

گزارش پروژه دوم

پروژه Cliff Walking

درس: مبانی و کاربردهای هوش مصنوعی

استاد راهنما:

دکتر حسین کارشناس نجف آبادی

اعضای گروه:

علی‌اکبر احراری- 4003613001

مهرآذین مرزوق- 4003613055

پاییز 1402

فهرست

[گزارش کار الگوریتم 3](#_Toc153238218)

[**Policy\_evaluation** 3](#_Toc153238219)

[**Policy\_iteration** 5](#_Toc153238220)

[**Main** 6](#_Toc153238221)

[نمونه خروجی 8](#_Toc153238222)

[منابع 9](#_Toc153238223)

# گزارش کار الگوریتم

**Policy\_evaluation**

def policy\_evaluation(policy):  
 Vp = np.zeros(env.nS)  
 Qp = np.zeros((env.nS, env.nA)) / env.nA  
 converged = False  
 t = 1  
  
 while t < 1000 and not converged:  
 delta = 0  
 old\_Vp = Vp.copy()  
  
 for state in range(env.nS):  
 if cliffs.\_\_contains\_\_(state):  
 Vp[state] = -100  
 elif state == 47:  
 Vp[state] = 2000  
 else:  
 for action in range(env.nA):  
 ans = 0  
 for probability, state\_next, rewards, complete in env.P[state][action]:  
 ans += (1 / 2) \* (rewards + gamma \* old\_Vp[state\_next])  
 for probability, state\_next, rewards, complete in env.P[state][(action - 1) % 4]:  
 ans += (1 / 4) \* (rewards + gamma \* old\_Vp[state\_next])  
 for probability, state\_next, rewards, complete in env.P[state][(action + 1) % 4]:  
 ans += (1 / 4) \* (rewards + gamma \* old\_Vp[state\_next])  
 Qp[state][action] = ans  
  
 Vp[state] = Qp[state][int(policy[state])]  
  
 # Calculate the change in utility value  
 delta = np.max(np.abs(old\_Vp - Vp[state]))  
  
 if delta < theta:  
 converged = True  
 t += 1  
  
 return Vp, Qp

این تابع برای تخمین ارزش valueها و q\_valueها با استفاده از یک سیاست به عنوان ورودی به کار می‌رود. مراحل اجرای این تابع بدین صورت است که ابتدا ارزش وضعیت های مختلف را با 0 مقداردهی اولیه می‌کنیم. دو متغیر برای تشخیص شرایط اتمام حلقه(محدودیت تکرار و همگرایی) تعریف می‌کنیم. برای مقایسه‌کردن Vp فعلی با Vpهای جدید، از متغیر old\_Vp استفاده می‌کنیم.

برای هر حالت در فضای بازی، حالات مختلفی که عامل می‌تواند در آن قرار بگیرد را بررسی می‌کنیم و بنا به آن Vp مناسب را انتخاب می‌کنیم. برای cliffها ارزش منفی صد و برای هدف اصلی، 2000 را در نظر می‌گیریم. در صورتی که عامل در حالتی دیگر باشد به صورت دیگری عمل می‌کنیم.

این بخش از کد، برای به‌روزرسانی تخمین‌های توابع ارزش وضعیت‌ها و عمل‌ها با استفاده از معادله بلمن استفاده می‌شود. این احتمالات وضعیت‌های بعدی، پاداش‌ها و اطلاعات ترانزیشن را از محیط دریافت می‌کند و بر اساس آن‌ها مقادیر توقعی تابع ارزش عمل را محاسبه می‌کند. سپس این مقادیر با وزن‌های مختلف به‌روزرسانی شده و تابع ارزش وضعیت نیز بر اساس سیاست جاری به‌روزرسانی می‌شود.

در پایان هر دور ایتریشن، میزان تغییرات در تخمین‌های تابع ارزش بررسی شده و اگر این تغییرات کمتر از یک حد مشخص (مانند theta) باشد، الگوریتم به‌صورت همگرا معلوم می‌شود و محاسبات متوقف می‌شود.

**Policy\_iteration**

def policy\_iteration():  
 policy = np.zeros(env.nS)  
 t = 1  
 converged = False  
 while t < 1000 and not converged:  
 old\_policy = policy.copy()  
 Vp, Qp = policy\_evaluation(policy)  
 for state in range(env.nS):  
 act = -1  
 maxQ = -np.inf  
 for action in range(env.nA):  
 if Qp[state][action] > maxQ:  
 if not (state >= 0) & (state < 12) & (action == 0):  
 if not (state > 35) & (state < 48) & (action == 2):  
 if not (state % 12 == 11) & (action == 1):  
 if not (state % 12 == 0) & (action == 3):  
 maxQ = Qp[state][action]  
 act = action  
  
 policy[state] = act  
  
 delta = np.max(np.abs(policy - old\_policy))  
 if delta < theta:  
 converged = True  
 t = t + 1  
  
 return policy

در این تابع به تکرار و بهینه سازی سیاست(policy) می‌پردازیم. ابتدا یک مقداردهی اولیه (مقدار 0) را در نظر می‌گیریم. برای تشخیص شرط اتمام حلقه، دو متغیر t و converged را مقداردهی اولیه می کنیم. در صورتی که تکرار از مقدار تعیین شده بگذرد و یا سیاست مورد نظر در نقطه ای همگرا شود، حلقه پایان می‌یابد. با استفاده از متغیر old\_policy می‌توان همگرایی سیاست را بررسی کرد. با استفاده از تابع policy\_evaluation مقادیر q\_value و value ها را بدست می‌آوریم.

حال به ازای هر state در محیط بازی، بین تمامی action های ممکن به انتخاب actionی می پردازیم که مقدارQp را به حداکثر ممکن برساند. در این بین، شروطی برای حرکت عامل وجود دارند که از برخورد آن به دیواره‌ها جلوگیری می‌کنند. پس از آن action مورد نظر انتخاب شده و سیاست به روز می شود.

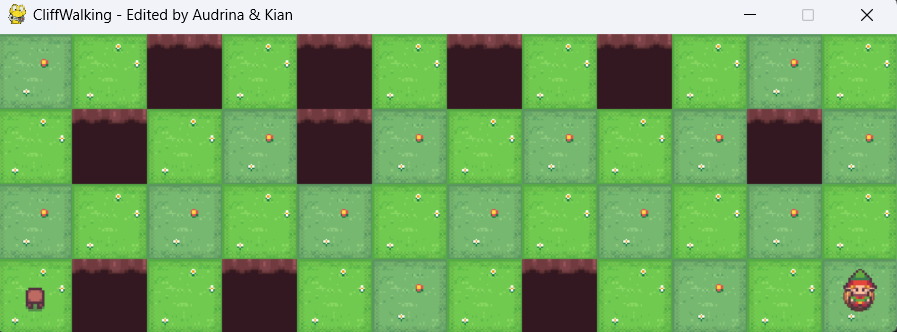
پس از هر بار به‌روز رسانی، بیشترین مقدار تغییرات در سیاست با استفاده از delta مورد بررسی قرار می‌گیرد و در صورتی که از مقدار theta کمتر باشد، شرط همگرایی برقرار شده است. در آخر، این تابع سیاست بهینه را بر می‌گرداند.

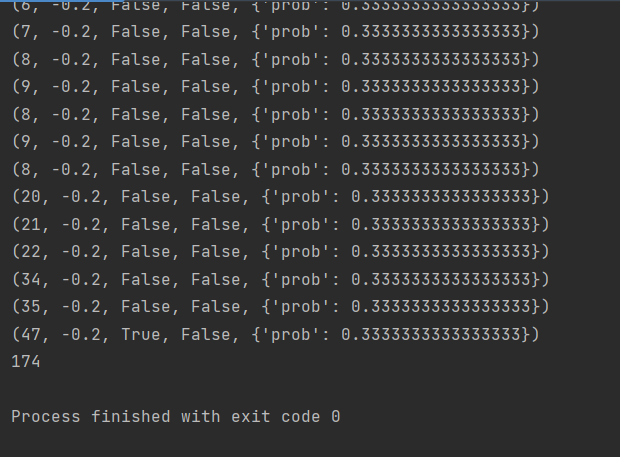
**Main**

# Create an environment  
env = CliffWalking(render\_mode="human")  
observation, info = env.reset(seed=30)  
cliffs = list()  
for i in range(10):  
 a = env.cliff\_positions  
 cliffs.append(a)  
  
gamma = 0.99  
theta = 1e-4  
  
Policy = policy\_iteration()  
  
print("Optimal policy:")  
print(Policy)  
  
# Define the maximum number of iterations  
max\_iter\_number = 1000  
done = False  
truncated = False  
reward = 0  
d = 0  
Action = Policy[observation]  
for i in range(max\_iter\_number):  
 print("----------------"f'{i}'"----------------")  
  
 while True:  
 # Perform the action and receive feedback from the environment  
 next\_state, reward, done, truncated, info = env.step(Action)  
 print(f'{next\_state, reward, done, truncated, info}')  
 Action = Policy[next\_state]  
 if info['prob'] == 1.0:  
 break  
 if done:  
 d = d + 1  
 env.reset()  
 break  
  
print(d)  
# Close the environment  
env.close()

در ابتدا، محیط بازی را ایجاد می‌کنیم و حالت نمایش آن را به روی human تنظیم می‌کنیم. حالت اولیه نیز به روی reset() تنظیم می‌شود. در قسمت بعدی کد cliff ها به محیط بازی اضافه می‌شوند. در قسمت بعدی کد، دو متغیر gamma و theta را برای استفاده در توابع برنامه مقداردهی می‌کنیم. پس از بدست آوردن سیاست بهینه به وسیله تابع policy\_iteration()، با محیط تعامل می‌کنیم. با استفاده از متغیر max\_iter\_number حداکثر تعداد دفعات تکرار مجاز را مشخص می‌کنیم و تغییرات وضعیت را چاپ می‌کنیم. در آخر نیز محیط را به پایان می‌رسانیم.

# نمونه خروجی





در اینجا می‌توان دید که برنامه 174 بار توانسته به هدف برسد.

# 

# منابع

[Bing AI - Search](https://www.bing.com/search?form=NTPCHB&q=Bing+AI&showconv=1)

[Bard (google.com)](https://bard.google.com/chat)

[ChatGPT (openai.com)](https://chat.openai.com/)

Artificial Intelligence: A Modern Approach, Textbook by Peter Norvig and Stuart J. Russell