Advanced Application of Artificial Intelligence and Digital Transformation

K.N. Toosi University of Technology

فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

جستجوی درختی(Brute-force search)

IBM Deep Blue vs. Kasparov 1997

environment: chess board
task: play and win a chess game
actions: move chess pieces
implementation: Min-Max algorithm

اساس برنامه دیپ بلو یادگیری ماشین به مفهوم امروزی نبود

ارزش موقعیت‌های مختلف صفحه (مانند ارزش مهره‌ها، کنترل خانه‌های مرکزی و ایمنی شاه) توسط مهندسان و اساتید شطرنج از قبل تعریف و کدنویسی شده بود(Brute-force search) بررسی تمام حالات بسیار زیاد بوده و از هرس آلفا-بتا برای کاهش حالات جستجو استفاده شد.

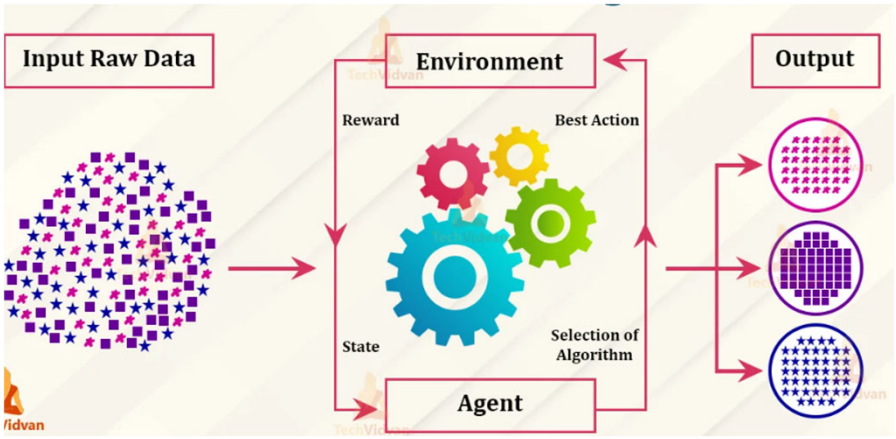
این روش برای بازیهای پیچیده تر مثل بازی GO جوابگو نیست

Adv. App. of AI and DT 3

یادگیری تقویتی(Reinforced Learning)

Adv. App. of AI and DT 4

یادگیری تقویتی(Reinforced Learning)



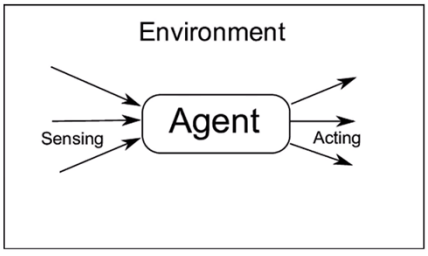
Adv. App. of AI and DT 5

یادگیری تقویتی(Reinforced Learning)

AlphaGO playing GO (2016)
(Deep Reinforcement Learning)

19 خط افقی و 19 خط عمودی
181 مهره مشکی و 180 مهره سفید
هدف حذف مهرهای مقابل از طریق محاصره
پیچیده تر از بازی شطرنج

Autonomous Agent عامل خود مختار



environment: GO board
task: Play Go game and win
actions: move pieces
implementation: reinforcement learning

Adv. App. of AI and DT 6

یادگیری تقویتی(Reinforced Learning)

برنامه آلفازیرو برای بازی شطرنج در سال ۲۰۱۷ نوشته شد

برنامه	Deep Blue (1997)	AlphaZero (2017)
روش اصلی	Minimax + Alpha-Beta	Neural Network + MCTS
تعداد وضعیت‌های بررسی شده	حدود ۲۰۰ میلیون در ثانیه	حدود ۶۰ هزار در ثانیه
دانش شطرنج	قوانین و ارزیابی دست‌ساز	یادگرفته شده
عمق جستجو	۱۰ تا ۲۰ حرکت (گاهی بیشتر)	معمولاً کمتر
سخت‌افزار	پردازنده‌های اختصاصی شطرنج	TPUهای گوگل
یادگیری	ندارد	یادگیری تقویتی

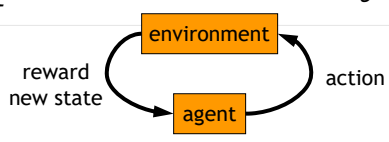
Deep Blue: تقریباً همه‌چیز را نگاه کن
AlphaZero : اول فکر کن، بعد نگاه کن

نسبت محاسبات
200,000,000/60,000≈3333

Reinforcement Learning

انواع یادگیری تقویتی

روش	توضیح	مثال کاربردی
مبتنی بر مدل **(Model-Based)**	عامل یک مدل از محیط می‌سازد و پیش‌بینی می‌کند (ربات نقشه محیط را یاد میگیرد سپس مسیر بهینه را برنامه ریزی می کند).	بهینه‌سازی انفجارهای معدن بهینه‌سازی طراحی شیب‌ها
بدون مدل **(Model-Free)**	عامل مستقیماً و بدون مدل‌سازی با محیط تعامل دارد(ربات فقط از تجربه حرکت و گرفتن جایزه یا تنبیه، یاد می‌گیرد کدام مسیر بهتر است).	برنامه ریزی حمل و نقل مثل تردد تراک‌ها در معادن
Q-Learning	از یک جدول (Q-Table) برای ذخیره ارزش هر عمل در هر حالت استفاده می‌کند.	مدیریت انرژی، ترافیک هوشمند بهینه‌سازی تزریق آب در مخازن نفت
Deep RL	ترکیب RL با شبکه‌های عصبی عمیق (مثل DQN:Deep Q-Network)	بهینه‌سازی شکست هیدرولیکی در مخازن شیل خودروهای خودران (Tesla)

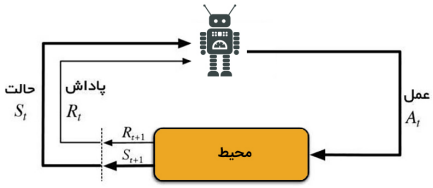


Reinforcement Learning

more general than supervised/unsupervised learning
learn from interaction w/ environment to achieve a goal

مفاهیم

- ✓ **عامل (Agent)** : موجودی که تصمیم‌گیری می‌کند (مثلاً یک ربات یا برنامه کامپیوتری).
- ✓ **محیط (Environment)** : دنیایی که عامل در آن عمل می‌کند (مثلاً یک بازی ویدیویی یا سیستم رانندگی خودکار).



- ✓ **حالت (State)** : وضعیت فعلی محیط (مثلاً موقعیت ربات)
- ✓ **عمل یا اقدام (Action)** : تصمیمی که عامل می‌گیرد (مثلاً حرکت به چپ یا راست).
- ✓ **پاداش (Reward)** : بازخورد عددی که محیط پس از هر عمل به عامل می‌دهد (مثلاً +۱ برای برد و −۱ برای باخت).
- ✓ **سیاست (Policy)** : استراتژی عامل برای انتخاب عمل‌ها در هر حالت (مثلاً اگر در حالت X هستی، عمل Y را انجام بده).

Reinforcement Learning

مفاهیم

- ✓ **اقدام (Action)** : شامل تمامی واکنش‌های محتملی است که عامل تصمیم‌گیرنده ممکن است در مواجهه با وضعیت ایجاد شده، از خود نشان دهد.
- ✓ اقدام‌ها با توجه به متناهی بودن و یا نامتناهی بودنشان به دو نوع اپیزودیک و یا مستر (غیر اپیزودیک) تقسیم‌بندی می‌شوند.
- ✓ اقدام اپیزودیک: دارای یک نقطه آغاز و یا پایان است. یک بازی را در نظر بگیرید که در آن بازی‌کننده کامپیوتری ممکن است بمیرد و یا به پایان مرحله برسد. در چنین حالتی می‌توان یک لیستی از وضعیت‌ها، اقدامات و یا پاداش‌ها داشت.
- ✓ اقدام مستمر یا غیر اپیوزدیک: نقطه پایانی برای اقدامات وجود ندارد. مثل تجارت سهام تا زمانی که عامل توسط نیروی خارجی (مثل کاربر انسانی) متوقف نشود، همواره به یادگیری و انجام اقدام مناسب به تعامل با محیط می‌پردازد.

ماتریس Q یا (Q-Table) در یادگیری تقویتی به ما می‌گوید که در یک **حالت (state)** مشخص، انتخاب هر **عمل (action)** چه پاداشی در بلندمدت (با در نظر گرفتن پاداش‌های آینده) دارد.

یادگیری تقویتی

در یادگیری تقویتی

عامل با انجام اقدامات مختلف و دریافت پاداش یا جریمه از محیط، سعی می‌کند یک سیاست بهینه پیدا کند که حداکثر پاداش تجمعی را در طول زمان به دست آورد.

مثال در بازی شطرنج:

پاداش: +۱ برای برد، −۱ برای باخت، ۰ برای تساوی.

حالت: موقعیت مهره‌ها روی صفحه.

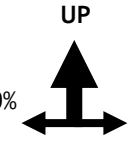
عمل: حرکت یک مهره.

عامل با میلیون‌ها بار بازی کردن، یاد می‌گیرد که چه حرکاتی شانس برد را افزایش می‌دهند.

حرکت یک ربات در اتاق

			+1
			-1
START			

Actions: UP, DOWN, LEFT, RIGHT



move UP 80%

move LEFT 10%

move RIGHT10%

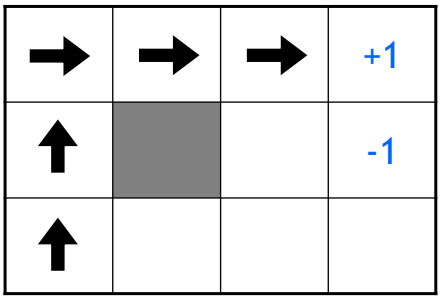
- reward +1 at [4,3], -1 at [4,2]
- reward -0.1 for each step

- what's the strategy to achieve max reward?
- what if the actions were deterministic?

Adv. App. of AI and DT 12

حرکت یک ربات در اتاق

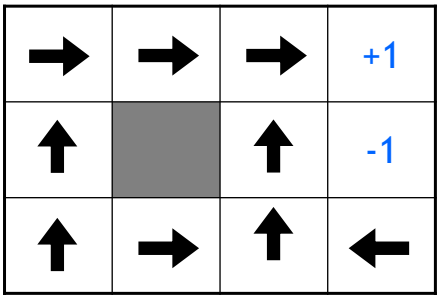
Is this a solution?



- only if actions deterministic
  - not in this case (actions are stochastic)
- solution/policy
  - mapping from each state to an action

Adv. App. of AI and DT 13

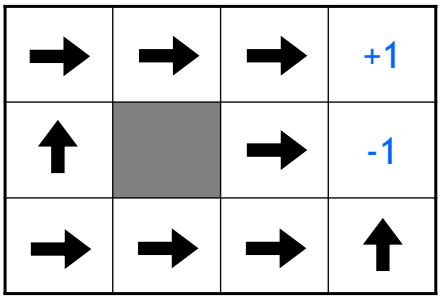
Optimal policy



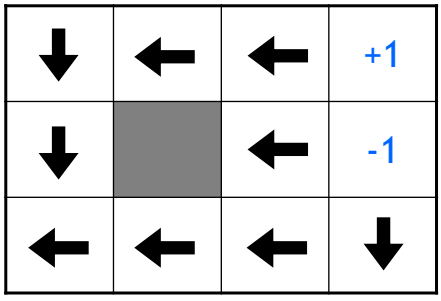
Reward for each step: -0.1

Adv. App. of AI and DT 14

تاثیر پاداش



Reward for each step: -2



Reward for each step: +0.1

Adv. App. of AI and DT 15

فهرست مطالب

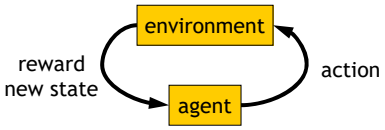
- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

Adv. App. of AI and DT 16

Markov Decision Process (MDP)

فرآیندهای تصمیم‌گیری مارکوف : مدل تصمیم‌گیری متوالی در محیط‌های تصادفی است که پایه‌ی نظری بسیاری از الگوریتم‌های یادگیری تقویتی را تشکیل می‌دهد.

- **S (States)**: **مجموعه حالت‌ها** که درمحیط ممکن است (مثلاً موقعیت ربات در یک نقشه)
- **A (Actions)**: **مجموعه اقدامات** که قابل انجام توسط عامل است (مثلاً حرکت به چپ/راست)
- **P (Transition Probability)**: **تابع انتقال** که احتمال گذار بین حالتها است
  $P(s'|s,a) =$ احتمال رفتن به حالت s' اگر در حالت s عمل a انجام شود
- **R (Reward Function)**: **تابع پاداش**
  $R(s,a,s') =$ پاداش دریافت‌شده پس از گذار از s به s' با عمل a



خاصیت مارکوف (فرض مارکوف): وضعیت آینده فقط به حالت فعلی و عمل فعلی بستگی دارد، نه به تاریخچه گذشته

$$P(s_{t+1} \mid s_t, a_t) = P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots)$$

Markov Decision Process (MDP)

مؤلفه‌های اضافی: ضریب تخفیف، استراتژی

- **γ (Discount Factor)**: ضریب تخفیف (بین ۰ تا ۱) برای ارزش‌دهی به پاداش‌های آینده
- **π (Policy)**: استراتژی عامل برای انتخاب عمل در هر حالت
  $\pi(a|s) =$ احتمال انتخاب عمل a در حالت s

هدف MDP: یافتن استراتژی است

یافتن استراتژی یا سیاست بهینه (π^*)که مقدار بازده تجمعی مورد انتظار را حداکثر کند

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}, s_{t+k+1}) \mid s_t = s\right]$$

تابع ارزش (تخمین تمام پاداشهای آتی) در یک حالت s

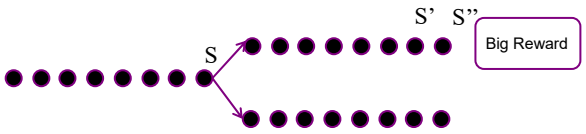
حل MDP: روش‌های مختلف از جمله **برنامه‌ریزی پویا**

الگوریتم ارزش‌تکرار (Value Iteration):

$$V_{k+1}(s) = \max_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V_k(s')]$$

تاخیر در پاداش

تاخیر در پاداش یعنی اقدام فعلی عامل ممکن است هم‌اکنون پاداش نداشته باشد یا حتی منفی باشد، ولی در آینده منجر به پاداش مطلوب شود (مثل قربانی کردن مهره در شطرنج).



عامل باید یاد بگیرد که:
فقط به پاداش فوری نگاه نکند، بلکه ارزش تجمعی آینده را نیز در نظر بگیرد.

راه حل:

- ✓ استفاده از تخفیف پاداش‌ها (پارامتر γ)
- ✓ استفاده از روش‌های مبتنی بر ارزش (Value-Based) مثل Q-Learning
- ✓ روش‌های سیاست‌محور (Policy Gradient) با حافظه بلندمدت

سیاست قطعی و آماری

$\pi(s)$ سیاست قطعی: اگر گرسنه‌ای (s)، همیشه غذا بخور(a)
$\pi(a|s)$ سیاست آماری: اگر گرسنه‌ای (s)، با احتمال ۰.۷ غذا بخور، با احتمال ۰.۳ بخواب

سیاست قطعی: برای محیط ساده و کاملاً قابل پیش‌بینی نظیر الگوریتم DDPG (یادگیری تقویتی برای محیط‌های پیوسته که ترکیبی از یادگیری عمیق و استراتژی قطعی است)

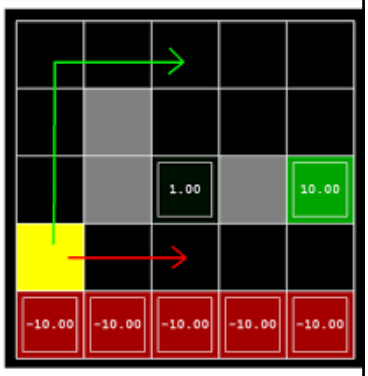
سیاست آماری: برای محیط تصادفی یا پیچیده (مثلاً با پاداش تأخیری یا نویز)

- ✓ حتی در الگوریتم‌های قطعی مثل DDPG در مرحله آموزش معمولاً نویز تصادفی اضافه می‌شود تا اکتشاف بیشتر انجام شود.

MDP مثال

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}, s_{t+k+1}) \mid s_t = s\right]$$

four preference scenarios:
(a) Prefer the close exit (+1), risking the cliff (-10)
(b) Prefer the close exit (+1), but avoiding the cliff (-10)
(c) Prefer the distant exit (+10), risking the cliff (-10)
(d) Prefer the distant exit (+10), avoiding the cliff (-10)

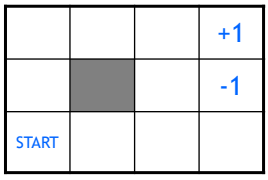


four parameter settings:

(1) γ=0.1,	noise = 0.5	(a)
(2) γ=0.1,	noise = 0	(b)
(3) γ=0.99,	noise = 0.5	(c)
(4) γ= 0.99,	noise = 0	(d)

Markov Decision Process (MDP)

- set of states S, set of actions A, initial state S_0
- transition model P(s,a,s')
  - P([1,1], up, [1,2]) = 0.8
- reward function r(s)
  - r([4,3]) = +1
- goal: maximize cumulative reward in the long run



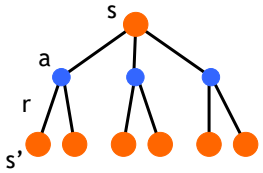
- policy: mapping from S to A
  - π(s) or π(s,a) (deterministic vs. stochastic)
- reinforcement learning
  - transitions and rewards usually not available
  - how to change the policy based on experience
  - how to explore the environment

محاسبه پاداش

- additive rewards — پاداش تجمعی
  - $V(s_0, s_1, \ldots) = r(s_0) + r(s_1) + r(s_2) + \ldots$
  - infinite value for continuing tasks
- discounted rewards — پاداش تجمعی تخفیف یافته
  - $V(s_0, s_1, \ldots) = r(s_0) + \gamma * r(s_1) + \gamma^2 * r(s_2) + \ldots$
  - value bounded if rewards bounded

Value functions تابع ارزش

- state value function: $V^{\pi}(s)$ — تابع ارزش حالت
  - expected return when starting in *s* and following π
- state-action value function: $Q^{\pi}(s,a)$ — تابع ارزش حالت – اقدام
  - expected return when starting in *s*, performing *a*, and following π
- useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using $Q^{\pi}(s,a)$



- Bellman equation — فرمول بلمن

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s') \right] = \sum_a \pi(s,a) Q^{\pi}(s,a)$$

تابع ارزش حالت- اقدام رابر اساس پاداش فعلی وارزش آینده محاسبه می کند

تابع ارزش بهینه

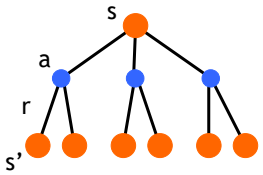
- there's a set of *optimal* policies
  - V^{π} defines partial ordering on policies
  - they share the same optimal value function

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- Bellman optimality equation

$$V^*(s) = \max_{a} \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^*(s') \right]$$

  - solve for V*(s)
  - easy to extract the optimal policy



- having Q*(s,a) makes it even simpler

$$\pi^*(s) = \arg\max_{a} Q^*(s,a)$**سیاست بهینه عملی را انتخاب می‌کند که بیشترین مقدار تابع ارزش حالت-اقدام بهینه را داشته باشد**

این فرمول پایه بسیاری از الگوریتم‌های یادگیری تقویتی مانند Q-learning و DQN است

مثال

مثال: تغییرات آب‌وهوا به روش فرایند تصمیم مارکوف بین سه حالت **"آفتابی"**، **"ابری"** و **"بارانی"**

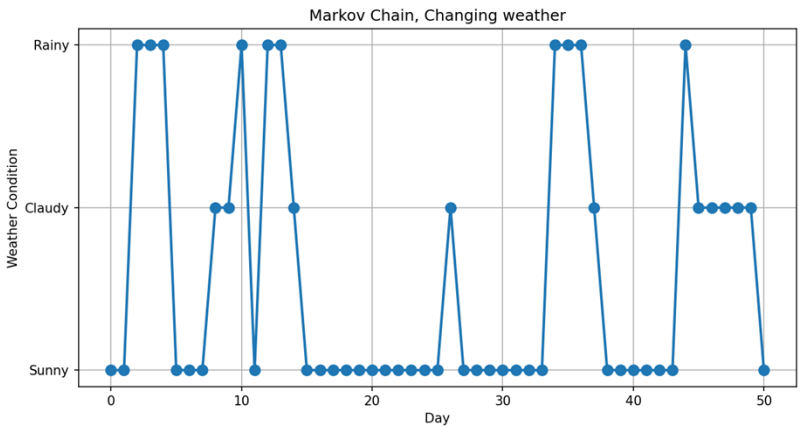
```
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict

# 1. تعریف حالت‌ها و ماتریس انتقال
states = ["Sunny", "Claudy", "Rainy"]
transition_matrix = np.array([
    [0.7, 0.2, 0.1],  # از آفتابی به [آفتابی, ابری, بارانی]
... # از ابری به [۰.۳, ۰.۴, ۰.۳]
... # از بارانی به [۰.۵, ۰.۳, ۰.۲]
])

# ۲. تابع تولید دنباله مارکوف
def generate_markov_sequence(initial_state, steps=100):
    current_state = initial_state
    sequence = [current_state]
    state_indices = {s: i for i, s in enumerate(states)}

    for _ in range(steps):
        prob = transition_matrix[state_indices[current_state]]
        current_state = np.random.choice(states, p=prob)
        sequence.append(current_state)

    return sequence
```

مثال

```
# ۳. تولید دنباله و ذخیره نتایج
np.random.seed(42)  # برای تکرارپذیری
sequence = generate_markov_sequence("Sunny", 50)

# 4. تبدیل حالت‌ها به اعداد برای رسم نمودار
state_to_num = {"Sunny": 0, "Claudy": 1, "Rainy": 2}
numeric_sequence = [state_to_num[s] for s in sequence]

# 5. رسم نمودار
plt.figure(figsize=(10, 5))
plt.plot(numeric_sequence, 'o-', markersize=8, linewidth=2)
plt.yticks([0, 1, 2], states)
plt.xlabel("Day")
plt.ylabel("Weather Condition")
plt.title("Markov Chain, Changing weather")
plt.grid(True)
plt.show()
```

یک نمودار خطی تولید می‌شود که محور **x** نشان‌دهنده روزها و
محور **y** وضعیت آب‌وهوا را نمایش می‌دهد

نتیجه مثال



فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

برنامه ریزی پویا

DP: Dynamic Programming

برنامه ریزی پویا (DP) الگوریتم‌هایی که از معادلات بازگشتی (مانند معادلات بلمن) استفاده می‌کنند تا تابع ارزش و سیاست بهینه را پیدا کند.

- ✓ تابع ارزش بهینه $V^*(s)$ یا تابع ارزش حالت-عمل بهینه $Q^*(s,a)$
- ✓ سیاست بهینه $\pi^*(s)$

پیش‌نیازهای استفاده از برنامه‌ریزی پویا(DP)

1. **مشخص بودن کامل مدل**: یعنی دانستن تابع انتقال $P(s'|s,a)$ و تابع پاداش $R(s,a)$
2. **فضای حالت و اقدام محدود**: الگوریتم‌های DP معمولاً به حافظه و زمان زیادی نیاز دارند.

Policy evaluation/improvement

- policy evaluation: π -> V^π
  - با نوشتن معادلات بلمن برای یک محیط با n حالت n معادله بدست می‌آید که با حل آنها توابع ارزش برای هر حالت تعیین می شود روش تکرار برای حل معادلات استفاده می شود.

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{k'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$$

  - start with an arbitrary value function V_0, iterate until V_k converges

- policy improvement: V^π -> π'
  - برای یک محیط با n خالت و m اقدام n*m معادله بدست می‌آید که با حل آنها توابع ارزش برای هرحالت- اقدام تعیین می شود .

$$\pi'(s) = \arg\max_a Q^\pi(s,a)$$
$$= \arg\max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^\pi(s') \right]$$

  - π' either strictly better than π, or π' is optimal (if $\pi = \pi'$)

فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

روش مونت کارلو (MC)

MC Methods: Monte Carlo methods

روش مونت کارلو : بر پایه‌ی اجرای اپیزودهای کامل و سپس استفاده از **میانگین پاداش‌های مشاهده شده** برای تخمین ارزش حالت‌ها یا جفت‌های حالت-اقدام است.

مراحل

1.اجرای سیاست مشخص π در محیط به‌صورت چندین اپیزود.

2. برای هر حالت s، پاداش تجمعی آینده G_t را محاسبه می شود:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$$

3. میانگین پاداش‌های تجمعی حالت s را در هر بار دیدن آن، محاسبه کنید:

$$V(s) = \text{average of all } G_t \text{ when } s \text{ is visited}$$

4.در نهایت، V(s) به صورت میانگین تجربی تخمین زده می‌شود.

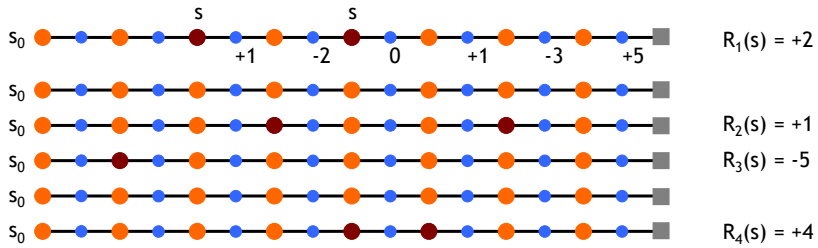
مزایا:
- ✓ ساده و شهودی
- ✓ بدون نیاز به مدل محیط
- ✓ قابل استفاده برای تخمین مستقیم سیاست π

معایب:
- ➢ فقط در محیط‌های اپیزودیک قابل استفاده است
- ➢ همگرایی کندتر نسبت به روش‌های مبتنی بر بوت‌استرپ
- ➢ ناپایدار بودن در حالت تغییرات زیاد

روش مونت کارلو- ارزیابی سیاست

- want to estimate $V^{\pi}(s)$
  = expected return starting from s and following π
  – estimate as average of observed returns in state s

- first-visit MC
  – average returns following the first visit to state s



$V^{\pi}(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$

فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

یادگیری اختلاف زمانی

TD Learning: Temporal Difference Learning

در این روش به جای اینکه صبر کنیم تا اپیزود تمام شود، در همان لحظه‌ای گرفتن پاداش، تخمین به‌روزرسانی می‌شود.

- combines ideas from MC and DP
  - like MC: learn directly from experience (don't need a model)
  - like DP: learn from values of successors
  - works for continuous tasks, usually faster than MC
- constant-alpha MC:
  - have to wait until the end of episode to update

$$V(s_t) \leftarrow V(s_t) + \alpha\,[R_t - V(s_t)]$$

target

- simplest TD
  - update after every step, based on the successor

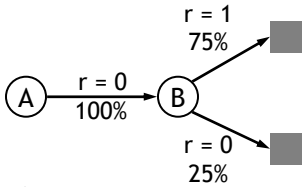
$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

MC vs. TD

- observed the following 8 episodes:

A - 0, B - 0	B - 1	B - 1	B - 1
B - 1	B - 1	B - 1	B - 0

- MC and TD agree on V(B) = 3/4
- MC: V(A) = 0
  - converges to values that minimize the error on training data
- TD: V(A) = 3/4
  - converges to ML estimate of the Markov process



مثال:پیش بینی قیمت طلا در شش ماه آینده

14 RL gold price 2026 Markov Chain.py

«وضعیت آینده بازار فقط به وضعیت فعلی وابسته است، نه کل گذشته.»
یعنی اگر امروز بازار در حالت «رشد شدید» باشد، احتمال فردا از روی همین وضعیت تعیین می‌شود.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import yfinance as yf
from datetime import datetime

# ========================================
# 1. Download REAL gold price data
# ========================================

# Gold Futures symbol in Yahoo Finance
symbol = "GC=F"

# Download data
gold_data = yf.download(
    symbol,
    start="2024-01-01",
    end=datetime.today().strftime('%Y-%m-
%d'),
    auto_adjust=True
)
```

دریافت تاریخچه قیمت از یاهو فاینانس

مثال:پیش بینی قیمت طلا در شش ماه آینده

14 RL gold price 2026 Markov Chain.py

```
# ========================================
# 2. Extract closing prices safely
# ========================================

try:
    # Standard format
    close_data = gold_data['Close']


    # Convert to 1D numpy array
    gold_prices = close_data.dropna().squeeze().to_numpy()

except Exception:
    # MultiIndex fallback
    close_data = gold_data[('Close', symbol)]
    gold_prices = close_data.dropna().to_numpy()

# Dates
dates = gold_data.index

# Check data
print("Gold Prices Shape:", gold_prices.shape)

if len(gold_prices) < 10:
    raise ValueError("Not enough data downloaded.")
# Current price
current_price = float(gold_prices[-1])

print(f"\nCurrent Gold Price: ${current_price:.2f}")
```

تعیین قیمت روز طلا

مثال:پیش بینی قیمت طلا در شش ماه آینده

```
# ========================================
# 3. Define market states
# ========================================
def get_state(price_change):
    if price_change < -1.0:
        return "Sharp Drop"
    elif -1.0 <= price_change < 0:
        return "Mild Drop"
    elif 0 <= price_change < 1.0:
        return "Mild Rise"
    else:
        return "Sharp Rise"

states = [
    "Sharp Drop",
    "Mild Drop",
    "Mild Rise",
    "Sharp Rise"
]
state_indices = {s: i for i, s in enumerate(states)}
```

دسته بندی تغییرات قیمت طلا

بازار فقط چهار وضعیت دارد:

وضعیت	معنی
Sharp Drop	افت شدید
Mild Drop	افت ملایم
Mild Rise	رشد ملایم
Sharp Rise	رشد شدید

مثال:پیش بینی قیمت طلا در شش ماه آینده

```
# 4. Calculate daily percentage changes
# ========================================
price_changes = (
    np.diff(gold_prices)
    / gold_prices[:-1]
) * 100
# Convert changes to state sequence
state_sequence = [
    get_state(change)
    for change in price_changes
]

# 5. Build transition matrix
# ========================================
transition_matrix = np.zeros((4, 4))
for i in range(len(state_sequence) - 1):
    current_state = state_indices[state_sequence[i]]
    next_state = state_indices[state_sequence[i + 1]]

    transition_matrix[current_state, next_state] += 1

# Normalize rows safely
row_sums = transition_matrix.sum(axis=1, keepdims=True)
# Avoid division by zero
row_sums[row_sums == 0] = 1
transition_matrix = transition_matrix / row_sums
print("\nTransition Matrix:\n")
print(pd.DataFrame(
    transition_matrix,
    index=states,
    columns=states
))
```

تعیین تغییرات قیمت طلا

تشکیل زنجیره مارکوف:
بعد از هر وضعیت، بازار بیشتر به کدام وضعیت رفته است؟

ساخت ماتریس انتقال
بعد از هر وضعیت، بازار با چه احتمالی به سایر وضعیت ها رفته است؟

مثال:پیش بینی قیمت طلا در شش ماه آینده

```
# 6. Predict future prices
# ============================================
def predict_future(
    last_price,
    transition_matrix,
    state_sequence,
    days=180):
    predicted_prices = [last_price]
    # Start from last observed state
    current_state = state_sequence[-1]
    for _ in range(days):
        probs = transition_matrix[
            state_indices[current_state] ]
        # Safety check
        if np.sum(probs) == 0:
            probs = np.ones(4) / 4
        # Choose next state
        next_state = np.random.choice(
            states,
            p=probs)
        # Simulate movement
        if next_state == "Sharp Drop":
            change = np.random.uniform(-2.5, -1.0)
        elif next_state == "Mild Drop":
            change = np.random.uniform(-1.0, 0)
        elif next_state == "Mild Rise":
            change = np.random.uniform(0, 1.0)
        else:
            change = np.random.uniform(1.0, 2.5)
```

برنامه بر اساس احتمال‌های مارکوف وضعیت بعدی را انتخاب می‌کند.
انتخاب تصادفی اما وزن‌دار مطابق با ماتریس انتقال انجام می‌شود.

مثال:پیش بینی قیمت طلا در شش ماه آینده

```
            # New price
            new_price = predicted_prices[-1] * (
                1 + change / 100
            )

            predicted_prices.append(new_price)
            current_state = next_state
    return predicted_prices
# Run prediction
predicted_prices = predict_future(
    current_price,
    transition_matrix,
    state_sequence,
    days=180
)
# 7. Create future dates
# ============================================

future_dates = pd.date_range(
    start=dates[-1],
    periods=181,
    freq='D'
)
```

مثال:پیش بینی قیمت طلا در شش ماه آینده

```
# 8. Plot results
# ========================================
plt.figure(figsize=(14, 7))
# Historical prices
plt.plot(
    dates[-180:],
    gold_prices[-180:],
    label='Historical Gold Price'
)
# Predicted prices
plt.plot(
    future_dates,
    predicted_prices,
    '--',
    label='6-Month Prediction'
)
# Current price line
plt.axhline(
    y=current_price,
    linestyle=':',
    label=f'Current Price (${current_price:.2f})'
)
plt.title('Gold Price Prediction Using Markov Chain')
plt.xlabel('Date')
plt.ylabel('Gold Price (USD/oz)')
plt.legend()
plt.grid(True)
plt.show()
```


مثال:پیش بینی قیمت طلا در شش ماه آینده

```
# 9. Prediction statistics
# ========================================
print("\nPrediction Results")
print("-" * 40)
print(f"Starting Price: ${current_price:.2f}")
print(
    f"Predicted Price After 6 Months: "
    f"${predicted_prices[-1]:.2f}")
print(
    f"Maximum Predicted Price: "
    f"${max(predicted_prices):.2f}")
print(
    f"Minimum Predicted Price: "
    f"${min(predicted_prices):.2f}")
change_percent = (
    (predicted_prices[-1] - current_price)
    / current_price) * 100
print(
    f"Expected Change: "
    f"{change_percent:.2f}%")
```

Current Gold Price: $4499.30

Transition Matrix:

	Sharp Drop	Mild Drop	Mild Rise	Sharp Rise
Sharp Drop	0.134831	0.224719	0.359551	0.280899
Mild Drop	0.119760	0.251497	0.437126	0.191617
Mild Rise	0.123853	0.311927	0.348624	0.215596
Sharp Rise	0.224806	0.286822	0.286822	0.201550

ماتریس انتقال

یعنی اگر امروز بازار Mild Riseباشد:

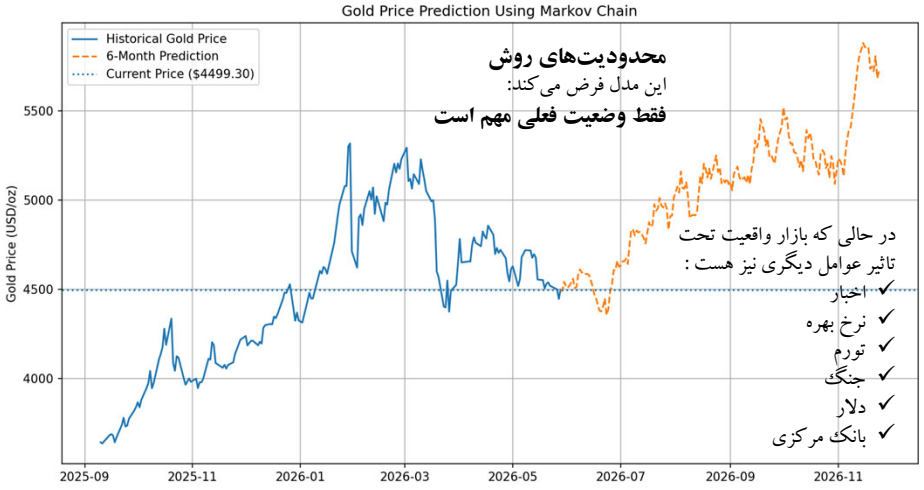
۳۵٪ احتمال دارد فردا Mild Rise

۳۱٪ احتمال Mild Drop

۲۱٪ احتمال Sharp Rise

۱۲٪ احتمال Sharp Drop باشد

پیش بینی قیمت طلا در شش ماه آینده



محدودیت‌های روش

این مدل فرض می‌کند:

فقط وضعیت فعلی مهم است

در حالی که بازار واقعیت تحت تاثیر عوامل دیگری نیز هست :

- ✓ اخبار
- ✓ نرخ بهره
- ✓ تورم
- ✓ جنگ
- ✓ دلار
- ✓ بانک مرکزی

Adv. App. of AI and DT 47

فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

Adv. App. of AI and DT 48

Q-learning

یادگیری Q : الگوریتم یادگیری تقویتی بدون سیاست (off-policy) که به عامل اجازه می‌دهد بدون نیاز به مدل محیط یاد بگیرد که در هر وضعیت چه عملی را انتخاب کند تا در بلندمدت پاداش بیشتری دریافت کند.

مقدار تابع Q را برای هر زوج (state, action) تخمین می‌زند و به‌روز می‌کند. تابع Q به عامل کمک می‌کند بهترین تصمیم را بگیرد.

- Q-learning: off-policy
  - use any policy to estimate Q

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

  - Q directly approximates Q* (Bellman optimality equation)
  - independent of the policy being followed
  - only requirement: keep updating each (s,a) pair

مزایا:
- ✓ بدون نیاز به مدل محیط
- ✓ الگوریتم پایه برای یادگیری عمیق
- ✓ ساده و قابل پیاده‌سازی آسان

معایب:
- نیاز به حافظه زیاد برای حالات بزرگ
- کند بودن در محیط‌های پیوسته یا بزرگ
- گاهی ناپایدار در محیط‌های پیچیده

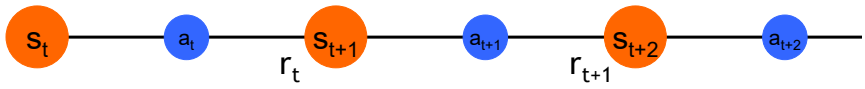
فهرست مطالب

- تعریف یادگیری تقویتی و مثال
- فرایند تصمیم گیری مارکوف Markov Decision Processes
- روشهای حل
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning
  - Q-learning
  - SARSA

SARSA: State,Action, Reward, State, Action

SARSAیک الگوریتم یادگیری **بر پایه سیاست (on-policy)** است که از تجربه‌ی واقعی در طول حرکت عامل (Agent) استفاده می‌کند، و تخمین تابع ارزش **Q(s,a)** را با استفاده از مشاهدات به‌روزرسانی می‌کند (یادگیری اختلاف زمانی).

- need Q(s,a), not just V(s)



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

- control
  - start with a random policy
  - update Q and π after each step
  - again, need ε-soft policies

مزایا:
- ✓ یادگیری مبتنی بر تجربه‌ی واقعی و هم‌راستا با رفتار عامل
- ✓ مناسب برای محیط‌های نویزی و پیچیده
- ✓ ساده و قابل پیاده‌سازی آسان

معایب:
- در مقایسه با Q-Learning ممکن است به جواب بهینه‌ی کمتری برسد (زیرا از سیاست جاری استفاده می‌کند، نه بهترین سیاست ممکن)

Adv. App. of AI and DT

Deep Reinforcement Learning

یادگیری تقویتی عمیق (Deep Reinforcement Learning)یعنی استفاده از شبکه‌های عصبی عمیق به عنوان تابع تقریبی در الگوریتم‌های یادگیری تقویتی برای حل مسائل پیچیده که فضای حالت آن‌ها بسیار بزرگ یا پیوسته است.

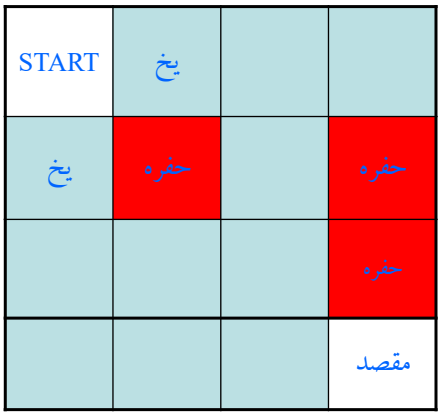
در یادگیری تقویتی کلاسیک، اگر فضای حالت بزرگ باشد، نگه داشتن Q-table غیرممکن می‌شود. در اینجا، یادگیری عمیق کمک می‌کند تا:
- به‌جای ذخیره Q در جدول، یک **شبکه عصبی** مقدار Q(s, a) را تقریب بزند.
- با این روش، می‌توان مسائل با فضای حالت بسیار زیاد مانند بازی‌ها، رباتیک و رانندگی خودکار را حل کرد.

الگوریتم	توضیح
DQN (Deep Q-Network)	از شبکه عصبی برای تخمین Q استفاده می‌کند
DDPG(Deep Deterministic Policy Gradient)	مناسب برای اعمال پیوسته (Continuous Actions)
A3C(Asynchronous Advantage Actor-Critic) / **A2C**(Advantage Actor-Critic)	الگوریتم‌های Actor-Critic برای بهبود پایداری
PPO (Proximal Policy Optimization)	الگوریتم پایدار و قدرتمند برای یادگیری سیاست

Adv. App. of AI and DT 52

مثال SARSA

محیط FrozenLake یک ماتریس ۴×۴ از یخ و حفره است. هدف رسیدن به خانه مقصد بدون افتادن در حفره است.



نصب کتابخانه‌های مورد نیاز

pip install gym matplotlib numpy

استفاده از محیط FrozenLake که یکی از محیط‌های کلاسیک برای آموزش الگوریتم‌های RL است

مثال SARSA

14 RL SARSA.py

```
import numpy as np
import matplotlib.pyplot as plt
import gymnasium as gym  # Using the newer gymnasium package

class SafeTimeLimit(gym.Wrapper):
    """Custom time limit wrapper to avoid import issues"""
    def __init__(self, env, max_episode_steps=None):
        super().__init__(env)
        self._max_episode_steps = max_episode_steps
        self._elapsed_steps = 0

    def step(self, action):
        obs, reward, terminated, truncated, info = self.env.step(action)
        self._elapsed_steps += 1

        if self._elapsed_steps >= self._max_episode_steps:
            truncated = True

        return obs, reward, terminated, truncated, info

    def reset(self, **kwargs):
        self._elapsed_steps = 0
        return self.env.reset(**kwargs)
```

مثال SARSA

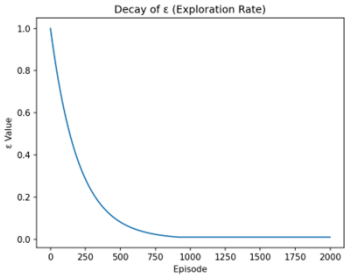
```
# Initialize environment with custom time limitenv = gym.make('FrozenLake-v1',
is_slippery=True,
render_mode='rgb_array')
env = SafeTimeLimit(env, max_episode_steps=100)  # Using our custom wrapper

# Q-learning parameters
state_size = env.observation_space.n
action_size = env.action_space.n
Q = np.zeros((state_size, action_size))

# Hyperparameters
alpha = 0.8
gamma = 0.95
epsilon = 1.0
epsilon_min = 0.01
epsilon_decay = 0.995
episodes = 2000
```

نرخ یادگیری: چقدر از اطلاعات جدید را در مقایسه با مقدار قدیمی بپذیریم – مقدار بیشتر یادگیری سریعتر

انتخاب تصادفی اولیه برای اکتشاف

حداقل مقدار برای اکتشاف

کاهش تدریجی در پایان هر اپیزود



سیاست ε-Greedy : ترکیبی هوشمندانه از اکتشاف و بهره‌برداری:

- با احتمال ε یک عمل تصادفی انتخاب می‌شود (**اکتشاف**)
- با احتمال ۱-ε عمل بهینه انتخاب می‌شود (**بهره‌برداری**)

مثال SARSA

On-Policy Method

همان سیاستی که برای تصمیم‌گیری استفاده می‌شود، برای یادگیری نیز استفاده می‌شود.

```
# Training loop
rewards = []
for episode in range(episodes):
    state, _ = env.reset()
    total_reward = 0

    # Initial action selection
    action = env.action_space.sample() if np.random.rand() < epsilon else
np.argmax(Q[state])
    while True:
        next_state, reward, terminated, truncated, _ = env.step(action)
        done = terminated or truncated

        # Next action selection
        next_action = (env.action_space.sample() if np.random.rand() < epsilon
                       else np.argmax(Q[next_state]))
        # SARSA update
        Q[state, action] += alpha * (reward + gamma * Q[next_state, next_action] -
Q[state, action])
```

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$$

برای تبدیل برنامه از SARSA به Q-Learning فقط کافی است دستور داخل مستطیل فوق را با دستور زیر جایگزین کنیم

```
    # SARSA update
Q[state, action] += alpha * ( reward + gamma * np.max(Q[next_state, :])- Q[state, action])
```

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

مثال SARSA

```
    state, action = next_state, next_action
    total_reward += reward
    if done:
        break
# Update exploration rate
epsilon = max(epsilon_min, epsilon * epsilon_decay)
rewards.append(total_reward)
```

```
# Plot results
window_size = 100
moving_avg = np.convolve(rewards, np.ones(window_size)/window_size, mode='valid')

plt.figure(figsize=(10, 5))
plt.plot(moving_avg)
plt.title("SARSA Learning Progress (FrozenLake)")
plt.xlabel("Episodes")
plt.ylabel(f"Average Reward ({window_size}-episode window)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

برنامه هیچ اطلاعی از مدل محیط ندارد:

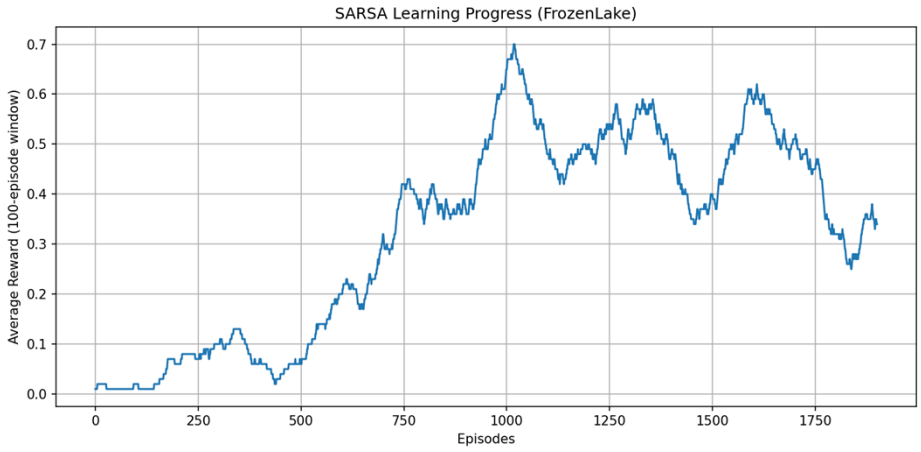
- ✓ احتمال انتقال حالت‌ها را نمی‌داند.
- ✓ تابع پاداش را نمی‌داند.
- ✓ فقط با تجربه کردن محیط یاد می‌گیرد.

بنابراین SARSA یک روش Model-Free RL است.

57

Adv. App. of AI and DT

مثال SARSA

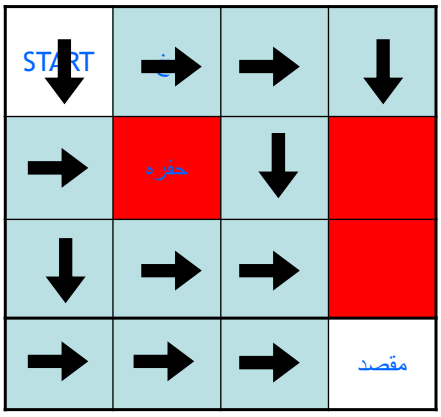


58

Adv. App. of AI and DT

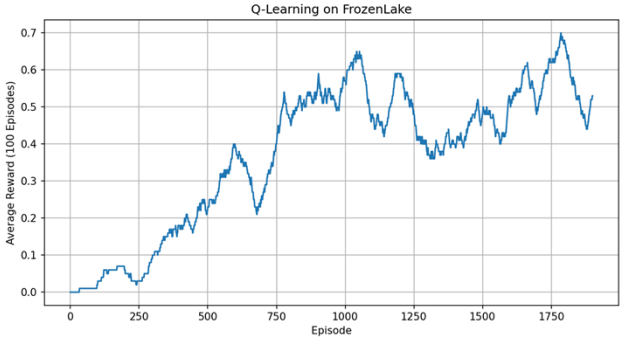
مثال SARSA

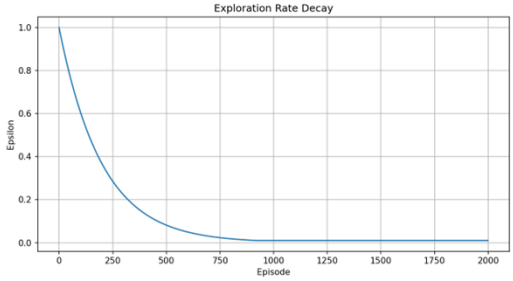
مسیر بهینه یادگیری شده توسط SARSA



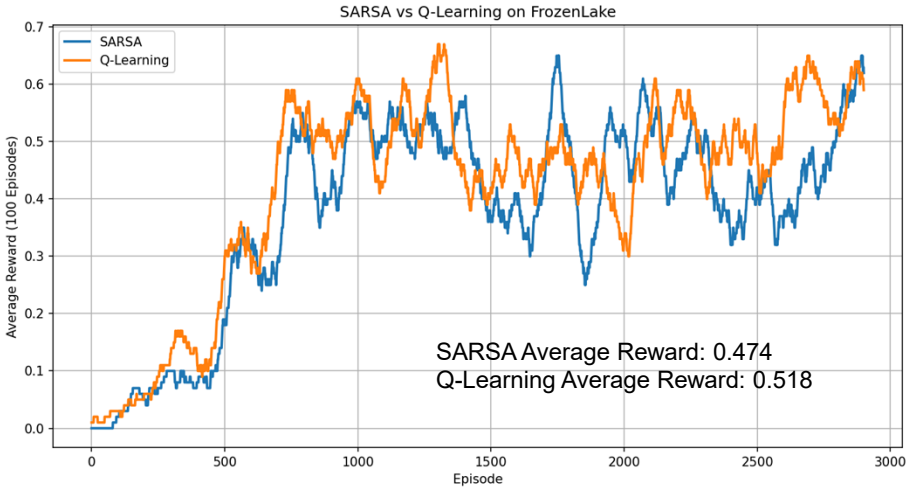
1. به دلیل لغزنده بودن محیط (is_slippery=True)، حرکات همیشه دقیقاً در جهت انتخاب شده انجام نمی‌شوند
2. مسیر نشان داده شده، مسیر بهینه‌ای است که عامل پس از یادگیری انتخاب می کند
3. احتمال موفقیت نهایی معمولاً بین ۶۰–۸۰٪ خواهد بود

مثال Q-learning





SARSA vs Q-Learning



SARSA vs Q-Learning

SARSA	Q-Learning
On-Policy	Off-Policy
از عمل واقعی بعدی استفاده می‌کند	از بهترین عمل ممکن استفاده می‌کند
Q[next_state,next_action]	max(Q[next_state,:])
محافظه‌کارتر	خوش‌بینانه‌تر
مناسب محیط‌های پرریسک	معمولاً سریع‌تر همگرا می‌شود

در محیط لغزنده FrozenLake، معمولاً Q-Learning سریع‌تر یاد می‌گیرد، اما SARSA اغلب سیاست ایمن‌تری پیدا می‌کند زیرا اثر اکتشاف (exploration) را در فرآیند یادگیری نیز در نظر می‌گیرد.

نمونه برای یادگیری تقویتی

بهینه‌سازی موقعیت آزمایش‌های ژئوتکنیک صحرایی

یک عامل یادگیری تقویتی، به صورت خودکار تصمیم می‌گیرد که آزمایش بعدی در چه فاصله‌ای از آزمایش قبلی انجام شود. این عامل بین دو هدف متضاد تعادل برقرار می‌کند: **حداقل کردن تعداد آزمایش‌ها** و **حداکثر کردن دقت** در تخمین نیمرخ خاک.

وضعیت (State) توسط یک بردار ویژگی ۵ بعدی شامل میانگین و انحراف معیار داده‌های آزمایش فعلی، تشابه با آزمایش‌های قبلی و موقعیت مکانی توصیف می‌شود .
اقدامات (Actions) شامل انتخاب فاصله تست بعدی در گام‌های ۲۵، ۵۰، ۱۰۰ یا ۱۵۰ واحدی در امتداد محور افقی است.

داده‌های آموزشی: مدل بر روی یک پایگاه داده شبیه‌سازی شده شامل ۱۶۰۰۰ تصویر از نیمرخ‌های خاک دو بعدی آموزش دیده است که ویژگی‌های واقعی مانند لایه‌های نامنظم و کانال‌های قدیمی را شبیه‌سازی می‌کند

این روش تطبیقی، در مقایسه با روش‌های سنتی با فاصله ثابت، خطای کمتری (RMSEکمتر) را برای همان تعداد آزمایش ارائه داده است.

B. Zuada Coelho et al.
Optimizing geotechnical in-situ site investigations using Deep Reinforcement Learning, Geodata and AI, Volume 6, 2026,

نمونه برای یادگیری تقویتی

بهینه‌سازی سازه نگهبان گودهای عمیق با در نظر گرفتن عدم قطعیت

ترکیب **استنتاج بیزین** (برای کمّی‌سازی عدم قطعیت) با **یادگیری تقویتی عمیق** (برای بهینه‌سازی تطبیقی).
مدل عددی قادر به شبیه‌سازی همزمان پدیده‌های مکانیکی، هیدرولیکی و حرارتی است.

•پیاده‌سازی این روش هوشمند در یک پروژه واقعی در شانگهای منجر به **کاهش ۴۲ درصدی نشست سطحی** (از ۲۸٫۵ به ۱۶٫۵ میلی‌متر) و **کاهش ۳۵ درصدی جابجایی دیوار** گود شد.

Gu, W. Bayesian
deep reinforcement learning for uncertainty quantification and adaptive support optimization in deep foundation pit engineering.
Sci Rep **15**, 35281 (2025). https://doi.org/10.1038/s41598-025-19002-w

تمرین برنامه نویسی

تمرین : بهینه‌سازی برنامه‌ریزی تعمیر و نگهداری یک سازه بتنی با استفاده از الگوریتم‌های یادگیری تقویتی Q-learning، SARSA، Monte Carlo

- سازه در هر دوره‌ی زمانی می‌تواند در یک سطح خرابی خاص (state) باشد:
  - سالم،
  - نیمه‌خراب،
  - شدیداً خراب
- در هر وضعیت، تصمیم(action) می‌تواند یکی از گزینه‌های زیر باشد:
  - هیچ کاری انجام نده
  - تعمیر جزئی
  - تعمیر اساسی
- هدف: کاهش مجموع هزینه‌ها در طول زمان (شامل هزینه تعمیر + ریسک خرابی شدید).
- پاداش بر اساس هزینه‌ها و ریسک‌ها تعریف می‌شود.
- با استفاده از الگوریتم‌های یادگیری تقویتی، سیاست بهینه‌ای برای تصمیم‌گیری در هر وضعیت به‌دست می‌آید.
- **خروجی مورد انتظار**
  - ✓ ماتریس Qبرای هر الگوریتم
  - ✓ سیاست بهینه (جدولی یا نموداری)
  - ✓ نمودار مقایسه هزینه تجمعی بین روش‌ها

Adv. App. of AI and DT 65