澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Group Report

speaker：Xiongyi Li

7th Dec 2023

Outline

- Deep Reinforcement Learning
  - Part Ⅲ
  - Part Ⅳ

Part III Policy-Based Reinforce Learning

- Revison
- Policy gradient
- Baseline
- High-level skills
- Continue control

Revision

- Discounted return(折扣回报):

$$U_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_n.$$

- Action-value function(动作价值函数):

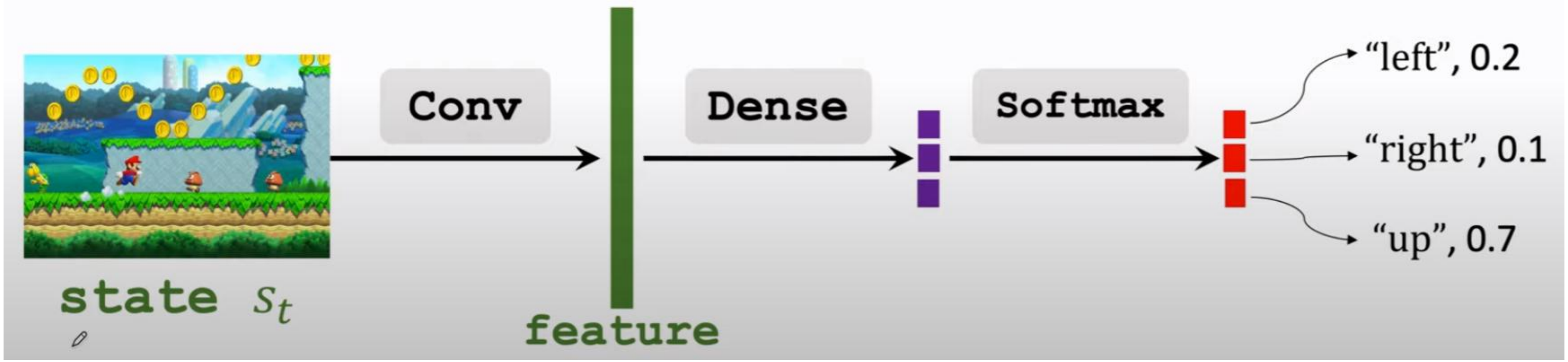
$$Q_\pi(s_t, a_t) = \mathbb{E}_{S_{t+1},A_{t+1},\cdots,S_n,A_n}\left[U_t \,\middle|\, S_t = s_t, A_t = a_t\right].$$

- State-value function(状态价值函数):

$$V_\pi(s) = \mathbb{E}_{A\sim\pi}\left[Q_\pi(s, A)\right].$$

Policy Gradient

- Policy Network $\pi(a|s;\theta)$
- To approximate $\pi(a|s)$



Policy Gradient

- $V_\pi(s) = \mathbb{E}_{A\sim\pi}\left[Q_\pi(s, A)\right]. = \sum_a \pi(a|st;\theta)Q_\pi(st, a)$
- θ dicide the value of $V\pi(s)$

$$\boxed{J(\boldsymbol{\theta}) = \mathbb{E}_S\left[V_\pi(S)\right].} \longrightarrow \boxed{\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).}$$

- Policy gradient ascent
- $\theta = \theta + \beta \frac{\partial V(s,\theta)}{\partial \theta}$

Policy Gradient

- $\frac{\partial V(s,\theta)}{\partial \theta} = \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} \mathrm{Q}_\pi(s,a)$
- $\frac{\partial V(s,\theta)}{\partial \theta} = \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} \mathrm{Q}_\pi(s,a)$

$$= \sum_a \pi(a|s;\theta) \frac{\partial \ln[\pi(a|s;\theta)]}{\partial \theta} \mathrm{Q}_\pi(s,a)$$

$$= \mathbb{E}_A\left[\frac{\partial \ln[\pi(a|s;\theta)]}{\partial \theta} \mathrm{Q}_\pi(s,a)\right]$$

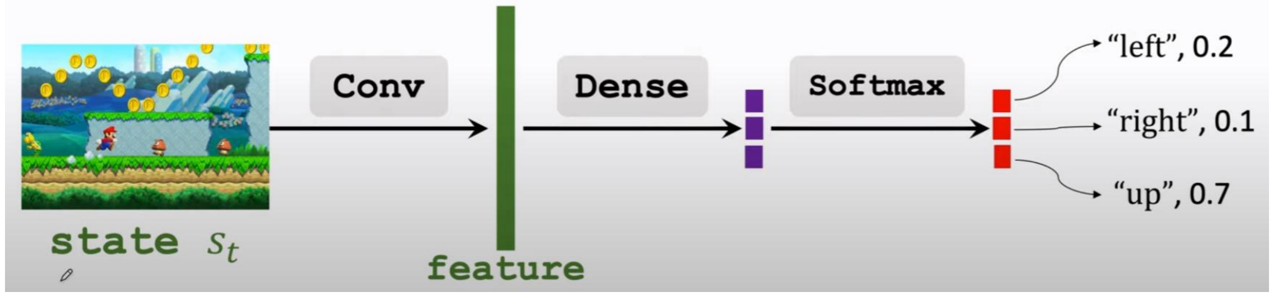
Policy Gradient

- Algorithm:
- Oberseve s_t
- Random action $a_t \leftarrow \pi(\cdot|st;\theta)$
- Compute $q_t \approx Q_\pi(st,at)$
- $G(a_t,\theta_t) = q_t \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial\theta}\Big|_{\theta=\theta t}$
- $\theta_{t+1} = \theta_t + \beta G(a_t,\theta_t)$

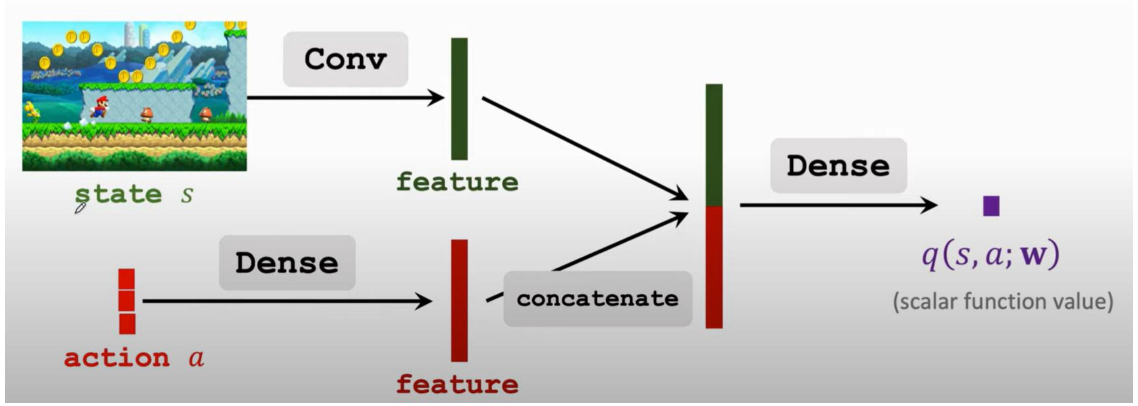
Reinforce

- Get s_1,a_1,r_1; s_2,a_2,r_2;…. s_t,a_t,r_t
- Monte Calro: $u_t = Q_\pi(s_t, a_t)$

Actor-Critic

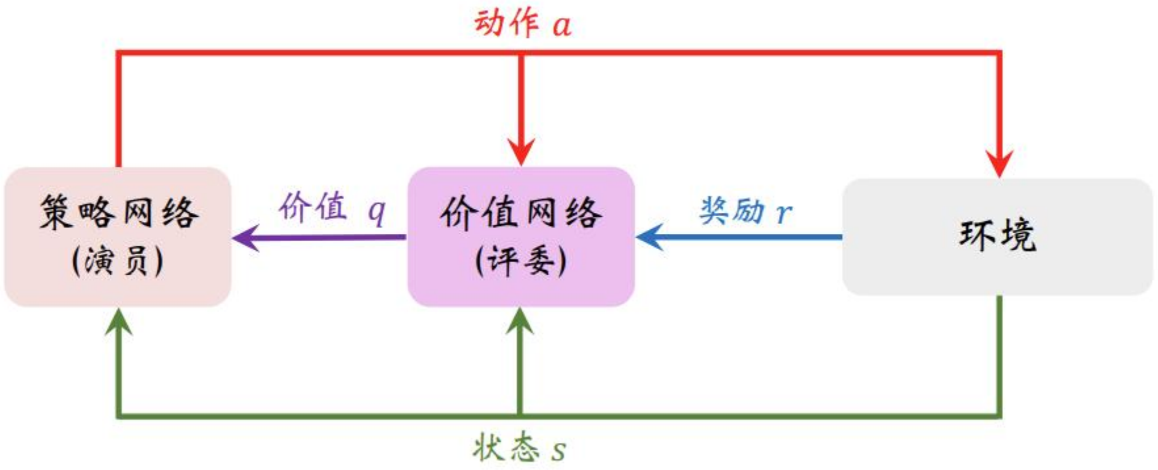


- $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$
- Actor: $\pi(a|s;\theta)$ approximate $\pi(a|s)$
- Critic: $q(s,a;\omega)$ approximate $Q_\pi(s,a)$
- $V_\pi(s) = \sum_a \pi(a|s;\theta) q(s,a;\omega)$
- Update θ to increase $V(s, \theta, \omega)$
- Update ω to make q more accurate



Actor-Critic

- Algorithm:
- Oberseve s_t
- Random action $a_t \leftarrow \pi(\cdot|st;\theta)$
- from a_t to observe S_{t+1} & r_t
- Update θ by TD
- Update ω by Policy Gradient



Policy Gradient with Baseline

- $\frac{\partial V(s,\theta)}{\partial \theta}$ = $\mathbb{E}_A[\frac{\partial \ln[\pi(a|s;\theta)]}{\partial \theta} Q_\pi(s,a)]$
- $\mathbb{E}_{A\text{-}\pi}[\frac{\partial \ln[\pi(A|s;\theta)]}{\partial \theta}\ b]$ (b is independent of A)

$= \mathbb{E}_{A\text{-}\pi}[\frac{\partial \ln[\pi(A|s;\theta)]}{\partial \theta}] * b$

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

$= b * \sum_a \frac{\partial \pi(a|s;\theta)}{\partial \theta} = b * \frac{\partial \sum_a \pi(a|s;\theta)}{\partial \theta} = 0$

Policy Gradient with Baseline

- $\frac{\partial V(s,\theta)}{\partial \theta}$ = $\mathbb{E}_A[\frac{\partial \ln [\pi(a|s;\theta)]}{\partial \theta}(Q_\pi(s,a) - b)]$
- During Monte Calro approximation, b can reduce variance and speed up convergence
- Often b = 0 or $V_\pi(s_t)$ $\quad V_\pi(s) = \mathbb{E}_{A\sim\pi}\left[Q_\pi(s,A)\right].$

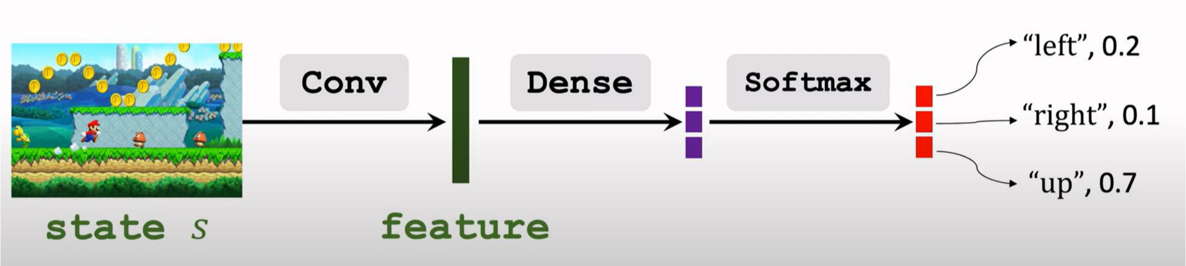
Reinforce with Baseline

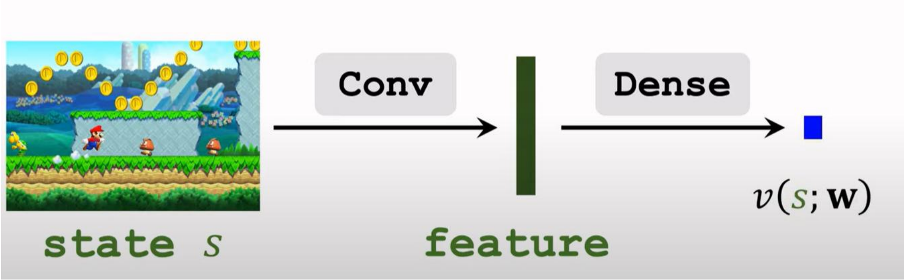
- Algorithm:
- Oberseve s_t
- Random action $a_t \longleftarrow \pi(\cdot \mid s_t; \theta)$
- Compute $q_t \approx Q_\pi(s_t, a_t)$
- $G(a_t, \theta_t) = q_t \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta} \Big|_{\theta=\theta t}$
- $\theta_{t+1} = \theta_t + \beta G(a_t, \theta_t)$

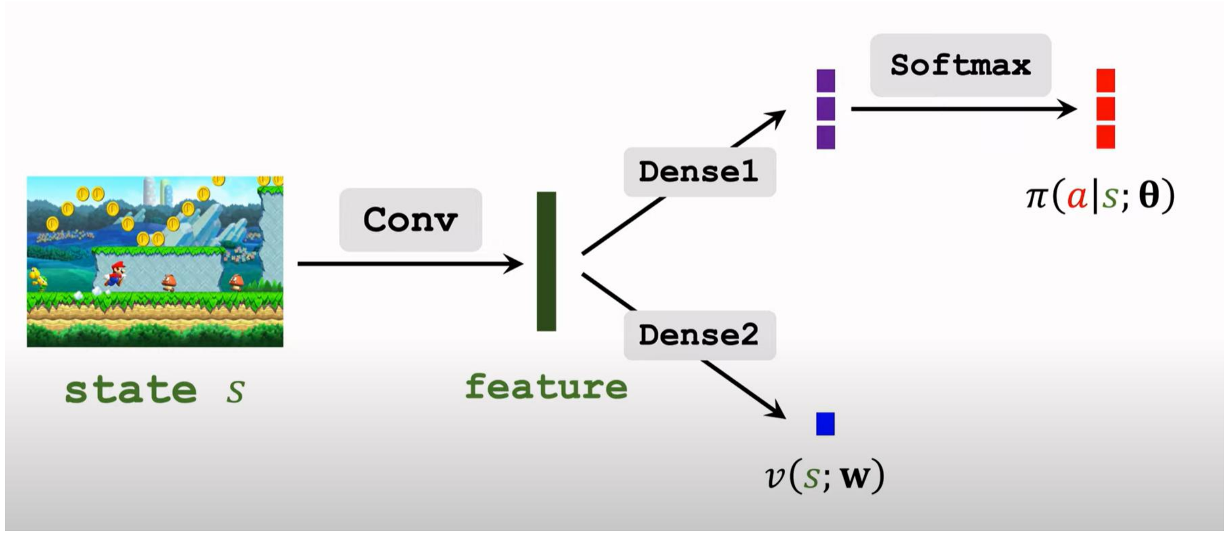
- $G(a_t) = \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta} (Q_\pi(st, at) - V_\pi(s_t))$

  $= \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta} (u_t - V_\pi(s_t))$
- Use neural network v(s;w) to approximate $V_\pi(s_t)$
- $G(a_t) = \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta} (u_t - v(s;w))$
- 3 approximation※

Policy and Value Network







Updating

- Policy gradient: θ = $\theta + \beta \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta}$ $(u_t -$v(s;w)) $\xrightarrow{\hspace{2em}} -\delta_t$
- Value network:
- Prediction error: δ_t = v(s$_t$;w) $-\ u_t$
- Gradient: $\frac{\partial \delta_t{}^2/2}{\partial w}$ = $\delta_t\ \frac{\partial \mathrm{v(st;w)}}{\partial \theta}$
- Gradient descent: w = w - $\alpha\delta_t\ \frac{\partial \mathrm{v(st;w)}}{\partial \theta}$

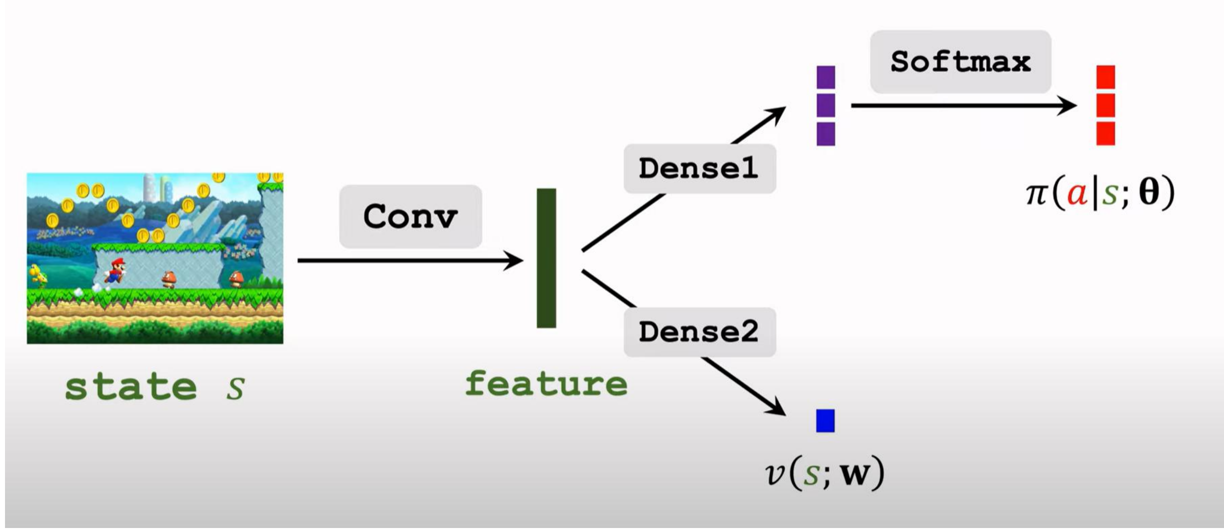
Advantage Actor-Critic (A2C)

- Observe a transition(s_t,s_t,r_t,s_{t+1})
- TD target: y_t = rt + γ v(s_{t+1};w))
- TD error: $\delta_t = \mathrm{v}(s_t;\mathrm{w}) - y_t$
- Update the policy network

$$\theta = \theta - \beta \frac{\partial \ln(\pi(a_t|s_t;\theta))}{\partial \theta} \delta_t$$

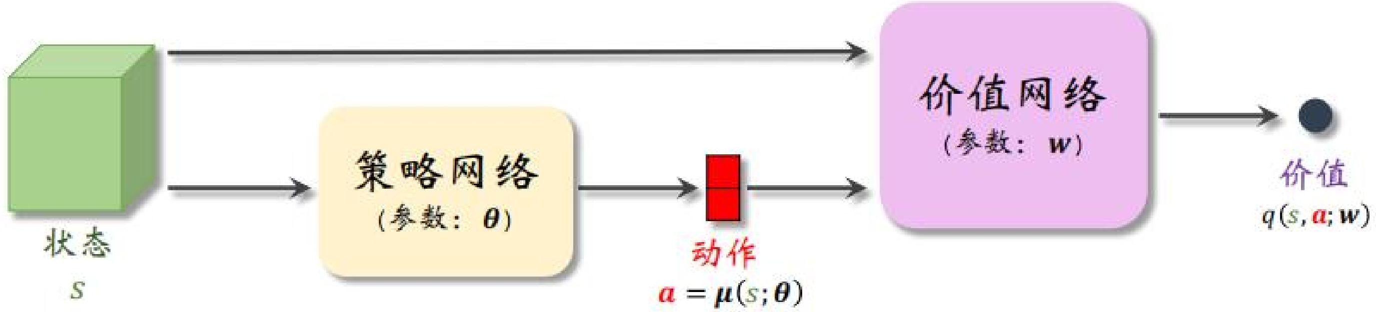
- Update the value network

$$\mathrm{w} = \mathrm{w} - \alpha\delta_t \frac{\partial \mathrm{v}(s_t;\mathrm{w})}{\partial \theta}$$



Continuous control

- Deterministic policy gradient(确定策略梯度)



- Transition(s_t,s_t,r_t,s_{t+1})
- Value network: $q_t = q(s_t,a_t;w)$
- $q_{t+1} = q(s_{t+1},a'_{t+1};w)$, $a'_{t+1} = \pi(s_{t+1};\theta)$
- TD error: $\delta_t = q_t - (r_t + \gamma q_{t+1})$
- Update: $w = w - \alpha\delta_t \frac{\partial q(s_t,a_t;w)}{\partial w}$

Continuous control

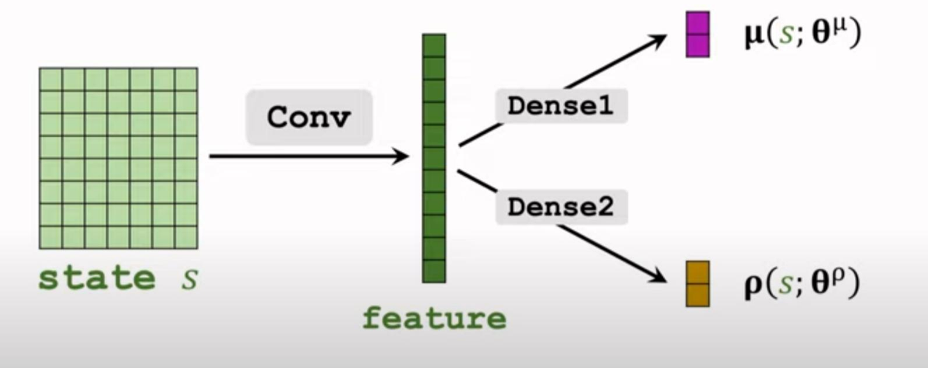
- Policy network
- Goal: increase q(s,a;w) & a = π(s;θ)
- DPG: g = $\frac{\partial q(s,\pi(s;\theta)\ ;w)}{\partial \theta}$ = $\frac{\partial a}{\partial \theta}\frac{\partial q(s,\mathrm{a}\ ;w)}{\partial a}$
- Gradient ascent: $\theta\ =\ \theta$ + βg

Continuous control

- Policy function: $\pi(a \mid s) = \frac{1}{\sqrt{6.28} \cdot \sigma(s)} \cdot \exp\left(-\frac{[a-\mu(s)]^2}{2 \cdot \sigma^2(s)}\right).$
- action a is d-dim

$$\pi(\mathbf{a}|s) = \prod_{i=1}^{d} \frac{1}{\sqrt{6.28}\,\sigma_i} \cdot \exp\left(-\frac{(a_i-\mu_i)^2}{2\sigma_i^2}\right).$$

- use a neural network $\mu(s;\vartheta^{\mu})$ to approximate μ(s)
- use a neural network $\rho(s;\vartheta^{\rho})$ to approximate $\rho = \ln\sigma^2$



Continuous control

- Observe state s.
- Compute mean and log variance using the neural network:

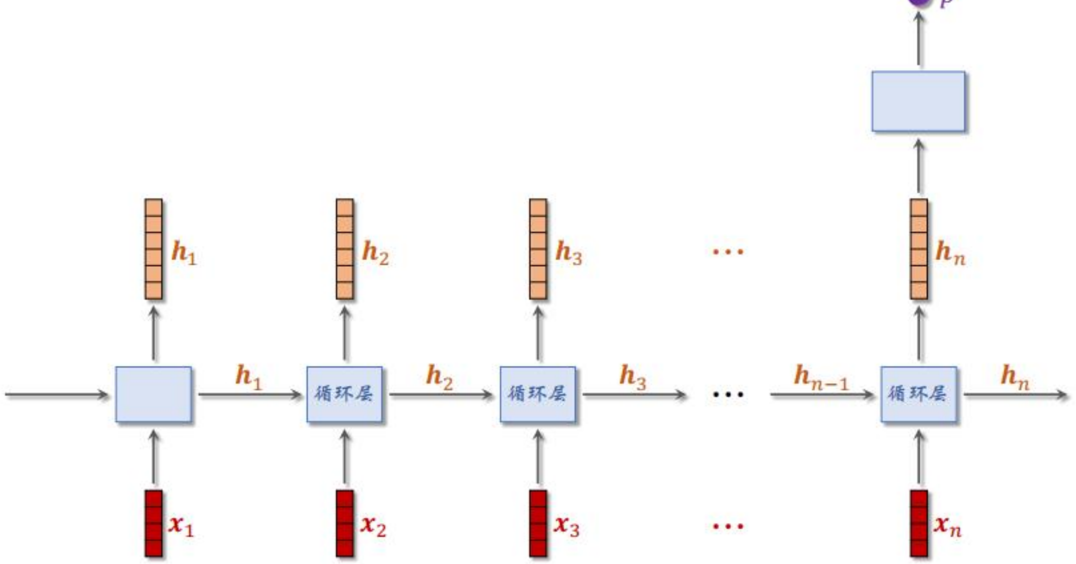
$$\widehat{\boldsymbol{\mu}} = \boldsymbol{\mu}(s; \boldsymbol{\theta}^{\mu}) \quad \text{and} \quad \widehat{\boldsymbol{\rho}} = \boldsymbol{\rho}(s; \boldsymbol{\theta}^{\rho}).$$

- Compute $\hat{\sigma}_i^2 = \exp(\hat{\rho}_i)$, for all $i = 1, \cdots, d$.
- Randomly sample action $\mathbf{a}$ by

$$a_i \sim \mathcal{N}\left(\hat{\mu}_i, \hat{\sigma}_i^2\right), \quad \text{for all } i = 1, \cdots, d.$$

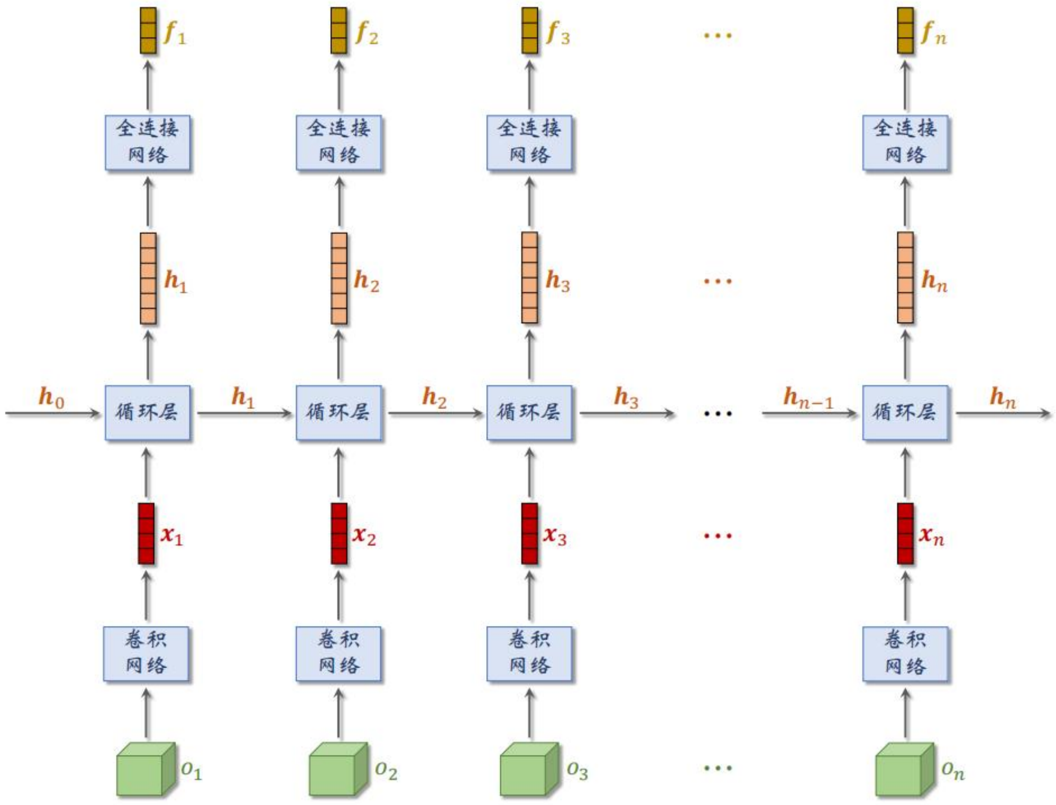
Recurrent neural network(RNN)

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



RNN

- RNN as policy network



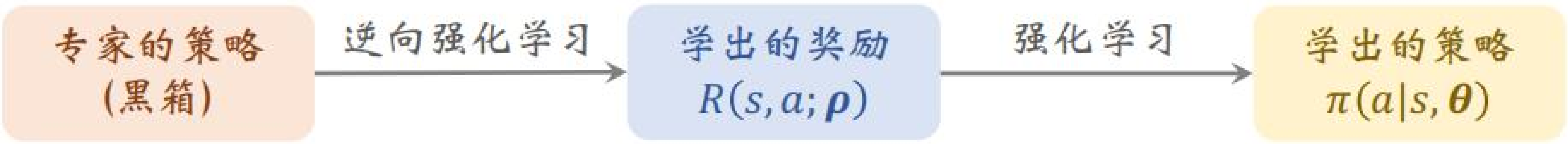
Behaviour cloning

- Goal: mimic human's action to make a random policy network $\pi(a|s;\theta)$ or a certain policy network μ(s;θ)
- Data set: $\mathcal{X} = \Big\{ (s_1, a_1), \cdots, (s_n, a_n) \Big\}.$
- s: state; s: action(human)

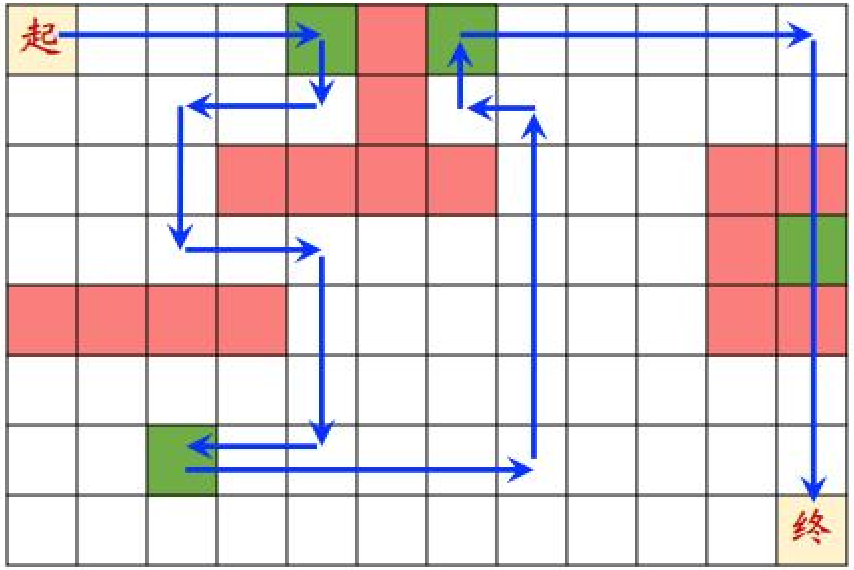
Inverse reinforcement learning

IRL setting: interact with environment without knowing what the reward is;unknown policy provided by human but can be used

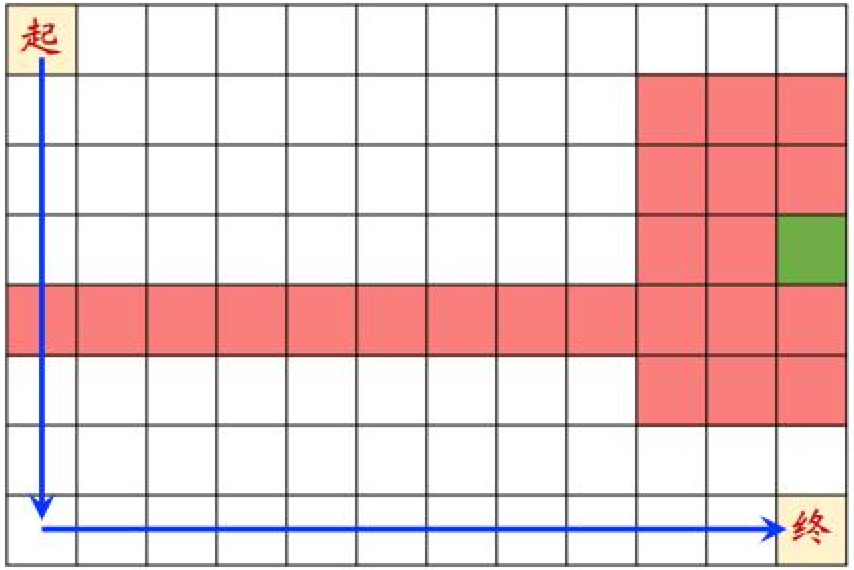
Goal: to make a policy network $\pi(a|s;\theta)$ and mimic the unknown policy



IRL



(a)



(b)

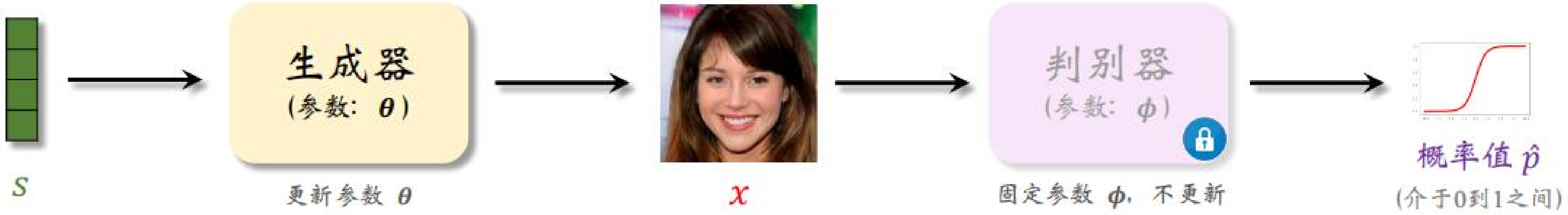
- Green: R+
- Red: R-

IRL

- Use learned reward function to train a policy network $\pi(a|s;\theta)$
- Learned reward function: R(s, a; ρ)
- From the trajectory $(s_1,a_1;s_2,a_2;s_3,a_3.....)$
- $r_t = R(s_t, a_t; \rho)$
- $\theta = \theta + \beta \sum_{t=1}^{n} \gamma^{t-1} u_t \frac{\partial ln(\pi(a|s;\theta))}{\partial \theta}$ (reinforce)

Generative adversarial imitation learning

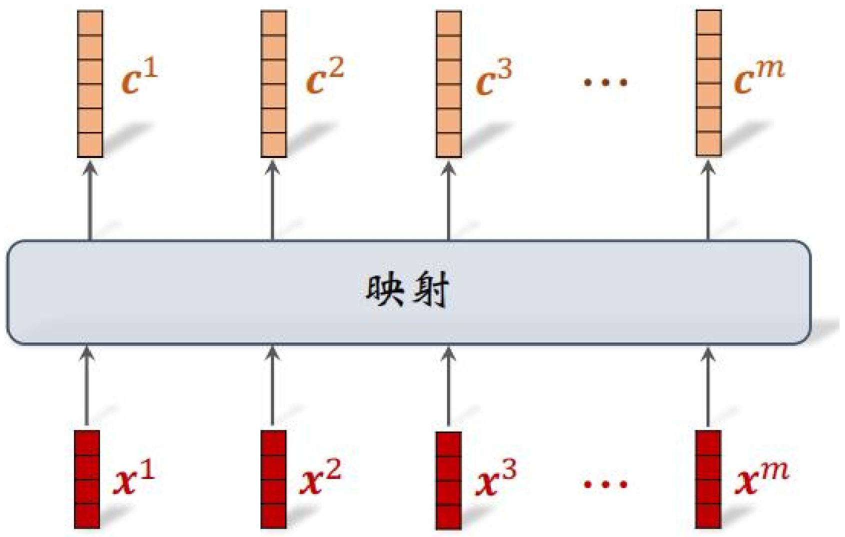
- Generative adversarial network (GAN)



- Loss: $E(s;\boldsymbol{\theta}) = \ln\left[1 - \underbrace{D(x;\boldsymbol{\phi})}_{\text{越大越好}}\right];$
- Output p = D(x;φ)
- Update: $\theta = \theta - \beta \frac{\partial E(s;\theta)}{\partial \theta}$

Attention

- Self-attention layer
- Solve two questions
- 1.m is uncertain and the parameter c
- 2.C^i is related to all input x



Attention

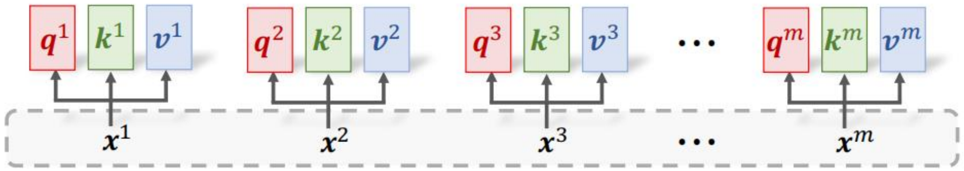


图 17.2: 首先把 $\boldsymbol{x}^i$ 映射到三元组 $(\boldsymbol{q}^i, \boldsymbol{k}^i, \boldsymbol{v}^i)$，$\forall i = 1, \cdots, m$。

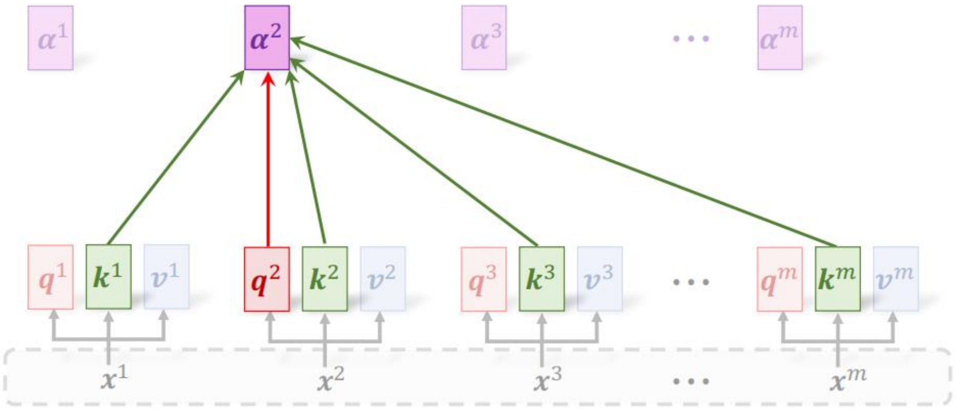


图 17.3: 然后用 $\boldsymbol{q}^i$ 和 $(\boldsymbol{k}^1, \cdots, \boldsymbol{k}^m)$ 计算权重向量 $\boldsymbol{\alpha}^i \in \mathbb{R}^m$，$\forall i = 1, \cdots, m$。

$$\boldsymbol{\alpha}^i \quad = \quad \text{softmax}\Big(\langle \boldsymbol{q}^i, \boldsymbol{k}^1 \rangle, \; \langle \boldsymbol{q}^i, \boldsymbol{k}^2 \rangle, \; \cdots, \; \langle \boldsymbol{q}^i, \boldsymbol{k}^m \rangle \Big),$$

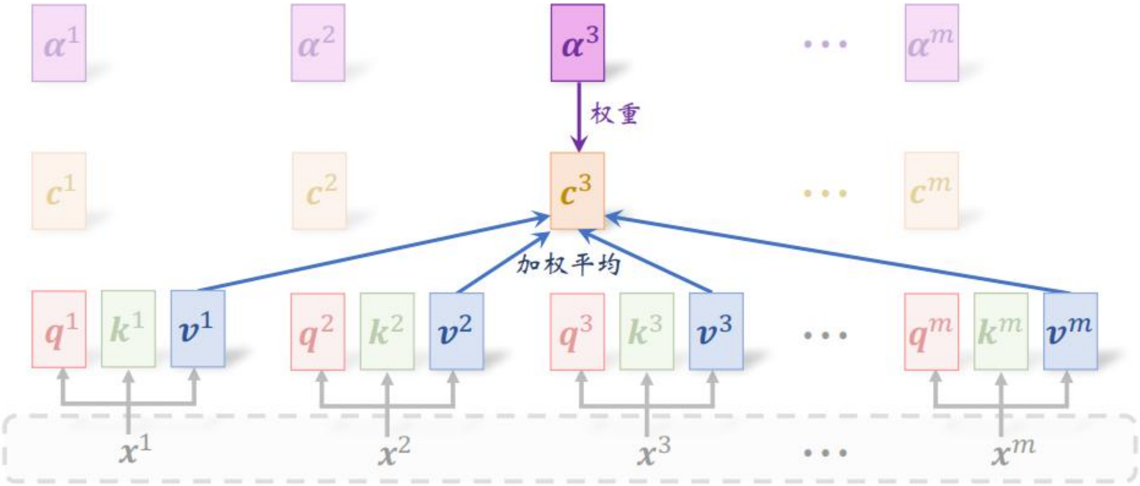


图 17.4: 最后用 $\boldsymbol{\alpha}^i$ 和 $(\boldsymbol{v}^1, \cdots, \boldsymbol{v}^m)$ 计算输出向量 $\boldsymbol{c}^i \in \mathbb{R}^{d_{\text{out}}}$，$\forall i = 1, \cdots, m$。

Attention

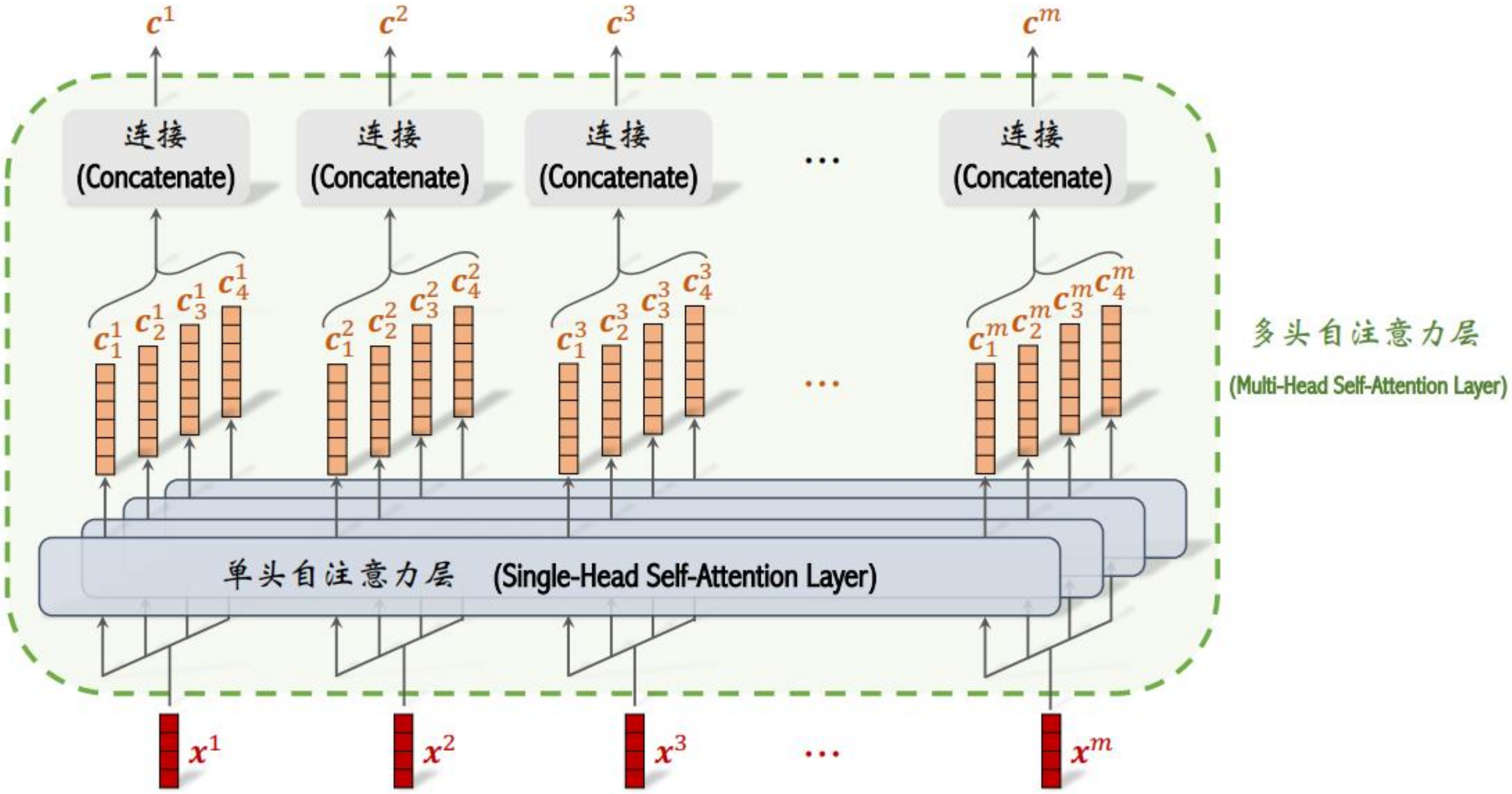


图 17.6: 这个例子中，多头自注意力层由 $l = 4$ 个单头自注意力层组成。

Thank You!

Avenida da Universidade, Taipa, Macau, China
Email： mc35289@um.edu.mo Website： www.um.edu.mo

um 澳大