بسم الله الرحمن الرحیم

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

فاز دوم پروژه در مبانی هوش و کاربردها

فرآیند تصمیم مارکوف و محیط ناشناخته

استاد درس: دکتر حسین کارشناس

دستیار استاد: پوریا صامتی

دانیال شفیعی
مهدی مهدیه
سید امیررضا نجفی

آذر ۱۴۰۳

فهرست مطالب

فصل ۱

فرآیند تصمیم مارکوف

۱.۱ مقدمه

برای بخش MDP [۱] در ابتدا تصمیم گرفتیم v_table استخراج کنیم که حاوی راه رسیدن به یک خوک هستند. بعد از خوردن هر خوک به v_table رفته و طبق سیاست‌هایی که از آن استنتاج کردیم پیش می‌رویم. این روش مشکلاتی را در پی داشت؛ مثل اینکه ممکن بود یک خوک توسط دیوارها محاصره شده باشد و یا اینکه ممکن بود در مسیر خوردن خوک مورد نظر، بعضی از خوک‌های دیگر نیز به دلیل تصادفی بودن حرکات خورده شوند و مشکلات دیگری که برای حل آنها نیاز به تعداد زیادی شرط بود. این شروط حس را می‌داد که دیگر مسئله را با فرآیند تصمیم مارکوف به صورت عمومی حل نمی‌کنیم و داریم با جزئی نگری زیادی به مسئله نگاه می‌کنیم. این جزئی نگری شاید در بعضی محیط‌ها به نفع ما باشد ولی ممکن است با تغییر محیط برایمان بسیار گران تمام شود.

۲.۱ خوشه‌بندی

ما تصمیم گرفتیم با استفاده از خوشه‌بندی، خوک‌های موجود در صفحه را به ۲، ۳ یا ۴ خوشه تقسیم می‌کنیم. سپس با مقایسه، خوشه‌ای که در آن خوک‌ها بهترین نماینده را درون آن خوشه دارند انتخاب کنیم. آنگاه برای هر خوشه یک v_table می‌سازیم که به خوک‌های موجود در آن خوشه امتیاز مثبت می‌دهیم ولی خوک‌هایی که در آن خوشه نیستند را نادیده می‌گیریم. سپس عامل در هر خوشه تا جای ممکن خوک‌های موجود را خورده و به سمت خوشه‌ی بعدی حرکت می‌کند و این روند همین‌گونه ادامه پیدا می‌کند تا زمانی که به خوشه‌ی آخر (تخم مرغ) که هدف نهایی است برسیم.

۱.۲.۱ انتخاب تعداد خوشه‌ها

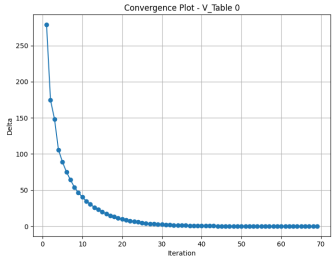
سوال کلیدی این است که الگوریتم چگونه متوجه می‌شود به چه ترتیبی v_table ها را پیمایش کند؟ یا به زبان ساده تر بهتر است ابتدا سیاستهای متعلق به اولین خوشه را اجرا کنیم یا دومین خوشه را؟ برای مقایسه، روی تمام مراکز خوشه‌ها جایگشت را حساب می‌کنیم و بهترین جایگشت را انتخاب می‌کنیم. در بیشترین حالت اگر ۴ خوشه انتخاب کرده باشیم، !۴ حالت دارد پس از نظر زمانی نیز لطمه‌ای به الگوریتم نمی‌زند.

---

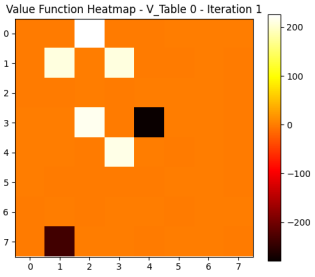
[۱] Markov Decision Process

۲.۲.۱ همگرایی خوشه‌بندی و تشکیل نقشه‌ی گرمایی

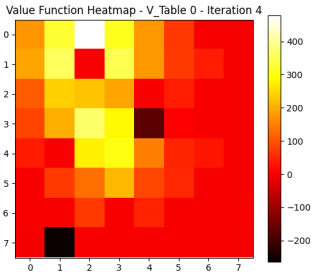
برای نمونه ما نقشه‌گرمایی محیط راه در خوشه‌بندی به ازای k برابر ۳ در چند تکرار مختلف نمایش می‌دهیم.



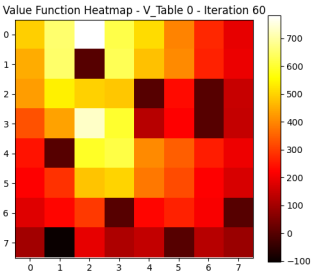
(آ) همگرایی خوشه‌بندی



(ب) تکرار اول



(ج) تکرار چهارم



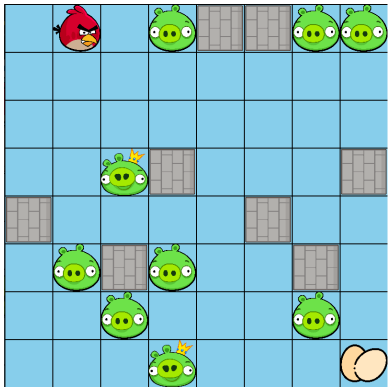
(د) تکرار شصتم

شکل ۱.۱: نمونه‌ای از همگرایی خوشه‌بندی به ازای k برابر ۳

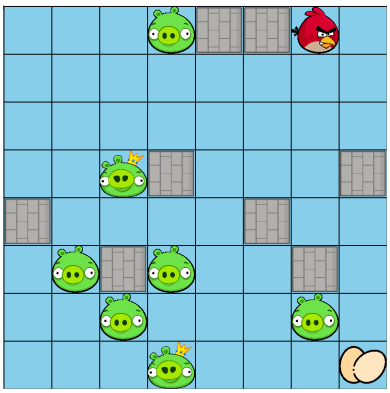
۳.۱ خروجی الگوریتم

در ادامه خروجی از تحلیل الگوریتم روی یک نمونه از محیط را می‌بینیم.

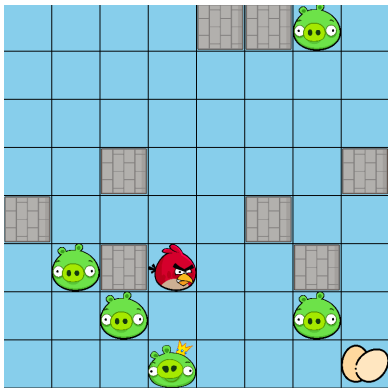
```
Best clustering with 4 clusters:
Cluster 1: [(5, 1), (5, 3), (6, 2)], Centroid: 5.33, 2.0
Cluster 2: [(0, 1), (0, 3)], Centroid: 0.0, 2.0
Cluster 3: [(6, 6)], Centroid: 6.0, 6.0
Cluster 4: [(0, 6), (0, 7)], Centroid: 0.0, 6.5
```



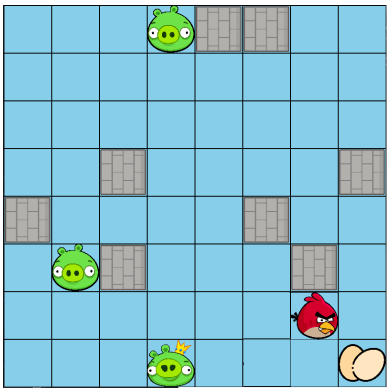
(آ) شروع



(ب) حرکت به سمت خوشه‌ی اول



(ج) حرکت به سمت خوشه‌ی دوم



(د) حرکت به سمت خوشه‌ی سوم (تخم‌مرغ)

شکل ۲.۱: روند حرکت پرنده‌ی خشمگین درون یک محیط تصادفی نمونه

فصل ۲

محیط ناشناخته

۱.۲ مقدمه

در این بخش از پروژه ما قصد داشتیم با تعداد محدودی تکرار و گردش [۱] در محیط ناشناخته [۲] به یک قاعده برسیم که در آن بتوانیم با استفاده از، به نحوی در این محیط تصادفی پرنده را هدایت کنیم که بیشترین امتیاز ممکن را کسب کند. نکات قابل توجه در این پروژه این است که نباید به TNT برخورد کنیم و تعداد حرکات هم محدود است و نمی‌توانیم بی‌نهایت در محیط بگردیم تا همه‌ی حالت‌ها را بدست بیاوریم.

۲.۲ شیوه‌ی پیاده‌سازی الگوریتم QLearning

در مرحله‌ی اول ما یک ماتریس ۸ در ۸ در ۴ داشتیم که امتیاز هر کنش در هر حالت را مشخص می‌کرد. حالت‌ها نمایانگر نقطه‌ی که پرنده در آن وجود داشت بودند.

۱.۲.۲ ابرپارامترهای آموزش

برای تغییرات مقادیر جدول Q یک سری ابر پارامتر وجود دارد. γ در کد ما تحت عنوان فاکتور تخفیف معرفی شده است و α تحت عنوان سرعت آموزش.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \qquad (۱.۲)$$

در هر اپیزود از اجرای الگوریتم، حالت به عنوان ورودی داده می‌شود و سپس بهترین کنش از جدول انتخاب می‌شود. سپس با انجام کنش به کمک تابع step می‌دهیم و خروجی آن را دریافت می‌کنیم و توابع و حالات جدید را بدست می‌آوریم.

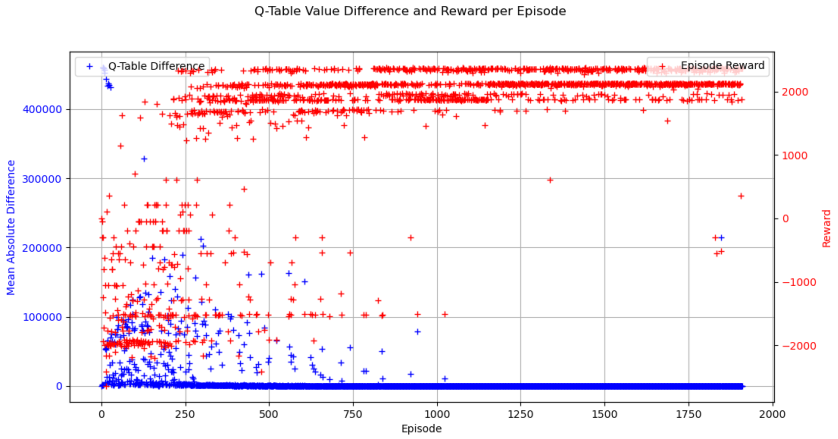
۲.۲.۲ همگرایی مقادیر

برای اینکه بعد از مدتی همگرا شود، ما از تکنیک Epsilon Greedy استفاده کردیم. در این راهبرد. در اوایل اجرا، پرنده تا جای ممکن تصادفی عمل می‌کند. اما به مرور زمان با توجه به نتایج به دست

---

[۱] Explore
[۲] Environment Unknown

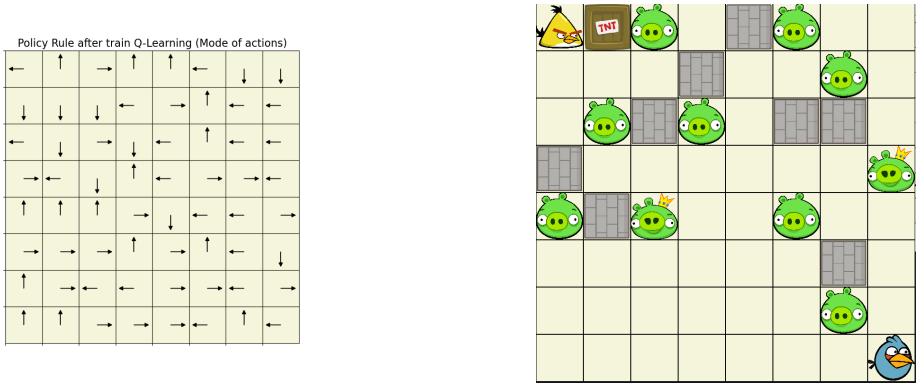
آمده بهترین کنش را انتخاب می‌کند. نرخ کاهش نیز بدین صورت است که تا زمانی که نتیجه در هر حال بهتر شدن است، همچنان اپسیلون و نرخ یادگیری تغییری نمی‌کند اما به محض کاهش پاداش کل برای اپسیلون گریدی و افزایش value_diff برای سرعت یادگیری، مقادیر کاهش پیدا می‌کنند.



شکل ۱.۲: همگرایی value_diff در یک محیط نمونه که پس از ۱۰ بار کوچک‌تر شدن اپسیلون از ۱۰ الگوریتم متوقف شده است.

۳.۲.۲ استخراج سیاست

در ابتدای کار وقتی شرایط زنده بودن خوک‌ها را در نظر نگرفته بودیم، می‌توانستیم یک سیاست واحد استخراج کنیم اما بعد از آن، ما از بین اکشن‌ها مُد می‌گیریم تا یک حس از اینکه در هر حالت به چه سمتی حرکت می‌کند مشخص شود. برای خانه‌هایی که مقادیرشان مشخص نیست، الگوریتم به صورت تصادفی سیاست استخراج می‌کند.



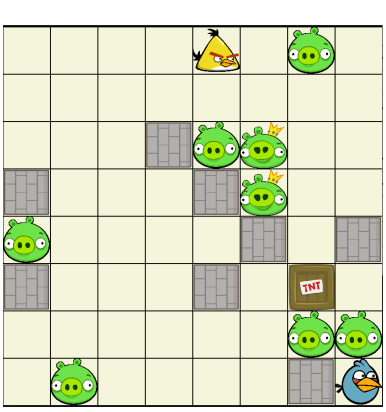
(آ) محیط نمونه (ب) مد کنش در حالت‌ها در محیط متناظر

شکل ۲.۲: استخراج سیاست به شیوه‌ی مد گرفتن از کنش‌ها در هر حالت

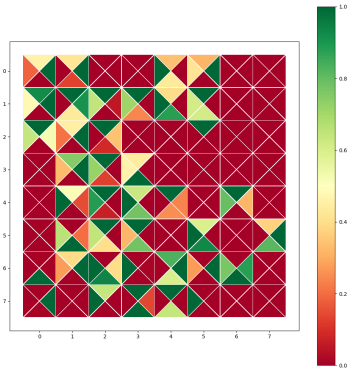
۴.۲.۲ سایر ویژگی‌ها

مهندسی پاداش وقتی الگوریتم به پایان می‌رسد، یک پاداش ۱۰۰۰- داریم که اگر این پاداش روی یک حالت و کنش برود، حتی اگر آن کنش خوب باشد، باعث می‌شود دیگر سمت این کنش نرویم! ما باید این اثر را حذف کنیم. همچنین چنین در نظر گرفته بودیم که اگر یک حالت و کنشی در یک اپیزود دفعات متکرر اجرا شد، به آن امتیاز منفی بدهیم که البته این باعث همگرایی نمی‌شود و ممکن است نتایج بدتری بدهد. همچنین برای اینکه از ملکه‌ها رد شویم و در دام آن‌ها نیفتیم، فرض کردیم اگر از آن‌ها بگذریم، امتیاز منفی کمتری دارد که البته این هم حالب نبود. در نهایت هم مقادیر جدول Q را به ازای همه‌ی کنش‌های آن حالت بروز می‌کردیم تا هیچ کنشی حتی کنش مقداردهی نشده هم به سمت TNT نرود.

مصورسازی برای نمایش اینکه در هر حالت چه کنشی بیشتر موردتوجه است و چه کنشی مضر است، از یک نقشه گرمایی استفاده کردیم که جهت‌های ممکن را به ما نشان می‌دهد. به این منظور کنش‌های هر حالت نرمالایز می‌شود و بر اسا مقادیر ۰ تا یک رنگ پیدا می‌کند. خانه‌هایی که همه‌ی آن‌ها قرمز هستند یعنی هیچ‌گاه مقداردهی نشده‌اند و خانه‌هایی که هر سه خانه سبز هستند یعنی یکی از خانه‌ها مقدار منفی دارد و احتمالا قدر مطلق آن بسیار زیاد است. واضح است خانه‌ای که TNT در آن موجود است از هر ۴ جهت قرمز شده است.



(آ) محیط نمونه



(ب) نمونه نقشه گرمایی در حالی که هیچ‌کدام از خوک‌ها خورده نشده‌اند

شکل ۳.۲: نقشه گرمایی جهت حرکت بر اساس جدول Q

اثرپذیری از حالت خوک‌ها به کمک این ویژگی متناسب با اینکه هر خوکی خورده شده است یا نه یک جدول سیاست کنش داریم. یعنی به‌جای ۱ جدول، ۲۵۶ جدول داریم که البته بسیاری از آن‌ها هرگز مقداردهی نمی‌شوند ولی همین‌هایی که پر می‌شوند به ما کمک می‌کنند تا امتیاز بسیار بالایی کسب کنیم و به‌خاطر یک خوک خورده شده گرفتار حلقه نشویم و محدودیت حرکتی نخوریم.

۳.۲ خروجی

نمونه‌ی خروجی این الگوریتم برای محیط نسبتا سختی که در شکل ۳.۲آ نمایش داده شده است به نحو زیر می‌باشد:

```
[False, False, False, False, False, False, False, False]
Episode finished with reward: 2347
[False, False, False, False, False, False, False, False]
Episode finished with reward: 2336
[False, False, False, False, False, False, False, False]
Episode finished with reward: 2285
[False, False, False, False, False, False, False, False]
Episode finished with reward: 2329
[False, False, False, False, False, False, False, False]
Episode finished with reward: 2303
MEAN REWARD: 2320.0
```

در هر ۵ حالت همه‌ی خوک‌ها خورده شده‌اند و امتیاز بالایی کسب شده است.

۴.۲ شبکه‌ی عصبی

۱.۴.۲ مقدمه

Q-Learning عمیق یا شبکه‌ی Q عمیق [۳] همان توسعه یافته‌ی الگوریتم Q-Learning است که در آن از شبکه‌های عصبی برای یادگیری عمیق و تقریب Q استفاده می‌شود. Q-Learning سنتی برای محیط‌هایی با تعداد حالت‌های کم و محدود به خوبی کار می‌کند، اما به دلیل اندازه جدول Q با فضاهای حالت بزرگ یا پیوسته مشکل دارد. Q-Learning عمیق با جایگزین کردن جدول Q با یک شبکه عصبی که می تواند مقادیر Q را برای هر جفت حالت–عمل تقریبی بزند و بر این محدودیت غلبه کند.

۲.۴.۲ پیاده‌سازی

ساختار شبکه **لایه اول:** ورودی با اندازه ۶۴ (که نشان‌دهنده‌ی وضعیت محیط ۸ در ۸ است) به ۱۲۸ نورون متصل می‌شود. **لایه دوم:** ۱۲۸ نورون به ۶۴ نورون متصل می‌شود. **لایه خروجی:** ۶۴ نورون به تعداد کنش‌های ممکن (بالا، پایین، چپ، راست) متصل می‌شود. **تابع فعال‌سازی:** از تابع ReLU برای افزودن غیرخطی بودن به شبکه استفاده می‌شود.

اپیزودها ما یک بهنیه‌ساز و تابع خطا تعریف می‌شود. همچنین وضعیت محیط به بردار ۸ در ۸ تبدیل می‌شود. که این بردار ورودی شبکه عصبی است. هر سلول بر اساس پاداش خود آن مقدار خاصی به بردار اضافه می‌کند. سپس یک کنش انتخاب می‌شود و یک قدم برداشته می‌شود. بعد با استفاده از راهبر اپسیلون گریدی یک اقدام تصادفی یا اقدامی با بیشترین مقدار انتخاب می‌شود. سپس مدل بر اساس پاداش تجربیات خود را ذخیره می‌کند و وزن‌های شبکه‌ی عصبی بروزرسانی می‌شود.
سیاست نهایی با انتخاب اقدام با بیشترین مقدار Q برای هر وضعیت ممکن استخراج می‌شود.

[۳] Deep Q Network