تمرین پنجم یادگیری ماشین

محمد زنگویی ۹۵۱۰۱۵۹۳

ReplayBuffer Class

این کلاس اطلاعات مسیرهایی که agent طی می کند را ذخیره می‌کند تا برای آموزش استفاده شود. درعمل، تعدادی آرایه از مجموعه ای از مسیرها به صورت concatenate شده در متغیرهای کلاس ذخیره می‌شوند. این متغیرها به قرار زیر هستند: مسیرها (path)، مشاهدات بازی در گام فعلی (obs)، اقدامات (acs)، پاداش‌ها (rews)، مشاهدات گام بعدی (next obs) و علامت پایان بازی (terminal). هم‌چنین این کلاس شامل سه متود است: ۱) add_rollouts که اطلات مسیرهای طی‌شده ورودی خود را به متغیرهای کلاس اضافه می‌کند. ۲) sample_random_data که تعدادی دلخواه از گام‌های مسیر را انتخاب می‌کند و به صورت batch برمی‌گرداند. ۳) sample_recent_data که همانند بخش قبل عمل می‌کند با این تفاوت که آخرین گام‌های مسیر را برمی‌گرداند.

Logger Class

Log_dir: محل ذخیره فایل log

max_queue: طول صفی از رخدادهای درانتظار و خلاصه ها که قبل از نوشتن در دیسک در آن باقی می‌مانند و هم‌زمان نوشته خواهندشد. باتوجه به مقدار پیشفرض تعیین شده در اینجا، حداکثر هر دوبار یک رخداد در دیسک نوشته خواهدشد.

flush_secs: حداکثر مدت زمانی را مشخص می‌کند که یک رخداد بایستی صبر کند تا در دیسک نوشته‌شود. در واقع محدودیت قائل می‌شود برای منتظرماندن رخدادها تا نوشته شدنشان.

Implementation Summary

۱. torch_utils.py

در این فایل، لایه های شبکه موردنظر ایجاد می شود. در یک حلقه با تعداد گام هایی به اندازه تعداد لایه های شبکه، لایه های شبکه را به همراه تابع activation اضافه میکنیم. توجه شود که عرض لایه اول و آخر با سایر لایه ها متفاوت است. همچنین از xavir initialization استفاده شده است با تابع گین tanh.

۲. rl_trainer.py

تابع run_training_loop دربردارنده حلقه اصلی یادگیری است، یعنی حرکت و نمونه برداری در محیط، بررسی حرکت اکسپرت و آموزش شبکه. برای بخش نمونه برداری در محیط، در ایتریشن اول از پالیسی اکسپرت استفاده می‌کنیم ولی در ایتریشن های بعدی از پالیسی خودمان که در حال آموزشش هستیم. در بخش بعدی که باید کامل کنیم، از پالیسی اکسپرت برای تعیین اقدام مناسب استفاده می‌کنیم، این کار را با استفاده از تابع get_action کلاس مربوطه انجام می‌دهیم.

۳. loaded_gaussian_policy.py

از کلاس موجود در این فایل، با خواندن از فایل های آماده که شبکه آموزش دیده این کار هستند، به عنوان اکسپرت پالیسی استفاده می‌کنیم. ابتدا بایستی پارامترهای این کلاس را تنظیم کنیم که به سادگی از فایل خوانده‌شده به صورت دیکشنری در دسترس است. برای تابع get_action نیز بایستی در حالت گسسته، خروجی مدل را به یک توزیع mutinomial بدهیم و از آن نمونه بگیریم و نتیجه را به فرمت numpy تبدیل کنیم، چراکه env مربوط به پکیج gym، چنین ورودی ای را قبول می‌کند. برای حالت پیوسته نیز، هم باید میانگین را از شبکه بگیریم و هم logstd را و یک متغیر گاوسی با آن میانگین و std تولید کنیم و مجدد به فرمت numpy تبدیل کنیم.

۴. MLP_policy.py

در این فایل، مدل آموزش ما ایجاد می‌شود و متدهای لازم یادگیری، نوشته می‌شوند.

متد get_action به کلی همانند بخش قبل است. در متد define_train بایستی برای هرکدام از حالات گسسته و پیوسته، الگوریتم بهینه سازی و تابع هزینه را تعریف کنیم. برای حالت گسسته همان پارامترهای شبکه را برای بهینه‌سازی انتخاب میکنیم با adam optimizer و نیز از cross enthropy برای تابع هزینه، اما برای حالت پیوسته علاوه بر پارامترهای شبکه، logstd را نیز بایستی برای بهینه سازی تعریف کنیم. در اینجا از MSELoss برای تابع هزینه استفاده می‌کنیم. در متد update نیز برای حالت گسسته، خروجی شبکه را به همراه پاسخ اکپسرت به تابع هزینه می‌دهیم و برای حالت پیوسته، متغیری گاوسی به میانگین خروجی شبکه و std که تعریف کرده بودیم را به تابع هزینه میدهیم. و در ادامه back propagation را اعمال میکنیم و یک گام متغیرهای بهینه سازی را آپدیت می‌کنیم

Result

با اجرای دستورهای خواسته شده در فایل README.md به نتایج زیر رسیدم:

```
1. $python hw5/scripts/run_hw5_behavior_cloning.py --expert_policy_file
   hw5/models/CartPole-v0.tar --env_name CartPole-v1 --exp_name
   test_bc_Cart --n_iter 1
```

```
********** Iteration 0 ************
Collecting data from the expert policy...
Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.8706, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(0.8235, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(0.8316, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(0.7621, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(0.8129, grad_fn=<NllLossBackward>)
itr # 0 :   loss: tensor(0.7775, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 13.125
Eval_StdReturn : 3.497767210006714
Eval_MaxReturn : 20.0
Eval_MinReturn : 9.0
Eval_AverageEpLen : 13.125
Train_AverageReturn : 486.3999938964844
Train_StdReturn : 16.668533325195312
Train_MaxReturn : 500.0
Train_MinReturn : 465.0
Train_AverageEpLen : 486.4
Train_EnvstepsSoFar : 0
TimeSinceStart : 1.4202356338500977
Initial_DataCollection_AverageReturn : 486.3999938964844
Done logging...
```

2. python hw5/scripts/run_hw5_behavior_cloning.py --expert_policy_file hw5/models/LunarLander-v2.tar --env_name LunarLander-v2 --exp_name test_bc_Lunar --n_iter 1

```
********** Iteration 0 ************

Collecting data from the expert policy...

Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(1.6324, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(1.6106, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(1.6506, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(1.4985, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(1.4743, grad_fn=<NllLossBackward>)
itr # 0 :   loss: tensor(1.6582, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : -359.1578369140625
Eval_StdReturn : 31.668027877807617
Eval_MaxReturn : -321.06005859375
Eval_MinReturn : -398.5958251953125
Eval_AverageEpLen : 72.66666666666667
Train_AverageReturn : 285.37738037109375
Train_StdReturn : 15.324864387512207
Train_MaxReturn : 302.8497619628906
Train_MinReturn : 256.501953125
Train_AverageEpLen : 225.44444444444446
Train_EnvstepsSoFar : 0
TimeSinceStart : 2.036033868789673
Initial_DataCollection_AverageReturn : 285.37738037109375
Done logging...
```

3. `python hw5/scripts/run_hw5_behavior_cloning.py --expert_policy_file hw5/models/LunarLanderContinuous-v2.tar --env_name LunarLanderContinuous-v2 --exp_name test_bc_LunarCont --n_iter 1`

```
********** Iteration 0 ************


Collecting data from the expert policy...


Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(2.4520, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.9078, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.6840, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.6619, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.5994, grad_fn=<MseLossBackward>)
itr # 0 :   loss: tensor(0.5626, grad_fn=<MseLossBackward>)


Beginning logging procedure...


Collecting data for eval...
Eval_AverageReturn : 266.40740966796875
Eval_StdReturn : 0.0
Eval_MaxReturn : 266.40740966796875
Eval_MinReturn : 266.40740966796875
Eval_AverageEpLen : 252.0
Train_AverageReturn : 173.0203857421875
Train_StdReturn : 17.28575897216797
Train_MaxReturn : 190.30613708496094
Train_MinReturn : 155.734619140625
Train_AverageEpLen : 1000.0
Train_EnvstepsSoFar : 0
TimeSinceStart : 4.536060094833374
Initial_DataCollection_AverageReturn : 173.0203857421875
Done logging...
```

4. python hw5/scripts/run_hw5_behavior_cloning.py --expert_policy_file hw5/models/LunarLander-v2.tar --env_name LunarLander-v2 --exp_name test_dagger_Lunar --n_iter 10 --do_dagger

به علت طولانی بودن خروجی، فقط ۲ ایتریشن نهایی را اینجا می اورم

```
********** Iteration 8 ************
Collecting data to be used for training...
Relabelling collected observations with labels from an expert policy...

Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(2.0662, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(2.0901, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(2.0340, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(2.0595, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(2.1209, grad_fn=<NllLossBackward>)
itr # 8 :   loss: tensor(2.1564, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : -125.03182983398438
Eval_StdReturn : 115.49903869628906
Eval_MaxReturn : 9.565048217773438
Eval_MinReturn : -272.47265625
Eval_AverageEpLen : 73.66666666666667
Train_AverageReturn : -276.2174377441406
Train_StdReturn : 166.0626220703125
Train_MaxReturn : 19.611557006835938
Train_MinReturn : -596.334228515625
Train_AverageEpLen : 81.84615384615384
Train_EnvstepsSoFar : 8269
TimeSinceStart : 13.106196880340576
Initial_DataCollection_AverageReturn : 285.37738037109375
Done logging...
```

```
********** Iteration 9 ************

Collecting data to be used for training...

Relabelling collected observations with labels from an expert policy...

Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(1.9665, grad_fn=<NllLossBackward>)
train step # 200   loss:  tensor(2.1876, grad_fn=<NllLossBackward>)
train step # 400   loss:  tensor(2.0688, grad_fn=<NllLossBackward>)
train step # 600   loss:  tensor(2.1171, grad_fn=<NllLossBackward>)
train step # 800   loss:  tensor(2.1219, grad_fn=<NllLossBackward>)
itr # 9 :   loss: tensor(2.0494, grad_fn=<NllLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : -243.10850524902344
Eval_StdReturn : 97.7813491821289
Eval_MaxReturn : -122.52119445800781
Eval_MinReturn : -362.0179748535156
Eval_AverageEpLen : 88.0
Train_AverageReturn : -250.55459594726562
Train_StdReturn : 103.06645965576172
Train_MaxReturn : -127.32776641845703
Train_MinReturn : -466.0693054199219
Train_AverageEpLen : 69.86666666666666
Train_EnvstepsSoFar : 9317
TimeSinceStart : 14.462376594543457
Initial_DataCollection_AverageReturn : 285.37738037109375
Done logging...
```

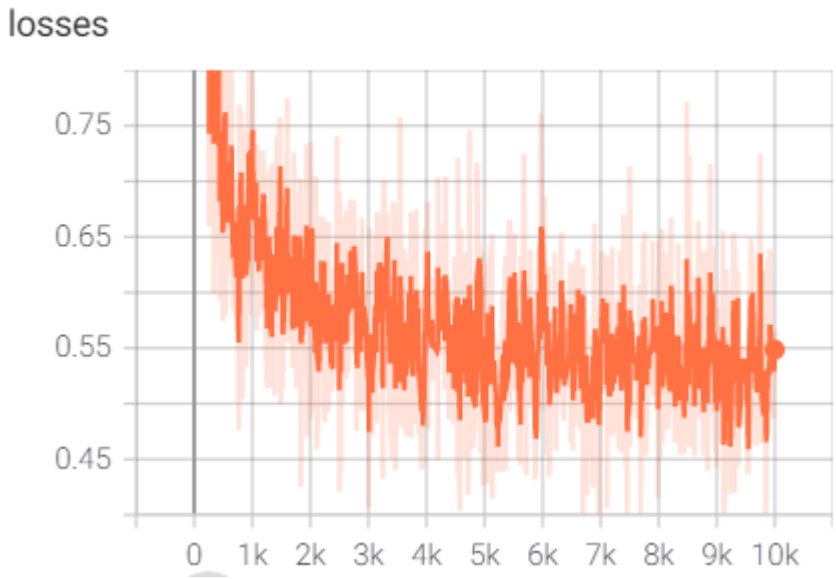
5. python hw5/scripts/run_hw5_behavior_cloning.py --expert_policy_file hw5/models/LunarLanderContinuous-v2.tar --env_name LunarLanderContinuous-v2 --exp_name test_dagger_LunarCont --n_iter 10 --do_dagger

به علت طولانی بودن خروجی، فقط ۲ ایتریشن نهایی را اینجا می اورم

```
********** Iteration 8 ************
Collecting data to be used for training...
Relabelling collected observations with labels from an expert policy...

Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.4602, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.6030, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.4596, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.6722, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.5887, grad_fn=<MseLossBackward>)
itr # 8 :   loss: tensor(0.4428, grad_fn=<MseLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 272.4130554199219
Eval_StdReturn : 0.0
Eval_MaxReturn : 272.4130554199219
Eval_MinReturn : 272.4130554199219
Eval_AverageEpLen : 211.0
Train_AverageReturn : 286.09429931640625
Train_StdReturn : 20.281475067138672
Train_MaxReturn : 316.68017578125
Train_MinReturn : 262.08428955078125
Train_AverageEpLen : 243.2
Train_EnvstepsSoFar : 9600
TimeSinceStart : 28.326191663742065
Initial_DataCollection_AverageReturn : 173.0203857421875
Done logging...
```

```
********** Iteration 9 ************

Collecting data to be used for training...

Relabelling collected observations with labels from an expert policy...

Training agent using sampled data from replay buffer...
train step # 0   loss:  tensor(0.5737, grad_fn=<MseLossBackward>)
train step # 200   loss:  tensor(0.4870, grad_fn=<MseLossBackward>)
train step # 400   loss:  tensor(0.5083, grad_fn=<MseLossBackward>)
train step # 600   loss:  tensor(0.6408, grad_fn=<MseLossBackward>)
train step # 800   loss:  tensor(0.4579, grad_fn=<MseLossBackward>)
itr # 9 :   loss: tensor(0.5550, grad_fn=<MseLossBackward>)

Beginning logging procedure...

Collecting data for eval...
Eval_AverageReturn : 272.9924621582031
Eval_StdReturn : 0.0
Eval_MaxReturn : 272.9924621582031
Eval_MinReturn : 272.9924621582031
Eval_AverageEpLen : 235.0
Train_AverageReturn : 281.63250732421875
Train_StdReturn : 21.17298126220703
Train_MaxReturn : 309.4694519042969
Train_MinReturn : 254.96287536621094
Train_AverageEpLen : 237.6
Train_EnvstepsSoFar : 10788
TimeSinceStart : 31.245308876037598
Initial_DataCollection_AverageReturn : 173.0203857421875
Done logging...
```

Plot



dagger_LunarCont_LunarLanderContinuous