RE: [FV] Code example of running an experiment in a 1D grid world environment using my RL library

发件人 : Daniel Mastropietro <daniel.mastropietro@gmail.com> 2022年6月23日, 周四, 下午 01:49
主题 : RE: [FV] Code example of running an experiment in a 1D grid world environment using my RL library
收件人 : mingliang yao <mingliang.yao@etu.inp-n7.fr>

> I think the modification from my side would not influence anything in your side as long as it’s not pushed.

Yes that’s true, but the idea is to integrate all your useful modifications to the repository :-)
For instance the Fleming-Viot implementation on the labyrinth environment should definitely be integrated.

From: mingliang.yao@etu.inp-n7.fr [mailto:mingliang.yao@etu.inp-n7.fr]
Sent: Thursday 23 June 2022 13:27
To: Daniel Mastropietro
Subject: Re: [FV] Code example of running an experiment in a 1D grid world environment using my RL library

Hi Daniel,

Thank you for helping me on the environment installation and so on,
Thank you for the summary.

I think the modification from my side would not influence anything in your side as long as it’s not pushed.

Good afternoon
YAO

在 2022年6月23日，12:35，Daniel Mastropietro <daniel.mastropietro@gmail.com> 写道：

> Hello Yao,
>
> To summarize what we did today about the installation of required Python modules in your computer just now:
> **1) We did NOT install any virtual environment**
> Instead, we installed the required modules gym=0.12.1 and unittest_data_provider=1.0.1 (as defined in Python/lib/requirements.txt) by running `pip install` from a Anaconda prompt (run as an administrator) as follows:
> pip install gym==0.12.1
> pip install unittest_data_provider==1.0.1
>
> Note that `conda install` does NOT work for the installation of these modules, because the modules are NOT available at any conda repositories, but at the PyPI repository.
> However conda is not “compatible” with PyPI… (Ref: https://stackoverflow.com/questions/48138836/conda-cant-find-packages-at-pypi-org-simple)
>
> **2) We did NOT downgrade**
> We did not downgrade **numpy**, **pandas**, **pytest**, which are the other three modules mentioned in Python/lib/requirements.txt having a specific version, different from the one installed in the latest Anaconda distribution.
> Reason: downgrading was complicated because:
> - `conda install` did not find them in any repository, not even when specifying the `archive` repository (https://repo.anaconda.com/pkgs/archive), nor when specifying the option `-c conda-forge` suggested here: https://stackoverflow.com/questions/57609893/conda-downgrade-numpy-version

- `pip install` generated lots of errors (red messages in the Anaconda prompt console) which would have been difficult to solve.

This should be ok anyway. Hopefully the modifications you make to the code will not break on my side. I think this is unlikely. The problems I had in the past in terms of incompatible versions had to do with the **pandas** module, and I don't preview that you are going to use pandas too often.

Note: In case we wanted to check the oldest version available for a certain python version for any package we can use the bash script defined in the accepted answer here (which uses `conda search`):
https://stackoverflow.com/questions/34790725/conda-search-for-the-oldest-version-of-numpy-constrained-by-python-version

Cheers
Daniel

From: mingliang.yao@etu.inp-n7.fr [mailto:mingliang.yao@etu.inp-n7.fr]
Sent: Wednesday 22 June 2022 13:32
To: Daniel Mastropietro
Subject: Re: [FV] Code example of running an experiment in a 1D grid world environment using my RL library

Hi Daniel,

Thank you so much for sharing.

Yes I had installed the environment and cloned your GitHub Repo, I am now learning your code and try implement in 1D grid world environment.

Good afternoon
YAO

在 2022年6月22日，11:53，Daniel Mastropietro <daniel.mastropietro@gmail.com> 写道：

Hello Yao,

Assuming you have cloned by public GitHub repository available at https://github.com/mastropi/RL and following what we discussed yesterday, I share a **code example** of running an RL experiment using the RL library defined in the repo.

The example is taken from a unit test defined in:

Class	Method	Defined in
Test_MC_Lambda_1DGridworld	test_mc_random_walk_gamma_not_1()	test_MC_lambda.py

and does the following:

Description		Class	Method	Defined in (all files are given relative to the path `Python/lib`)
What it does?	Runs 20 episodes of an RL experiment	Simulator	run()	simulators.py
On what environment?	1D grid world	EnvGridworld1D	--	environments/gridworlds.py
With what policy?	Random walk	PolRandomWalkDiscrete	--	agents/policies/random_walks.py
With what learner?	Monte-Carlo	LeaMCLambda	--	agents/learners/episodic/discrete/mc.py
What is learned?	The state value function V(s)	LeaMCLambda	learn_pred_V()	agents/learners/episodic/discrete/mc.py

1) Details about the example

For your convenience, below I have copied the relevant excerpt of the aforementioned test file (test_MC_lambda.py), namely the setup class method and the method that runs the test, where I have highlighted, in this order:

- the creation of the environment
- the creation of the learner
- the creation of the agent which depends on the **learner** and on the **policy**.
- the creation of the simulator, which depends on the **environment** and on the **agent**.

From line 59:

```
@classmethod
def setUpClass(cls):    # cls is the class
                        # IMPORTANT: All attributes defined here can be then be referenced using self!
                        # (i.e. they belong to the "object" instantiated by this class)
    # Plot settings
    cls.max_rmse = 0.8
    cls.color_rmse = "blue"

    # Environment definition
    cls.nS = 19             # Number of non-terminal states in the 1D gridworld
    cls.env = gridworlds.EnvGridworld1D(length=cls.nS+2)  # nS states plus the two terminal states
    # True state value function when gamma = 1.0
    cls.V_true = np.arange(-cls.nS-1, cls.nS+2, 2) / (cls.nS+1)
    cls.V_true[0] = cls.V_true[-1] = 0

    # Random walk policy on the above environment
    cls.policy_rw = random_walks.PolRandomWalkDiscrete(cls.env)
```

From line 116:

```
def test_mc_random_walk_gamma_not_1(self):
    #-- All tests are run using seed = 1717, nepisodes = 20, start_state = 10
    print("\n*** Testing " + self.id() + " ***")

    # Learner and agent definition
    params = dict({'alpha': 1.0,
                   'gamma': 0.7,
                   'alpha_min': 0.0,
                   })
    learner_mclambda = mc.LeaMCLambda(self.env, alpha=params['alpha'], gamma=params['gamma'],
                                      alpha_update_type=AlphaUpdateType.FIRST_STATE_VISIT,  # First-visit is the default
                                      adjust_alpha=True, adjust_alpha_by_episode=False, alpha_min=params['alpha_min'],
                                      learner_type=mc.LearnerType.MC,
                                      debug=False)
    agent_rw_mclambda = agents.GenericAgent(self.policy_rw, learner_mclambda)

    # Simulation
    sim = simulators.Simulator(self.env, agent_rw_mclambda, debug=False)
    _, _, RMSE_by_episode, MAPE_by_episode, learning_info = \
                                        sim.run(nepisodes=self.nepisodes, start=self.start_state, seed=self.seed,
                                                compute_rmse=True, state_observe=10,
                                                verbose=True, verbose_period=100,
                                                plot=False, pause=0.1)

    expected = np.array([ 0.000000, -0.496388, -0.197257, -0.077399, -0.043423, -0.016056, -0.000258,
                         -0.000125, -0.000079, -0.000045,  0.000455,  0.000814,  0.001235,  0.001883,
                          0.006338,  0.009091,  0.014105,  0.035182,  0.137579,  0.640122,  0.000000])
    observed = agent_rw_mclambda.getLearner().getV().getValues()
    print("\nobserved: " + test_utils.array2str(observed))
    if self.plot:
        # Add the lambda parameter in the params dictionary which is required by plot_results()
        params['lambda'] = 1.0
        self.plot_results(params,
                          observed, self.V_true, RMSE_by_episode, learning_info['alpha_mean'],
                          max_rmse=self.max_rmse, color_rmse=self.color_rmse)
    assert np.allclose(observed, expected, atol=1E-6)
```

2) How to run the example

Follow the steps described in the following table:

Step	Type	Description	Recommended method	Details	Comments
1	One-time system	Install Python 3 (I think	Install a **Python**	Go to www.anaconda.com and download the desired Anaconda	On my side I have installed an

	preparation step	3.9 is the latest version, I am using version 3.6.4 on Windows 10).	**distribution** which includes the most common packages used for computing and data analysis (e.g. numpy, pandas, etc.). I have **Anaconda** distribution installed. (https://www.anaconda.com)	version for your operating system.	*archived* version of Anaconda from 2021 because I had problems using the latest version at the time (12-Sep-2021). But you may want to install the currently latest available version and then use a virtual environment to install the necessary package versions.

2	One-time system preparation step	Install necessary modules with specified versions.	Create a **virtual environment** with the necessary modules and their versions.
			This way you don't interfere with overlapping modules already installed within the Anaconda version you installed.
				The necessary modules (e.g. gym, etc.) are listed in `Python/lib/requirements.txt` file in my repo.
				
				If you installed Python Anaconda, you should be able to create a virtual environment using the `conda env create` command from the **Anaconda prompt** (which is one of the programs coming with Anaconda –you can access it by searching for "Anaconda prompt" and running it).
				
				For an example of creating and using conda environments you can take a look at: https://stackoverflow.com/questions/48174935/conda-creating-a-virtual-environment
				
				IMPORTANT: Once you have created the environment, you need to **activate** it in order to use it. See details in the above link.
					1) In order to create the virtual environment, you may need to run the Anaconda prompt as an administrator… I am not sure.
					
					2) You may need to install the unittest_data_provider module *manually* using `pip install`. Note that this installation must happen inside the virtual environment, if you decided to go that way.
					
					3) Once you activate the virtual environment, one way of checking if all the modules listed in requirements.txt (only 5 modules are listed) with their respective versions are installed is by opening a Python session and running the following two lines which here are shown for the numpy module:
					`import numpy`
					`numpy.__version__`

3	To be run every time you want to run the example	Run the test using the `unittest` module.	Write a script, import the necessary modules, create a unit test suite, and run the test suite.	Create a Python script with the following content, save it and run it (I tested successfully on my side):
				
				`import unittest`
				
				`from test_MC_lambda import Test_MC_Lambda_1DGridworld`
				
				`if __name__ == "__main__":`
				`    # Create a test runner`
				`    runner = unittest.TextTestRunner()`
				
				`    # Create a test suite`
				`    test_suite = unittest.TestSuite()`
				
				`test_suite.addTest(Test_MC_Lambda_1DGridworld("test_lambda_return_random_walk_gamma_not_1"))`
				
				`    # Run the test suite`
				`    runner.run(test_suite)`
					I successfully tested the script from Spyder-3.2.6 and from PyCharm-2021.2.2
					
					As output from this run you should expect:
					- One successful assertion, by seeing the word "OK" shown in the console output.
					- Two graphs showing

If you have any questions or problems, let me know!

Cheers
Daniel