Policy Gradient Methods

2018.12.11
발표자 : 장유환
발표자료 원본 / 원작자 출처
https://docs.google.com/presentation/d/1I3QqfY6h2Pb0a-KEIbKy6v5NuZtnTMLN16Fl-IuNtUo/export/pptx?id=1I3QqfY6h2Pb0a-KEIbKy6v5NuZtnTMLN16Fl-IuNtUo&pageid=p

https://www.facebook.com/littleqoo

Agenda

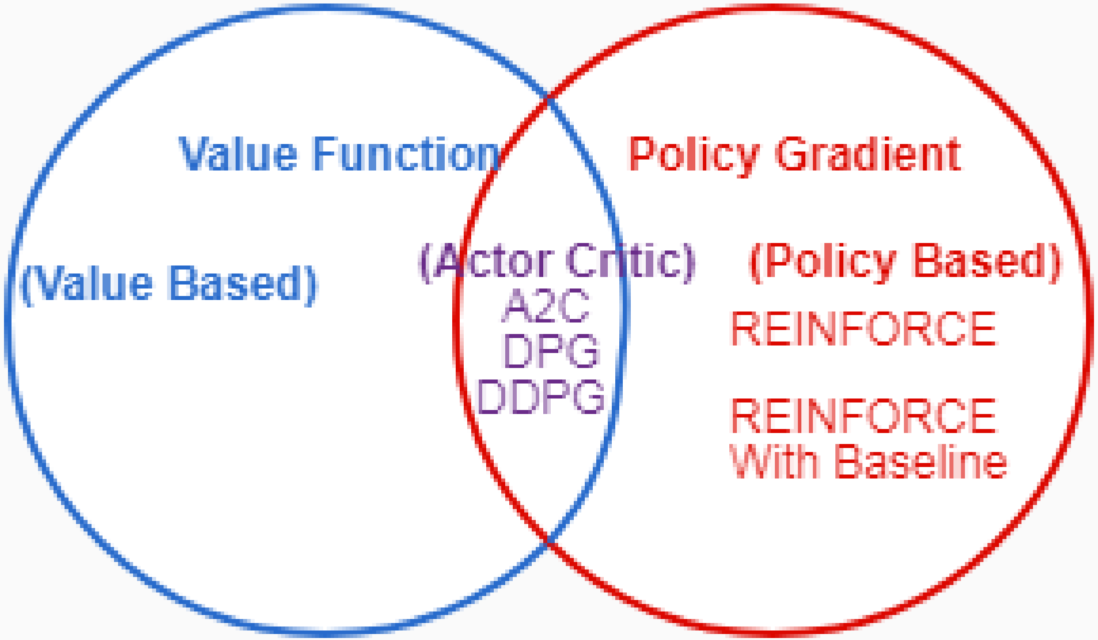
Ch.13 Policy Gradient Methods

- Intro & Policy Gradinet Theorem
- REINFORCE: Monte -Carlo Policy Gradient
- One-Step Actor Critic
- Actor Critic with Eligibility Trace (Eposodic and Continuing Case)
- **Policy Gradient for Continuing Problems (??)**
- Policy Parameterization for Continuous Actions
- Summary

Introduction

Reinforcement Learning Classification

- Value-Based
  - Learned Value Function
  - Implicit Policy (usually Ɛ-greedy)
- Policy-Based
  - No Value Function
  - Explicit Policy Parameterization
- Mixed(Actor-Critic)
  - Learned Value Function
  - Policy Parameterization



Policy Gradient Method

Goal : $\pi(a|s,\theta) = Pr(A_t = a | S_t = s, \theta_t = \theta)$

Performance Messure : $J(\theta)$

Optimization : Gradient Ascent $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$

[Actor-Critic Method] : Learn approximation to both policy and value function $\hat{v}(s,w)$

Policy Approximation (Discrete Actions)

- Ensure exploration we generally require that the policy **never becomes deterministic**

$$\pi(a|s,\theta) \in (0,1)$$

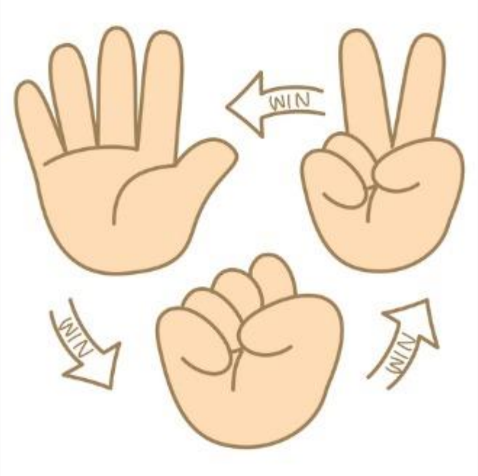
- for discrete action spaces - **Softmax in action preferences**
  - discrete action space can not too large
- Action preferences can be parameterization arbitrarily(linear, ANN...)

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

$$h(s,a,\theta) = \theta^T x(s,a)$$

Advantage of Policy Approximation

1. Enables the selection of actions with **arbitrary probabilities**
    - Bluffing in poker, Action-Value methods have no natural way
2. Prevents dramatic policy changes while approximating
    - In value-based approach, drastic policy change could be occurred due to small value fluctuation



Short Corridor With Switched Actions

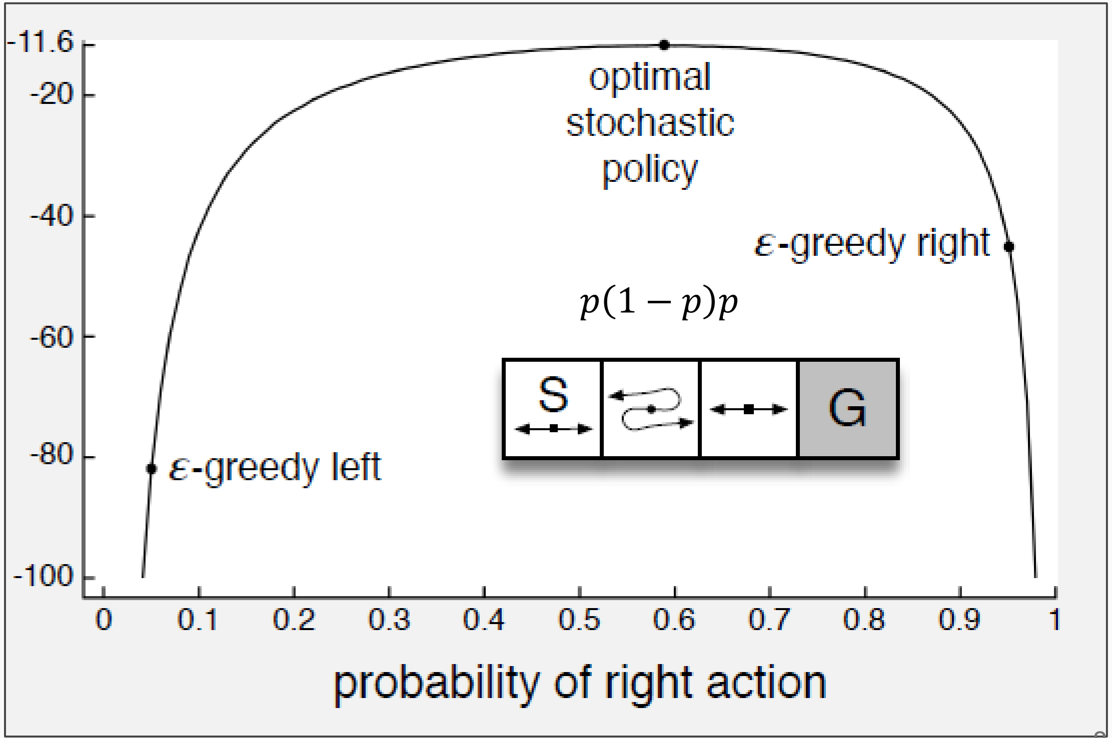
- All the states appear identical under the function approximation

- A method can do significantly better if it can learn a specific probability with which to select *right*

- The best probability is about 0.59

$$x(s, right) = [1, 0]^T$$
$$x(s, left) = [0, 1]^T$$

$$J(\theta) = V_{\pi_\theta}(S)$$



The Policy Gradient Theorem (Episodic)

The Policy Gradient Theorem

- Stronger **convergence** of guarantees are available for policy-gradient method than for action-value methods
  - Ɛ-greedy selection may change dramatically for an arbitrary small action value change that results in having the maximal value
- There are two cases define the different performance messures
  - Episodic Case - value of the start state of the episode
  - Continuing Case - no end even start state (Refer to Chap10.3)

$$\eta(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(S_0) \qquad \eta(\boldsymbol{\theta}) = \sum_s d_{\pi_{\boldsymbol{\theta}}}(s) v_{\pi_{\boldsymbol{\theta}}}(s)$$

The Policy Gradient Theorem (Episodic)

$$\nabla_{\pi_\theta} \sum_{s',r} p(s',r|s,a) \cdot r(s,a,s') = \nabla_{\pi_\theta} E[r(s,a,s')] = 0$$

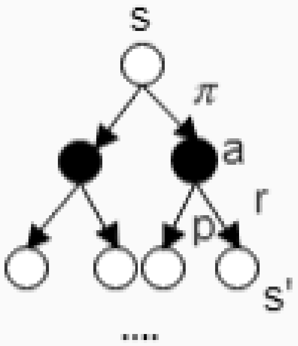
- Performance

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

- Gradient Ascent

$$\theta_{t+1} = \theta_t + \nabla J(\theta_t)$$

- Discount = 1



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

Bellman Equation

Cont.

- Performance

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

- Gradient Ascent

$$\theta_{t+1} = \theta_t + \nabla J(\theta_t)$$

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

recurisively unroll

The Policy Gradient Theorem (Episodic)

- Performance

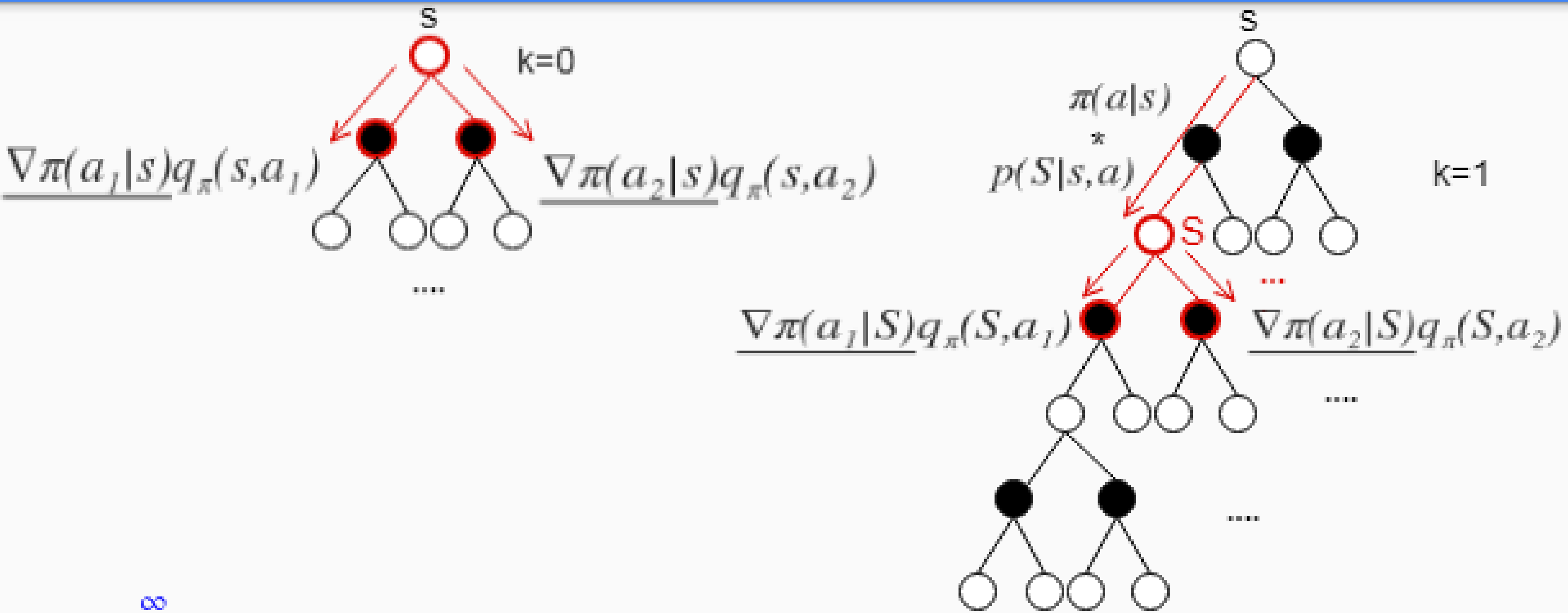
$$J(\theta) \doteq v_{\pi_\theta}(s_0)$$

- Gradient Ascent

$$\theta_{t+1} = \theta_t + \nabla J(\theta_t)$$

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

The Policy Gradient Theorem (Episodic) - Basic Meaning



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

The Policy Gradient Theorem (Episodic) - On Policy Distribution

$$\mu(s) = \frac{\eta(s)}{\sum_{s''} \eta(s'')}$$

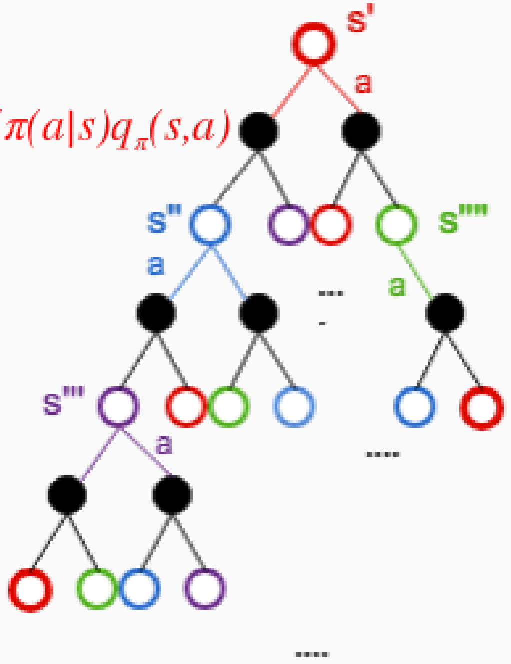
fraction of time spent in s that is usually under on-policy training

(on-policy distribution, the same as p.43)

$$\begin{aligned}
\nabla J(\theta) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\boxed{\sum_{s'} \eta(s')}} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)
\end{aligned}$$

better be written in $\sum_{s''} \eta(s'')$

The Policy Gradient Theorem (Episodic) - Concept



$$s \in \{s', s'', s''', s''''\}$$

$$\mu(s) = \frac{\eta(s)}{\sum_{s''} \eta(s'')}$$

Ratio of s that appears in the state-action tree

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

$$s \in \{s', s'', s''', s''''\}$$

Gathering gradients over all action spaces of every state

The Policy Gradient Theorem (Episodic)：

Sum Over States Weighted by How Offen the States Occur Under The Policy

- Policy gradient for episodic case $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi (s,a) \nabla \pi(a|s,\theta)$
- The distribution $\mu(s)$ is the on-policy distribution under π
- The constant of proportionality $\propto$ is the **average length** of an episode and can be absorbed to step size α $\left(\sum_{s'} \eta(s') \approx average \;\; length \;\; of \;\; an \;\; episode \right)$
- Performance's gradient ascent does <u>*not*</u> involve the derivative of the state distribution

REINFORCE : Monte-Carlo Policy Gradient

Classical Policy Gradient

REINFORCE Algorithm

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

$$= \mathrm{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right]_{\text{sample } S_t \sim \rho^\pi}$$

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

$$= \mathrm{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right]_{\text{sample } A_t \sim \pi}$$

$$= \mathrm{E}_\pi\left[G_t \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right]$$

because $E_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t)$

All Actions Method

$$\Longrightarrow \underline{\theta_{t+1} = \theta_t + \alpha \sum_a \hat{q}(S_t,a,w) \nabla \pi(a|S_t,\theta)}$$

Classical Monte-Carlo

$$\Longrightarrow \underline{\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}}$$

REINFORCE Meaning

- The update increases the parameter vector in this direction proportional to the return
- **inversely proportional to the action probability** (make sense because otherwise actions that are selected frequently are at an advantage)

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

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

$$=\mathrm{E}_\pi\left[G_t\frac{\nabla\pi(a|S_t,\theta)}{\boxed{\pi(a|S_t,\theta)}}\right]$$

Action is a summation. If using samping by action probability, we have to average gradient by sampling number

REINFORCE Algorithm

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

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

Wait Until One Episode Generated

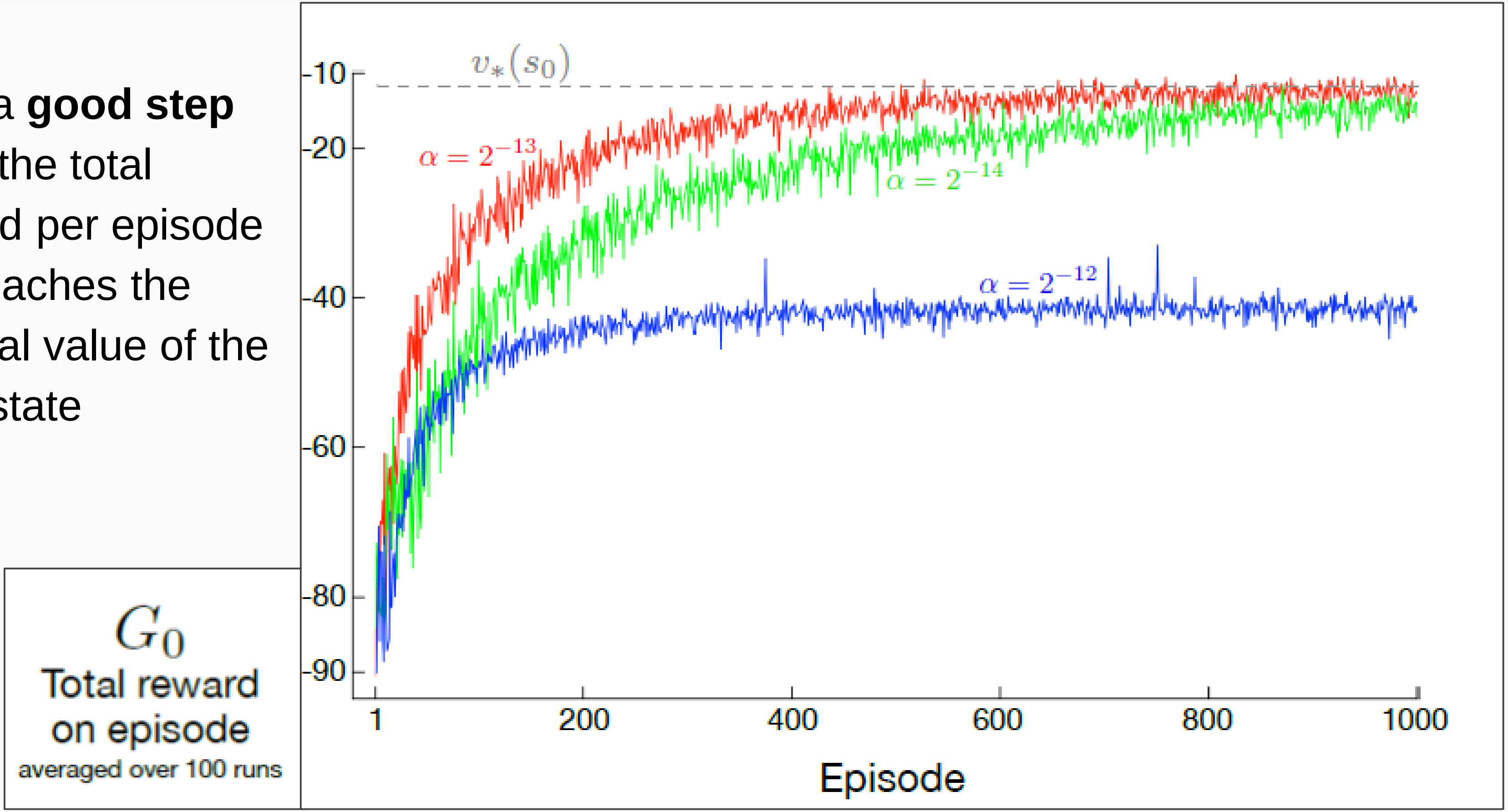
Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\quad (G_t)$
        $\boldsymbol{\theta}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

$$R_0, \ \gamma R_1, \ \gamma^2 R_2 \ldots$$

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

REINFORCE on the short-corridor gridworld

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



REINFORCE Defect & Solution

- Slow Converge
- High Variance From Reward
- Hard To Choose Learning Rate

REINFORCE with Baseline (episodic)

- Expected value of the update unchanged(**unbiased**), but it can have a large effect on its **variance**
- Baseline can be any function, evan a random variable
- For MDPs, the baseline should vary with state, one natural choice is **state value function**
  - some states all actions have high values => a high baseline
  - in others, => a low baseline

$$\Longrightarrow \nabla J(\theta) \propto \sum_s \mu(s) \sum_a \left(q_\pi(s,a) - b(s)\right) \nabla \pi(a|s,\theta)$$

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

Treat State-Value Function as a Independent Value-function Approximation!

$$\arg\min_w \left(v(s) - \hat{v}(s,w)\right)^2$$

$$w_{t+1} = w_t + \alpha\left(G_t - \hat{v}(S_t,w)\right) \nabla \hat{v}(S_t,w)$$

REINFORCE with Baseline (episodic)

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 $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\underline{\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

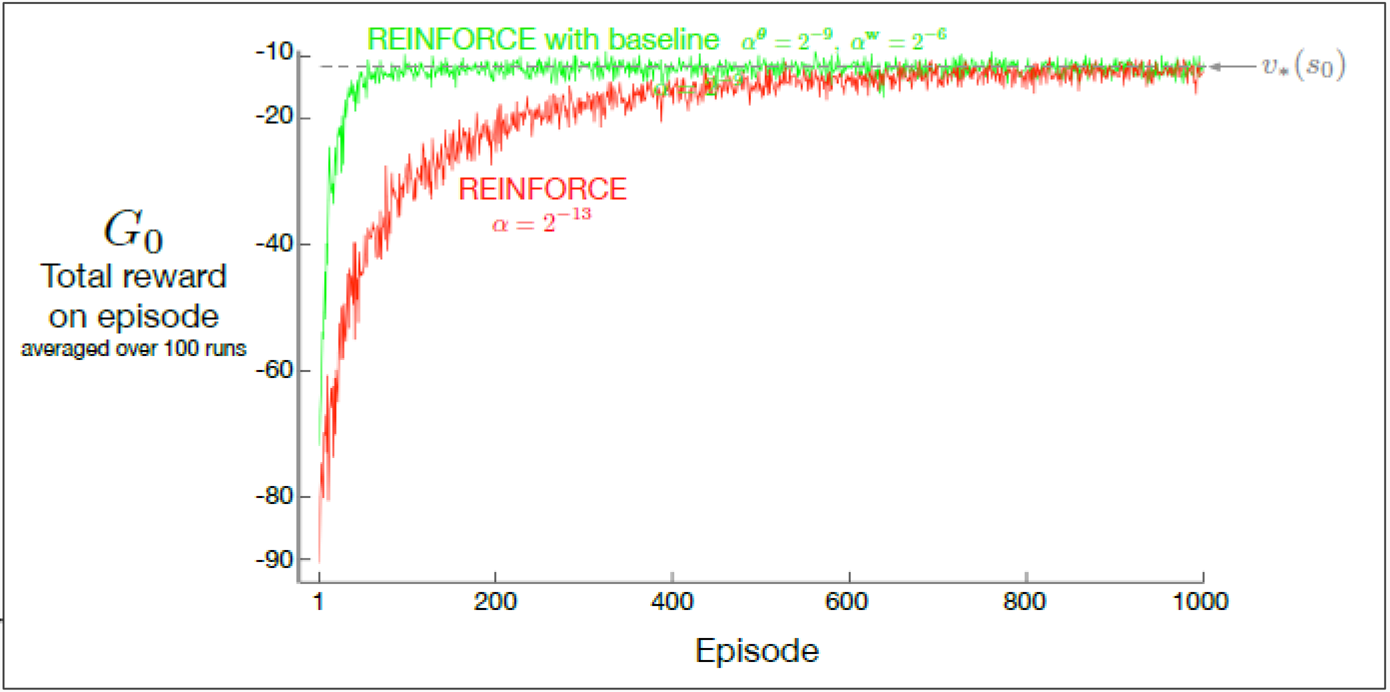
$\hat{v}(S_t,w)$ can be learned by any methods$^{(G_t)}$ of previous chapters independently.
We use the same Monte-Carlo here.(Section 9.3 Gradient Monte-Carlo)

Short-Corridor GridWorld

- Learn much faster
- Policy parameter is much less clear to set
- State-Value function paramenter (Section 9.6)

$$\alpha^w = \frac{0.1}{E[||\nabla \hat{v}(S_t, w)||_\mu^2]}$$

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



Defects

- Learn Slowly (product estimates of high variance)

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

- Incovenient to implement online or continuing problems

Actor-Critic Methods

Combine Policy Function with Value Function

One-Step Actor-Critic Method

- Add One-step bootstrapping to make it **online**
- But TD Method always **introduces bias**
- The TD(0) with only one random step has **lower variance** than Monte-Carlo and **accelerate learning**

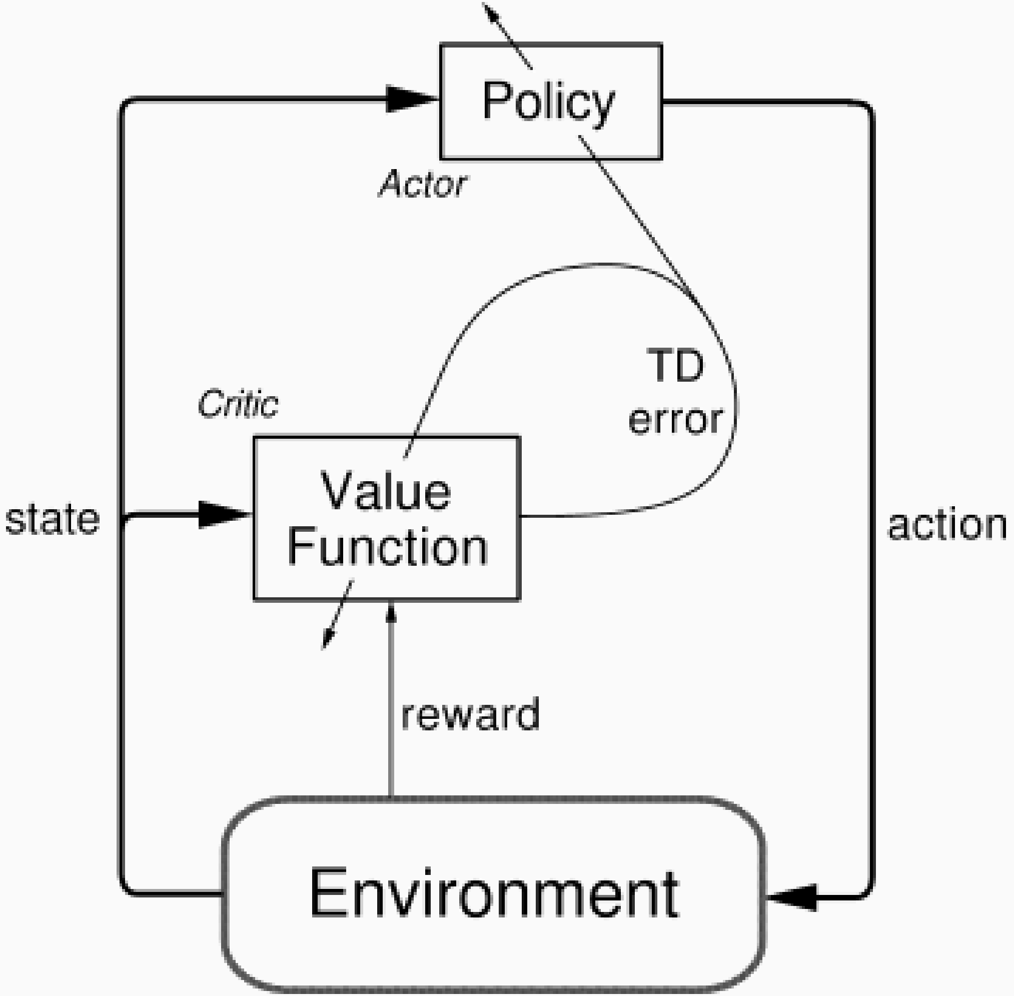
$$\theta_{t+1} \doteq \theta_t + \alpha\left(G_{t:t+1} - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

$$= \theta_t + \alpha\left(R_{t+1} + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

$$= \theta_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

Actor-Critic

- Actor - Policy Function
- Critic- State-Value Function
- Critic Assign Credit to Cricitize Actor’s Selection



https://cs.wmich.edu/~trenary/files/cs5300/RLBook/node66.html

One-step Actor-Critic Algorithm (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
        $\boxed{\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})}$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Independent Semi-Gradient TD(0) (Session 9.3)

TD(0), which uses $U_t \doteq R_{t+1} + \gamma \hat{v}(S_{t+1},\mathbf{w})$

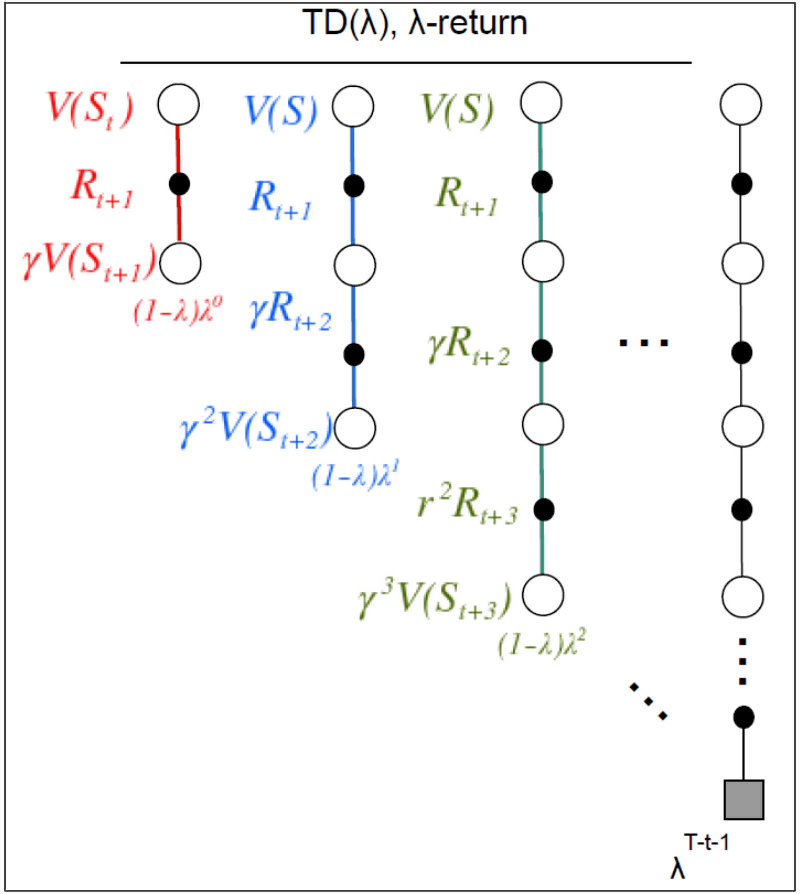
Actor-Critic with Eligiblity Traces (episodic)

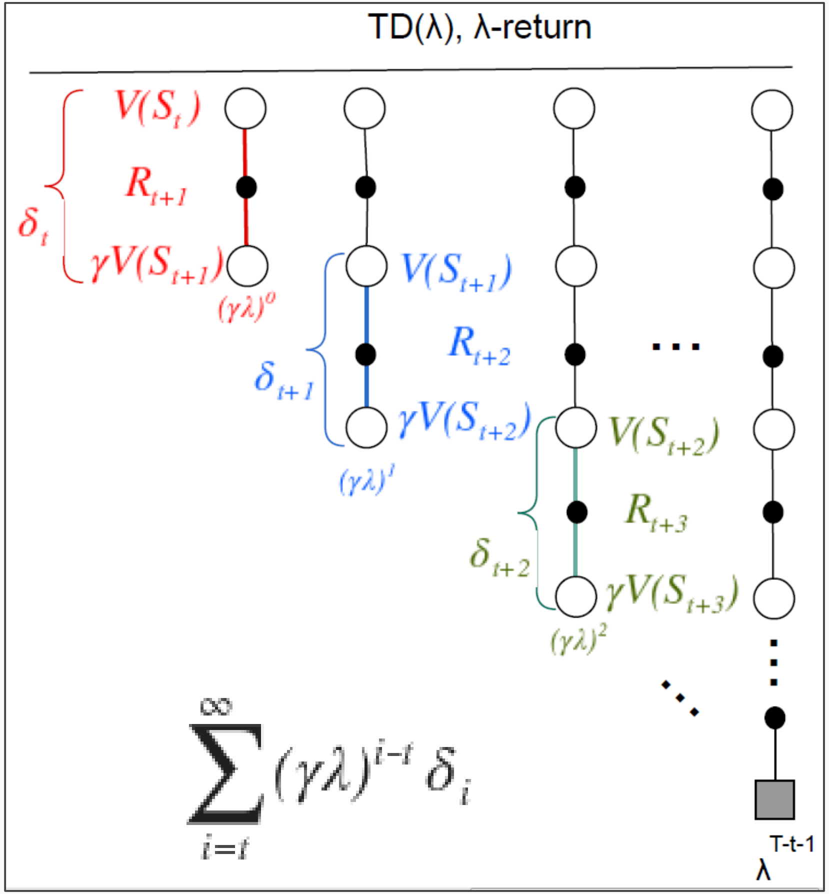
- Weight Vector is a long-term memory
- Eligibility trace is a short-term memory, keeping track of which components of the weight vector have contributed to recent state

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

Recall Eligibility Traces - Forward View (Optional)

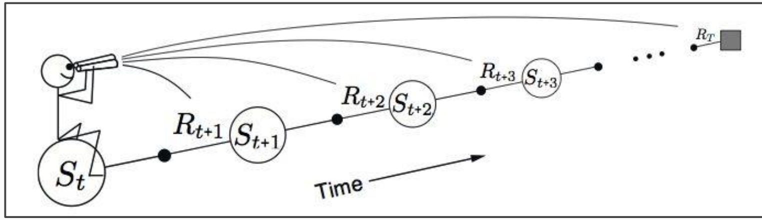




Recall Eligibility Traces - Forward View (Optional)

$$
\begin{aligned}
G_t^\lambda - V(S_t) = -V(S_t) &+ (1-\lambda)\lambda^0[R_{t+1}+\gamma V_t(S_{t+1})] \quad \textbf{TD(0)} \\
&+ (1-\lambda)\lambda^1[R_{t+1}+\gamma R_{t+2}+\gamma^2 V_t(S_{t+2})] \quad \textbf{TD(1)} \\
&+ (1-\lambda)\lambda^2[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\gamma^3 V_t(S_{t+3})] \quad \textbf{TD(2)} \\
&+ \ldots
\end{aligned}
$$

$$
\begin{aligned}
G_t^\lambda - V(S_t) = &\ (\gamma\lambda)^0[R_{t+1}+\gamma V_t(S_{t+1})-V(S_t)] \\
&+ (\gamma\lambda)^1[R_{t+2}+\gamma V_t(S_{t+2})-V(S_{t+1})] \\
&+ (\gamma\lambda)^2[R_{t+3}+\gamma V_t(S_{t+3})-V(S_{t+2})] \\
&+ \ldots = \sum_{i=t}^{\infty}(\gamma\lambda)^{i-t}\delta_i
\end{aligned}
$$



Eligibility Trace Indution Link https://cacoo.com/diagrams/gof2aiV3fCXFGJXF/36A44

The Policy Gradient Theorem (Continuing)

- **"Ergodicity Assumption"**
  - Any early decision by the agent can have only a **temporary effect**
  - **State Expectation** in the long run depends on **policy** and MDP **transition probabilities**
  - **Steady state distribution** $\mu(s)$ is assumed to **exist** and to be independent of S_0

Average Rate of Reward per Time Step

$$J(\theta) \doteq r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathrm{E}[R_t \mid S_0, A_{0:t-1} \sim \pi]$$

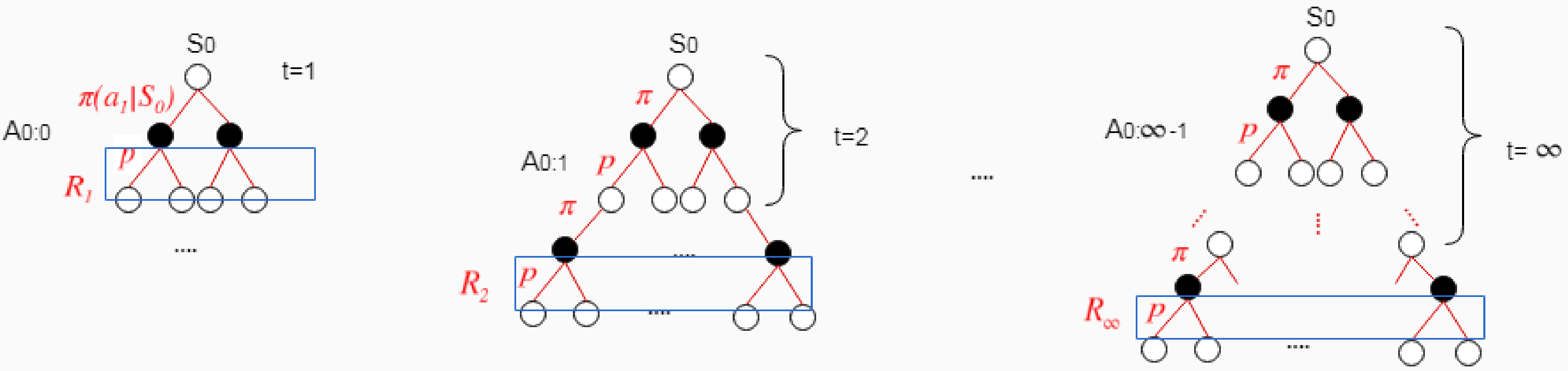
guarantee limit exist

$$= \lim_{h\to\infty} \mathrm{E}[R_t \mid S_0, A_{0:t-1} \sim \pi]$$

$$= \sum_s \mu(s) \underbrace{\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) r}_{V(s)}$$

($\gamma(\pi)$ is a fixed parameter for any π . We will treat it later as a linear function independent of s in the theorem)

The Policy Gradient Theorem (Continuing) - Performance Measure Definition



$\mathrm{E}[R_1 \mid S_0, A_{0:0}]$ $\mathrm{E}[R_2 \mid S_0, A_{0:1}]$ $\mathrm{E}[R_\infty \mid S_0, A_{0:\infty-1}]$

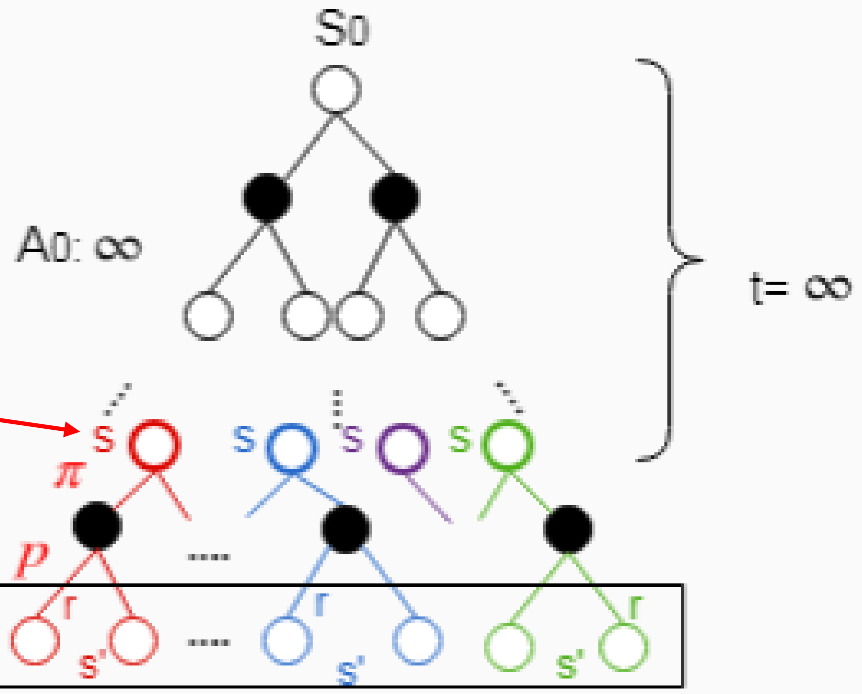
$$J(\theta) \doteq r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathrm{E}[R_t \mid S_0, A_{0:t-1} \sim \pi]$$
$$= \lim_{h\to\infty} \mathrm{E}[R_t \mid S_0, A_{0:t-1} \sim \pi]$$

“Every Step’s Average Reward Is The Same”

The Policy Gradient Theorem (Continuing) - Steady State Distribution

$$\mu(s) \doteq \lim_{t \to \infty} Pr\{S_t = s | A_{0:t} \sim \pi\}$$

Steady State Distribution Under π



$$r(\pi_\theta) = \sum_s \mu(s) \sum_a \pi(a|s,\theta) \sum_{s',r} p(s',r|s,a)\, r$$

Replace Discount with *Average Reward* for Continuing Problem(Session 10.3, 10.4)

- Continuing problem with discounted setting γ is useful in **tabular case**, but questionable for **function approximation** case
- In Continuing problem, performance measure with discounted setting is proportional to the average reward setting (They has almost the same effect)(session 10.4)
- Discounted setting is problematic with function approximation
  - with function approximation we have lost the ***policy improvement theorem*** (session 4.3) important in Policy Iteration Method

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \ldots$$

Proof The Policy Gradient Theorem (Continuing) 1/2

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} \qquad \text{(Exercise 3.18)}$$

$$\sum_{s',r} p(s',r|s,a) = 0$$

$$= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] \qquad \text{(product rule of calculus)}$$

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

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

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

Gradient Definition

$$\nabla r(\theta) = \nabla \sum_s \mu(s) v_\theta(s) = \sum_s \mu(s) \nabla v_\theta(s)$$

Parameterization of policy by replacing discount with average reward setting

Proof The Policy Gradient Theorem (Continuing) 2/2

$$\nabla r(\theta) = \nabla J(\theta) = \sum_s \mu(s) \nabla r(\theta)$$

$$because \sum_s \mu(s) = 1$$ **Trick**

- Introduce steady state distribution $\mu(s)$ and its property

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

By Definistion, it's independent of s

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

$$+ \boxed{\sum_s \mu(s) \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)} \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s)$$

steady state distribution property

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

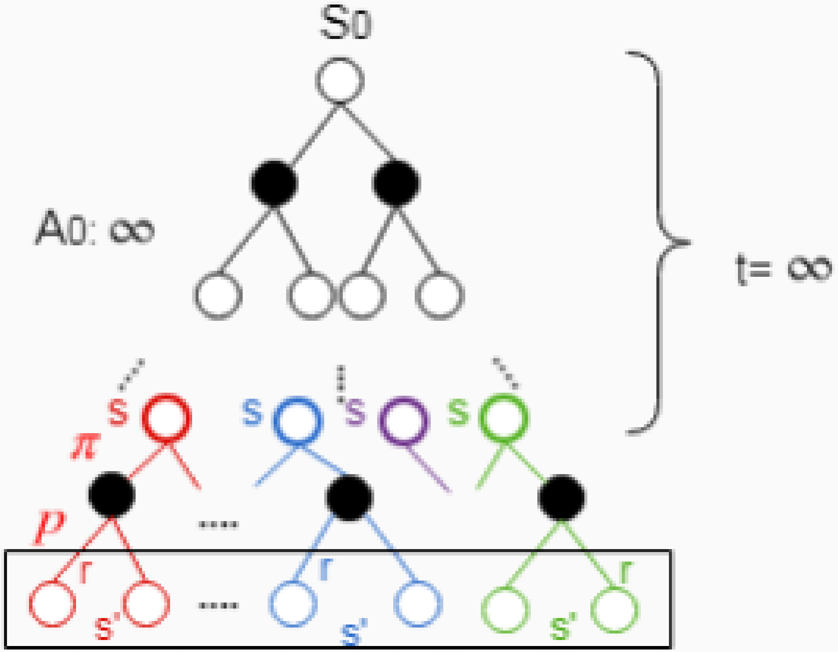
$$+ \boxed{\sum_{s'} \underbrace{\sum_s \mu(s) \sum_a \pi(a|s) p(s'|s,a)}_{\mu(s') \ (13.16)}} \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s)$$

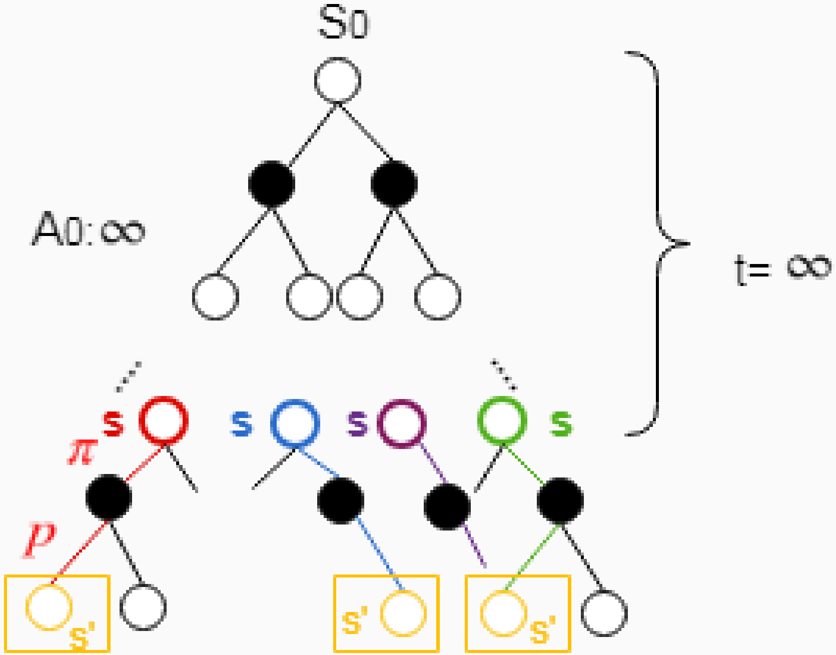
$$= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_{s'} \mu(s') \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s)$$

$$= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a). \qquad \text{Q.E.D.}$$

Steady State Distribution Property

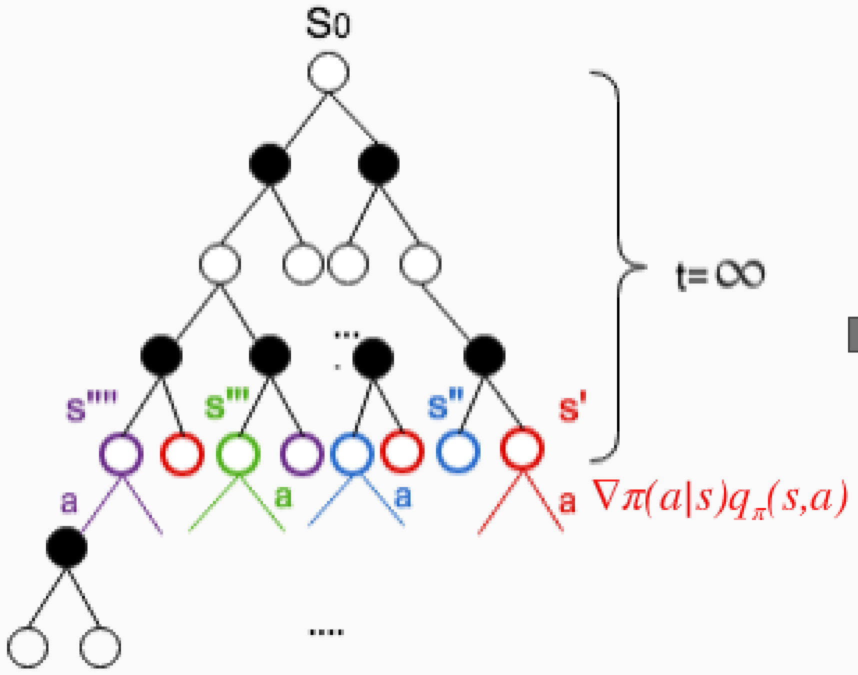
$$r(\pi_\theta)=\sum_s \mu(s) \sum_a \pi(a|s,\theta) \sum_{s',r} p(s',r|s,a)\, r$$





$$\sum_s \mu(s) \sum_a \pi(a|s,\theta) p(s'|s,a)=\mu(s')$$

Policy Gradient Theorem (Continuing) Final Concept



$$\mu(s) \doteq \lim_{t \to \infty} Pr\{S_t = s | A_{0:t} \sim \pi\}$$

$$\nabla J(\theta) = \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$s \in \{s', s'', s''', s''''\}$$

Actor-Critic with Eligibility Traces (continuing)

- Replace Discount γ with average reward $\bar{R}$
- Traing $\bar{R}$ with Semi-Gradient TD(0)

Actor–Critic with Eligibility Traces (continuing), 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: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{R} > 0$
Initialize $\bar{R} \in \mathbb{R}$ (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
    $\delta \leftarrow \boxed{R - \bar{R}} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
    $\boxed{\bar{R} \leftarrow \bar{R} + \alpha^{R}\delta}$
    $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S, \boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $S \leftarrow S'$

Independent Semi-Gradient TD(0)

$$\arg\max_{\bar{R}} (\underline{G(s,\bar{R})} - \hat{v}(s,w))^2$$

$$\bar{R} = \bar{R} + \alpha^R(\underline{R - \bar{R} + \hat{v}(s',w)} - \hat{v}(s,w))\underbrace{\boxed{\nabla \bar{R}}}_{}^{=1}$$

one step bootstrapping

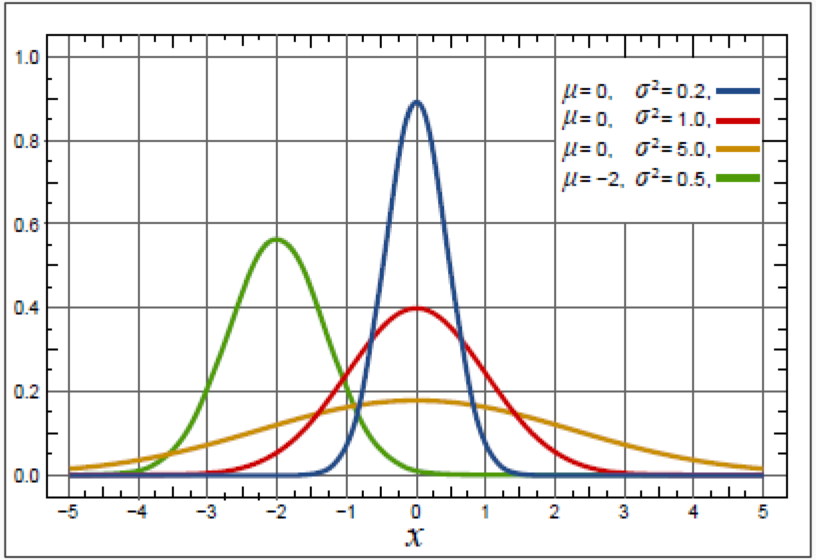
Policy Parameterization for Continuous Actions

- Can deal with **large** or **infinite continue actions spaces**
- Normal distribution of the actions are through the state's parameterization

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

$$\mu(s,\theta) \doteq \theta_\mu{}^T x_\mu(s), \quad \sigma(s,\theta) \doteq \boxed{exp}(\theta_\sigma{}^T x_\sigma(s))$$

Make it Positive



$x_\mu(s), \quad x_\sigma(s)$ Feature vectors constructed by Polynomial, Fourier... (Session 9.5)

Chapter 13 Summary

- Policy gradient is superior to Ɛ-greedy and action-value method in
  - Can learn specific probabilities for taking the actions
  - Can approach deterministic policies asymptotically
  - Can naturally handle continuous action spaces
- ***Policy gradient theorem*** gives an exact formula for how performance is a affected by the policy parameter that does not involve derivatives of the state distribution.
- REINFORCE method
  - Add State-Value as Baseline -> reduce variance without introducing bias
- Actor-Critic method
  - Add state-value function for bootstrapping ->introduce bias but reduce variance and accelerate learning
  - Critic assign credit to cricitize Actor’s selection