Chapter 13: Policy Gradient Methods

Seungjae Ryan Lee

endtoend.ai

Preview: Policy Gradients

- Action-value Methods
  - Learn values of actions and select actions with estimated action values
  - Policy derived from action-value estimates
- **Policy Gradient Methods**
  - Learn parameterized policy that can select action without a value function
  - Can still use value function to *learn* the policy parameter

Policy Gradient Methods

- Define a performance measure J(θ) to maximize
- Learn policy parameter θ through *approximate gradient ascent*

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \underbrace{\widehat{\nabla J(\boldsymbol{\theta}_t)}}_{\text{Stochastic estimate of } J(\theta)}$$

Soft-max in Action Preferences

- Numerical preference h(s, a, θ) for each state-action pair
- Action selection through soft-max

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

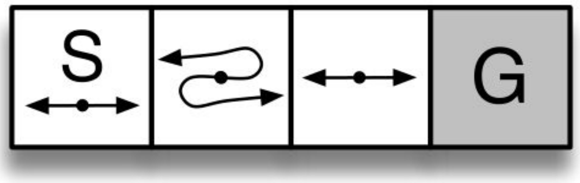
Soft-max in Action Preferences: Advantages

1. **Approximate policy can approach deterministic policy**
    - No “limit” like ε-greedy methods
    - Using soft-max on action values cannot approach deterministic policy

Soft-max in Action Preferences: Advantages

2. **Allow stochastic policy**
    - Best approximate policy can be stochastic in problems with significant function approximation

- Consider small corridor with -1 reward on each step
    - States are indistinguishable
    - Action transition is reversed in the second state

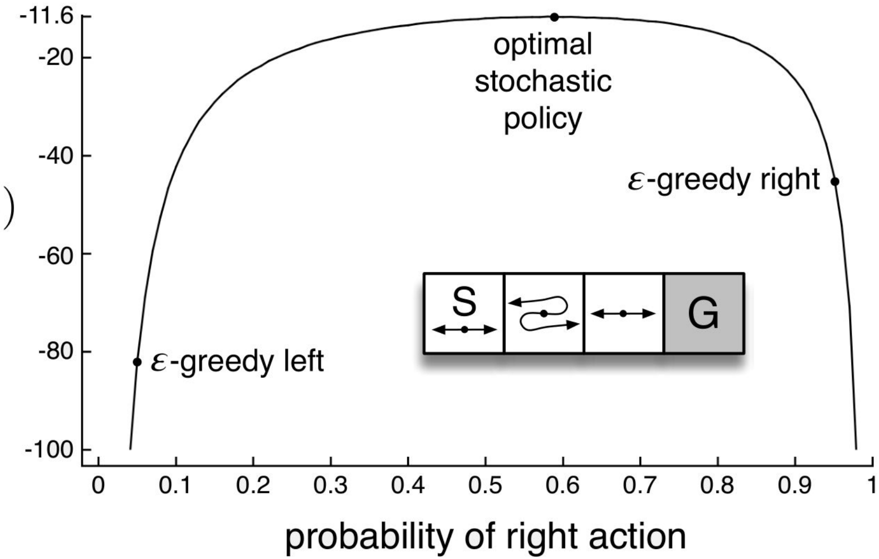


Soft-max in Action Preferences: Advantages

2. Allow stochastic policy

- ε-greedy methods (ε=0.1) cannot find optimal policy

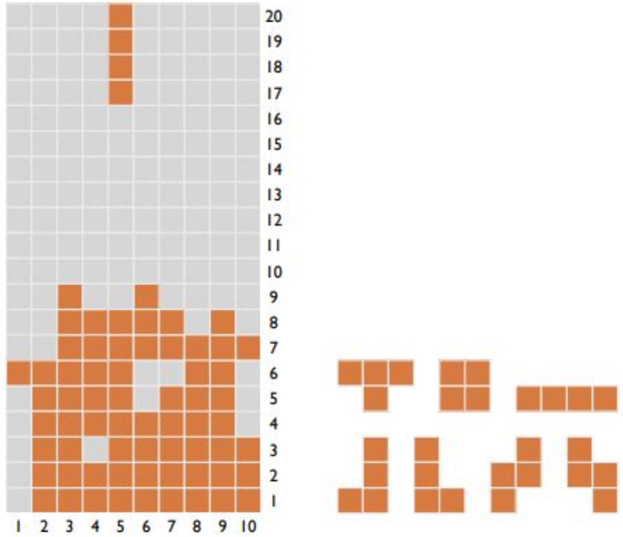
$$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\mathsf{S})$$



endtoend.ai

Soft-max in Action Preferences: Advantages

3. **Policy may be simpler to approximate**
   - Differs among problems

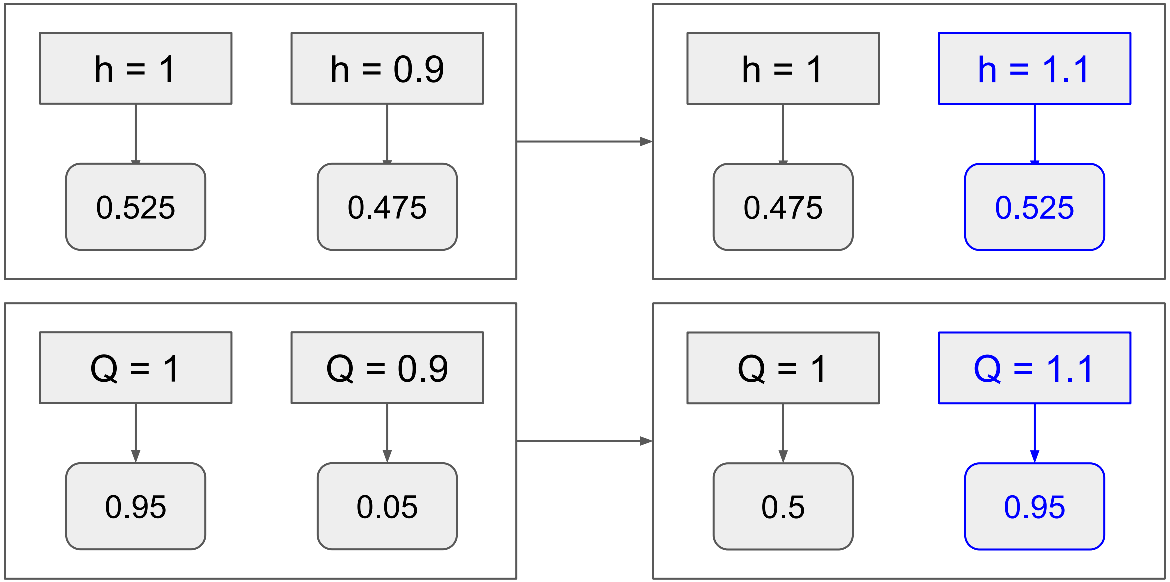


http://proceedings.mlr.press/v48/simsek16.pdf

endtoend.ai

Theoretical Advantage of Policy Gradient Methods

- Smooth transition of policy for parameter changes
- Allows for stronger convergence guarantees



endtoend.ai

Policy Gradient Theorem

- Define performance measure as value of the start state

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

- Want to compute $\nabla J(\boldsymbol{\theta})$ w.r.t. policy parameter $\boldsymbol{\theta}$

Policy Gradient Theorem

On-policy state distribution

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

Episodic: Average episode length
Continuing: 1

Policy Gradient Theorem: Proof

$$\nabla v_\pi(s)$$

$$\downarrow$$

$$\nabla \left[\sum_a \pi(a|s) q_\pi(s,a)\right]$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\nabla \left[\sum_a \pi(a|s) q_\pi(s,a)\right]$$

$$\xrightarrow{\text{Product rule}}$$

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a)\Big]$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a)\Big]$$

$$\downarrow$$

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r\,|\,s,a)\big(r + v_\pi(s')\big)\Big]$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s', r \,|\, s, a) \big(r + v_\pi(s') \big) \Big]$$

$$\Big\downarrow \; \nabla r = 0$$

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s' \,|\, s, a) \nabla v_\pi(s') \Big]$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s' \,|\, s,a) \nabla v_\pi(s') \Big]$$

$$\big\downarrow \text{Unrolling}$$

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s' \,|\, s,a)$$
$$\sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s'' \,|\, s',a') \nabla v_\pi(s'') \big] \Big]$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s' \,|\, s,a)$$

$$\sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s'' \,|\, s', a') \nabla v_\pi(s'') \big] \Big]$$

$$\downarrow$$

$$\sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)$$

Policy Gradient Theorem: Proof

$$\sum_{x \in \mathcal{S}} \sum_{k=0} \Pr(s \rightarrow x, k, \pi) \sum_{a} \nabla \pi(a|x) q_{\pi}(x, a)$$

$$\downarrow$$

$$\sum_{s} \eta(s) \sum_{a} \nabla \pi(a|s) q_{\pi}(s, a)$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$\downarrow$$

$$\sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\sum_{s'} \eta(s') \sum_{s} \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_{a} \nabla \pi(a|s) q_\pi(s, a)$$

$$\downarrow$$

$$\sum_{s'} \eta(s') \sum_{s} \mu(s) \sum_{a} \nabla \pi(a|s) q_\pi(s, a)$$

endtoend.ai

Policy Gradient Theorem: Proof

$$\nabla v_\pi(s) = \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$\Big\downarrow \ \textstyle\sum_{s'} \eta(s') \text{ is a constant}$$

$$\nabla v_\pi(s) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

Stochastic Gradient Descent

- Need samples with expectation $\nabla J(\theta)$

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

Stochastic Gradient Descent

$$\sum_{s} \mu(s) \sum_{a} q_\pi(s, a) \nabla \pi(a|s, \boldsymbol{\theta})$$

μ is an on-policy state distribution of π

$$\mathbb{E}_\pi \left[\sum_{a} q_\pi(S_t, a) \nabla \pi(a|S_t, \boldsymbol{\theta}) \right]$$

endtoend.ai

Stochastic Gradient Descent

$$\mathbb{E}_\pi\left[\sum_a q_\pi(S_t, a)\nabla\pi(a|S_t, \boldsymbol{\theta})\right]$$

$$\downarrow$$

$$\mathbb{E}_\pi\left[\sum_a \pi(a|S_t, \boldsymbol{\theta})q_\pi(S_t, a)\frac{\nabla\pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})}\right]$$

Stochastic Gradient Descent

$$\mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right]$$

Replace a with sample $A_t \sim \pi$

$$\mathbb{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$

Stochastic Gradient Descent: REINFORCE

$$\mathbb{E}_\pi\left[q_\pi(S_t, A_t)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$

$$\downarrow$$

$$\mathbb{E}_\pi\left[G_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$

REINFORCE (1992)

- Sample return like Monte Carlo
- Increment proportional to return
- Increment inverse proportional to action probability
    - Prevent frequent actions dominating due to frequent updates

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

REINFORCE: Pseudocode

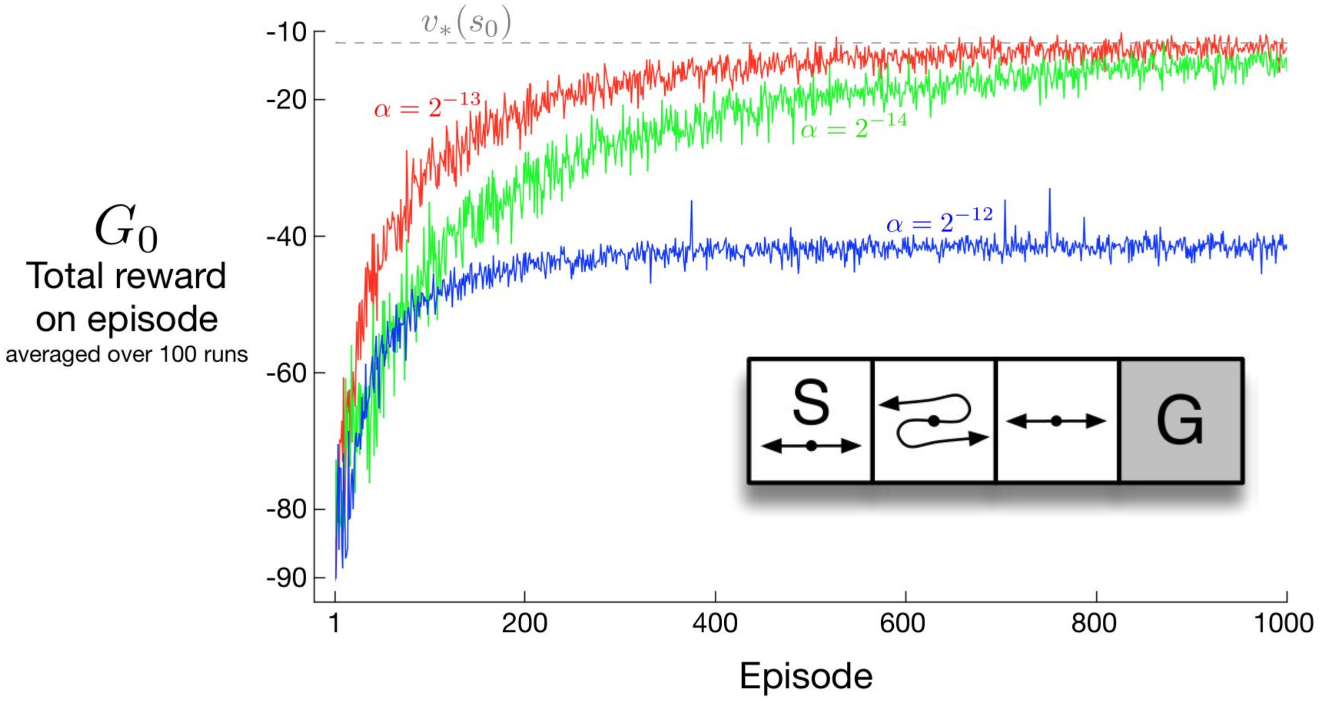
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\quad (G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \boxed{\nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})}$ Eligibility vector

endtoend.ai

REINFORCE: Results



endtoend.ai

REINFORCE with Baseline

- REINFORCE
  - Good theoretical convergence
  - Bad convergence speed due to **high variance**

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

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

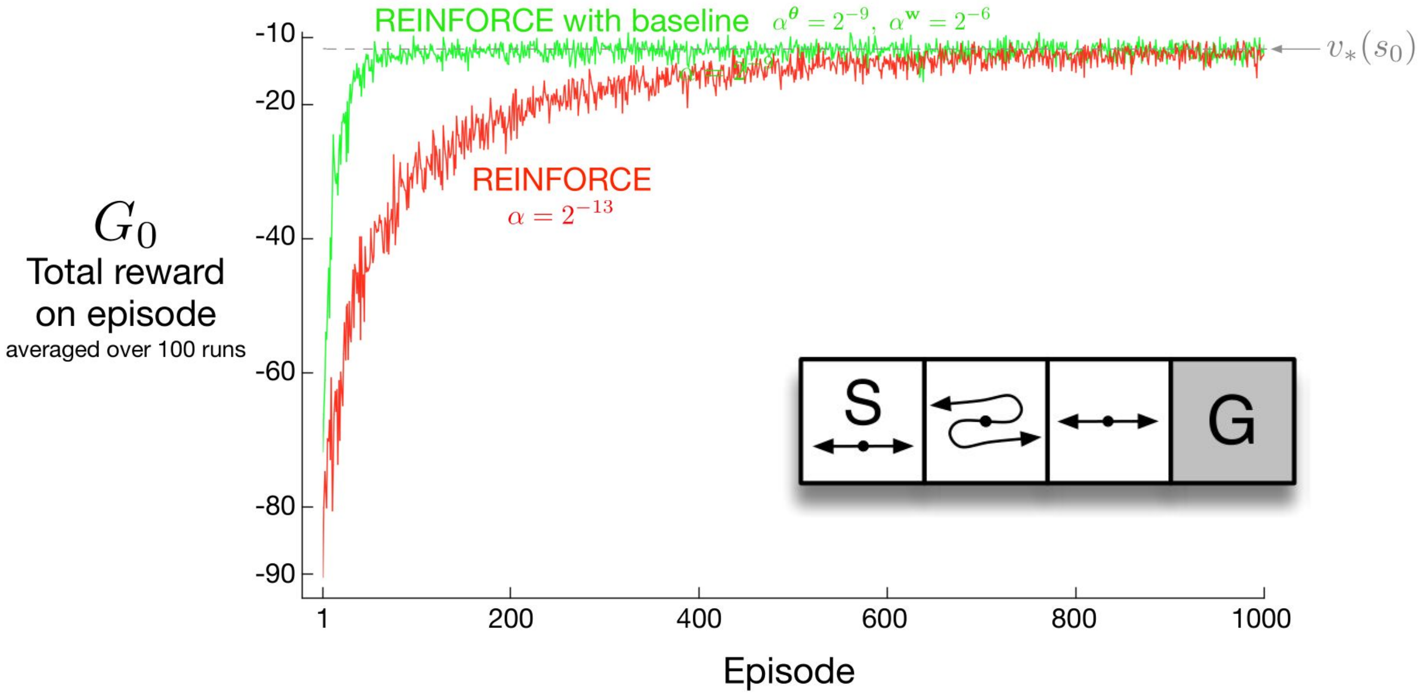
REINFORCE with Baseline: Pseudocode

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)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\boxed{\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})}$ Learn state value with MC to use as baseline
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

REINFORCE with Baseline: Results



endtoend.ai

Actor-Critic Methods

- Learn approximations for both policy (**Actor**) and value function (**Critic**)
- Critic vs Baseline in REINFORCE
  - Critic is used for *bootstrapping*
  - Bootstrapping introduces bias and relies on state representation
  - Bootstrapping reduces variance and accelerates learning

$$\pi(A, S, \theta)$$

Actor

$$\hat{v}(S, \mathbf{w})$$

Critic

One-step Actor Critic

- Replace return with one-step return
- Replace baseline with approximated value function (**Critic**)
    - Learned with semi-gradient TD(0)

REINFORCE:

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

One-step AC:

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

One-step Actor-Critic

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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$ Update **Critic** (value function) parameters
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$ Update **Actor** (policy) parameters
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Actor-Critic with Eligibility Traces

Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_\theta \approx \pi_*$

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

Average Reward for Continuing Problems

- Measure performance in terms of average reward

$$
\begin{aligned}
J(\boldsymbol{\theta}) \doteq r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\
&= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\
&= \sum_{s} \mu(s) \sum_{a} \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a) r,
\end{aligned}
$$

Actor-Critic for Continuing Problems

Actor–Critic with Eligibility Traces (continuing), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\bar{R}} > 0$
Initialize $R \in \mathbb{R}$ (e.g., to 0)
Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
Loop forever (for each time step):
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  Take action A, observe S', R
  $\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}}\delta$
  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

Policy Gradient Theorem Proof (Continuing Case)

1. Same procedure:

$$\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r\,|\,s,a) \big(\underline{r - r(\boldsymbol{\theta})} + v_\pi(s') \big) \Big] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \big[\underline{-\nabla r(\boldsymbol{\theta})} + \sum_{s'} p(s'\,|\,s,a) \nabla v_\pi(s') \big] \Big]
\end{aligned}$$

2. Rearrange equation:

$$\nabla r(\boldsymbol{\theta}) = \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'\,|\,s,a) \nabla v_\pi(s') \Big] - \nabla v_\pi(s).$$

$$\nabla r(\boldsymbol{\theta}) \downarrow \nabla J(\boldsymbol{\theta})$$

endtoend.ai

Policy Gradient Theorem Proof (Continuing Case)

$$\nabla J(\boldsymbol{\theta}) = \sum_a \Big[\nabla\pi(a|s)q_\pi(s,a) + \pi(a|s)\sum_{s'} p(s'|s,a)\nabla v_\pi(s')\Big] - \nabla v_\pi(s).$$

3. Sum over all states weighted by state-distribution $\mu(s)$
   a. Nothing changes since neither side depend on *s* and $\sum_s \mu(s) = 1$

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \underline{\sum_s \mu(s)} \left(\sum_a \Big[\nabla\pi(a|s)q_\pi(s,a) + \pi(a|s)\sum_{s'} p(s'|s,a)\nabla v_\pi(s')\Big] - \nabla v_\pi(s) \right) \\
&= \sum_s \mu(s) \sum_a \nabla\pi(a|s)q_\pi(s,a) \\
&\quad + \sum_s \mu(s) \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\nabla v_\pi(s') - \sum_s \mu(s)\nabla v_\pi(s) \\
&= \sum_s \mu(s) \sum_a \nabla\pi(a|s)q_\pi(s,a) \\
&\quad + \sum_{s'} \sum_s \mu(s) \sum_a \pi(a|s)p(s'|s,a)\,\nabla v_\pi(s') - \sum_s \mu(s)\nabla v_\pi(s)
\end{aligned}$$

Policy Gradient Theorem Proof (Continuing Case)

$$\nabla J(\boldsymbol{\theta}) = \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_{s'} \underline{\sum_s \mu(s) \sum_a \pi(a|s) p(s'|s,a)} \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s)$$

4. Use **ergodicity**: $\sum_s \mu(s) \sum_a \pi(a|s,\boldsymbol{\theta}) p(s'|s,a) = \mu(s'), \text{ for all } s' \in \mathcal{S}.$

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_{s'} \underline{\mu(s')} \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a).\end{aligned}$$

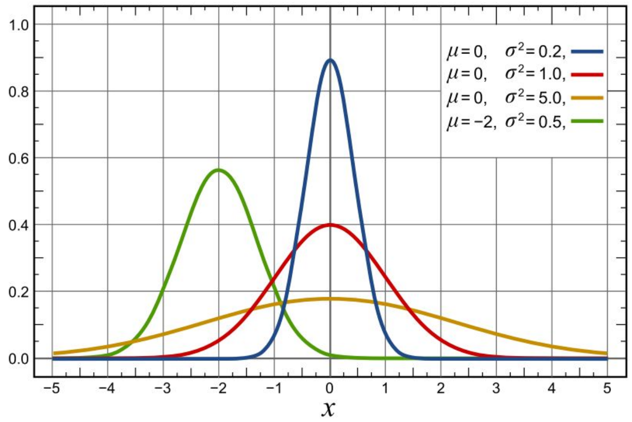
Q.E.D.

Policy Parameterization for Continuous Actions

- Policy based methods can handle **continuous action spaces**
- Learn statistics of the probability distribution
  - ex) mean and variance of Gaussian
- Choose action from the learned distribution
  - ex) Gaussian distribution

$$p(x) \doteq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{1}{\underline{\sigma(s,\boldsymbol{\theta})}\sqrt{2\pi}} \exp\left(-\frac{(a-\underline{\mu(s,\boldsymbol{\theta})})^2}{2\underline{\sigma(s,\boldsymbol{\theta})}^2}\right)$$



endtoend.ai

Policy Parametrization to Gaussian Distribution

- Divide policy parameter vector into mean and variance: $\boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]^\top$
- Approximate mean with linear function:

$$\mu(s, \boldsymbol{\theta}) \doteq {\boldsymbol{\theta}_\mu}^\top \mathbf{x}_\mu(s)$$

- Approximate variance with exponential of linear function:
    - Guaranteed positive

$$\sigma(s, \boldsymbol{\theta}) \doteq \exp\Big({\boldsymbol{\theta}_\sigma}^\top \mathbf{x}_\sigma(s)\Big)$$

- All PG algorithms can be applied to the parameter vector

Summary

- **Policy Gradient methods** have many advantages over action-value methods
  - Represent stochastic policy and approach deterministic policies
  - Learn appropriate levels of exploration
  - Handle continuous action spaces
  - Compute effect of policy parameter on performance with Policy Gradient Theorem
- **Actor-Critic** estimates value function for bootstrapping
  - Introduces bias but is often desirable due to lower variance
  - Similar to preferring TD over MC

"Policy Gradient methods provide a significantly different set of strengths and weaknesses than action-value methods."

Thank you!

Original content from

- Reinforcement Learning: An Introduction by Sutton and Barto

You can find more content in

- github.com/seungjaeryanlee
- www.endtoend.ai