Lecture 4: Introduction to Policy-Based Learning and Humanoid Project

GEARS Program
Summer 2024

Junxi Zhu

From previous lecture…

Terminology of Reinforcement Learning

- **State $\boldsymbol{s_t}$**: A state represents the status or condition of the environment at a given point t in time. It is a snapshot of all the necessary information that describes the current situation in which an agent is operating.
- **Action $\boldsymbol{a_t}$**: An action is a decision or move made by the agent at a given point t in time that affects the state of the environment. It is a part of the mechanism through which the agent interacts with the environment.
- **Reward $\boldsymbol{r_t}$**: A scalar feedback signal provided to the agent at time step t after taking an action a_t in a state s_t. It indicates how beneficial or detrimental the outcome of that action was with respect to the agent's objectives.
- **Return $\boldsymbol{U_t}$** : Equal to the total accumulated reward received from a state onward until the end of an episode, discounted by the factor γ at each step.

$$U_t = \sum_{k=t}^{n} \gamma^{k-t} R_k$$

Action-Value Functions $Q(s, a)$

- Return (discounted)

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

- Action-value function for policy π

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

- Optimal action-value function

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

- Whatever policy function π is used, the result of taking a_t at state s_t cannot be better than $Q^*(s_t, a_t)$

Q Learning and DQN

- **Goal:** Win the game (≈maximize the total reward)
- **Question**: If we know $Q^*(s, a)$, what is the best action?

NC STATE UNIVERSITY
Biomechatronics and Intelligent Robotics (BIRO) Lab

Q Learning and DQN

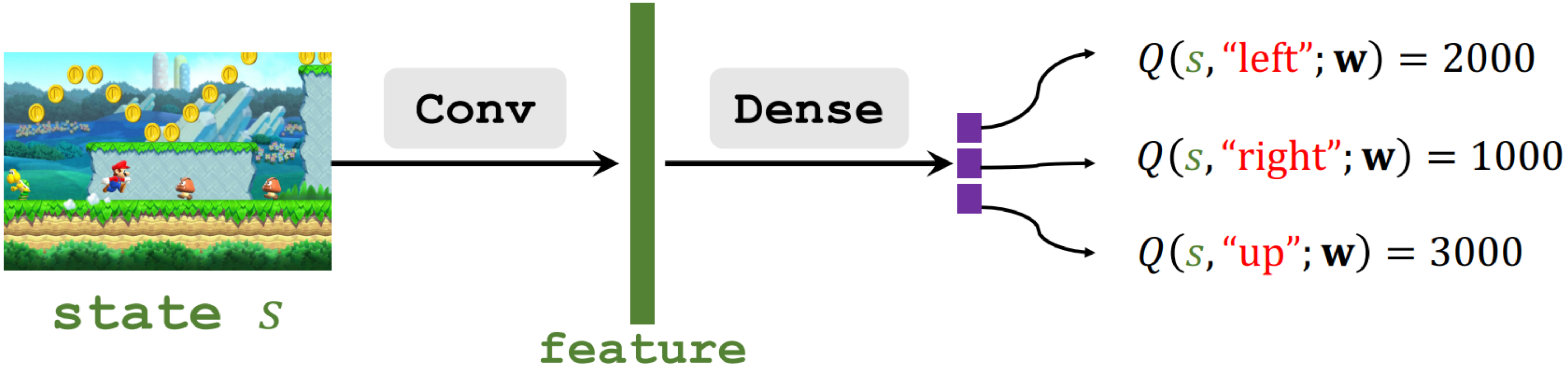
- **Goal**: Win the game (≈maximize the total reward)
- **Question**: If we know $Q^*(s, a)$, what is the best action?
- Obviously, the best action is $a^* = \underset{a}{\operatorname{argmax}}\, Q^*(s, a)$
- Q^* is an indicator of how good it is for an agent to pick action a while being in the state s

Q Learning and DQN

- **Goal**: Win the game (≈maximize the total reward)
- **Question**: If we know $Q^*(s, a)$, what is the best action?
- Obviously, the best action is $a^* = \underset{a}{\operatorname{argmax}}\, Q^*(s, a)$
- Q^* is an indicator of how good it is for an agent to pick action a while being in the state s
- **Challenge**: We do not know $Q^*(s, a)$
  - Solution: Deep Q Network (DQN)
  - Use neural network $Q(s, a; w)$ to approximate $Q^*(s, a)$

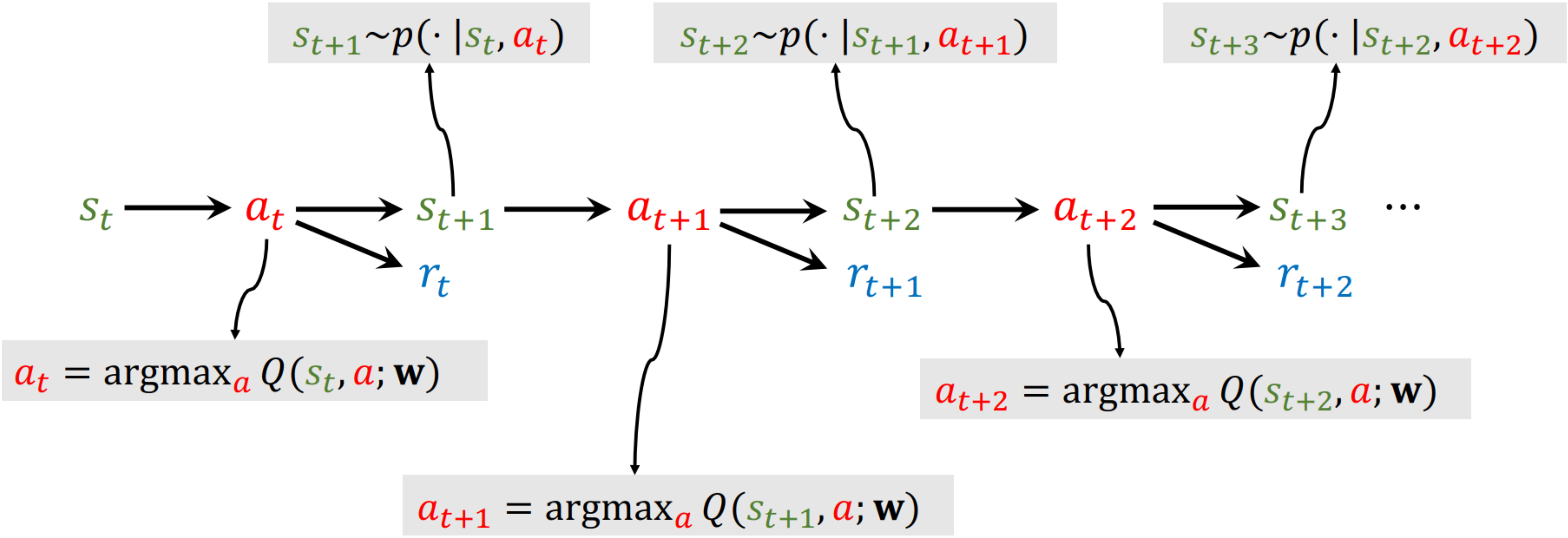
Deep Q Network (DQN)

- **Question**: Based on the predictions, what should be the action?



Apply DQN to Play Game

- Workflow



How to Train DQN?

- Recall

$$
\begin{aligned}
U_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots \\
&= R_t + \gamma (R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots) \\
&= R_t + \gamma U_{t+1}
\end{aligned}
$$

- DQN's output, $Q(s_t, a_t; w)$, is an estimate of U_t
- DQN's output, $Q(s_{t+1}, a_{t+1}; w)$, is an estimate of U_{t+1}

How to Train DQN?

- Recall

$$\begin{aligned} U_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots \\ &= R_t + \gamma (R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots) \\ &= R_t + \gamma U_{t+1} \end{aligned}$$

- DQN's output, $Q(s_t, a_t; w)$, is an estimate of U_t
- DQN's output, $Q(s_{t+1}, a_{t+1}; w)$, is an estimate of U_{t+1}
- Thus

$$Q(s_t, a_t; w) \approx \mathbb{E}[R_t + \gamma Q(s_{t+1}, a_{t+1}; w)]$$

How to Train DQN?

- From previous slide…

$$Q(s_t, a_t; w) \approx \mathbb{E}[R_t + \gamma Q(s_{t+1}, a_{t+1}; w)]$$

- An alternative perspective…

$$\underbrace{Q(s_t, a_t; w)}_{\text{Prediction}} \approx \underbrace{r_t + \gamma Q(s_{t+1}, a_{t+1}; w)]}_{\text{Target (temporal difference, TD)}}$$

Prediction

Target (temporal difference, TD)

Next lecture

How to Train DQN?

- Train DQN using TD learning
- Prediction: $Q(s_t, a_t; w)$
- TD Target

$$\begin{aligned} y_t &= r_t + \gamma Q(s_{t+1}, a_{t+1}; w_t) \\ &= r_t + \gamma \max_a Q(s_{t+1}, a; w_t) \end{aligned}$$

- Loss:

$$L_t = \frac{1}{2}[Q(s_t, a_t; w) - y_t]^2$$

- Gradient descent:

$$w_{t+1} = w_t - \alpha \frac{\partial L_t}{\partial w}\bigg|_{w=w_t}$$

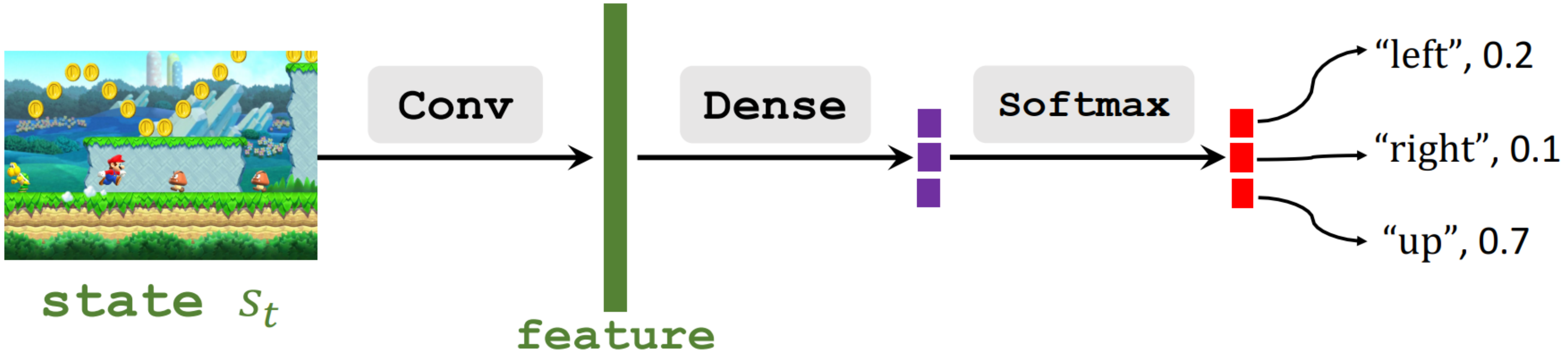
Policy-Based Reinforcement Learning

Policy Function $\pi(a|s)$

- Policy function $\pi(a|s)$ is a probability density function
- It takes state s as input
- In output the probabilities for all the actions, e.g.

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

- Randomly sample action a drawn from this distribution



Policy Function $\pi(a|s)$

- Question: Can We Directly Learn Policy Function $\pi(a|s)$?

Can We Directly Learn Policy Function $\pi(a|s)$?

- If there are only a few states and actions, then yes, we can
- Draw a table (matrix) and learn the entries

	Action a_1	Action a_2	Action a_3	Action a_4	$\cdots$
State s_1					
State s_2					
State s_3					
$\vdots$					

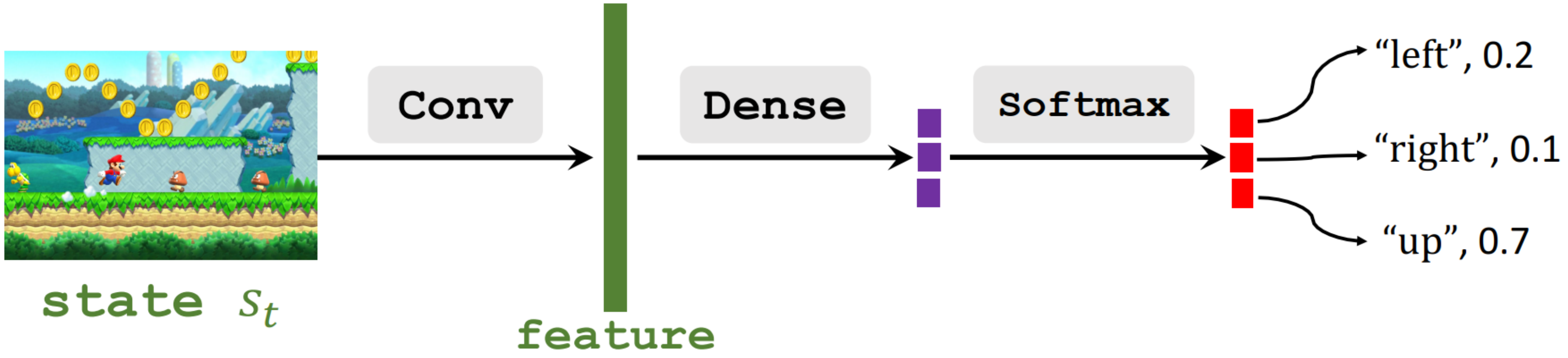
Can We Directly Learn Policy Function $\pi(a|s)$?

- If there are only a few states and actions, then yes, we can
- Draw a table (matrix) and learn the entries
- What if there are too many (or infinite) states or actions?

	Action a_1	Action a_2	Action a_3	Action a_4	$\cdots$
State s_1					
State s_2					
State s_3					
$\vdots$					

Policy Function $\pi(a|s;\theta)$

- Policy network: use a neural network to approximate $\pi(a|s)$
- Use policy network $\pi(a|s;\theta)$ to approximate $\pi(a|s)$
- θ are trainable parameters of the neural network
- $\sum_{a\in\mathcal{A}} \pi(a|s;\theta) = 1$, where $\mathcal{A} = \{\text{left}, \text{right}, \text{up}\}$ is set of all actions



Comparison with Value-Based Learning

- Return (discounted)

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

- Action-value function for policy π

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

- Optimal action-value function

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

- Goal of **value-based learning** is to learn or approximate $Q^*(s_t, a_t)$

Comparison with Value-Based Learning

- Return (discounted)

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

- Action-value function for policy π

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

- Optimal action-value function

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

- State-value function

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)]$$

Integrate out action $A \sim \pi(\cdot\,|s_t)$

Comparison with Value-Based Learning

- Return (discounted)

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

- Action-value function for policy π

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

- Optimal action-value function

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

- State-value function

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$$

Integrate out action $A \sim \pi(\cdot\,|s_t)$

NC STATE UNIVERSITY

Biomechatronics and Intelligent Robotics (BIRO) Lab

Comparison with Value-Based Learning

- Return (discounted)

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

- Action-value function for policy π

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

- State-value function

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$$

- Approximate policy function $\pi(a|s_t)$ by policy network $\pi(a|s_t; \theta)$
- Approximate value function $V_\pi(s_t)$ by

$$V(s_t; \theta) = \sum_a \pi(a|s_t; \theta) \cdot Q_\pi(s_t, a)$$

Comparison with Value-Based Learning

- Return (discounted)

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

- Action-value function for policy π

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

- State-value function

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$$

- Approximate policy function $\pi(a|s_t)$ by policy network $\pi(a|s_t; \theta)$
- Approximate value function $V_\pi(s_t)$ by

$$V(s_t; \theta) = \sum_a \pi(a|s_t; \theta) \cdot Q_\pi(s_t, a)$$

- Goal of **policy-based learning** is to learn θ that maximizes $J(\theta) = \mathbb{E}_S[V(S; \theta)]$

Let’s check the code…