Readme はじめにお読みください

本ドキュメントは本研究を引き継ぐ方または使用される方に向け、本研究の内容やファイル構造・プログラムの解説を行うことを目的として作成されました。 本研究について各種ファイルなどの詳細はhttps://github.com/kimshow/Distillation_Gym_feedoptをご覧下さい。

Distillation-Gym-feedopt の概要

- これは Midgley のDistillation-Gymを基に作成されました。
- Distillation-Gym に対し蒸留塔のフィード位置を最適化する機能があります。
- 強化学習により化学工学における分離プロセス合成を自動的に行います。
- COCOシミュレーターと ChemSep(COCO インストール時に同時インストールされる)を用います。
- 実行には run_SAC.py ファイルを実行してください。
- 実行に際し、Colab や Jupyterlab 等のノートブックで行うことを推奨します。

Agent

強化学習エージェントは Soft-Actor-Critic(SAC)アルゴリズムを採用しています。Soft-Actor-Critic についてはhttps://arxiv.org/abs/1812.05905を参照してください。

SAC_Agent フォルダの Agent.py がこのエージェントを記述しています。

Environment

強化学習環境は COCO シミュレーターによって行われます。 COCO のシミュレーターファイルを環境として扱い、この各ファイルの設定は Env フォルダの STANDARD_CONFIG.py で行います。既存環境の設定変更や新規環境の設定追加を行います。そのほか必要なクラスの定義などは DC_class.py, DC_gym.py, ClassDefinition.py によって行われます。

Utilities

学習の結果作成されたプロセスを BFD として出力するファイル BFD_maker.py や、経験再生に使うファイル memory.py を格納しています。

引き継ぎ

ここからは引き継ぎについて説明します。

初期設定

以下の手順で必要なものをインストールしてください。

1. PC に Python 3.6.8 をインストール(PATH を通してください)
2. ローカルの適当なディレクトリを作って(今後の作業フォルダになる)、そのフォルダに移動(cd)し、下記コマンドを入力して python 3.6.8　と表示されれば ok

```
python -V
```

3. 次にそのまま下記コードを実行。ディレクトリに venv

```
python -m venv venv
```

4. 次に下記コマンドを実行, これで仮想環境が立ち上がる

```
./venv/Scripts/activate
```

5. pip のアップグレードをする

```
python -m pip install --upgrade pip
```

6. requirements.txt をそのフォルダに持ってきて従ってライブラリをインストール

```
pip install -r requirements.txt
```

7. NVIDIA のドライバー, CUDA toolkit, cuDNN をインストールする。

    - NVIDIA のドライバー: タスクマネージャーなどから使用している GPU の型番を調べ、それに応じたものを NVIDIA 公式サイトからインストールする
    - CUDA toolkit: tensorflow のバージョンに合ったものを使う。tensorflow2.1.0 を使うので、CUDA toolkit 10.1 を NVIDIA 公式サイトからインストール
    - cuDNN: CUDA toolkit のバージョンに会うものを使う CUDA toolkit 10.1 なら cuDNN 7.6.5 をインストール(NVIDIA アカウントの作成が必要)

8. COCO シミュレーターをインストールする

9. Git を始める

    - Git をインストール
    - Github のアカウントを作ってこのリポジトリを自分のアカウントに fork
    - 適当なローカルのディレクトリにそのリポジトリを clone

Distillation-Gym を実行してみる

1. ターミナルで仮想環境を作ったディレクトリに移動

2. 初期設定の 4 番と同じように仮想環境を起動

3. ターミナルで Jupyterlab を起動する

```
jupyter lab
```

4. ブラウザに jupyter lab が立ち上がるから、適当に.ipynb ファイルを作成して名前を付けて保存。

5. ここから Jupyter lab で行う その ipynb ファイルで、https://note.nkmk.me/python-import-module-search-path/ などを参考にしながら、下記コードで一時的に SAC(run_SAC.py があるフォルダ)の親フォルダ Distillation_Gym_fo のパスを通す。

```
%cd "作業ディレクトリのPATH"
from os.path import dirname, abspath
import sys
parent_dir = dirname(abspath("run_SAC.py")) #__file__は実行ファイル
     sys.path.append(parent_dir)
```

6. これによって各種ライブラリをエラーなく使えるようになる

7. 実行は下記コード

```
%time %run "./SAC/run_SAC.py"
```

8. 実行が終わると tensorboard でいろんなデータを確認できる

```
%load_ext tensorboard
%tensorboard --logdir ./logs
```

ファイル構造

distillation_gym_fo ├─Asynchronous_test：使わない

├─DDPG：使わない(別の手法) ├─Env：環境の設定 ├─Hard_Actor_Critic：使わない(別の手法) ├─logs：実行ログ ├─SAC：エージェントの設定 │ ├─__pycache__：気にするな │ ├─ BFDs：実行後に出てくるプロセスのブロックフロー図が格納されます。各図のファイルは、"./SAC/BFDs/"+self.description+dt_today.strftime("%m%d")"フォルダ内に格納されています。 │ │ ├─memory_data：agent.py で fill_memory を行うとこ

こに学習に使う用のデータが格納された memory が各プロセスの設定ごとに作成されます。 | | ├─CONFIG_n__rondom_memory.obj：fill_memory を行いリプレイバッファを作るたびにそれが記録されるメモリー(fill_memory のたびに更新されるので同じものを残したければ別途対応すること) | | ├─CONFIG_n__memory.obj：実行のたびに実行で得たデータを記録するメモリー(実行のたびに更新されるので同じものを残したければ別途対応すること) | | ├─CONFIG_n__なんか数字の羅列.obj：実行のたびに実行で得たデータを記録するメモリー(日付で記録してくれている) Utils：そのほかの設定 Deep_Reinforcement_Learning_for_Process_Synthesis.pdf：論文 readme.md：このファイル requirements.txt：ライブラリインストール用

Tips

各ファイルにコメントしています

もうやったこと

- Agent.py のハイパラ変える → 有意差なし(勾配法のハイパラは下手に触ると怖いから変えてない、Actor.py とかのニューラルネットのはやる価値はあるかも)

やるとよさそうなこと

- 単ガウスから混合ガウス方策などの表現力の高い方策への転換
- 反応器などの他の機器への対応
- 方策勾配法アルゴリズムの変更(Adam 以外にいいのがあるかも)
- 計算速度の改善(いい感じのインタプリタに変えるとか、GPT-4 に聞くとか? ChatGPT は入力を学習に使われるし正誤判定むずいんでやめた方がいいかも)

あとがき

コーディングは VSCode でやるのがおすすめ、"vscode 使い方"とか"vscode ショートカット"とか"vscode 拡張機能"とかいろいろ調べると便利な機能出てきます。 個人的には Ctrl+クリックでクラスとかの依存関係参照できるのすごく便利。

なんかあったら先輩経由で呼んでください、暇だったら手伝いに行きます。