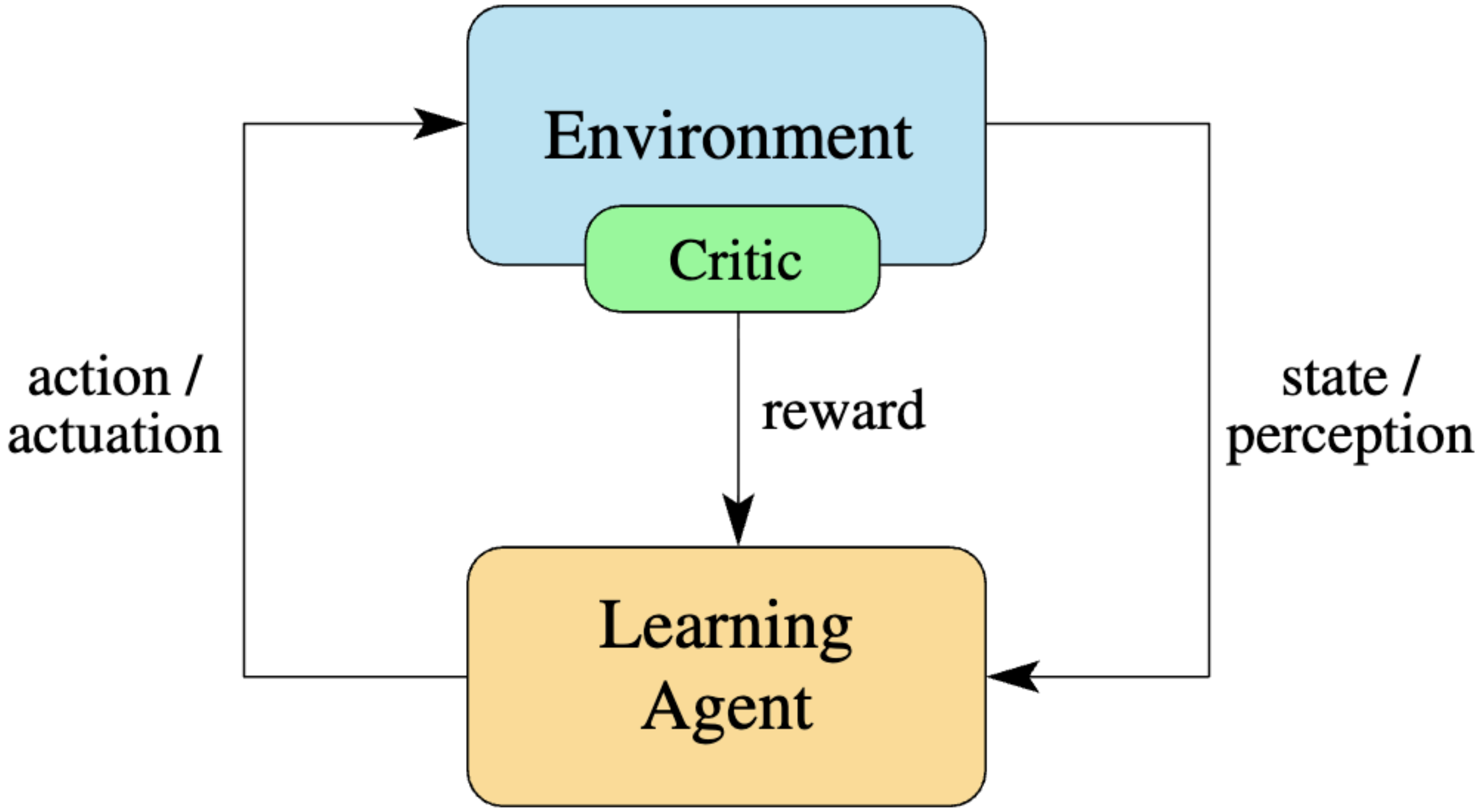
Reinforcement Learning - 3

Recap

- **Last class.**
  - Markov Process + Reward + Decision
  - Evaluating the policy
    - Value function & Q-function
  - <u>Assumption</u>. Full knowledge of MDP:
    - the state transition $p(s' \mid s, a)$
    - the reward $r(s, a)$
- **Today.**
  - Evaluating the policy without full knowledge

Recap: Reinforcement learning

- We were interested in a sequential decision-making scenario
  - State transitions as some Markov process $s_{t+1} \sim p(\cdot \mid s_t, a_t)$
  - Agent makes actions according to some policy $a_t \sim \pi(\cdot \mid s_t)$
  - Agent collects the reward $r_t \sim r(\cdot \mid s_t, a_t)$



for $t = 1, \ldots, n$ **do**
    The agent perceives state s_t
    The agent performs action a_t
    The environment evolves to s_{t+1}
    The agent receives reward r_t
end for

Recap: Evaluation of policy

- We were interested in the return — i.e., accumulated discounted reward

$$G_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{H-1} r_{t+H-1}$$

- We discussed how we can evaluate the policy π by computing:
  - **Value function.** Expected return, beginning at some state

$$V^\pi(s) = \mathbb{E}[r_t + \gamma r_{t+1} + \cdots + \gamma^{H-1} r_{t+H-1} \mid s_t = s]$$

  - **Q-function.** Expected return from making an action at some state, assuming that we'll continue using the policy π afterwards

$$Q^\pi(s, a) = \mathbb{E}[G_t \,|\, r_t + \gamma r_{t+1} + \cdots + \gamma^{H-1} r_{t+H-1} \mid s_t = s, a_t = a]$$

Recap: Evaluation of policy

- We can use Bellman equation to simplify this for infinite horizon
  - **Value function**

$$V^{\pi}(s) = \sum_{a \in A} \pi(a \mid s)\left(R(a, s) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi}(s)\right)$$

  - **Q-function**

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi}(s')$$

- Can be computed by iterative method
  - Requires knowing $R(s, a)$ and $p(s' \mid s, a)$

Evaluation through experience

- Today, we assume that we don't know $R(s, a)$ or $p(s' \mid s, a)$

- Instead, we assume that we can collect data from environment
  - Deploy an agent that uses $\pi(a \mid s)$, and collect

$$e = (s_1, a_1, r_1, s_2, a_2, r_2 \ldots)$$

  - Collect multiple episodes $e_1, e_2, \ldots, e_n$

- **Goal.** Estimate $V^\pi(s)$ from $e_1, \ldots, e_n$

Monte Carlo policy evaluation

- **Idea.** Simply measure the mean return, starting from each state
  - Look at each sub-trajectory
    - At each starting state s_t, compute the G_t
  - Average over all sub-trajectories with the same starting state

- There are two well-known variants
  - First-visit
  - Every-visit

First-visit Monte Carlo

- Initialize $N(s) = 0, G(s) = 0 \quad \forall s \in S$
- Loop:
  - Sample episode i = $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
  - Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T_i - 1} r_{i,T_i}$ as the return from time step t onwards in i-th episode
  - For each time step t until T_i – If this is the first time t that state s is visited in episode i
    - Increment the counter $N(s) \leftarrow N(s) + 1$
    - Increment the total return $G(s) \leftarrow G(s) + G_{i,t}$
    - Update the estimate $\hat{V}^{\pi}(s) = G(s)/N(s)$

Every-visit Monte Carlo

- Initialize $N(s) = 0, G(s) = 0 \quad \forall s \in S$
- Loop:
  - Sample episode i = $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
  - Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T_i-1} r_{i,T_i}$ as the return from time step t onwards in i-th episode
  - For each time step t until T_i:
    - Increment the counter $N(s) \leftarrow N(s) + 1$
    - Increment the total return $G(s) \leftarrow G(s) + G_{i,t}$
    - Update the estimate $\hat{V}^{\pi}(s) = G(s)/N(s)$

Every-visit Monte Carlo

- Note that we can think of an equivalent, sequential-update version
  - Natural, as we usually collect data in sequence rather than in batch

- For state s visited at time step t in episode i,
  - Increment the counter $N(s) \leftarrow N(s) + 1$
  - Update the estimate:

$$\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s) + \frac{1}{N(s)}(G_{i,t} - \hat{V}^{\pi}(s))$$

- Can be viewed as iterative updates
  - $1/N(s)$ is some learning rate

Properties

- **First-visit**
  - $\hat{V}^\pi(s)$ is unbiased: $\mathbb{E}[\hat{V}^\pi(s)] = V^\pi(s)$
  - $\hat{V}^\pi(s)$ is consistent: $\lim_{n\to\infty} \hat{V}^\pi(s) = V^\pi(s)$
- **Every-visit**
  - $\hat{V}^\pi(s)$ is biased: $\mathbb{E}[\hat{V}^\pi(s)] \neq V^\pi(s)$
  - $\hat{V}^\pi(s)$ is consistent: $\lim_{n\to\infty} \hat{V}^\pi(s) = V^\pi(s)$
  - Often has a smaller MSE:

$$\mathbb{E}[\|\hat{V}^\pi_{\mathrm{ev}} - V^\pi\|^2] \leq \mathbb{E}[\|\hat{V}^\pi_{\mathrm{fv}} - V^\pi\|^2]$$

Properties

- Both first-visit and every-visit Monte Carlo estimate tend to have a large variance, i.e.,

$$\mathbb{E}[\|\hat{V}^{\pi}_{\mathrm{ev}} - V^{\pi}\|^2] \gg 0$$

- Requires many samples for convergence
  - In RL, the data collection is usually very expensive
    - Example. Robot Learning
    - Example. Drug discovery

- Requires the episode to terminate before updating

Temporal Difference Learning

- **Idea.** Use recursive estimates, just like Bellman's equation

$$V^{\pi}(s) = \sum_{a \in A} \pi(a \mid s)\left(R(a,s) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi}(s)\right)$$

  - We can use the current estimates of $\hat{V}^{\pi}(s_{t+1})$, instead of waiting for the episode to be fully unrolled

- **TD(0) Learning.** At every transition (s_t, a_t, r_t, s_{t+1}), do:

$$\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s_t) + \alpha\left([r_t + \gamma \hat{V}^{\pi}(s_{t+1})] - \hat{V}^{\pi}(s_t)\right)$$

  - α is some "learning rate"

Properties

- Biased, in general
- Usually has a smaller variance and is more sample-efficient
- Requires less storage, and updates online

Wrapping up

So far

- We have looked at various ML algorithms:
  - From small-scale to large-scale methods

- **Before mid-term: Classic ML**
  - Empirical Risk Minimization vs. Bayesian Approach
  - Supervised Learning
    - Linear models, SVM, Nonlinear methods (based on kernels)
  - Unsupervised Learning
    - Clustering, GMM, PCA

So far

- **After mid-term: Modern ML**
  - Deep Learning & Optimization techniques
    - SGD and Backpropagation
  - Vision
    - Convolutions, Self-supervised Learning VAE, GAN, Diffusion,
  - Language
    - Tokenization, Language models, Post-training, Multimodal AI
  - Reinforcement Learning
    - Markov Decision Process, Temporal Difference Learning

What is missing?

- We have focused on fundamental topics, mostly
  - Stayed away from research frontiers
- Here is a very brief look at some "emerging topics"
  - **Dynamic world model.** https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/
  - **Weather forecasting.** https://blog.google/technology/google-deepmind/weathernext-2/
- Common: Want to encode "physical understanding" to save computations and training data needed

</lecture 25>