Hamed Ajorlou
97101167
February 7, 2021

ML HW5 - Report

Imitation Learning

قصد داریم راجع به اجزا و عملکرد کلاس Replay buffer و utilsصحبت کنیم . در ابتدا هدف کلي اين كلاس را ذكر مي نماييم و سپس به جزييات وبه متغيير هاي مختلف اين كلاس مي پردازيم . Agentدر حال يادگيري ، مسير هاي مختلفي را بنابر policyتعيين شده طي مي نمايد . حال براي ما مطلوب است كه اين مسير هاي طي شده را در حافظه ذخيره سازي بنماييم تا بعدا اين داده ها را تحليل بنماييم ومدل را trainكنيم . path به منظور ذخيره سازي اين مسير طي شده مي باشد . در واقع كلاس pathاطلاعات هر Rolloutرا دسته بندی کرده و در یک دیکشنری قرار می دهد . با هر قدم مشاهدات جديدي به agentداده مي شود كه توسط دستور appendدر ليست مشاهدات obsذخيره ميشود و بنابر اين مشاهدات ، جوایز جدید تحت عنوان rewsبه لیست جوایز rewardsافزوده میشود. در RLبه روش های تکرارشونده تا رسیدن به نتیجه ی دلخواه و سرمنزل مقصود اصطلاحا Roll outگفته می شود . در این کلاس متغیری تحت عنوان Rollout_doneدر نظر گرفته شده است که بنابر طول مسیری که قصد ذخیره ی آن را داریم و یا رسیدن به نتیجه ، مقداردهی می شود و حلقه خاتمه می یابد . تابع convert_listofrollouts نیز اجزای مختلف دیکشنری ها را دریافت میکند و در آرایه های جدا جدا آنها را دسته بندی میکند . ازآن در کلاس Replay Bufferاستفاده شده و اجزای مختلف مسیر در آرایه های مربوطه دسته بندی می شوند سپس در add_rollouts به شی بافر اضافه می شوند .تابع sample_random_dataکه در Replay Bufferتعریف شده است به تعدادی که در ورودی به آن داده می شود به طور تصادفی اجزای مختلف مسیر را باز میگرداند ولی در تابع دیگر sample_recent_dataمشخصا مراحل اخیری که در ورودی تابع ذکر شده است را باز میگرداند .

حال به کلاس loggerمی پردازیم و درباره ی متغیرهای ذکر شده توضیحاتی تقدیم می نماییم . آنچه در initذکر شده است یک Summarywriterمی سازد که رخداد ها و summaryها را در directoryمشخص شده توسط log_dirمی نویسد بنابر این log_dirاین محل ذخیره سازی را تعیین می نماید . آنچه باید ثبت شود در انتظار می ماند و پس از گذشت زمانی که توسط flush_secsتعیین می شود بر روی دیسک نوشته می شود و سایز این صف که در انتظار نوشته شدن می ماند توسط max_queueتعیین می شود . (برای کسب اطلاعات بیشتر ازاین منبع کمک گرفتم : pytorch.org)

حال قصد داریم آنچه در حین کامل کردن کد انجام دادیم را شرح دهیم و توضیحاتی پیرامون آن تقدیم بنماییم . از بخش gym-introductionآغاز می نماییم که در این قسمت observationها دریافت می شود و وزن هایی رندم مربوط به ۴ پارامتر تاثیرگذار ایجاد می نماییم و با ضرب کردن آنها و ساین گرفتن از نتیجه ی آن ، action حاصل می شود .حال به شکلی تکرار شونده این actionهای بدست آمده را دوباره به enviormentمی دهیم تا دوباره observation دریافت کنیم این روند ادامه می یابد تا به نقطه ی Doneبرسیم و حلقه متوقف شود و بازی تمام شود برای مثال میله افقی شود . همچنین بیشترین مرحله ای که از شروع تا خاتمه ی بازی میگذرد و در متغیر cntثبت می شود را به عنوان بهترین وزن ذخیره مینماییم ودر انتها یک بار با این وزن ها الگوریتم را اجرا می نماییم .

در قسمت torch utilsیک مدل نورال نت طراحی می نماییم به مانند تمرین قبلی و نکته ی مهم و قابل ذکر در مفهوم جدیدی به نام xavier initializationمی باشد . دلیل اینکه از xavier initializationاستفاده می نماییم آن است که سطح نورال نت ساخته شده لوکال مینیمم های فراوانی دارد و برای آنکه به نتیجه ی مطلوب برسد به وزن های اولیه توزیع احتمالاتی خاصی نسبت می دهیم . من با جستجو در این باره از توزیع احتمالاتی یونیفورم استفاده کردم .

طبق آنچه صحبت کردیم policyمدل از یک نورال نت نشئت می گیرد ودر قسمت MLP policyقصد داریم ازین شبکه actionها را استخراج کنیم . برای این کار observationها را به عنوان ورودی به مدل می دهیم و خروجی آن را دریافت می نماییم . با توجه به خروجی ، در صورتی که enviormentگسسته باشد ، خروجی شبکه را به عنوان یک مدل احتمالاتی چند جمله ای در نظر میگیریم و درصورتیکه enviormentپیوسته باشد ، اکشن از یک مدل گاوسی گرفته می شود که مطابق زیر است .

$$a_i \sim \text{mean}(s_i) + e^{\text{logstd}}\mathcal{N}(0,1)$$

سپس برای دو حالت گسسته و پیوسته با Loss functionهای تعیین شده در کامنت ، optimizerطراحی می نماییم و در بخش updateاین Loss functionرا مینیمایز می نماییم . در بخش loaded_gaussian_policyمدل های ارائه شده به عنوان expertدر مدل load میشوند و در قسمت rl_trainerدر itteration =0تعیین مینماییم که از policy تعیین شده توسط expertتبعیت کند و در itterationهای بعدی از collect policyو در ادامه observationها را به expert policyمی دهیم و actionهای بدست آمده را جایگزین میکنیم با actionهایی که خود expertپیش بینی کرده بود که ممکن است اشتباه باشد و به نتیجه ی دلخواه convergeنکند . با تکمیل همه ی این قسمت ها و ران کردن کد های ذکر شده در فایل Read meنتایج بدست آمده ، که در ادامه ذکر می شود و همچنین تصاویر تنسور بورد که از local hostدریافت می شود در انتهای گزارش درج شده است .

خروجی بدست آمده در Gym – introduction

```
The best length is: 79.55
The best length is: 130.86
The best length is: 139.48
The best length is: 200.0
The best length is: 200.0
The best length is: 200.0
The best length is: 200.0
The best length is: 200.0
The best length is: 200.0
The best length is: 200.0
game lasted 10 moves
```

سپس کد های ذکر شده مربوط به یادگیری تقلیدی را به ترتیب ران میکنیم :

```
Collecting data for eval...
Eval_AverageReturn : 500.0
Eval_StdReturn : 0.0
Eval_MaxReturn : 500.0
Eval_MinReturn : 500.0
Eval_AverageEpLen : 500.0
Train_AverageReturn : 496.3999938964844
Train_StdReturn : 5.083305835723877
Train_MaxReturn : 500.0
Train_MinReturn : 487.0
Train_AverageEpLen : 496.4
Train_EnvstepsSoFar : 0
TimeSinceStart : 2.9023168087005615
Initial_DataCollection_AverageReturn : 496.3999938964844
Done logging...
```

```
Collecting data for eval...
Eval_AverageReturn : 114.39260864257812
Eval_StdReturn : 131.46856689453125
Eval_MaxReturn : 245.86117553710938
Eval_MinReturn : -17.075965881347656
Eval_AverageEpLen : 158.5
Train_AverageReturn : 218.96922302246094
Train_StdReturn : 87.94571685791016
Train_MaxReturn : 280.5846252441406
Train_MinReturn : 55.61897277832031
Train_AverageEpLen : 230.22222222222223
Train_EnvstepsSoFar : 0
TimeSinceStart : 3.4859776496887207
Initial_DataCollection_AverageReturn : 218.96922302246094
Done logging...
```

```
Collecting data for eval...
Eval_AverageReturn : 258.10260009765625
Eval_StdReturn : 0.0
Eval_MaxReturn : 258.10260009765625
Eval_MinReturn : 258.10260009765625
Eval_AverageEpLen : 331.0
Train_AverageReturn : 172.6307373046875
Train_StdReturn : 15.753379821777344
Train_MaxReturn : 188.38412475585938
Train_MinReturn : 156.8773651123047
Train_AverageEpLen : 1000.0
Train_EnvstepsSoFar : 0
TimeSinceStart : 5.518812417984009
Initial_DataCollection_AverageReturn : 172.6307373046875
Done logging...
```

: Dagger

۳ نتیجه از ۱۰ بار تکرار برای هر یک از دگر ها در گزارش ذکر شده است :

LunarLander

```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.5199, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(0.5926, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(0.6996, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(0.5138, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(0.4728, grad_fn=<NllLossBackward>)
itr # 7 :   loss: tensor(0.5944, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 306.7672119140625
Eval_StdReturn : 0.0
Eval_MaxReturn : 306.7672119140625
Eval_MinReturn : 306.7672119140625
Eval_AverageEpLen : 256.0
Train_AverageReturn : 277.0559997558594
Train_StdReturn : 10.193094253540039
Train_MaxReturn : 293.86517333984375
Train_MinReturn : 263.8361511230469
Train_AverageEpLen : 216.4
Train_EnvstepsSoFar : 7730
TimeSinceStart : 24.794593334197998
Initial_DataCollection_AverageReturn : 218.96922302246094
Done logging...
```

```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.7175, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(0.5548, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(0.6474, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(0.4478, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(0.5638, grad_fn=<NllLossBackward>)
itr # 8 :   loss: tensor(0.5479, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 268.0810852050781
Eval_StdReturn : 0.0
Eval_MaxReturn : 268.0810852050781
Eval_MinReturn : 268.0810852050781
Eval_AverageEpLen : 251.0
Train_AverageReturn : 172.53292846679688
Train_StdReturn : 107.80461883544922
Train_MaxReturn : 270.86602783203125
Train_MinReturn : 28.444793701171875
Train_AverageEpLen : 219.8
Train_EnvstepsSoFar : 8829
TimeSinceStart : 28.025717735290527
Initial_DataCollection_AverageReturn : 218.96922302246094
Done logging...
```

```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.5691, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(0.6448, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(0.5738, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(0.5859, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(0.5347, grad_fn=<NllLossBackward>)
itr # 9 :   loss: tensor(0.5369, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 250.9696502685547
Eval_StdReturn : 0.0
Eval_MaxReturn : 250.9696502685547
Eval_MinReturn : 250.9696502685547
Eval_AverageEpLen : 225.0
Train_AverageReturn : 194.44985961914062
Train_StdReturn : 127.45550537109375
Train_MaxReturn : 308.0882263183594
Train_MinReturn : 37.22319412231445
Train_AverageEpLen : 204.6
Train_EnvstepsSoFar : 9852
TimeSinceStart : 30.928033351898193
Initial_DataCollection_AverageReturn : 218.96922302246094
Done logging...
```

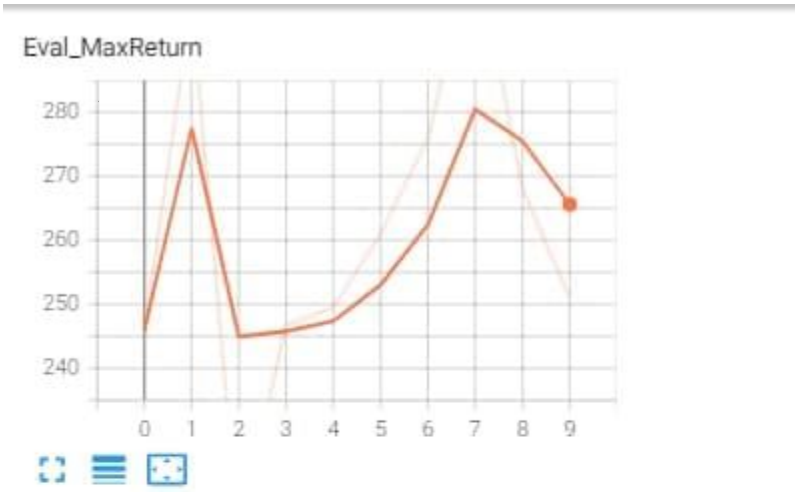
LunarLandercontinous

```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.5436, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.2331, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.3271, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.1652, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.2435, grad_fn=<MseLossBackward>)
itr # 7 :   loss: tensor(0.2926, grad_fn=<MseLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 180.29721069335938
Eval_StdReturn : 94.48086547851562
Eval_MaxReturn : 274.778076171875
Eval_MinReturn : 85.81634521484375
Eval_AverageEpLen : 195.5
Train_AverageReturn : 255.3571014404297
Train_StdReturn : 14.89013957977295
Train_MaxReturn : 271.17547607421875
Train_MinReturn : 232.22781372070312
Train_AverageEpLen : 258.0
Train_EnvstepsSoFar : 8435
TimeSinceStart : 36.78365659713745
Initial_DataCollection_AverageReturn : 172.6307373046875
Done logging...
```

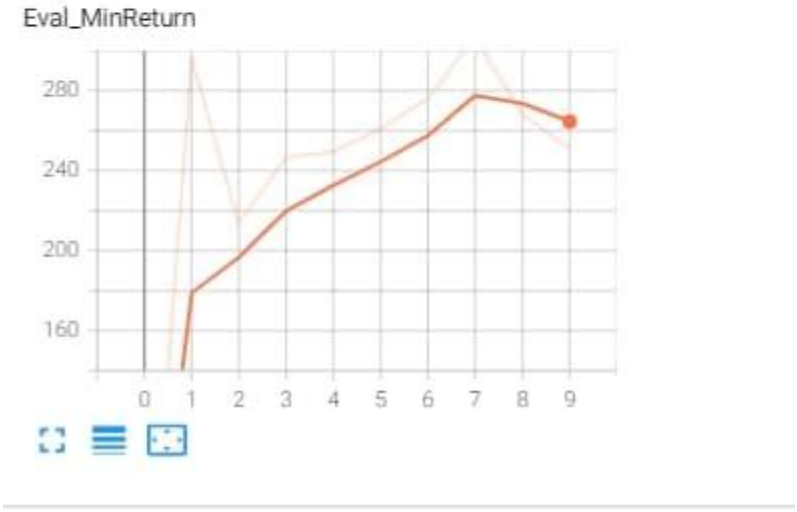
```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.2470, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.2110, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.2203, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.3742, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.2181, grad_fn=<MseLossBackward>)
itr # 8 :   loss: tensor(0.3115, grad_fn=<MseLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 308.8662109375
Eval_StdReturn : 0.0
Eval_MaxReturn : 308.8662109375
Eval_MinReturn : 308.8662109375
Eval_AverageEpLen : 280.0
Train_AverageReturn : 265.43487548828125
Train_StdReturn : 17.038076400756836
Train_MaxReturn : 285.86065673828125
Train_MinReturn : 241.5272979736328
Train_AverageEpLen : 270.25
Train_EnvstepsSoFar : 9516
TimeSinceStart : 40.414562702178955
Initial_DataCollection_AverageReturn : 172.6307373046875
Done logging...
```

```
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.4508, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.3913, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.2752, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.2460, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.2213, grad_fn=<MseLossBackward>)
itr # 9 :   loss: tensor(0.1992, grad_fn=<MseLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 43.98357009887695
Eval_StdReturn : 0.0
Eval_MaxReturn : 43.98357009887695
Eval_MinReturn : 43.98357009887695
Eval_AverageEpLen : 255.0
Train_AverageReturn : 226.8162841796875
Train_StdReturn : 120.89540100097656
Train_MaxReturn : 323.73065185546875
Train_MinReturn : 19.594528198242188
Train_AverageEpLen : 283.75
Train_EnvstepsSoFar : 10651
TimeSinceStart : 44.074424266815186
Initial_DataCollection_AverageReturn : 172.6307373046875
Done logging...
```

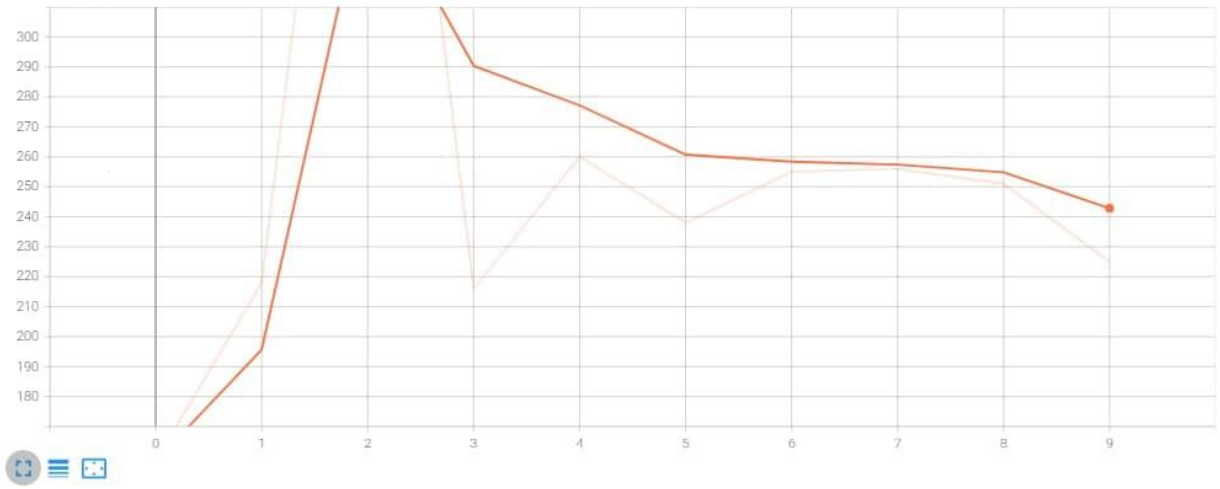
برخی از تنسور بورد های بدست آمده :



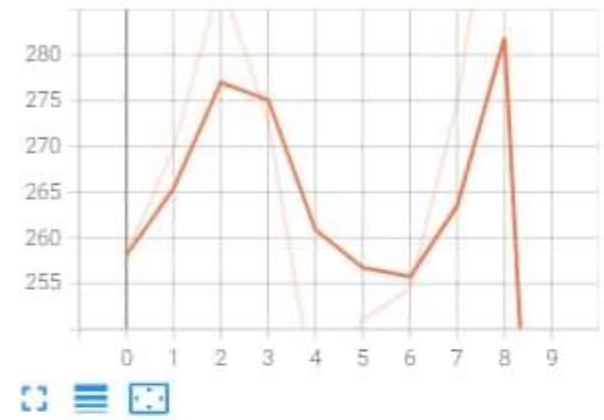
Eval_MinReturn



Eval_AverageEpLen

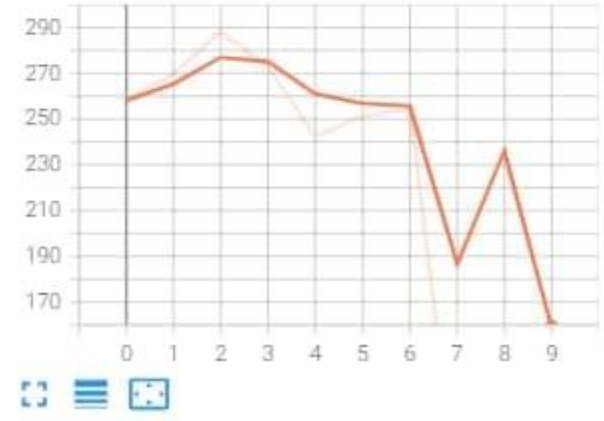


Eval_MaxReturn



Eval_MinReturn

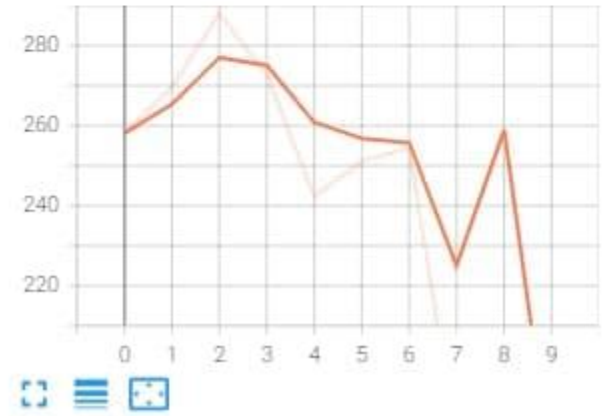
Eval_MinReturn



Eval_AverageEpLen

Eval_AverageReturn

Eval_AverageReturn



با تشکر از همراهی شما