



LunarLander-v2.0 Solve

Best score:231

Knn meisters
Александр Мячин
Потемин Роман

Метрика

LunarLander -v2 определяет решение как получение среднего reward 200 за 100 последовательных испытаний



ХОД ИССЛЕДОВАНИЯ



Шаг первый

Взяли бейзлайн модели и реализацию [DQN](#) агента и адаптировали к нашей задаче, изучив результаты [работы](#) второго места лидерборда. По факту, почти весь код был переписан. Таким образом мы получили средний reward 200 за более чем 1000 итераций(на обучении)



Шаг второй

Изменили алгоритм обновления весов фиксированной нейросети. Вместо простой замены весов мы стали их взвешивать.

Это дало средний reward 200 за 800 итераций (на обучении)

```
def update_weight(self, model_from, model_to, tau):  
    for from_p, to_p in zip(model_from.parameters(), model_to.parameters()):  
        to_p.data.copy_(tau*from_p.data + (1.0-tau)*to_p.data)
```

Шаг третий

Переписали память на питру массивы – это дало серьезный прирост к скорости обучения модели. И инициализировали ее размером батча (делая случайные действия в среде)



Шаг четвертый

Теперь нам нужно “затюнить” нашу модель, изменяя различные гиперпараметры. Сам важным параметром является EPSILON(контролирует начальное изучение среды), определяющий вероятность совершения случайного действия, также мы минимизировали его различными методами.Линейно, экспоненциально и.т.д.Теперь мы получили средний reward 200 за около 600 итераций на обучении



Шаг пятый – general feature

В нашей задаче максимальный reward за действие равен 100 и дается он при успешной посадке.

Наш алгоритм пытается получить максимальный reward за итерацию и начинает летать, пытаясь получить reward за успешную посадку еще раз. Таким образом, чтобы избежать уменьшения сора за использование основного двигателя, мы заканчиваем эпизод, если наш “Луноход” приземлился на землю.

Хотя данное изменение не сильно повлияло за скор, но позволило сильно сократить время обучения.

```
def kostil(reward): # Отключение
    return (
        reward == 100 or
        reward == -100 or
        reward == 10 or
        reward == 200
    )
```


Torch model

Наша модель имеет 3 слоя, с 64 нейронами на каждом слою

Функция активации:

Relu между слоями

Linear на выходе

Оптимизатор – Adam

Функция ошибки – Huber

Uses Huber loss instead of squared loss on Bellman error:

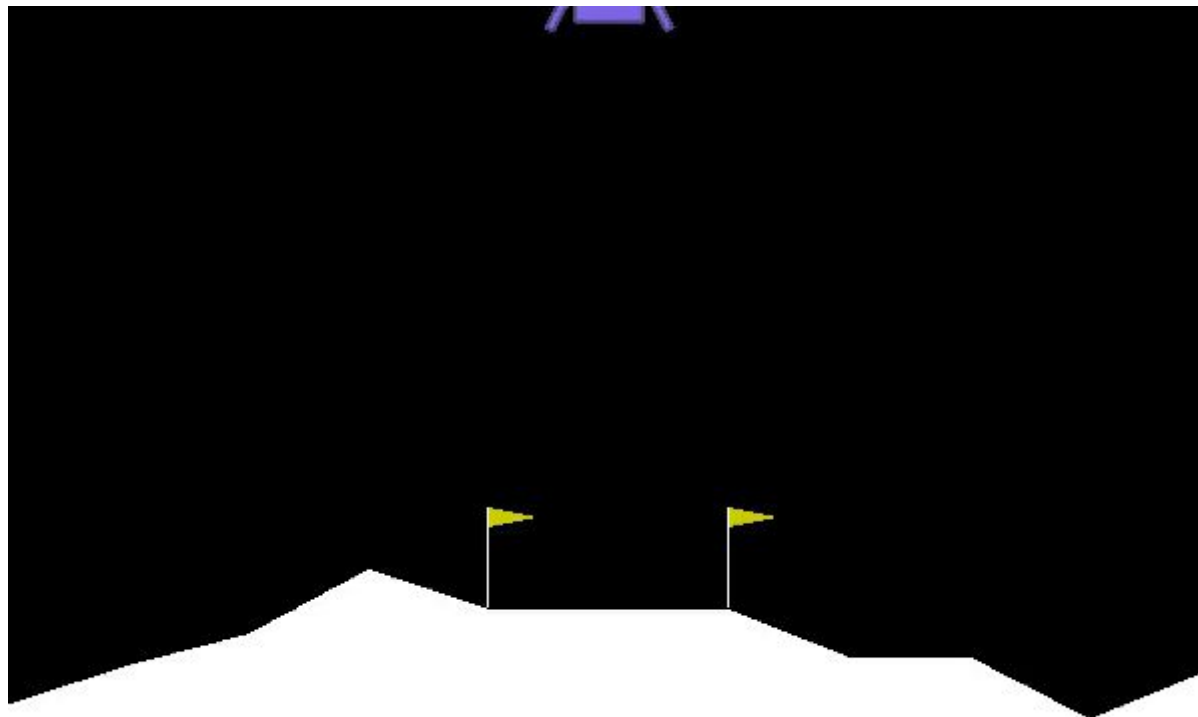
$$L_{\delta}(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

100

Используя все описанное выше, мы получили решение(т.е средний reward 200) за 231 одну итерацию(на обучении)

```
surfa@furfaserv:~/work/LunarLander-v2-Solve/Scripts$ ipython MAIN.1.py
100%|██████████████████████████████████████████████████████████████████████████████| 64/64 [00:00<00:00, 86.19it/s]
Заполнение памяти случайными действиями завершено
[S] -100.37 [MS] -72.16:   2%|███████████| 100/5000 [02:17<1:52:22, 1.38s/it]
[MEAN_SCORE] -72.16 [STD_SCORE] 114.36
-----
[S] 189.04 [MS] 155.83:   4%|███████████| 200/5000 [03:55<1:34:08, 1.18s/it]
[MEAN_SCORE] 155.83 [STD_SCORE] 123.77
-----
[S] 233.90 [MS] 200.38:   5%|███████████| 231/5000 [04:15<1:27:45, 1.10s/it]
Early stopping
```

Результаты



Библиотеки

tqdm

```
In [*]: from tqdm import trange, tqdm_notebook
        from time import sleep
        for i in trange(4, desc='1st loop'):
            for j in trange(100, desc='2nd loop'):
                sleep(0.01)
```

1st loop: 25% 1/4 [00:01<00:03, 1.25s/it]

2nd loop: 100% 100/100 [00:01<00:00, 88.89it/s]

```
In [3]: from tqdm import trange, tqdm_notebook
        from time import sleep
        for i in tqdm_notebook(xrange(4), desc='1st loop'):
            for j in tqdm_notebook(xrange(100), desc='2nd loop', leave=False):
                sleep(0.01)
```



Walker2d-v0
Make a 2D robot
walk.



Ant-v0
Make a 3D four-
legged robot walk.

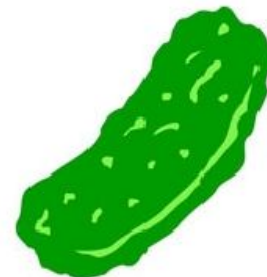


Humanoid-v0
Make a 3D two-
legged robot walk.

 PyTorch

 OpenAI

 NumPy

 pickle



Спасибо

Knn meisters

Github link:

Александр Мячин

Потемин Роман