

RLChina第二次习题课参考教程

题目要求:

开搞:

第一步: Clone代码到本地

第二步: 使用Pycharm打开项目

第三步: 在cliffwalking环境, 训练tabularq (Q-learning)

第四步: 完成submission.py文件 提交代码

说在最后:

代码附录: submission.py

这个版本:

1. 增加了anaconda 和pycharm的安装链接
2. 增加了`--reload_config` 参数的用法

RLChina第二次习题课参考教程

嘻嘻~~ 我又来了~

题目要求:

作业的目的:

本次难度递进稍稍有一些大, 但是新手同学大家别灰心, 可以多花一点的时间慢慢去摸索实现。

本次作业的目的就是为了让大家

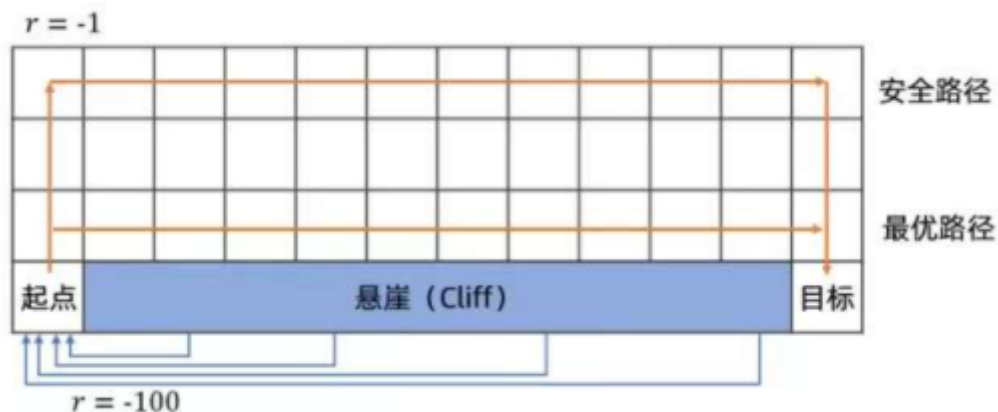
- 熟悉SARSA、Q-Learning算法,
- 通过实际动手训练, 建立起一个算法和游戏的联系
- 其中预留了一点点代码让大家去补充一下(实在不会的同学可以看下答案 理解一下)

通过实际动作去感受下, 如何用强化学习算法学习寻找到一条最优的路径

作业方式及要求:

通过给悬崖寻路小游戏, 提交一个SARSA或者Q-Learning的算法代码训练结果

算法比random好10%就行(咋也得找到路 不能比瞎走还low吧 哈哈)



开搞：

这种小量级的代码，一台笔记本足矣搞定，本地训练就完了

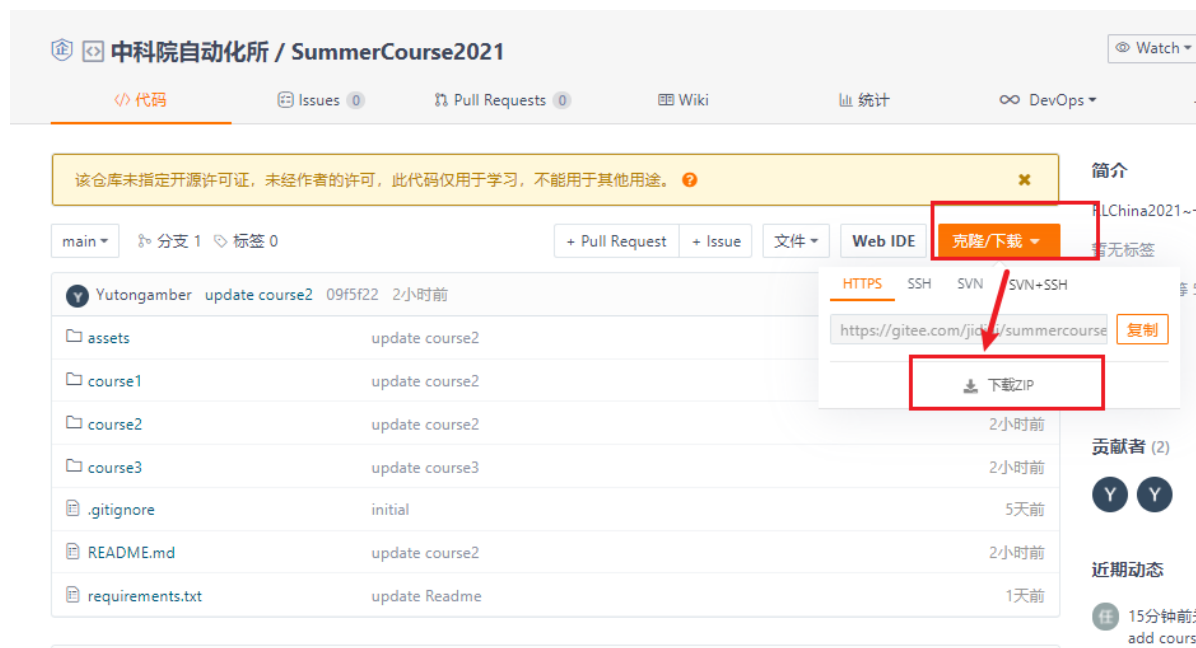
第一步：Clone代码到本地

Github: <https://github.com/jidiai/SummerCourse2021/blob/HEAD/course2/README.md>

Gitee: <https://gitee.com/jidiai/summercourse2021/tree/main>

打开上面的任意一个链接：这里以gitee为例

点击下载zip即可(会使用git的同学可以使用git clone到本地 方便后面更新和版本控制 不会也木事)



第二步：使用Pycharm打开项目

这里只是推荐我常用的编辑器打开哈，大家有自己习惯的vscode、jupyter啥的，用自己熟悉的就行

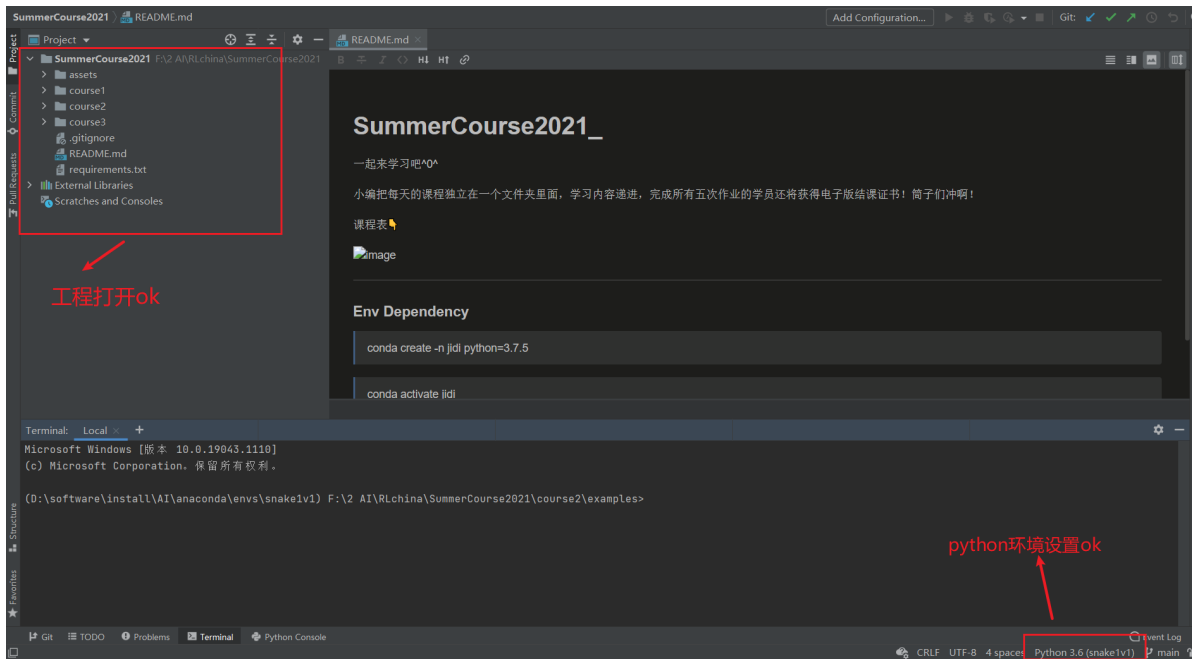
安装anaconda 和 pycharm 有问题的同学复制这个句话上百度搜一下学习一下哈

这里给大家放一个可行的吧

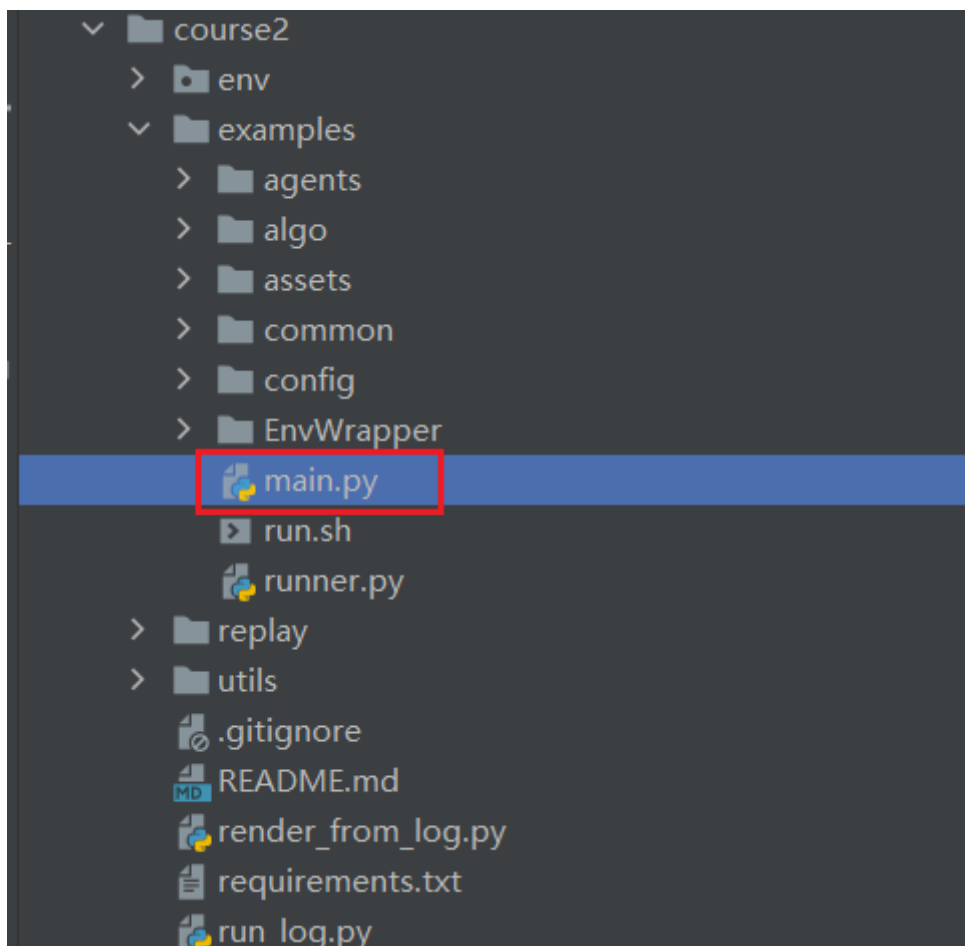
[PyCharm与Anaconda超详细安装配置教程] https://blog.csdn.net/qg_32892383/article/details/116137730

这里默认大家配置python环境啥的都没有问题啦

有问题私聊我帮大家远程看看也行(时间允许的话哈 建议自己搞搞)



原文件结构



试运行一下：运行一下main.py函数

在terminal中 cd到 `summerCourse2021\course2\examples>` 路径下

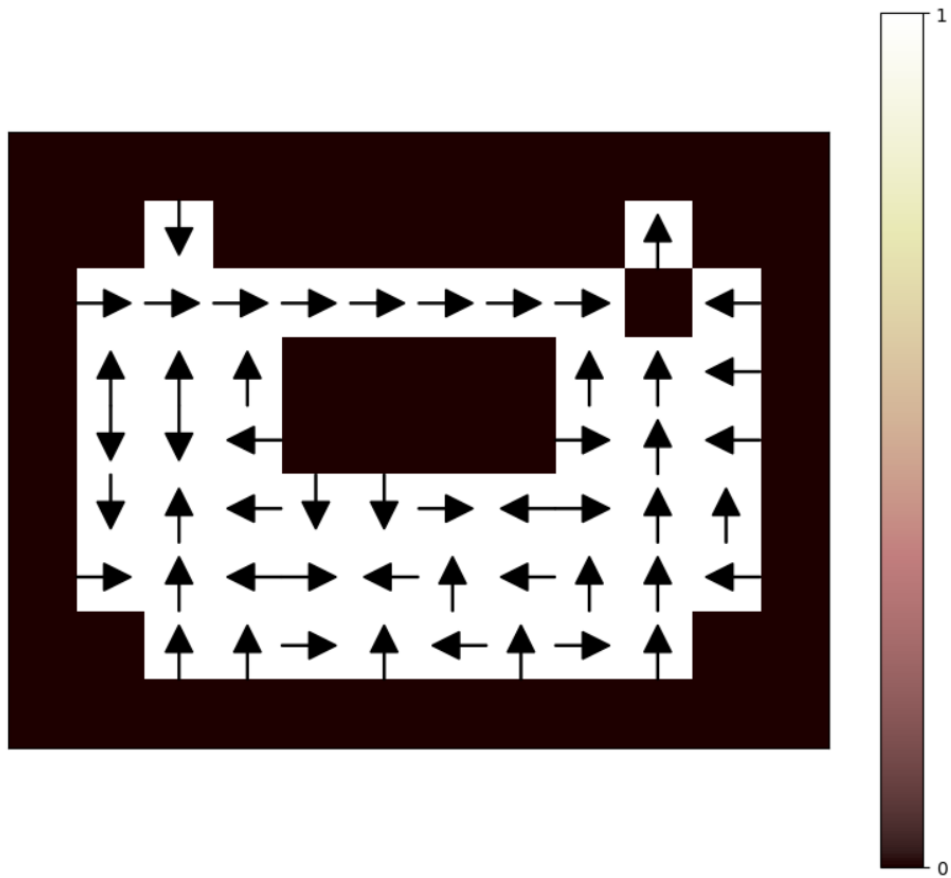
执行：

```
1 | python main.py
```

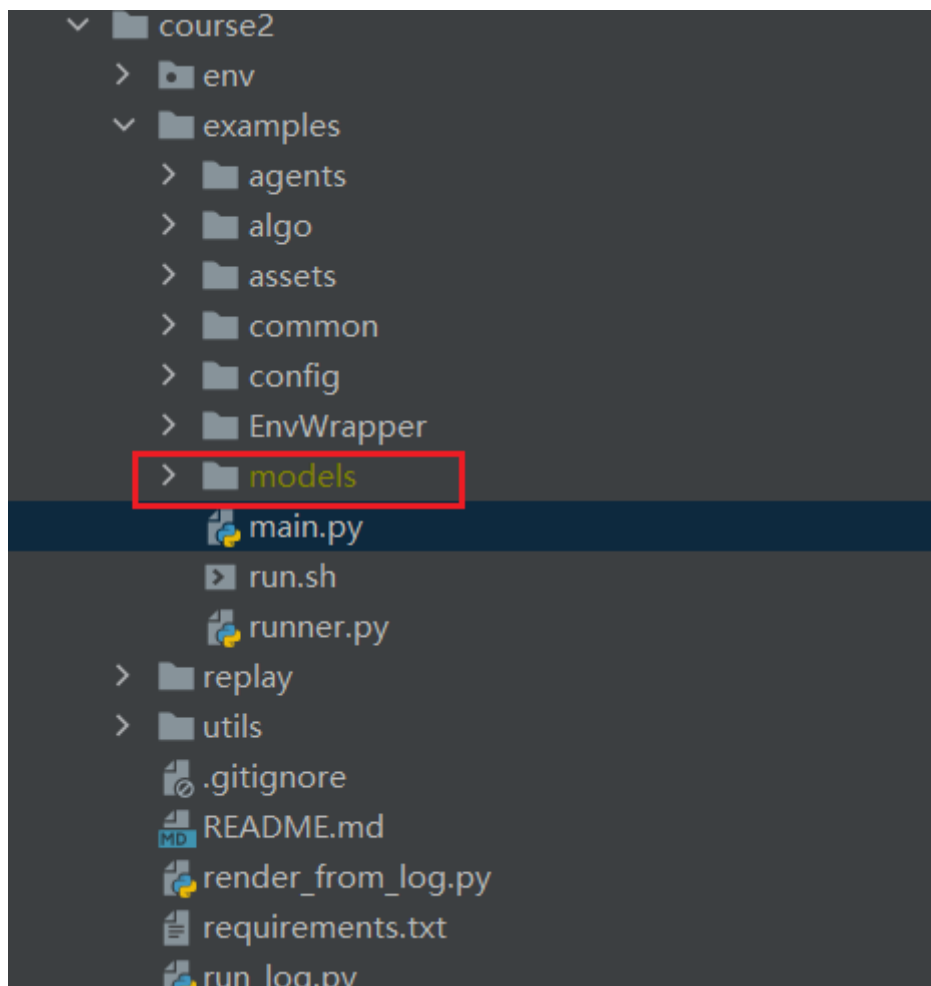
然后可以看到下面的打印信息，并且输出一张训练的结果图

如果看到这些信息，说明自己的环境已经Ok，准备工作已经ok,可以开始搞事情了

```
(D:\software\install\AI\anaconda\envs\snake1v1) F:\2 AI\RLchina\SummerCourse2021\course2\examples>python main.py
===== args: Namespace(algo='tabularq', reload_config=False, scenario='gridworld')
== args.reload_config: False
i_epoch: 1 Gt: -808.00
i_epoch: 2 Gt: -1174.00
i_epoch: 3 Gt: -1182.00
i_epoch: 4 Gt: -1728.00
i_epoch: 5 Gt: -254.00
```



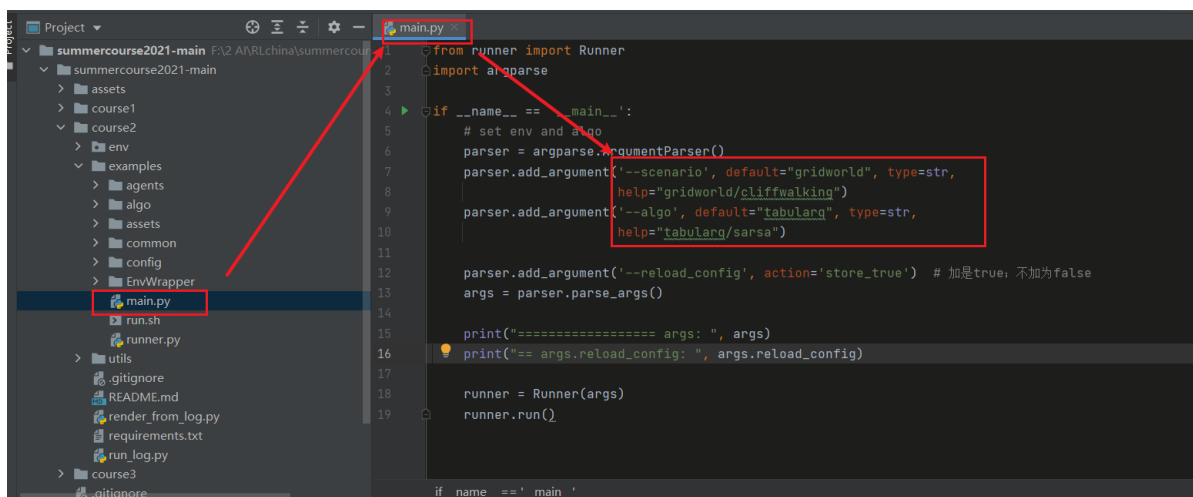
细心的同学可以发现，此时我们的工程里多了一个model文件



依次点开，文件的名字分别含义如下



其实它的参数是main.py 的默认运行参数，定义在这里



第三步：在cliffwalking环境，训练tabularq (Q-learning)

知道了参数在哪，我们可以粗暴直接一点，直接修改代码

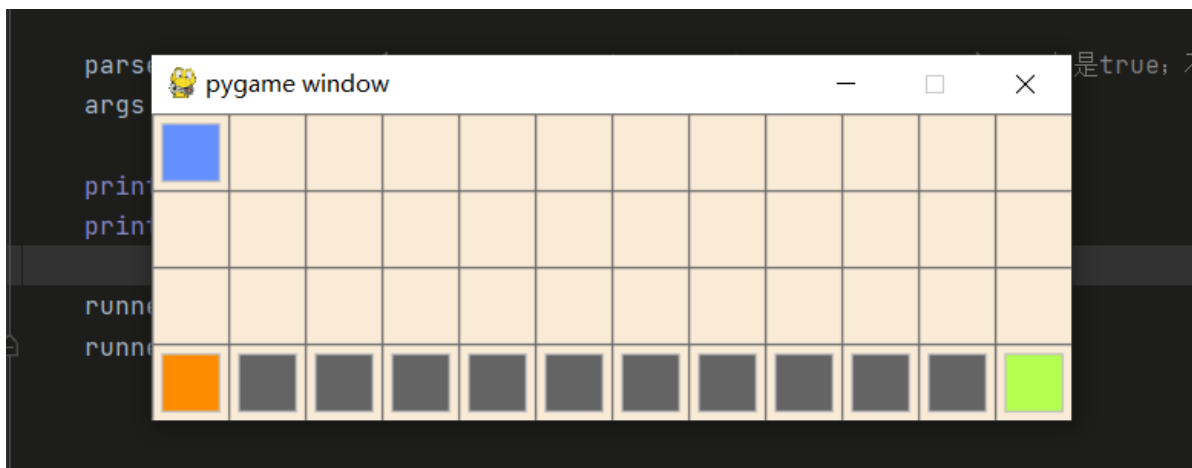
把这里修改成 cliffwalking

```
1 from runner import Runner
2 import argparse
3
4 if __name__ == '__main__':
5     # set env and algo
6     parser = argparse.ArgumentParser()
7     parser.add_argument('--scenario', default="cliffwalking", type=str,
8                         help="gridworld/cliffwalking")
9     parser.add_argument('--algo', default="tabularq", type=str,
10                        help="tabularq/sarsa")
11
12     parser.add_argument('--reload_config', action='store_true') # 加是true; 不加为false
13     args = parser.parse_args()
14
15     print("===== args: ", args)
16     print("== args.reload_config: ", args.reload_config)
17
18     runner = Runner(args)
19     runner.run()
```

然后

```
1 python main.py
```

可以看到可视化图像一顿操作猛如虎。刷刷进行了1000轮的尝试训练



可以看到它最后收敛到了-13的地方

```
i_epoch: 993 Gt: -13.00
i_epoch: 994 Gt: -13.00
i_epoch: 995 Gt: -13.00
i_epoch: 996 Gt: -13.00
i_epoch: 997 Gt: -13.00
i_epoch: 998 Gt: -13.00
i_epoch: 999 Gt: -13.00
i_epoch: 1000 Gt: -13.00
```

为啥13步呢，因为你数一下，从起点到终点，至少也要13步，走一步-1。最少也要-13.所哟这是最好的结果了。

贴着悬崖走，很冒进(验证了留作业的小可爱这句话)

✈ Bonus

gridworld和cliffwalking都是网格环境，智能体tabularq依然是“冒险家”，sarsa还是“保险主义”。运行试试吧^0^

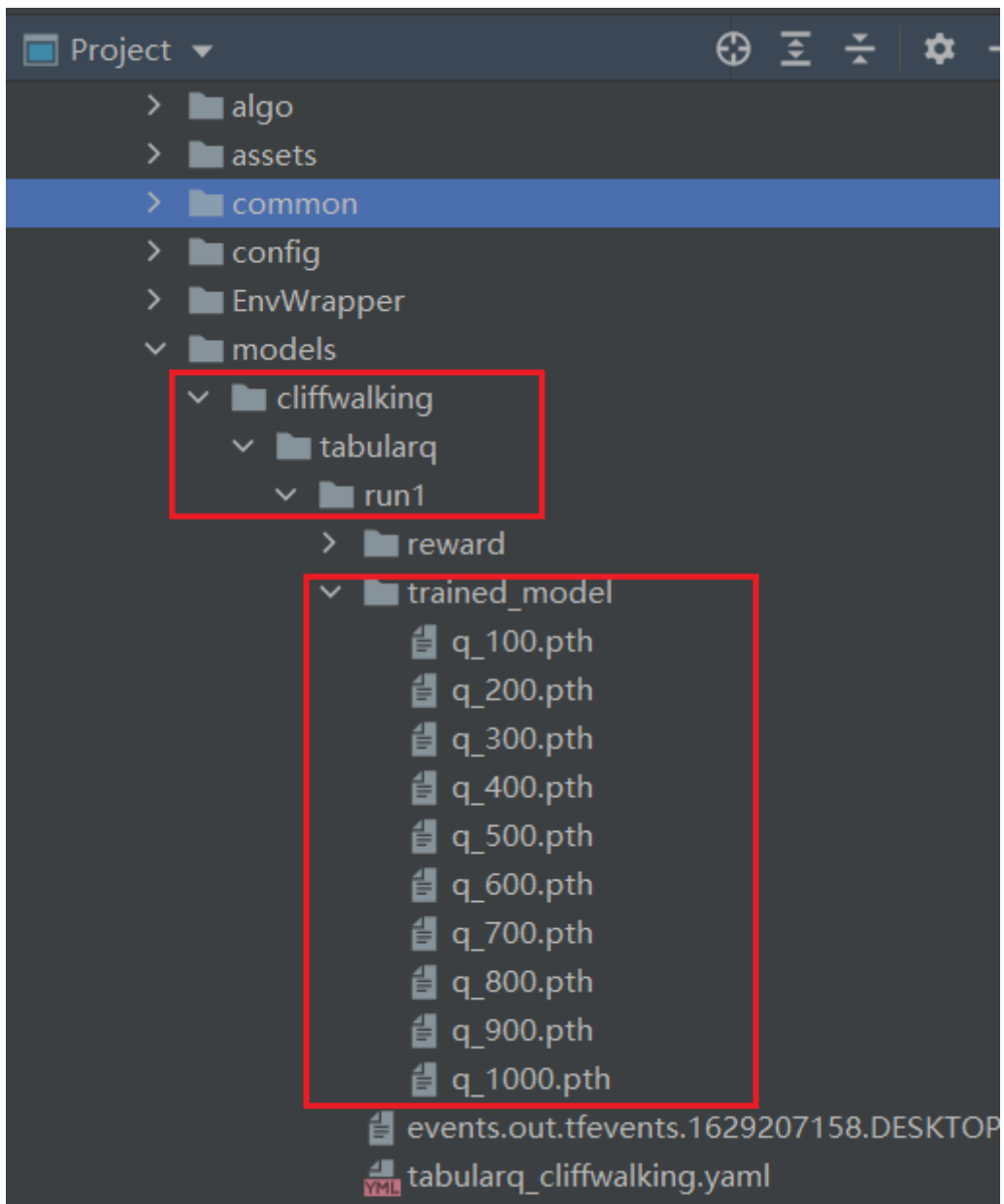
从训练结果也可以看到

```
i_epoch: 969 Gt: -13.00
i_epoch: 970 Gt: -13.00
i_epoch: 971 Gt: -13.00
i_epoch: 972 Gt: -13.00
i_epoch: 973 Gt: -13.00
i_epoch: 974 Gt: -108.00
i_epoch: 975 Gt: -13.00
i_epoch: 976 Gt: -13.00
i_epoch: 977 Gt: -13.00
i_epoch: 978 Gt: -13.00
```

这么冒进，一不小心就踏入了惩罚100的悬崖

啊 跑偏了 我们继续写作业。

这个时候可以看到 model里多了套文件。其中pth就是我们的模型。我们选择最后一个(我不确定是不是最好的)。提交到及第平台即可



补充(看不看都可): 当然也可以优雅一点, 从命令行做调整, 在终端指定算法和环境

```
1 | python main.py --scenario cliffwalking --algo tabularq
```

大家有看到官方教程中是这样的

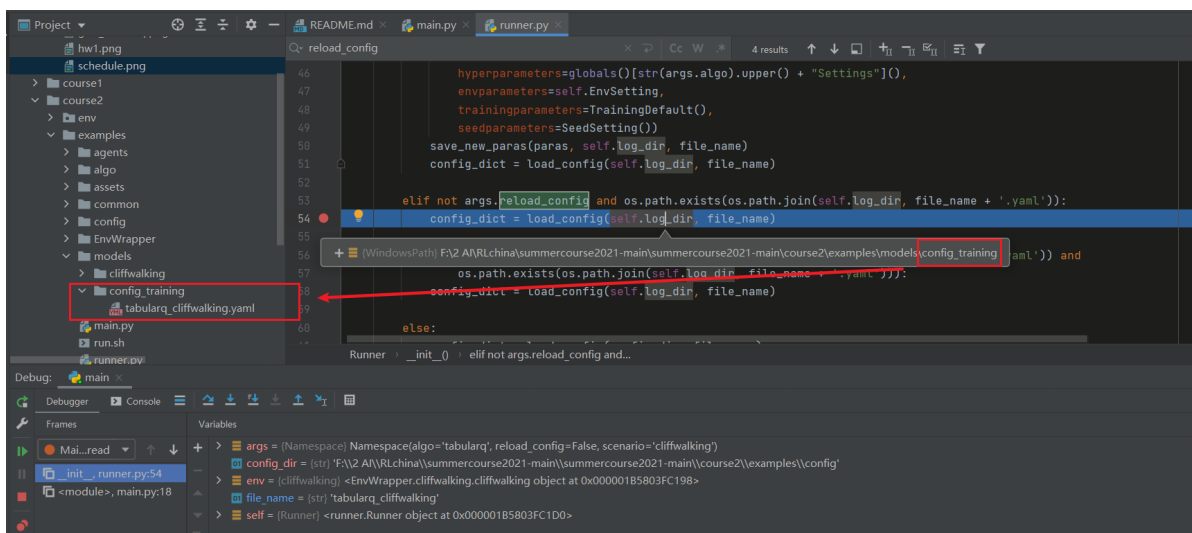
```
1 | python main.py --scenario cliffwalking --algo tabularq --reload_config
```

多了一个参数重加载 `--reload_config`, 这是啥作用呢

定位到它的用处()


```
34
35     self.run_dir, self.log_dir = make_logpath(args.scenario, args.algo)
36     self.writer = SummaryWriter(str(self.run_dir))
37
38     config_dir = os.path.join(os.getcwd(), "config")
39     file_name = args.algo + "_" + args.scenario
40
41     if (not args.reload_config and not os.path.exists(os.path.join(self.log_dir, file_name + '.yaml'))) \
42         or (args.reload_config and not os.path.exists(os.path.join(config_dir, file_name + '.yaml'))) and
43         not os.path.exists(os.path.join(self.log_dir, file_name + '.yaml')):
44         paras = TrainerSettings(
45             algo=args.algo,
46             hyperparameters=globals()[str(args.algo).upper() + "Settings"](),
47             envparameters=self.EnvSetting,
48             trainingparameters=TrainingDefault(),
49             seedparameters=SeedSetting())
50         save_new_paras(paras, self.log_dir, file_name)
51         config_dict = load_config(self.log_dir, file_name)
52
53     elif not args.reload_config and os.path.exists(os.path.join(self.log_dir, file_name + '.yaml')):
54         config_dict = load_config(self.log_dir, file_name)
55
56     elif (args.reload_config and not os.path.exists(os.path.join(config_dir, file_name + '.yaml'))) and
57         os.path.exists(os.path.join(self.log_dir, file_name + '.yaml')):
58         config_dict = load_config(self.log_dir, file_name)
59
60     else:
61         config_dict = load_config(config_dir, file_name)
```

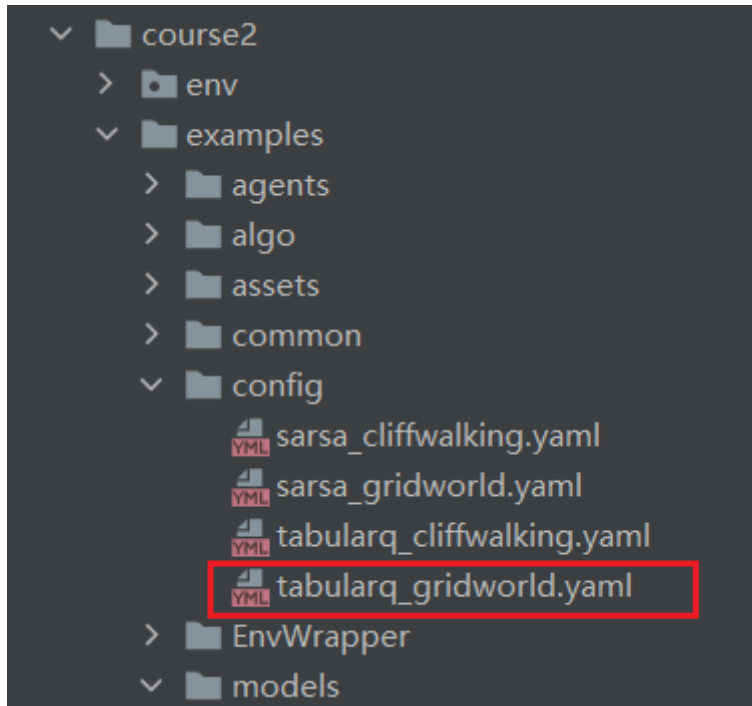
看着乱乱的，直接打个断点 debug



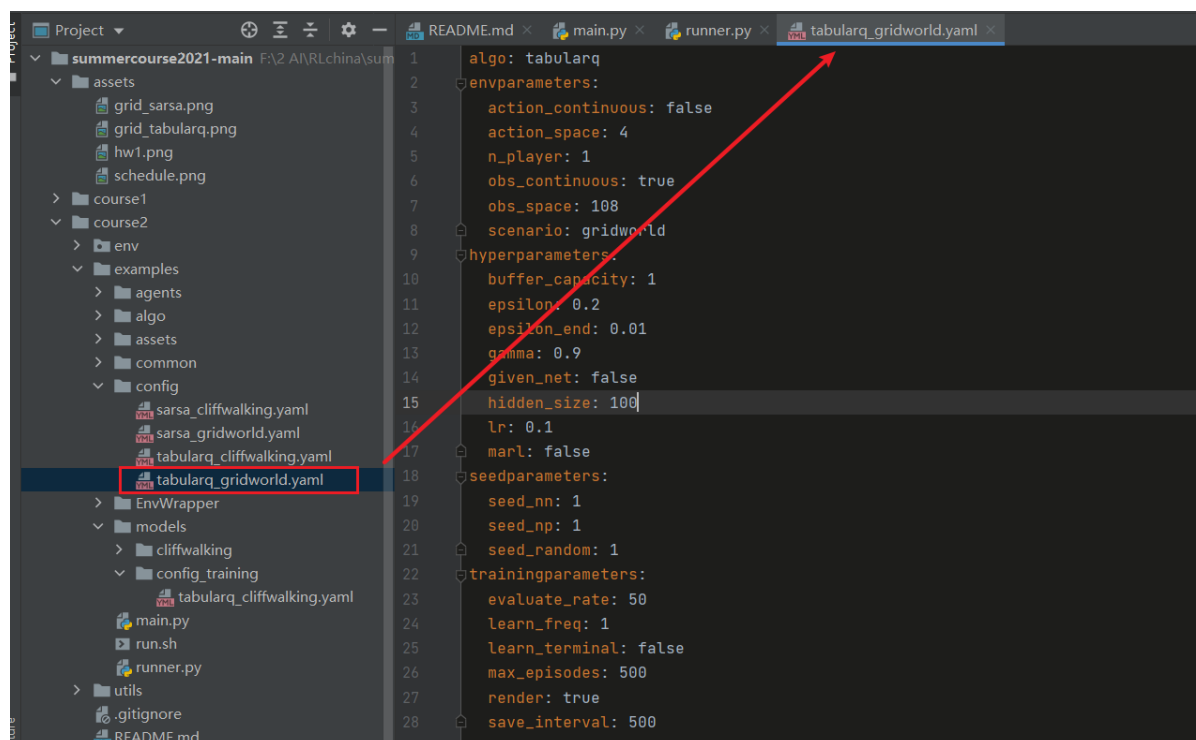
可以看到，

- 不添加 `--reload_config` 这个参数，它就会默认读取model/config_training下的参数

- 添加 `--reload_config` 这个参数，它就会读取config文件下的参数



有啥用呢？我们打开yaml文件看看



知道了数据的加载方式，我们在后期调参的时候，就可以在这个文件里直接修改参数，不用去代码中修改。改乱了还不知道原参数是啥了 (/(T o T)/~~)

那为啥搞俩文件呢？

我猜就是model/config_training下的参数可以反复的尝试

当有结果比较好的，或者稳定的。我们在config文件下备份一套。这样当后面在把参数搞得乱七八糟的时候，心里清晰的知道不管再乱

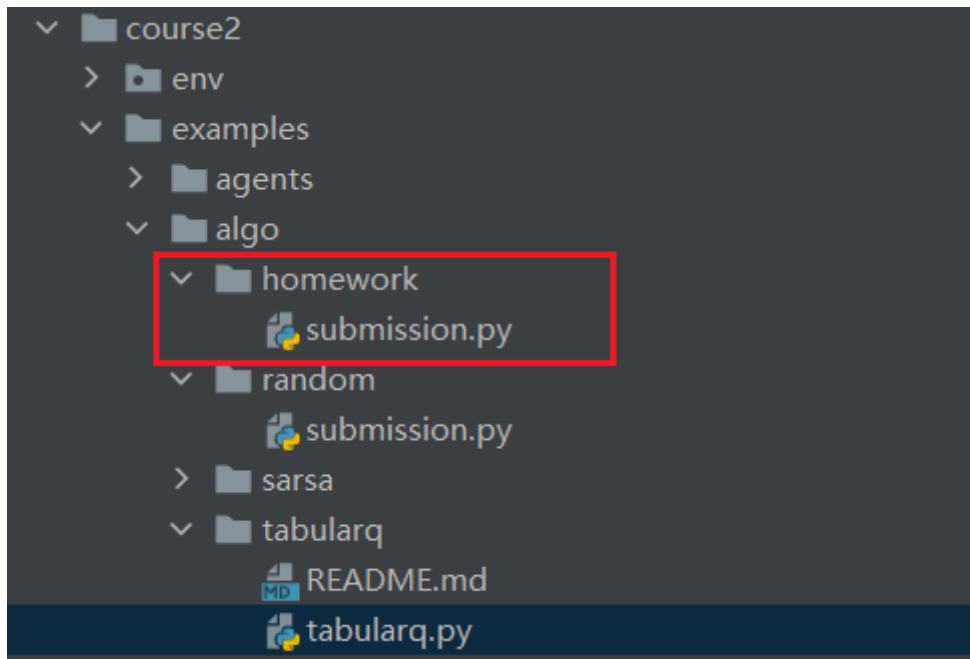
最后我们只要在终端增加一个 `--reload_config` 参数

我们就有一套比较棒的参数可以用~~

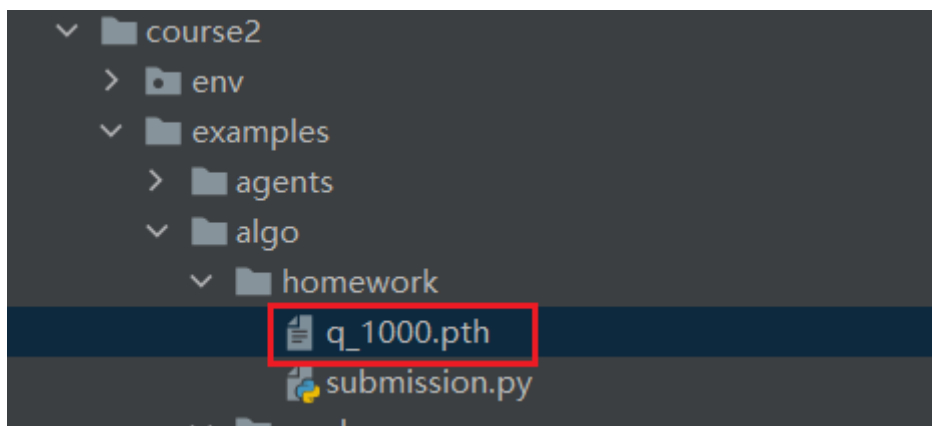
哔哔叨：点赞baseline的设计~ 考虑的很周全~ ♡

第四步：完成submission.py文件 提交代码

homework下的文件下已经给大家备好了基本的框架



首先我们先把训练好的模型复制到这个文件夹下，供submission.py调用



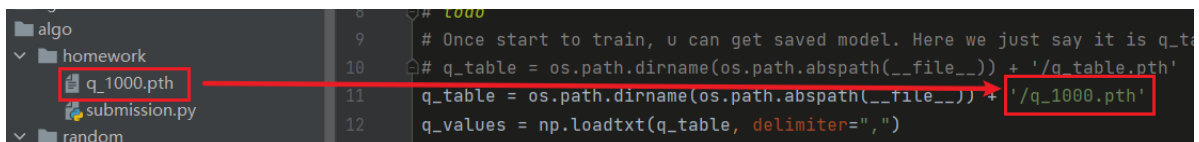
然后完成submission.py种需要自己写的函数

```
README.md × main.py × submission.py × tabularq.py ×
8 # todo
9 # Once start to train, u can get saved model. Here we just say it is q_table
10 q_table = os.path.dirname(os.path.abspath(__file__)) + '/q_table.pth'
11 q_values = np.loadtxt(q_table, delimiter=",")
12
13
14 def action_from_algo_to_env(joint_action):
15     joint_action_ = []
16     for a in range(1):
17         action_a = joint_action
18         each = [0] * 4
19         each[action_a] = 1
20         joint_action_.append(each)
21     return joint_action_
22
23
24 # todo
25 def behaviour_policy(q):
26     pass
27
28
29 # todo
30 def epsilon_greedy(q_values):
31     pass
32
33
34 # todo
35 def my_controller(observation, action_space, is_act_continuous=False):
36     obs = observation['obs']
37     pass
```

PS: 完成代码可以通过参考(实在有问题的话 直接划到文末去复制我的代码 理解一下也ok)

- tabularq.py中的函数
- random/submission.py中my_controller函数

然后别忘了修改导入模型处更改成你的模型名字




```
algo
├── homework
│   └── q_1000.pth
├── submission.py
└── random

8 # todo
9 # Once start to train, u can get saved model. Here we just say it is q_table
10 q_table = os.path.dirname(os.path.abspath(__file__)) + '/q_table.pth'
11 q_table = os.path.dirname(os.path.abspath(__file__)) + '/q_1000.pth'
12 q_values = np.loadtxt(q_table, delimiter=",")
```

然后上传应该不用说了，有问题可以参考我上一个文档

然后结果：

>	悬崖行走	q-learning	-10000.00	2021-08-17 23:16:36	通过		
---	------	------------	-----------	---------------------	----	---	---

分数是-10000是还没开始测评哈 大家莫急

说在最后:

Ok, 这就是一个简单的流程了, 大家可以尝试去更换一个训练算法。或者换个环境玩一玩

训练结束后, 更希望大家能去理一理代码, 和算法公式对应一下。更加直观的感受一下从理论到实现的过程。【如果需要讲解这个流程的话 可以去我公众号后台push我 哈哈】

如果对算法有疑问的话, 建议回看下8.17日(今天)上午两位老师的回放, 真的是通俗易懂 i了i了

还有其他问题的话, 为了使更多同学受益, 也为了防止同样的问题被多次问到, 大家可以上及第的**论道板块**进行提问。后面我会多关注这个板块, 在这里和大家多多讨论

[论道 - 及第]: <http://www.jidi.ai/forum>

啊 谈话间已经OK了

>	悬崖行走	q-learning	-13.00	2021-08-17 23:16:36	通过	山	出
---	------	------------	--------	---------------------	----	---	---

这样代码大家可以放心参考:

作者: HandsomeWu(公众号同步)



也欢迎关注公众号: **RLCN** 在后台提问:



代码附录: submission.py

```
1  ## This is homework.
2  ## Load your model and submit this to jidi
3
4
5  import numpy as np
6  import os
7
8  # todo
9  # Once start to train, u can get saved model. Here we just say it is
   q_table.pth.
10 # q_table = os.path.dirname(os.path.abspath(__file__)) + '/q_table.pth'
11 q_table = os.path.dirname(os.path.abspath(__file__)) + '/q_1000.pth'
12 q_values = np.loadtxt(q_table, delimiter=",")
13
14
15 def action_from_algo_to_env(joint_action):
16     joint_action_ = []
17     for a in range(1):
18         action_a = joint_action
19         each = [0] * 4
20         each[action_a] = 1
21         joint_action_.append(each)
22     return joint_action_
23
24
25 # todo
26 def behaviour_policy(q):
27     # pass
28     # eps = max(eps_end, eps - eps_delay)
29     return epsilon_greedy(q)
30
31 # todo
32 def epsilon_greedy(q_values):
33     # pass
34     return np.argmax(q_values)
35
36 # todo
37 def my_controller(observation, action_space, is_act_continuous=False):
38     obs = observation['obs']
39     # pass
40     action = behaviour_policy(q_values[obs, :])
41     action_onehot = action_from_algo_to_env(action)
42     return action_onehot
```