ProbStructs

Martin Majlis

Aug 07, 2020

CONTENTS:

ProbStructs library provides probabilistic structures to count elements in the stream.

CHAPTER

ONE

CLASSES

- *CountMinSketch* - frequency table of events in a stream
- *ExponentialHistorgram* - frequency of specific event in the last N elements from a stream
- *ExponentialCountMinSketch* - frequency table of events in the last N elements from a stream
- *Hash* - hashing function

CHAPTER

TWO

EXAMPLE

```
using namespace probstructs;

ExponentialCountMinSketch<int> sketch(100, 4, 8);

uint ts = 0;

ts = 0;
sketch.inc("aaa", ts, 1);
sketch.inc(std::string("bbb"), ts, 4);
sketch.inc("ccc", ts, 8);

std::cerr << sketch.get(std::string("aaa"), 4, ts) << std::endl;
// 1

std::cerr << sketch.get("bbb", 4, ts) << std::endl;
// 4

std::cerr << sketch.get("ccc", 4, ts) << std::endl;
// 8

std::cerr << sketch.get("ddd", 4, ts) << std::endl;
// 0

ts = 4;
std::cerr << sketch.get("aaa", 2, ts) << std::endl;
// 0
std::cerr << sketch.get("bbb", 2, ts) << std::endl;
// 0
std::cerr << sketch.get(std::string("ccc"), 2, ts) << std::endl;
// 0
std::cerr << sketch.get("ddd", 2, ts) << std::endl;
// 0

std::cerr << sketch.get("aaa", 8, ts) << std::endl;
// 1
std::cerr << sketch.get("bbb", 8, ts) << std::endl;
// 4
std::cerr << sketch.get("ccc", 8, ts) << std::endl;
// 8
std::cerr << sketch.get("ddd", 8, ts) << std::endl;
// 0
```

CHAPTER

THREE

WRAPPERS

- Python - https://github.com/martin-majlis/py-probstructs/

3.1 Classes

3.1.1 CountMinSketch

Documentation

template<class **T**>
class probstructs::**CountMinSketch**

Count–min sketch (CM sketch) is a probabilistic data structure that serves as a frequency table of events in a stream of data. It uses hash functions to map events to frequencies, but unlike a hash table uses only sub-linear space, at the expense of overcounting some events due to collisions.

Paper: Cormode, Graham; S. Muthukrishnan (2005). "An Improved Data Stream Summary: The Count-Min Sketch and its Applications" (https://sites.google.com/site/countminsketch/cm-latin.pdf) Wiki: https://en.wikipedia.org/wiki/Count%E2%80%93min_sketch

Public Functions

CountMinSketch(uint32_t *width*, uint8_t *depth*)

Create CM sketch with width {width} and depth {depth}.

void **inc**(**const** std::string &*key*, *T delta*)

Increase counter for {key} by {delta}.

T **get**(**const** std::string &*key*)

Get count for {key}.

Example

```
using namespace probstructs;

CountMinSketch<int> sketch(100, 4);
sketch.inc("aaa", 1);
sketch.inc(std::string("bbb"), 5);
sketch.inc("aaa", 2);
```

(continues on next page)

(continued from previous page)

```
std::cerr << sketch.get(std::string("aaa") << std::endl;
// 3

std::cerr << sketch.get("bbb") << std::endl;
// 5

std::cerr << sketch.get("ccc") << std::endl;
// 0
```

3.1.2 ExponentialHistorgram

Documentation

template<class **T**>
class probstructs::**ExponentialHistorgram**

Exponential histogram (EH) is a probabilistic data structure that serves as a frequency counter for specific elements in the last N elements from stream..

Paper: MAYUR DATAR, ARISTIDES GIONIS†, PIOTR INDYK, AND RAJEEV MOTWANI (2002). "MAINTAINING STREAM STATISTICS OVER SLIDING WINDOWS" (http://www-cs-students.stanford.edu/~datar/papers/sicomp_streams.pdf)

Public Functions

ExponentialHistorgram(uint32_t *window*)

Create exponential histogram for last {window} elements.

void **inc**(uint32_t *tick*, *T delta*)

Increase counter by {delta} when on the position {tick} in the stream.

T **get**(uint32_t *window*, uint32_t *tick*)

Get the counter for last {window} elements when on the position {tick} in the stream.

Example

```
using namespace probstructs;

ExponentialHistorgram<int> eh(4);
uint ts = 0;

ts = 0;
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 0, 0, 0

eh.inc(ts, 1);

std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 1, 1, 1

ts = 1;
```

(continues on next page)

(continued from previous page)

```
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 0, 1, 1

eh.inc(ts, 1);

std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 1, 2, 2

ts = 3;
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 0, 2, 2

eh.inc(ts, 1);
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 1, 3, 3

ts = 5;
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 0, 1, 1

eh.inc(ts, 1);
std::cerr << eh.get(1, ts) << ", " << eh.get(4, ts) << ", " << eh.get(8, ts) <<␣
↪std::endl;
// 1, 2, 2
```

3.1.3 ExponentialCountMinSketch

Documentation

template<class **T**>
class probstructs::**ExponentialCountMinSketch**

Exponential count-min sketch (ECM-Sketch) combines CM-Sketch with EH to count number of different elements in the last N elements in the stream.

Paper: Odysseas Papapetrou, Minos Garofalakis, Antonios Deligiannakis (2015). "Sketching distributed sliding-window data streams" (http://users.softnet.tuc.gr/~adeli/papers/journals/VLDBJ2015.pdf)

See *CountMinSketch*

See *ExponentialHistorgram*

Public Functions

ExponentialCountMinSketch(uint32_t *width*, uint8_t *depth*, uint32_t *window*)
Create ECM-Sketch with width {width}, depth {depth} to count elmenets in the last {window} elements.

void **inc**(**const** std::string &*key*, uint32_t *tick*, *T delta*)
Increase counter for {key} by {delta} when on the position {tick} in the stream.

T **get**(**const** std::string &*key*, uint32_t *window*, uint32_t *tick*)
Get counter for {key}for last {window} elements when on the position {tick} in the stream.

Example

```
using namespace probstructs;

ExponentialCountMinSketch<int> sketch(100, 4, 8);

uint ts = 0;

ts = 0;
sketch.inc("aaa", ts, 1);
sketch.inc(std::string("bbb"), ts, 4);
sketch.inc("ccc", ts, 8);

std::cerr << sketch.get(std::string("aaa"), 4, ts) << std::endl;
// 1

std::cerr << sketch.get("bbb", 4, ts) << std::endl;
// 4

std::cerr << sketch.get("ccc", 4, ts) << std::endl;
// 8

std::cerr << sketch.get("ddd", 4, ts) << std::endl;
// 0

ts = 4;
std::cerr << sketch.get("aaa", 2, ts) << std::endl;
// 0
std::cerr << sketch.get("bbb", 2, ts) << std::endl;
// 0
std::cerr << sketch.get(std::string("ccc"), 2, ts) << std::endl;
// 0
std::cerr << sketch.get("ddd", 2, ts) << std::endl;
// 0

std::cerr << sketch.get("aaa", 8, ts) << std::endl;
// 1
std::cerr << sketch.get("bbb", 8, ts) << std::endl;
// 4
std::cerr << sketch.get("ccc", 8, ts) << std::endl;
// 8
std::cerr << sketch.get("ddd", 8, ts) << std::endl;
// 0
```

3.1.4 Hash

Documentation

class probstructs::**Hash**

Hash is wrapper for MurmurHash3 - 32bit

Wiki: https://en.wikipedia.org/wiki/MurmurHash#MurmurHash3

Public Functions

Hash(uint32_t *seed*)

Create hashing function with {seed}.

uint32_t **hash**(**const** std::string &*key*)

Hash {key}.

Example

```
using namespace probstructs;
Hash h1(1);

std::cerr << h1.hash("aaa") << std::endl;
// 390644701
std::cerr << h1.hash(std::string("bbb")) << std::endl;
// 2512199470
```

genindex

INDEX

P