v6-crosstab-py

B. van Beusekom, H. Alradhi, F.C. Martin

May 15, 2024

CONTENTS

CHAPTER

ONE

DESCRIPTION

This algorithm computes a cross-table a.ka. contingency table for two or more categorical variables. The algorithm takes categorical variables as input and returns a table with the counts of the number of occurrences of each combination of categories.

CHAPTER

TWO

AUTHORS

Bart van Beusekom, Frank Martin, Hasan Alradhi, *Netherlands Comphrensive Cancer Organisation (IKNL).*

CHAPTER

THREE

SOURCE CODE

Source code is available in the following GitHub repository.

CHAPTER

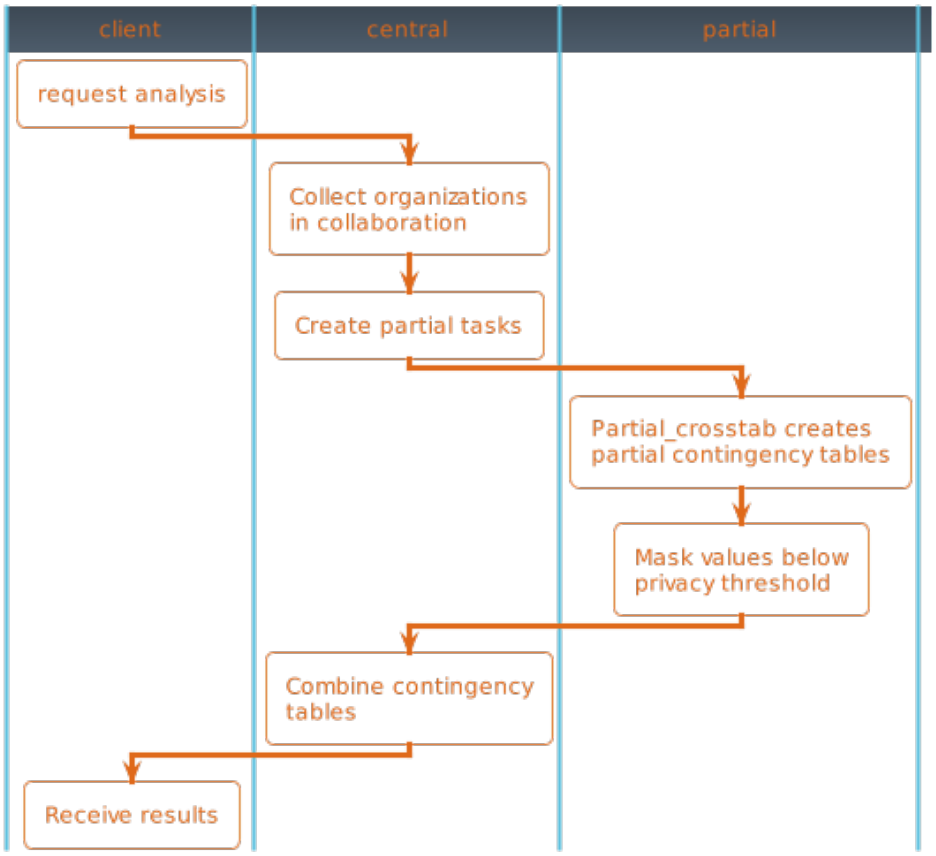
FOUR

CONTENTS

4.1 Implementation

4.1.1 Overview

The implementation is rather straightforward. The central part requests the partial contingency table from each node, which they compute in one go. The central part then aggregates the partial contingency tables to the final table.



The central part of the algorithm is responsible for the orchestration and aggregation of the algorithm. The partial parts are executed on each node.

4.1.2 Partials

Partials are the computations that are executed on each node. The partials have access to the data that is stored on the node. The partials are executed in parallel on each node.

`partial_crosstab`

The partial function computes the local contingency table. Any values below the privacy threshold are converted to a range. For example, if the privacy threshold is 5, then all values below 5 are converted to '0-4'. The local contingency table is sent to the central part.

The partial function includes several privacy checks - see the *privacy guards* section for more information.

4.1.3 Central

The central part is responsible for the aggregation of the cross-tables of individual nodes.

`central_crosstab`

The central part sums up the local contingency tables resulting in the global contingency table.

- If the local contingency table contains values below the privacy threshold, the central part will show a range instead of the actual value. For example, if the privacy threshold is 5, and the two nodes included in an analysis report back 0-4 and 11, the central part will show 11-15 as the value for the corresponding cell.
- If one node does not contain a cell that another node do have, the central part will simply count zero for the missing cell, i.e. the central part will simply show the sum of the values that *are* reported.

4.2 How to use

4.2.1 Input arguments

4.2.2 Python client example

To understand the information below, you should be familiar with the vantage6 framework. If you are not, please read the documentation first, especially the part about the Python client.

Let's say you want to know how many males and females are overweight in different age groups, e.g. something like:

AgeGroup	isOverweight	Male	Female
0-18	True	11	6
0-18	False	30	29
18-65	True	55	44
18-65	False	50	56
65+	True	5	10
65+	False	15	14

Such a result could be obtained by running the following Python client code. Note that `AgeGroup`, `isOverweight`, and `Gender` should be categorical values in your dataset, and that you should replace the values at the top to authenticate with your vantage6 server.

```python
from vantage6.client import Client

server = 'http://localhost'
port = 5000
api_path = '/api'
private_key = None
username = 'root'
password = 'password'

# Create connection with the vantage6 server
client = Client(server, port, api_path)
client.setup_encryption(private_key)
client.authenticate(username, password)

input_ = {
  'method': 'central_crosstab',
  'kwargs': {
      'results_col': 'Gender',
      'group_cols': ["AgeGroup", "isOverweight"]
  }
}

my_task = client.task.create(
    collaboration=1,
    organizations=[1],
    name='Compute contingency table',
    description='Create a contingency table showing the relationship between two or more␣
↪variables',
    image='harbor2.vantage6.ai/algorithms/v6-crosstab-py:latest',
    input=input_,
    databases=[
        {'label': 'default'}
    ]
)

task_id = my_task.get('id')
results = client.wait_for_results(task_id)
```

4.3 Privacy

4.3.1 Guards

There are several guards in place to protect sharing too much information on individual records:

- **Thresholding**: The system will only share information if there are at least *n* records in the group. This is to prevent sharing information on individual records. By default, the threshold is set to 5 records. Node administrators can change this threshold by adding the following to their node configuration file:

```yaml
algorithm_env:
  CROSSTAB_PRIVACY_THRESHOLD: 5
```

and setting the value to the desired threshold. This configuration will ensure that an environment variable *CROSSTAB_PRIVACY_THRESHOLD* is set to the desired threshold and passed to the algorithm container.

Note that the algorithm also requires at least one field of the contingency table to pass the threshold. This is to prevent that if a task is created for a column that contains only unique values, the result would reveal which unique values are present in the column.

- **Setting the allowed columns**: The node administrator can set on which columns they want to allow or disallow the computation of the contingency table by adding the following to the node configuration file:

```
algorithm_env:
  CROSSTAB_ALLOWED_COLUMNS: ["ageGroup", "isOverweight"]
  CROSSTAB_DISALLOWED_COLUMNS: ["age", "weight"]
```

This configuration will ensure that only the columns *ageGroup* and *isOverweight* are allowed to be used in the computation of the contingency table. The columns *age* and *weight* are disallowed and will not be used in the computation. Usually, there should either be an allowed or disallowed list, but not both: if there is an explicit allowed list, all other columns are automatically disallowed.

We recommend to define these list to ensure that the contingency table can only be computed for categorical tables, and not for numeric ones. The latter run a risk of revealing that certain values are present in the data. We chose to let the node administrator handle this as they know their own data best.

- **Minimum number of data rows to participate**: A node will only participate if it contains at least *n* data rows. This is to prevent nodes with very little data from participating in the computation. By default, the minimum number of data rows is set to 5. Node administrators can change this minimum by adding the following to their node configuration file:

```
algorithm_env:
  CROSSTAB_MINIMUM_ROWS_TOTAL: 5
```

- **Not allowing zero values**: By default, the system will not values of zero to be shared. In principle, it should be OK to share zero values, since this only confirms an absence of certain combinations of values. However, it may be possible to infer information from zero values. For example, for a rather sparse contingency table, the information which combinations exist at a certain data is more valuable than for a dense table. It is therefore possible not to share this information by not sharing zero values.

If you do wish to share zero values, you can add the following to your node configuration:

```
algorithm_env:
  CROSSTAB_ALLOW_ZERO: true
```

4.3.2 Data sharing

The only intermediate data that is shared, are the local contingency tables. These are formatted in the same way as the final result, but contain only the data from the local node. The risk of sharing this data is low, as it concerns aggregated data.

4.3.3 Vulnerabilities to known attacks

Attack	Risk eliminated?	Risk analysis
Reconstruction	✓	
Differencing		May be possible by making smart selection with preprocessing, or by sending multiple tasks before and after data is updated.
Deep Leakage from Gradients (DLG)	✓	
Generative Adversarial Networks (GAN)	✓	
Model Inversion	✓	
Watermark Attack	✓	

4.4 Validation

A test script is available in the *test* directory. It can be run with the following command:

```
python test/test.py
```

The script will run the crosstab algorithm via the vantage6 `MockAlgorithmClient`.

4.5 References

4.5.1 Cite this implementation

This particular algorithm has not been published yet. If you use this code in your research, please cite the following paper:

1. Moncada-Torres, Arturo, et al. "VANTAGE6: an open source priVAcy preserviNg federaTed leArninG infrastructurE for Secure Insight eXchange." *AMIA annual symposium proceedings*. Vol. 2020. American Medical Informatics Association, 2020. [link]

4.5.2 Used sources

This implementation has been inspired by earlier implementations of the same algorithm:

- https://github.com/IKNL/vantage6-algorithms/tree/crosstab
- https://github.com/IKNL/v6-starter-crosstabulation-py