High-Order Tomography

Kevin D. Smith

Apr 04, 2022

CONTENTS:

Network tomography is a family of techniques for inferring information about the structure and internal state of a network from end-to-end measurements. Classically, tomography has been used to estimate properties of links, such as their average delay or packet loss rate, assuming that the routing matrix is known. But routing matrices are becoming increasingly difficult to identify. The traditional approach of using traceroutes is becoming less effective, as more and more routers are configured to ignore traceroute probes. Without access to the routing matrix, most applications of network tomography become impossible.

Fortunately, network tomography has come to its own rescue in recent work, as techniques have been developed to infer the routing matrix itself from end-to-end measurements. Tomography does not require internal routers to cooperate with traceroutes, since the probes can be disguised as normal network traffic: only the statistics of delays, loss rates, etc. are relevant. In particular, second-order statistics known as *path sharing metrics* (such as covariances in delays) can be used to reconstruct multicast trees.

High-Order Tomography (HOT) aims to infer general routing matrices from end-to-end data, without making any assumptions about the routing behavior. In order to accomplish this, we use high-order statistics of end-to-end data, going beyond second-order path sharing metrics. The mathematics are detailed in our paper Topology Inference with Multivariate Cumulants: The Möbius Inference Algorithm, which is to appear in *IEEE/ACM Transactions on Networking*.

Finally, a word of warning: since this project is a prototype for research purposes, this code should not be used in a production setting. We make no guarantees regarding its accuracy.

CHAPTER

ONE

INSTALLATION

1.1 Requirements

HOT requires the following packages:

- NumPy
- SciPy
- CVXPY
- NetworkX
- PyMoments

1.2 Installation

We suggest installing HOT in a new environment. Using Anaconda, create and activate a new environment for HOT:

```
conda create --name hot python=3.8
conda activate hot
```

Navigate to your directory of choice, and clone the source:

```
git clone https://github.com/KevinDalySmith/high-order-tomography.git
```

Finally, install the package:

```
python -m pip install ./high-order-tomography/
```

CHAPTER

TWO

GETTING STARTED

2.1 Create a Synthetic Dataset

To begin, we will need some data. We can create synthetic measurements of path delays using the *generate* command. Creating this dataset first needs an underlying network, with nodes representing routers and edges corresponding to links. In the `/high-order-tomography/data/rocketfuel/` directory, we have provided several real-world networks from the Rocketfuel database, stored as edgelists. Let's create a scenario based on the AS2914 topology.

Navigate to the `high-order-tomography` root directory, and run the following:

```
$ hot generate \
    ./data/rocketfuel/AS2914.txt \
    ./data/generated/AS2914samples.csv \
    ./data/generated/AS2914links.json \
    --monitors 5 --samples 100000
```

Out:

```
Constructing graph from ./data/rocketfuel/AS2914.txt...
Selecting monitor paths...
Sampling delays...
Writing delay data to ./data/generated/AS2914samples.csv...
Writing routing matrix to ./data/generated/AS2914links.json...
Done! Sampled 100000 delays from 10 monitor paths traversing 18 logical links.
```

In this scenario, we have randomly designated 5 leaves from the AS2914 network as "monitor nodes". For each of the 10 pairs of monitor nodes, we have selected a between these nodes as a "monitor path". The monitor paths do not span the entire network; rather, as the output of the `generate` command indicates, these 10 monitor paths only utilize 18 logical links, leading to a 10x18 routing matrix. The contents of this ground-truth routing matrix are contained in the newly created file `./data/rocketfuel/AS2914links.json`:

```
[
    [0, 2, 6, 7],
    [2],
    [4, 6],
    [0, 1, 3, 8],
    [9],
    [0, 6, 7],
    [2, 4, 5],
    [2, 3, 4, 5],
    [8, 7],
```

(continues on next page)

(continued from previous page)

```
    [3],
    [2, 4],
    [0, 1, 8],
    [1],
    [1, 4, 6, 9],
    [8, 9, 5, 7],
    [5],
    [4],
    [0, 8]
]
```

(Note the order of lists and of the numbers within them may be different, as these represent unordered sets.) Each list corresponds to a link, and the numbers indicate which paths utilize the link. For example, the set `[0, 2, 6, 7]` represents that the routing matrix contains a column where rows 0, 2, 6, and 7 have an entry of one, while all other entries are zero.

The second newly created file, `./data/rocketfuel/AS2914samples.csv`, is an array of path delay samples. Each column corresponds to a monitor path, and each row is a sample of the total delay along that path.

2.2 Infer the Routing Matrix

Next, we will try to use the data in `./data/rocketfuel/AS2914samples.csv` to try and reconstruct the routing matrix represented in `./data/rocketfuel/AS2914links.json`. From the `high-order-tomography` directory, run the following:

```
$ hot infer \
    ./data/generated/AS2914samples.csv \
    ./output/AS2914/ \
    --order 3 --alphas 1e-40 1e-30 --powers 0.95 --thresholds 0.85 \
    --max-size 4 --l1-weight 1.0 --l1-exponent 0.5 --n-groups 50
```

Out:

```
Estimating bounding topology...
Estimating common cumulants...: 100% 60/60 [00:29<00:00,  2.03it/s]
```

The first argument is the synthetic dataset we created in the previous section, and the second argument is the directory where the routing matrix estimate and auxiliary files will be written. The remaining settings are tunable parameters that are described in the command line documentation.

The directory `high-order-tomography/output/AS2914/` contains 5 new files. For the purposes of this tutorial, we are only interested in `predicted-links.json`. The file contains several attributes recording the parameters to reproduce this particular estimate, as well as the `predicted-links` attribute itself. The value of the `predicted-links` attribute is a list-of-lists encoding a routing matrix in the same format as `./data/rocketfuel/AS2914links.json`: each list is a link, containing the set of monitor paths that utilize the link.

2.3 Evaluating the Prediction

How accurate was the prediction? We can find out using the `evaluate` command:

```
$ hot evaluate ./output/AS2914/predicted-links.json ./data/generated/AS2914links.json
```

The first line of the output evaluates the bounding topology, i.e., estimate for which common cumulants are nonzero. The precision and recall indicate that the bounding topology estimate was 100% accurate; i.e., the algorithm was able to determine exactly which path sets correspond to nonzero common cumulants:

```
Bounding topology precision: 1.0000, recall: 1.0000
```

The next lines indicate the true positives. These are all of the columns of the routing matrix (represented by the list of paths corresponding to nonzero entries) that the algorithm correctly identified:

```
Found 17 true positives:
    {0, 2, 6, 7}
    {2}
    {4, 6}
    {0, 1, 3, 8}
    {9}
    {0, 6, 7}
    {2, 4, 5}
    {2, 3, 4, 5}
    {8, 7}
    {3}
    {0, 1, 8}
    {1}
    {1, 4, 6, 9}
    {8, 9, 5, 7}
    {5}
    {4}
    {0, 8}
```

The next lines indicate the false positives, i.e., columns of the routing matrix that the algorithm falsely detected:

```
Found 4 false positives:
    {7}
    {8}
    {0}
    {6}
```

The next lines are the columns of the routing matrix that were missed:

```
Missed 1 false negatives:
    {2, 4}
```

Finally, the last line reports the precision, recall, and F1 score of the routing matrix estimate:

```
Routing topology precision: 0.8095, recall: 0.9444, f1: 0.8718
```

CHAPTER

THREE

COMMAND LINE DOCUMENTATION

```
usage: hot [-h] {infer,generate,evaluate} ...
```

3.1 Sub-commands:

3.1.1 infer

Infer routing topology from additive path data.

```
hot infer [-h] [--order ORDER] [--alphas ALPHAS [ALPHAS ...]]
          [--powers POWERS [POWERS ...]]
          [--thresholds THRESHOLDS [THRESHOLDS ...]] [--n-groups N_GROUPS]
          [--resample-size RESAMPLE_SIZE] [--max-size MAX_SIZE]
          [--l1-weight L1_WEIGHT] [--l1-exponent L1_EXPONENT]
          [--nnz-threshold NNZ_THRESHOLD] [--solver_args SOLVER_ARGS]
          [--seed SEED]
          data_filename output_directory
```

Positional Arguments

data_filename	CSV file containing the path data sample.
output_directory	Directory where output files from the inference are saved.

Named Arguments

--order	Maximum cumulant order to evaluate. Default: 3
--alphas	Significance threhsolds for nonzero common cumulant tests at orders 2, 3, 4, . . . , <order>. E.g., "–alphas 1e-40 1e-30" accepts nonzero 2nd-order ccs to a significance of 1e-40 and 3rd-order ccs to a significance of 1e-30.
--powers	Statistical power estimates for the nonzero common cumulant tests at orders 3, 4, . . . , <order>. E.g., "–powers 0.95 0.9" estimates the test power to be 0.95 for 3rd-order ccs and 0.9 for 4th-order ccs. If a single float is provided, this power is used for all orders.

--thresholds	Robustness thresholds for tightening the bounding topology at orders 3, 4, ..., <order>. Similar to –powers. If a single float is provided, this threshold is used for all orders.
--n-groups	Number of resamples to generate by bootstrapping.
--resample-size	Size of each resample to generate by bootstrapping. If 0, the resample sizes are the same size as the original sample. Default: 0
--max-size	Largest acceptable size of a non-maximal path set.
--l1-weight	Weight for the lasso regularizer. Default: 0.1
--l1-exponent	Exponent for the lasso regularizer. Default: 0.0
--nnz-threshold	Threshold to decide if estimated exact cumulants are nonzero. Default: 0.001
--solver_args	JSON object of arguments for the CVXPY solver. See CVXPY documentation at https://www.cvxpy.org/tutorial/advanced/index.html#setting-solver-options Default: "{"verbose": false, "solver": "OSQP", "eps_abs": 1e-10, "eps_rel": 1e-10}"
--seed	Random seed. Default: 1234

3.1.2 generate

Generate synthetic path delay data.

```
hot generate [-h] [--monitors MONITORS] [--samples SAMPLES]
             [--delay-mean DELAY_MEAN] [--delay-std DELAY_STD]
             [--delay-scale DELAY_SCALE] [--seed SEED]
             edgelist_filename data_filename links_filename
```

Positional Arguments

edgelist_filename	Edge list file for the underlying network.
data_filename	Path for the output file of path delay samples.
links_filename	Path for the output file with the ground-truth links (as path sets).

Named Arguments

--monitors	Number of monitor nodes. Default: 5
--samples	Number of path delay samples. Default: 100000
--delay-mean	Average of mean delays for links. Default: 10.0
--delay-std	Std. dev. of mean delays for links. Default: 2.0
--delay-scale	Scale parameter for link delay gamma distributions. Default: 4.0
--seed	Random seed. Default: 1234

3.1.3 evaluate

Evaluate an estimated routing topology against ground truth.

```
hot evaluate [-h] estimate_filename true_filename
```

Positional Arguments

estimate_filename	JSON file containing the predicted-links (as path sets).
true_filename	JSON file containing the ground-truth links (as path sets).

CHAPTER

FOUR

LICENSE, CITATIONS, AND ACKNOWLEDGEMENTS

```
@article{KDS-SJ-FB-AS:22,
  author = {K. D. Smith and S. Jafarpour and A. Swami and F. Bullo},
  title = {Topology Inference with Multivariate Cumulants: {The} {M\"obius} Inference␣
↪Algorithm},
  journal = {IEEE/ACM Transactions on Networking},
  year = 2022,
  note = {To appear},
  url  {https://arxiv.org/pdf/2005.07880.pdf}
}
```

This work was supported by the U.S. Defense Threat Reduction Agency, under grant HDTRA1-19-1-0017.