به نام خدا

فرزان رحمانی ۹۹۵۲۱۲۷۱

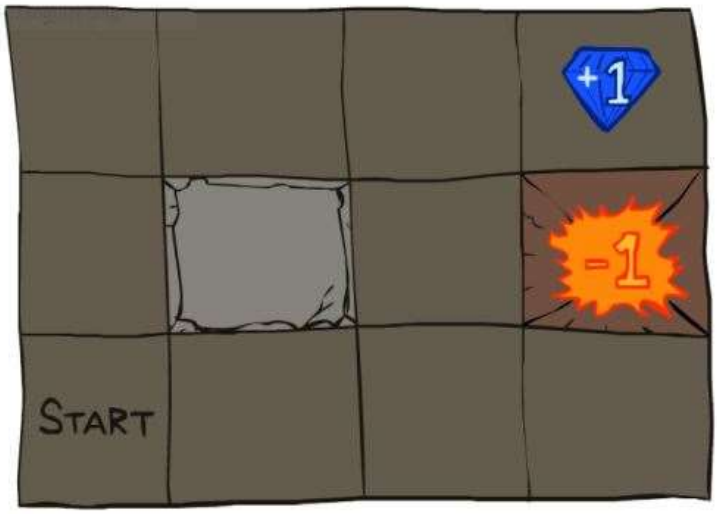
تمرین دوازدهم هوش مصنوعی و سیستم های خبره
قسمت عملی (سوال ۲ – پیاده سازی Q-learning)

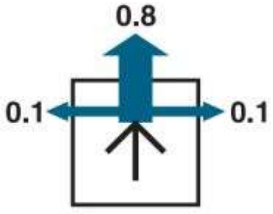
۲.۱ ارزیابی پیاده سازی

به چه علت است که همواره agent مطابق جهتی که شما میدهید حرکت نمیکند؟
زیر محیط بازی GridWorld محیط non-deterministic(stochastic) هست و transition model آن احتمالی می باشد.
به عبارتی دارای noise = 0.2 می باشد.

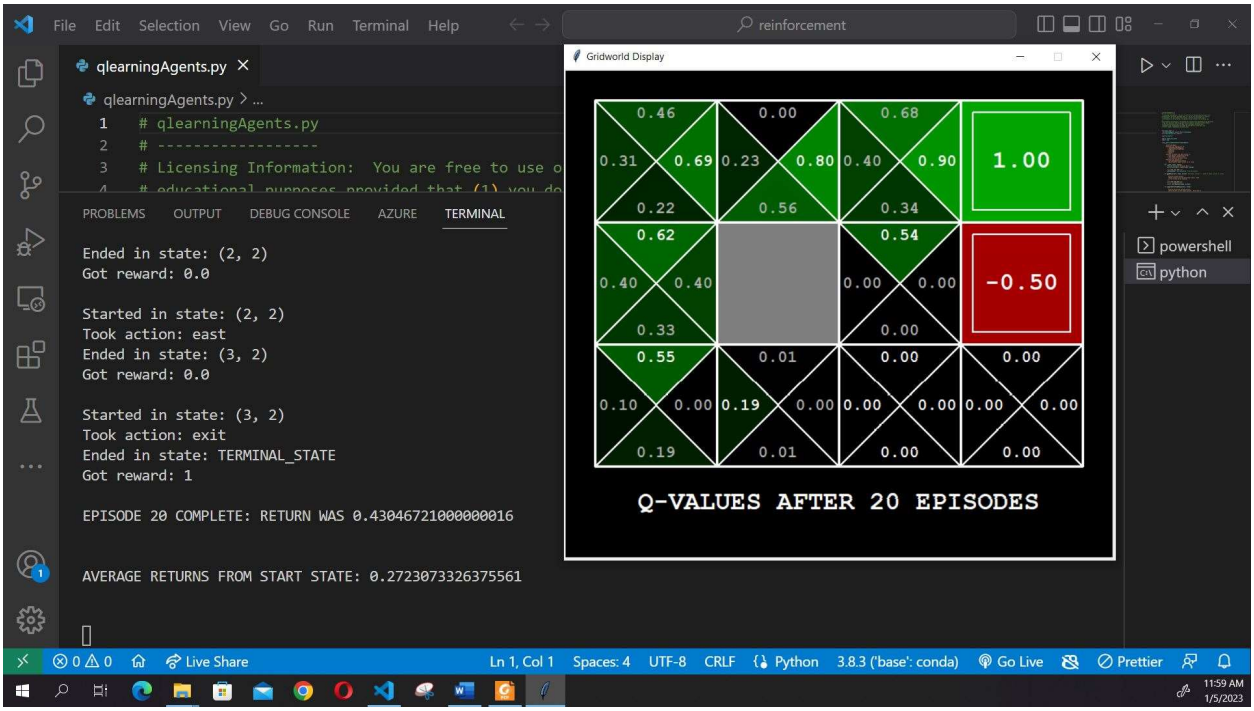
Noisy movement: actions do not always go as planned

- 80% of the time, the action North takes the agent North (if there is no wall there)
- 10% of the time, North takes the agent West; 10% East
- If there is a wall in the direction the agent would have been taken, the agent stays put

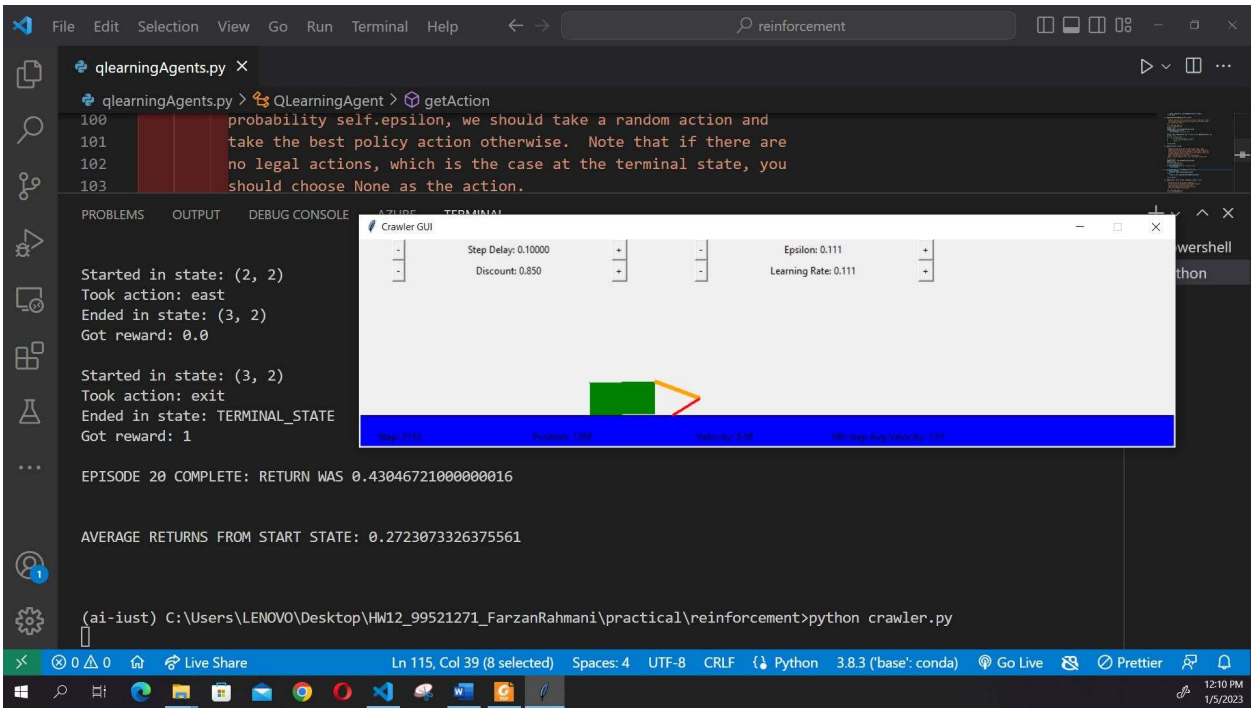




گزارش نتیجه بدست آمده از اجرا:



با تغییر iteration ها از ۵ به ۲۰ agent بیشتر محیط را یاد میگیرد و Q های بهتری داریم. اما هنوز کامل و همگرا نشده است و با افزایش iteration ها بهتر محیط را یاد میگرد. همان طور که دیده می شود بیشتر اطراف حالت پایانی بهینه کشف شده است چرا که از computeActionFromQValues استفاده کرده ایم. البته به علت استفاده از epsilon-greedy و flipCoin به انتخاب های جدید نیز امکان انتخاب شدن داده ایم. در این شکل QValue های هر state را می بینیم و برای استخراج سیاست از بین آنها max را انتخاب می کنیم. با توجه به مقادیر موجود عامل ما ابتدا بالا می رود و بعد از دوبار بالا رفتن به راس می رود. چرا که وجود مانع باعث می شود اگر طبق اکشن پیش نرفتیم و وجود noise باعث نشود داخل آتش بیفتیم.



Step Delay: با افزایش آن سرعت ربات کند تر می شود و با کاهش آن ربات سریع تر عمل می کند.

Epsilon(ε): احتمال حرکت تصادفی و نه با توجه به سیاست فعلی و QValue هایی که داریم و با افزایش آن Exploration رو محیط بیشتر می شود و اکشن های تصادفی بیشتری داریم.

- Simplest: random actions (ε-greedy)
  - Every time step, flip a coin
  - With (small) probability ε, act randomly
  - With (large) probability 1-ε, act on current policy

Discount(γ): باعث می شود تا به پاداش و استیت های فعلی نسبت به پاداش و استیت های آینده ارزش بیشتری بدهیم و از یه حدی به بعد صفر شوند.

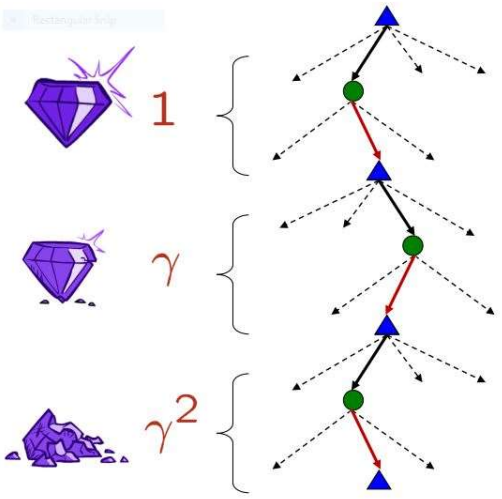
Discounting

- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially

Worth Now | Worth Next Step | Worth In Two Steps

Discounting

- How to discount?
  - Each time we descend a level, we multiply in the discount once
- Why discount?
  - Reward now is better than later
  - Can also think of it as a 1-gamma chance of ending the process at every step
  - Also helps our algorithms converge
- Example: discount of 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3
  - U([1,2,3]) < U([3,2,1])



- Discounting with γ solves the problem of infinite reward streams!
  - Geometric series: $1 + \gamma + \gamma^2 + \ldots = 1/(1 - \gamma)$
  - Assume rewards bounded by $\pm R_{max}$
  - Then $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ is bounded by $\pm R_{max}/(1 - \gamma)$ → bounded utilities

Learning Rate(α): مشخص می کند که برای میانگین QValue ها به نمونه های جدید چه ارزشی بدهد و هر چه بیشتر باشد نمونه های جدید ارزش بیشتری خواهند داشت و گذشته اش را فراموش می کند.

Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

temporal difference

Exponential Moving Average

- Exponential moving average
  - The running interpolation update: $\bar{x}_n = (1-\alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
  - Makes recent samples more important
  - Forgets about the past (distant past values were wrong anyway)
- Decreasing learning rate (alpha) can give converging averages

پایان