澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Reinforcement Learning

GongYing
gong.ying@connect.umac.mo
2023.12.08

Contents

- **Policy-based reinforcement learning**

  Policy gradient

  REINFORCE & actor-critic

- **Policy gradient with baseline**

  REINFORCE with baseline

  A2C: advantage actor-critic

- **Advanced techniques in policy-based RL**

  TRPO: trust region policy optimization

  Entropy regularization

- **Partial observations**

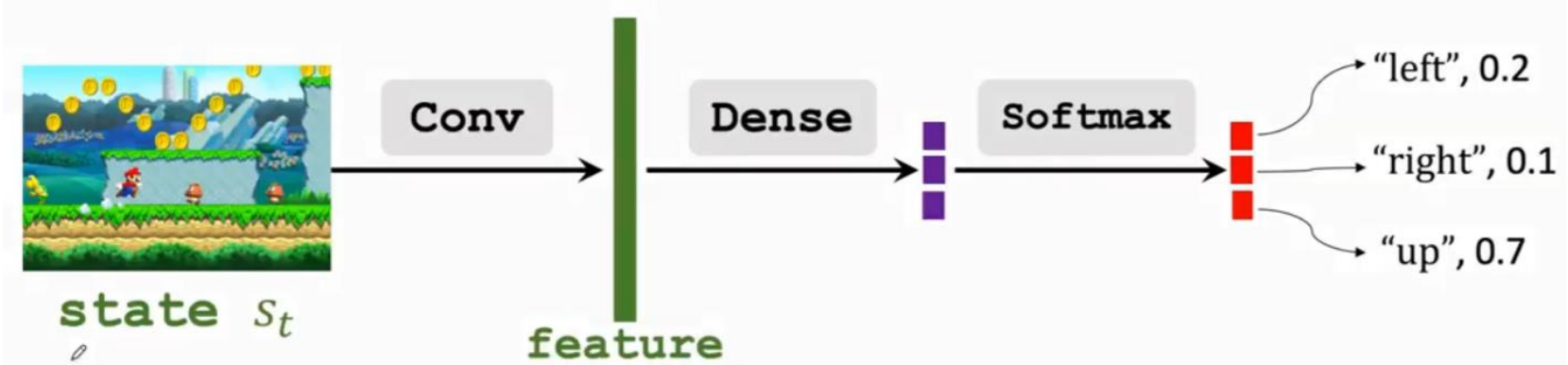
  RNN: recurrent neural network

Policy gradient

- **Policy function**: $\pi(a|s)$, $\sum_{a\in A}\pi(a|s)=1$.
- **Input**: s.
- **Output**: probabilities for all actions, e.g.,

$\pi(\text{left}|s)=0.2$,
$\pi(\text{right}|s)=0.1$,
$\pi(\text{up}|s)=0.7$.

- **Policy network**: use $\pi(a|s;\theta)$ to approximate $\pi(a|s)$.



Policy gradient

- **Discounted return:**

$$\mathrm{U_t} = \mathrm{R_t} + \gamma R_{t+1} + \gamma^2 \mathrm{R_{t+2}} + \gamma^3 \mathrm{R_{t+3}} + \cdots$$

- **Action-value function:**

$$Q_\pi(s_t, a_t) = E[U_t | S_t = s_t, A_t = a_t]$$

- **State-value function:**

$$V_\pi(s_t) = E_A[Q_\pi(s_t, A)]$$

$$\text{(If descrete)} = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a).$$

- **Approximate state-value function:**

$$V(s; \boldsymbol{\theta}) = \sum_a \pi(a|s; \boldsymbol{\theta}) \cdot Q_\pi(s, a)$$

Policy gradient

- **Approximate state-value function:**

$$V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$$

If a policy is good, the mean value of $V_\pi(s_t)$ is supposed to be big,

so we can define the target function:

$$J(\boldsymbol{\theta}) = E_S[V_\pi(S)].$$

The target is to maximize $J(\boldsymbol{\theta})$ by updating $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \boxed{\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}}$$

policy gradient

Policy gradient

$$V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a), \frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial V(s;\theta)}{\partial \theta} = \frac{\partial \sum_a \pi(a|s;\theta) \cdot Q_\pi(s,a)}{\partial \theta}$$

$$= \sum_a \frac{\partial \pi(a|s;\theta) \cdot Q_\pi(s,a)}{\partial \theta}$$

(assume that $Q_\pi(s,a)$ is independent of θ)

$$= \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} \cdot Q_\pi(s,a) \quad \text{— Form 1}$$

$$= \sum_a \left\{ \pi(a|s;\theta) \cdot \left[\frac{\partial \ln \pi(a|s;\theta)}{\partial \theta} \cdot Q_\pi(s,a) \right] \right\}$$

$$= E_A \left[\frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A) \right] \quad \text{— Form 2}$$

Policy gradient

2 forms of policy gradient:

- Form 1: $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} \cdot Q_\pi(s,a)$
- Form 2: $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = E_{A\sim\pi(\cdot|s;\boldsymbol{\theta})}[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)]$

Policy gradient

- Use form 1: $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(s,a)$

  1. For every a∈A, calculate f(a,θ)=$\frac{\partial \pi(a|s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(s,a)$.

  2. $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_a f(a,\theta)$.

- E.g., A={“left”, “right”,”up”}.

$$\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = f(\text{"left"},\theta) + f(\text{"right"},\theta) + f(\text{"up"},\theta).$$

Policy gradient

- Use form 2: $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = E_{A\sim\pi(\cdot|s;\boldsymbol{\theta})}[\frac{\partial \log\pi(A|s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(s,A)]$

  1. (Monte Carlo) Randomly sample $\hat{a}$ according to $\pi(\cdot\,|s;\boldsymbol{\theta})$.
  2. Calculate $g(\hat{a},\theta)=\frac{\partial \log\pi(\hat{a}|s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(s,\hat{a})$.
  3. Use $g(\hat{a},\theta)$ to approximate $\frac{\partial V(s;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

Policy gradient

Algorithm:

1. Observe s_t.
2. Randomly sample a_t according to $\pi(\cdot\,|s;\boldsymbol{\theta})$.
3. **Compute $\mathbf{q_t \approx Q_\pi(s_t, a_t)}$.**
4. Differentiate policy network: $\boldsymbol{d}_{\theta,t} = \frac{\partial \log \pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
5. Approximate policy gradient: $g(a_t, \boldsymbol{\theta}_t) = q_t \cdot \boldsymbol{d}_{\theta,t}$.
6. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot g(a_t, \boldsymbol{\theta}_t)$.

REINFORCE & actor-critic

REINFORCE: Monte Carlo

- Play the whole game and record the trajectory:

$$s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t$$

- Compute the discounted return $u_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ for all t.
- Because $Q_\pi(s_t, a_t) = E[U_t | S_t = s_t, A_t = a_t]$, we can use u_t to approximate $Q_\pi(s_t, a_t)$.
- Use $q_t = u_t$.

REINFORCE & actor-critic

Actor-critic method: use neural network to approximate Q_π

- **State-value functon**

  $$V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s) \cdot Q_\pi(s,a) \approx \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot q(s,a;\boldsymbol{w})$$

- **Policy network – actor**

  Use neural network $\pi(a|s;\boldsymbol{\theta})$ to approximate $\pi(a|s)$.

  Use policy network to control the agent.

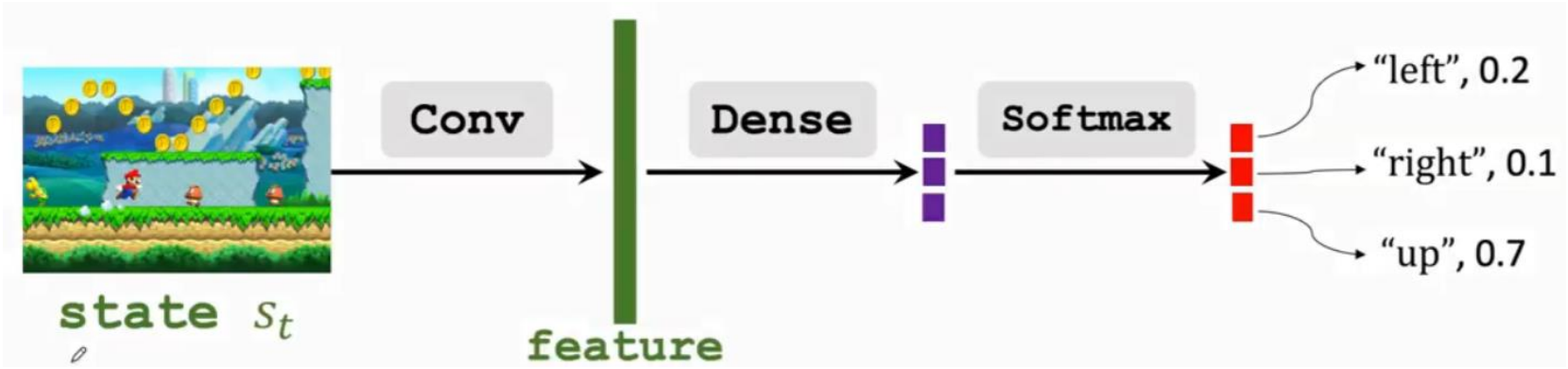
- **Value network – critic**

  Use neural network $q(s,a;\boldsymbol{w})$ to approximate $Q_\pi(s,a)$.

  It's only used to give a grade, not to control the agent.

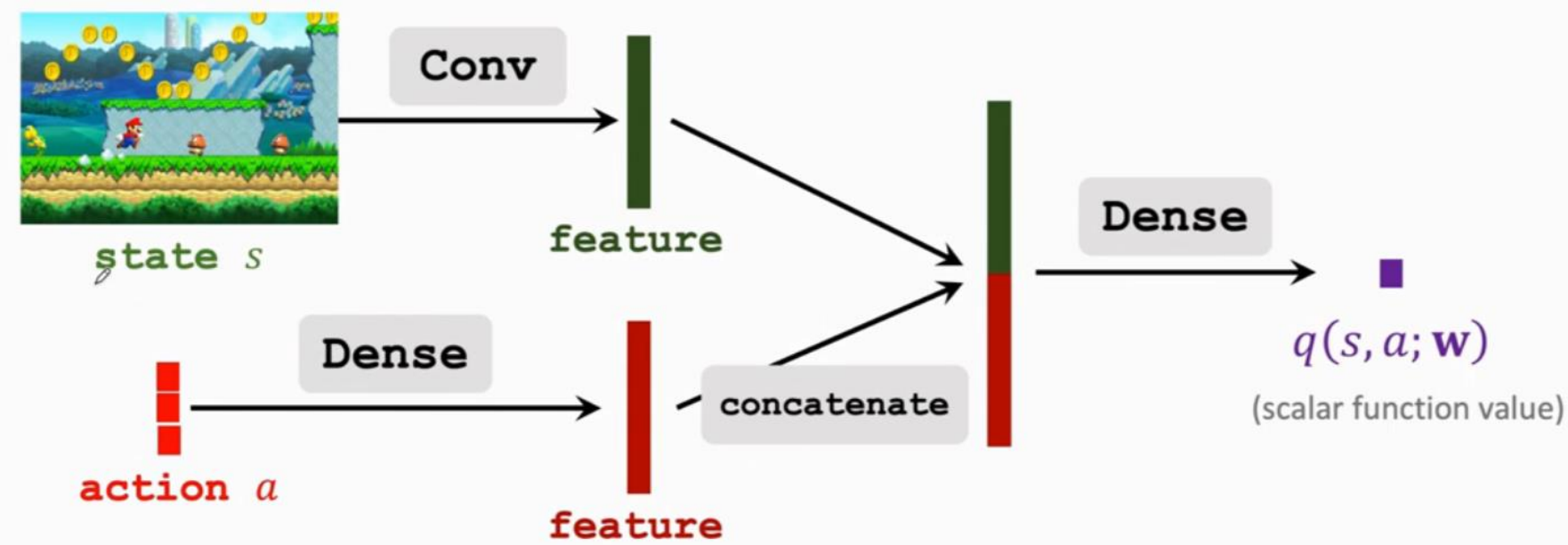
REINFORCE & actor-critic

Policy network – actor: $\pi(a|s;\theta)$



REINFORCE & actor-critic

Value network – critic: $q(s, a; \boldsymbol{w})$



REINFORCE & actor-critic

Actor-critic method: use neural network to approximate Q_π

- **State-value functon**

  $V(s;\theta) = \sum_a \pi(a|s) \cdot Q_\pi(s,a) \approx \sum_a \pi(a|s;\theta) \cdot q(s,a;\boldsymbol{w})$

- **Policy network – actor**

  $\pi(a|s;\theta)$ is updated to increase state-value $V(s;\theta)$.

- **Value network – critic**

  $q(s,a;\boldsymbol{w})$ is updated to estimate the return more accurately.

REINFORCE & actor-critic

Train actor-critic

1. Observe s_t.
2. Randomly sample a_t according to $\pi(\cdot\,|s_t;\theta_t)$.
3. Perform a_t and observe new state s_{t+1} and reward r_t.
4. Update **w** in value network $q(s,a;\boldsymbol{w})$, using TD.
5. Update **θ** in policy network $\pi(a|s;\boldsymbol{\theta})$, using policy gradient.

REINFORCE & actor-critic

Summary of actor-critic

1. Observe s_t and randomly sample a_t according to $\pi(\cdot\,|s_t;\theta_t)$.
2. Perform a_t and observe new state s_{t+1} and reward r_t.
3. Randomly sample $\tilde{a}_{t+1}$ according to $\pi(\cdot\,|s_{t+1};\theta_t)$. (NOT perform $\tilde{a}_{t+1}$)
4. Evaluate value network: $q_t = q(s_t, a_t; \boldsymbol{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \boldsymbol{w}_t)$.
5. Compute TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$. TD target
6. Differentiate value network: $\boldsymbol{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}}|_{\boldsymbol{w}=\boldsymbol{w}_t}$.
7. Update value network $q(s, a; \boldsymbol{w})$: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \cdot \delta_t \cdot \boldsymbol{d}_{w,t}$.
8. Differentiate value network: $\boldsymbol{d}_{\theta,t} = \frac{\partial log\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network $\pi(a|s;\theta)$: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot q_t \cdot \boldsymbol{d}_{\theta,t}$.

REINFORCE & actor-critic

Summary of actor-critic

1. Observe s_t and randomly sample a_t according to $\pi(\cdot\,|s_t;\theta_t)$.
2. Perform a_t and observe new state s_{t+1} and reward r_t.
3. Randomly sample $\tilde{a}_{t+1}$ according to $\pi(\cdot\,|s_{t+1};\theta_t)$. (NOT perform $\tilde{a}_{t+1}$)
4. Evaluate value network: $q_t = q(s_t, a_t; \boldsymbol{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \boldsymbol{w}_t)$.
5. Compute TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$. TD target
6. Differentiate value network: $\boldsymbol{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}}|_{\boldsymbol{w}=\boldsymbol{w}_t}$.
7. Update value network $q(s, a; \boldsymbol{w})$: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \cdot \delta_t \cdot \boldsymbol{d}_{w,t}$.
8. Differentiate value network: $\boldsymbol{d}_{\theta,t} = \frac{\partial log\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network $\pi(a|s;\theta)$: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \delta_t \cdot \boldsymbol{d}_{\theta,t}$. With baseline

REINFORCE with baseline

State-value function:

$$V_\pi(s) = E_A[Q_\pi(s,A)] = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a).$$

Policy gradient:

$$\frac{\partial V_\pi(s)}{\partial \boldsymbol{\theta}} = E_{A\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right]$$

Baseline: make b independent of A

- Prove $E_{A\sim\pi}\left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta}\right] = 0$:

$$E_{A\sim\pi}\left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta}\right] = b \cdot E_{A\sim\pi}\left[\frac{\partial \ln \pi(A|s;\theta)}{\partial \theta}\right]$$

$$= b \cdot \sum_a \pi(a|s;\theta) \cdot \left[\frac{\partial \ln \pi(a|s;\theta)}{\partial \theta}\right]$$

$$= b \cdot \sum_a \pi(a|s;\theta) \cdot \left[\frac{1}{\pi(a|s;\theta)} \cdot \frac{\partial \pi(a|s;\theta)}{\partial \theta}\right]$$

$$= b \cdot \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} \quad \sum_a \pi(a|s;\theta) = 1$$

$$= b \cdot \frac{\partial \left(\sum_a \pi(a|s;\theta)\right)}{\partial \theta}$$

$$= b \cdot \frac{\partial 1}{\partial \theta} = 0$$

$$E_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right] = E_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right] - 0$$

$$= E_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right] - E_{A\sim\pi}\left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta}\right]$$

$$= E_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial log\pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$$

REINFORCE with baseline

Policy gradient with baseline

$$\frac{\partial V_\pi(s_t)}{\partial \boldsymbol{\theta}} = E_{A_t\sim\pi}[\frac{\partial ln\pi(A_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot (Q_\pi(s_t, A_t) - b)]$$

Monte carlo approximation

- Randomly sample $a_t \sim \pi(\cdot\,|s_t;\theta)$ and compute $g(a_t)$
- $g(a_t)$ is an unbiased estimate of the policy gradient:

$$E_{A_t\sim\pi}[g(A_t)] = \frac{\partial V_\pi(s_t)}{\partial \boldsymbol{\theta}}.$$

Stochastic policy gradient

$$g(a_t) = \frac{\partial ln\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot (Q_\pi(s_t, a_t) - b).$$

Stochastic policy gradient ascent:

$$\theta \leftarrow \theta + \beta \cdot g(a_t).$$

b (independent of A_t) doesn't affect $E_{A_t\sim\pi}[g(A_t)]$, but affects $g(a_t)$. A well chosen b leads to small variance and speeds up convergence.

REINFORCE with baseline

Stochastic policy gradient (b= $V_\pi(s_t)$)

$$g(a_t)=\frac{\partial ln\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot (Q_\pi(s_t,a_t)-V_\pi(s_t)).$$

- (Monte carlo) In REINFORCE:

$$u_t \approx Q_\pi(s_t,a_t).$$

- Approximate $V(s;\boldsymbol{\theta})$ with value network $v(s;\boldsymbol{w})$.

Approximate policy gradient:

$$\frac{\partial V_\pi(s_t)}{\partial \boldsymbol{\theta}} \approx g(a_t) \approx \frac{\partial ln\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot (u_t - v(s_t;\boldsymbol{w})).$$

REINFORCE with baseline

Summary of REINFORCE with baseline

- Play a game to the end and observe the trajectory:

$$s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t$$

- Compute the discounted return $u_t = \sum_{k=t}^{n} \gamma^{k-t} \cdot r_k$ and $\delta_t = v(s_t; \boldsymbol{w}) - u_t$.
- Update the policy network:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \cdot \delta_t \cdot \frac{\partial ln\pi(a_t | s_t; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}.$$

- Update the value network:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \delta_t \cdot \frac{\partial v(s_t; \boldsymbol{w})}{\partial \mathbf{w}}.$$

Advantage actor-critic (A2C)

Stochastic policy gradient (b= $V_\pi(s_t)$)

Advantage

$$g(a_t) = \frac{\partial ln\pi(a_t|s_t;\theta)}{\partial \theta} \cdot \left(Q_\pi(s_t, a_t) - V_\pi(s_t)\right).$$

- **Policy network** – actor: $\pi(a|s;\theta)$
  - Approximation to policy function $\pi(a|s)$.
  - Controls the agent.
- **Value network** – critic: $v_\pi(s;\boldsymbol{w})$.
  - Approximation to state value function $V_\pi(s)$.
  - Evaluates the state s.

Advantage actor-critic (A2C)

Train A2C

- Observe a transition (s_t, a_t, r_t, s_{t+1}).
- TD target: $y_t = r_t + \gamma \cdot v(s_{t+1}, \boldsymbol{w})$.
- TD error: $\delta_t = v(s_t, \boldsymbol{w}) - y_t$.
- Update policy network (actor) with:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \cdot \delta_t \cdot \frac{\partial ln\pi(a_t|s_t;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}.$$

- Update the value network (critic) with:

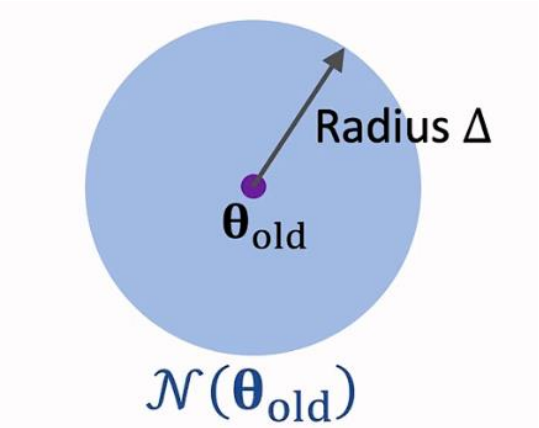
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \delta_t \cdot \frac{\partial v(s_t;\boldsymbol{w})}{\partial \mathbf{w}}.$$

TRPO: trust region policy optimization

Trust region methods, target: $\max_{\theta} J(\theta)$

- Trust region: $N(\theta_{now}) = \{\theta | \|\theta - \theta_{now}\|_2 \leq \Delta\}$.
- Construct function $L(\theta|\theta_{now})$, satisfying:

  $L(\theta|\theta_{now})$ is very close to $J(\theta)$, $\forall \theta \in N(\theta_{now})$
- $J(\theta)$ can be replaced by $L(\theta|\theta_{now})$ when $\theta \in N(\theta_{now})$



TRPO: trust region policy optimization

Train

- **Approximate**

  a) Present policy network parameter is θ_{now}. Use $\pi(a|s;\theta_{now})$ to control the agent, record trajectories: $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$.

  b) For all t, calculate u_t.

  c) Approximate function: $\tilde{L}(\theta|\theta_{now}) = \frac{1}{n}\sum_{t=1}^{n}\frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{now})}\cdot u_t$.

- **Maximize**

$$\theta_{new} = argmax_{\theta}\tilde{L}(\theta|\theta_{now});\ \ s.t.\|\theta-\theta_{now}\|_2 \le \Delta.$$

Entropy regularization

- **Entropy**

$$\text{Entropy(p)}=-\sum_{i=1}^{n} p_i \cdot lnp_i.$$

- Small entropy means probability is too concentrated, while big entropy means greater randomness (prefered).
- **Entropy of probability distribution**

$$H(s;\boldsymbol{\theta}) \triangleq Entropy[\pi(\cdot\,|s;\boldsymbol{\theta})] = -\sum_{a\in \mathrm{A}} \pi(a|s;\boldsymbol{\theta}) \cdot ln\pi(a|s;\boldsymbol{\theta}).$$

- **Policy-based RL using entropy regularization**

$$max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \lambda \cdot E_S[H(s;\boldsymbol{\theta})].$$

- **Gradient and update $\boldsymbol{\theta}$**

$$\tilde{\boldsymbol{g}}(s,a;\boldsymbol{\theta}) \triangleq [Q_\pi(s,a) - \lambda \cdot ln\pi(a|s;\boldsymbol{\theta}) - \lambda] \cdot \nabla_{\boldsymbol{\theta}} ln\pi(a|s;\boldsymbol{\theta}).$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \tilde{\boldsymbol{g}}(s,a;\boldsymbol{\theta}).$$

RNN: recurrent neural network

- **Problem:** in many cases, the observations are partial, leading to difficult decision-making. Replacing s_t with o_t is the simplest way, but it's not efficient enough.
- **Solution: remember all the o_i: $o_1, o_2, \dots, o_t$**, denoted by $o_{1:t} = [o_1, o_2, \dots, o_t]$, and policy network is $\pi(a_t|o_{1:i}; \boldsymbol{\theta})$.
- However, with the growth of t, the length of $o_{1:t}$ changes accordingly.
- We can use recurrent neural network (RNN).

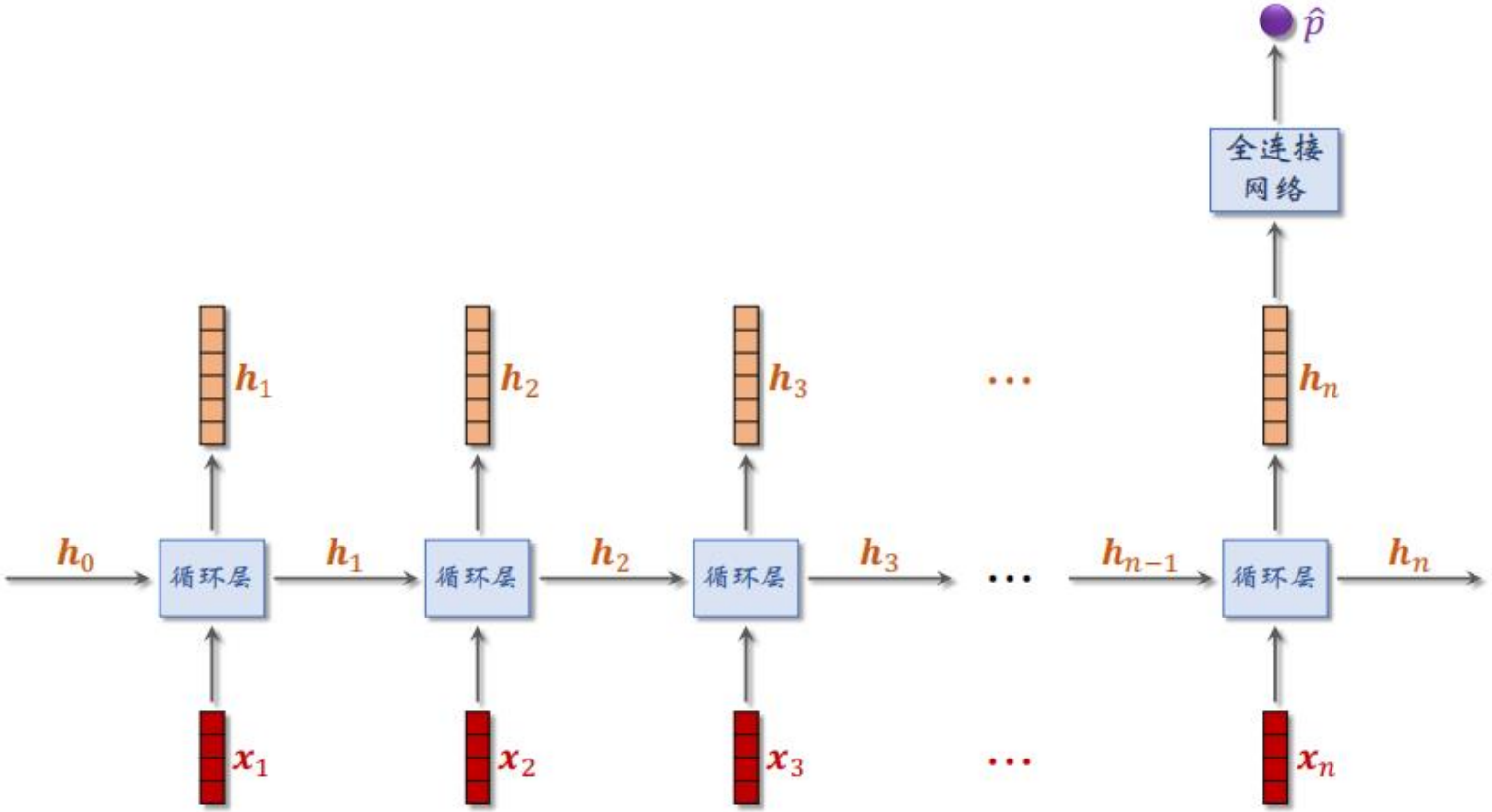
RNN: recurrent neural network

- For all t=1, 2, …, n, recurrent layer maps $[x_1, x_2, \ldots, x_t]$ to h_t.

$$
\begin{aligned}
(\boldsymbol{x}_1) &\Longrightarrow \boldsymbol{h}_1, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2) &\Longrightarrow \boldsymbol{h}_2, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) &\Longrightarrow \boldsymbol{h}_3, \\
&\vdots \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_{n-1}) &\Longrightarrow \boldsymbol{h}_{n-1}, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_{n-1}, \boldsymbol{x}_n) &\Longrightarrow \boldsymbol{h}_n.
\end{aligned}
$$

- The length of h_t is fixed.

RNN: recurrent neural network

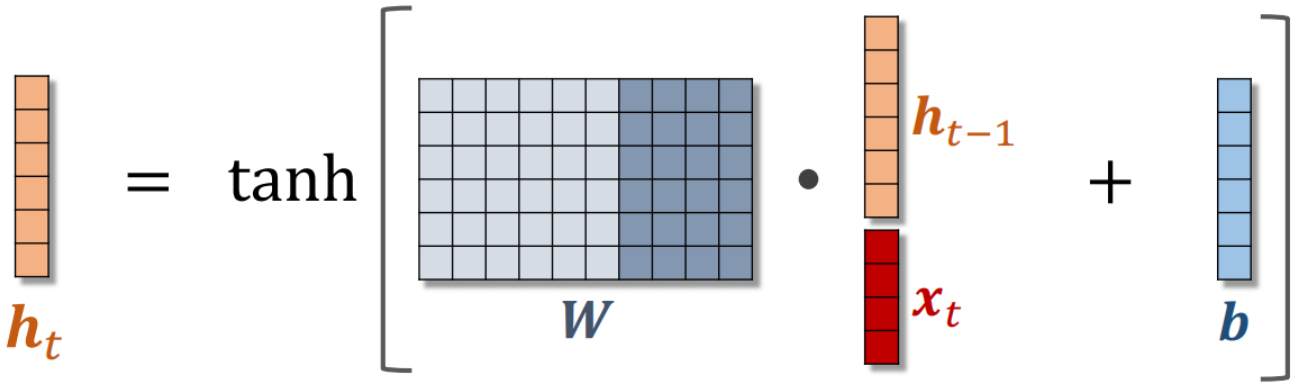


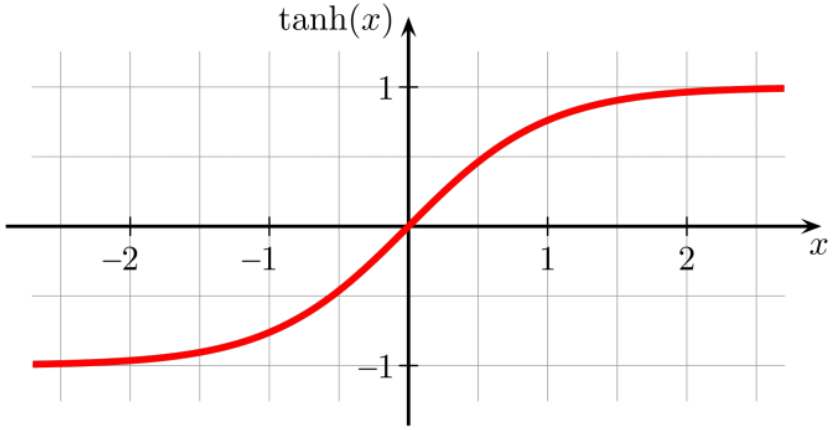
Input is $[x_1, x_2, \ldots, x_n]$, output is $\hat{p}$.

RNN: recurrent neural network

- **Simple recurrent layer**

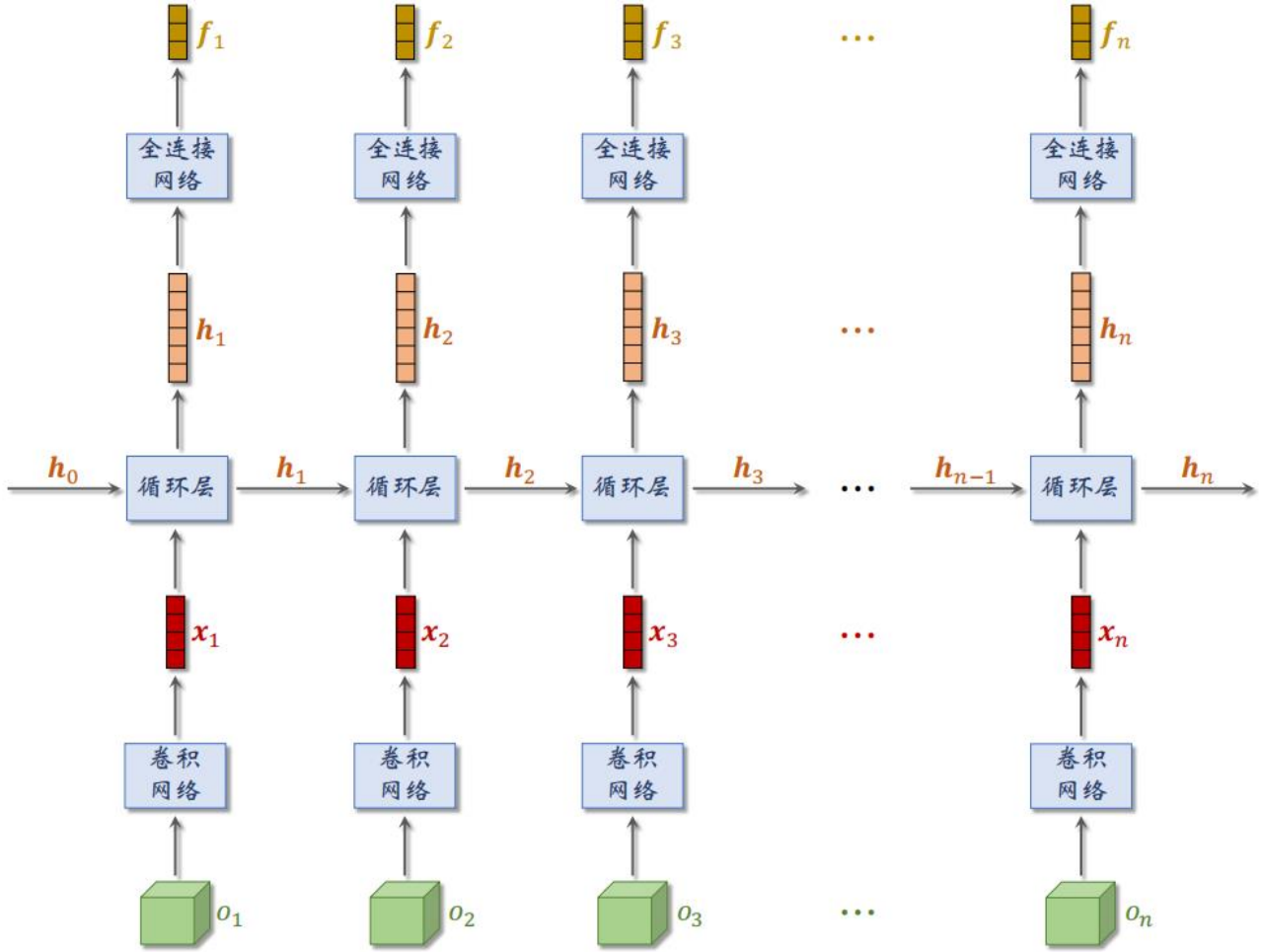
$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}).$$





RNN: recurrent neural network

RNN as policy network: $\boldsymbol{f}_t = \pi(a_t|\boldsymbol{o}_{1:t};\boldsymbol{\theta})$.



Thank you.