
  - Both of these are affected by the policy parameter θ
- Seems impossible to solve without knowing the transition function (or the distribution of visited states)
  - $p(s'|s,a)$ is unknown in model free RL
- The PG theorem allows us to evaluate $\widehat{\nabla_\theta J(\theta)}$ without the need for $p(s'|s,a)$

Monte-Carlo Policy Gradient

- Sample-based gradient estimation
- $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s)$
- $\sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s) = \mathbb{E}_\pi[\sum_a q_\pi(S_t,a) \nabla_\theta \pi(a|S_t)]$
- We can now use this gradient estimation for PG steps
  - $\theta_{t+1} = \theta_t + \alpha \sum_a \widehat{q_\pi}(S_t,a) \nabla_\theta \pi(a|S_t;\theta)$
  - Requires an approximation to q_π, denoted $\hat{q}$, for every possible action
  - Can we avoid this approximation?

REINFORCE [Williams, 1992]

1. $\theta_{t+1} = \theta_t + \alpha \sum_a \widehat{q_\pi}(S_t, a) \nabla_\theta \pi(a|S_t)$
   - Can we avoid this approximation? YES!
2. $\nabla J(\theta) \propto \mathbb{E}_\pi[\sum_a q_\pi(S_t, a) \nabla_\theta \pi(a|S_t)]$
3. $= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t;\theta) q_\pi(S_t, a) \frac{\nabla_\theta \pi(a|S_t)}{\pi(a|S_t)} \right]$ (multiplied and divided by the same number)
4. $= \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla_\theta \pi(A_t|S_t)}{\pi(A_t|S_t)} \right]$ ($\sum_x \Pr(x) f(x) = \mathbb{E}_{x\sim \Pr(x)}[f(x)]$)
5. $= \mathbb{E}_\pi \left[G_t \frac{\nabla_\theta \pi(A_t|S_t)}{\pi(A_t|S_t)} \right]$ ($\mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t)$)

- We can now use this gradient estimation for PG steps
  - $\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_\theta \pi(A_t|S_t)}{\pi(A_t|S_t)}$

REINFORCE [Williams, 1992]

•• $\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_\theta \pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta)}$

- Each increment is proportional to the product of a return G_t and a vector
  - The gradient of the probability of taking the action actually taken over the (current) probability of taking that action
  - The vector is the direction in parameter space that most increases the probability of repeating the action A_t on future visits to state S_t
- The update increases the parameter vector (*) proportional to the return, and (**) inversely proportional to the action probability
  - (*) makes sense because it causes the parameter to move most in the directions that favor actions that yield the highest return
  - (**) makes sense because otherwise actions that are selected frequently are at an advantage (the updates will be more often in their direction) and might win out even if they do not yield the highest return

REINFORCE [Williams, 1992]

- $\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_\theta \pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta)}$
- REINFORCE uses G_t: the complete return from time t until the end of the episode
- In this sense REINFORCE is a Monte Carlo algorithm and is well defined only for the episodic case with all updates made in retrospect after the episode is completed

REINFORCE [Williams, 1992]

* In the literature you might encounter "grad log pi" instead of "grad ln pi". That's not a problem since: $\nabla \ln(f(x)) \propto \nabla \log(f(x))$

Specifically: $\nabla \ln(f(x)) = \nabla \log_a(f(x)) \ln(a)$

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

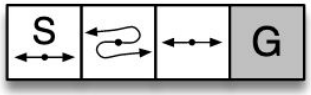
Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step t
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

How did we get here from $\frac{\nabla_\theta \pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta)}$?

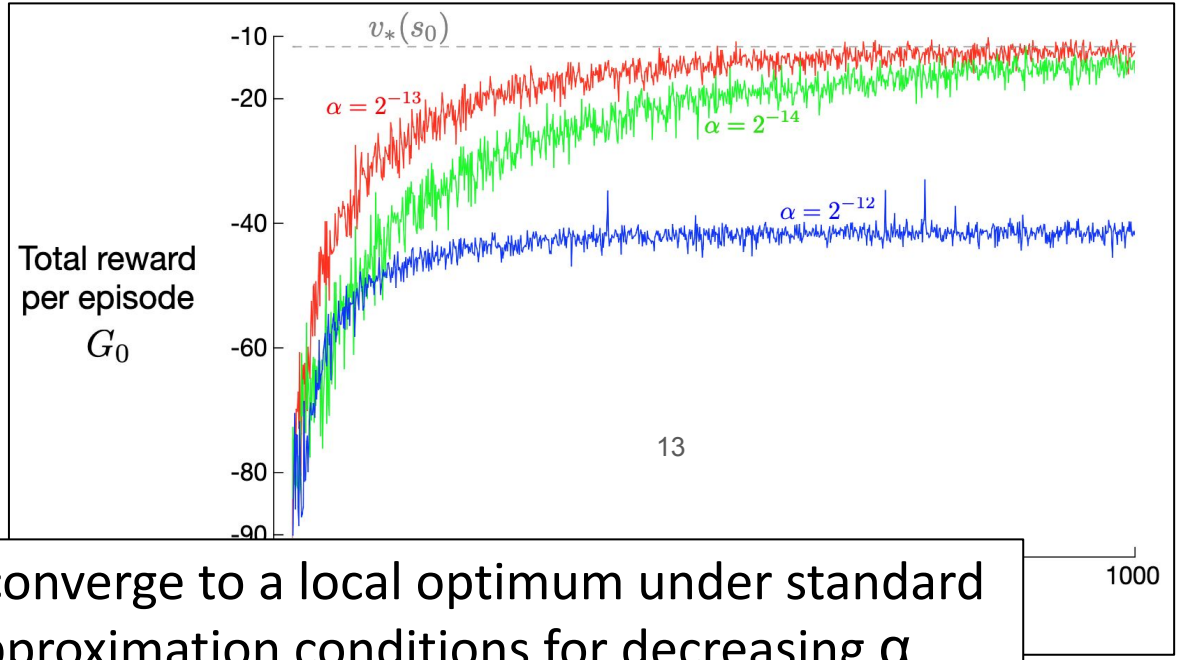
$$\nabla \ln(f(x)) = \frac{\nabla f(x)}{f(x)}$$

REINFORCE [Williams, 1992]

- The crazy corridor domain



- With the right step size, the total reward per episode approaches the optimal value of the start state



Guaranteed to converge to a local optimum under standard stochastic approximation conditions for decreasing α

Calibrating REINFORCE

- Imagine a domain with two possible episode trajectories (rollouts)
  - $\tau_1 = \{S_0, A_0, R_1, S_1, \ldots, A_{T-1}, R_T, S_T\}$ with $G = 1001$
  - $\tau_2 = \{S'_0, A'_0, R'_1, S'_1, \ldots, A'_{T-1}, R'_T, S'_T\}$ with $G' = 1000$
  - Further assume that $R_{i<T} = R'_i = 0$ and $R_T = G, R'_T = G'$
- Following REINFORCE: $\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- How will the policy be updated after sampling each of the trajectories?
  - $\tau_1 ++, \tau_2 ++$
- How should the policy be updated after sampling each of the trajectories?
  - $\tau_1 +, \tau_2 -$
- How can we address this gap?

PG with baseline

- Normalize the returns with a baseline!
- $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \big(q_\pi(s,a) - \mathrm{b(s)}\big) \nabla \pi(a|s)$
- Are we allowed to do that???
- Can we subtract $\sum_a b(s) \nabla \pi(a|s)$ from $\nabla J(\theta)$?
- Yes! As long as $b(s)$ isn't a function of a
- $\sum_a b(s) \nabla \pi(a|s) = b(s) \nabla \sum_a \pi(a|s) = b(s) \nabla 1 = 0$ (actions' probabilities sum to 1)
- REINFORCE with baseline: $\theta_{t+1} = \theta_t + \alpha \big(G_t - b(S_t)\big) \nabla_\theta \ln \pi_\theta(A_t|S_t)$

PG with baseline

- In the bandit algorithms the baseline was the average of the rewards seen so far
- For MDPs the baseline should be different for each states
  - In some states all actions have high (q) values, and we need a high baseline to differentiate the higher valued actions from the less highly valued ones
  - In other states all actions will have low values and a low baseline is appropriate
- What would be the equivalent of average reward (bandits) for a given state (MDP) ?
  - $v_\pi(s)$

PG with baseline

- $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \big(q_\pi(s,a) - v_\pi(s)\big) \nabla \pi(a|s)$
- Actions that improve on the current policy are encouraged
  - $q_\pi(s,a) - v_\pi(s) > 0$
- Actions that damage the current policy are discouraged
  - $q_\pi(s,a) - v_\pi(s) < 0$
- Also known as the **advantage** of action a in state s
- How can we learn $v_\pi(s)$?
- Another function approximator $\hat{v}(s;w)$
  - On top of the policy

REINFORCE with baseline

REINFORCE with Baseline (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

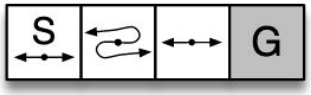
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G_t \leftarrow$ return from step t
    $\delta \leftarrow G_t - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \gamma^t \delta \nabla_{\mathbf{w}} \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t,\boldsymbol{\theta})$

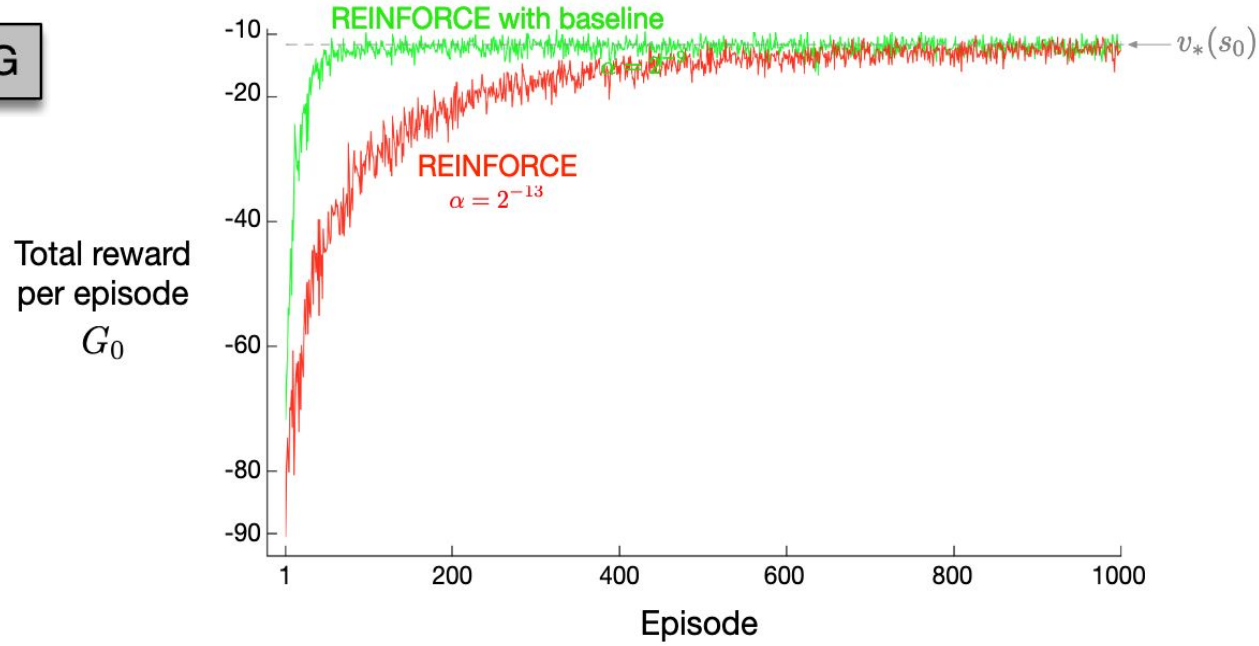
Each approximator has its unique learning rate

REINFORCE with baseline

- The crazy corridor domain

- $\alpha^{\theta} = 2^{-9}$
- $\alpha^{w} = 2^{-6}$





Policy Gradient

- $\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)}$
- PG theorem: $\widehat{\nabla J(\theta_t)} \propto \big(q_\pi(S_t, A_t) - b(S_t)\big)\nabla_\theta \log \pi(A_t|S_t;\theta)$
  - REINFORCE+baseline: $\theta_{t+1} = \theta_t + \alpha\big(G_t - b(S_t)\big)\nabla_\theta \log \pi(A_t|S_t;\theta)$
- **Pros**: approximating the optimal policy directly is more accurate and faster to converge in many domains when compared to value-based approaches
- **Cons**: using G_t as an estimator for $q_\pi(S_t, A_t)$ is noisy (high variance) though unbiased
  - Unstable learning

Add a critic

- $\widehat{\nabla J(\theta_t)} \propto \left(q_\pi(S_t, A_t) - b(S_t)\right) \frac{\nabla_\theta \pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta)}$
- Define a new estimator: $\hat{q}_\pi(s, a; \theta) \approx q_\pi(s, a)$
  - To be used instead of G_t in REINFORCE
- How should we train $\hat{q}_\pi(s, a; \theta)$?
  - Monte-Carlo updates
    - High variance samples
    - Requires a full episode
  - Bootstrapping e.g., Q-learning
    - Lower variance (though introduces bias)

Critic's duties

1. Approximate state or action or advantage ($q(s,a) - v(s)$) values
2. Trained via bootstrapping, criticizes the action chosen by the actor

   adjusting the actor's (policy) gradient

- Is REINFORCE (+ state value baseline) an actor-critic framework?
  - $\theta_{t+1} = \theta_t + \alpha\big(G_t - \hat{v}_\pi(S_t)\big)\nabla_\theta \ln \pi(A_t|S_t;\theta)$
  - NO! the state-value function is used only as a baseline, not as a critic. Moreover, it doesn't utilize bootstrapping (uses high-variance MC updates)
- The bias introduced through bootstrapping is often worthwhile because it reduces variance and accelerates learning

Benefits from a critic

- REINFORCE with baseline is unbiased* and will converge asymptotically to a local optimum
  - * With a linear state value approximator, and when *b* is not a function of *a*
  - Like all Monte-Carlo methods it tends to learn slowly (produce estimates of high variance)
  - Not suitable for online or for continuing problems
- Temporal-difference methods can eliminate these inconveniences
- In order to gain the TD advantages in the case of policy gradient methods we use actor–critic methods

Actor+critic

- Actor-critic algorithms are a derivative of policy iteration, which alternates between policy evaluation—computing the value function for a policy—and policy improvement—using the value function to obtain a better policy
- In large-scale reinforcement learning problems, it is typically impractical to run either of these steps to convergence, and instead the value function and policy are optimized jointly
- The policy is referred to as the actor, and the value function as the critic

Advantage function

- Eventually we would like to shift the policy towards actions that result in higher return
- what is the benefit from taking action a at state s while following policy π?
  - $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$
- This resembles PG with baseline but not the same as q_π is approximated:
  - $\widehat{\nabla J(\theta_t)} = A_\pi(s_t,a_t)\nabla_\theta \ln \pi(a_t|s_t;\theta)$
- Actions that improve on the current policy are encouraged
  - $A_\pi(s,a) > 0$
- Actions that damage the current policy are discouraged
  - $A_\pi(s,a) < 0$

One-step actor-critic

- $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$
  - Does that mean that approximating the advantage function requires two function approximators ($\hat{q}$ and $\hat{v}$) ?
  - No since q values can be derived from state values + one step transition
  - $\hat{q}_\pi(s_t,a_t) - \hat{v}_\pi(s_t;w) = \widehat{\mathbb{E}}[r_{t+1} + \gamma\hat{v}(s_{t+1};w)] - \hat{v}(s_t;w)$
- $\theta_{t+1} = \theta_t + \alpha\big(r_{t+1} + \gamma\hat{v}(s_{t+1};w) - \hat{v}(s_t;w)\big)\nabla_\theta \ln\pi(a_t|s_t;\theta)$

δ – TD error

- One-step Actor-Critic is a fully online, incremental algorithm, with states, actions, and rewards processed as they occur and then never revisited

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Keep track of accumulated discount

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Follow the current policy

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\ \delta\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\,\delta\nabla_{\boldsymbol{\theta}}\ln\pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Compute the TD error

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Update the critic without the accumulated discount. (The discount factor is included in the TD error)

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Update the actor with discounting. Early actions matter more.

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Update accumulated discount and progress to the next state

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

In practice: training the network at every step on a single observation is inefficient (slow and correlated)

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Instead: store all state approx values, log probabilities, and rewards along the episode. Train once at the end of the episode.

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value 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 $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

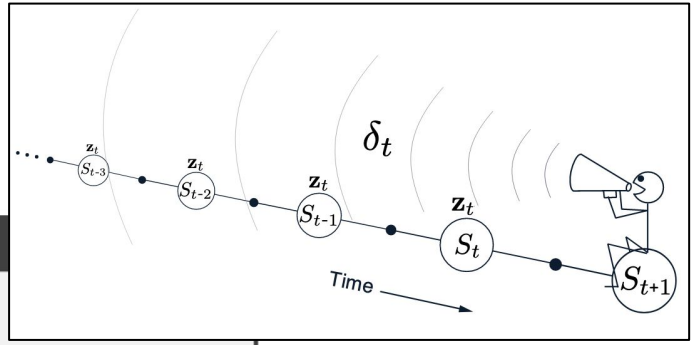
Instead: store all state approx. values, log probabilities, and rewards along the episode. Train once at the end of the episode.

```
values = torch.FloatTensor(values)
Qvals = torch.FloatTensor(Qvals)
log_probs = torch.stack(log_probs)
advantage = Qvals - values
```

Advantage Actor-Critic (A2C)

- Initialize the actor π_θ and the critic $\hat{V}_w$
- For each episode:
  - Init empty episode memory
  - $s = init\ env$
  - For each time step:
    - $a \sim \pi(s)$
    - $s', r_t \sim env(s, a)$
    - Store (s, a, r) in episode memory
    - $s = s'$
  - Reset $\mathrm{d}\theta = 0, dw = 0$
  - Backwards iteration (i from length to 0) over the episode
    - Compute advantage $\delta_t = r_i + \gamma \hat{V}_w(s_{i+1}) - \hat{V}_w(s_i)$
    - Accumulate the policy gradient using the critic: $d\theta \leftarrow d\theta + \delta \nabla_\theta \log \pi_\theta(s_i, a_i)$
    - Accumulate the critic gradient: $dw \leftarrow dw + \delta \nabla_w \hat{V}_w(s_i)$
    - Update the actor and the critic with the accumulated gradients using gradient descent

Add eligibility traces



Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Gradient eligibility per tunable parameter for both the actor and the critic approximators

What did we learn?

- In many domains it's more efficient to learn the policy directly
  - Instead of deriving one from state or action values
- Applicable for continuous action spaces and stochastic policies
- Approximate the change to the policy that results in the highest increase in the expected return: $\widehat{\nabla_\theta J(\theta)}$
- Such approximation is made possible when by the policy gradient theorem
- Should we reinforce polices that result in high return?
  - Not always, a different policy might yield higher return
  - We need to use a baseline to determine if a policy is **relatively** good

What did we learn?

- REINFORCE:
  - $\widehat{\nabla J(\theta_t)} = G_t \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- Q Actor-Critic:
  - $\widehat{\nabla J(\theta_t)} = \hat{q}(S_t, A_t; w) \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- REINFORCE + baseline:
  - $\widehat{\nabla J(\theta_t)} = (G_t - \hat{v}(S_t;w)) \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- Advantage Actor-Critic:
  - $\widehat{\nabla J(\theta_t)} = (R_{t+1} + \gamma \hat{v}(S_{t+1};w) - \hat{v}(S_t;w)) \nabla_\theta \ln \pi(A_t|S_t;\theta)$

Required Readings

1. Chapter-6 of Introduction to Reinforcement Learning,2nd Ed., Sutton & Barto

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Thank you