Self-guided Approximate Linear Programs

Last update: 12/13/2023

Topics: approximate dynamic programming, approximate linear programming, model-based reinforcement learning, random Fourier features, inventory management

What is this repository?

This repository hosts implementations of diverse algorithms discussed in the paper titled "Self-Guided Approximate Linear Programs: Randomized Multi-Shot Approximation of Discounted Cost Markov Decision Processes" authored by Parshan Pakiman, Selvaprabu Nadarajah, Negar Soheili, and Qihang Lin. The paper is available at Management Science.

The included algorithms comprise the Feature-based Approximate Linear Program (FALP), Self-guided FALP, Policy-guided FALP, and Least Squares Monte Carlo (LSM). Additionally, the repository incorporates two algorithms for computing the lower bound on optimal policy cost and determining optimality gaps: Information Relaxation and Duality, alongside a heuristic based on constraint violation learning.

To facilitate experimentation with these methods on specific Markov Decision Processes, this repository offers two applications: perishable inventory control and Bermudan options pricing. All results in the paper are based on these two applications.

How to use this repository?

The steps detailed below are designed for Mac OS 14.1.2 and Ubuntu 20.04 systems. Windows users are encouraged to create similar procedures accordingly.

1. Download the repository on your local system and extract the zip file.
2. Open *Terminal* on your machine.
3. Please check the version of your Python. For example, run the code below:

```
python3 --version
```

4. Please confirm that Python 3.10.6 is installed on your machine. There are various methods available for installing Python. We leave this step to the user's discretion.
5. Create a virtual environment called "ALP". For example, use the following code:

```
python -m venv ALP
```

6. Activate the ALP environment as follows

```
source ALP/bin/activate
```

7. We use pip to showcase how to use this package; however, users can utilize alternatives. If opting for pip, please use the following code to update it:

```
pip install --upgrade pip
```

7. Please install the following libraries on the ALP environment using pip:

    - numpy (python -m pip install numpy)
    - pandas (python -m pip install pandas)
    - scipy (python -m pip install scipy)
    - numba (python -m pip install numba)
    - multiprocessing (python -m pip install multiprocessing)
    - tqdm (python -m pip install tqdm)
    - emcee (python -m pip install emcee)
    - sampyl (python -m pip install sampyl)
    - importlib (python -m pip install importlib)
    - sampyl_mcmc (python -m pip install sampyl_mcmc)
    - nengo (python -m pip install nengo)
    - gurobipy (python -m pip install gurobipy)

8. This repository relies on Gurobi for solving large-scale linear programs. Please ensure the installation of the Gurobi and its license. If you're affiliated with academia, Gurobi offers a free academic license. For further details, visit this page.

9. Provided that the ALP Python environment and Gurobi are correctly installed, you can employ the following code to solve test instances of perishable inventory control application. Please run the following code:

```
    ./run_PIC.sh
```

10. The provided code solves instance number 1 using the FALP algorithm with 20 random Fourier features employing a uniform state-relevance distribution. You can find the specifications of instance number 1 under *MDP/PIC/Instances/instance_1.py*. To solve this instance using an alternate algorithm, you can modify the file *run_PIC.sh*. For instance, changing the *algo_name* from "FALP" to "SG-FALP" in this file and rerunning *run_PIC.sh* will display the output for the self-guided FALP algorithm applied to the instance number 1. A screenshot of the output of these algorithms is attached below:

```
Instance number:        1
Algorithm name:         FALP
Basis function:         fourier
State relevance:        uniform_non_adaptive
# inner updates:       0
Random basis seed:      111
Optimality threshold:   5.0%
|   # Basis |    # Constrs |        FALP Obj | ALP ConT | ALP SlvT |     Lower Bound |    LB RT |     Policy Cost |    UB RT |     Opt Gap (%) |    TOT RT |
|        20 |       200000 |          2132.2 |    0.166 |    0.086 |          2003.4 |    0.399 |          2081.6 |    0.543 |             3.0 |    1.206 |
Instance number:        1
Algorithm name:         SGFALP
Basis function:         fourier
State relevance:        uniform_non_adaptive
Random basis seed:      111
Optimality threshold:   5.0%
|   # Basis |    # Constrs |        FALP Obj |      SGFALP Obj | ALP ConT | ALP SlvT |     SG T |     Lower Bound |    LB RT |     Policy Cost |    UB RT |     Opt Gap (%) |    TOT RT |
|         5 |       200000 |          1796.7 |          1796.7 |    0.100 |    0.008 |    0.000 |          1711.2 |    0.229 |          2495.7 |    0.574 |            31.0 |    0.911 |
|        10 |       200000 |          1968.5 |          1968.5 |    0.108 |    0.017 |    0.059 |          1872.5 |    0.318 |          4364.8 |    0.566 |            57.0 |    1.979 |
|        15 |       200000 |          2121.2 |          2121.2 |    0.106 |    0.027 |    0.078 |          1994.0 |    0.375 |          2079.6 |    0.571 |             4.0 |    3.136 |
|        20 |       200000 |          2138.1 |          2137.7 |    0.111 |    0.078 |    0.137 |          2002.9 |    0.867 |          2056.4 |    0.578 |             2.0 |    4.908 |
```

11. To solve the test instance of Bermudan options pricing problem, please run the file named *run_BerOpt.sh*.

```
./run_BerOpt.sh
```

12. To work with different instances or employ alternative algorithms provided in this repository, users need to make adjustments in the following files: *run_PIC.sh* and *run_BerOpt.sh*.

Appendix

This package has been tested on a 2021 MacBook Pro featuring an M1 Pro CPU with 16 GB of memory, running Mac OS 14.1.2. Below is the list of Python package versions used during the testing of this repository.

Package	Version
autograd	1.6.2
emcee	3.1.4
future	0.18.3
gurobipy	11.0.0
importlib	1.0.4
llvmlite	0.41.1
nengo	4.0.0
numba	0.58.1
numpy	1.26.2
pandas	2.1.4
pip	23.3.1
python-dateutil	2.8.2
pytz	2023.3.post1

Package	Version
sampyl-mcmc	0.3
scipy	1.11.4
setuptools	63.2.0
six	1.16.0
tqdm	4.66.1
tzdata	2023.3