DistoGram Documentation

Release 3.0.0

R. Picard

Feb 05, 2022

Contents:

DistoGram is a library that allows to compute histogram on streaming data, in distributed environments. The implementation follows the algorithms described in Ben-Haim's Streaming Parallel Decision Trees

CHAPTER 1

Get Started

First create a compressed representation of a distribution:

```
import numpy as np
import distogram

distribution = np.random.normal(size=10000)

# Create and feed distogram from distribution
# on a real usage, data comes from an event stream
h = distogram.Distogram()
for i in distribution:
    h = distogram.update(h, i)
```

Compute statistics on the distribution:

```
nmin, nmax = distogram.bounds(h)
print("count: {}".format(distogram.count(h)))
print("mean: {}".format(distogram.mean(h)))
print("stddev: {}".format(distogram.stddev(h)))
print("min: {}".format(nmin))
print("5%: {}".format(distogram.quantile(h, 0.05)))
print("25%: {}".format(distogram.quantile(h, 0.25)))
print("50%: {}".format(distogram.quantile(h, 0.50)))
print("75%: {}".format(distogram.quantile(h, 0.75)))
print("95%: {}".format(distogram.quantile(h, 0.95)))
print("max: {}".format(nmax))
```

```
count: 10000
mean: -0.005082954640481095
stddev: 1.0028524290149186
min: -3.5691130319855047
5%: -1.6597242392338374
25%: -0.6785107421744653
50%: -0.008672960012168916
```

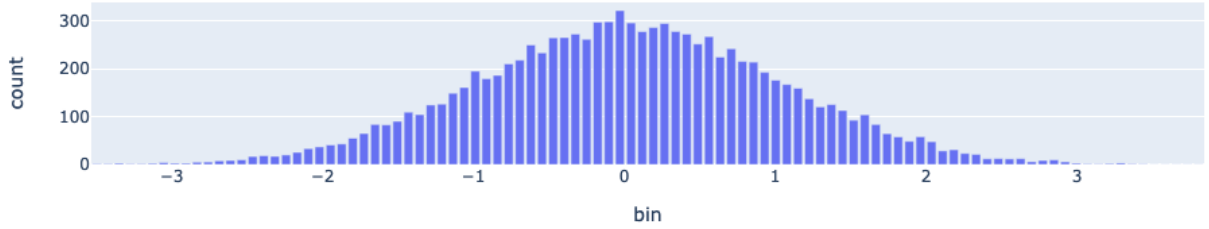
(continues on next page)

(continued from previous page)

```
75%: 0.6720718926935414
95%: 1.6476822301131866
max: 3.8800560034877427
```

Compute and display the histogram of the distribution:

```
hist = distogram.histogram(h)
df_hist = pd.DataFrame(np.array(hist), columns=["bin", "count"])
fig = px.bar(df_hist, x="bin", y="count", title="distogram")
fig.update_layout(height=300)
fig.show()
```



CHAPTER 2

Reference

class distogram.**Distogram**(*bin_count: int = 100*, *weighted_diff: bool = False*)
Compressed representation of a distribution.

distogram.**update**(*h: distogram.Distogram*, *value: float*, *count: int = 1*) → distogram.Distogram
Adds a new element to the distribution.

Parameters

- **h** – A Distogram object.
- **value** – The value to add on the histogram.
- **count** – [Optional] The number of times that value must be added.

Returns A Distogram object where value as been processed.

Raises ValueError if count is not strictly positive.

distogram.**merge**(*h1: distogram.Distogram*, *h2: distogram.Distogram*) → distogram.Distogram
Merges two Distogram objects

Parameters

- **h1** – First Distogram.
- **h2** – Second Distogram.

Returns A Distogram object being the composition of h1 and h2. The number of bins in this Distogram is equal to the number of bins in h1.

distogram.**count_at**(*h: distogram.Distogram*, *value: float*)
Counts the number of elements present in the distribution up to value.

Parameters

- **h** – A Distogram object.
- **value** – The value up to what elements must be counted.

Returns An estimation of the real count, computed from the compressed representation of the distribution. Returns None if the Distogram object contains no element or value is outside of the distribution bounds.

`distogram.`**`count`**(*h: distogram.Distogram*) → float
Counts the number of elements in the distribution.

Parameters **`h`** – A Distogram object.

Returns The number of elements in the distribution.

`distogram.`**`bounds`**(*h: distogram.Distogram*) → Tuple[float, float]
Returns the min and max values of the distribution.

Parameters **`h`** – A Distogram object.

Returns A tuple containing the minimum and maximum values of the distribution.

`distogram.`**`mean`**(*h: distogram.Distogram*) → float
Returns the mean of the distribution.

Parameters **`h`** – A Distogram object.

Returns An estimation of the mean of the values in the distribution.

`distogram.`**`variance`**(*h: distogram.Distogram*) → float
Returns the variance of the distribution.

Parameters **`h`** – A Distogram object.

Returns An estimation of the variance of the values in the distribution.

`distogram.`**`stddev`**(*h: distogram.Distogram*) → float
Returns the standard deviation of the distribution.

Parameters **`h`** – A Distogram object.

Returns An estimation of the standard deviation of the values in the distribution.

`distogram.`**`histogram`**(*h: distogram.Distogram*, *bin_count: int = 100*) → Tuple[List[float], List[float]]
Returns a histogram of the distribution in numpy format.

Parameters

- **`h`** – A Distogram object.
- **`bin_count`** – [Optional] The number of bins in the histogram.

Returns An estimation of the histogram of the distribution, or None if there is not enough items in the distribution.

`distogram.`**`frequency_density_distribution`**(*h: distogram.Distogram*) → Tuple[List[float], List[float]]
Returns a histogram of the distribution

Parameters **`h`** – A Distogram object.

Returns An estimation of the frequency density distribution, or None if there are not enough values in the distribution.

`distogram.`**`quantile`**(*h: distogram.Distogram*, *value: float*) → Optional[float]
Returns a quantile of the distribution

Parameters

- **`h`** – A Distogram object.
- **`value`** – The quantile to compute. Must be between 0 and 1

Returns An estimation of the quantile. Returns None if the Distogram object contains no element or value is outside of [0, 1].

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index