

@zblu Reinforcement Learning Theory

INF8250AE Course Notes

RL 5: Learning from a Stream of Data 1: Reward Learning

Stochastic Approximation · Online Mean Estimation · Reward Function Learning · Exploration–Exploitation Tradeoff

Key Topics: Stochastic Approximation, Online Mean Estimation, Reward Function Learning, Exploration – Exploitation Tradeoff

Reference: <https://amfarahmand.github.io/IntroRL/>

Notes on Reinforcement Learning – Nov 1, 2025



Reinforcement Learning 5: Learning from a Stream of Data 1: Reward Learning

Zhuobie

2025 年 11 月 17 日

1. Learning from a Stream of Data: Reward Learning

This chapter studies how to estimate reward functions and solve control when the model is unknown and data arrive online. We cover Monte Carlo (MC), Temporal Difference (TD), and their control variants (Q-Learning, SARSA). Stochastic Approximation (SA) provides the convergence framework.

1.1 RL Setting and Data Stream

We observe an online sequence

$$X_1, A_1, R_1, X_2, A_2, R_2, \dots,$$

with $A_t \sim \pi(\cdot|X_t)$, $X_{t+1} \sim P(\cdot|X_t, A_t)$, $R_t \sim R(\cdot|X_t, A_t)$. The goal is to learn V^π, Q^π and eventually V^*, Q^* (and π^*). Throughout, we focus on the tabular/finite MDP case.

2. Online Estimation of the Mean of a Random Variable

2.1 Sample Average Estimator and Its Properties

Consider i.i.d. random variables $Z_1, \dots, Z_t \sim \nu$ with mean $m = \mathbb{E}[Z]$ and variance $\sigma^2 = \text{Var}(Z)$. The *sample average estimator* is

$$m_t = \frac{1}{t} \sum_{i=1}^t Z_i.$$

Unbiasedness. By linearity of expectation,

$$\mathbb{E}[m_t] = \frac{1}{t} \sum_{i=1}^t \mathbb{E}[Z_i] = m,$$

so m_t is an unbiased estimator of m .

Variance. Using the independence of $\{Z_i\}$,

$$\text{Var}(m_t) = \mathbb{E} \left[\left(\frac{1}{t} \sum_{i=1}^t (Z_i - \mu) \right)^2 \right] = \frac{1}{t^2} \mathbb{E} \left[\sum_{i,j=1}^t (Z_i - \mu)(Z_j - \mu) \right].$$

Split into diagonal and off-diagonal parts:

$$\frac{1}{t^2} \left(\sum_{i=1}^t \mathbb{E}[(Z_i - \mu)^2] + \sum_{i \neq j} \mathbb{E}[(Z_i - \mu)(Z_j - \mu)] \right) = \frac{1}{t^2} (t\sigma^2 + 0) = \boxed{\frac{\sigma^2}{t}}.$$

Hence the dispersion of m_t around m decreases as $1/t$.

2.2 Concentration via the Weak Law of Large Numbers

Applying Markov's inequality to the non-negative random variable $|m_t - m|^2$, for any $\varepsilon > 0$ we have

$$\mathbb{P}\{|m_t - m| > \varepsilon\} = \mathbb{P}\{|m_t - m|^2 > \varepsilon^2\} \leq \frac{\mathbb{E}[|m_t - m|^2]}{\varepsilon^2} = \frac{\text{Var}(m_t)}{\varepsilon^2} = \frac{\sigma^2}{t\varepsilon^2}.$$

As $t \rightarrow \infty$, this upper bound tends to 0:

$$\boxed{m_t \xrightarrow{\mathbb{P}} m.}$$

This is the **weak Law of Large Numbers (LLN)**. Under stronger moment assumptions, we even have $m_t \rightarrow m$ almost surely (the strong LLN).

2.3 How to Get an Online Estimator

A direct computation of m_t requires storing all Z_i , which is impractical when t is large. We can rewrite the recursion as

$$m_{t+1} = \frac{1}{t+1} \sum_{i=1}^{t+1} Z_i = \left(1 - \frac{1}{t+1}\right) m_t + \frac{1}{t+1} Z_{t+1}.$$

Defining the step size $\alpha_t = \frac{1}{t+1}$,

$$\boxed{m_{t+1} = (1 - \alpha_t) m_t + \alpha_t Z_{t+1}.}$$

This is the prototype of a *stochastic approximation (SA)* update.

2.4 Stochastic Approximation View

More generally, consider

$$\theta_{t+1} = (1 - \alpha_t) \theta_t + \alpha_t Z_t.$$

Case 1: Constant step size $\alpha_t = \alpha \in (0, 1)$. Taking expectations,

$$\bar{\theta}_{t+1} = (1 - \alpha) \bar{\theta}_t + \alpha m \Rightarrow \bar{\theta}_t = m [1 - (1 - \alpha)^t] \rightarrow m.$$

Thus $\mathbb{E}[\theta_t] \rightarrow m$. However, the variance satisfies

$$\text{Var}(\theta_{t+1}) = (1 - \alpha)^2 \text{Var}(\theta_t) + \alpha^2 \sigma^2 \Rightarrow \lim_{t \rightarrow \infty} \text{Var}(\theta_t) = \frac{\alpha \sigma^2}{2 - \alpha}.$$

Proof

For $\theta_{t+1} = (1 - \alpha)\theta_t + \alpha Z_t$ with $\mathbb{E}[Z_t] = m$, define $e_t = \theta_t - m$ and $v_t = \text{Var}(e_t)$:

$$v_{t+1} = (1 - \alpha)^2 v_t + \alpha^2 \sigma^2.$$

Solving this recursion,

$$v_t = (1 - \alpha)^{2t} v_0 + \frac{\alpha \sigma^2}{2 - \alpha} (1 - (1 - \alpha)^{2t}), \quad \Rightarrow \quad \boxed{\lim_{t \rightarrow \infty} \text{Var}(\theta_t) = \frac{\alpha \sigma^2}{2 - \alpha}}.$$

so the estimate fluctuates around m .

Hence with constant α , the estimate fluctuates around m and never fully stabilizes.

Case 2: Diminishing step size. To ensure almost sure convergence, we need

$$\boxed{\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty.}$$

This balance guarantees that learning continues indefinitely (first condition) but that the accumulated noise remains bounded (second condition).

2.5 Numerical Illustration

For example, with $\alpha_t = \frac{1}{t+1}$ we have

$$\sum_t \frac{1}{t+1} = \infty, \quad \sum_t \frac{1}{(t+1)^2} < \infty,$$

so both SA conditions are satisfied.

This basic SA recursion will reappear later for learning value functions in reinforcement learning.

3. Online Learning of the Reward Function

3.1 SA Estimators for Rewards

For every state–action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$, the immediate reward $R_t \sim R(\cdot|x, a)$ can be viewed as an independent random variable with unknown mean $r(x, a) = \mathbb{E}[R_t|x, a]$. We maintain a separate stochastic approximation (SA) estimator

$$\hat{r}_t(x, a) \approx r(x, a),$$

and an individual step size $\alpha_t(x, a)$ for each pair.

3.2 Online Update Rule

At time t , the agent observes (X_t, A_t, R_t) . Only the visited pair (X_t, A_t) is updated:

$$\boxed{\hat{r}_{t+1}(X_t, A_t) = (1 - \alpha_t(X_t, A_t)) \hat{r}_t(X_t, A_t) + \alpha_t(X_t, A_t) R_t,}$$

while all other $\hat{r}_t(x, a)$ remain unchanged. A typical design is

$$\alpha_t(x, a) = \frac{1}{n_t(x, a)}, \quad n_t(x, a) = |\{i \leq t : (X_i, A_i) = (x, a)\}|,$$

so that $\hat{r}_t(x, a)$ becomes the running average of all rewards collected at (x, a) . For convergence, every (x, a) must be visited infinitely often and $\alpha_t(x, a)$ must satisfy the SA conditions.

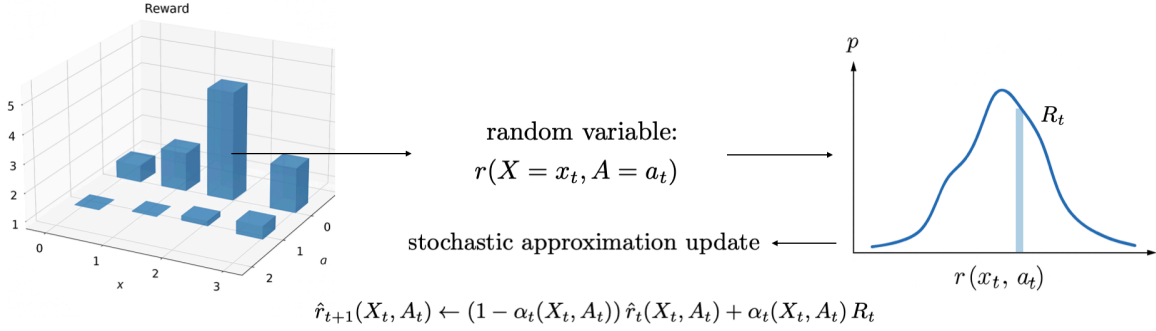


Figure 1: Each pair (x, a) has its own reward estimator $\hat{r}_t(x, a)$.

3.3 Greedy Policy and Its Failure

A greedy policy chooses

$$A_t = \pi_g(X_t; \hat{r}_t) = \arg \max_{a \in \mathcal{A}} \hat{r}_t(X_t, a),$$

but it can get stuck. Consider one state x_1 with two actions a_1, a_2 :

$$r(x_1, a_1) = 1, \quad r(x_1, a_2) = 2,$$

and assume the agent initially estimates $\hat{r}_1(x_1, a_1) = \hat{r}_1(x_1, a_2) = 0$. If it happens to select a_1 first,

$$\hat{r}_2(x_1, a_1) = (1 - \alpha_1) 0 + \alpha_1 \times 1 > 0, \quad \hat{r}_2(x_1, a_2) = 0.$$

Then the greedy policy keeps picking a_1 , never updating a_2 , so $\hat{r}_t(x_1, a_1)$ becomes accurate while $\hat{r}_t(x_1, a_2)$ remains wrong. The agent is trapped in a suboptimal loop.

3.4 Ensuring Exploration-Exploitation Tradeoff

Two common remedies:

1. ε -greedy policy. With probability $1 - \varepsilon$, act greedily; with probability ε , pick a random action:

$$\pi_\varepsilon(x; \hat{r}_t) = \begin{cases} \pi_g(x; \hat{r}_t), & \text{w.p. } 1 - \varepsilon, \\ \text{Uniform}(\mathcal{A}), & \text{w.p. } \varepsilon. \end{cases}$$

This ensures that every (x, a) continues to be explored.

2. Boltzmann (softmax) exploration. Choose actions probabilistically according to

$$\pi(a|x; \hat{r}_t) = \frac{\exp(\hat{r}_t(x, a)/\tau)}{\sum_{a' \in \mathcal{A}} \exp(\hat{r}_t(x, a')/\tau)}.$$

Small τ makes the policy nearly greedy; large τ makes it close to uniform. Both strategies balance *exploration* and *exploitation*.

Remark

To balance exploration across rarely and frequently visited pairs, we may include a class-balanced weight $w_t(x, a) = \frac{1-\beta}{1-\beta^{n_t(x, a)}}$ in the softmax policy:

$$\pi_t(a|x) \propto \exp(\hat{r}_t(x, a) w_t(x, a)/\tau_t).$$

Early on, under-sampled (x, a) receive larger $w_t(x, a)$, encouraging exploration; as $n_t(x, a) \rightarrow \infty$, $w_t(x, a) \rightarrow 1 - \beta$, and the policy gradually behaves like standard Boltzmann exploration, ensuring convergence.

4. Monte Carlo Estimation for Policy Evaluation

4.1 From Reward Learning to Value Function Learning

The reward learning problem is a special case of value function learning where each episode has length 1. In general episodic RL, the agent interacts with the environment over multiple steps and collects a *return*.

Given a policy π , the *return* from time t is

$$G_t^\pi := \sum_{k \geq t} \gamma^{k-t} R_k,$$

and the value function is

$$V^\pi(x) = \mathbb{E}[G_t^\pi | X_t = x].$$

Thus G_t^π plays the same role as Z in mean estimation. Monte Carlo (MC) methods approximate $V^\pi(x)$ by averaging sample returns obtained from rollouts starting at x .

Key analogy.

- Mean estimation: $m = \mathbb{E}[Z]$, with samples Z_1, Z_2, \dots
- Policy evaluation: $V^\pi(x) = \mathbb{E}[G_t^\pi | X_t = x]$, with samples G_t^π obtained from episodes starting at x

MC simply replaces the expectation with the sample average, exactly like in the online mean – estimation setting.

4.2 Monte Carlo State-Value Estimator

Fix a state x . Generate t episodes starting from x and following π until termination:

$$(X_1^{(i)}, A_1^{(i)}, R_1^{(i)}, \dots, X_{T_i}^{(i)}), \quad i = 1, \dots, t.$$

The full return of episode i is

$$G_1^{(i)} = \sum_{k=1}^{T_i} \gamma^{k-1} R_k^{(i)}.$$

The MC estimator of $V^\pi(x)$ is

$$\hat{V}_t^\pi(x) = \frac{1}{t} \sum_{i=1}^t G_1^{(i)}, \quad \hat{V}_t^\pi(x) \xrightarrow{\mathbb{P}} V^\pi(x).$$

4.3 Online (SA) Monte Carlo Update

Let $\hat{V}_t^\pi(x)$ be the estimate after t episodes. Upon observing episode $t+1$ with return $G_1^{(t+1)}$:

$$\hat{V}_{t+1}^\pi(x) = (1 - \alpha_t(x)) \hat{V}_t^\pi(x) + \alpha_t(x) G_1^{(t+1)}$$

A common choice:

$$\alpha_t(x) = \frac{1}{N_t(x)}, \quad N_t(x) = |\{i \leq t : X_1^{(i)} = x\}|,$$

so $\hat{V}_t^\pi(x)$ becomes the running sample average.

Remark

If $\alpha_t(x)$ satisfies the SA conditions

$$\sum_t \alpha_t = \infty, \quad \sum_t \alpha_t^2 < \infty,$$

and x is visited infinitely often, then $\hat{V}_t^\pi(x) \rightarrow V^\pi(x)$ almost surely.

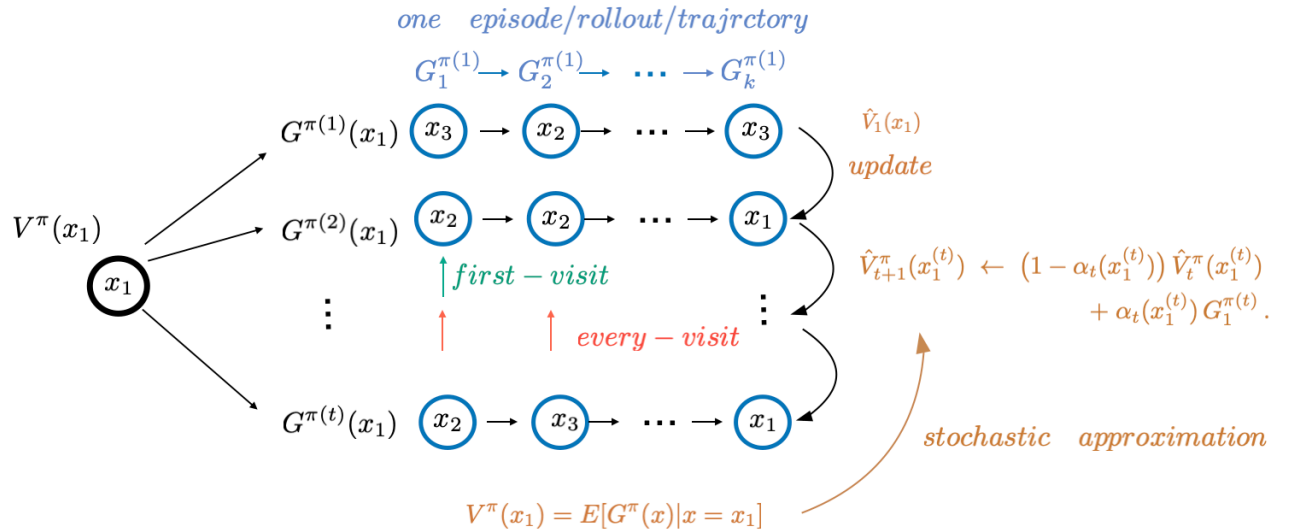


Figure 2: Monte Carlo estimation of $V^\pi(x)$ by averaging returns from episodes starting at x .

4.4 Initial-State-Only Monte Carlo

When interested only in performance from the initial distribution ρ :

$$J(\pi) = \mathbb{E}_{X_1 \sim \rho}[V^\pi(X_1)] = \mathbb{E}[G_1^\pi],$$

we update only the value of the starting state of each episode, using the same MC update rule.

4.5 First-Visit and Every-Visit Monte Carlo

A state x may appear multiple times in one episode:

$$X_{k_1}^{(t)} = x, \quad X_{k_2}^{(t)} = x, \dots$$

First-visit MC. Use only the first k such that $X_k^{(t)} = x$, and update with $G_k^{(t)}$.

Every-visit MC. Use *all* occurrences k_i with $X_{k_i}^{(t)} = x$, updating with each $G_{k_i}^{(t)}$.

Example. There is an example in the figure above about the calculation of x_2 , red and green.

Why first-visit and every-visit samples can be averaged.

MC assumes a *time-homogeneous episodic MDP*: the distribution of future rewards after visiting x depends only on the state x , not on the absolute time step t of the visit. Therefore each $G_{k_i}^{(t)}$ is a sample from the same random variable

$$G^\pi \mid X = x,$$

and both first-visit and every-visit MC are unbiased estimators of $V^\pi(x)$.

First-visit MC has less correlation between samples; every-visit MC uses more samples per episode and often has lower variance.

4.6 A Crucial Clarification: Infinite-Horizon vs Finite-Horizon

A common source of confusion is the belief that “the value of x at early steps should differ from the value of x at later steps.”

Key distinction.

- **Infinite-horizon / time-homogeneous episodic MDPs (END with special X, no relationship with t).**

The value is

$$V^\pi(x),$$

which does *not* depend on the physical time index t . Every time the agent enters x , the future is distributed identically. Thus all returns from visits to x are samples from the same quantity.

- **Finite-horizon MDP (fixed deadline T).** The correct value is

$$V_t^\pi(x) = \mathbb{E} \left[\sum_{k=t}^T \gamma^{k-t} R_k \mid X_t = x \right],$$

which *does* depend on the remaining steps $T - t$. Returns after early and late visits correspond to different value functions ($V_{k_1}^\pi(x)$ vs. $V_{k_2}^\pi(x)$), and cannot be mixed.

Conclusion. Standard MC methods implicitly assume time-homogeneity. For finite-horizon problems, one must either maintain estimates $V_t(x)$ for each time t , or use truncated K -step returns with a fixed window.

4.7 Advantages and Limitations of Monte Carlo Methods

Advantages.

- Simple and model-free: only requires complete returns.
- Does not need the transition kernel P or reward model.
- Unbiased estimation of $V^\pi(x)$.

Limitations.

- Requires termination or a truncation horizon.
- High variance when episodes are long or rewards noisy.
- Ignores the recursive Bellman structure, which TD methods exploit.

4.8 Bridge to Temporal-Difference Learning

MC policy evaluation approximates $V^\pi(x)$ by averaging *complete returns*. However, it waits until the episode ends and does not use Bellman recursion. TD learning combines bootstrapping with sampling, producing online updates that can learn before termination and exploit the Markov structure.