README

This is the code for the article *Tree Search in DAG Space with Model-based Reinforcement Learning for Causal Discovery* by Victor-Alexandru Darvariu, Stephen Hailes and Mirco Musolesi, published in Proceedings of the Royal Society A. If you use this code, please consider citing our article.

If you run into any issues when using this code, contact Victor-Alexandru Darvariu at victord@robots.ox.ac.uk.

License

Setting up

The two components

Due to incompatibilities between Python dependencies, there are two versions of the CD-UCT algorithm corresponding to continuous (`causal-discovery`) and discrete (`causal-discovery-dv`) random variables respectively. They can be found in the corresponding subdirectories.

Prerequisites

Docker is used extensively for managing dependencies and making it substantially easier to reproduce the reported results. Please see this URL for how to obtain and install Docker.

This project was developed on a Mac with an ARM M2 chip and MacOS Ventura 14.4.1. It is, in principle, also compatible with Linux variants. It can, in principle, also be adapted to Windows through WSL. The remainder of this user guide defaults to Mac and mentions the necessary adjustments for running on Linux where relevant.

If running on a Mac, the following are required before starting:

- The commands and basic scripts that are provided use `bash`; so make sure they are ran within a bash shell (alternatively, switch the system default shell to bash).
- (M1/M2 only): specify the default platform `export DOCKER_DEFAULT_PLATFORM=linux/amd64` in e.g. your .bashrc.

Checking that setup has succeeded

This guide provides commands for reproducing the reported experiments. We highlight that they require a significant amount of compute (*years* of single-core CPU time) and cannot realistically be

reproduced quickly on a single desktop machine.

Several integration tests are provided that "smoke-test" the key parts of each of the 2 components described above. To check that the components were set up successfully, we recommend running these integration tests, which are substantially faster than the experiments themselves.

Configuration

Unzip the provided `code.zip` file in a directory, e.g., `/home/jane/git`.

Set the following environment variables e.g. in your `.bashrc`, adjusting paths and directories as needed.

```
# Source directories
export CD_SOURCE_DIR=/home/jane/git/causal-discovery
export CDDV_SOURCE_DIR=/home/jane/git/causal-discovery-dv

# Experiment data and results will be stored here.
export CD_EXPERIMENT_DATA_DIR=/home/jane/experiments/causal-discovery
```

Setting up the data

Underlying datasets for all experiments are provided. To set up this data, unzip the provided `datasets.zip` file to `$CD_EXPERIMENT_DATA_DIR`. Paths such as `$CD_EXPERIMENT_DATA_DIR/datasets/sachs`, etc. should now be accessible.

`causal-discovery` component: continuous variables

Managing the Docker container

Some scripts are provided for convenience. To build the containers (note, this may take a significant amount of time e.g. 2 hours, as some packages are built from source):

```
$CD_SOURCE_DIR/scripts/update_container.sh
```

To start them:

```
$CD_SOURCE_DIR/scripts/manage_container.sh up
```

To stop them:

```
$CD_SOURCE_DIR/scripts/manage_container.sh stop
```

Note that the `$CD_SOURCE_DIR/docker/base/Dockerfile` contains lines (L74-75) to fix the RL-BIC Tensorflow dependency for M1/M2 Apple chips. These can be removed if running on Linux.

Running tests

Run the following tests to check everything is set up correctly:

```
docker exec -it cd-manager /bin/bash -c "source activate cd-env && python -m pytest"
```

Running experiments

Relevant commands are provided below for reproducing all the experiments in the paper.

Note that these require a significant amount of compute (years of single-core CPU time) and cannot realistically be reproduced on a single desktop machine. However, many of the experiments are trivially parallelizable -- the task execution in the loop of `run_experiments.sh` script can be ran on multiple servers in parallel (such as a cluster) so that the experiments take hours instead.

A strategy for running on a single machine is to drastically reduce number of seeds, number of hyperparameters, number of simulations carried out by CD-UCT and Random Search, and number of nodes of the considered graphs. These can be adjusted in the `cdrl/evaluation/experiment_conditions.py`

```
# Primary experiments on sachs and syntren (Table 1)
$CD_SOURCE_DIR/scripts/recipes/primary_experiments.sh

# Budget experiments on sachs dataset (Figure 2)
$CD_SOURCE_DIR/scripts/recipes/budget_experiments.sh hyperopt
# [await completion]
$CD_SOURCE_DIR/scripts/recipes/budget_experiments.sh eval

# Timings of CD-UCT versus UCT with naive cycle checking (Figure 3)
$CD_SOURCE_DIR/scripts/recipes/timings_experiments.sh

# Experiment with n=50 synthetic graph (Table 4)
$CD_SOURCE_DIR/scripts/run_experiments.sh eval scaleup scaleup 1000 synth50qr

# Experiment with synthetic graphs with varying density / N (Fig 4)
$CD_SOURCE_DIR/scripts/recipes/synthetic_experiments.sh eval synthetic vardensity er 1000 density
$CD_SOURCE_DIR/scripts/recipes/synthetic_experiments.sh eval synthetic vardata er 1000 data
```

To reproduce the experiments in Tables 6 and 5 (the score function with equal variances), follow the recipe above for the primary experiments, replacing the score function in experiment_conditions.py to "BIC" instead of "BIC_different_var".

Generating plots and tables via Jupyter

The container runs a Jupyter service at `http://localhost:8888`. Run the notebooks under the `$CD_SOURCE_DIR/notebooks/paper` directory to produce the tables and figures based on the experiments ran from the instructions given above. These outputs will be stored under `$CD_EXPERIMENT_DATA_DIR/aggregate_cdrl/figures`.

`causal-discovery-dv` component: discrete variables

Setting up

Build and run the relevant container as before. There are no platform-specific dependencies that need to be adjusted.

```
# stop container for prior component if running
$CD_SOURCE_DIR/scripts/manage_container.sh stop

# build and start container
$CDDV_SOURCE_DIR/scripts/update_container.sh
$CDDV_SOURCE_DIR/scripts/manage_container.sh up
```

Running tests

Run the following tests to check everything is set up correctly:

```
docker exec -it cddv-manager /bin/bash -c "source activate cddv-env &&
python -m pytest"
```

Running experiments and plotting results

```
# Experiments with discrete variables (Table 2)
$CDDV_SOURCE_DIR/scripts/recipes/discretevars_experiments.sh
```

As before, navigate to the Jupyter service at `http://localhost:8888`, run the notebooks under `$CDDV_SOURCE_DIR/notebooks/paper`, retrieve the tables and figures from `$CD_EXPERIMENT_DATA_DIR/aggregate_cdrl/figures`.