Actuarial Computation and Simulation

Week 03: Monte Carlo & TD Learning (SARSA, Q-learning)

Aprida Siska Lestia

August 31, 2025

Daftar Isi

Benang Merah Materi

- **Week 1: Multi-Armed Bandit**
  Fokus: memilih aksi terbaik (*-greedy*, *UCB*) tanpa mempertimbangkan state.
- **Week 2: Dynamic Programming (DP)**
  Fokus: Markov Decision Process (MDP) dengan *Value Iteration* & *Policy Iteration*.
  Asumsi: model transisi dan reward diketahui lengkap.
- **Week 3: Monte Carlo & TD Learning**
  Fokus: belajar *tanpa model* dengan pengalaman langsung.
  Perbandingan: **SARSA (on-policy)** vs **Q-learning (off-policy)**.

Alur Konsep:

Eksplorasi Aksi (Bandit) $\rightarrow$ MDP dengan Model Lengkap (DP) $\rightarrow$ Belajar dari Pengalaman Nyata (MC/TD)

Motivasi Week 3

- Week 2 (DP) membutuhkan **model lengkap**: probabilitas transisi $p(s'|s,a)$ dan fungsi reward $r(s,a)$.
- Dunia nyata: model *sering tidak diketahui*.
- **Week 3** memperkenalkan **model-free RL**: belajar langsung dari pengalaman interaksi.

Arah belajar: DP (model-based) $\Rightarrow$ MC/TD (model-free)

Intuisi Monte Carlo

- Belajar dari **episode penuh**: jalankan episode sampai selesai, hitung total return.
- **Tanpa bootstrap** (murni dari sampel).
- Tidak perlu tahu model transisi (probabilitas pindah antar state), cukup kumpulkan episode pengalaman lalu hitung return.
- Cocok saat kita bisa/harus menunggu episode selesai.

Alur Konsep: Return $\rightarrow$ Value $\rightarrow$ Estimasi

1. Return pada langkah t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

- Total reward diskonto mulai dari langkah t sampai akhir episode.
- Menjadi bahan mentah untuk menilai baik/buruknya suatu state atau aksi.
- R : reward, γ : discount factor, dan T : langkah terakhir episode

Alur Konsep: Return → Value → Estimasi

1. Return pada langkah t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

- Total reward diskonto mulai dari langkah t sampai akhir episode.
- Menjadi bahan mentah untuk menilai baik/buruknya suatu state atau aksi.
- R : reward, γ : discount factor, dan T : langkah terakhir episode

2. State Value

$$V(s) \approx \mathbb{E}[G_t \mid S_t = s]$$

dengan G_i adalah return episode ke-i saat s pertama kali dikunjungi.

- Rata-rata return jika agent mulai dari state s.
- Menjawab pertanyaan: seberapa bagus state s?

Alur Konsep: Return $\rightarrow$ Value $\rightarrow$ Estimasi

3. Action Value

$$Q(s, a) \approx \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

- Rata-rata return jika dari state s agent memilih aksi a.
- Lebih detail daripada $V(s)$, digunakan untuk memilih aksi terbaik.

Alur Konsep: Return → Value → Estimasi

3. Action Value

$$Q(s, a) \approx \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

- Rata-rata return jika dari state s agent memilih aksi a.
- Lebih detail daripada $V(s)$, digunakan untuk memilih aksi terbaik.

4. Estimasi Monte Carlo (First-visit MC)

$$V(s) \leftarrow \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i$$

- Mengestimasi $V(s)$ dengan rata-rata return nyata dari banyak episode.
- G_i = return dari episode ke-i saat s pertama kali dikunjungi.

Algoritma Monte Carlo (First-Visit MC)

1. Inisialisasi $V(s)$ atau $Q(s, a)$ sembarang.
2. Jalankan banyak episode dengan policy tertentu.
3. Untuk setiap state atau pasangan (s, a) yang muncul:
   - Hitung return G_t.
   - Simpan hasil return tersebut.
   - Update estimasi $V(s)$ atau $Q(s, a)$ dengan rata-rata return.

First-visit vs Every-visit

Tujuan: Estimasi $V(s) = \mathbb{E}[G_t \mid S_t = s]$ pada tugas episodik.

- **First-visit MC**: update $V(s)$ hanya pada kunjungan pertama s di sebuah episode.
- **Every-visit MC**: update $V(s)$ pada setiap kemunculan s di episode.

Kelebihan MC: tidak butuh model, konsisten saat banyak episode.
Keterbatasan: harus menunggu episode selesai, variansi tinggi di horizon panjang.

Monte Carlo Prediction: First-Visit vs Every-Visit

Contoh ilustratif:

Episode 1: s muncul di t=2 $\Rightarrow$ $G = 5$, dan di t=5 $\Rightarrow$ $G = 3$

Episode 2: s muncul sekali di t=1 $\Rightarrow$ $G = 7$

FV: Sampel $\{5, 7\}$ $\Rightarrow$ $\hat{V}_{\text{FV}}(s) = \dfrac{5+7}{2} = 6.0$

EV: Sampel $\{5, 3, 7\}$ $\Rightarrow$ $\hat{V}_{\text{EV}}(s) = \dfrac{5+3+7}{3} = 5.0$

Interpretasi:

- Perbedaan pada contoh terjadi karena EV memasukkan seluruh kunjungan (termasuk yang kurang menguntungkan).
- untuk episode yang besar, FV dan EV akan *konvergen* ke nilai yang sama; EV sering lebih stabil (varians lebih kecil) karena lebih banyak sampel per episode.

Jalankan : ACS_2025/MC_Blackjack.ipynb

Temporal Difference (TD) Learning

1. Ide Dasar

- Tidak perlu menunggu episode selesai. sebelum episode selesai, dilakukan update setiap transisi/langkah.
- Update nilai setiap langkah dengan **bootstrap**, gunakan estimasi $V(s_{t+1})$ untuk memperbaiki $V(s_t)$.

$$V(s_t) \leftarrow V(s_t) + \alpha\Big[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big]$$

Learning rate $\alpha \in (0, 1]$, *discount* $\gamma \in [0, 1)$.

Temporal Difference (TD) Learning

1. Ide Dasar

- Tidak perlu menunggu episode selesai. sebelum episode selesai, dilakukan update setiap transisi/langkah.
- Update nilai setiap langkah dengan **bootstrap**, gunakan estimasi $V(s_{t+1})$ untuk memperbaiki $V(s_t)$.

$$V(s_t) \leftarrow V(s_t) + \alpha\Big[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big]$$

Learning rate $\alpha \in (0, 1]$, *discount* $\gamma \in [0, 1)$.

2. TD Error

$$\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

- δ_t adalah **kesalahan prediksi** pada langkah t.
- Update = nilai lama + learning rate $\times$ error.

Contoh Temporal Difference (TD) Learning

Misalkan agent berada di state s, kemudian berpindah ke s' dengan reward $r = 1$.

- Estimasi awal: $V(s) = 2.0,\ V(s') = 3.0$
- Parameter: $\alpha = 0.5,\ \gamma = 0.9$

$$V(s) \leftarrow V(s) + \alpha\Big[r + \gamma V(s') - V(s)\Big]$$

$$V(s) \leftarrow 2.0 + 0.5\big[1 + 0.9 \cdot 3.0 - 2.0\big]$$

$$V(s) = 2.85$$

Interpretasi

Nilai $V(s)$ naik karena reward langsung + estimasi masa depan lebih besar daripada prediksi awal.

Contoh Temporal Difference (TD) Learning

Hitung TD Error

$$\delta_t = r + \gamma V(s') - V(s)$$

$$\delta_t = 1 + 0.9 \cdot 3.0 - 2.0 = 1.7$$

Update Nilai

$$V(s) \leftarrow V(s) + \alpha \cdot \delta_t = 2.0 + 0.5 \cdot 1.7 = 2.85$$

Interpretasi

Error $\delta_t = 1.7$ menunjukkan prediksi awal terlalu rendah, sehingga nilai $V(s)$ diperbesar menjadi 2.85.

MC vs TD

- **Target:** MC pakai return penuh; TD bootstrap.
- **Waktu update:** MC setelah episode; TD tiap langkah (online).
- **Biasvarian:** MC rendah bias, tinggi varian; TD sedikit bias, varian lebih rendah.
- **Efisiensi & cakupan:** TD lebih sample-efficient; cocok untuk continuing tasks; MC perlu episode selesai.

Action-Value Learning

- **Tujuan:** belajar fungsi $Q(s, a)$ untuk mengendalikan policy (memilih aksi terbaik pada setiap state).
- **Definisi:**

$$Q(s, a) \approx \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

yaitu rata-rata return jika dari state s agent mengambil aksi a.

- **Monte Carlo:**

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{N(s,a)} G_i$$

$\rightarrow$ update dengan rata-rata return dari banyak episode penuh.

Action-Value Learning

- **Temporal Difference (TD):**
  - **SARSA (on-policy):**

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

  - **Q-learning (off-policy):**

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

- Perbedaan: SARSA lebih konservatif (ikut aksi yang diambil), Q-learning lebih agresif (asumsi aksi terbaik).

SARSA (On-policy)

Transisi: (s, a, r, s', a') mengikuti policy saat ini (mis. ϵ-greedy).
Update:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big).$$

- s : state saat ini
- a : aksi yang dipilih pada state s
- r : reward yang diperoleh setelah aksi a
- s' : state berikutnya
- a' : aksi berikutnya (dipilih oleh policy ϵ-greedy)
- α : learning rate
- γ : discount factor

Ciri:

- *On-policy*: belajar dari aksi yang benar-benar diambil.
- Lebih **konservatif** saat eksplorasi; sering lebih **stabil** di lingkungan berisiko.

Q-learning (Off-policy)

Transisi: (s, a, r, s'); aksi berikutnya untuk target adalah aksi **greedy** terbaik.

Update:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big).$$

- s : state saat ini
- a : aksi yang dipilih pada state s (dengan policy ϵ-greedy)
- r : reward yang diperoleh setelah aksi a
- s' : state berikutnya
- a' : semua aksi yang mungkin diambil di s', pilih max
- α : learning rate
- γ : discount factor

Ciri:

- *Off-policy*: target memakai aksi optimal walau perilaku eksploratif.
- **Agresif** menuju optimalitas; konvergen ke Q^* (syarat standar).

Perbandingan Singkat

- **Target**: SARSA pakai $Q(s', a')$ (aksi diambil), Q-learning pakai $\max_{a'} Q(s', a')$.
- **Eksplorasi**: SARSA merasakan risiko aksi eksploratif; Q-learning mengasumsikan aksi terbaik.
- **Praktik**: SARSA sering lebih aman di lingkungan *stochastic/slippery*; Q-learning sering lebih cepat ke optimum.

Lab :

- Jalankan TD_FrozenLake_SARSA_Q.ipynb di: *github_ACS_2025*
- Environment **FrozenLake** : grid dengan *slippery dynamics*.
- Tujuan: capai goal tanpa jatuh ke hole.

Assignment

1. Implementasikan SARSA & Q-learning pada lingkungan **FrozenLake** (Gunakan notebook yang tersedia; jalankan pelatihan dengan parameter default.).
2. Replikasi untuk konsistensi (Ulangi pelatihan beberapa kali dengan mengganti seed (set np.random.seed(...) & random.seed(...) di sel Setup) atau re-run dari awal. Catat rata-rata success rate akhir.).
3. Plot & baca hasil (Gunakan grafik success rate evaluasi berkala (default: tiap 200 episode). Boleh menurunkan interval evaluasi bila perlu).
4. Lakukan analisis:
   - Bandingkan kecepatan konvergensi & stabilitas kurva SARSA vs Q-learning.
   - Pada kondisi apa SARSA lebih unggul? Kapan Q-learning lebih baik?
   - Hubungkan temuan ini dengan sifat lingkungan stokastik (is_slippery=True).

Diskusi Hasil Eksperimen TD Learning (FrozenLake)

Pertanyaan untuk membimbing interpretasi:

- Bagaimana perbedaan pola success rate SARSA vs Q-learning?
- Algoritma mana yang lebih cepat stabil? Mengapa (on-policy vs off-policy, risiko slip)?
- Peran sifat lingkungan diskrit + licin terhadap performa?
- Trade-off agresif (Q-learning) vs konservatif (SARSA)?
- **(Pengaya untuk MC Learning :)** Bandingkan kualitas kebijakan MC vs hasil TD di FrozenLake secara konsep (bukan angka mentah)

Catatan : FrozenLake pakai success rate (probabilitas sampai goal), bukan reward/episode.

(Opsional) Ekstensi untuk TD Learning di Lingkungan CartPole

Jalankan TD_CartPole_SARSA_Q.ipynb di: *github_ACS_2025*

- Jalankan SARSA & Q-learning, ulangi 3-5 seed.
- Plot reward rata-rata per 100 episode (bukan success rate).
- Jawab singkat:
  - Bentuk kurva reward SARSA vs Q-learning?
  - Mengapa pola CartPole bisa berbeda dari FrozenLake (kontinu vs diskrit, tidak licin)?

Interpretasi Hasil SARSA vs Q-learning

FrozenLake (diskrit, licin):

- Q-learning lebih cepat naik, stabil di success rate $\sim$0.7–0.8.
- SARSA lebih lambat, banyak fluktuasi, tapi mendekati performa Q-learning.
- $\Rightarrow$ Q-learning cocok untuk lingkungan diskrit dengan risiko sederhana.

CartPole (kontinu, didiskretisasi):

- SARSA rata-rata reward $\sim$275 (lebih tinggi).
- Q-learning rata-rata reward $\sim$206 (lebih rendah, plateau lebih cepat).
- SARSA on-policy $\Rightarrow$ lebih stabil dalam transisi halus.

Secara umum:

- Lingkungan berisiko/noisy: SARSA cenderung lebih aman.
- Lingkungan diskrit sederhana: Q-learning lebih cepat optimal.
- Lingkungan kontinu/diskretisasi: SARSA sering lebih konsisten.

Kesimpulan

- Week 3: **model-free RL** belajar dari pengalaman tanpa model.
- **MC** (episode penuh) vs **TD** (update per langkah).
- **SARSA** (on-policy) vs **Q-learning** (off-policy): trade-off aman vs cepat.
- **Perbandingan lingkungan :**
  - *FrozenLake*: Metriknya success rate (= rata-rata return, karena reward hanya $+1$ saat goal). Tugas capai goal .
  - *CartPole*: Metriknya rataan reward/episode (lama bertahan). Target klasik 195 (rata-rata 100 episode). Tugas ketahanan waktu