sRCD

The sRCD repository hosts the modified version of RCD algorithm which allows feedback loops or cycles in the relational causal models. The RCD algorithm, developed by Maier et. al. is the state-of-the-art relational causal discovery algorithm. However, it is designed for acyclic relational causal models. Recently we showed that under certain assumptions and constraints RCD produces correct results even for relational causal models that contain feedback loops under σ-separation. This repository contains the code associated with the experimental evaluations in the paper.

We recommend the users to go through the original RCD documentation before getting started with sRCD. A fork of the original RCD repository can be found here. The rest of this documentation assumes the reader is fairly familiar with RCD.

Example

The *runOracleRCD.py* file contains an example with RCD running on cyclic relational causal model with σ-separation oracle. It is very similar to the example in the RCD repository. The biggest difference here is that we are using σ-separation oracle instead of d-separation.

Experiment

The experiments corresponding to the results presented in the paper are executed using a single source file named *experiment.py*. Here's the general command to run it:

```
python experiment.py -config <config_path> [<options>]
```

`<config_path>` refers to the configuration file of a specific experiment `[<options>]` are for optional arguments

The configuration files are stored in the configs/ directory. For example, `configs/sample.json` is sample config to try out

```
python experiment.py -config configs/sample.json
```

The optional arguments are:

-d refers to debug mode, **1** to turn on debug mode, **0** to turn off (default). Usage:

```
python experiment.py -config configs/sample.json -d 1
```

--o refers to the output directory (default: **`out/`**). Usage:

```
python experiment.py -config configs/sample.json -d 1 -o out/
```

---nop disables parallel run, it is useful for debugging specific cases. Usage:

```
python experiment.py -config configs/sample.json -d 1 --nop
```

Config

Let’s look at the sample config:

```
{
    "seed"      : 123,
    "algos"     : ["d-RCD"],
    "target"    : "num_feedback_loops",
    "num_trials": 5,
    "params" : {
        "num_entities"      :   2,
        "num_dependencies"  :   6,
        "num_feedback_loops":   [2],
        "hop_threshold"     :   2,
        "max_depth"         :   3
    }
}
```

Here are the descriptions for each of the keys:

seed -> seed for random generation
algos -> list of algorithms to compare
target -> parameter to vary (usually the x axis in the result plot)
num_trials -> number of trials
num_entities -> Number of entity types
num_dependencies -> number of dependencies
num_feedback_loops -> Number of feedback loops
Hop_threshold -> hop threshold of the model
max_depth -> maximum depth for considering separation sets

So, the sample.json config refers to an experiment setup where we are only running the vanilla RCD (d-RCD) for 5 trials on randomly generated relational causal models with 2 entity types and 6 dependencies consisting 2 feedback loops and hop threshold 2. We can

Model Generation

```
generateModel(schema, hopThreshold, numDependencies,
numFeedbackLoops, maxNumParents=None,
dependencies=None, randomPicker=random.sample)
    numFeedbackLoops: number of feedback loops
```

The high level idea of the model generation is following:

1. Generate an acyclic model based on the given parameters
2. Iterate for the number of feedback loops
    a. Randomly pick an existing dependency
    b. *Create a copy of the dependency and reverse it
    c. Add the reverse dependency in the model
3. Create a model with both the set of dependencies in step 1 and 2
4. **Validate based on size
5. ***Validate for relational acyclifcation

* There is a strong constraint to avoid creating models with invalid relational acyclification.

- The **`effects`** dictionary records the number of times an attribute acts as an effect of a relational dependency
- While creating the reverse dependency we only consider the dependencies for which the attribute of the corresponding effect variable has value at most 1 in the **`effects`** dictionary
- The reason is that when a node is part of a cycle and there are multiple incoming edges to it then it increases the chance that the relational acyclification would require a higher hop threshold than the model. We want to avoid that.

** The function `validAggSize(schema, model, hopThreshold)` filters out models for which the true sigma-AGG contains more than 30 nodes

*** The function `hasValidRelAcyclifications(schema, model)` filters out models that have invalid relational acyclifications

Evaluation

A brief overview of the evaluation criteria is given in the paper. The evaluation is based on two parental queries: `isPossibleAncestor` and `isPossibleCycle`. The following shows example calls to run those queries:

```
p, r, f1 = ModelEvaluation.parentalQuery(trueAggs, learnedAggs,
isPossibleAncestor)
```

`trueAggs:` dictionary holding the *true* AGGs from different perspectives

`learnedAggs:` dictionary holding the *learned* AGGs from different perspectives

`isPossibleAncestor`: a function that performs the parental query

`p, r, f1 =` precision, recall, f1 score

Output

The output is generally a csv file containing the query results. The file name convention is following:

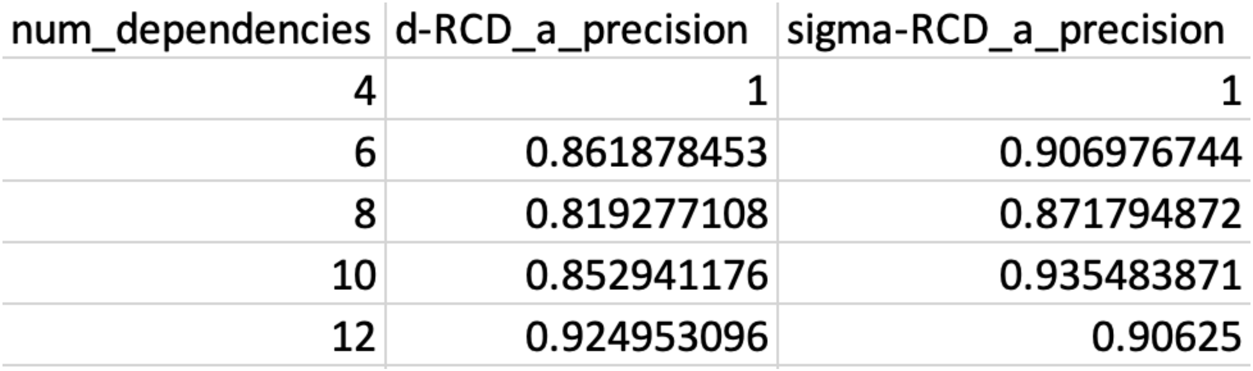
<config_name>_<query_initial>_<eval_metric>.csv

For example, consider the following output file name:

deps1_a_precision.csv
It refers to the config file *deps1.json* and it reports the *precision* of *isPossibleAncestor* (initial: ‘a’) query

Similarly, deps1_f_recall.csv refers to the config file *deps1.json* and it reports the *recall* of *isPossibleCycle* (initial: ‘f’) query

The results look like this:



num_dependencies	d-RCD_a_precision	sigma-RCD_a_precision
4	1	1
6	0.861878453	0.906976744
8	0.819277108	0.871794872
10	0.852941176	0.935483871
12	0.924953096	0.90625

Rows corresponding to increased number of dependencies and columns to algorithm-metric pair

Plots

The plots shown in the paper are generated using the output csv files. The specific method for plot generation is given in **`notebook/plots.ipynb`**