dgim Documentation

Release 0.2.0

Simon Dollé

January 05, 2015

Contents

Contents:

CHAPTER 1

dgim

Python implementation of the DGIM algorithm: a compact datastructure to estimate the number of *True* statements in the last N elements of a boolean stream.

1.1 Features

- Estimation of the number of "True" statements in the last N element of a boolean stream
- Low memory footprint.
- Tunable error rate (the lower the error rate, the higher the memory footprint)

1.2 Applications

When processing large streams of data such as clicks streams, server logs, financial streams. It is often necessary to maintain statistics about the N latest elements. If N is big or if you have many streams to process, it is not possible to store the N latest elements.

In such situations, if the processed stream is made of boolean, the DGIM algorithm can help you estimate the number of *True* statements in the last elements.

For instance, if the stream is made of server logs, DGIM algorithm can estimate the proportion of visits that come from search engines. (as opposed to direct access, or access through paid search)

1.3 Installation

At the command line:

```
$ pip install dgim
```

1.4 Usage

Sample code:

```
from dgim import Dgim
dgim = Dgim(N=32, error_rate=0.1)
for i in range(100):
    dgim.update(True)
dgim_result = dgim.get_count() # 30 (exact result is 32)
```

1.5 Documentation

https://dgim.readthedocs.org.

1.6 License

The project is licensed under the BSD license.

1.7 Authors

- Simon Dollé <simon.dolle@gmail.com>
- No other contributor yet. Why not joining?

1.8 How to contribute

1. Check for open issues or open a fresh issue to start a discussion around a feature idea or a bug.
2. Fork the repository on GitHub to start making your changes to the **master** branch (or branch off of it).
3. Write a test which shows that the bug was fixed or that the feature works as expected.
4. Send a pull request and bug the maintainer until it gets merged and published. :) Make sure to add yourself to AUTHORS.

1.9 References

- Datar, Mayur, et al. "Maintaining stream statistics over sliding windows." SIAM Journal on Computing 31.6 (2002): 1794-1813.
- Rajaraman, Anand, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2011. Chapter 4. http://infolab.stanford.edu/~ullman/mmds/ch4.pdf
- Mining of Massive Datasets Coursera MOOC: http://infolab.stanford.edu/~ullman/mmds/ch4.pdf

CHAPTER 2

Code Documentation

class dgim.**Dgim**(*N*, *error_rate=0.5*)

An implementation of the DGIM algorithm. It estimates the number of "True" present the last N elements of a boolean stream. The datastructure it uses is very compact and has a memory complexity of O(log(N)^2).

The algorithm is described in: Datar, Mayur, et al. "Maintaining stream statistics over sliding windows." SIAM Journal on Computing 31.6 (2002): 1794-1813.

An explanation of the algorithm can also be found here: http://infolab.stanford.edu/~ullman/mmds/ch4.pdf

__init__(*N*, *error_rate=0.5*)

Constructor

Parameters

- **N** (*int*) – sliding window width. The algorithm will return an estimate of the number of "True" in the last N elements of the stream.
- **error_rate** – the maximum error made by the algorithm.

The error rate is in]0, 1] Let c be the true result and e the estimate returned by the dgim algorithm. abs(c-e) < error_rate * c The lower the error, the higher the object footprint. :type error_rate: float

get_count()

Returns an estimate of the number of "True" in the last N elements of the stream.

Return type int

update(*elt*)

Update the stream with one element.

Parameters **elt** (*bool*) – the latest element of the stream

CHAPTER 3

History

0.2.0 (2015-01-05)

- Improved documentation
- Make most methods and attribute private.

CHAPTER 5

0.1.0 (2015-01-04)

- First release on PyPI.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

Symbols

D

G

U