۱ـ در این سوال دو مسیز حرکت به نام های A, B داده شده اند.

A = 1 14 15 16 17 18 19 20 21 8 7

B = 1 14 15 28 29 30 31 32 3 3 34 35 22 21 8 7

از ما خواسته شده است با درنظر گیری دو سیاست سارسا و کیو لرنینگ تعیین کنیم که کدام یک از مسیر ها با کدام سیاست اموزش میبینند.

با توجه به اینکه در سیاست کیو لرنینگ به این صورت عمل میکند که میخواهد همواره مقدار q value را بیشینه کند, خیلی به فاز exploration اهمیت نمیدهد و این باعث میشود که راه حل (در اینجا مسیر) بهینه (مسیر کوتاه تر) را نتواند پیدا کند, با این حساب مسیر B محتمل تر است که بهتر توسط q learning یادگرفته شود, زیرا که میزان پاداش عبور آن از خانه های سبز بیشتر است.

در مقابل سیاست سارسا است که برخلاف q learning که همواره سعی در بیشینه کردن پاداش دارد, روش سارسا با یک احتمالی از exploration هم بهره میگرد, بنابراین مسیر A با این سیاست بهتر یاد گرفته میشود.

البته باید این را در نظر داشته باشیم که میزان یادگیری در این دو روش به عوامل مختلفی discount factor, learning rate و exploration rate بستگی دارد.

۲- الف) یک رویکرد برای حل مساله استفاده از روش value iteration است. برای انجام این کار ما پاداش هر یک از مربع های سفید را منفی یک در نظر میگیریم و پاداش مربع های قرمز و سبز را به ترتیب منفی ۵ و ۵ در نظر میگیریم. همچنین مقدار پاداش در مربع های سفید را توسط معادله ی بلمن اپدیت میکنیم.

ب) با اضافه کردن عدد c ثابت را به همه پاداش هاو اگرچه سیاست بهینه تغییری نمی کند، اما چون افزودن یک ثابت بر ترتیب نسبی مقادیر تأثیر نمی‌گذاردو فقط بر مقادیر مطلق آنها تأثیر می‌گذارد, مقادیر مربع های سفید با مقدار c افزایش یا کاهش می یابند.

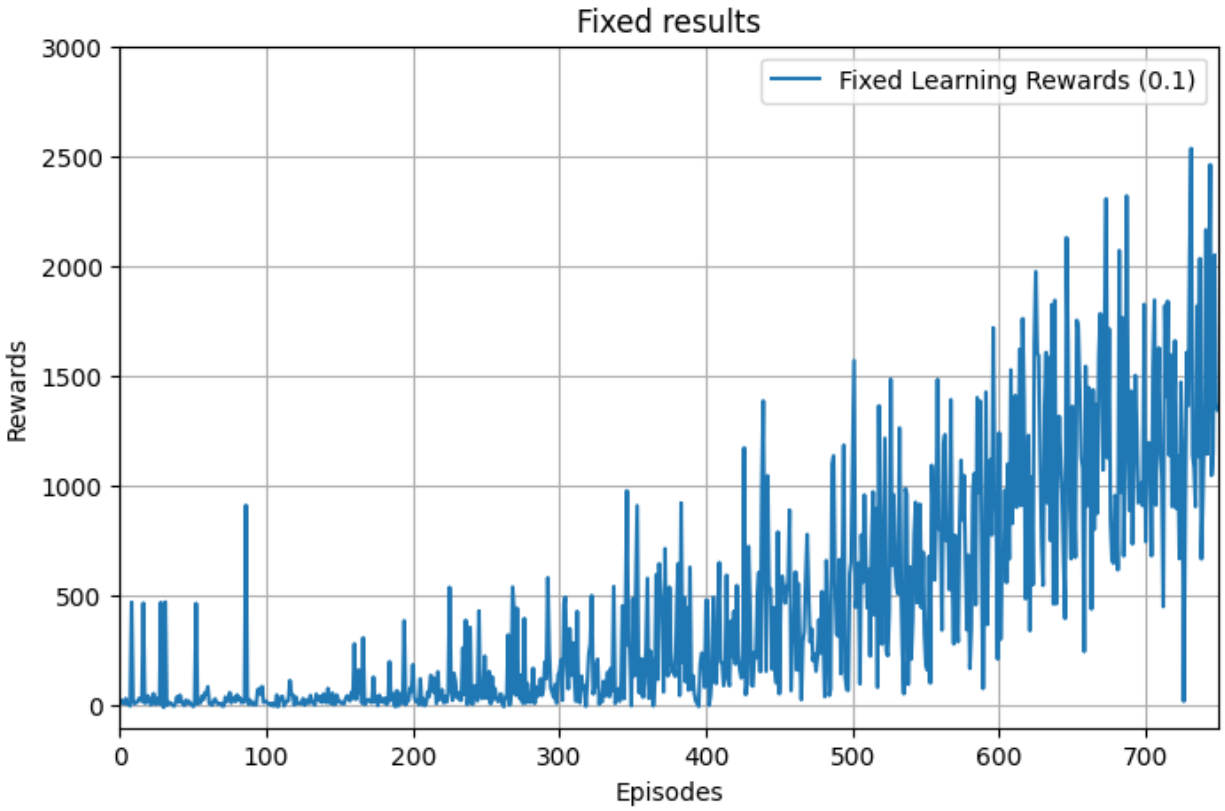
ج) تغییر γ می تواند سیاست بهینه را تغییر دهد. ضریب γ تعیین کننده تأکید بر پاداش های فوری در مقابل پاداش های آینده است. اگر γ نزدیک به 1 باشد، عامل پاداش های آینده را اولویت بندی می کند و ممکن است مسیر طولانی تری را برای رسیدن به هدف طی کند. اگر γ نزدیک به 0 باشد، عامل پاداش های فوری را در اولویت قرار می دهد و ممکن است مسیر کوتاه تری را طی کند.

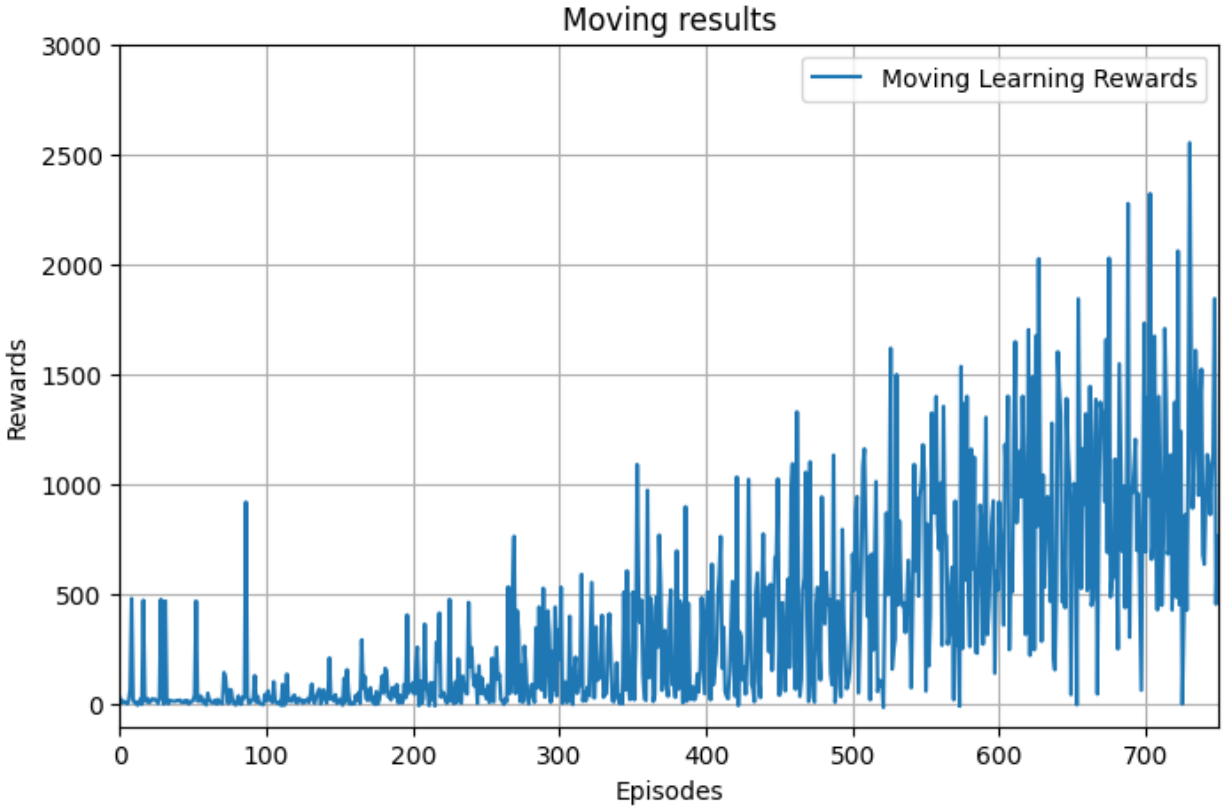
- در حل این سوال از chay gpt استفاده شده است.
- پرامپت مدنظر به این صورت بوده که ابتدا صورت مساله و شکل را برای مدل توضیع دادم سپس سوالات را پرسیدم.

۳- Expected-SARSA : این روش تابع action-value را به طور مکرر تخمین می زند و یک سیاست بهینه را با به روز رسانی سیاست بر اساس مقادیر Q تخمین زده می آموزد. به دلیل اینکه همه جفت‌های state-action به تعداد دفعات زیادی دیده می‌شوند، به سیاست بهینه همگرا می‌شود.
SARSA: این روش هم تابع action- value را به صورت کمرر تخمین می زند، اما سیاست را بر اساس جفت های مشاهده شده state-action-reward-sate-action بعدی به اپدیت می کند. این سیاست به یک سیاست بهینه e-optimal همگرا می شود، که لزوماً بهترین سیاست نیست، اما در حاشیه کمی از بهینه بودن قرار دارد. SARSA به exploration-explotation balance که توسط توسط الگوریتم epsilon-greedy ارایه میشود, نیاز دازد تا اکشن های مختلف را کشف کند. به طور کلی، Expected-SARSA و SARSA در مراحل به روز رسانی و مفهوم بهینه بودن متفاوت هستند.
Expected-SARSA مقدار مورد انتظار را بر روی اکشن های مختلف محاسبه می کند که منجر به واریانس کمتر در اپدیت کردن می شود، در حالی که SARSA مقادیر Q را بر اساس سیاست فعلی به روز می کند و به یک سیاست e-optimal همگرا می شود. هدف هر دو الگوریتم تخمین سیاست بهینه است، اما SARSA محافظه کارتر است و در حین محاسبه مقادیر Qسعی میکند که اکشن های مختلف را هم بررسی می کند.
در سیاست SARSA برخی اوقات که احتمال رخداد اکشن بهینه کم است این سیاست به جستوجوی اکشن های غیر بهینه میپردازد که این باعث میشود به سختی به اکشن بهینه همگرا شود.
در سیاست expected-SARSA با درنظر گرفتن تمام اکشن های ممکن در حالت بعدی و وزن‌دهی به احتمالات آنها، مقدار نوسان کمتری در به‌روزرسانی دارد و بیشتر به سیاست بهینه همگرا می‌کند.
بنابراین، با کاهش exploration و انتخاب تعداد بیشتری اکشن های بهینه از طریق مقدار انتظار، regret Expected-SARSA کاهش می‌یابد و به سیاست بهینه همگرا می‌شود.

سوالات طراحی

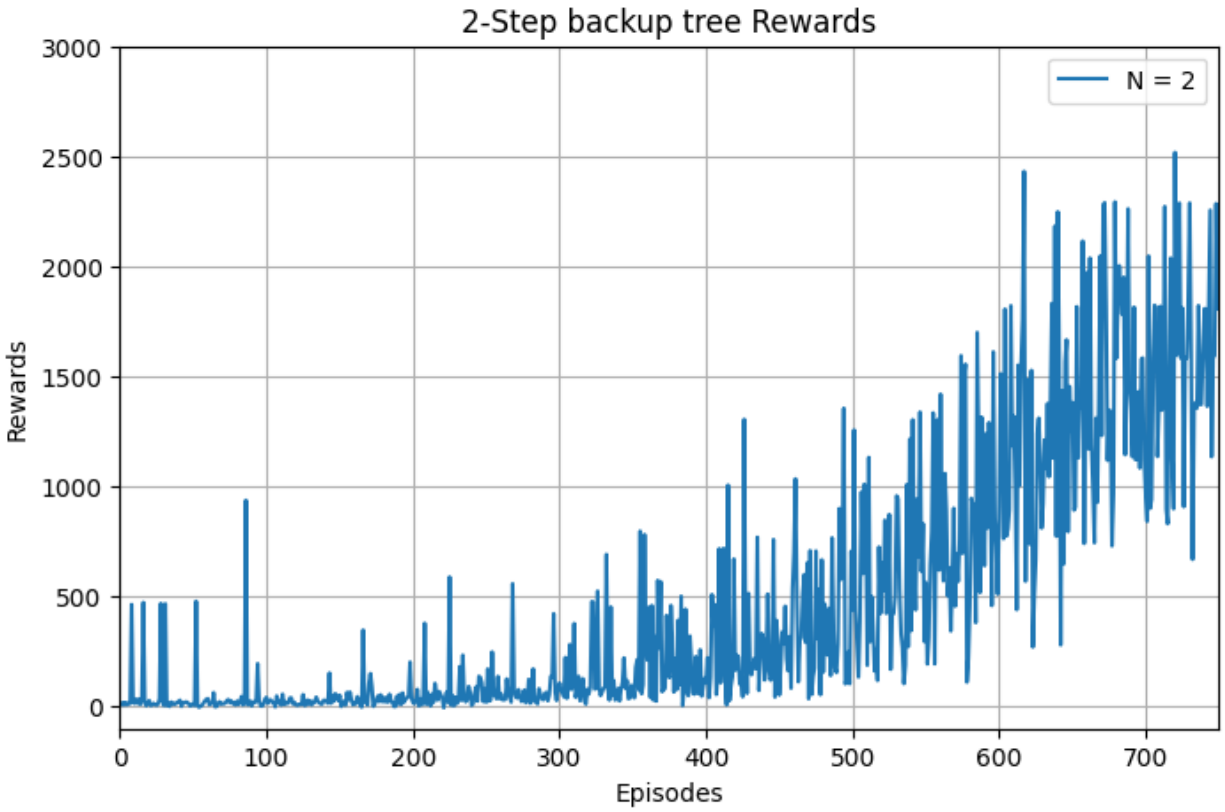
۱- با توجه به شکل های زیر مشاهده میشود که هنگامی که دارای نرخ یادگیری متغییر (در این جا کاهشی) باشیم به مقدار های بالاتر از پاداش در تعداد اپیزود ثابت دست پیدا میکنیم. همچنین باید توجه داشته باشیم که اگر از روش نرخ یادگیری متغییر استفاده کنیم با روند افرایشی به پاداش میرسیم. در صورتی که اگر از روش نرخ یادگیری ثابت استفاده کنیم حتی در زمانی که به پاداش ماکزیمم میرسیم باز هم دچار نوسان هستیم این نشان میدهد که عامل پایدار نیست.

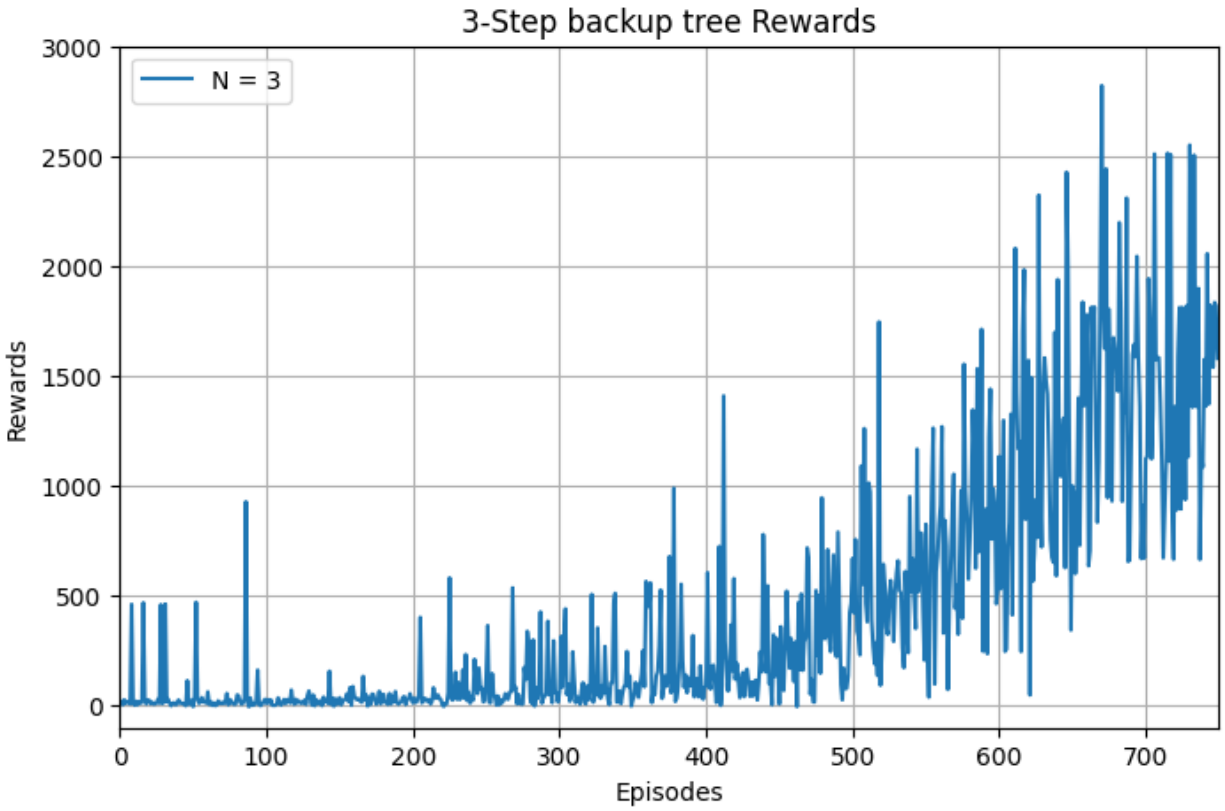

Fixed results
Fixed Learning Rewards (0.1)
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700

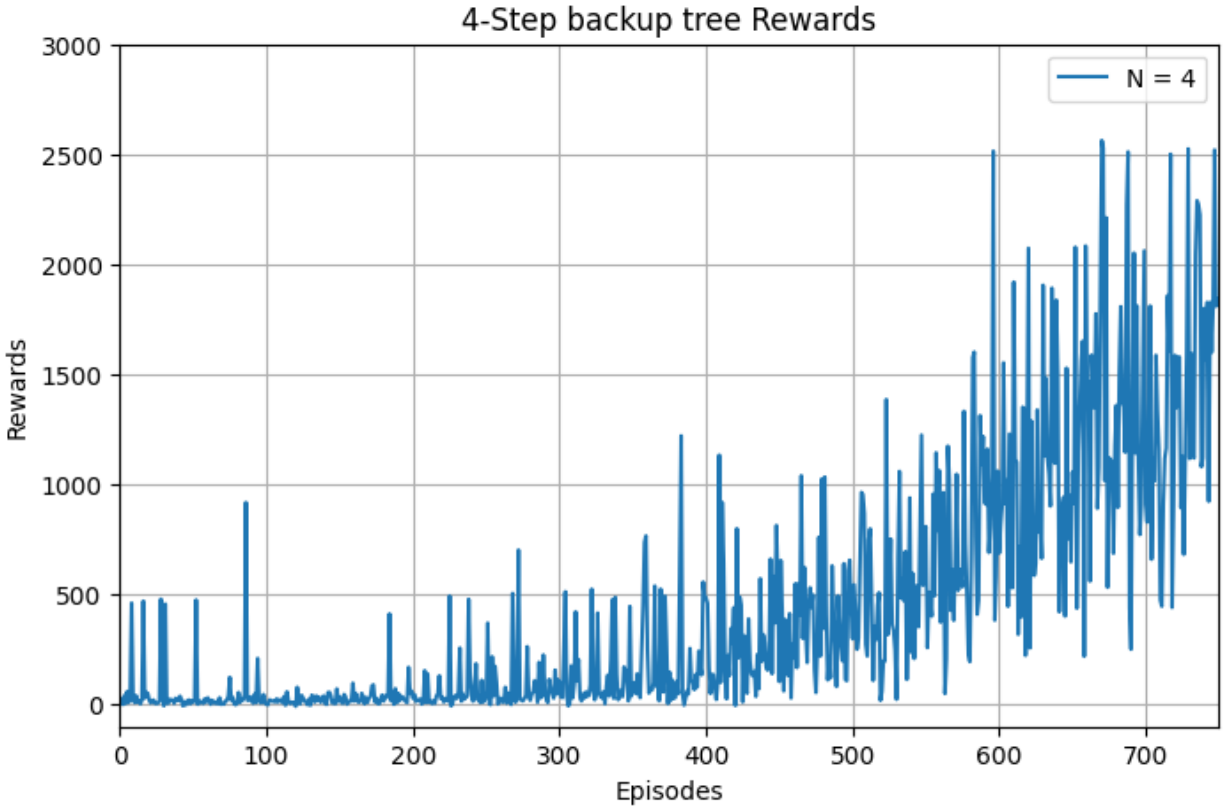

Moving results
Moving Learning Rewards
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700

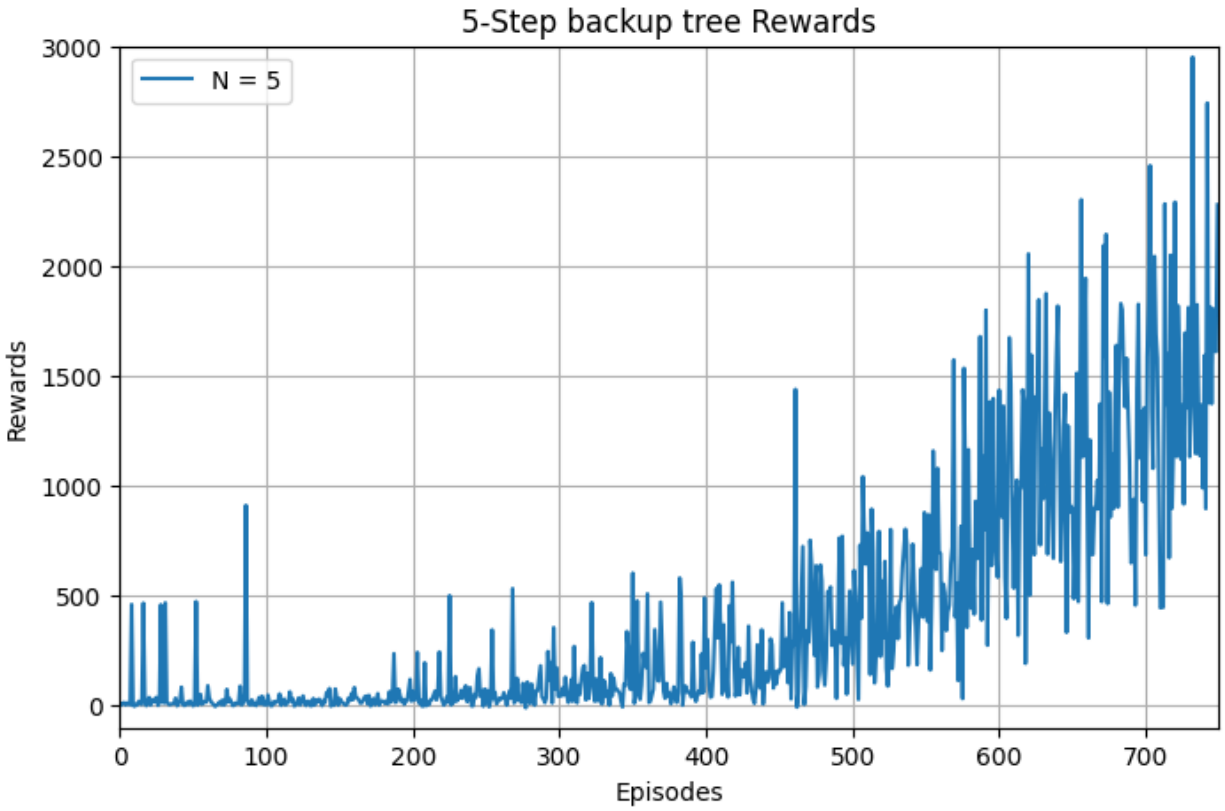
نرخ یادگیری که در اینجا الفا نام گذاری شده با جلوتر رفتن قسمت ها کاهش می یابد. این روند کاهشی با استفاده از تابع np.linspace و با استفاده از درونیابی خطی بین مقدار ماکزیمم و مینیمم الفا انجام میشود. تابع np.linspace سه آرگومان می گیرد: مقدار شروع (max_alpha)، مقدار پایان (min_alpha)، و تعداد مراحل این یک آرایه یک بعدی از قسمت های طولی با مقادیر مساوی بین max_alpha و min_alpha ایجاد می کند.

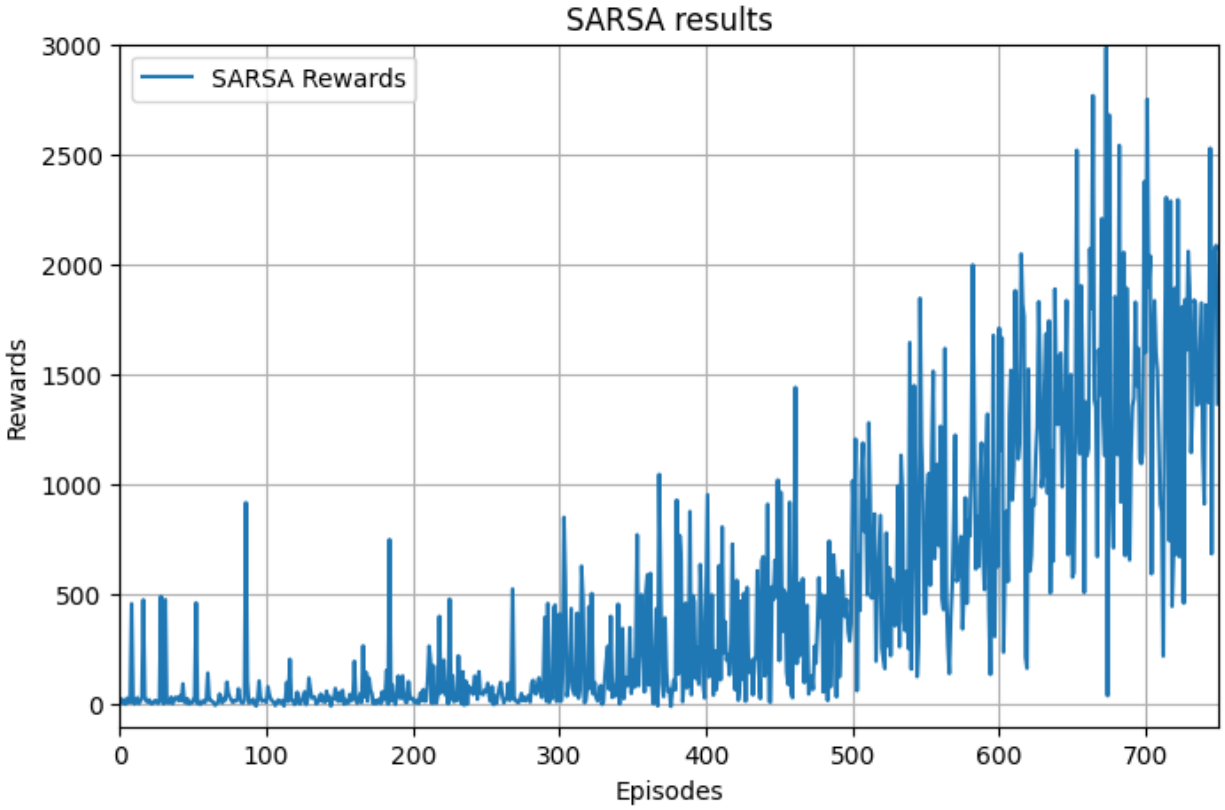
۲-




3-Step backup tree Rewards
N = 3
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700


4-Step backup tree Rewards
N = 4
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700


5-Step backup tree Rewards
N = 5
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700


SARSA results
SARSA Rewards
Rewards
Episodes
0
500
1000
1500
2000
2500
3000
0
100
200
300
400
500
600
700

همان طور که از تصویر قابل مشاهده است در هر دو روش SARSA و N step ابتدا مدل در فاز explore می باشد و بعد وارد فاز exploit می شود. البته باید توجه داشت که فاز explore کردن در برخی n ها طولانی تر از روش SARSA است. همچنین در N=5 تونستیم به پاداش بالاتری نسبت به سایر n ها برسیم.

۳.۵- در اینجا ابرپارامترها به طور مستقیم را تنظیم نمیشوند. این ابرپارامتر ها عبارت اند از:

tol: مقدار تحمل مورد استفاده برای همگرایی در تابع value_iteration. این تعیین می کند که چه زمانی باید تکرار را متوقف کرد و فرآیند تکرار ارزش را همگرا در نظر گرفت.

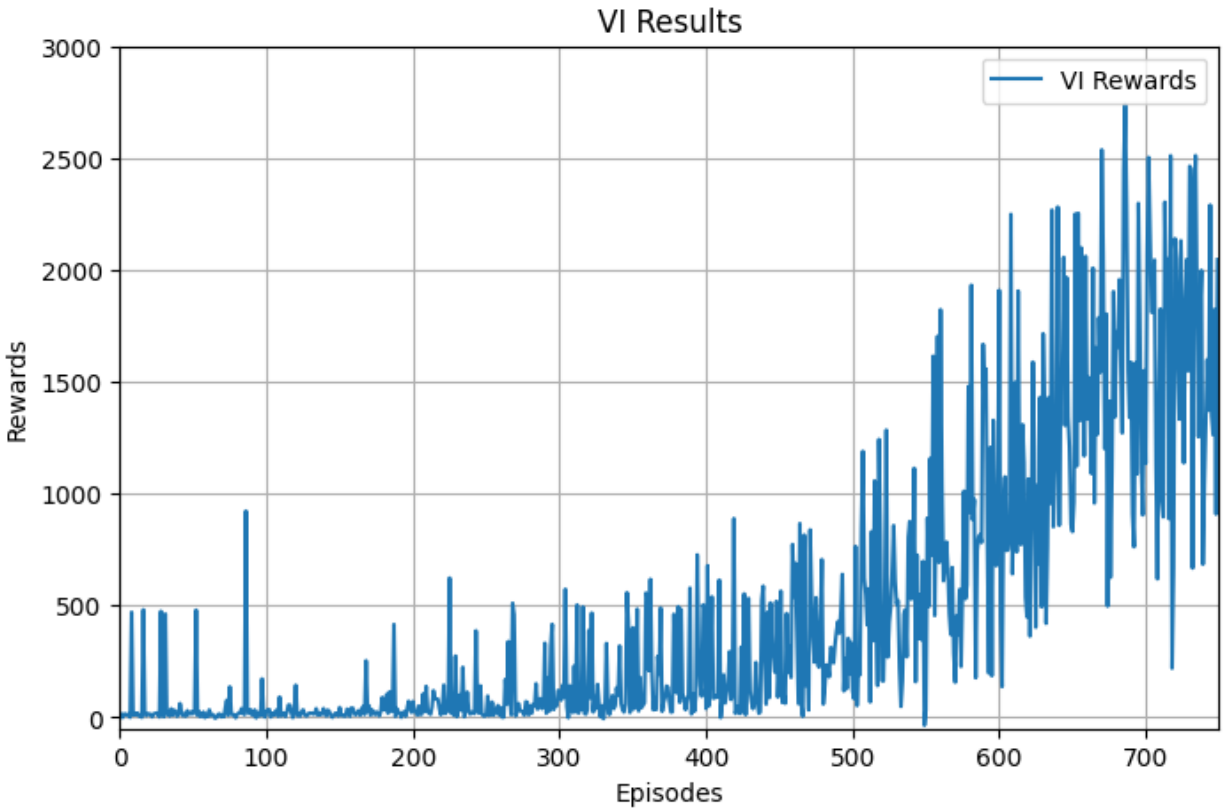
گاما: ضریب تخفیف مورد استفاده در محاسبه پاداش های آینده. برای تعیین مقدار حالت فعلی، در مقدار حالت بعدی در معادله بلمن ضرب می شود.

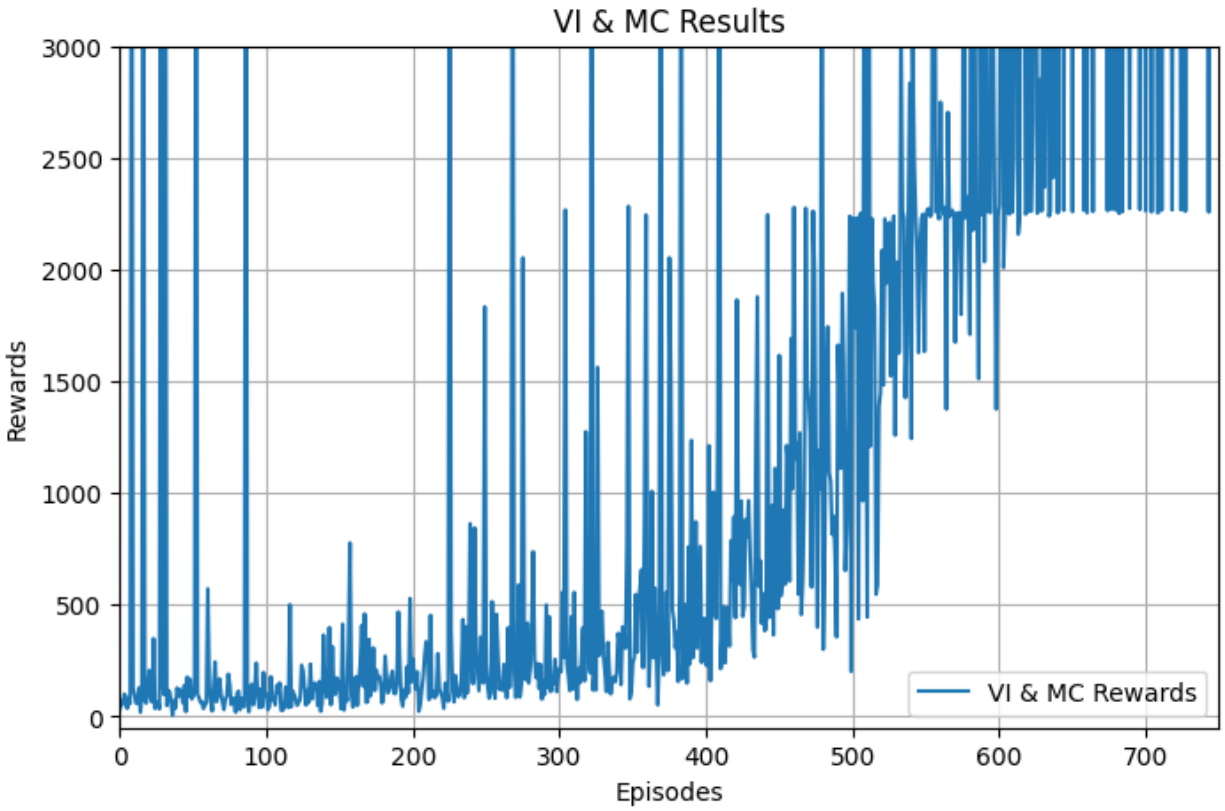
اپیزود: تعداد قسمت های موجود در حلقه آموزشی. این تعیین می‌کند که عامل روی چند تکرار از محیط آموزش ببیند.

epsilons: نرخ اکتشاف مورد استفاده در سیاست epsilon-greedy. احتمال انجام یک اقدام تصادفی به جای اقدام بهینه را بر اساس نتایج تکرار ارزش (بهینه_سیاست) تعیین می کند.

جدا از این ابرپارامترها، کد از ویژگی های world.observation_space.n و
world.action_space.n برای بدست آوردن تعداد حالت ها و اقدامات در محیط به ترتیب استفاده می کند. این ویژگی ها مختص محیط (دنیا), به صراحت در کد تنظیم نشده اند. بنابراین، کد به اجرای محیط برای بدست آوردن این مقادیر بستگی دارد.

۳.۶-


VI Results
VI Rewards
Rewards
Episodes
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
600
700



همانطور که از تصاویر فوق مشاهده میشود هنگامی که محیط توسط الگوریتم MC اموزش میبیند و سپس الگوریتم VIرا اجرا میکند به میزان پاداش بالاتری در تعداد قسمت کمتر می رسد. این در صورتی است که بدون استفاد از MC پس از گذشت ۷۰۰ قسمت هنوز به پاداش ۳۰۰۰ نرسیدیم.

۴- با توجه به نمودار های روش های مختلف که شامل Q Learning, SARSA, Backup n step, Off policy MC, VI, VI & MC هستند مشاهده میکنیم در بین ان ها روش VI&MC نه تنها سرعت یادگسری بالاتری داشته بلکه به پاداش بیشتری هم دست پیدا کرده است. البته باید این را در نظر داشته باشم که این روش دارای نوسانات بیشتری نسبت به سایر روش هاست. بعد از این روش, روش n step با n=5 دارای سرعت و پاداش بیشتر در یادگیری است. روش بعدی از لحاظ سرعت روش SARSA میباشد. روش های دسگر نیز از نظر سرعت و پایداری تا حدودی مشابع با یکدیگر هستند. البته روش(Off policy MC(fixed epsilon عملکرد مطلوبی ندارد چرا که با طی شدن ۷۰۰ قسمت هیچ روند افزایشی مشهودی در آن مشاهده نمیشود و همچنین از لحاظ پاداش دریافتی هم مقدار کمی دارد.

۵- در یک محیط تصادفی، همگرایی و حسرت الگوریتم های RL ممکن است به روش های زیر تحت تأثیر قرار گیرد:

- همگرایی: الگوریتم های RL با تخمین ارزش یا توابع سیاست بر اساس تجربه یاد می گیرند. در یک محیط تصادفی با تصادفی بودن باد و پاداش‌های تصادفی، عدم قطعیت در محیط ممکن است منجر به همگرایی با نرخ کندتر شود زیرا الگوریتم‌ها باید اقدامات مختلف و نتایج آنها را قبل از انتخاب بهینه‌ترین آنها بررسی کنند. این مبادله اکتشاف و بهره برداری می تواند در یک محیط تصادفی چالش برانگیزتر باشد.
- حسرت: در یک محیط تصادفی، تصادفی بودن در پاداش‌ها و قرار دادن target/blockade می‌تواند منجر به حسرت بیشتر شود، زیرا ممکن است عامل به دلیل نوسانات غیرقابل پیش‌بینی در پاداش‌ها یا موقعیت‌های target/blockade، نتایجی کمتر از حد مطلوب را تجربه کند.
- مدل‌های پیاده‌سازی‌شده: الگوریتم‌های RL برای یک پهپاد شبیه‌سازی‌شده که بر فراز نقشه‌ای شبکه‌مانند با تصادفی باد و پاداش‌های تصادفی پرواز می‌کند، باید ماهیت تصادفی محیط را در نظر بگیرد. مدل‌های پیاده‌سازی شده باید عدم قطعیت در جهت باد، پاداش‌های تصادفی و همچنین امکان موقعیت‌های متغیر برای اهداف و محاصره را در نظر بگیرند. این می‌تواند شامل ترکیب مدل‌های احتمالی مناسب یا روش‌های مونت کارلو برای تخمین نتایج و هدایت فرآیند تصمیم‌گیری الگوریتم‌های RL باشد.

**در حل این سوال از مدل زبانی Chat GPT استفاده شده است.

** prompt : Reply with concise answers and do not explain definitions unless asked other wise, Focus on the question asked and show your reasoning to validate your claim. Now answer. How does a stochastic environment effect RL algorithms' convergence and regret and the implemented models? The environment is a simulated model of a drone flying over a grid like map with 4 directions, it has wind randomness and also randomness in rewards, place of target, place of origin, and place of blockades.

چیز هایی که میشود اضافه کرد:
اطلاعاتی مثل مکان blockade ها (که البته state های زیادی داره) یا به جای ان فاصله به نزدیک ترین blockade. مکان base (شروع) برای بهبود باطری و سلامتی (یا باز هم فاصله به ان) خود باطری و خود سلامت. فاصله به خود target هم میتوند بهبود بیابد.