10703 Recitation 3

Actor-Critic Methods and Value-Based Methods

Gene Yang

Recap

Definition: The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π:

- $\mathrm{v}_\pi(s) = \mathbb{E}[G_t \,|\, S_t = s]$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy:

- $\mathrm{q}_\pi(s, a) = \mathbb{E}[G_t \,|\, S_t = s, A_t = a]$

Learning V with DP

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(\textit{terminal})$ to 0

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Improving π with DP

Policy Improvement
policy-stable $\leftarrow$ *true*
For each $s \in \mathcal{S}$:
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
    If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(\mathit{terminal}) \doteq 0$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\quad \Delta \leftarrow 0$
| $\quad$ Loop for each $s \in \mathcal{S}$:
| $\quad\quad v \leftarrow V(s)$
| $\quad\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$
| $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\quad \pi(s) = \arg\max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$

Q-Learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

    until S is terminal

SARSA

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \underbrace{\boxed{\gamma Q(S',A')}}_{\text{on-policy}} - Q(S,A)]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Recap

Definition: The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π:

Deep RL
$V_\pi(s; \boldsymbol{\Phi_1})$ or $V_{\Phi 1}^{\pi}(s)$

- $v_\pi(s) = \mathbb{E}[G_t \,|\, S_t = s]$

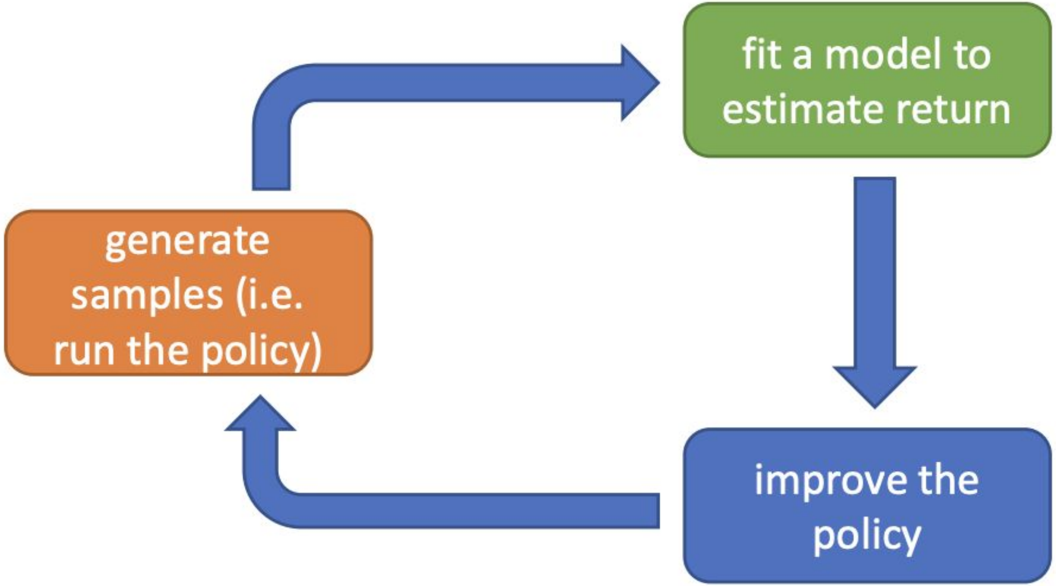
The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy:

Deep RL
$Q_\pi(s, a; \boldsymbol{\Phi_2})$ or $Q_{\Phi 2}^{\pi}(s,a)$

- $q_\pi(s, a) = \mathbb{E}[G_t \,|\, S_t = s, A_t = a]$

Actor-Critic

$$\text{fit } \hat{V}^{\pi}$$



$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Advantage Actor-Critic

0. Initialize policy parameters θ and critic parameters ϕ .

1. Sample trajectories $\{\tau_i = \{s_t^i, a_t^i\}_{t=0}^{T}\}$ by deploying the current policy $\pi_\theta(a_t \,|\, s_t)$.

2. Fit value function $V_\phi^\pi(s)$ by MC or TD estimation (update ϕ)

3. Compute action advantages $A^\pi(s_t^i, a_t^i) = R(s_t^i, a_t^i) + \gamma V_\phi^\pi(s_{t+1}^i) - V_\phi^\pi(s_t^i)$

4. $\nabla_\theta U(\theta) \approx \hat{g} = \dfrac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\alpha_t^i \,|\, s_t^i) A^\pi(s_t^i, a_t^i)$

5. $\theta \leftarrow \theta + \alpha \nabla_\theta U(\theta)$

Advantage Actor-Critic

0. Initialize policy parameters θ and critic parameters ϕ .

1. Sample trajectories $\{\tau_i = \{s_t^i, a_t^i\}_{t=0}^T\}$ by deploying the current policy $\pi_\theta(a_t \mid s_t)$.

2. Fit value function $V_\phi^\pi(s)$ by MC or TD estimation (update ϕ)

3. Compute action advantages $A^\pi(s_t^i, a_t^i) = \boxed{R(s_t^i, a_t^i) + \gamma V_\phi^\pi(s_{t+1}^i)} - V_\phi^\pi(s_t^i)$

4. $\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) A^\pi(s_t^i, a_t^i)$

biased, lower variance

5. $\theta \leftarrow \theta + \alpha \nabla_\theta U(\theta)$

DQN

Algorithm 1 - Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $50,000$ following a random policy.
Initialize action-value function Q with random weights θ
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
    Execute action a_t and observe reward r_t and state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
    Sample random minibatch of (s_i, a_i, r_i, s_{i+1}) of size N from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ using Adam
  end for
end for

DQN

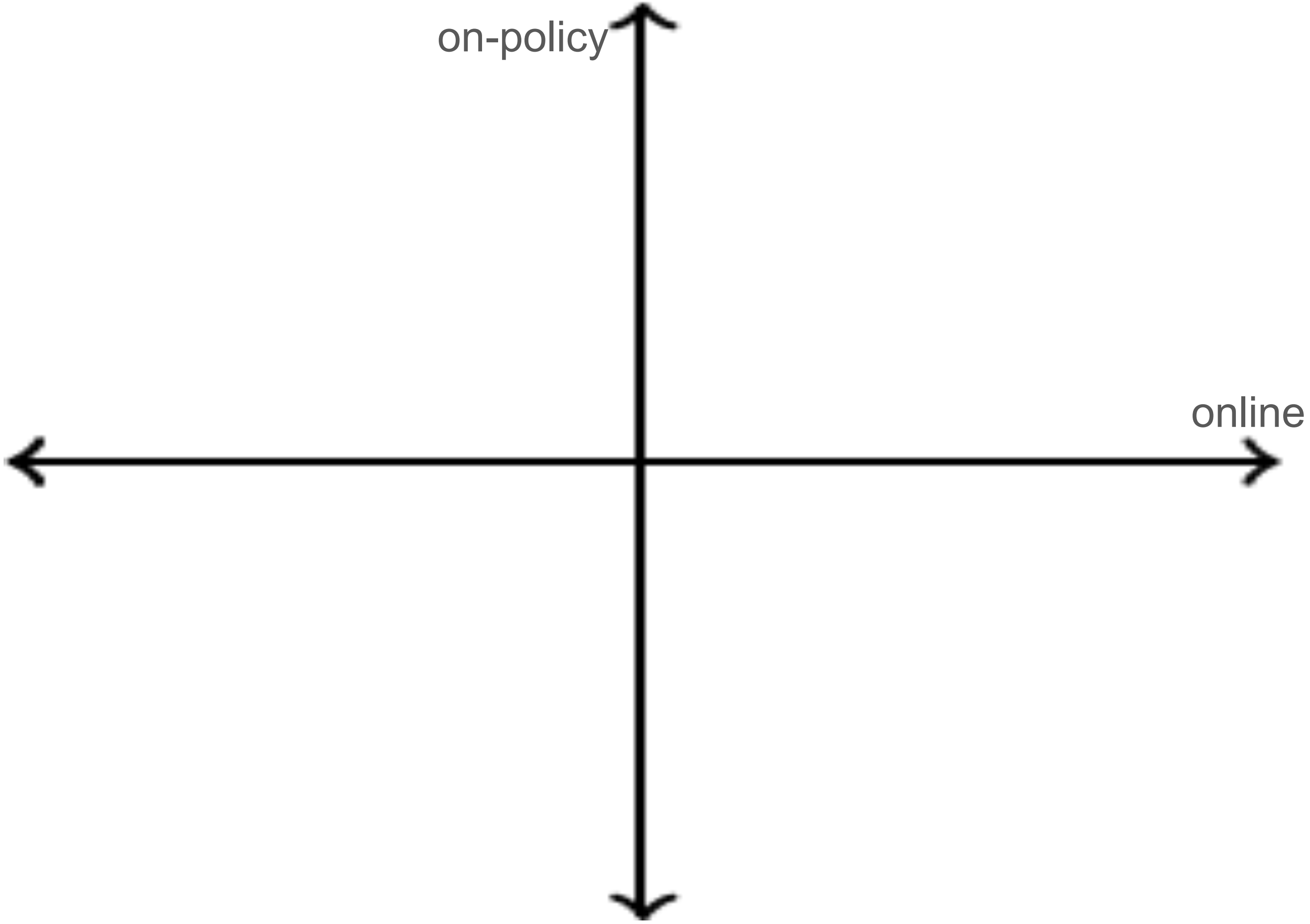
Algorithm 1 - Deep Q-learning with Experience Replay

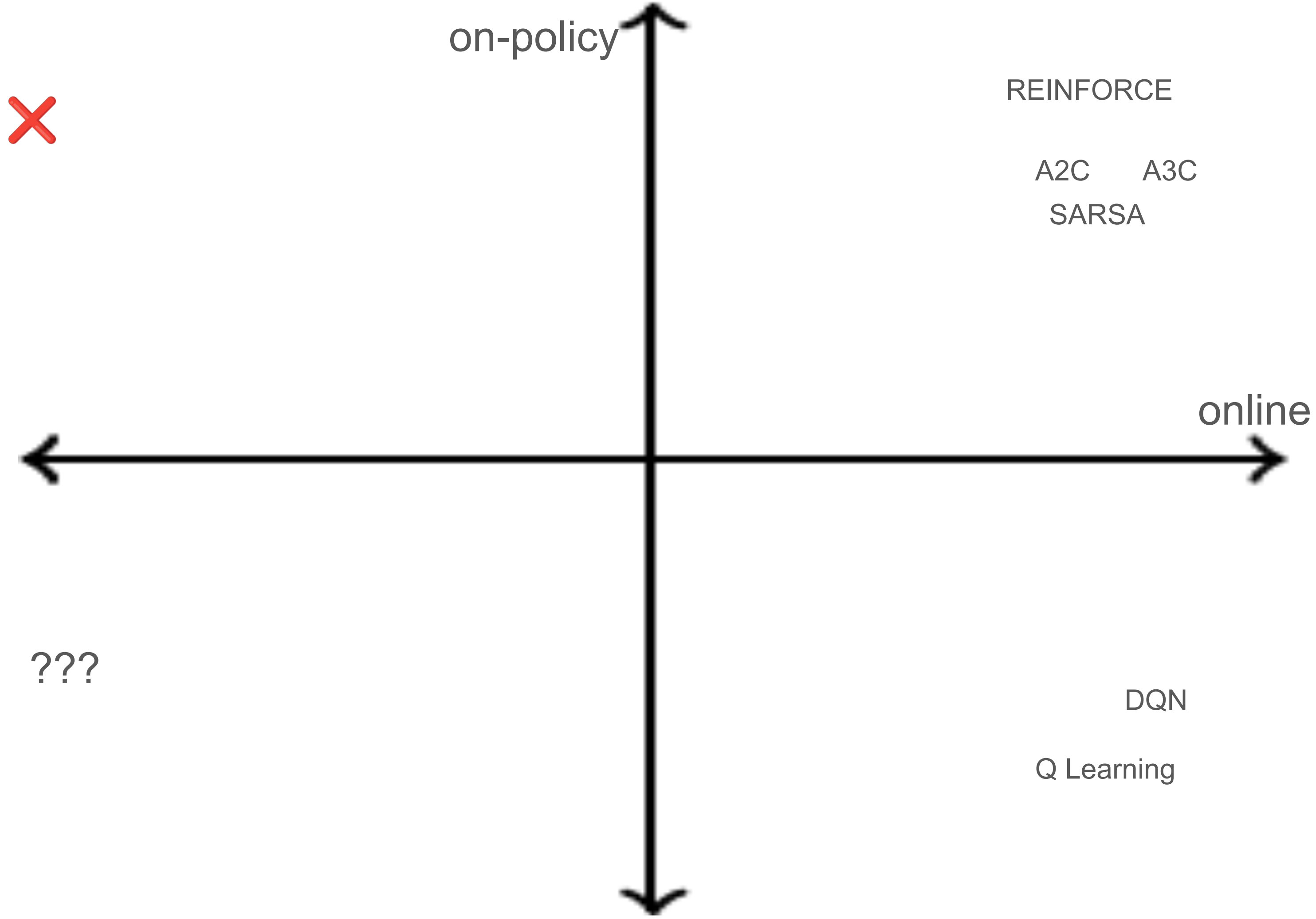
```
Initialize replay memory 𝒟 to capacity 50,000 following a random policy.
Initialize action-value function Q with random weights θ
for episode = 1, M do
    for t = 1, T do
        With probability ε select a random action a_t
        otherwise select a_t = argmax_a Q(φ(s_t), a; θ)
        Execute action a_t and observe reward r_t and state s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in 𝒟.
        Sample random minibatch of (s_i, a_i, r_i, s_{i+1}) of size N from 𝒟
        Set y_j = { r_j                                  for terminal s_{j+1}
                  { r_j + γ max_{a'} Q(s_{j+1}, a'; θ)   for non-terminal s_{j+1}
        Perform a gradient descent step on (y_j − Q(s_j, a_j; θ))^2 using Adam
    end for
end for
```

decorrelate

$$y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$$

- REINFORCE
- Q Learning
- A2C
- A3C
- SARSA
- DQN




on-policy
REINFORCE
A2C
A3C
SARSA
online
???
DQN
Q Learning