Introduction to Reinforcement Learning
Week 7

Dr. Apurva Narayan
http://anarayan.com

Outline

- Stochastic policy gradient
  - REINFORCE algorithm
- AlphaGo

Model-free Policy-based Methods

- Q-learning
  - **Model-free value-based method**
  - No explicit policy representation

- Policy gradient
  - **Model-free policy-based method**
  - No explicit value function representation

Stochastic Policy

- Consider stochastic policy $\pi_\theta(a|s) = \Pr(a|s;\theta)$ parametrized by θ.

- Finitely many discrete actions

  Softmax: $\pi_\theta(a|s) = \frac{\exp(h(s,a;\theta))}{\sum_{a'} \exp(h(s,a';\theta))}$

  where $h(s,a;\theta)$ might be

  linear in θ: $h(s,a;\theta) = \sum_i \theta_i f_i(s,a)$

  or **non-linear** in θ: $h(s,a;\theta) = neuralNet(s,a;\theta)$

- Continuous actions:

  Gaussian: $\pi_\theta(a|s) = N(a|\mu(s;\theta), \Sigma(s;\theta))$

Supervised Learning

- Consider a stochastic policy $\pi_\theta(a|s)$
- Data: state-action pairs $\{(s_1, a_1^*), (s_2, a_2^*), \dots\}$
- Maximize log likelihood of the data

$$\theta^* = argmax_\theta \sum_n \log \pi_\theta(a_n^*|s_n)$$

- Gradient update

$$\theta_{n+1} \leftarrow \theta_n + \alpha_n \nabla_\theta \log \pi_\theta(a_n^*|s_n)$$

Reinforcement Learning

- Consider a stochastic policy$\pi_\theta(a|s)$

- Data: state-action-reward triples
  $\{(s_1, a_1, r_1), (s_2, a_2, r_2), \dots\}$

- Maximize discounted sum of rewards
  $$\theta^* = argmax_\theta \sum_n \gamma^n E_\theta[r_n | s_n, a_n]$$
- Gradient update
  $$\theta_{n+1} \leftarrow \theta_n + \alpha_n \gamma^n G_n \nabla_\theta \log \pi_\theta(a_n | s_n)$$
  where $G_n = \sum_{t=0}^{\infty} \gamma^t r_{n+t}$

Stochastic Gradient Policy Theorem

- Stochastic Gradient Policy Theorem

$$\boxed{\nabla V_\theta(s_0) \propto \sum_s \mu_\theta(s) \sum_a \nabla\, \pi_\theta(a|s)\, Q_\theta(s,a)}$$

$\mu_\theta(s)$: stationary state distribution when executing policy parametrized by θ

$Q_\theta(s,a)$: discounted sum of rewards when starting in s, executing a and following the policy parametrized by θ thereafter.

Derivation

$$\nabla V_\theta(s_0) = \nabla\left[\sum_{a_0} \pi_\theta(a_0|s_0)\, Q_\theta(s_0, a_0)\right] \qquad \forall s_0 \in S$$

$$= \sum_{a_0} [\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0, a_0) + \pi_\theta(a_0|s_0)\, \nabla Q_\theta(s_0, a_0)]$$

$$= \sum_{a_0}\left[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0, a_0) + \pi_\theta(a_0|s_0)\, \nabla \sum_{s_1, r_0} \Pr(s_1, r_0|s_0, a_0)\left(r_0 + \gamma V_\theta(s_1)\right)\right]$$

$$= \sum_{a_0}\left[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0, a_0) + \pi_\theta(a_0|s_0) \sum_{s_1} \gamma \Pr(s_1|s_0, a_0)\, \nabla \mathrm{V}_\theta(s_1)\right]$$

$$= \sum_{a_0} [\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0, a_0) + \pi_\theta(a_0|s_0) \sum_{s_1} \gamma \Pr(s_1|s_0, a_0)$$

$$\sum_{a_1} [\nabla\, \pi_\theta(a_1|s_1)\, Q_w(s_1, a_1) + \pi_\theta(a_1|s_1) \sum_{s_2} \gamma \Pr(s_2|s_1, a_1)\, \nabla \mathrm{V}_\theta(s_2)]$$

$$= \sum_{s \in S} \sum_{n=0}^{\infty} \gamma^n \underbrace{\Pr(s_0 \to s; n, \theta)}_{} \sum_a \nabla\, \pi_\theta(a|s)\, Q_\theta(s, a)$$

Probability of reaching s from s_0 at time step n

Since $\mu_\theta(s) \propto \sum_{n=0}^{\infty} \gamma^n \Pr(s_0 \to s; n, \theta)$ then

$$\propto \sum_s \mu_\theta(s) \sum_a \nabla\, \pi_\theta(a|s)\, Q_\theta(s, a)$$

REINFORCE: Monte Carlo Policy Gradient

- $\nabla V_\theta(s_0) = \sum_{s \in S} \sum_{n=0}^{\infty} \gamma^n \Pr(s_0 \to s; n, \theta) \sum_a \nabla \pi_\theta(a|s)\, Q_\theta(s,a)$

$$
\begin{aligned}
&= E_\theta\left[\sum_{n=0}^{\infty} \gamma^n \sum_a Q_\theta(S_n, a) \nabla \pi_\theta(a|S_n)\right] \\
&= E_\theta\left[\sum_{n=0}^{\infty} \gamma^n \sum_a \pi_\theta(a|S_n)\, Q_\theta(S_n, a)\, \frac{\nabla \pi_\theta(a|S_n)}{\pi_\theta(a|S_n)}\right] \\
&= E_\theta\left[\sum_{n=0}^{\infty} \gamma^n\, Q_\theta(S_n, A_n)\, \frac{\nabla \pi_\theta(A_n|S_n)}{\pi_\theta(A_n|S_n)}\right] \\
&= E_\theta\left[\sum_{n=0}^{\infty} \gamma^n\, G_n\, \frac{\nabla \pi_\theta(A_n|S_n)}{\pi_\theta(A_n|S_n)}\right] \\
&= E_\theta\left[\sum_{n=0}^{\infty} \gamma^n\, G_n \nabla \log \pi_\theta(A_n|S_n)\right]
\end{aligned}
$$

- Stochastic gradient at time step n

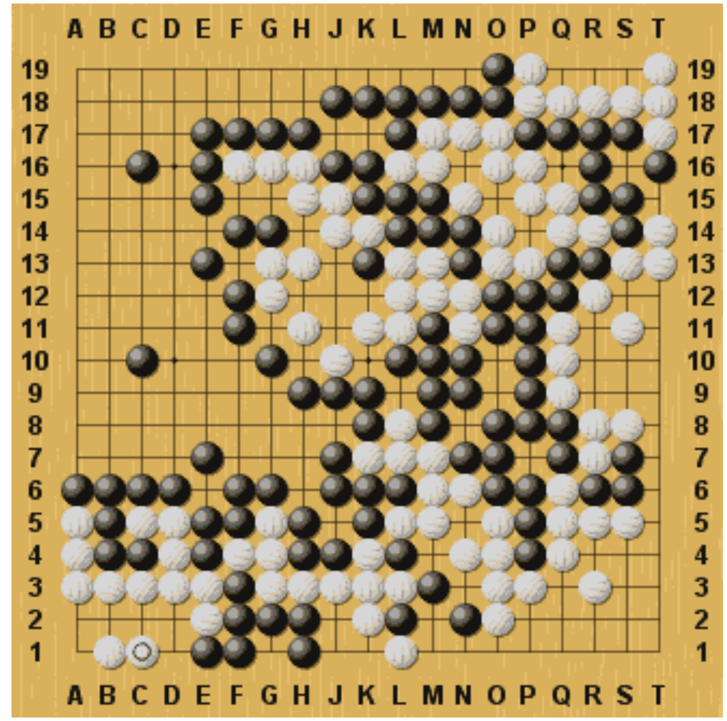
$$\nabla V_\theta \approx \gamma^n G_n \nabla \log \pi_\theta(a_n|s_n)$$

REINFORCE Algorithm (stochastic policy)

REINFORCE(s_0, π_θ)
Initialize π_θ to anything
Loop forever (for each episode)
Generate episode $\mathrm{s}_0, \mathrm{a}_0, \mathrm{r}_0, \mathrm{s}_1, \mathrm{a}_1, r_1, \dots, s_T, a_T, r_T$ with π_θ
Loop for each step of the episode $n = 0, 1, \dots, T$
$G_n \leftarrow \sum_{t=0}^{T-n} \gamma^t \, r_{n+t}$
Update policy: $\theta \leftarrow \theta + \alpha \, \gamma^n G_n \nabla \log \pi_\theta(a_n | s_n)$
Return π_θ

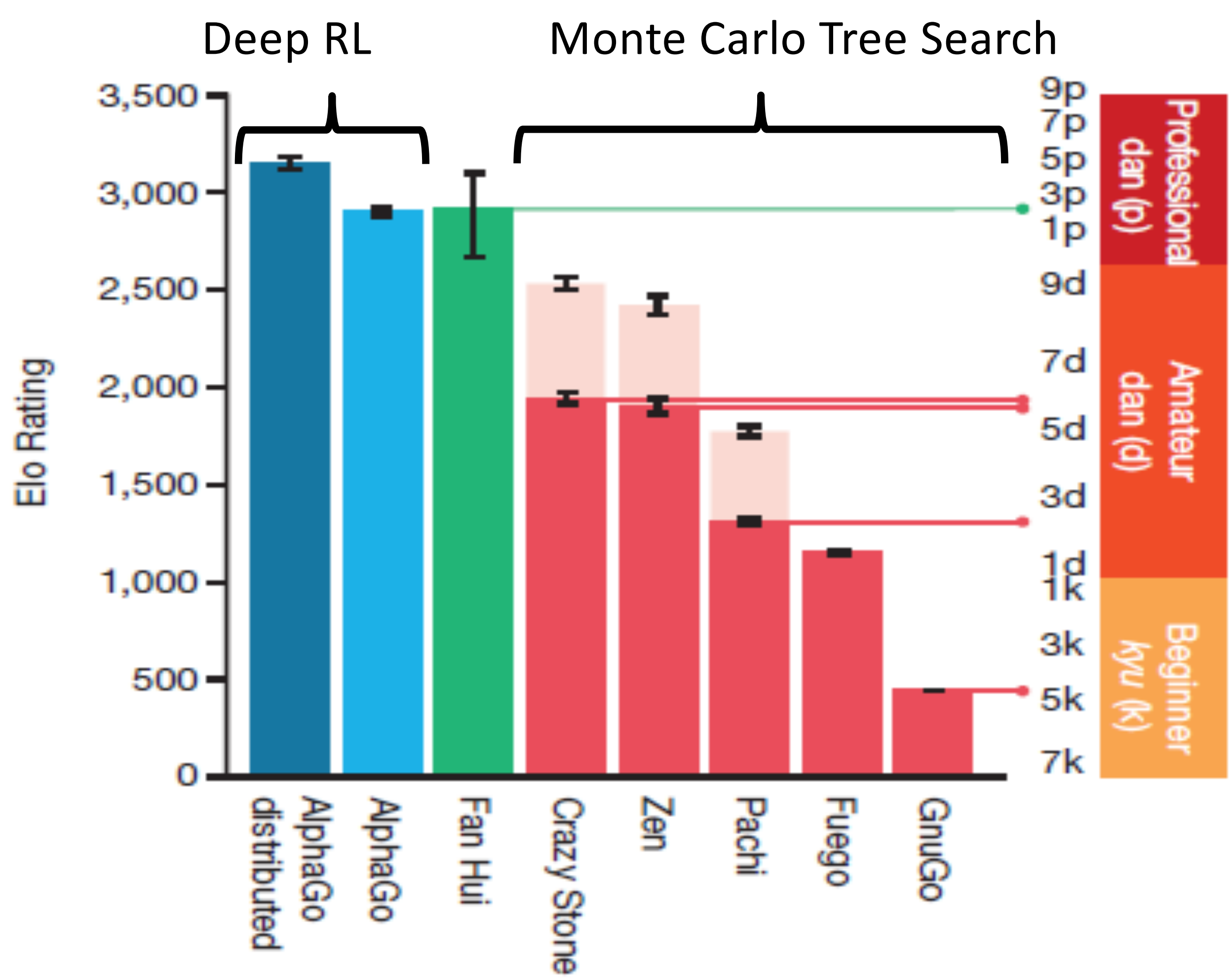
Example: Game of Go

- (simplified) rules:
  - Two players (black and white)
  - Players alternate to place a stone of their color on a vacant intersection.
  - Connected stones without any liberty (i.e., no adjacent vacant intersection) are captured and removed from the board
  - Winner: player that controls the largest number of intersections at the end of the game



Computer Go

- Oct 2015:



Computer Go

- March 2016: AlphaGo defeats Lee Sedol (9-dan)

  “[AlphaGo] can’t beat me” Ke Jie (world champion)

- May 2017: AlphaGo defeats Ke Jie (world champion)

  “Last year, [AlphaGo] was still quite humanlike when it played. But this year, it became like a god of Go” Ke Jie (world champion)

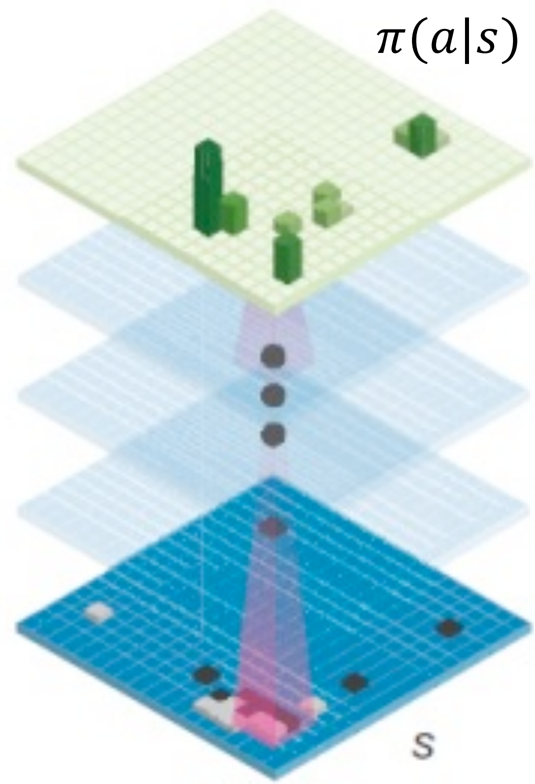
Winning Strategy

Four steps:

1. Supervised Learning of Policy Networks
2. Policy gradient with Policy Networks
3. Value gradient with Value Networks
4. Searching with Policy and Value Networks

Policy Network

- Train policy network to imitate Go experts based on a database of 30 million board configurations from the KGS Go Server.
- Policy network: $\pi(a|s)$
  - Input: state s (board configuration)
  - Output: distribution over actions a (intersection on which the next stone will be placed)



Supervised Learning of the Policy Network

- Let θ be the weights of the policy network

- Training:
  - Data: suppose a is optimal in s
  - Objective: maximize $\log \pi_\theta(\mathrm{a}|\mathrm{s})$
  - Gradient: $\nabla\theta = \frac{\partial \log \pi_\theta(a|s)}{\partial\theta}$
  - Weight update: $\theta \leftarrow \theta + \alpha\nabla\theta$

Policy gradient for the Policy Network

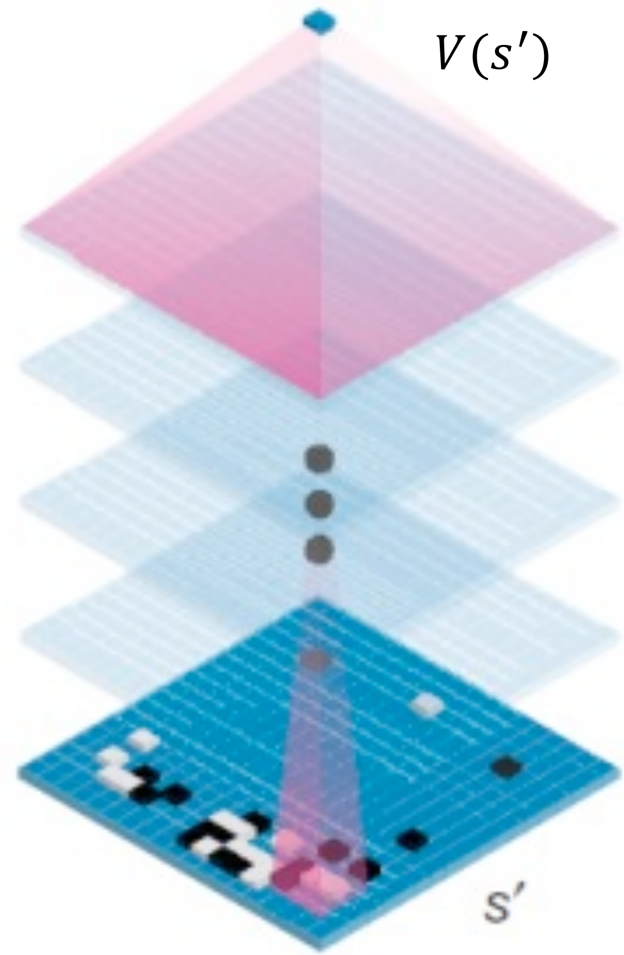
- How can we update a policy network based on reinforcements instead of the optimal action?
- Let $G_n = \sum_t \gamma^t \, r_{n+t}$ be the discounted sum of rewards in a trajectory that starts in s at time n by executing a.
- Gradient: $\nabla\theta = \frac{\partial \; log \; \pi_\theta(a|s)}{\partial\theta} \, \gamma^n G_n$
  - Intuition rescale supervised learning gradient by G_n
- Policy update: $\theta \leftarrow \theta + \alpha\nabla\theta$

Policy gradient for the Policy Network

- In computer Go, program repeatedly plays games against its former self.
- For each game $G_n = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
- For each (s_n, a_n) at turn n of the game, assume $\gamma = 1$ and compute
  - Gradient: $\nabla\theta = \frac{\partial \, log \, \pi_\theta(a|s)}{\partial\theta} \gamma^n G_n$
  - Policy update: $\theta \leftarrow \theta + \alpha\nabla\theta$

Value Network

- Predict $V(s')$ (i.e., who will win game) in each state s' with a value network
  - Input: state s (board configuration)
  - Output: expected discounted sum of rewards $V(s')$



Gradient Value Learning with Value Networks

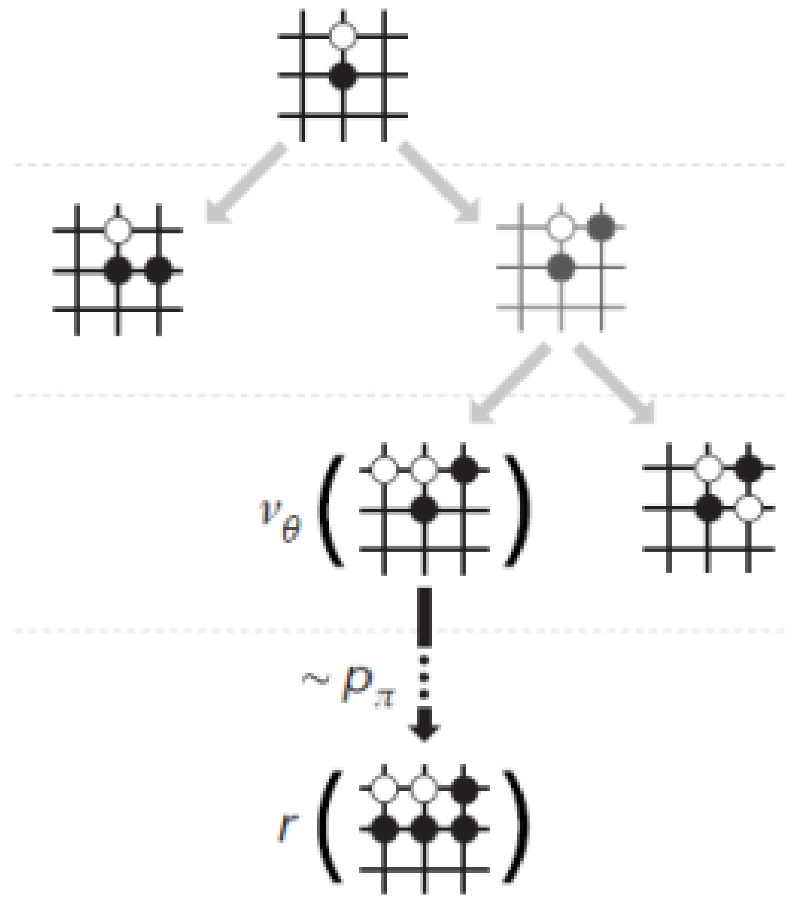
- Let $\boldsymbol{w}$ be the weights of the value network
- Training:
  - Data: (s, G) where $G = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
  - Objective: minimize $\frac{1}{2}(V_{\boldsymbol{w}}(s) - G)^2$
  - Gradient: $\nabla \boldsymbol{w} = \frac{\partial V_{\boldsymbol{w}}(s)}{\partial \boldsymbol{w}}(V_{\boldsymbol{w}}(s) - G)$
  - Weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla \boldsymbol{w}$

Searching with Policy and Value Networks

- AlphaGo combines policy and value networks into a **Monte Carlo Tree Search (MCTS)** algorithm

- Idea: construct a search tree
  - Node: s
  - Edge: a

- We will discuss MCTS in a few lectures



Competition

Extended Data Table 1 | Details of match between AlphaGo and Fan Hui

Date	Black	White	Category	Result
5/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by 2.5 points
5/10/15	Fan Hui	*AlphaGo*	Informal	Fan Hui wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Informal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
9/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
9/10/15	*AlphaGo*	Fan Hui	Informal	Fan Hui wins by resignation

The match consisted of five formal games with longer time controls, and five informal games with shorter time controls. Time controls and playing conditions were chosen by Fan Hui in advance of the match.

Next Phase

- Policy gradient with a baseline
- Actor Critic algorithms
- Deterministic policy gradient

Actor Critic

- Q-learning
  - **Model-free value-based method**
  - No explicit policy representation
- Policy gradient
  - **Model-free policy-based method**
  - No explicit value function representation
- Actor Critic
  - **Model-free policy and value based method**

Stochastic Gradient Policy Theorem

- Stochastic Gradient Policy Theorem

$$\boxed{\nabla V_\theta(s_0) \propto \sum_s \mu_\theta(s) \sum_a \nabla \pi_\theta(a|s)\, Q_\theta(s,a)}$$

- Equivalent Stochastic Gradient Policy Theorem with a baseline $b(s)$

$$\boxed{\nabla V_\theta(s_0) \propto \sum_s \mu_\theta(s) \sum_a \nabla \pi_\theta(a|s)\, [Q_\theta(s,a) - b(s)]}$$

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

Baseline

- Baseline often chosen to be b(s) $\approx V^{\pi}(s)$
- Advantage function: $A(s,a) = Q(s,a) - V^{\pi}(s)$
- Gradient update:
$$\theta \leftarrow \theta + \alpha\, \gamma^{n} A(s_n, a_n) \nabla \log \pi_{\theta}(a_n | s_n)$$
- Benefit: faster empirical convergence

REINFORCE Algorithm with a baseline

REINFORCEwithBaseline(s_0, π_θ)

  Initialize π_θ to anything

  Initialize V_w to anything

  Loop forever (for each episode)

    Generate episode $\mathrm{s}_0, \mathrm{a}_0, \mathrm{r}_0, \mathrm{s}_1, \mathrm{a}_1, r_1, \dots, s_T, a_T, r_T$ with π_θ

    Loop for each step of the episode $n = 0, 1, \dots, T$

      $G_n \leftarrow \sum_{t=0}^{T-n} \gamma^t r_{n+t}$

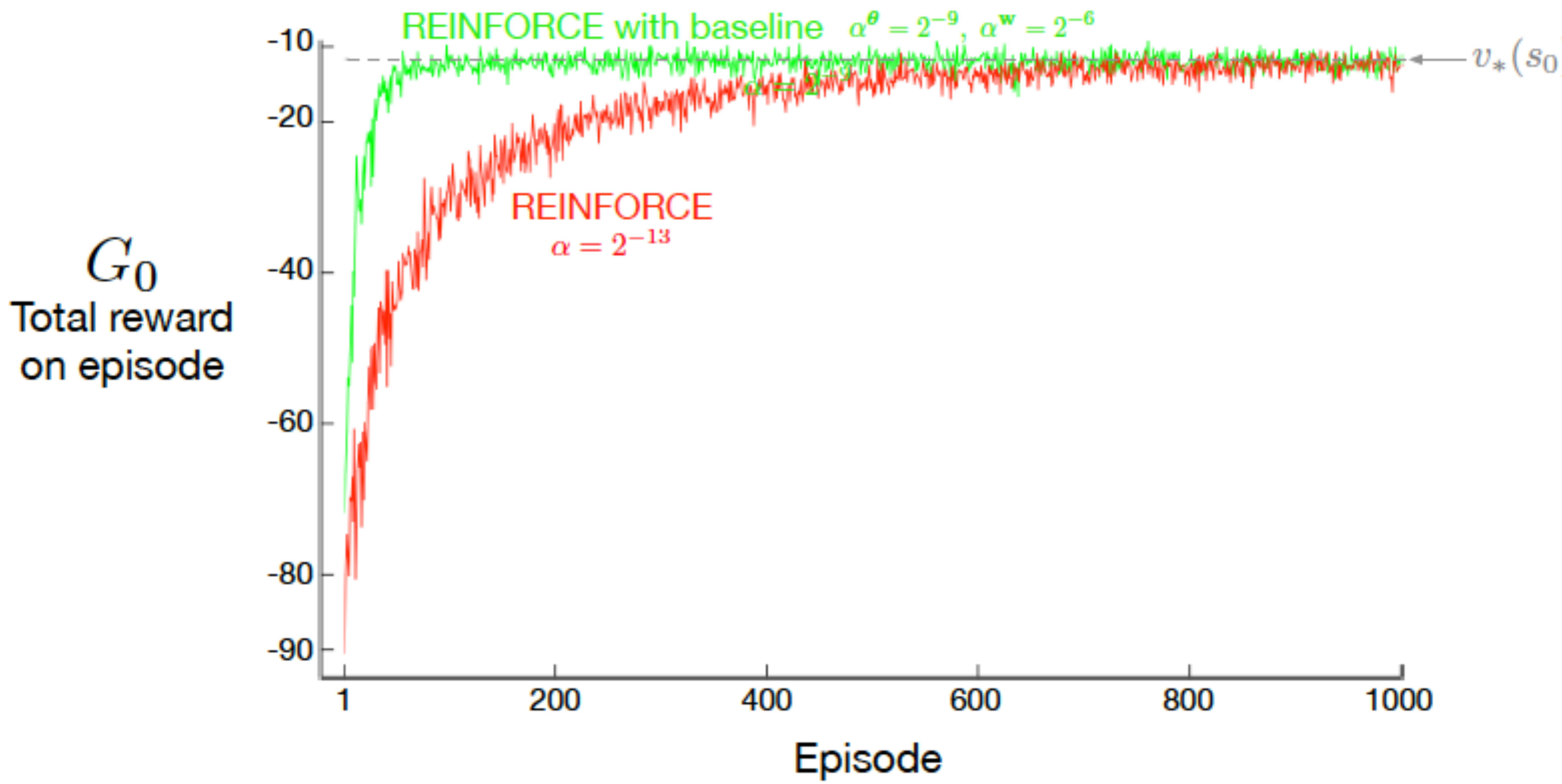
      $\delta \leftarrow G_n - V_w(s_n)$

      Update value function: $w \leftarrow w + \alpha_w \gamma^n \delta \nabla V_w(s_n)$

      Update policy: $\theta \leftarrow \theta + \alpha_\theta \gamma^n \delta \nabla \log \pi_\theta(a_n|s_n)$

Return π_θ

Performance Comparison



Temporal difference update

- Instead of updating V(s) by Monte Carlo sampling

$$\delta \leftarrow G_n - V_w(s_n)$$

Bootstrap with temporal difference updates

$$\delta \leftarrow r_n + \gamma V_w(s_{n+1}) - V_w(s_n)$$

- Benefit: reduced variance (faster convergence)

Actor Critic Algorithm

ActorCritic(s_0, π_θ)

Initialize π_θ to anything

Initialize Q_w to anything

Loop forever (for each episode)

Initialize s_0 and set $n \leftarrow 0$

Loop while s is not terminal (for each time step n)

Sample $a_n \sim \pi_\theta(a|s_n)$

Execute a_n, observe s_{n+1}, r_n

$\delta \leftarrow r_n + \gamma V_w(s_{n+1}) - V_w(s_n)$

Update value function: $w \leftarrow w + \alpha_w \gamma^n \delta \nabla V_w(s_n)$

Update policy: $\theta \leftarrow \theta + \alpha_\theta \gamma^n \delta \nabla \log \pi_\theta(a_n|s_n)$

$n \leftarrow n + 1$

Return π_θ

Advantage update

- Instead of doing temporal difference updates
$$\delta \leftarrow r_n + \gamma V_w(s_{n+1}) - V_w(s_n)$$
- Update with the advantage function
$$A(s_n, a_n) \leftarrow r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - \sum_a \pi_\theta(a|s_n) Q(s_n, a)$$
$$\theta \leftarrow \theta + \alpha_\theta \gamma^n A(s_n, a_n) \nabla \log \pi_\theta(a_n|s_n)$$
- Benefit: faster convergence

Advantage Actor Critic (A2C)

A2C()

Initialize π_θ to anything

Loop forever (for each episode)

Initialize s_0 and set $n \leftarrow 0$

Loop while s is not terminal (for each time step n)

Select a_n

Execute a_n, observe s_{n+1}, r_n

$\delta \leftarrow r_n + \gamma \max_{a_{n+1}} Q_w(s_{n+1}, a_{n+1}) - Q_w(s_n, a_n)$

$A(s_n, a_n) \leftarrow r_n + \gamma \max_{a_{n+1}} Q_w(s_{n+1}, a_{n+1})$
$\quad - \sum_a \pi_\theta(a|s_n) Q_w(s_n, a)$

Update Q: $w \leftarrow w + \alpha_w \gamma^n \delta \, \nabla_w Q_w(s_n, a_n)$

Update π: $\theta \leftarrow \theta + \alpha_\theta \gamma^n A(s_n, a_n) \nabla \log \pi_\theta(a_n|s_n)$

$n \leftarrow n + 1$

Continuous Actions

- Consider a deterministic policy $\pi_\theta(s) \rightarrow a$

- Deterministic Gradient Policy Theorem

$$\boxed{\nabla V_\theta(s_0) \propto E_{s\sim\mu_\theta(s)}\left[\nabla_\theta \pi_\theta(s) \nabla_{\mathrm{a}} Q_\theta(s,a)\,\Big|_{a=\pi_\theta(s)}\right]}$$

Proof: see Silver et al. 2014

- Stochastic Gradient Policy Theorem

$$\boxed{\nabla V_\theta(s_0) \propto \sum_s \mu_\theta(s) \sum_a \nabla_\theta\, \pi_\theta(a|s)\, Q_\theta(s,a)}$$

Deterministic Policy Gradient (DPG)

DPG(s, π_θ)

Initialize π_θ to anything

Loop forever (for each episode)

Initialize s_0 and set $n \leftarrow 0$

Loop while s is not terminal (for each time step n)

Select $a_n = \pi_\theta(s_n)$

Execute a_n, observe s_{n+1}, r_n

$\delta \leftarrow r_n + \gamma Q_w\left(s_{n+1}, \pi_\theta(s_{n+1})\right) - Q_w(s_n, a_n)$

Update Q: $w \leftarrow w + \alpha_w \gamma^n \delta \, \nabla_w Q_w(s_n, a_n)$

Update π: $\theta \leftarrow \theta + \alpha_\theta \gamma^n \, \nabla_\theta \, \pi_\theta(s_n) \, \nabla_a Q_w(s_n, a_n)|_{a_n = \pi_\theta(s_n)}$

$n \leftarrow n + 1$

Return π_θ