VOUW

Pattern Mining on Matrices

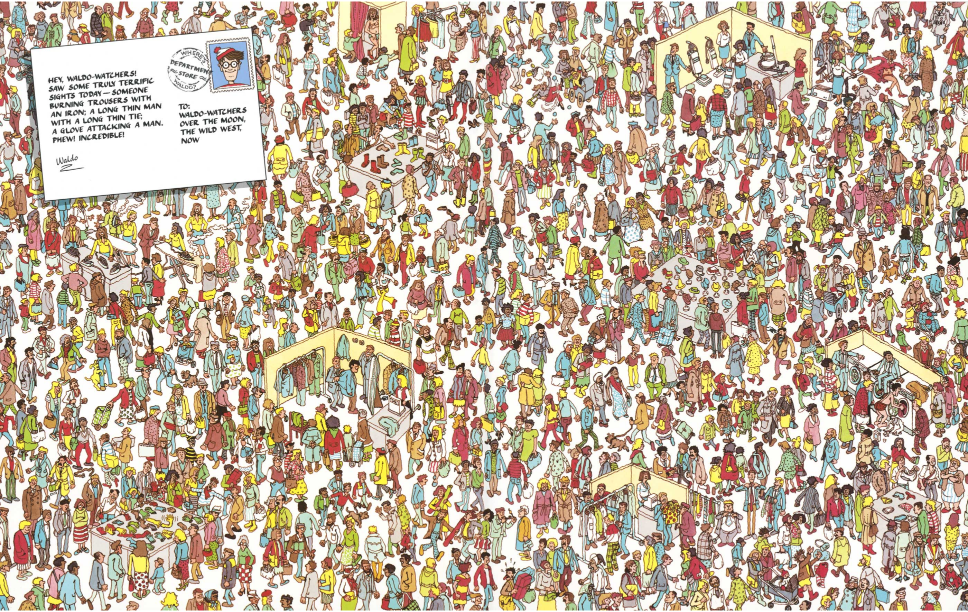
Micky Faas

March 14, 2019

With VOUW we propose a method of **mining patterns on matrices**. It encompasses:

- A theoretical framework
- An optimization problem formulated using MDL
- A heuristic algorithm.

Where is Waldo?

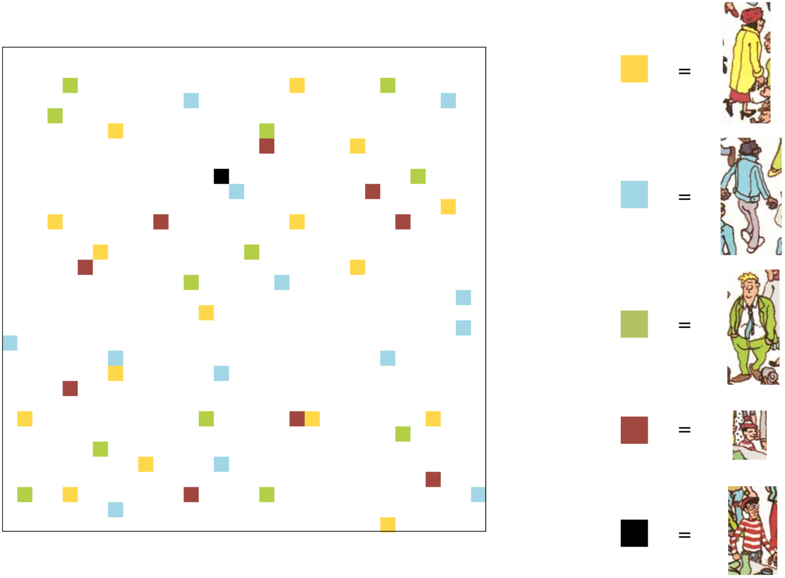


Informal problem

Say we have some $M \times N$ matrix A, we want to discover and extract recurring structure. Why?

- Data analysis
- Distance metric for comparison
- Clustering

Optimized Waldo



Patterns and Instances

Term?	Form	Encodes...
Pattern	Submatrix of A	...relative positions of elements
Offset	$(i, j) \in M \times N$	...position of entire pattern
Instance	$M \times N$ matrix	...absolute positions of elements
Instantiation	$M \times N$ matrix	...what pattern's instance should be at which index

$$\underbrace{\begin{bmatrix} 1 & \cdot \\ \cdot & 1 \end{bmatrix}}_{\text{Pattern}}, \underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{\text{Instance with offset (2,2)}}$$

Example

$$
A = \overbrace{\begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & 1 \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & \cdot & \cdot \end{bmatrix}}^{\text{Original matrix}}, \quad G = \overbrace{\begin{bmatrix} X & \cdot & \cdot & \cdot & \cdot & Y \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot & X & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & X & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}}^{\text{Instantiation}}
$$

$$
\overbrace{H = \left\{ X = \underbrace{\begin{bmatrix} 1 & \cdot \\ \cdot & 1 \end{bmatrix}}_{\text{Pattern}}, \; Y = \underbrace{\begin{bmatrix} 1 \end{bmatrix}}_{\text{Pattern}} \right\}}^{\text{Model}}
$$

Building Patterns

We can construct complex patterns by repeatedly combining simpler ones. We use instances for this as they encode the position of one pattern relative to another.

$$\underbrace{\begin{matrix} X = \begin{bmatrix} 0 \end{bmatrix} \\ Y = \begin{bmatrix} 1 \end{bmatrix} \end{matrix}}_{\text{Singleton patterns}} \longrightarrow \underbrace{\begin{matrix} \bar{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \\ \bar{Y} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix} \end{matrix}}_{\text{Instances}} \longrightarrow \bar{X} + \bar{Y} = \underbrace{\begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix}}_{\text{Matrix sum}} \longrightarrow \underbrace{\begin{bmatrix} 0 & \cdot \\ \cdot & 1 \end{bmatrix}}_{\text{New pattern}}$$

Model Space

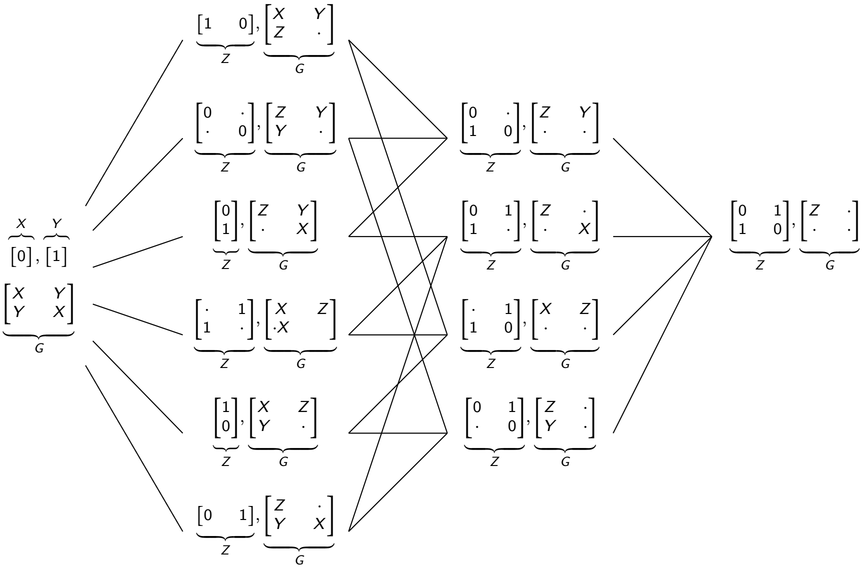
Idea: we inductively define the model space by starting with singletons for each element of A. Then we repeatedly merge instances until we finally obtain a pattern that equals A.

- At the beginning we have the completely underfit model
- We end up with a completely overfit model

Everything in between is a possible solution, but *we want the solution that describes A best.*

For example $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$: what is the space of possible solutions?

2x2 Model Space Lattice



How complex is this lattice? Very complex!
Idea: pick two instances to combine $MN - 1$ times.

$$\prod_{n=0}^{MN-2} \binom{MN-n}{2} = \prod_{n=0}^{MN-2} \frac{MN-n}{2(MN-n-2)!} = \frac{(MN)!(MN-1)!}{2^{MN-1}}.$$

We will need to use heuristics.

Optimization using MDL

Idea: the best solution is a balance of model and instantiation complexity. We use two-part MDL:

$$\underbrace{L(H)}_{\text{Model}} + \underbrace{L(A|H)}_{\text{Data given model}}$$

In this case:

$$L\left(\underbrace{\overbrace{\begin{bmatrix} 1 & \cdot \\ \cdot & 1 \end{bmatrix}}^{\text{Pattern}}, \overbrace{\begin{bmatrix} 1 \end{bmatrix}}^{\text{Pattern}}}_{\text{Model}}\right) + L\left(\underbrace{\overbrace{\begin{matrix} X & \cdot & \cdot & \cdot & \cdot & Y \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot & X & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & X & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{matrix}}^{\text{Instantiation Matrix}}}_{\text{Data given model}}\right)$$

Why does this work? MDL is founded on these principles:

Kolmogorov Complexity of given data is the shortest computer program to produce that data.

Occam's Razor . Given competing hypotheses, pick the one with the fewest assumptions.

No-hypercompression theorem . Data with no inherent structure, cannot be compressed.

Kraft Inequality . One-to-one correspondence between code lengths and probabilities.

A Simple Algorithm

Idea: look at A once to build an instantiation matrix G. Make a model H using only singleton patterns, one for each possible value in A.

Then we constantly refine our description by merging instances to form patterns. Once we merge instances, we never reconsider (greedy approach).

$$\begin{array}{c} \overbrace{\begin{bmatrix}0\end{bmatrix}}^{X}, \overbrace{\begin{bmatrix}1\end{bmatrix}}^{Y} \\ \underbrace{\begin{bmatrix} X & Y \\ Y & X \end{bmatrix}}_{G} \end{array} \longrightarrow \underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{Z}, \underbrace{\begin{bmatrix} X & Y \\ Z & \cdot \end{bmatrix}}_{G} \longrightarrow \underbrace{\begin{bmatrix} 0 & \cdot \\ 1 & 0 \end{bmatrix}}_{Z}, \underbrace{\begin{bmatrix} Z & Y \\ \cdot & \cdot \end{bmatrix}}_{G}$$

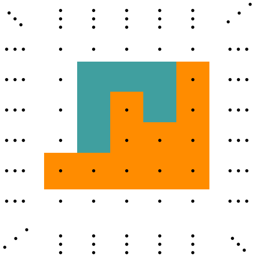
Which pair of instances do we combine? We will first derive a list of candidates from the instantiation matrix:

$$\underbrace{\begin{bmatrix} X & Y & X & Y & X & X \\ X & Y & X & X & Y & Y \\ Y & Y & X & X & X & Y \\ X & X & X & Y & X & X \\ X & X & X & X & X & X \\ X & Y & X & X & X & Y \end{bmatrix}}_{\text{Instantiation matrix}} \longrightarrow \underbrace{\begin{matrix} X & \\ & Y \end{matrix}}_{\text{Candidate}}$$

The candidate that minimizes the MDL equation best is picked.

Adjacency

We constrain the idea of the last slides by only looking at **adjacent** instances.



This gives two benefits: (1) visits all candidates just once, (2) reduces candidate space.

VOUW

A Simple Algorithm (2)

This simple algorithm gives an acceptable complexity:

- Each iteration (find candidates and merge best) is (almost) linear.
- $MN - 1$ iterations worst-case, but we need only a fraction in any practical case.

However:

- Greedy approach sometimes gives completely wrong results.
- No tolerance to noise.

Open Problems

A ‘wish list’ for the future:

- Actual data and use-cases (!)
- Noise robustness.
- Multi-instance candidates (more than 2).
- Hierarchical encoding of patterns.
- Improvement of the current (prequential) encoding.

Here is Waldo