Policy Gradient Methods

CMPS 4660/6660: Reinforcement Learning

Acknowledgement: slides adapted from David Silver's RL course

Agenda

- Introduction
- Finite Difference Policy Gradient
- Monte-Carlo Policy Gradient
- Actor-Critic Policy Gradient

Policy-Based Reinforcement Learning

- So far we have approximated the value or action-value function using parameters,

$$\hat{v}_w(s) \approx v_\pi(s)$$

or $\hat{q}_w(s,a) \approx q_\pi(s,a)$

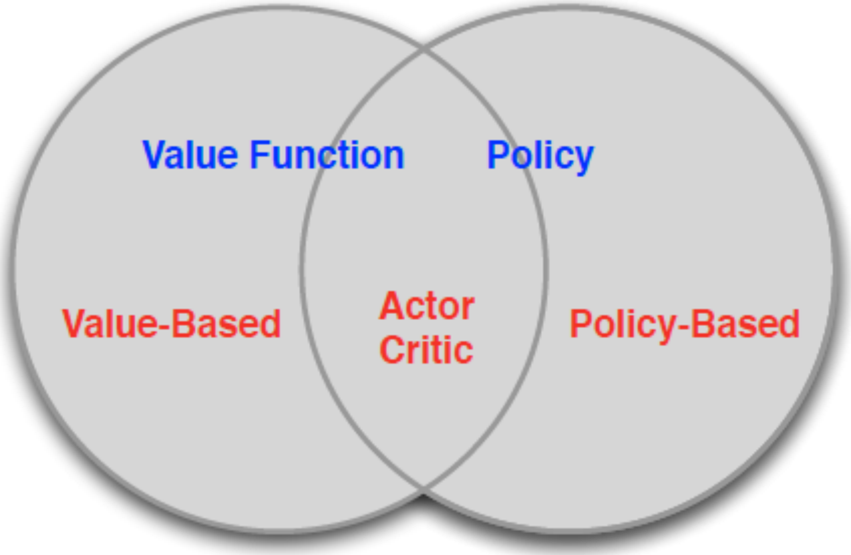
- A policy was generated directly from the value function
  - E.g., using ϵ-greedy
- Alternatively, we can directly parameterize the policy

$$\pi_\theta(a|s) = \Pr(A_t = a|S_t = s, \theta)$$

- We will focus again on model-free reinforcement learning

Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy

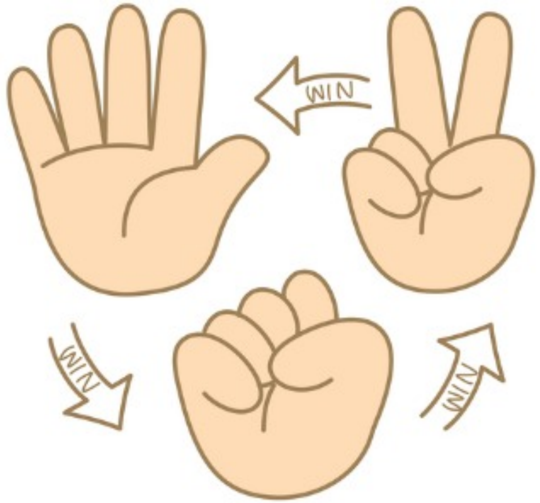


Advantages of Policy-Based RL

- Advantages
  - Better convergence properties
  - Effective in high-dimensional or continuous action spaces
  - Can learn stochastic policies
  - A good way of injecting prior knowledge about the policy into RL
- Disadvantages:
  - Typically converge to a local rather than global optimum
  - Evaluating a policy is typically inefficient and has high variance

Example: Rock-Paper-Scissors

- Two-player game of rock-paper-scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Consider policies for iterated rock-paper-scissors
  - A deterministic policy is easily exploited
  - A uniform random policy is optimal (i.e. Nash equilibrium)



Example: Aliased Gridworld (1)

- The agent cannot differentiate the grey states
- Consider features of the following form (for all N, E, S, W)

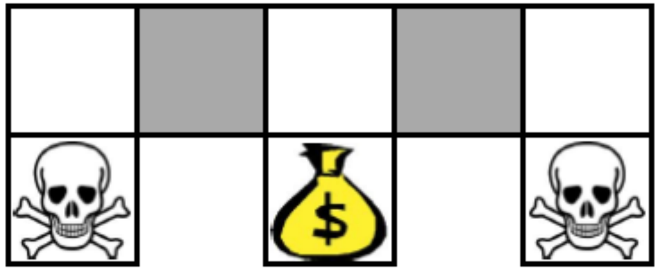
$$\mathrm{x}(s, a) = \mathbf{1}(\text{wall to N and S}, a\text{=E})$$

- Compare value-based RL, using an approximate value function

$$\hat{q}_w(s, a) = f(\mathrm{x}(s, a), w)$$

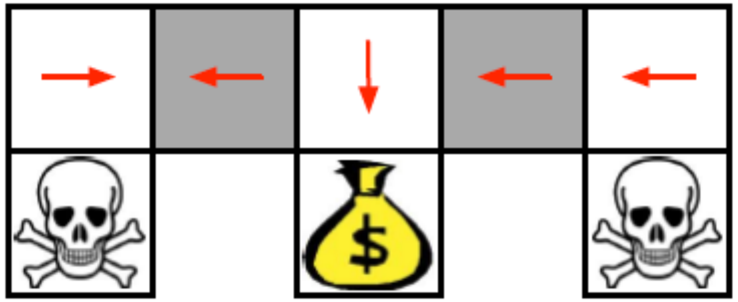
- To policy-based RL, using a parametrized policy

$$\pi_\theta(a|s) = g(\mathrm{x}(s, a), \theta)$$



Example: Aliased Gridworld (2)

- Under aliasing, an optimal deterministic policy will either
  - move W in both grey states (shown by red arrows)
  - move E in both grey states
- Either way, it can get stuck and never reach the money
- Value-based RL learns a near-deterministic policy
  - e.g. greedy or ϵ-greedy
- So it will traverse the corridor for a long time



Example: Aliased Gridworld (3)

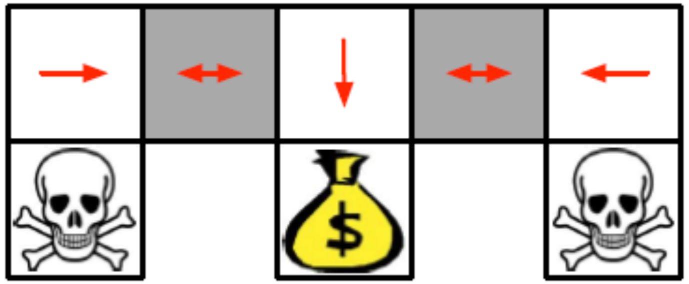
- An optimal stochastic policy will randomly move E or W in grey states

$$\pi_\theta(\text{move E} \mid \text{wall to N and S}) = 0.5$$

$$\pi_\theta(\text{move W} \mid \text{wall to N and S}) = 0.5$$

- It will reach the goal state in a few steps with high probability

- Policy-based RL can learn the optimal stochastic policy



Agenda

- Introduction
- Finite Difference Policy Gradient
- Monte-Carlo Policy Gradient
- Actor-Critic Policy Gradient

Policy Objective Functions

- Goal: given policy $\pi_\theta(a|s)$ with parameters θ, find best θ
- But how do we measure the quality of a policy π_θ?
- In episodic environments we can use the start value

$$J_1(\theta) = v_{\pi_\theta}(s_0)$$

- In continuing environments we can use the average value

$$J_{avV}(\theta) = \sum_{s \in \mathcal{S}} \mu_{\pi_\theta}(s)\, v_{\pi_\theta}(s)$$

  - $\mu_{\pi_\theta}(\cdot)$ is the stationary distribution of states under π_θ

- Or the average-reward per time-step

$$J_{avR}(\theta) = \sum_{s} \mu_{\pi_\theta}(s) \sum_{a} \pi_\theta(a|s)\, R(s,a)$$

- For continuous state and actions spaces, replace summations by integrals and interpret $\pi_\theta(\cdot\,|s)$ as a density function

Policy Optimization

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

Policy Gradient

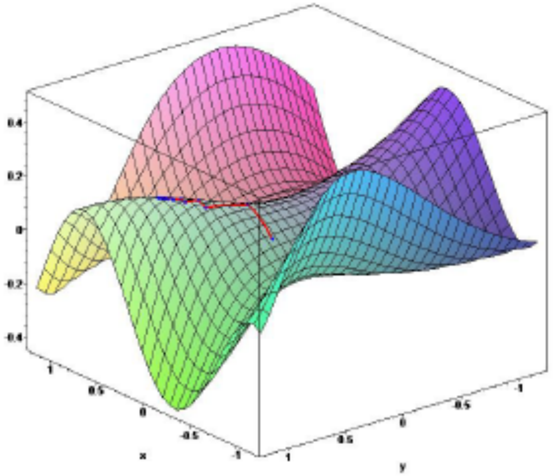
- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters

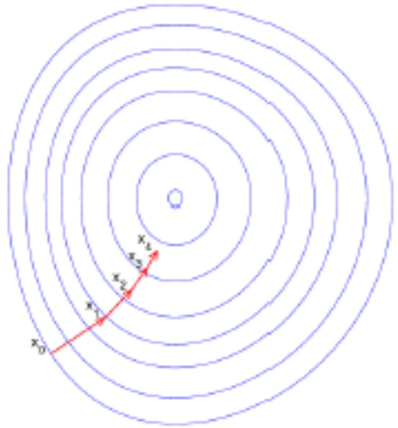
$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t) \qquad \Delta\theta = \alpha \nabla_\theta J(\theta)$$

- where $\nabla_\theta J(\theta)$ is the policy gradient

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

- and α is a step-size parameter





Computing Gradients By Finite Differences

- To evaluate policy gradient of $\pi_\theta(a|s)$
- For each dimension $k \in \{1, \ldots, d'\}$
  - Estimate kth partial derivative of objective function w.r.t. θ
  - By perturbing θ by small amount ϵ in kth dimension

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

where u_k is unit vector with 1 in kth dimension and 0 otherwise

- Uses d' evaluations to compute policy gradient in d' dimensions
- Simple, noisy, inefficient - but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable

Agenda

- Introduction
- Finite Difference Policy Gradient
- Monte-Carlo Policy Gradient
- Actor-Critic Policy Gradient

Score Function

- We now compute the policy gradient *analytically*
- Assume policy π_θ is differentiable whenever it is non-zero
- and we know the gradient $\nabla_\theta \pi_\theta(a|s)$
- Likelihood ratios exploit the following identity

$$\begin{aligned}\nabla_\theta \pi_\theta(a|s) &= \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \\ &= \pi_\theta(a|s)\, \nabla_\theta \log \pi_\theta(a|s)\end{aligned}$$

- The score function is $\nabla_\theta \log \pi_\theta(a|s)$

One-Step MDPs

- Consider a simple class of one-step MDPs
  - Starting in state $s \sim \mu(s)$
  - Terminating after one time-step with reward $R(s, a)$
- Use likelihood ratios to compute the policy gradient

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[R(s, a)] \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) R(s, a) \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s) R(s, a) \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} R(s, a) \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) R(s, a)
\end{aligned}
$$

Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward R with long-term value $q_\pi(s, a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective (with different constants)

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}$, the policy gradient is

$$\nabla_\theta J_1(\theta) = \sum_{s \in \mathcal{S}} d_\pi(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s)\, q_\pi(s, a)$$

where $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s | s_0, \pi)$

Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward R with long-term value $q_\pi(s,a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective (with different constants)

Theorem

For any differentiable policy $\pi_{\boldsymbol{\theta}}$, the policy gradient is

$$\nabla_\theta J_1(\theta) = \sum_{s\in\mathcal{S}} d_\pi(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a|s)\, q_\pi(s,a)$$

where $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s | s_0, \pi)$

Thus for $\gamma = 1$, $\nabla_\theta J_1(\theta) \propto \sum_{s\in\mathcal{S}} \mu_\pi(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\, q_\pi(s,a)$

$$= \mathbb{E}_{s\sim\mu_\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, q_\pi(s,a)]$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, q_\pi(s,a)]$$

Proof of Policy Gradient Theorem for J_1 and $\gamma = 1$

$$\nabla_\theta v_\pi(s) = \nabla_\theta [\textstyle\sum_a \pi_\theta(a|s) q_\pi(s,a)]$$

$$= \textstyle\sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a) + \pi_\theta(a|s) \nabla_\theta q_\pi(s,a)]$$ (product rule of calculus)

$$= \textstyle\sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a) + \pi_\theta(a|s) \nabla_\theta \sum_{s',r} p(s',r|s,a)\,(r + v_\pi(s')]$$

$$= \textstyle\sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a) + \pi_\theta(a|s) \sum_{s'} p(s'|s,a)\, \nabla_\theta v_\pi(s')]$$

$$= \textstyle\sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a) + \pi_\theta(a|s) \sum_{s'} p(s'|s,a)$$ (unrolling)

$$\textstyle\sum_{a'} [\nabla_\theta \pi_\theta(a'|s') q_\pi(s',a') + \pi_\theta(a'|s') \sum_{s''} p(s''|s',a')\, \nabla_\theta v_\pi(s'')]$$

$$= \textstyle\sum_{x \in \mathcal{S}} \sum_{t=0}^{\infty} \Pr(s_t = x | s_0 = s, \pi_\theta) \sum_a [\nabla_\theta \pi_\theta(a|x) q_\pi(x,a)]$$

Proof of Policy Gradient Theorem for J_1 and $\gamma = 1$ (cont.)

$$\nabla_\theta J_1(\theta) = \nabla_\theta v_\pi(s_0)$$

$$= \sum_s \sum_{t=0}^{\infty} \Pr(s_t = s | s_0, \pi_\theta) \sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a)]$$

$$= \sum_s d_\pi(s) \sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a)]$$

$$= \sum_{s'} d_\pi(s') \sum_s \frac{d_\pi(s)}{\sum_{s'} d_\pi(s')} \sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a)]$$

$$= \sum_{s'} d_\pi(s') \sum_s \mu_\pi(s) \sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a)]$$

$$\propto \sum_s \mu_\pi(s) \sum_a [\nabla_\theta \pi_\theta(a|s) q_\pi(s,a)]$$

- $\sum_{s'} d_\pi(s') = \sum_{s'} \sum_{t=0}^{\infty} \Pr(s_t = s' | s_0, \pi_\theta)$: the average length of an episode

Softmax Policy

- We will use a softmax policy as a running example
- Weight actions using linear combination of features $\phi(s,a)^\top\theta$
- Probability of action is proportional to exponentiated weight

$$\pi_\theta(a|s) = \frac{e^{\phi(s,a)^\top\theta}}{\sum_b e^{\phi(s,b)^\top\theta}}$$

- The score function is

$$\nabla_\theta \log \pi_\theta(a|s) = \phi(s,a) - \sum_b \pi_\theta(b|s)\phi(s,b)$$

Gaussian Policy

- In continuous action spaces, a Gaussian policy is natural
- Mean is a linear combination of state features $\nu(s) = \phi(s)^\top \theta$
- Variance may be fixed σ^2, or can also parametrized
- Policy is Gaussian, $a \sim N(\nu(s), \sigma^2)$
- The score function is

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{\big(a - \nu(s)\big)\phi(s)}{\sigma^2}$$

where $\pi_\theta(\cdot\,|s)$ is interpreted as a density function

Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return G_t as an unbiased sample of $q_\pi(s_t, a_t)$

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t$$

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

Agenda

- Introduction
- Finite Difference Policy Gradient
- Monte-Carlo Policy Gradient
- Actor-Critic Policy Gradient

Reducing Variance Using a Critic

- Monte-Carlo policy gradient still has high variance
- We use a critic to estimate the action-value function

$$\hat{q}_w(s,a) \approx q_\pi(s,a)$$

- Actor-critic algorithms maintain two sets of parameters
  - Critic Updates action-value function parameters w
  - Actor Updates policy parameters, in direction suggested by critic
- Actor-critic algorithms follow an approximate policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, \hat{q}_w(s,a)]$$

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t)\, \hat{q}_w(s_t,a_t)$$

Estimating the Action-Value Function

- The critic is solving a familiar problem: policy evaluation
- How good is policy for current parameters w?
- This problem was explored before, e.g.
  - Monte-Carlo policy evaluation
  - Temporal-Difference learning
  - TD(λ)
- Could also use e.g. least-squares policy evaluation

Action-Value Actor-Critic

- Simple actor-critic algorithm based on action-value critic

function QAC
  Initialize θ, w, S
  Sample $A \sim \pi_\theta(\cdot\,|S)$
  for each step do
    Take action A, observe R, S'
    Sample $A' \sim \pi_\theta(\cdot\,|S')$
    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_\theta(A|S)\, \hat{q}_w(S, A)$
    $\delta \leftarrow R + \gamma \hat{q}_w(S', A') - \hat{q}_w(S, A)$
    $w \leftarrow w + \alpha_w \delta \nabla_w\, \hat{q}_w(S, A)$
    $A \leftarrow A', S \leftarrow S'$
  end for
end function

On-policy, similar to Sarsa(0)

Bias in Actor-Critic Algorithms

- Approximating the policy gradient introduces bias
- A biased policy gradient may not find the right solution
  - e.g. if $\hat{q}_w(s, a)$ uses aliased features, can we solve gridworld example?
- Luckily, if we choose value function approximation carefully
- Then we can avoid introducing any bias
- i.e. We can still follow the *exact* policy gradient

Compatible Function Approximation

Theorem (Compatible Function Approximation Theorem)

If the following two conditions are satisfied:

- Value function approximator is compatible to the policy

$$\nabla_w \hat{q}_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^\top \text{ or } \hat{q}_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^\top w$$

i.e., value function approximators are linear in "features" of the stochastic policy

Compatible Function Approximation

Theorem (Compatible Function Approximation Theorem)

If the following two conditions are satisfied:

- Value function approximator is compatible to the policy

$$\nabla_w \hat{q}_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^\top \text{ or } \hat{q}_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^\top w$$

  i.e., value function approximators are linear in “features” of the stochastic policy

- Value function parameters w minimize the mean-squared error

$$\epsilon^2(w) = \mathbb{E}_{\pi_\theta}\left[\left(q_{\boldsymbol{\pi}}(s,a) - \hat{q}_w(s,a)\right)^2\right]$$

Then the policy gradient is exact, $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, \hat{q}_w(s,a)]$

Proof of Compatible Function Approximation Theorem

- If w is chosen to minimize mean-squared error, gradient of ϵ w.r.t. w must be zero

$$\nabla_w \epsilon^2(w) = 0$$

$$\mathbb{E}_{\pi_\theta}[(q_\pi(s,a) - \hat{q}_w(s,a))\nabla_w \hat{q}_w(s,a)] = 0$$

$$\mathbb{E}_{\pi_\theta}[(q_\pi(s,a) - \hat{q}_w(s,a))\nabla_\theta \log \pi_\theta(a|s)] = 0$$

$$\mathbb{E}_{\pi_\theta}[q_\pi(s,a)\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_{\pi_\theta}[\hat{q}_w(s,a)\nabla_\theta \log \pi_\theta(a|s)]$$

So $\hat{q}_w(s,a)$ can be substituted directly into the policy gradient,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, \hat{q}_w(s,a)]$$

Reducing Variance Using a Baseline

- We subtract a baseline function $B(s)$ from the policy gradient
- This can reduce variance, without changing expectation

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\,[\hat{q}_w(s,a) - B(s)]\right] = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\,\hat{q}_w(s,a)]$$

because $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\,B(s)] = \sum_{s\in\mathcal{S}} \mu(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta\,(a|s) B(s)$

$$= \sum_{s\in\mathcal{S}} \mu(s) B(s) \nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(a|s)$$

$$= 0$$

Reducing Variance Using a Baseline (cont.)

- A good baseline is the state value function $B(s) \;=\; v_\pi(s)$
- So we can rewrite the policy gradient using the advantage function

$$A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, A_\pi(s,a)]$$

- The advantage function can significantly reduce variance of policy gradient
  - E.g., when all actions have high values, a high baseline can be used to differentiate the actions

Estimating the Advantage Function (1)

- So the critic should really estimate the advantage function
- For example, by estimating both $v_\pi(s)$ and $q_\pi(s,a)$
- Using two function approximators and two parameter vectors,

$$\hat{v}_w(s) \approx v_\pi(s)$$

$$\hat{q}_{w'}(s,a) \approx q_\pi(s,a)$$

$$\hat{A}(s,a) = \hat{q}_{w'}(s,a) - \hat{v}_w(s)$$

And updating both value functions by e.g. TD learning

Estimating the Advantage Function (2)

- For the true value function $v_\pi(s)$, the TD error

$$\delta_\pi = R(s,a) + \gamma v_\pi(s') - v_\pi(s)$$

- is an unbiased estimate of the advantage function

$$\begin{aligned}\mathbb{E}_\pi(\delta_\pi \mid s,a) &= \mathbb{E}_\pi[R(s,a) + \gamma v_\pi(s')|s,a] - v_\pi(s) \\ &= q_\pi(s,a) - v_\pi(s) \\ &= A_\pi(s,a)\end{aligned}$$

- So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\delta_{\pi_\theta}\right]$$

- In practice we can use an approximate TD error $\delta_w = R(s,a) + \gamma \hat{v}_w(s') - \hat{v}_w(s)$
- This approach only requires one set of critic parameters *w*

TD Actor-Critic (episodic)

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

TD(λ) Actor-Critic (episodic)

Actor–Critic with Eligibility Traces (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: 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 $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop 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)
    $I \leftarrow 1$
    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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla \ln \pi(A|S,\boldsymbol{\theta})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Summary of Policy Gradient Algorithms

- The policy gradient has many equivalent forms

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) G_t] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \hat{q}_w(s,a)] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \hat{A}(s,a)] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \delta_w] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}[\mathbf{z}^\theta \delta_w] && \text{TD}(\lambda) \text{ Actor-Critic}
\end{aligned}$$

- Each leads a stochastic gradient ascent algorithm
- Critic uses policy evaluation (e.g. MC or TD learning) to estimate $\hat{q}_w(s,a)$, $\hat{A}(s,a)$, or δ_w

Off-Policy Actor-Critic

- π: target policy, β: behavior policy
- Policy objective function

$$J(\theta) = \sum_s \mu_\beta(s)\, v_{\pi_\theta}(s)$$
$$= \sum_s \mu_\beta(s) \sum_a \pi_\theta(a|s)\, q_\pi(s,a)$$

- Off-policy policy-gradient

$$\nabla_\theta J(\theta) \approx \sum_s \mu_\beta(s) \sum_a \nabla_\theta \pi_\theta(a|s)\, q_\pi(s,a)$$
$$= \sum_s \mu_\beta(s) \sum_a \beta(a|s) \frac{\pi_\theta(a|s)}{\beta(a|s)} \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} q_\pi(s,a)$$
$$= \mathbb{E}_\beta \left[\frac{\pi_\theta(a|s)}{\beta(a|s)} \nabla_\theta \log \pi_\theta(a|s)\, q_\pi(s,a) \right]$$

- Both the actor and the critic use an importance sampling ratio $\frac{\pi_\theta(a|s)}{\beta(a|s)}$ to adjust

Deterministic Policy Gradient

- Deterministic policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$
- Policy objective function

$$J(\theta) = \sum_s p_1(s)\, v_{\pi_{\boldsymbol{\theta}}}(s)$$
$$= \sum_s p_1(s)\, q_\pi(s, \pi_\theta(s))$$

- Deterministic policy-gradient

$$\nabla_\theta J(\theta) \propto \sum_s \mu_{\pi_\theta}(s) \nabla_\theta \pi_\theta(s)\, \nabla_a q_\pi(s,a)|_{a=\pi_\theta(s)}$$

Deterministic Policy Gradient (cont.)

- On-Policy Deterministic Actor-Critic

$$\delta_t \leftarrow R_t + \gamma \hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t)$$
$$w_{t+1} \leftarrow w_t + \alpha_w \delta_t \nabla_w \hat{q}_w(s_t, a_t)$$
$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \nabla_\theta \pi_\theta(s) \nabla_a \hat{q}_w(s, a)|_{a=\pi_\theta(s)}$$

→ Critic uses Sarsa updates

- Off-Policy Deterministic Actor-Critic (with trajectories generated by $\beta(a|s)$)

$$\delta_t \leftarrow R_t + \gamma \hat{q}_w(s_{t+1}, \pi_\theta(s_{t+1})) - \hat{q}_w(s_t, a_t)$$
$$w_{t+1} \leftarrow w_t + \alpha_w \delta_t \nabla_w \hat{q}_w(s_t, a_t)$$
$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \nabla_\theta \pi_\theta(s) \nabla_a \hat{q}_w(s, a)|_{a=\pi_\theta(s)}$$

→ Critic uses Q-learning: no importance sampling needed

→ Actor uses deterministic policy: no importance sampling needed

Deep Deterministic Policy Gradient (DDPG)

- model-free off-policy actor-critic algorithm combining DPG and DQN

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

- Lillicrap, et al., "Continuous control with deep reinforcement learning", ICLR, 2016

Key DRL Algorithms

- On-Policy
  - REINFORCE (1987)
  - Vanilla Policy Gradient (VPG, 2000)
  - Trust Region Policy Optimization (TRPO, 2015)
  - Proximal Policy Optimization (PPO, 2017)
  - …
- Off-Policy
  - Deep Q-Networks (DQN, 2013)
  - Deep Deterministic Policy Gradient (DDPG, 2015)
  - Soft Actor-Critic (SAC, 2018)
  - …