

مبانی و کاربرد های هوش مصنوعی

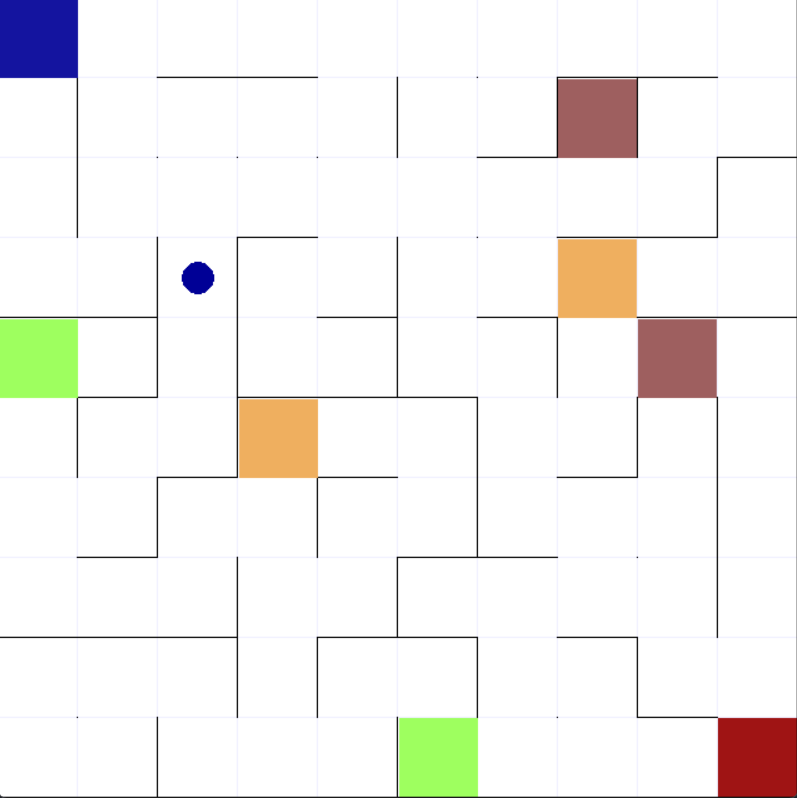
فاز سوم پروژه

مسیریابی به کمک یادگیری تقویتی در

محیط هزارتو

آرین جعفری

محمد حسن حیدری



* نمای کلی پروژه

محیطی که عامل در این فاز در آن مشغول به مسیریابی است ، محیطی غیرقطعی ، مشاهده پذیر جزئی و فاقد مدل انتقال میباشد ؛ شرایطی که عامل را مجبور میکند همزان با اکتشاف ، محیط و ارزش هر کنش را یاد بگیرد . محیط ، یک هزار تو ( Maze ) و شامل تعدادی پورتال است که در صورت ورود عامل به هر یک از این پورتال ها ، به پورتال همرنگ و جفت آن وارد میشود . خانه ای که عامل از آن یادگیری را شروع میکند خانه ی آبی رنگ با مختصات ( 0, 0 ) و خانه مقصد ، خانه ی قرمز با مختصات ( n, n ) است .

* یادگیری تقویتی

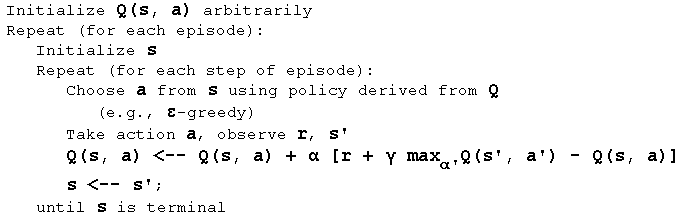
یادگیری تقویتی یا Reinforcement Learning نوعی یادگیری ماشین است که با فرایندهای تصمیم‌گیری متوالی سروکار دارد و شامل یک عامل ( Agent ) ، یک محیط ( Environment ) و یک مکانیسم بازخورد برای هدایت اقدامات عامل است. عامل یاد می‌گیرد که اقداماتی را در محیط انجام دهد تا سیگنال پاداش تجمعی

( Returned Reward ) را به حداکثر برساند. این سیگنال پاداش جمعی به‌ عنوان نیروی محرکه برای یادگیری عمل می‌کند.

* الگوریتم Q Learning

در فرایند های تصمیم مارکوف ، کنش عامل تنها وابسته به موقعیتی است که اکنون در آن قرار دارد ، و نه اینکه چگونه و با چه کنش هایی به موقعیت فعلی رسیده است . عامل در هر زمان با توجه به سیاست خود ، بهترین کنش را انتخاب میکند . در سیستم هایی با سیاست از قبل مشخص شده نیازی به یادگیری بهترین سیاست نیست ؛ اما هدف یادگیری عمیق این است که عامل در هر زمان ، کنشی را انجام دهد که بیشترین بازگشت پاداش را برای آن داشته باشد . با توجه به این مورد ، یادگیری محیط در شرایطی که پاداش ها از قبل مشخص شده و مدل انتقال محیط در دسترس باشد مفید نخواهد بود ؛ اما چنین شرایطی معمولا در مسائل مختلف در دسترس نیست . در واقع تا قبل از یادگیری محیط و مشخص کردن ارزش هر کنش ، عامل سیاست بهینه برای انتخاب بهترین کنش را در اختیار ندارد .

یکی از الگوریتم هایی که به ما کمک میکند ارزش هر کنش در هر موقعیت از محیط را بدست بیاوریم ، الگوریتم یادگیری کیفیت ( Q Learning ) است . به کمک این الگوریتم ارزش هر کنش به صورت متوالی بر اساس ارزش وزن دار کنش بعد و بر اساس نرخ یادگیری آلفا به روز رسانی خواهد شد .



* پیاده سازی

*import* gym  
*import* gym\_maze  
*import* numpy *as* np

مطابق با قطعه کد بالا ، ماژول های مورد نیاز برای پیاده سازی پروژه را استفاده میکنیم

*# Create an environment*env = gym.make("maze-random-10x10-plus-v0")  
observation = env.reset()  
  
*#initializing Q table and hyperparameters*q\_table = np.zeros((100, 4))  
learning\_rate = 0.1  
discount\_factor = 0.9 *#gamma  
  
#number of wins*k = 0

محیط شخصی سازی شده هزارتو را مطابق قطعه کد بالا پیاده سازی میکنیم . سپس جدول Q را مقدار دهی اولیه میکنیم . این جدول به تعداد موقعیت ها سطر و به تعداد کنش های ممکن برای هر موقعیت ، ستون دارد . محیط ما شامل 100 خانه و هر خانه شامل 4 کنش ممکن است .

نرخ یادگیری را معادل 0.1 و وزن کاهشی ( discount weight ) را معدل 0.9 قرار میدهیم . وزن کاهشی به ما گوشزد میکند که پاداشی که اکنون بدست آید بهتر است از پاداشی که در آینده در اختیار عامل قرار گیرد . ( این مقادیر بنا بر اختیار توسعه دهنده میتوانند شخصی سازی شوند )

حلقه اصلی پروژه جایی است که عامل همزمان با اکتشاف محیط ، ویژگی های آنرا یاد میگیرد . این حلقه به صورت زیر پیاده میشود :

*# Define the maximum number of iterations*NUM\_EPISODES = 1000  
  
*for* episode *in* range(NUM\_EPISODES):  
 state = env.reset()  
  
 *for* t *in* range(100):  
  
 row = state[0]  
 col = state[1]  
 state = int(row\*10 + col)  
  
 *#choosing best action based on our policy* action = np.argmax(q\_table[state])  
  
 next\_state, reward, done, info = env.step(action)  
 next\_max = np.max(q\_table[next\_state[0] \* 10 + next\_state[1]])  
  
 *# Update Q-value for the current state-action pair* q\_table[state][action] = (1 - learning\_rate) \* q\_table[state][action] + learning\_rate \* (reward + discount\_factor \* next\_max)

state = next\_state  
  
 *if* done:  
 k += 1  
 print(f'Wins : {k}')  
 *break* env.render()  
  
*# Close the environment*env.close()

در ابتدای هر اپیزود ، موقعیت عامل در خانه اول قرار میگیرد . با ورود به حلقه داخلی ، عامل به تعداد 100 کنش جلو میرود و با توجه به معادله اصلی الگوریتم ، پاداش کنش های انتخاب شده هر موقعیت را به روز میکند . سیاست عامل برای انتخاب کنش بعدی ، بهترین کنش است ؛ یعنی کنشی که بیشترین پاداش بازگشتی را داشته باشد . این مقدار در ابتدای شروع الگوریتم برای تمامی کنش ها در تمامی موقعیت ها برابر با صفر است . وظیفه الگوریتم این است که به تعداد مناسبی تکرار شده و این مقادیر را مطابق با پاداش خانه هدف به روز کند .

عامل پس از انتخاب هر کنش به احتمال 0.6 وارد خانه بعدی میشود : s 🡨 s’

پس از ورود به خانه ی بعدی ، دوباره معادله اصلی برای موقعیت فعلی عامل محاسبه شده و این چرخه به تعداد 100 کنش تکرار میشود تا اپیزود فعلی به پایان برسد .

* منابع استفاده شده
* Coursera Machine Learning Specialization
* Youtube contents
* Microsoft copilot