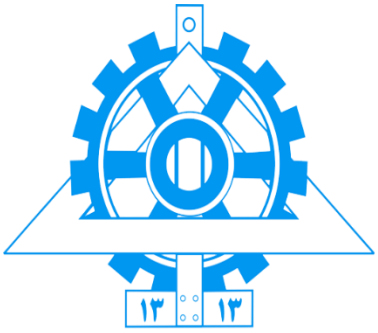
به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



پروژه سوم

بهینه سازی توزیع شده

دکتر کبریایی

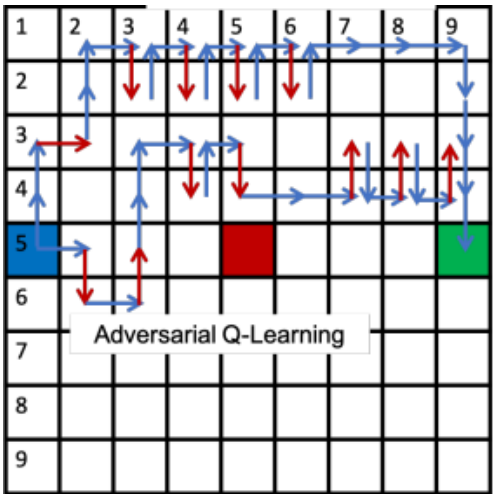
نام: شیرین

نام خانوادگی: جمشیدی

شماره دانشجویی : ۸۱۰۱۹۹۵۷۰

نیمسال اول ۰۳-۱۴۰۲

بخش ۱ : Adversarial Windy Gridworld



در این مسئله، یک بالن میخواهد از خانه‌ی آبی به خانه‌ی سبز برسد. خانه‌ی قرمز تله میباشد و افتادن بالن در این خانه، -۱۰۰ هزینه را در پی دارد. برای هر استپ عامل نیز -۱ هزینه متحمل میشویم. در این مسئله، رقیب بالن، باد میباشد که سعی دارد تا جایی که امکان دارد، بالن هزینه‌ی بیشتری متحمل شود. پس جایزه‌ی باد به این صورت تعریف میشود که اگر بالن را در تله بیندازد، +۱۰۰ تا جایزه و به ازای هر استپ بالن، +۱ جایزه میگیرد. برای حالت تک عامله، سیاست باد را ثابت درنظر میگیریم تا بتوان آن را عنصر ثابتی از محیط فرض کرد اما در حالت چند عامله، سیاست باد نیز در راستای ماکزیمم کردن جایزه‌اش میباشد و در رقابت با بالن میباشد.

بخش ۲: Single Agent, Q-learning

در این بخش با استفاده از الگوریتم Q-learning مسئله‌ی تک عامله را حل کردیم. مسئله را به ازای ۵۰ اپیزود حل کردیم که در هر اپیزود، موقعیت مکانی سیستم به خانه‌ی آبی بازیابی میشود و پس از آنکه با اپدیت Q-value و سیاست اپسیلون گریدی به خانه‌ی سبز رسید، هزینه کل محاسبه شده و به اپیزود بعدی میرویم.

$Initialize\ Q(s,a), for\ all\ s \in S, a \in A(s)$

$Loop\ for\ each\ episode$:

$Initialize\ S$

$Loop\ for\ each\ step\ of\ episode$:

$Choose\ A\ from\ S\ using\ greedy\ policy$

$Take\ action\ A, observe\ R, S'$

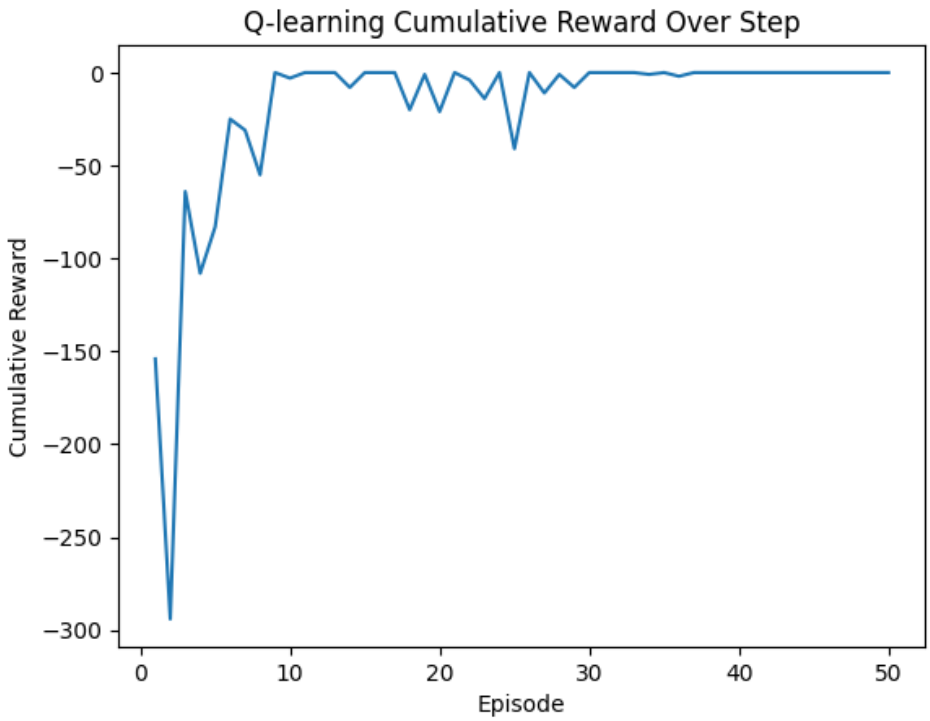
$$Q(s,a) \leftarrow Q(s,A) + \alpha[R + \gamma . \max_a Q(S',a) - Q(S,A)]$$

$S \leftarrow S'$

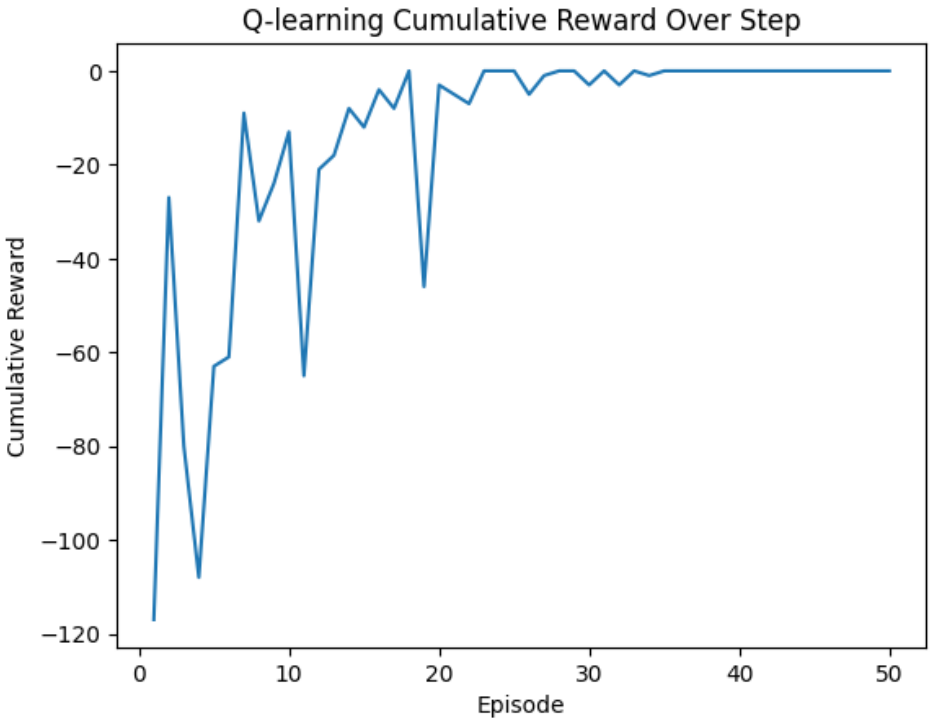
$Until\ S\ is\ terminal$

end

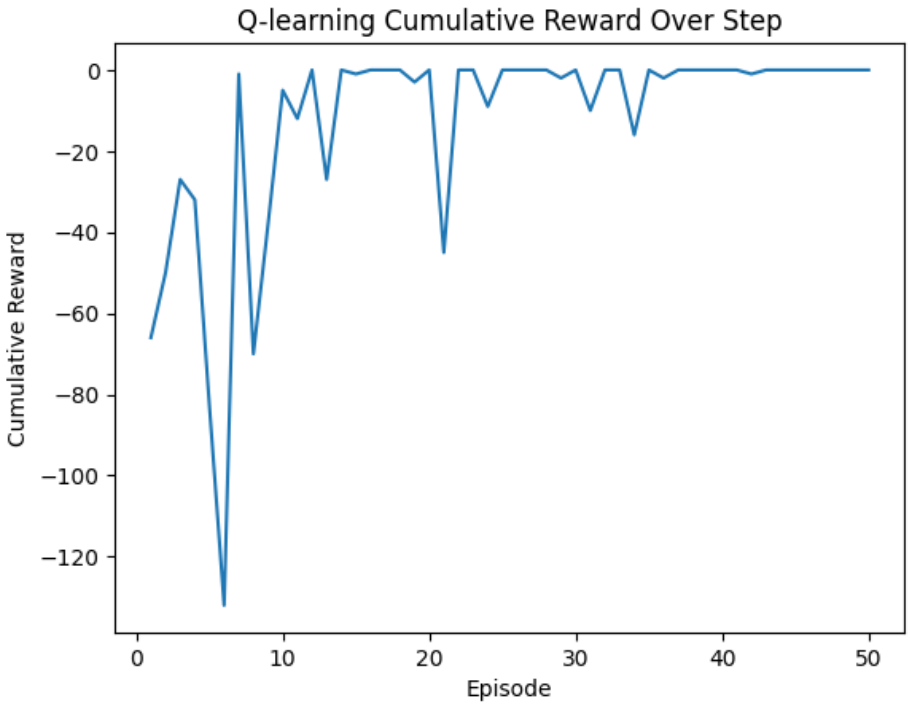
به ازای discount factor = 0.9 داریم:



به ازای discount factor = 0.5 داریم:



به ازای discount factor = 0.1 داریم:



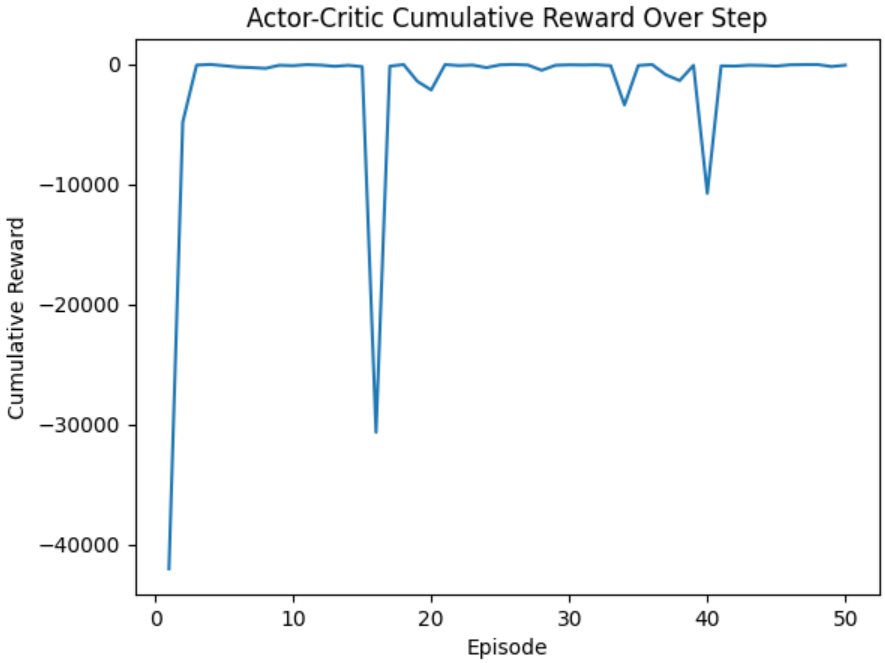
میبینیم که به ازای discount factorهای کمتر روند یادگیری، دامنه نوسانات بیشتری دارد و دیرتر به همگرایی میرسد. علت امر این است که discount factor کمتر موجب کمتر شدن افق دید عامل شده و ممکن است در مسیرهای sub-optimal محلی گیر کرده و موجب همگرایی ضعیف‌تر یا کلا عدم همگرایی شود.

بخش ۳: Single Agent, Actor-Critic

در این قسمت با استفاده از خطای temporal difference و soft-max policy، ارزش هر state را تخمین زده و حرکت بعدی با استفاده از این ارزش‌ها مشخص میشود. Actor، وظیفه‌ی یادگیری سیاستی را دارد که استیت‌ها را به اکشن‌ها مپ کند و critic، تابع ارزش را تخمین میزند تا کیفیت سیاست را ارزیابی کند.

Initialize: $V(s) = 0$, $Q(s,a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$,

Loop for each episode:
Initialize S
Loop for each step of episode:
Choose A from S using actor softmax policy derived from Q
Take action A, observe R, S'
At critic:
$\delta_t = r_{t+1} + \gamma.V(S_{t+1}) - V(S_t)$
$V_{t+1}(S_t) = V_t(S_t) + \alpha.\delta_t$
At actor:
$Q(S_t, a_t) \leftarrow Q(S_t, a_t) + \beta.\delta_t$
$\pi_t(S,a) = Pra_t = a|s_t = s = \frac{e^{Q(s,a)}}{\sum_b e^{Q(s,b)}}$
$S \leftarrow S'$
Until S is terminal
end



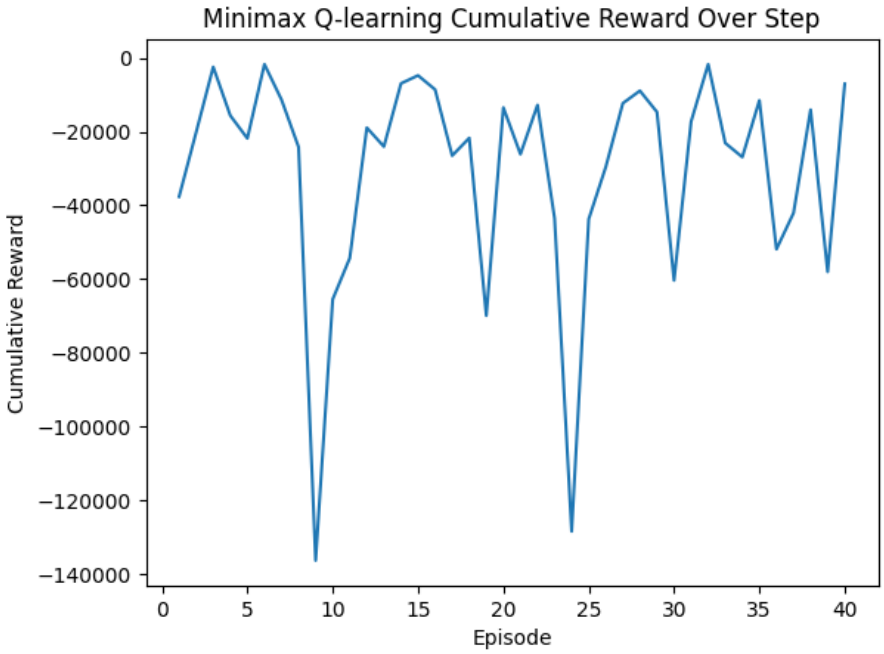
میبینیم که الگوریتم Q-learning با discount factor=0.9، عملکرد بهتری نسبت به روش عملگر نقاد دارد و علاوه بر نوسانات کمتر و سرعت بیشتر همگرایی، مینیمم هزینه‌ی بسیار کمتری دارد. که به این معناست که در این مسئله سیاست حریصانه و مبتنی بر تابع ارزش-عمل(Q)، عملکرد بهتری نسبت به تنها تابع ارزش هر استیت(V) دارد.

بخش ۴: Multi-Agent, Minimax Q-learning

در الگوریتم minimax، فرض بر این است که agent حریف، اکشنی را انتخاب میکند که بدترین حالت را برای ما رقم میزند و یادگیری بر اساس این فرض اتفاق می‌افتد. تابع ارزش خواهد شد:

$$V(s) = \max_{\mathrm{p}_{1\epsilon\pi(\mathrm{A}_1)}} \min_{a_2 \epsilon A_2} \sum_{a_1 \epsilon \mathrm{A}_1} P_1(a_1) Q(S, (a_1, a_2))$$

خواهیم داشت:

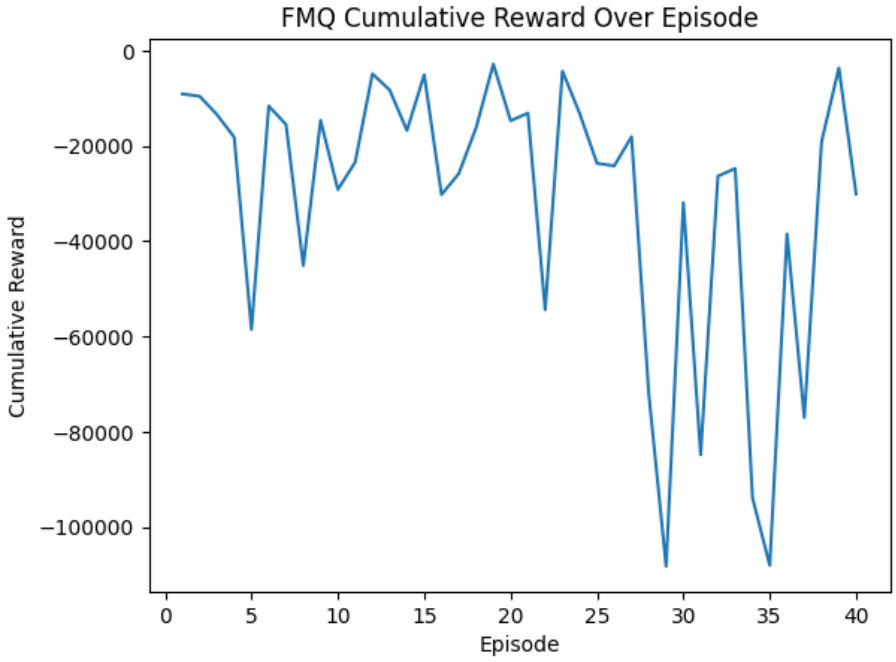


میبینیم که نوسانات این قسمت نسبت به قسمت تک عامله بسیار بیشتر است و در نهایت به همگرایی چشمگیری هم نرسیدیم.

علت این امر این است که علاوه بر بالن، باد هم توانسته یادگیری داشته باشد و در جهت ماکزیمم کردن ریوارد خود، که معادل است با ماکزیمم کردن هزینه بالن میباشد. پس در این قسمت باید به یک بازه همگرایی بسنده کنیم. هرچند که با کاهش اپسیلون و افزایش طول گام‌های اپسیلون و افزایش تعداد اپیزودها به نتایج بهتری رسید.

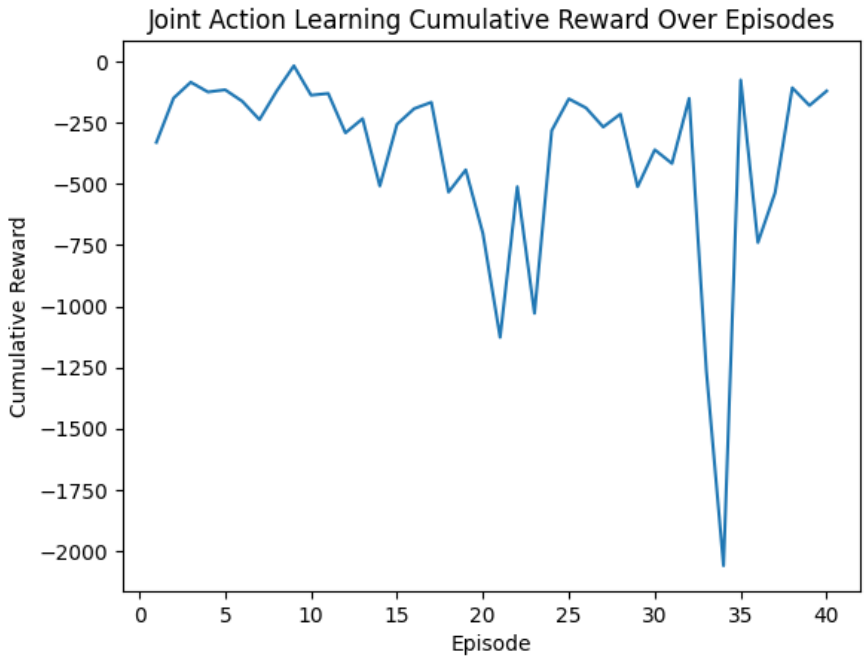
بخش ۵: Multi-Agent, FMQ

الگوریتم این قسمت در این مسئله بسیار مشابه با قسمت قبلی میباشد. در این الگوریتم، هر عامل در هر استپ ابتدا عامل دیگری را نادیده میگیرد و تصمیم میگیرد که کدام اکشن ریوارد بهینه‌تری منتج میشود و سپس با استفاده از آن اکشن بهینه و تعداد تکرار این ریوارد بهینه، تابع ارزش-عمل را آپدیت میکند.



بخش ۶: Multi-Agent, JAL

الگوریتم این روش از ایده belief based استفاده میکند و به این نحو است که هر عامل، محیط خود را observe میکند و بر اساس اینکه عامل دیگر i action ام را چندبار تکرار کرده است، یک تخمینی از حرکت بعدی عامل رقیب میزند و بر اساس آن تابع ارزش-عمل را اپدیت میکند و تصمیم میگیرد حرکت بعدی‌اش چه باشد.



میبینیم که بازه‌ی نوسان در این قسمت بسیار معقول‌تر و کمتر از دو قسمت قبلی میباشد.

بخش ۷: نتیجه‌گیری

همانطور که دیدیم، در قسمت تک عامله خیلی زود به همگرایی رسیدیم و بالن توانست پس از چند اپیزود محدود و آزمون و خطا، به نتیجه‌ی مطلوب برسد و در این مورد عملکرد روش Q-learning بهتر از روش actor-critic بود.

همچنین در قسمت دو عامله دیدیم که فقط میتوان به بازه‌ی همگرایی رسید و تمرکزمان باید روی کوچک‌تر کردن بازه‌ی همگرایی باشد که در این مورد عملکرد روش JAL، بهتر از روش‌های Q-learningای بود. علت اینکه به همگرایی نمیرسیم و تنها بازه‌ی همگرایی داریم، همانطور که بیان شد، این میباشد که علاوه بر بالن، هوا که عامل رقیب میباشد نیز توانایی یادگیری دارد و با مشاهده حرکات و سیاست‌های بالن و یادگیری آنها، میتواند حرکات خود را اصلاح کند به نحوی که بالن متحمل هزینه‌ی بیشتری شود.