Experience Replay

Shusen Wang

http://wangshusen.github.io/

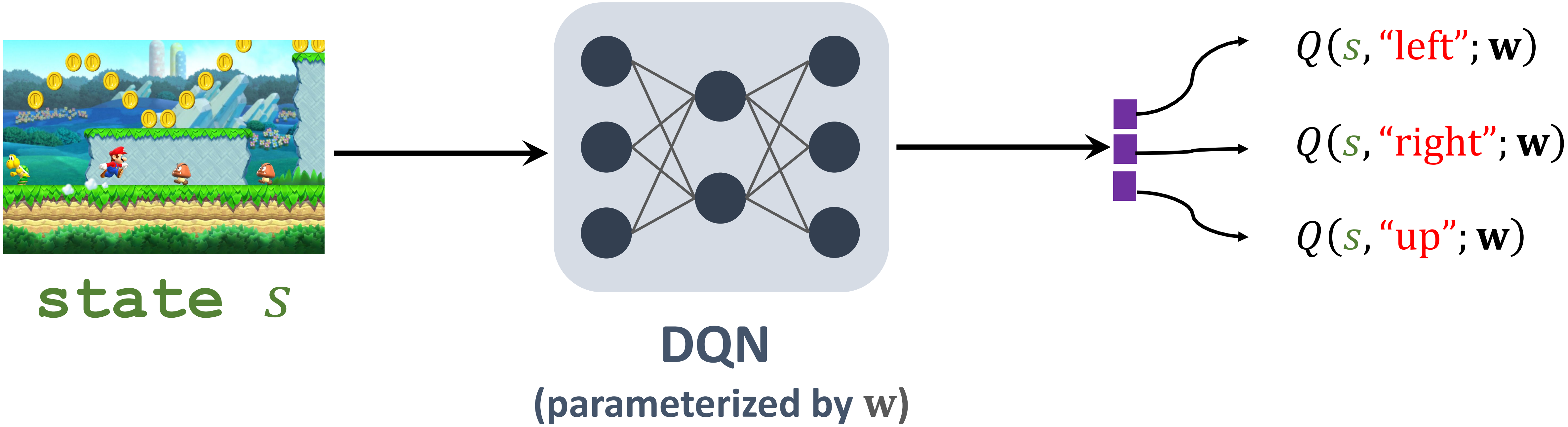
Revisiting DQN and TD Learning

Deep Q Network (DQN)

Approximate the optimal action-value function, $Q^{\star}(s, a)$, by $Q(s, a; \mathbf{w})$.

Deep Q Network (DQN)

Approximate the optimal action-value function, $Q^\star(s, a)$, by $Q(s, a; \mathbf{w})$.



Temporal Difference (TD) Learning

- Observe state s_t and perform action a_t.
- Environment provides new state s_{t+1} and reward r_t.
- **TD target:** $y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w})$.

Temporal Difference (TD) Learning

- Observe state s_t and perform action a_t.
- Environment provides new state s_{t+1} and reward r_t.
- **TD target:** $y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w})$.
- **TD error:** $\delta_t = q_t - y_t$, where $q_t = Q(s_t, a_t; \mathbf{w})$.
- **Goal:** Make q_t close to y_t, for all t. (Equivalently, make δ_t^2 small.)

Temporal Difference (TD) Learning

- **TD error:** $\delta_t = q_t - y_t$, where $q_t = Q(s_t, a_t; \mathbf{w})$.
- **TD learning:** Find $\mathbf{w}$ by minimizing $L(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T} \frac{\delta_t^2}{2}$.

Temporal Difference (TD) Learning

- **TD error:** $\delta_t = q_t - y_t$, where $q_t = Q(s_t, a_t; \mathbf{w})$.
- **TD learning:** Find $\mathbf{w}$ by minimizing $L(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T} \frac{\delta_t^2}{2}$.
- **Online gradient descent:**
  - Observe (s_t, a_t, r_t, s_{t+1}) and compute δ_t.
  - Compute gradient: $\mathbf{g}_t = \frac{\partial\, \delta_t^2/2}{\partial\, \mathbf{w}} = \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$
  - Gradient descent: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \mathbf{g}_t$.

Temporal Difference (TD) Learning

- **TD error:** $\delta_t = q_t - y_t$, where $q_t = Q(s_t, a_t; \mathbf{w})$.
- **TD learning:** Find $\mathbf{w}$ by minimizing $L(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T} \frac{\delta_t^2}{2}$.
- **Online gradient descent.**
- Discard (s_t, a_t, r_t, s_{t+1}) after using it.

Shortcoming 1: Waste of Experience

- **A transition:** (s_t, a_t, r_t, s_{t+1}).
- **Experience:** all the transitions, for $t = 1, 2, \cdots$.
- Previously, we discard (s_t, a_t, r_t, s_{t+1}) after using it.
- It is a waste…

Shortcoming 2: Correlated Updates

- Previously, we use (s_t, a_t, r_t, s_{t+1}) sequentially, for $t = 1, 2, \cdots$, to update $\mathbf{w}$.
- Consecutive states, s_t and s_{t+1}, are strongly correlated (which is bad.)

Experience Replay

Experience Replay

- **A transition:** (s_t, a_t, r_t, s_{t+1}) .
- Store recent n transitions in a **replay buffer**.

$$\left.\begin{array}{c} \cdots \\ (s_t, a_t, r_t, s_{t+1}) \\ (s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}) \\ (s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}) \\ \cdots \end{array}\right\}$$

Replay Buffer
(n transitions)

Experience Replay

- **A transition:** (s_t, a_t, r_t, s_{t+1}) .
- Store recent n transitions in a **replay buffer**.
- Remove old transitions so that the buffer has at most n transitions.
- Buffer capacity n is a tuning hyper-parameter [1, 2].
    - n is typically large, e.g., $10^5 \sim 10^6$.
    - The setting of n is application-specific.

Reference:

1. Zhang & Sutton. A deeper look at experience replay. In *NIPS workshop*, 2017.
2. Fedus et al. Revisiting fundamentals of experience replay. In *ICML*, 2019.

TD with Experience Replay

- Find $\mathbf{w}$ by minimizing $L(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T}\frac{\delta_t^2}{2}$.
- Stochastic gradient descent (SGD):
  - Randomly sample a transition, (s_i, a_i, r_i, s_{i+1}), from the buffer.
  - Compute TD error, δ_i.
  - Stochastic gradient: $\underline{\mathbf{g}_i} = \underline{\frac{\partial\, \delta_i^2/2}{\partial\, \mathbf{w}}} = \delta_i \cdot \frac{\partial\, Q(s_i, a_i; \mathbf{w})}{\partial\, \mathbf{w}}$
  - SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \mathbf{g}_i$.

Benefits of Experience Replay

1. Make the updates uncorrelated.

2. Reuse collected experience many times.

History

- Experience replay was proposed by Long-Ji Lin [1].
- The DQN paper [2] popularized experience replay.
- There are many improvements, e.g., [3].

Reference:

1. Lin. Reinforcement Learning for Robots Using Neural Networks. *PhD Dissertation*, 1993.
2. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
3. Schaul et al. Prioritized experience replay. In *ICLR*, 2016.

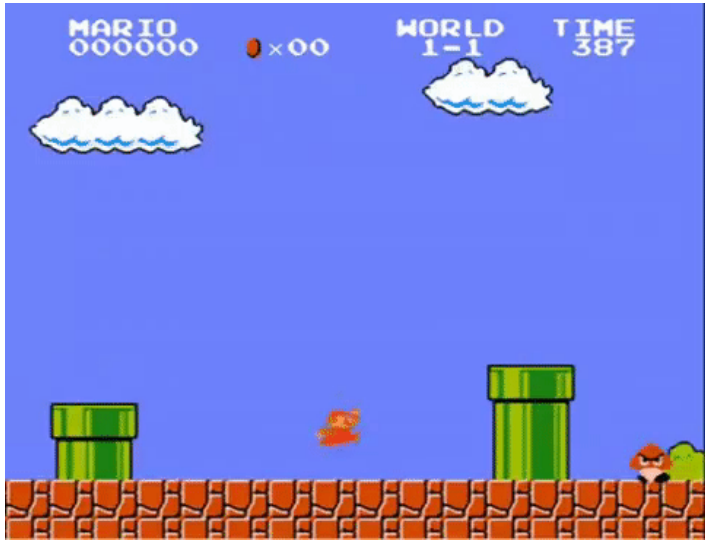
Prioritized Experience Replay

Reference:

1. Schaul, Quan, Antonoglou, & Silver. Prioritized experience replay. In *ICLR*, 2016.

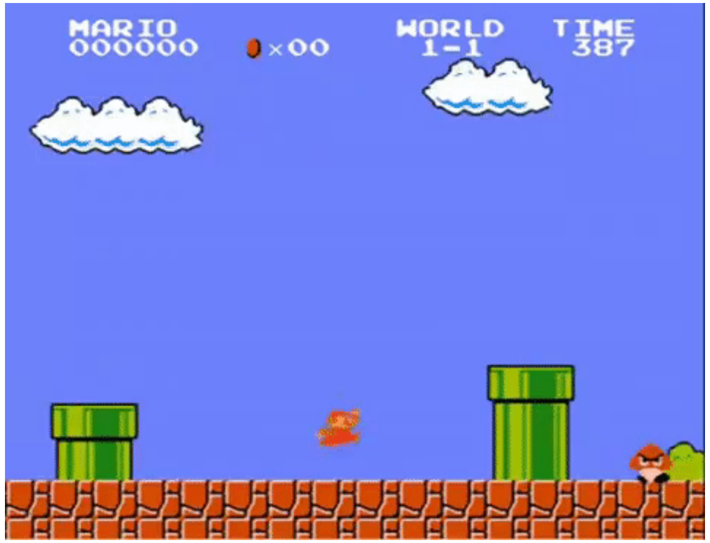
Basic Idea

- Not all transitions are equally important.
- Which kind of transition is more important, left or right?



Basic Idea

- How do we know which transition is important?
- If a transition has high TD error $|\delta_t|$, it will be given high priority.



Importance Sampling

- Use importance sampling instead of uniform sampling.
- Option 1: Sampling probability $p_t \propto |\delta_t| + \epsilon$.

Importance Sampling

- Use importance sampling instead of uniform sampling.
- Option 1: Sampling probability $p_t \propto |\delta_t| + \epsilon$.
- Option 2: Sampling probability $p_t \propto \frac{1}{\mathrm{rank}(t)}$.
  - The transitions are sorted so that $|\delta_t|$ is in the descending order.
  - $\mathrm{rank}(t)$ is the rank of the t-th transition.
- In sum, big $|\delta_t|$ shall be given high priority.

Scaling Learning Rate

- SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \mathbf{g}$, where α is the learning rate.
- If uniform sampling is used, α is the same for all transitions.
- If importance sampling is used, α shall be adjusted according to the importance.

Scaling Learning Rate

- Scale the learning rate by $(n\, p_t)^{-\beta}$, where $\beta \in (0,1)$.
- If $p_1 = \cdots = p_n = \frac{1}{n}$ (uniform sampling), the scaling factor is equal to 1.
- High-importance transitions (with high p_t) have low learning rates.
- In the beginning, set β small; increase β to 1 over time.

Update TD Error

- Associate each transition, (s_t, a_t, r_t, s_{t+1}), with a TD error, δ_t.
- If a transition is newly collected, we do not know its δ_t.
    - Simply set its δ_t to the maximum.
    - It has the highest priority.
- Each time (s_t, a_t, r_t, s_{t+1}) is selected from the buffer, we update its δ_t.

Transitions	Sampling Probabilities	Learning Rates
$\cdots$	$\cdots$	$\cdots$
$(s_t, a_t, r_t, s_{t+1}), \delta_t$	$p_t \propto \lvert\delta_t\rvert + \epsilon$	$\alpha \cdot (n\, p_t)^{-\beta}$
$(s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}), \delta_{t+1}$	$p_{t+1} \propto \lvert\delta_{t+1}\rvert + \epsilon$	$\alpha \cdot (n\, p_{t+1})^{-\beta}$
$(s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}), \delta_{t+2}$	$p_{t+2} \propto \lvert\delta_{t+2}\rvert + \epsilon$	$\alpha \cdot (n\, p_{t+2})^{-\beta}$
$\cdots$	$\cdots$	$\cdots$

Transitions	Sampling Probabilities	Learning Rates
$\cdots$	$\cdots$	$\cdots$
$(s_t, a_t, r_t, s_{t+1}), \delta_t$	$p_t \propto \lvert\delta_t\rvert + \epsilon$	$\alpha \cdot (n\, p_t)^{-\beta}$
$(s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}), \delta_{t+1}$	$p_{t+1} \propto \lvert\delta_{t+1}\rvert + \epsilon$	$\alpha \cdot (n\, p_{t+1})^{-\beta}$
$(s_{t+2}, a_{t+2}, r_{t+2}, s_{t+3}), \delta_{t+2}$	$p_{t+2} \propto \lvert\delta_{t+2}\rvert + \epsilon$	$\alpha \cdot (n\, p_{t+2})^{-\beta}$
$\cdots$	$\cdots$	$\cdots$

Big $|\delta_t|$ ==> High probability ==> Small learning rate

Thank you!