Stream Mining One-Hot Encoding and DGIM

Zeno Adrian Weil

Data Science 1 Goethe University Frankfurt

7th June 2022

 One-Hot Encoding
 The DGIM Algorithm
 The Mushroom Data Set
 Implementation
 Reference

 ●
 0
 0
 0
 0
 0

• categorical features common

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

- categorical features common
- need for numbers in algorithms

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

- 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

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

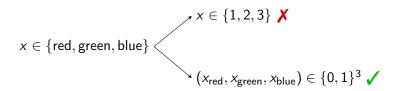
- categorical features common
- need for numbers in algorithms
- naïve approach: number serially
 - meaningless numerical calculations

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

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

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

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



Hot Encoding The DGIM Algorithm The Mushroom Data Set Implementation Reference

O ΟΩΩΩ Ο

The Datar-Gionis-Indyk-Motwani Algorithm



Encoding The DGIM Algorithm The Mushroom Data Set Implementation Ref

The Datar-Gionis-Indyk-Motwani Algorithm

coding The DGIM Algorithm The Mushroom Data Set Implementation Reference

The Datar-Gionis-Indyk-Motwani Algorithm

Objectives

• 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
- O(log₂ N) buckets

$$\boxed{\dots 101 \ | 10110001 \ | 0 \ | 11101 \ | 1001 \ | 0 \ | 1 \ | 1}$$

- 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
- $\mathcal{O}(\log_2 N)$ buckets
 - timestamp
 - size = number of ones

- 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	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; end with 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
 - include all ones; end with ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

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

Objectives

- 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: end with ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

estimation: 16

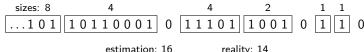
- Estimate the number of ones in a bit stream!
- Be space-efficient!
- window size N
- O(log₂ N) buckets
 - timestamp
 - size = number of ones
 - powers of two
 - include all ones: end with ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets

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

Objectives

- 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; end with ones
- estimation: half the size of the oldest bucket + sum of sizes of all other buckets
 - error rate: 50%

- 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: end with 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)

Hot Encoding The DGIM Algorithm The Mushroom Data Set Implementation References
O O ●000 O

Implementation

Zeno Adrian Weil

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
- \bullet Python package $\ensuremath{\mbox{\bf dgim}}$ for the $\ensuremath{\mbox{\bf algorithm}}$

- 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

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



- 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-bot encoding denotes the technique of regulating a categorical attribute with it possible values by a binary k-ary topic where the i-fit element is 1 if and only if the attribute was set to the i-fit value. The Datar Gioris-holdy-Mohavari algorithm is a technique to estimate the number of ones in the last Visits of binary string. This program demonstrates the DGRI algorithm on a data set of mushbooms. It estimates the number of refilled and polisposus numberous for a chosen order and compare in the three of our con-



- 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 k possible values by a bigany string. This program demonstrates the DGM algorithm on a data set of mushrooms. It estimates the number of edible and poisonous must rooms for a chosen odour and compares it to the real count.



- 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 One-hot encoding denotes the technique of replacing a categorical attribute with k possible values by a

bigany string. This program demonstrates the DGM algorithm on a data set of mushrooms. It estimates the number of edible and poisonous must rooms for a chosen odour and compares it to the real count.



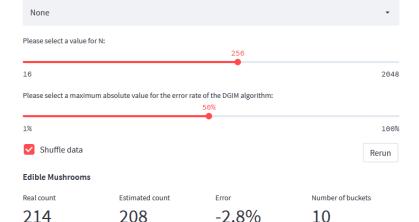
- load CSV with Python
- 2D array for the one-hot encoding of the odour
- Python package dgim for the algorithm
- Streamlit for the interface
- options
 - odour type
 - window size N
 - error rate

Topic 4: One-Hot Encoding and DGIM

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







Poisonous Mushrooms

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

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% 100%

✓ Shuffle data Rerun

50%

Edible Mushrooms

 Real count
 Estimated count
 Error
 Number of buckets

 1675
 1872
 11.76%
 15

Poisonous Mushrooms

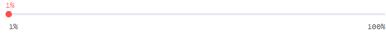
Real count Estimated count Error Number of buckets 7.46% 9

Please select an odour:

None •

Please select a value for N:

16 2048



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

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