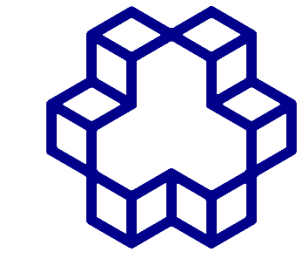
Google drive: https://drive.google.com/drive/folders/1n8uWALN39KWm7Zrq6tpUlCi1efUUQn4D?usp=sharing

Github: https://github.com/AliBagheriNejad/ML-AliYari



گزارش نیمچه پروژه ۴ درس یادگیری ماشین

علی باقری نژاد

۴۰۲۰۲۸۵۴

فهرست

سوال ۱

الف: ساخت و آموزش مدل ها

کد شامل ۳ بخش اصلی می باشد:

۱. تعریف wumpusworld

۲. ایجنت Q-Learning

۳. ایجنت Deep Q-Network

Wunpus World

در یک کلاس به اسم WumpusWorld، محیط جهان مورد نظر بوجود آمد. این کلاس مشخصاتی مثل مکان چاله ها یا نقطه شروع را از کاربر گرفته و و سپس محیط مورد نظر را ایجاد می کند. محیط بوجود آمده چیـزی جز ماتریس reward نیست که در آن، برای هر خانه، یک امتیاز درنظر گرفته شده است.

ساختار استفاده شده برای این پروژه به صورت زیر می باشد:

pit			pit
wumpus		gold	
		pit	
Agent			

در طراحی نقشه به Wumpus اجازه حرکت داده شده است و از نقطه معلوم شده شروع کرده و با هر حرکت ایجنت، یک خانه در ناحیه قرمز به صورت ساعتگرد حرکت می کند. از طرفی این اجازه به ایجنت داده شده است که یکبار تیر بزند.

متدهای تعریف شده:

Initialization: طراحی محیط بازی

Is_terminal_state: آیا ایجنت به خانه ای غیر عادی رسیده است یا نه؟ اگر خانه دارای امتیاز -۱ بود، False را بازمی گرداند.

Move: انجام تغییرات لازم برای action انتخاب شده. تغییر مختصات ایجنت، تیراندازی و بازگرداندن reward متناسب. مقدار reward هر حرکت منتاسب با صورت مسئله درنظر گرفته شد ولی اگر شلیک انجام شد ولی به هدف برخورد نکرد، یک امتیاز -۵۰۰ اعمال می شود.

Set_wumpus: تغییر دادن مختصات Wumpus و تغییر دادن امتیاز خانه های موجود در ماتریس reward.

این کلاس، کلاس اصلی ما بود اما به صورت مستقیم از این کلاس استفاده نمی شود. ایجنت ها به صورت یک زیرکلاس این کلاس تعریف می شوند.

به جای استفاده از نرخ اکتشاف، از نرخ استخراج استفاده شد و به صورت خطی، با هر اپیزود افزایش پیدا می کند.

QLearning

این ایجنت ویژگی های زیر را دریافت می کند:

rows,

```
columns,
start_point
pit
wumpus,
gold,
episodes,
discount_factor,
learning_rate
```

و سپس بر اساس این ویژگی ها، محیط را طراحی می کرد.

متد ها:

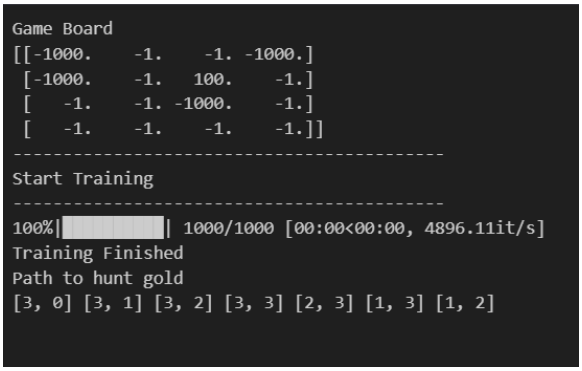
Initialization: علاوه بر ویژگی های متد کلاس WumpusWorld، تنسور q values در این متد تعریف می شود. تنسور q-values دارای شکل ۴*۴*۸ می باشند. ۸ برای تعداد اکشن های موجود می باشد. ۴ جهت حرکت و ۴ جهت تیراندازی.

Next_action: انتخاب یک action با استفاده از الگوریتم epsilon greedy

Updata_q_tabel: بـــا اســـــتفاده از الگـــــوریتم $Q(s,a) := (1-\alpha)Q(s,a) + \alpha\big(R(s) + \gamma Q(s',a')\big)$

Train: متدی که برای آموزش مدل برای تعداد اپیزود های لازم طراحی شده است

Best_path: استخراج بهترین مسیر به ازای هر نقطه شروع



تصویر بالا خروجی نهایی آموزش مدل q learning را نمایش میدهد. مسیر مشخص شده، مسیری است که انتخاب شده تا ایجنت به gold برسد.

Deep Q-Network

این ایجنت ورودی های زیر را دریافت می کند:

```
rows

columns

start_point

pits

wumpus

gold

episodes

discount_factor

learning_rate

replay_memory_capacity

batch_size
```

متدها:

Initialization: در این بخش، شبکه های q_network و target_network به همراه یک لیست خالی به نام replay_memory تعریف می شوند.

Build_q_network: با استفاده از keras یک مدل sequential با ساختار زیر ساخته می شود:

لایه آخر به تعداد تمامی اکشن ها نورون دارد. ۴ جهت حرکت و ۴ جهت تیراندازی. تابع بهینه ساز Adam و تابع هزینه MSE برای این شبکه انتخاب شده اند.

Process_state: faltten کردن state ۴*۴ و تبدیل آن به یک آرایه ۱*۱۶

Next_action: انتخاب action . در اینجا تعریف شده است که بر مبنای جایگاه ایجنت در محیط، حرکات غیرممکن انتخاب نشوند.

Update_q_network: آموزش دادن شبکه های target و q و آپیدیت کردن q_network

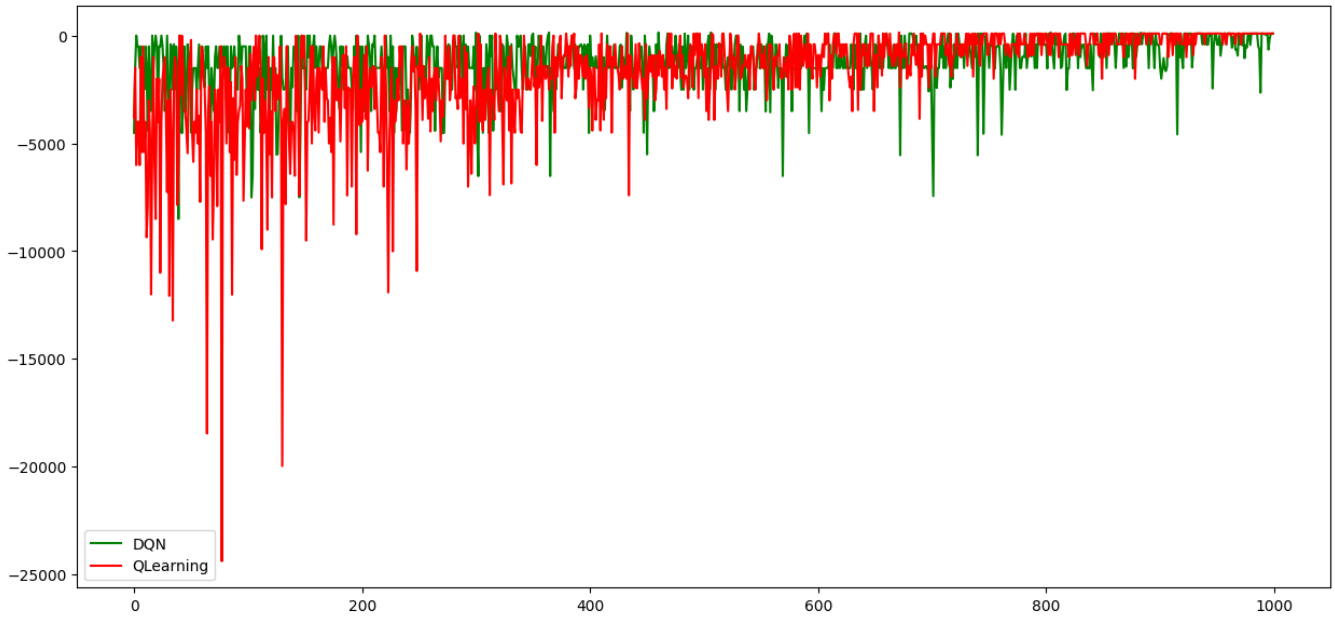
Store_transition: ذخیره کردن (state,action,reward,next state) در لیست replay memory

sample_transitions: نمونه گیری از transition های موجود در replay memory و انتخاب به اندازه batch_size تعریف شده

train: آموزش مدل به تعداد اپیزود های تعریف شده. هر اپیزود تا آنجایی پخش می شود که یا ایجنت بمیرد یا این که طلا را پیدا کند.

Best_path: محاسبه بهترین مسیر ممکن از نقطه آغازین

ب: عملکرد policy



تصویر بالا، مجموع امتیاز های دریافت شده توسط ایجنت ها، طی اپیزود ها به نمایش گذاشته است.

در اوایل کار که نرخ استخراج پایین است (نرخ اکتشاف بالا است) مدل راه های بسیار مختلفی را امتحان می کند تا یا بمیرد یا به طلا برسد. همانطور که مشخص است، با گشت و گذار ایجنت در محیط، policy بروز رسانی می شود و هر چه که نرخ اسخراج بالاتر رود (نرخ اکتشاف پایین آید) امتیاز ایجنت بالاتر می رود و مقادیر منفی دریافت شده توسط مدل کمتر و کمتر می شوند. این به معنای این است که مدل بهترین راه حل را یاد میگیرد.

```
Q-Learning Mean score through all episodes:
-1726.109
Deep Q-Network Mean score  through all episodes:
-1218.964
```

با توجه به شکل بالا می توان یافت که میانگین بدست آمده برای DQN کمتر می باشد. می توان نتیجه گرفت که با استفاده از DQN میتوان به policy بهینه تری دست یافت.

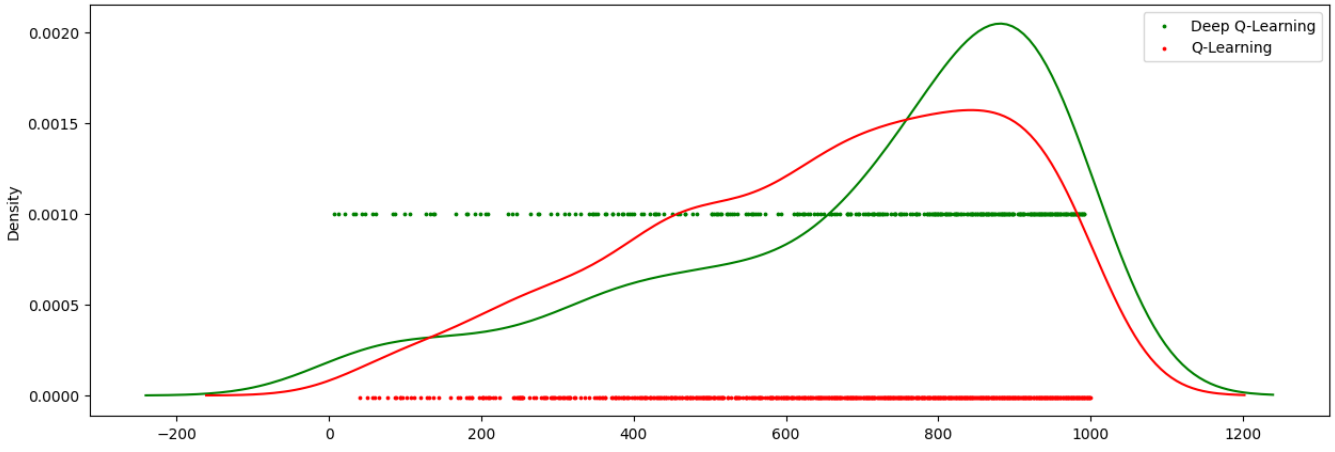
ج: تاثیر نرخ اکتشاف

در حل این سوال، از نرخ استخراج به جای نرخ اکتشاف استفاده شد ولی برای تحیلی این قسمت، نرخ اکتشاف را تحیلی می کنیم.

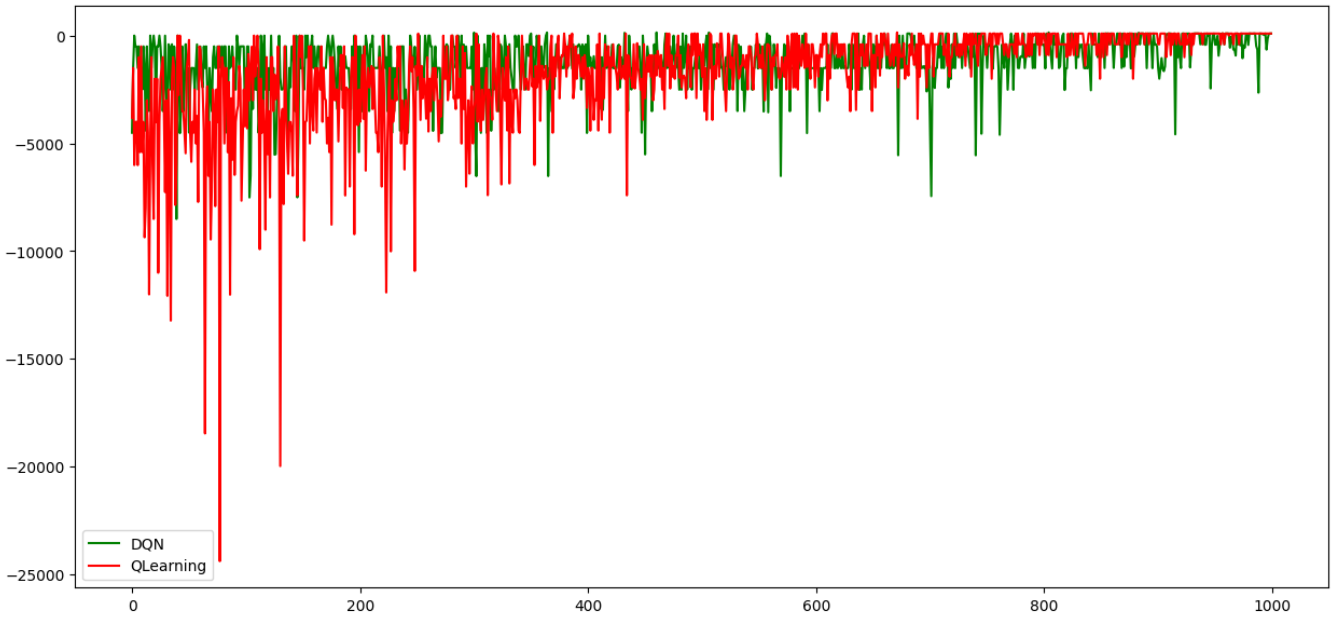
هرچه که نرخ اکتشاف بالاتر باشد، حرکت های رندوم ایجنت نیز بیشتر می شود. یعنی در اکثر اوقات به جای این که actionای را انتخاب کند که بیشترین امتیاز را دارد، یک action رندوم انتخاب می شود. در اوایل کار که policy به صورت رندوم انتخاب شده است، انتخاب action با بیشترین qvalue نمی تواند کارساز باشد. در نتیجه بهتر است که حرکات رندوم را انتخاب کنیم. اما هرچه که جلو تر می رویم و هر چه ایجنت به آخر بازی می رسد، policy آپدیت می شود و مقادیر موجود برای policy آپدیت می شوند. درنتیجه میتواند گفت که میتوانیم از randomness موجود برای انتخاب action بکاهیم. با نزدیکتر شدن به تعداد اپیزود های مورد نظر، از randomness می کاهیم زیر که فرض می کنیم با رسیدن به این تعداد اپیزود، مدل به یک policy مناسب رسیده است و می توان با اطمینان بیشتری به policy عمل کرد.

نرخ اکتشاف بالا امتیاز های پایین تری به ما می دهد ولی با کاهش نرخ اکتشاف (اگر policy نهایی مناسب باشد) امتیاز بالاتری بوجود می آید.

د: کارایی یادگیری



در Q-Learning، با توجه به شکل، مدل سریع تر به حالی می رسد که تراکم یافتن طلاها و نمردن بیشتر است. اما از آنجایی که مقدار epsilon درحال تغییر بوده و درصد انتخاب random درحال تغییر است نمیتوان با اطمینان روی این آمار حساب باز کرد ولی با این حال، Q-Learning سریع تر به policy مناسب رسیده است.



طبق شکل بالا، در اوایل کار، هنگامی که نرخ اکتشاف بالا می باشد، عملکرد Q-Learning بسیار ضعیف تر از مدل DQN می باشد. اما در ادامه و بعد از تقریبا ۵۰۰ اپیزود، مقدار امتیازی که مدل Q-Learning بدست می آورد نسبت به مدل DQN کمتر بود. بر این اساس می توان گفت که مدل Q-Learning policy بهینه را زود تر یادگرفته است.

د: معماری DQN

ساختار شبکه q-network:

Input > 16 تعداد خانه های موجود در محیط بازی

HiddenLayer1 > 64

HiddenLayer2 > 64

تعداد اکشن های ممکن که ایجنت می تواند آن ها را انجام دهد Output > 8

ورودی ۱۶ بعدی می باشد و تمامی این ها ۰ هستند مگر خانه ای که ایجنت روی آن قرار دارد. یعنی این شبکه باید بر اساس خانه فعلی ایجنت، خروجی (اکشنی) متناسب با آن را خروجی دهد.

تعداد لایه های مخفی هم اهمیتی ندارد و بهترین آن ها با استفاده از سعی و خطا بدست می آید که در این جا تعداد نورون های موجود در آن ها تغییر نکرد.

خروجی باید به تعداد اکشن های ممکن باشد. در این جا ۸ خروجی داریم. ۴ جهت ممکن برای حرکت و ۴ جهت ممکن برای تیراندازی. البته باید این نکته را در ذهن داشت که نمیتوان همواره از تمامی این مقادیری استفاده کرد زیرا ممکن است بعضی اوقات استفاده از یک اکشن ممکن نباشد. مثلا هنگامی که تیز پرتاب شده است، دیگر نمی توان از ۴ جهت تیراندازی استفاده کرد. به همین دلیل در تعریف next-action در کلاس DQN مکانیزمی طراحی شد که جلو اتفاق افتادن حرکات غیرمجاز را بگیرد.