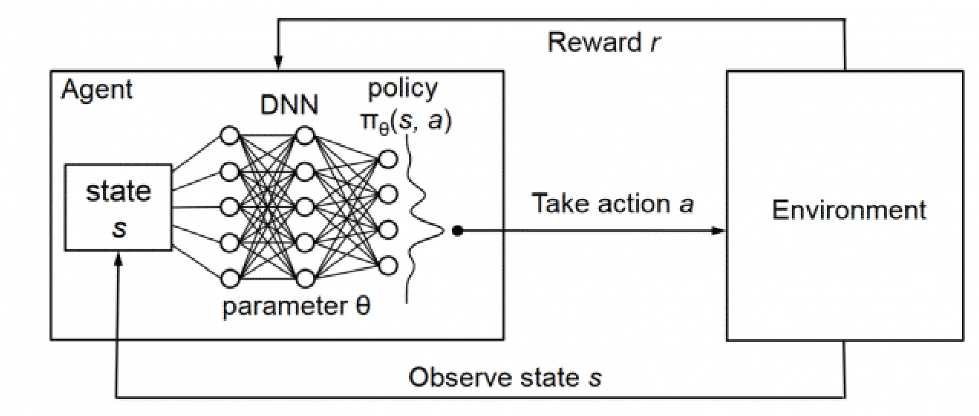
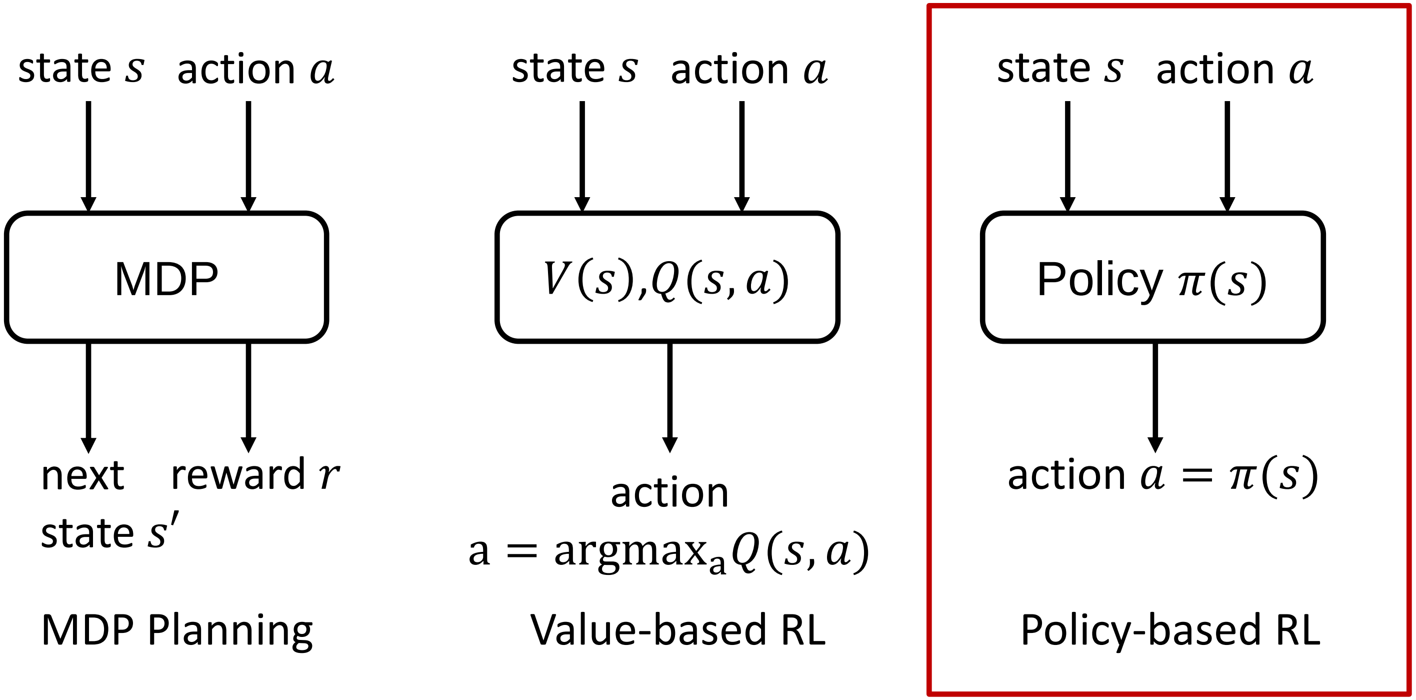
L7.3 Policy-based RL

Zonghua Gu 2021



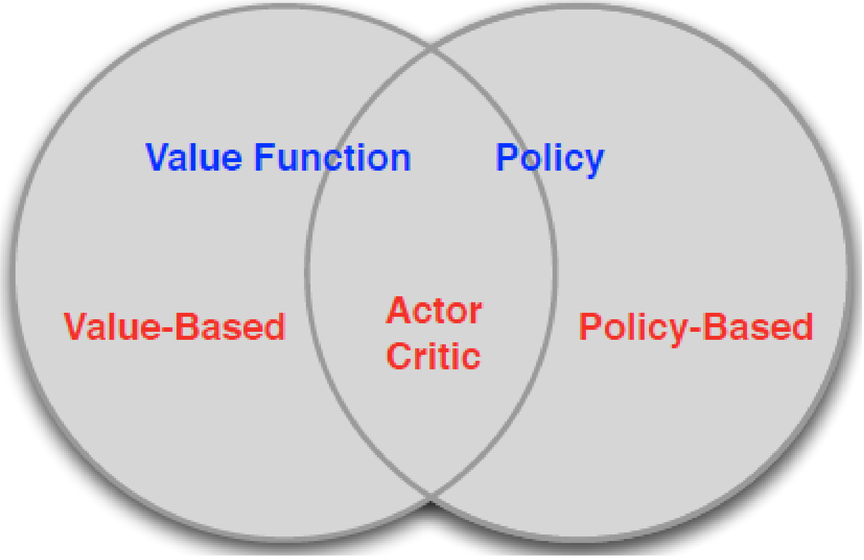
Acknowledgement: slides based on https://www.coursera.org/specializations/reinforcement-learning
And textbook by Sutton and Barto http://incompleteideas.net/book/the-book-2nd.html

Policy-based RL



CH13 Policy Gradient Methods

- 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

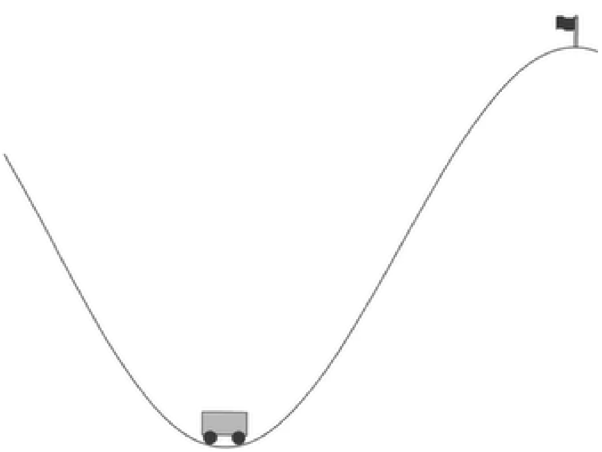


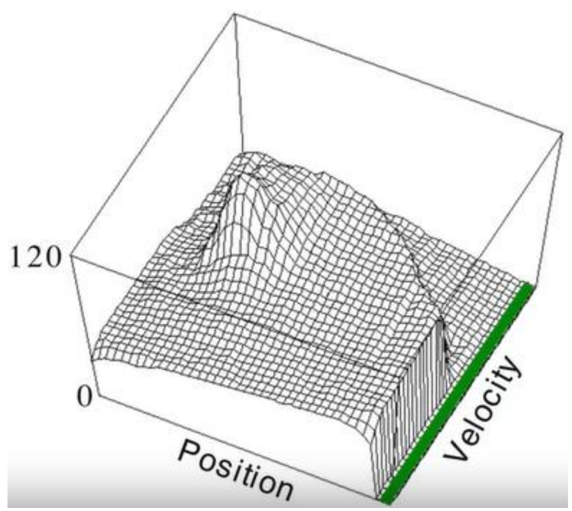
Policy-based RL Pros and Cons

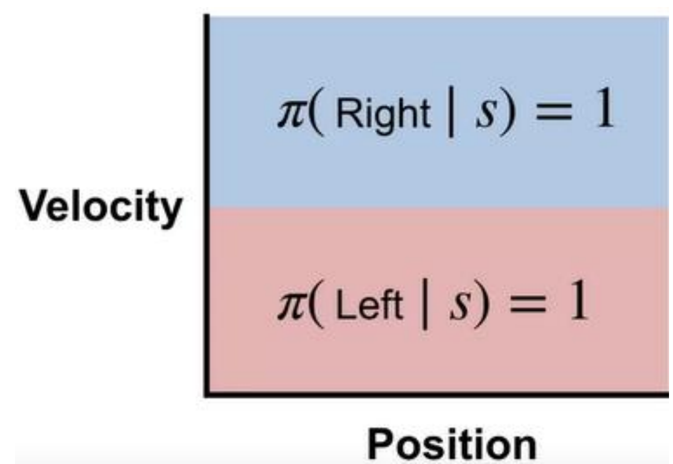
- Pros:
  - Effective in high-dimensional or continuous action space.
    - Value-based RL is only applicable to discrete action space; inefficient to discretize continuous actions for high-dim action space, as taking $\mathrm{argmax}_{\mathrm{a}}\, Q(s, a)$ may be expensive.
  - Can learn stochastic policies
    - Value-based RL learns a near-deterministic policy (greedy or ϵ-greedy).
    - For the game rock-paper-scissors, deterministic policy does not work well.
  - Better convergence properties.
- Disadvantages:
  - Typically converges to a local rather than global optimum.
  - Evaluating a policy is typically inefficient and high variance.

Mountain Car Example

- Middle: a complex value function
- Right: a simple policy that works well: accelerate in the direction of current velocity.

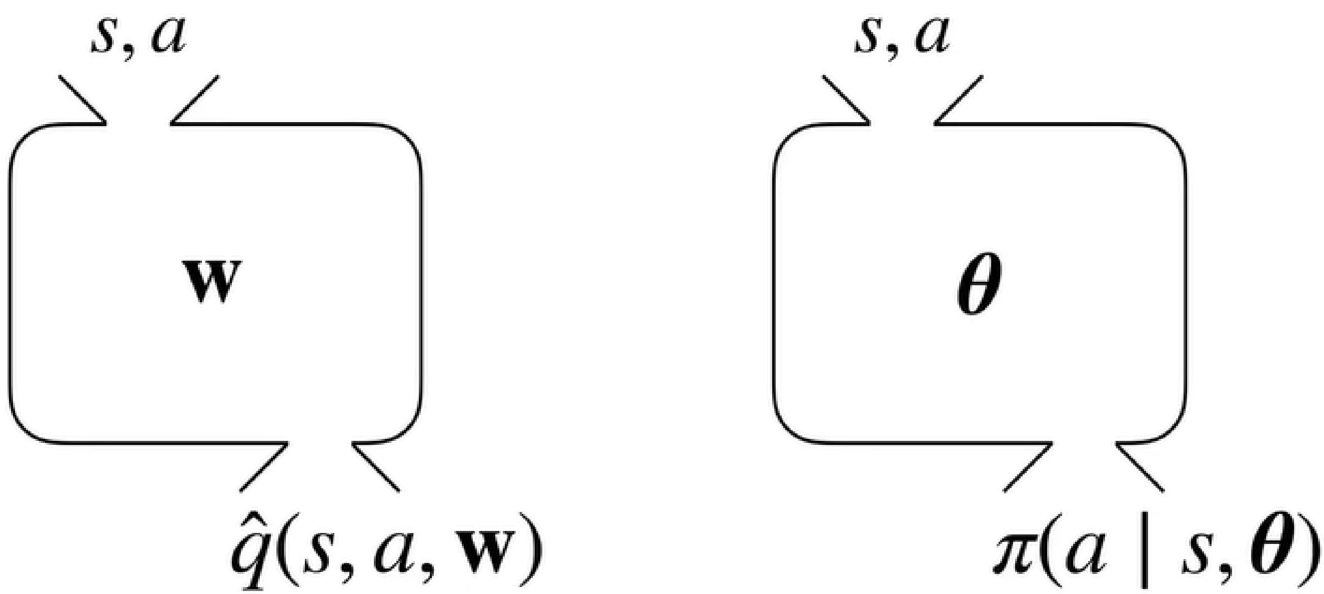






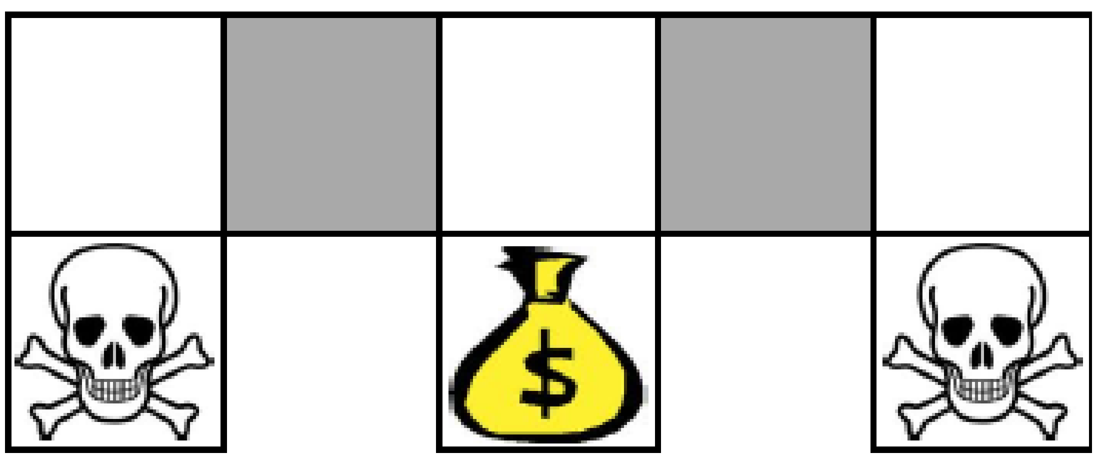
Function Approximation for Action Value Function vs. Policy

- Left: function approximation for action value function $\hat{q}(s, a, \mathbf{w})$.
- Right: function approximation for policy $\pi(a|s, \boldsymbol{\theta})$:
  - Probability that action a is taken in state s, with parameter $\boldsymbol{\theta}$.
  - $\pi(a|s, \boldsymbol{\theta}) \geq 0, \forall a \in \mathcal{A} \wedge \forall s \in \mathcal{S}$
  - $\sum_{a \in \mathcal{A}} \pi(a|s, \boldsymbol{\theta}) = 1, \forall s \in \mathcal{S}$



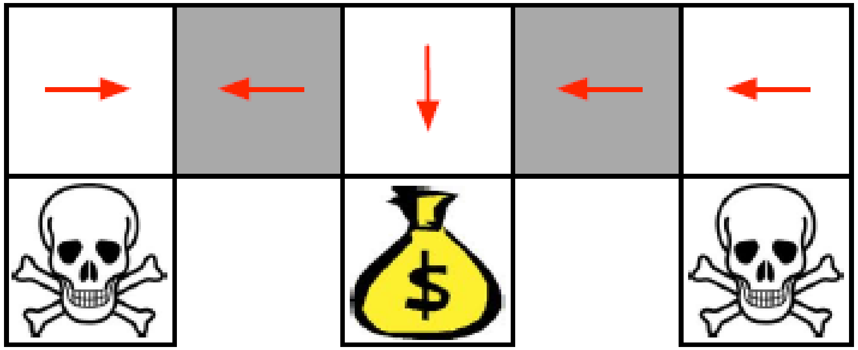
Example: Aliased Gridworld

- Agent cannot observe its position directly; it can only observe features of the following form (for all directions $d_1, d_2, \ldots \in (N, E, S, W)$):
  - $\phi(s) = \mathbf{1}(\text{wall to } d_1, d_2 \ldots)$ (if there are walls in each direction)
  - $\mathbf{1}(\boldsymbol{x})$ is indicator function: $\mathbf{1}(\boldsymbol{x}) = \mathbf{1}$ if $x = \text{true}$
  - Hence agent cannot differentiate between the 2 grey states (it is a Partially Observable MDP (POMDP))
- Value-based RL: $\hat{q}(s, a, \mathbf{w}) = f(\phi(s), \mathbf{w})$
- Policy-based RL: $\pi(a|s, \boldsymbol{\theta}) = g(\phi(s), \boldsymbol{\theta})$

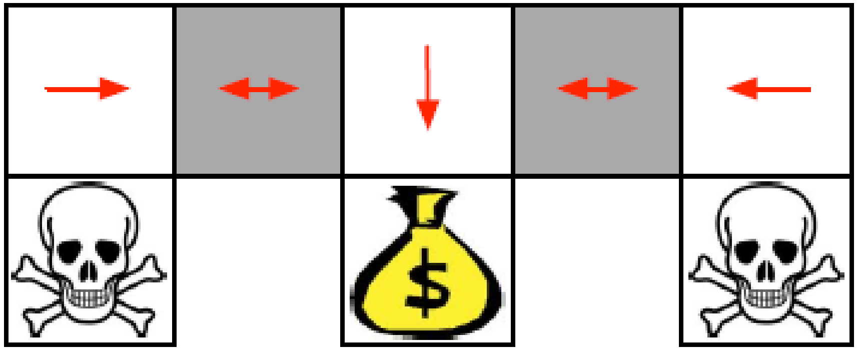


Example: Aliased Gridworld

- An optimal deterministic policy:
  - Either move W in both grey states (red arrows), or 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 (greedy or ϵ-greedy), So it may traverse the corridor for a long time.
- An optimal stochastic policy:
  - $\pi(\text{move } E \mid \text{wall to N and S}, \boldsymbol{\theta}) = \pi(\text{move } W \mid \text{wall to N and S}, \boldsymbol{\theta}) = 0.5$
  - Randomly move E or W in grey states.
- It will reach the goal state in a few steps with high probability



Opt det policy

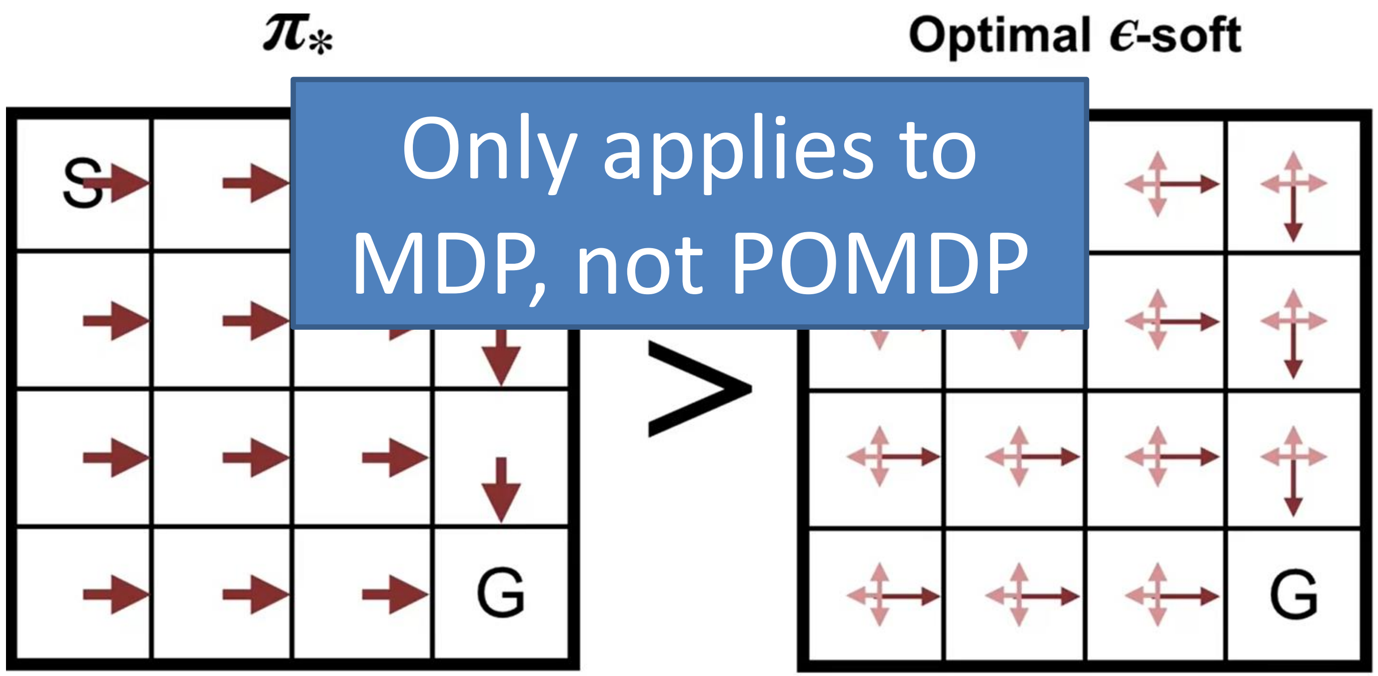


Opt Sto policy

Recall

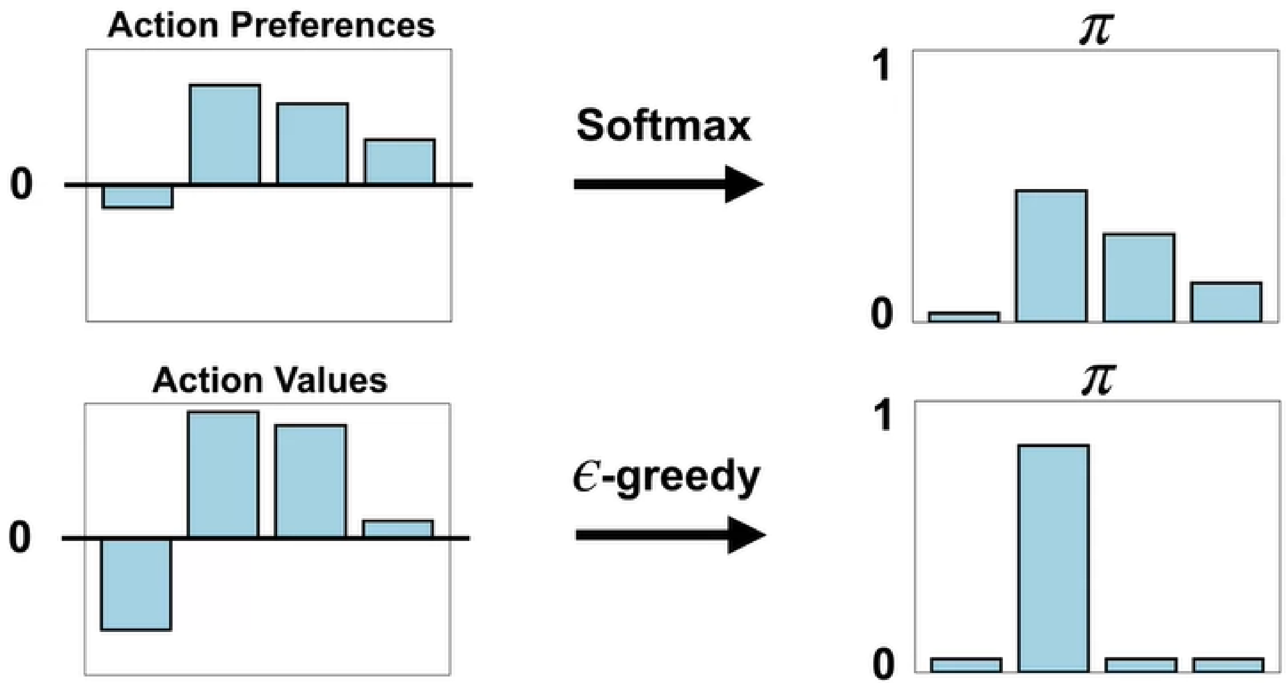
Optimal ϵ-Soft Policy

- The optimal ϵ-soft policy is the policy with the highest value in each state among all ϵ-soft policies. It performs worse than the optimal greedy deterministic policy π_* in general.
- But it often performs reasonably well, and avoids exploring starts.



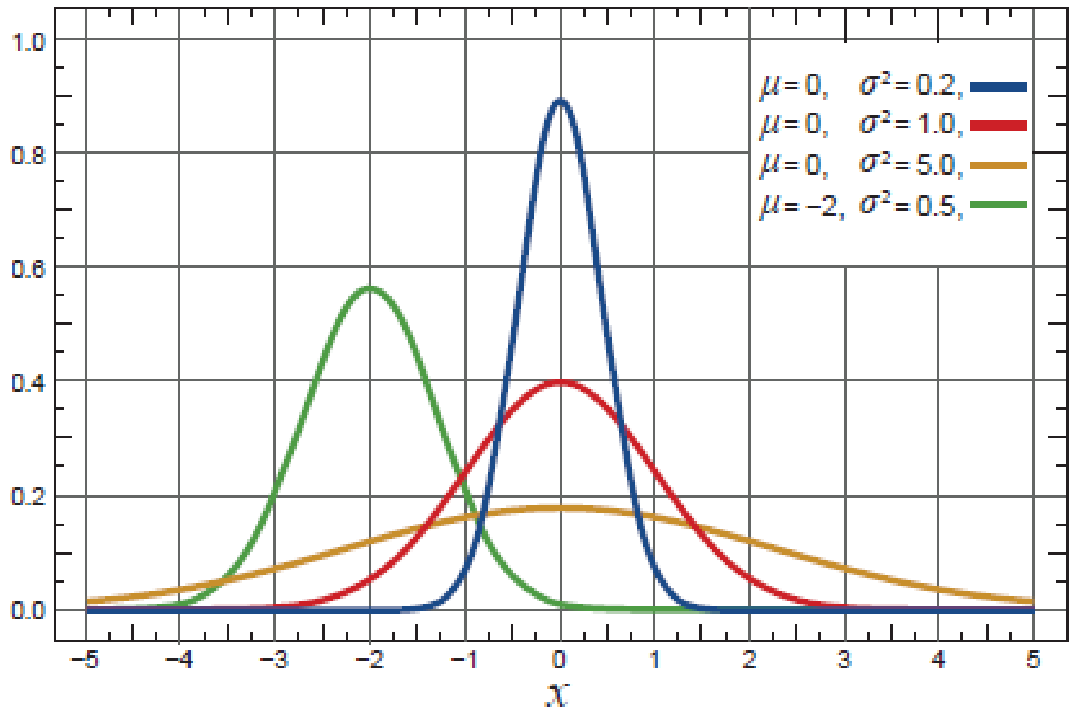
SoftMax Policy for Discrete Actions

- $\pi(a|s, \boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_{a\in\mathcal{A}} e^{h(s,a,\boldsymbol{\theta})}}$
  - $h(s, a, \boldsymbol{\theta})$ is action preference, which may be linear function $\boldsymbol{\theta}^T \mathbf{x}(s, a)$, or a Deep Neural Network.
- ϵ-greedy:
  - No distinction between policies that are not the optimal one.



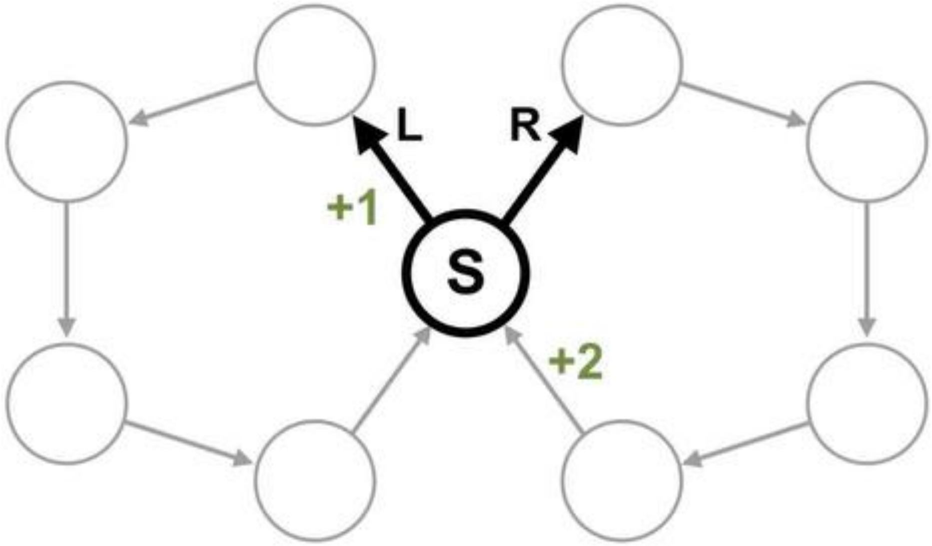
Gaussian Policy for Continuous Actions

- Gaussian Policy $\pi(a|s, \mathbf{\theta}) \doteq$
$$\frac{1}{\sigma(s,\mathbf{\theta})\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s,\mathbf{\theta})\right)^2}{2\sigma(s,\mathbf{\theta})^2}\right)$$
  - Mean $\mu(s, \mathbf{\theta})$, variance $\sigma(s, \mathbf{\theta})$



An Example MDP (Det Env)

- Always left policy π_L:
  - Geometric series: $V_{\pi_L}(S) = 1 + \gamma^5 + \gamma^{10} + \cdots = \frac{1}{1-\gamma^5}$
  - Recursion:$V_{\pi_L}(S) = 1 + \gamma\left(0 + \gamma\left(0 + \gamma\left(0 + \gamma\left(0 + \gamma V_L(s)\right)\right)\right)\right), V_L(S) = \frac{1}{1-\gamma^5}$
- Always right policy π_R:
  - Geometric series: $V_{\pi_R}(S) = 2\gamma^4 + 2\gamma^9 + \cdots = \frac{2\gamma^4}{1-\gamma^5}$
  - Recursion: $V_{\pi_R}(S) = 0 + \gamma\left(0 + \gamma\left(0 + \gamma\left(0 + \gamma\left(2 + \gamma V_R(s)\right)\right)\right)\right), V_R(S) = \frac{2\gamma^4}{1-\gamma^5}$
- Optimal policy depends on discount factor γ. $V_{\pi_L}(S) = V_{\pi_R}(R) \Rightarrow \gamma = 2^{-\frac{1}{4}} \approx 0.841$
- If the cycles are longer, say each with 100 states, then $\gamma = 2^{-\frac{1}{99}} \approx 0.993$
- Larger values of γ result in larger and more variables sums, which may be difficult to learn with RL.



γ	$V_{\pi_L}(S)$	$V_{\pi_R}(S)$	**Opt. Pol.**
0.5	≈ 1	≈ 0.1	π_L
0.841	≈ 0.58	≈ 0.58	π_L or π_R
0.9	≈ 2.4	≈ 3.2	π_R

Average Reward Objective

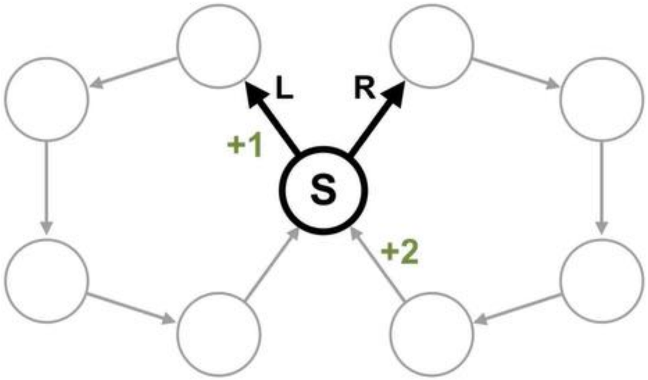
- Average Reward $r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] = \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) r$
  - $\sum_{s',r} p(s',r|s,a) r = \mathbb{E}[R_t | S_t = s, A_t = a]$: expected reward if we start in state s and take action a, taking exp over all possible s', r.
  - $\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) r = \mathbb{E}_\pi[R_t | S_t = s]$: expected reward under policy π from state s, taking exp over all possible a.
  - $r(\pi) = \mathbb{E}_\pi[R_t]$: average reward under policy π by weighting expected reward of each state s under policy π with $\mu_\pi(s)$, fraction of time spent in state s under policy π.
  - Measures "rate of reward" or "reward per timestep".
- Recall: Bellman Exp Equation for State Value Function w. cumulative discounted reward: $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_\pi(s')]$

Return vs. Differential Return

- Differential Return for continuing task: $G_t \doteq \sum_{k=0}^{\infty}(R_{t+k+1} - r(\pi))$
  - It measures how much better it is to take an action in a state than average reward $r(\pi)$ under a certain baseline policy π. It is used to compare actions if the same policy π is followed on subsequent time steps.
- c.f. return for regular MDP w. discount factor γ: return (discounted cumulative reward): $G_t \doteq \sum_{k=0}^{T-1} \gamma^k R_{t+k+1}$
  - Episodic task with finite T, or continuing task with $T \to \infty$

Differential Return

- Average reward $r(\pi_L) = 0.2$ (1 reward every 5 steps); $r(\pi_R) = 0.4$ (2 reward every 5 steps)
- For policy π_L:
  - Take L action, then follow π_L forever: $q_{\pi_L}(s, L) = G_t = 1 - .2 + 0 - .2 + 0 - .2 + 0 - .2 + 0 - .2 + \cdots = 0.4$ (Cesàro Sum)
  - Take R action, then follow π_L forever: $q_{\pi_L}(s, R) = G_t = 0 - .2 + 0 - .2 + 0 - .2 + 0 - .2 + 2 - .2 + .4 = 1.4$
  - Optimal action is R.
- For policy π_R:
  - Take R action, then follow π_R forever: $q_{\pi_R}(s, R) = G_t = 0 - .4 + 0 - .4 + 0 - .4 + 0 - .4 + 2 - .4 + \cdots = -0.8$ (Cesàro Sum)
  - Take L action, then follow π_R forever: $q_{\pi_R}(s, L) = G_t = 1 - .4 + 0 - .4 + 0 - .4 + 0 - .4 + 0 - .4 - .8 = -1.8$
  - Optimal action is R.



Side Note: Cesàro Summation

- Grandi's series: $G = \sum_{n=0}^{\infty} a_n = 1 - 1 + 1 - 1 + \cdots$
  - One view: $G = (1 - 1) + (1 - 1) + \cdots = 0$
  - Another view: $G = 1 + (-1 + 1) + (-1 + 1) + \cdots = 1$
  - Another view: $G = 1 - (1 - 1 + 1 - 1 + \cdots) = 1 - G \Rightarrow G = 0.5$
- Cesàro Summation:
  - Sequence of partial sums $s_k = \sum_{i=0}^{k} a_i$, $(s_k) = (1,0,1,0, \ldots)$
  - Cesàro Sum: $t_n = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} s_k = 0.5$

n	t_n
1	$\frac{1}{1} = 1$
2	$\frac{1+0}{2} = .5$
3	$\frac{1+0+1}{3} \approx .67$
4	$\frac{1+0+1+0}{4} = .5$
5	$\frac{1+0+1+0+1}{5} = .6$
6	$\frac{1+0+1+0+1+0}{6} = .5$
∞	$\approx .5$

Bellman Equations for Average Reward

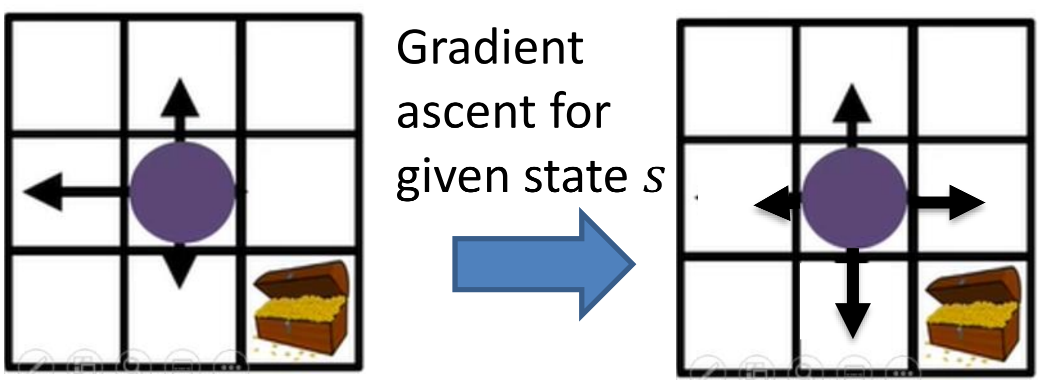
- Bellman Expectation Equations:
  - $v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a) \left[r - r(\pi) + v_\pi(s')\right]$
  - $q_\pi(s,a) = \sum_{r,s'} p(r,s'|s,a) \left[r - r(\pi) + \sum_{a'} \pi(a'|s')\, q_\pi(s',a')\right]$
- Bellman Optimality Equations:
  - $v_*(s) = \max_a \sum_{r,s'} p(r,s'|s,a) \left[r - \max_\pi r(\pi) + v_*(s')\right]$
  - $q_*(s,a) = \sum_{r,s'} p(r,s'|s,a) \left[r - \max_\pi r(\pi) + \max_{a'} q_*(s',a')\right]$
- Differences from Bellman Equations for discounted cumulative reward:
  - Remove γ, and replace all rewards by difference between the reward and the true average reward.
- Recall: Bellman Expectation Equations:
  - $v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a) \left[r + \gamma v_\pi(s')\right] =$
  - $q_\pi(s,a) = \sum_{r,s'} p(r,s'|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s',a')\right]$
- Recall: Bellman Optimality Equations:
  - $v_*(s) = \max_a \sum_{r,s'} p(r,s'|s,a) \left[r + \gamma v_*(s')\right]$
  - $q_*(s,a) = \sum_{r,s'} p(r,s'|s,a) \left[r + \gamma \max_{a'} q_*(s',a')\right]$

Policy Optimization w. Average Reward Objective

- $r(\pi) \doteq \sum_s \mu_\pi(s) \sum_a \pi(a|s, \boldsymbol{\theta}) \sum_{s',r} p(s', r|s, a) r$
- Gradient ascent to maximize $r(\pi)$:
  - $\nabla r(\pi) = \nabla \sum_s \mu_\pi(s) \sum_a \pi(a|s, \boldsymbol{\theta}) \sum_{s',r} p(s', r|s, a) r$
  - We cannot move the gradient ∇ inside the expectation over $\mu_\pi(s)$, since it depends on the policy params $\boldsymbol{\theta}$, i.e., modifying the policy $\pi(a|s, \boldsymbol{\theta})$ changes distribution $\mu_\pi(s)$.
- Recall gradient descent for optimizing value function under a fixed policy π:
  - $\nabla \overline{VE} = \nabla \sum_{s \in \mathcal{S}} \mu(s) [v_\pi(s) - \hat{v}(s, \mathbf{w})]^2 = \sum_{s \in \mathcal{S}} \mu(s) \nabla [v_\pi(s) - \hat{v}(s, \mathbf{w})]^2$
  - We can move the gradient ∇ inside the expectation over $\mu(s)$, since distribution $\mu(s)$ does not depend on the value params $\mathbf{w}$.

Policy Gradient Theorem

- $\nabla r(\pi) = \sum_s \mu_\pi(s) \sum_a \nabla\pi(a|s,\boldsymbol{\theta}) q_\pi(s,a)$
- For a given state s: gradient ascent maximizes average reward $\sum_a \pi(a|s,\boldsymbol{\theta}) q_\pi(s,a)$.
- For all states: gradient ascent maximizes overall average reward $r(\pi)$ by updating $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\nabla r(\pi)$.

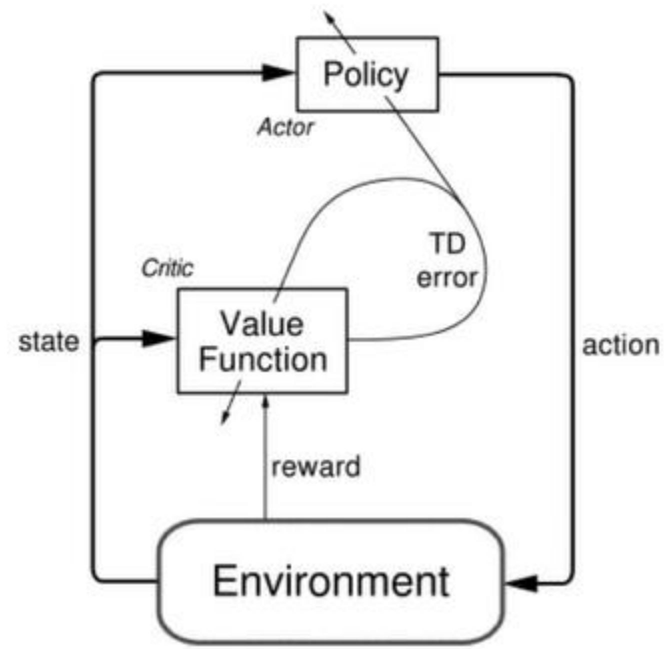


SGD for Policy Gradient

- $\nabla r(\pi) = \sum_s \mu_\pi(s) \sum_a \nabla \pi(a|s, \boldsymbol{\theta}) q_\pi(s, a)$
- $= \mathbb{E}_\pi[\sum_a \nabla \pi(a|S_t, \theta) q_\pi(S_t, a)]$
- $= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t, \boldsymbol{\theta}) \frac{\nabla \pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} q_\pi(S_t, a)\right]$
- $= \mathbb{E}_\pi[\nabla \log \pi(A_t|S_t, \boldsymbol{\theta}) q_\pi(S_t, A_t)]$ (replace a by sample $A_t \sim \pi$)
- SGD update:
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t|S_t, \boldsymbol{\theta}) q_\pi(S_t, A_t)$
  - $\nabla \log \pi(A_t|S_t, \boldsymbol{\theta})$: score function
  - $q_\pi(S_t, A_t)$: action value function

Actor-Critic Algorithm

- Instead of $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t|S_t, \boldsymbol{\theta}) q_\pi(S_t, A_t)$
- We use $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t|S_t, \boldsymbol{\theta})[R_{t+1} - \bar{R} + \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})] = \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t|S_t, \boldsymbol{\theta})\delta_t$
- We subtract the baseline reward $\hat{v}(S_t, w)$ to reduce update variance to get the TD error δ_t, since we only care about the relative ranking of different actions in state S_t. The baseline does not affect the expected update since $\mathbb{E}_\pi[\nabla \log \pi(A_t|S_t, \boldsymbol{\theta})\, \hat{v}(S_t, \mathbf{w})|S_t = s] = 0$
- After we execute an action A_t in state S_t, critic uses TD error δ_t to decide how good the action was compared to the average for that state $\hat{v}(S_t, \mathbf{w})$. If $\delta_t > 0$, then it means A_t resulted in a higher value than expected, so actor updates policy parameters $\boldsymbol{\theta}$ to increase the probability of A_t in state S_t; and vice versa if $\delta_t < 0$.
- Actor and Critic learn at the same time, constantly interacting. The actor is continually changing the policy params $\boldsymbol{\theta}$ to exceed the critics expectation, and the critic is constantly updating its value function params $\mathbf{w}$ to evaluate the actors changing policy.



Actor-Critic Algorithm

- TD error $\delta_t = R_{t+1} - \bar{R} + \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)$
- Critic: $w = w + \alpha^w \delta \nabla \hat{v}(S_t, w)$ (semi-gradient TD)
- Actor: $\theta = \theta + \alpha^\theta \delta \nabla \log \pi(A_t | S_t, \theta)$ (Policy Gradient)

Actor-Critic (continuing), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a \mid s, \boldsymbol{\theta})$

Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$

Initialize $\bar{R} \in \mathbb{R}$ to 0

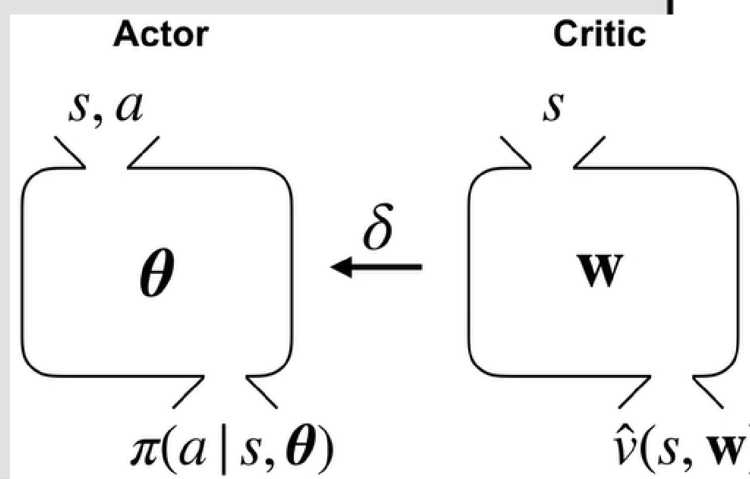
Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g. to 0)

Algorithm parameters: $\alpha^{\mathbf{w}} > 0, \alpha^{\boldsymbol{\theta}} > 0, \alpha^{\bar{R}} > 0$

Initialize $S \in \mathcal{S}$

Loop forever (for each time step):

- $A \sim \pi(\cdot \mid S, \boldsymbol{\theta})$
- Take action A, observe S', R
- $\delta \leftarrow R - \bar{R} + \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
- $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}} \delta$
- $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S, \mathbf{w})$
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \nabla \ln \pi(A \mid S, \boldsymbol{\theta})$
- $S \leftarrow S'$



Actor-Critic with Softmax Policies

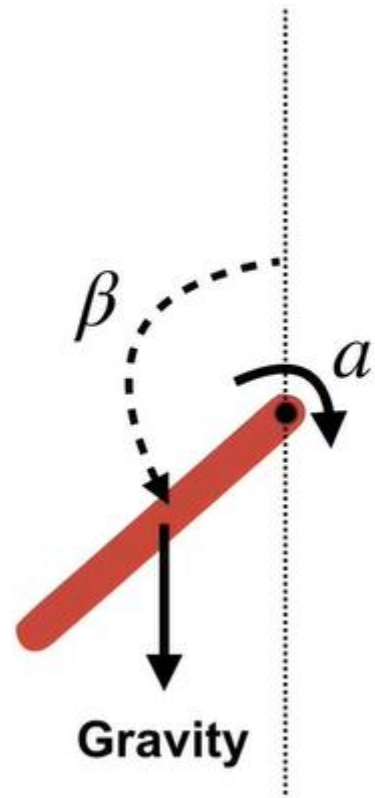
- Softmax policy $\pi(a|s,\boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_{b\in\mathcal{A}} e^{h(s,a,\boldsymbol{\theta})}}$
- Assume linear value function $\hat{v}(s,\mathbf{w}) \doteq \mathbf{w}^T\mathbf{x}(s)$ and linear preference function $h(s,a,\boldsymbol{\theta}) \doteq \boldsymbol{\theta}^T\mathbf{x}_h(s,a)$
- Critic update: $\mathbf{w} = \mathbf{w} + \alpha^w\delta\nabla\hat{v}(S_t,\mathbf{w}) = \mathbf{w} + \alpha^w\delta\mathbf{x}(s)$
- Actor update: $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha^\theta\delta\nabla\log\pi(A_t|S_t,\boldsymbol{\theta})$
  - $\log\pi(A_t|S_t,\boldsymbol{\theta}) = x_h(s,a) - \sum_b \pi(b|s,\boldsymbol{\theta})x_h(s,b)$

Many Variants of PG

- Original REINFORCE:
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t | S_t, \boldsymbol{\theta}) v_\pi(S_t)$
- Q Actor-Critic:
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t | S_t, \boldsymbol{\theta}) q_\pi(S_t, A_t)$
- Advantage Actor-Critic:
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t | S_t, \boldsymbol{\theta}) A_\pi(S_t, A_t)$
  - $A_\pi(S_t, A_t) = q_\pi(S_t, A_t) - v_\pi(S_t)$
- TD Actor-Critic:
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \log \pi(A_t | S_t, \boldsymbol{\theta}) \delta$
- Critic uses policy evaluation (e.g. MC or TD learning) to estimate $q_\pi(S_t, A_t), A_\pi(S_t, A_t)$

Pendulum Swingup

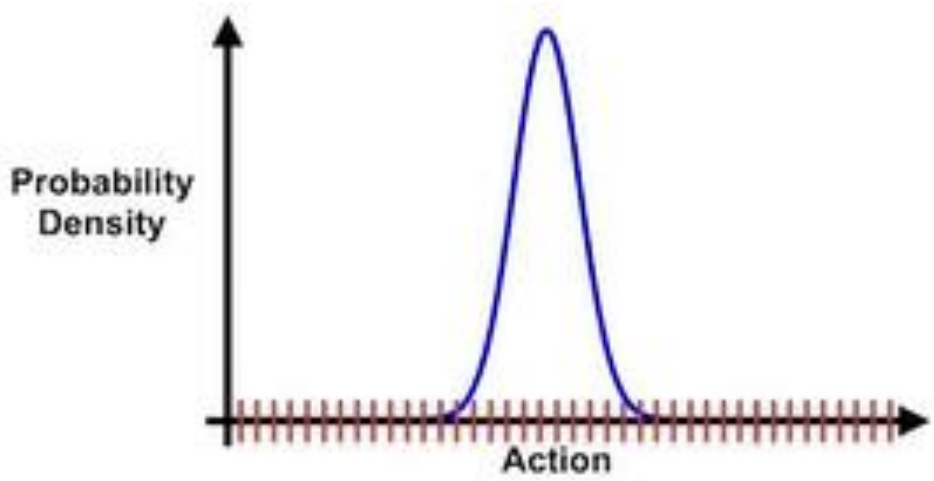
- Continuing task
- State $s \doteq \begin{bmatrix} \beta \\ \dot{\beta} \end{bmatrix}$
  - $\beta \in [-\pi, \pi]$: angular position;
  - $\dot{\beta} \in [-2\pi, 2\pi]$: angular velocity within user-specified range. If range is exceeded, it is reset to resting position $\beta = \pi$.
- Action $a \in \{-1,0,1\}$
- Reward $R \doteq -|\beta|$
  - Goal is to get the pendulum pointing directly up and keep it that way.



- For Critic: Value function $\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s)$
- For Actor: Preference function $h(s, a, \boldsymbol{\theta}) \doteq \boldsymbol{\theta}^T \mathbf{x}_h(s, a)$
- We update critic at faster rate than actor: $\alpha^{\boldsymbol{\theta}} < \alpha^{\mathbf{w}}$

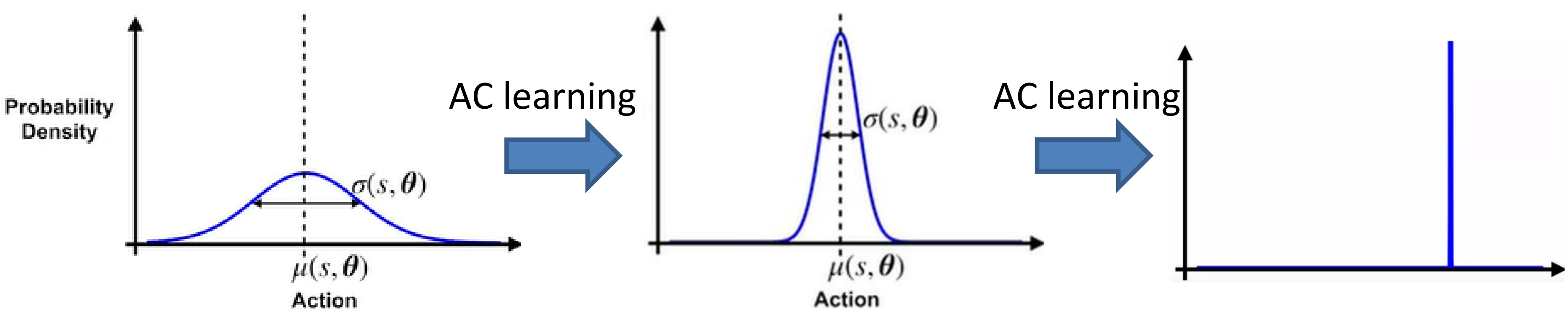
Advantages of Continuous Actions

- It might not be straightforward to choose a proper discrete set of actions
- Continuous actions allow us to generalize over actions
  - If an action is good, its neighboring actions are also likely to be good
  - Discrete actions lack generalization: each action is independent of others, including its neighbors (similar to value functions for discrete states)



Gaussian Policies for Continuous Actions

- Action $a \in [-3,3]$
- Gaussian Policy $\pi(a|s,\boldsymbol{\theta}) \doteq \frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}}\exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2\sigma(s,\boldsymbol{\theta})^2})$
  - Mean $\mu(s,\boldsymbol{\theta}) \doteq \boldsymbol{\theta}_\mu^T x(s)$ (assumed to be linear func)
  - Variance $\sigma(s,\boldsymbol{\theta}) \doteq \exp \boldsymbol{\theta}_\sigma^T x(s)$ (assumed to be exponential of linear func)

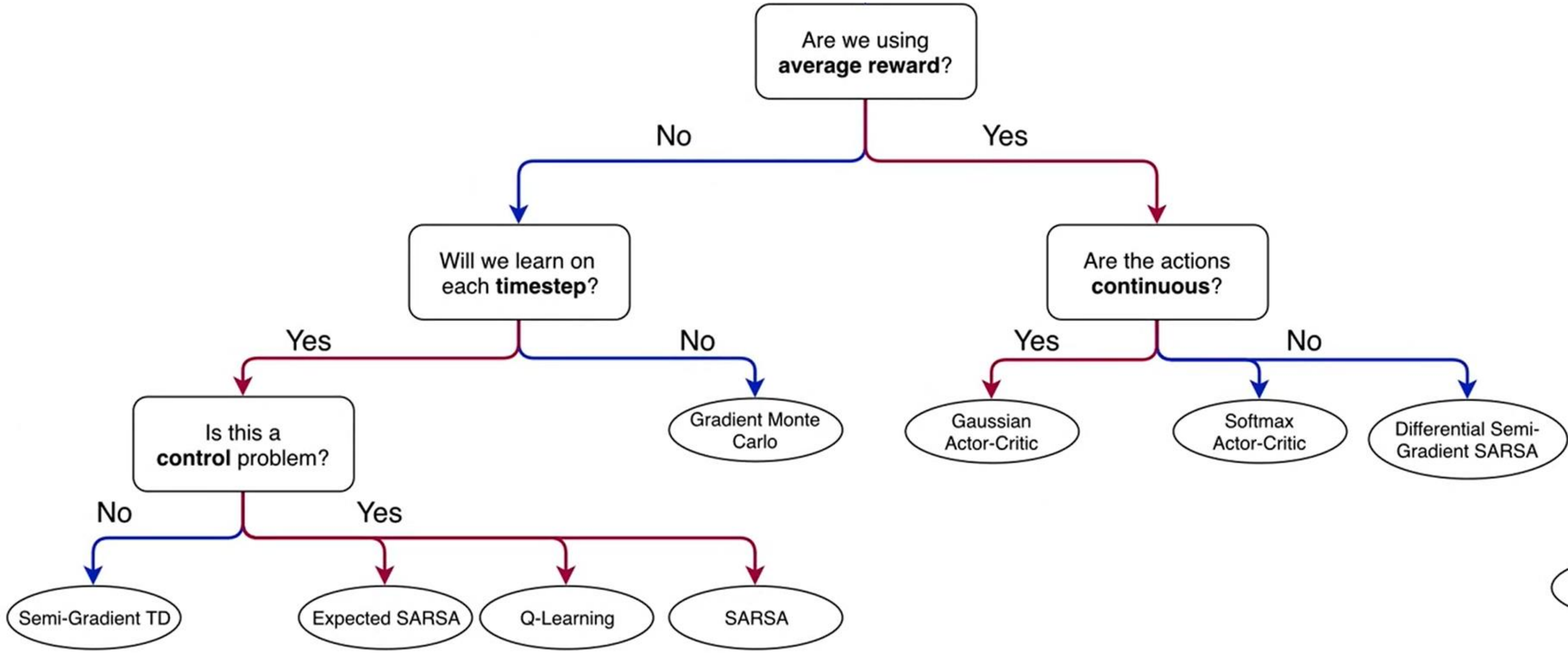


Policy variance $\sigma(s,\theta)$ initially large

$\sigma(s,\theta)$ gradually reduced during learning w. PG, converging towards deterministic policy

Taking Gradients of Log Policy

- $\pi(a|s, \boldsymbol{\theta}) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} \exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta}_\mu)\right)^2}{2\sigma(s,\boldsymbol{\theta}_\sigma)^2})$
- $\nabla \log \pi\left(a\middle|s, \boldsymbol{\theta}_\mu\right) = \frac{1}{\sigma(s,\theta)^2}\left(a - \mu(s, \boldsymbol{\theta})\right)x(s)$
- Proof: $\nabla \log \pi\left(a\middle|s, \boldsymbol{\theta}_\mu\right) = \frac{\nabla\pi\left(a\middle|s, \boldsymbol{\theta}_\mu\right)}{\pi\left(a\middle|s, \boldsymbol{\theta}_\mu\right)} = \frac{\frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}} \nabla \exp(-\frac{\left(a-\boldsymbol{\theta}_\mu^T x(s)\right)^2}{2\sigma(s,\boldsymbol{\theta})^2})}{\frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}} \exp(-\frac{\left(a-\boldsymbol{\theta}_\mu^T x(s)\right)^2}{2\sigma(s,\boldsymbol{\theta})^2})} = \frac{1}{\sigma(s,\boldsymbol{\theta})^2}\left(a - \mu(s, \boldsymbol{\theta})\right)x(s)$
- $\nabla \log \pi(a|s, \boldsymbol{\theta}_\sigma) = (\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{\sigma(s,\boldsymbol{\theta})^2} - 1)x(s)$
- Proof: $\nabla \log \pi(a|s, \boldsymbol{\theta}_\sigma) = \frac{\nabla\pi(a|s, \boldsymbol{\theta}_\sigma)}{\pi(a|s, \boldsymbol{\theta}_\sigma)} = \frac{\nabla \frac{1}{\exp \boldsymbol{\theta}_\sigma^T x(s)\sqrt{2\pi}} \exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2 \exp 2\boldsymbol{\theta}_\sigma^T x(s)})}{\frac{1}{\exp \boldsymbol{\theta}_\sigma^T x(s)\sqrt{2\pi}} \exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2 \exp 2\theta_\sigma^T x(s)})} = \frac{(-x(s) \exp -\boldsymbol{\theta}_\sigma^T x(s) + \exp -\boldsymbol{\theta}_\sigma^T x(s)\; x(s)\left(a-\mu(s,\boldsymbol{\theta})\right)^2 \exp -2\boldsymbol{\theta}_\sigma^T x(s)) \exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2 \exp 2\theta_\sigma^T x(s)})}{\exp -\boldsymbol{\theta}_\sigma^T x(s) \exp(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2 \exp 2\boldsymbol{\theta}_\sigma^T x(s)})} = (\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{\sigma(s,\boldsymbol{\theta})^2} - 1)x(s)$



Are we using average reward?
No
Yes
Will we learn on each timestep?
Are the actions continuous?
Yes
No
Yes
No
Is this a control problem?
Gradient Monte Carlo
Gaussian Actor-Critic
Softmax Actor-Critic
Differential Semi-Gradient SARSA
No
Yes
Semi-Gradient TD
Expected SARSA
Q-Learning
SARSA