دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

گزارش تمرین شماره 4

درس یادگیری تعاملی

پاییز 1401

امیرحسین بیرژندی

...

810198367

...

فهرست

چکیده

هدف این تمرین آشنایی با الگوریتمهایی برای حل مسئلهی MDP با فرض ناشناخته بودن محیط می‌باشد. از این روشها در ادبیات به عنوان روشهای بدون مدل(Free-Model) یاد میشود. در این تمرین دو سوال تحلیلی و یک مسئله‌ی پیاده‌سازی که شامل بخش‌های مختلف می‌شود در نظر گرفته شد که طی آن با الگوریتم‌های SARSA, Expected SARSA, Q-learning, Tree Backup n-step آشنا می‌شویم.

بخش 1 - تحلیلی

سوال 1

ابتدا الگوریتم های Sarsa و Expected Sarsa را بررسی می‌کنیم تا بتوانیم به تفاوت های آن‌ها پی ببریم.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

رابطه 1 – نحوه آپدیت ارزش استیت-اکشن sarsa

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\ &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big], \end{aligned}$$

رابطه 2 – نحوه آپدیت ارزش استیت-اکشن در expected sarsa

1.1 و 1.2 در یافتن سیاست e-optimal چه تفاوتی دارند؟ در یافتن سیاست بهینه چه تفاوتی دارند؟

به طور کلی چه در یافتن سیاست e-optimal و چه در یافتن سیاست بهینه، expected sarsa با توجه به رابطه 2 که امید ریاضی نسبت به همه اکشن های موجود در استیت بعدی می‌گیرد sample efficiency بسیار بالاتری دارد و می‌تواند سرعت همگرایی(از نظر اپیزود) بیشتری نسبت به sarsa داشته باشد. زیرا sarsa صرفا با استفاده از behavior policy اکشن بعدی را انتخاب می‌کند و طبیعتا با امید ریاضی گرفتن بهتر ارزش ها آپدیت می‌شوند.

نکته دیگر این است که sarsa از نظر ساختار پیچیدگی کمتری دارد در نتیجه با لود کمتر و زمان کمتر می‌تواند همگرا شود.

پس باید بین جفت سرعت همگرایی و دقت و جفت پیچیدگی و زمان یکی را با توجه به مسئله انتخاب کنیم.

1.3 سرعت کاهش اپسیلون و نرخ یادگیری؟

از آنجایی که گفته شد expected sarsa امید ریاضی می‌گیرد و خطای کمتری دارد در نتیجه به بروزرسانی های به دست آمده از آن می‌توان بیشتر اتکا کرد و در نتیجه چه اپسیلون و چه نرخ یادگیری را می‌توان سریع‌تر کاهش داد. اما در sarsa باتوجه به خطای بیشتر این روش باید الگوریتم فرصت exploration بیشتری داشته باشد و اپسیلون را کندتر نسبت به expected sarsa آپدیت کند. نرخ یادگیری نیز همین است زیرا sarsa دیر تر می‌تواند به نتایج بدست آمده اتکا کند و مقادیر را بروزرسانی کند.

سوال 2

n-step return:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{V}_{t+n-1}(S_{t+n})$$

حال به فرض اگر ارزش π را دانسته باشیم به صورت زیر خواهد بود:

$$V_\pi(s) = E_\pi[G^{\pi}_{t:t+n} \mid S_t = s]$$

حال فقط را به صورت زیر تعریف می‌کنیم:

$$\max_s |\hat{V}(s) - V_\pi(s)| = \max_s \left| E_\pi[G_{t:t+n} \mid S_t = s] - E_\pi[G^{\pi}_{t:t+n} \mid S_t = s] \right|$$

مقداری که تخمین زده می‌شود ↓ — مقدار بیشترین واقعی ↓

$$= \max_s \left| E_\pi[G_{t:t+n} - G^{\pi}_{t:t+n} \mid S_t = s] \right|$$

$$= \max_s \left| E_\pi[R_{t:t+n} + \gamma^n \hat{V}_{t+n-1}(S_{t+n}) - R_{t:t+n} - \gamma^n V^{\pi}_{t+n-1}(S_{t+n})] \right|$$

$$= \max_s \left| E_\pi[\gamma^n \hat{V}(S_{t+n}) - \gamma^n V_\pi(S_{t+n})] \right|$$

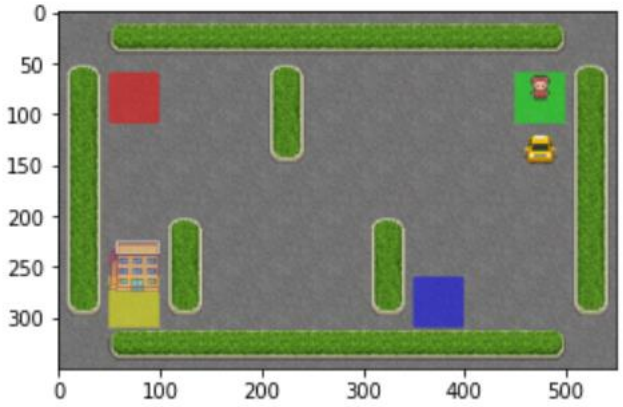
$$= \max_s \left| \gamma^n (\hat{V}(S_{t+n}) - V_\pi(S_{t+n})) \right|$$

$$= \gamma^n \max_s \left| \hat{V}(S_{t+n}) - V_\pi(S_{t+n}) \right|$$

بخش 2 - پیاده‌سازی

هدف سوال

در این بخش قرار است در محیط taxi-v3 با استفاده از الگوریتم‌های متفاوت یادگیری تعاملی به علی کمک کنیم تا یک تاکسی اینترنتی راه‌اندازی کند.



تصویر بالا پس از ریست کردن محیط با seed = 367 با رندر کردن محیط بدست آمده است. قرار است با استفاده از الگوریتم‌های یادگیری تعاملی این تاکسی مسافر را در کمترین زمان سوار کند و به مقصد برساند.

نکته مهم: با توجه به مکان اولیه تاکسی و مسافر توقع داریم پس از یادگیری با 10 حرکت تاکسی مسافر را سوار و پیاده کند. در نتیجه با توجه به مقادیر پاداش محیط توقع داریم الگوریتم ها به مقدار 10 همگرا شوند.

نکته: توقع داریم با توجه به حالت اولیه تاکسی با انجام اکشن های زیر به هدف خود دست یابد.

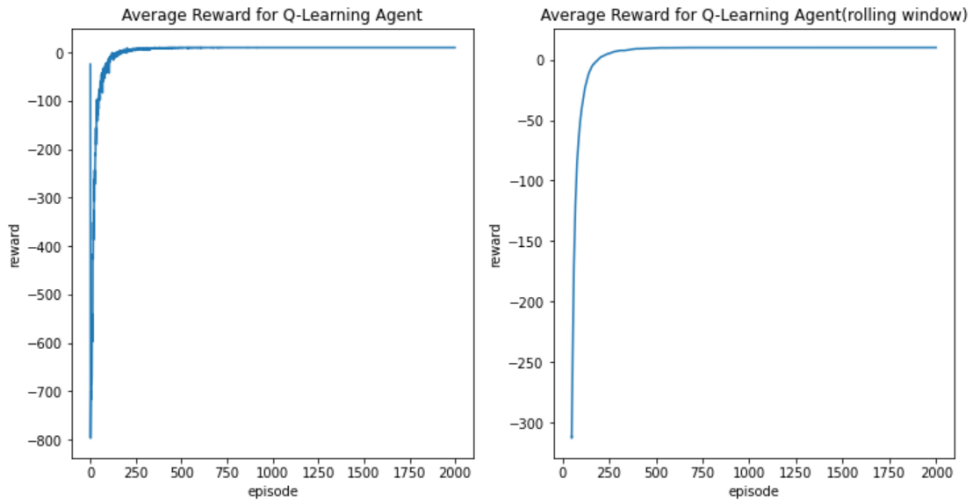
```
actions to reach the goal:
[1, 4, 0, 3, 3, 0, 3, 3, 0, 0]
```

سوال 1

در این سوال الگوریتم Q-Learning را برای آلفا ثابت و آلفا کاهشی پیاده می‌کنیم.

Q-Learning with fixed learning rate

ابتدا الگوریتم Q-Learning را برای $\varepsilon = 0.1$ اجرا می‌کنیم.



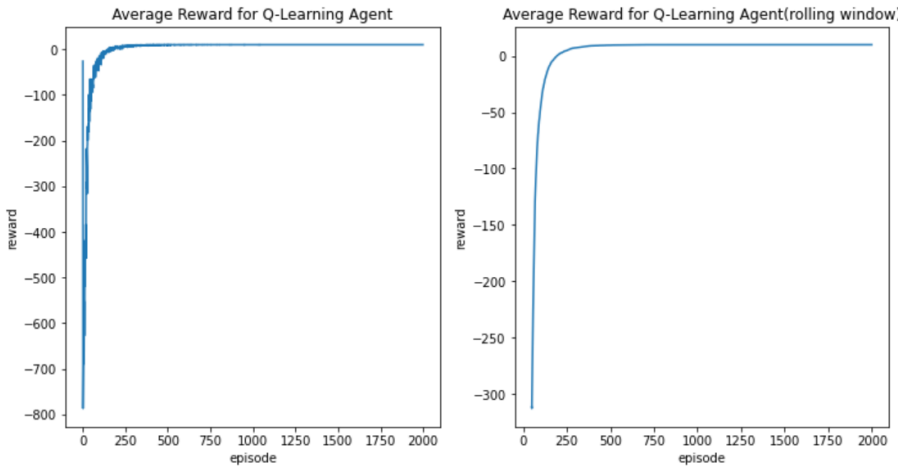
نمودار 1 – میانگین مجموع پاداش هر اپیزود

```
Q-Learning(Learning rate = 0.1) Average reward in last 100 episodes: 10.0

actions to reach the goal:
[1, 4, 0, 0, 3, 3, 3, 3, 0, 0]
```

همانطور که در نمودارهای بالا مشاهده می‌کنید ا میانگین پاداش در 100 اپیزود آخر 10 گزارش شده است که با انتظارات ما تطابق دارد.

Q-Learning with decaying learning rate



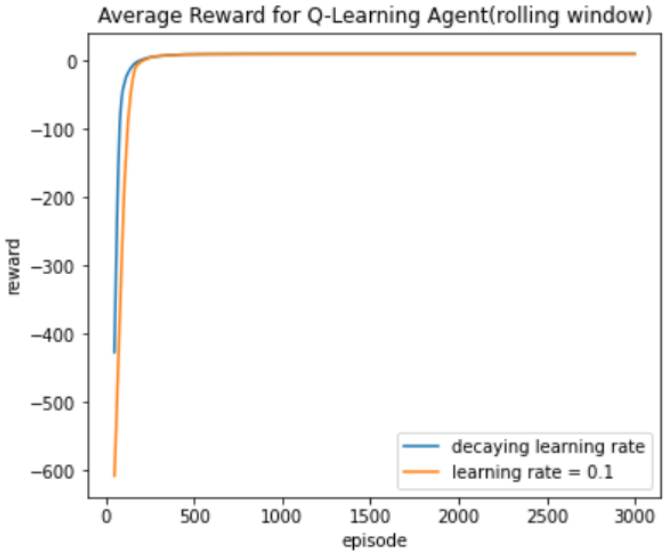
نمودار 2– میانگین مجموع پاداش هر اپیزود

```
Decaying learning rate Q-Learning Average reward in last 100 episodes: 10.0
```

```
actions to reach the goal:
[1, 4, 0, 3, 3, 0, 3, 3, 0, 0]
```

همانطور که در نمودار 2 مشاهده می‌کنیم همگرایی خیلی خوبی داشته‌ایم و میانگین پاداش در 100 اپیزود آخر برابر 10 شده است. این مقدار با مقدار مورد انتظار محاسبه شده برای همگرایی برابر است.

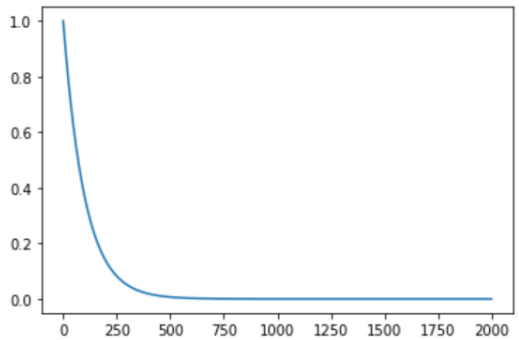
مقایسه دو راهکار



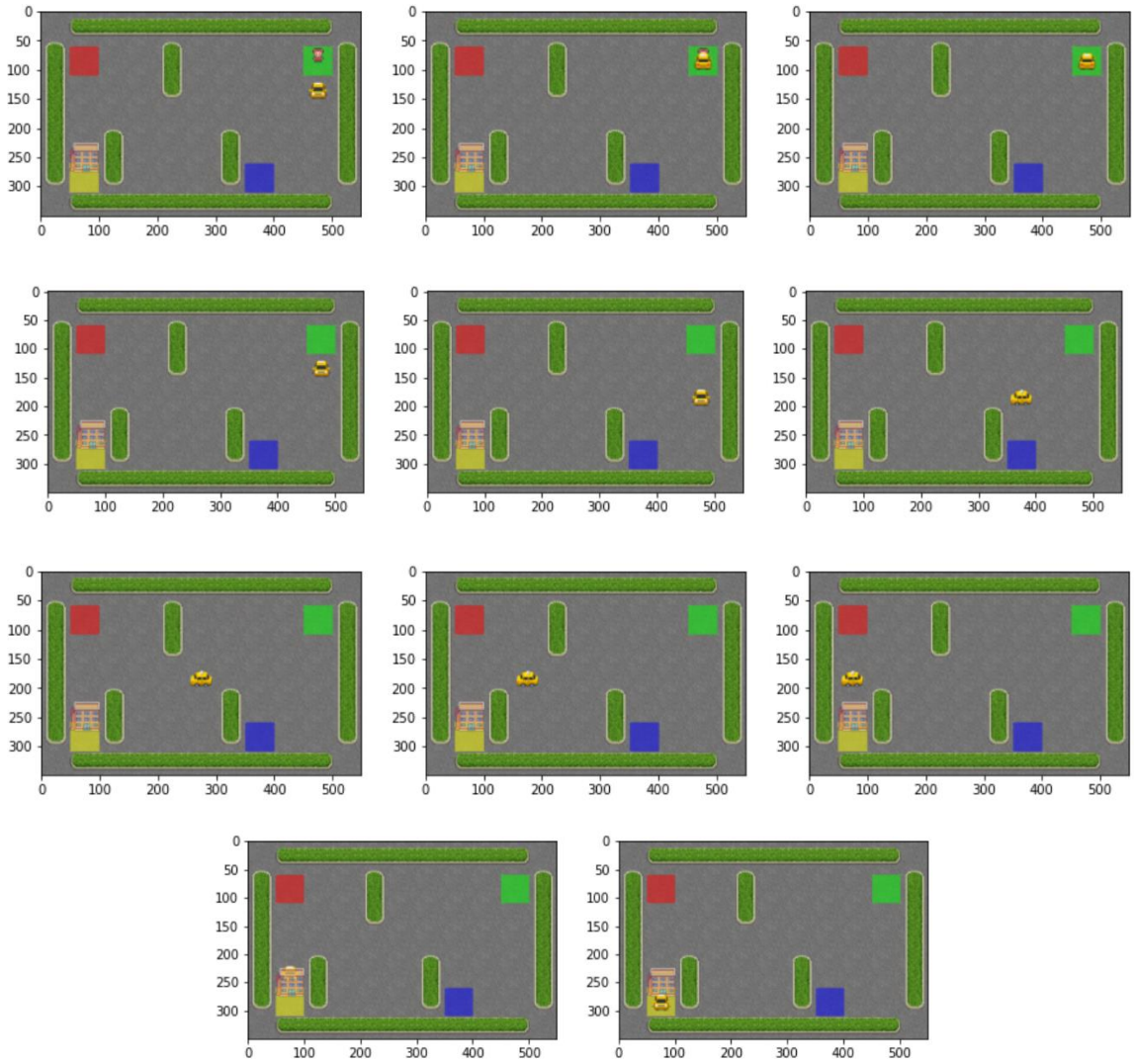
همانطور که در تصویر بالا مشاهده می‌کنیم در روش آلفا کاهشی سریع‌تر همگرا شده‌ایم. هر دو به مقدار 10 همگرا شده و Q value ها را به گونه‌ای بروزرسانی کرده‌اند که تاکسی مسیر بهینه را برای رسیدن به هدف طی می‌کند. مشخصا استفاده از این تکنیک می‌تواند همانطور که در مسائل عمومی یادگیری ماشین باعث افزایش سرعت می‌شود اینجا نیز بهتر می‌شود.

نحوه پیاده‌سازی اپسیلون کاهشی

هدف پیاده‌سازی اپسیلون کاهشی در واقع برقراری Exploration-Exploitation-Balance می‌باشد؛ به همین دلیل برای پیاده‌سازی ابتدا با اپسیلون برابر 1 که به معنی Full Exploration است شروع کرده و به صورت نمایی به مرور زمان اپسیلون را کاهش می‌دهیم تا به Full Exploitation برسیم. برای اینکار از رابطه $\varepsilon = \exp\{-0.01 * episode\}$ استفاده می‌کنیم. در شکل زیر نحوه کاهش اپسیلون را مشاهده می‌کنیم.



مشاهده مسیریابی پس از یادگیری



سوال 2

همانطور که در محیط taxi-v3 مشاهده می‌کنیم 25 خانه می‌توانند محل اولیه تاکسی، 5 خانه می‌توانند محل اولیه مسافر و 4 خانه می‌توانند محل مقصد مسافر باشد. در نتیجه 500 استیت قابل دسترس داریم.

اما اگر کمی دقت کنیم حالت هایی هستند که آن ها قابل دسترسی نیستند؛ زمانی که در حالت اولیه مکان مسافر و مقصد یکسان باشد که 4 حالت است و 25 مکان تاکسی نیز وجود دارد در نتیجه 100 حالت قابل دسترسی نیستند. اما نباید فراموش کنیم که ممکن است در پایان یک قسمت باشیم و تاکسی مسافر را در مقصد گذاشته باشد که این شامل 4 حالت می‌شود. در نتیجه کل امکان حضور در 404 استیت را داریم.

نکته قابل توجه این است که پس از یک بار reset کردن محیط که مکان اولیه مسافر و مقصد مشخص می‌شوند، خودکار به چندین استیت نمی‌توانیم دسترسی داشته باشیم. اما در حالت کلی امکان حضور در 404 استیت را داریم.

باید توجه کنیم که نیازی به حل این مشکل نیست زیرا زمانی که در این استیت ها نمی‌توانیم حضور داشته باشیم و تاثیری نیز از حضور آن‌ها نمی بینیم نیازی به حذف آن‌ها نیست. جدول Q values نیز در این استیت ها همواره صفر خواهد بود.

برای بدست آوردن شماره این خانه هایی که قابل دسترس نیستند می توانیم از متد encode استفاده کنیم. که با دادن موقعیت طول و عرض تاکسی به عنوان دو ورودی اول و لوکیشن مسافر و مقصد به عنوان ورودی سوم و چهارم شماره این استیت را بدست آوریم. برای مثال برای بدست آوردن حالت های غیر قابل می‌توانیم محل مسافر و مقصد را یکی کنیم و استیت ها متناظر آن ها را بیابیم.

سوال 3

در این سوال الگوریتم های Sarsa و n step tree backup را پیاده‌سازی می‌کنیم.

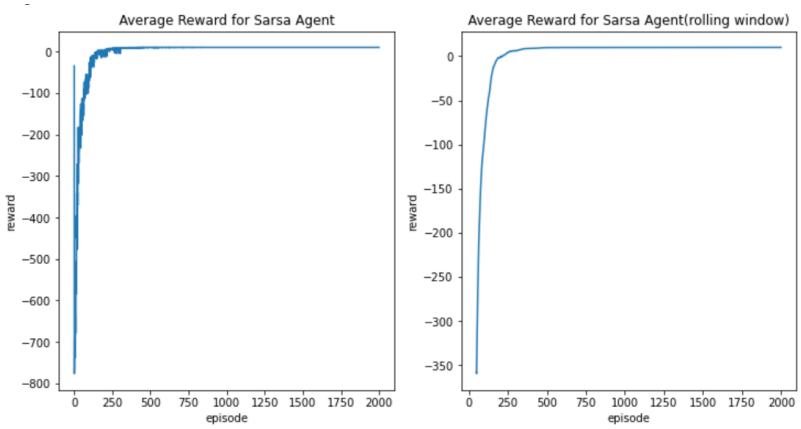
Sarsa

ابتدا به نحوه پیاده‌سازی الگوریتم Sarsa می‌پردازیم. این الگوریتم را با توجه به شبه کد زیر [1] پیاده‌سازی می‌کنیم.

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal




```
Sarsa Average reward in last 100 episodes: 10.0

actions to reach the goal:
[1, 4, 0, 3, 3, 0, 3, 3, 0, 0]
```

همانطور که در نمودار مشاهده می‌کنیم همگرایی خیلی خوبی داشته‌ایم و میانگین پاداش در 100 اپیزود آخر برابر 10 شده است. این مقدار با مقدار مورد انتظار محاسبه شده برای همگرایی برابر است. همچنین با مسیر بهینه این تاکسی مسافر را پیاده می‌کند.

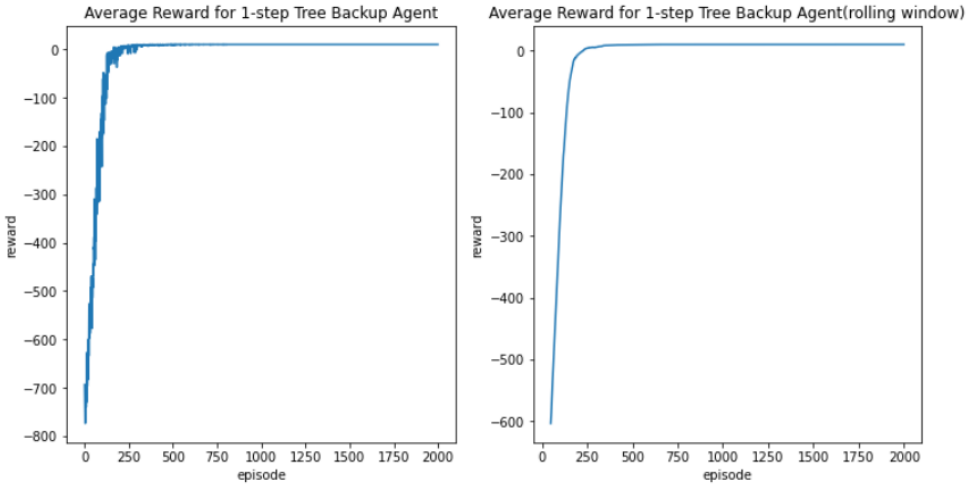
1-Step Tree Backup

ابتدا به نحوه پیاده‌سازی الگوریتم n-Step Tree Backup می‌پردازیم. این الگوریتم را با توجه به شبه کد زیر [1] پیاده‌سازی می‌کنیم.

n-step Tree Backup for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations can take their index mod $n+1$

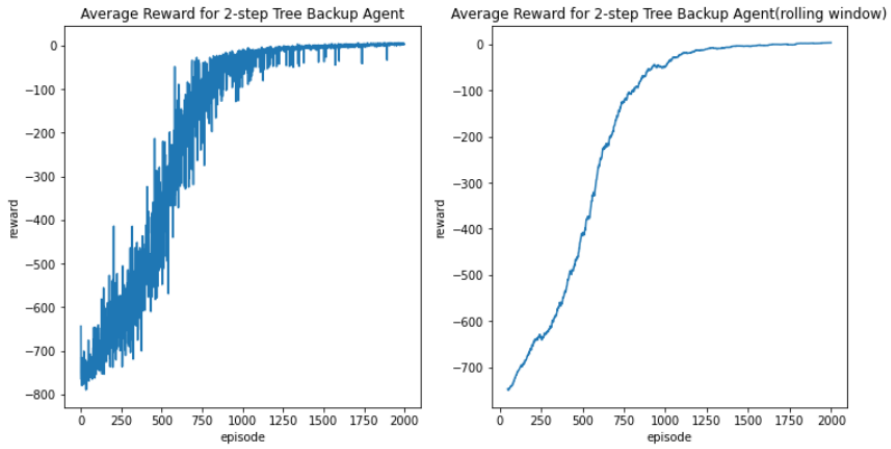
Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Choose an action A_0 arbitrarily as a function of S_0; Store A_0
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$:
  | Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
  | If S_{t+1} is terminal:
  | $T \leftarrow t+1$
  | else:
  | Choose an action A_{t+1} arbitrarily as a function of S_{t+1}; Store A_{t+1}
  | $\tau \leftarrow t+1-n$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | If $t+1 \geq T$:
  | $G \leftarrow R_T$
  | else
  | $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
  | Loop for $k = \min(t, T-1)$ down through $\tau + 1$:
  | $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T - 1$




```
1-Step Tree Backup Average reward in last 100 episodes: 10.0

actions to reach the goal:
[1, 4, 0, 3, 3, 0, 3, 3, 0, 0]
```

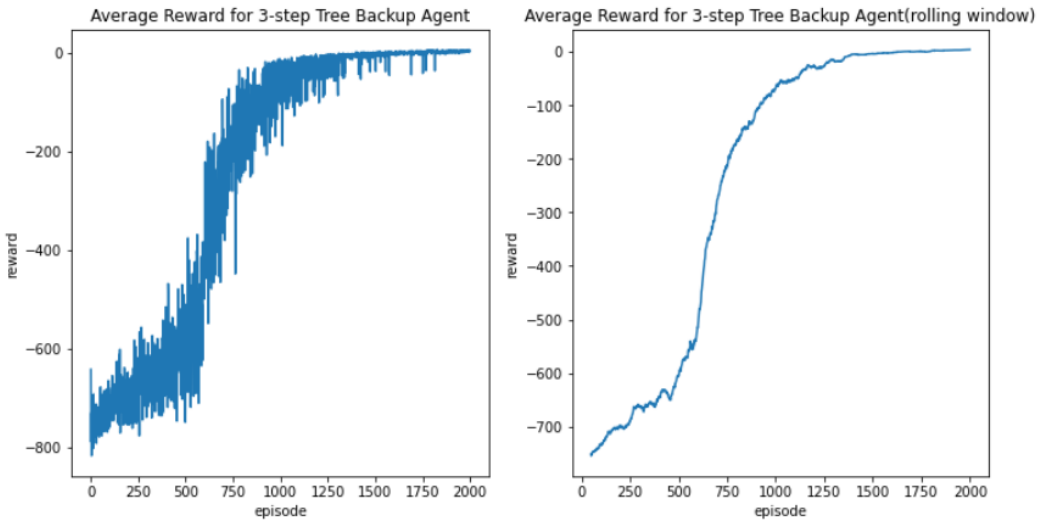
همانطور که در نمودار مشاهده می‌کنیم همگرایی خیلی خوبی داشته‌ایم و میانگین پاداش در 100 اپیزود آخر برابر 10 شده است. این مقدار با مقدار مورد انتظار محاسبه شده برای همگرایی برابر است. همچنین با مسیر بهینه این تاکسی مسافر را پیاده می‌کند.




```
2-Step Tree Backup Average reward in last 100 episodes: 3.146
```

```
actions to reach the goal:
[1, 4, 0, 3, 3, 0, 3, 3, 0, 0]
```

همانطور که مشاهده می‌کنیم در این الگوریتم به مقدار مورد انتظار 10 همگرا نشده‌ایم. اما باید دقت کنیم که این مقدار همگرا شده دلیلی بر اشتباه عمل کردن الگوریتم نیست و صرفا به دلیل دو گامی است که با سیاست epsilon-greedy انجام می‌دهیم. در واقع هر چه با سیاست نرم (epsilon-greedy) بیشتر زندگی کنیم و هر چه در محیط کاوش بیشتری انجام دهیم امکان اینکه پاداش منفی بگیریم را بیشتر کرده‌ایم اما دلیل بر اشتباه بهینه شدن نیست به عبارتی این الگوریتم Q values را به گونه‌ای بروزرسانی کرده است که در بهینه ترین مسیر مسافر را پیاده می‌کند.

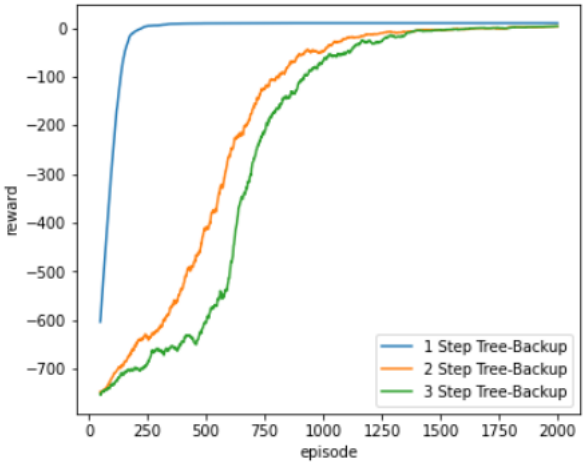



```
3-Step Tree Backup Average reward in last 100 episodes: 3.119
```

```
actions to reach the goal:
[1, 4, 3, 0, 3, 0, 3, 3, 0, 0]
```

همانطور که مشاهده می‌کنیم در این الگوریتم به مقدار مورد انتظار 10 همگرا نشده‌ایم. اما باید دقت کنیم که این مقدار همگرا شده دلیلی بر اشتباه عمل کردن الگوریتم نیست و صرفا به دلیل سه گامی است که با سیاست epsilon-greedy انجام می‌دهیم. به عبارتی این الگوریتم Q values را به گونه‌ای بروزرسانی کرده است که در بهینه ترین مسیر مسافر را پیاده می‌کند. در اینجا نیز همانند بالا به مقدار پایین‌تری همگرا شده‌ایم زیرا 3 مرحله با سیاست نرم کاوش می‌کنیم.

Comparing different n's in n step tree backup



همانطور که مشاهده می‌کنیم هر سه الگوریتم به مقدار خوبی همگرا شده اند اما سرعت 1-step از دو حالت دیگر بیشتر است. که طبیعی است زیرا در دو گام دیگر وقت بیشتر برای گشتن محیط می‌گذاریم اما در عوض اطلاعات بیشتری راجب محیط خواهیم داشت و در حالتی که seed نداشتیم اطلاعات بیشتری نسبت به محیط کسب می‌کردیم. نکته دیگر این است که دلیل اینکه در 2-step و 3-step به مقدار کوچکتر از 10 همگرا شدیم این است که این دو الگوریتم نیاز به اپیزود بیشتری دارند اما این دلیلی بر انتخاب حرکت غیر صحیح نیست و Q values به درستی آپدیت شده‌اند.

نکته دیگر این است که با توجه به نحوه کاهش اپسیلون با 2000 اپیزود به اپسیلون نزدیک 0.13 می رسیم که یعنی همچنان با احتمال 0.1 دو گام یا سه گام غیر ماکسیمم را طی می‌کنیم در نتیجه ممکن است ریوارد کمتر از 10 در طول مسیر بگیریم.

سوال 4

Off-policy Monte Carlo

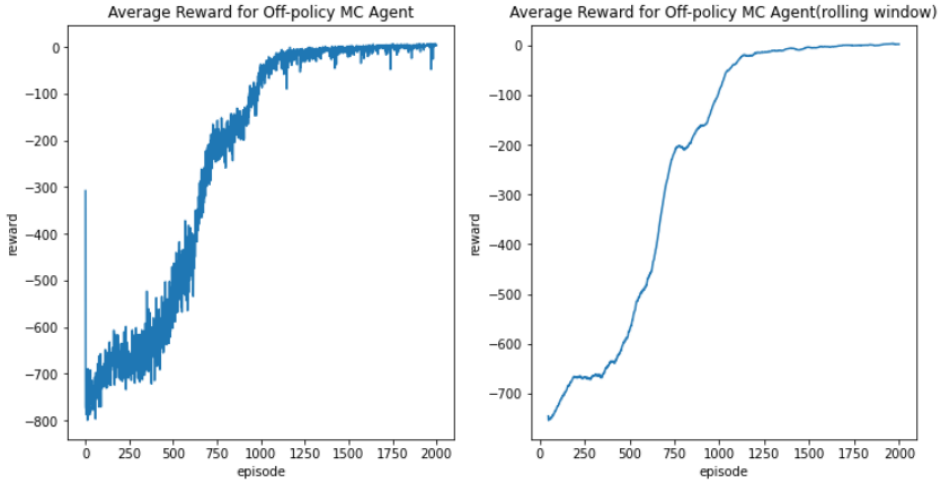
ابتدا به نحوه پیاده‌سازی الگوریتم off-policy Monte Carlo می‌پردازیم. این الگوریتم را با توجه به شبه کد زیر [1] پیاده‌سازی می‌کنیم.

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Off-Policy MC with decaying epsilon



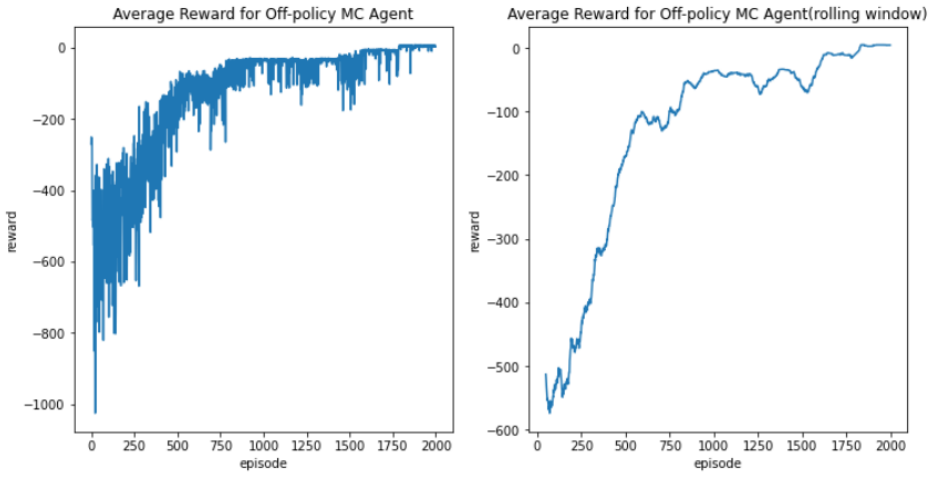

```
Decaying epsilon Off-policy MC Average reward in last 100 episodes: 2.171
```

```
actions to reach the goal:
[1, 4, 3, 0, 3, 0, 3, 3, 0, 0]
```

در نمودار بالا مشاهده می‌کنیم که این روش کم کم در حال همگرا شدن است و میانگین پاداش در 100 اپیزود آخر برابر با 2.171 است. اما نکته مهم تر این است که این روش نیز مانند سایر روش ها توانست Q value ها را به گونه‌ای بروزرسانی کند که تاکسی با مسیر بهینه مسافر را سوار و پیاده می‌کند. دلیل اینکه سرعت همگرایی پایین تر است به ماهیت خود الگوریتم Monte Carlo بر می‌گردد که برای اپدیت کردن ارزش ها باید یک اپیزود را کامل و گام به گام طی کند و سپس بروزرسانی کند در نتیجه باید بیشتر در محیط بگردیم و در 2000 اپیزود همچنان به مقدار کمتری همگرا می‌شویم. نکته

دیگر این است که با توجه به نحوه کاهش اپسیلون با 2000 اپیزود به اپسیلون نزدیک 0.13 می رسیم که یعنی همچنان با احتمال 0.1 چندین گام غیر ماکسیمم را طی می‌کنیم در نتیجه ممکن است ریوارد کمتر از 10 در طول مسیر بگیریم.

Off-Policy MC with fixed epsilon

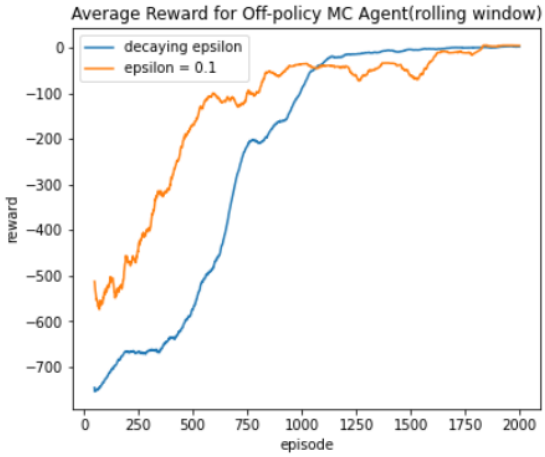



```
fixed epsilon Off-policy MC Average reward in last 100 episodes: 4.411
```

```
actions to reach the goal:
[1, 4, 3, 0, 3, 0, 3, 3, 0, 0]
```

در این بخش نیز در حال همگرا شدن هستیم و در 100 اپیزود آخر به مقدار 4.411 همگرا شده‌ایم. اما نکته مهم تر این است که این روش نیز مانند سایر روش ها توانست Q value ها را به گونه‌ای بروزرسانی کند که تاکسی با مسیر بهینه مسافر را سوار و پیاده می‌کند.

مقایسه دو راهکار



شاید ابتدا فکر کنیم که با اپسیلون کاهشی بهتر همگرا شویم اما باید دقت کنیم در monte carlo نیاز به زمان زیادی برای گشتن داریم و اگر از اپسیلون 1 شروع کنیم و به صورت کاهشی تا 0.13 بیایم خیلی ممکن است در این 2000 اپیزود

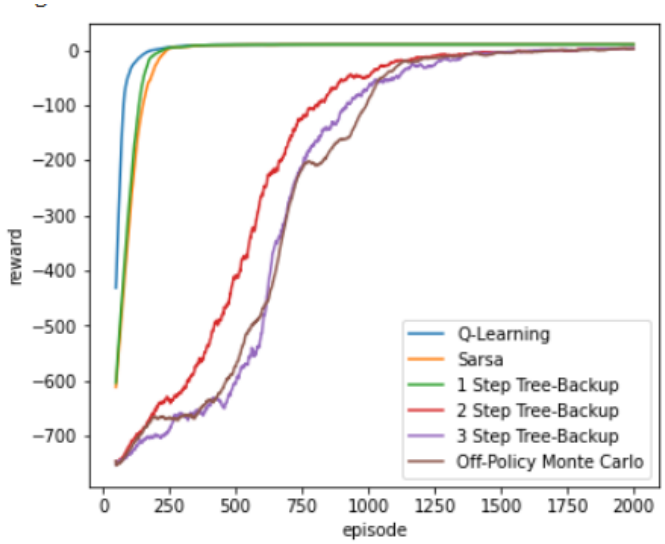
تصمیمات اشتباه بگیریم و با اینکه با اپسیلون 0.1 اکسپلوریشن کمتری داریم اما دیگر امکان کار های اشتباه کمتر می‌شود و سریع‌تر می‌توانیم همگرا بشویم. راجب مقادیر همگرا شده نیز همه این موضوعات مطرح شده برقرار است.

سوال 5

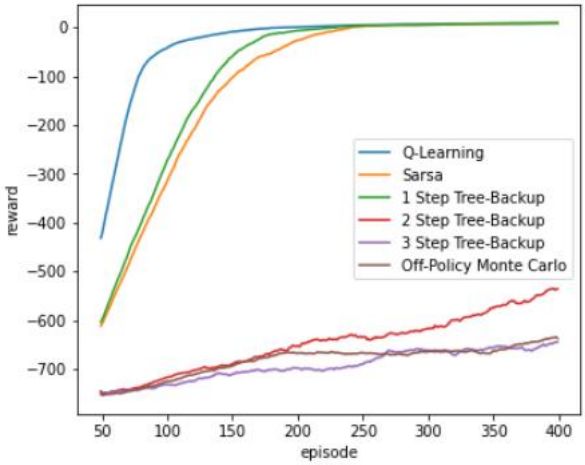
در الگوریتم Monte carlo سرعت یادگیری کمتر است و در واقع می‌گوییم این الگوریتم sample efficiency پایینی دارد. دلایل زیر را می‌توان برای این موضوع مطرح کرد:

1- سایر الگوریتم ها در هر گام یا نهایتا 3 گام با استفاده از ارزش استیت های کناری خود را آپدیت می‌کنند. اما MC یک اپیزود را کامل طی می‌کند و سپس به صورت معکوس ارزش را گام به گام آپدیت می‌کند.

2- حال دلیل دیگر که باعث کندی این روش که مخصوص حالت MC off policy است امکان صفر شدن ρ می‌باشد. یعنی امکان دارد ما در behavior policy با اکشنی زندگی‌کنیم که احتمال آن در evaluation policy صقر باشد در نتیجه اون اپیزود تا مقطعی دیگر قابل بروزرسانی نخواهد بود.



در تصویر بالا نتایج روش های مختلف را مشاهده می‌کنیم.



در تصویر زیر نیز این متد ها را در 400 اپیزود اول رسم کرده‌ایم تا اگر بنا به کم کردن پشیمانی داشته باشیم و budget محدودی نیز داشتیم از چه روشی استفاده کنیم.

منابع

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.