Stream Mining One-Hot Encoding and DGIM

Zeno Adrian Weil

Data Science 1 Goethe University Frankfurt

7th June 2022

• categorical features common

$$x \in \{\text{red}, \text{green}, \text{blue}\}$$

- categorical features common
- need for numbers in algorithms

$$x \in \{\text{red}, \text{green}, \text{blue}\}$$

- categorical features common
- need for numbers in algorithms
- naïve approach: number serially

$$x \in \{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$$

- categorical features common
- need for numbers in algorithms
- naïve approach: number serially

$$x \in \{1, 2, 3\}$$
 $X \in \{red, green, blue\}$

- categorical features common
- need for numbers in algorithms
- naïve approach: number serially
 - danger of meaningless calculations

$$x \in \{ \text{red, green, blue} \}$$

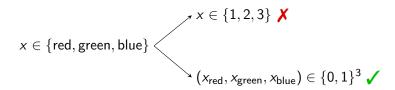
One-Hot Encoding

- categorical features common
- need for numbers in algorithms
- naïve approach: number serially
 - danger of meaningless calculations
- one-hot encoding

$$x \in \{ \text{red, green, blue} \}$$

One-Hot Encoding

- categorical features common
- need for numbers in algorithms
- naïve approach: number serially
 - danger of meaningless calculations
- one-hot encoding
 - one binary feature for each possible value





Goals

• Estimate the number of ones in a bit stream!



- Estimate the number of ones in a bit stream!
- Be space-efficient!

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- $\mathcal{O}(\log_2 N)$ buckets

$$[\dots 1 \ 0 \ 1] [1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1] [1 \ 0 \ 1] [1 \ 0 \ 0] [1] [1] [0$$

The Datar-Gionis-Indyk-Motwani Algorithm

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp



- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones

sizes: 8	· · · · · · · · · · · · · · · · · · ·		4	2		1	1	
101	10110001	0	11101	1001	0	1	1	0

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two

sizes: 8	4		4	2		1	1	
101	10110001	0	11101	1001	0	1	1	0

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones

sizes: 8	4		4	2		1	1	
101	10110001	0	1 1 1 0 1	1001	0	1	1	0

The Datar-Gionis-Indyk-Motwani Algorithm

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

sizes: 8	4		4	2		1	1	
101	10110001	0	1 1 1 0 1	1001	0	1	1	0

The Datar-Gionis-Indyk-Motwani Algorithm

Goals

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

estimation: 16

The Datar-Gionis-Indyk-Motwani Algorithm

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

sizes: 8	4	4	2	1 1
101	10110001 0	11101	1001	0 1 1 0
	estimation: 16	real	ity: 14	

The Datar-Gionis-Indyk-Motwani Algorithm

Goals

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets
 - error rate: $\pm 50\%$

sizes: 8 4 4 2 $\dots 101 10110001 0 1 11101 1001$

estimation: 16

reality: 14

The Datar-Gionis-Indyk-Motwani Algorithm

- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets
 - error rate: ±50%
- needs only O((log₂ N)²) bits

• 8124 samples of 23 mushroom species



- 8124 samples of 23 mushroom species
 - 4208 edible
 - 3916 poisonous

- 8124 samples of 23 mushroom species
 - 4208 edible
 - 3916 poisonous
- 22 attributes with 128 possible values

- 8124 samples of 23 mushroom species
 - 4208 edible
 - 3916 poisonous
- 22 attributes with 128 possible values
- saved as CSV

- 8124 samples of 23 mushroom species
 - 4208 edible
 - 3916 poisonous
- 22 attributes with 128 possible values
- saved as CSV

Are there simple rules to determine edibility?

- 8124 samples of 23 mushroom species
 - 4208 edible
 - 3916 poisonous
- 22 attributes with 128 possible values
- saved as CSV

Are there simple rules to determine edibility? Yes! (e.g. odour)

load CSV with Python

- load CSV with Python
- 2D array for the one-hot encoding of the odour

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package **dgim** for the **algorithm**

Implementation

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface

Topic 4: One-Hot Encoding and DGIM

On-bet recording denotes the technique of replacing a cotopolical attribute with 2 possible values by a binary X-any typic where the i-th element is 1 if and only if the attribute was set to the i-th value. The Datar Ginsh-beyly-Monovari algorithm is a technique to estimate the number of ones in the last X bits of a binary string. This program demonstrates the DGM algorithm on a data set of much soons. It estimates the number of selfield and polisonous sensorious for a chosen other and compares it to the real count.

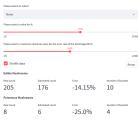


Implementation

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface
- options

Topic 4: One-Hot Encoding and DGIM

One-hot encoding denotes the technique of replacing a categorical attribute with k possible values by a binary string. This program demonstrates the DGIN algorithm on a data set of mushrooms. It estimates the number of edible and poisonous must more for a chosen odour and compares it to the real count.



Implementation

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface
- options
 - odour type

Topic 4: One-Hot Encoding and DGIM

One-hot encoding denotes the technique of replacing a categorical attribute with A possible values by a binary Avery taptie where the inh element is 1 in and only if the attribute was set to the inh value. The Data Glossin-Holy-Morkean's algorithms is attenting to estimate the manter of ensure in the last N bits of binary string. This program demonstrates the DRM algorithm on a data set of mushoroms. It estimates the number of deliberation recommendation of the program of the p



Implementation

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface
- options
 - odour type
 - window size N

Topic 4: One-Hot Encoding and DGIM



Implementation

- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface
- options
 - odour type
 - window size N
 - error rate

Topic 4: One-Hot Encoding and DGIM

One-hot encoding denotes the technique of replacing a categorical attribute with k possible values by a binary strine. This program demonstrates the DGM algorithm on a data set of mushmons. It estimates the number of edible and poisonous must more for a chosen odour and compares it to the real count.



Please select an odour:

None

Please select a value for N:

16 2048

Please select a maximum absolute value for the error rate of the DGIM algorithm:

1%



Real count Estimated count Error Number of buckets

214 208 -2.8% 10

Poisonous Mushrooms

Real count Estimated count Error Number of buckets 11 12 9.09% 6

Rerun

Please select an odour:

None

Please select a value for N:

16 2048

Please select a maximum absolute value for the error rate of the DGIM algorithm: 50%

1% 100%

✓ Shuffle data Rerun

Edible Mushrooms

 Real count
 Estimated count
 Error
 Number of buckets

 1675
 1872
 11.76%
 15

Poisonous Mushrooms

Real count Estimated count Error Number of buckets 7.46% 9

Please select an odour:

None

Please select a value for N:

16 2048

Please select a maximum absolute value for the error rate of the DGIM algorithm:

1%

1% 100%

Shuffle data

Edible Mushrooms

Real count Estimated count Error Number of buckets

1678 1672 -0.36% 410

Poisonous Mushrooms

Real count Estimated count Error Number of buckets

68 68 0.0% 68

Rerun

References

- Project code: https://github.com/s9770652/DS1-DGIM
- Mushroom data set: https://archive-beta.ics.uci.edu/ml/datasets/mushroom
- Streamlit: https://streamlit.io/
- Python package dgim: https://pypi.org/project/dgim/
- Description of one-hot encoding: https://sherbold.github.io/intro-to-data-science/04_ Data-Analysis-Overview.html#Features
- Description of the DGIM algorithm (Section 4.6): http://infolab.stanford.edu/~ullman/mmds/ch4.pdf