0.0.1 In a Nutshell

`[Short clip of a 'noise' image to revealing some hidden letters]`

Narrated: *This seemingly random noise contains structures that can not easily be separated. We can find these structures by compressing the image. Compression is simply a way to encode the same information in a more structured and concise way. We can use this to actually learn something about an unknown dataset. In this case, we find regions that contain more recurrence than others and therefore compress more easily. Eventually, we separate the truly random data (noise) from the structures we are interested in (signal).*

0.0.2 Introduction

Hello, my name is Micky and I am a Computer Science graduate student from Leiden University. I this video, I will briefly present to you my paper titled *Vouw: Geometric Pattern Mining using the MDL Principle.*

`[Slide #1 - geometric pattern mining]`

Geometric Pattern Mining is a subclass of frequent pattern mining. The goal is to extract recurring substructures (patterns) from grid-like data. This data can have any number dimensions, but we only look at 2D geometric matrices of discrete data - let's just call them matrices from now on. One of the differences between existing methods and our approach, is that we consider irregular shaped patterns in the data, as long as their elements are adjacent. Let's look at a very simple example.

`[Slide #2 - example matrix]`

This Boolean matrix contains four different kinds of triangles, that occur multiple times. These triangles can be thought of as recurring patterns in the context of frequent pattern mining.

`[Slide #3 - example matrix with triangles colored]`

Instead of storing this matrix as separate values, we could store each triangle once and then make a list of where each triangle occurs in the original matrix.

`[Slide #4 - example matrix 'compressed']`

By doing so we have found a more concise way to store the same information (compression), but at the same time know something about the data that we did not know before - namely that it contains four kinds of triangles and where they occur. This suggests that compression and learning are two sides of the same coin. This also suggests that we can compress our matrix by separating the structure (four different triangles) from the accidental information (where they occur).

0.0.3 MDL

`[Slide #5 - MDL?]`

This intuition is formalized in the Minimum Description Length principle, or MDL for short. Simply said, MDL tells us to search for the smallest loss-less compression of our data. According to MDL, the shortest description provides the most insight into the data. While MDL primarily aims to solve model selection, it has also been used successfully for many explanatory data analysis problems such as this.

`[Slide #6 - two-part MDL]`

In these cases, a simplified version of MDL is commonly used. Two-part MDL, as shown here, is a minimization problem consisting of two terms. Both terms are length functions, which represent the length of the *model* and the length of the *data given the model, respectively.*

`[Slide #7 - example matrix (again), model and instantiation highlighted]`

In the case of geometric pattern mining, the model (left) is simply the set of geometric patterns. We call the second part the *instantiation matrix*, as it actually describes how/where the patterns must be instantiated to obtain the original matrix. According to two-part MDL, equation we must minimize the sum of the lengths of both model and instantiation, but why does that work? First notice how we have reduced the size of the data to 90% in this solution. Let's look at a different solution for the same example.

`[Slide #8 - example matrix, underfit]`

This is also a valid model and instantiation for the original matrix. However, the model is really small and only contains singleton elements, while the instantiation basically contains the entire matrix. This is clearly under-fit and not a good solution as no compression was achieved.

`[Slide #9 - example matrix, overfit]`

The other extreme is the completely over-fit scenario, where the model contains the entire matrix. Again, the absence of compression tells us this is not a good solution. Ideally we want to use this compression ratio to find a point between these two extremes that reveals the most useful information about the matrix.

0.0.4 Algorithm

`[Slide #10 - lattice]`

Let's look at all possible solutions for a simpler matrix.

In our algorithm we start at the left-side of this lattice, the under-fit solution. We then continue to the right by always picking the best branch from a greedy perspective. Stepping to the right in the lattice means that the model H grows, while the instantiation I shrinks. Recall that we want to minimize the sum of their lengths according to the MDL equation. This raises two important questions: (1) how do we compute these lengths and (2) how do we make one 'step' in the lattice?

`[Slide #11 - candidates]`

First we enumerate the candidates by traversing the instantiation matrix. This efficiently limits the scope to patterns that are actually in the data. The best candidate is usually a combination of two patterns that occurs often, because this removes many elements from the instantiation matrix - decreasing its length. However, this relation is a little more complex.

`[Slide #12 - Length and gain]`

For each candidate we compute its gain. Gain is the number of bits saved by merging elements in the instantiation matrix minus the cost of storing the newly merged pattern. To compute any of these lengths, we use a hypothetical encoding scheme. This scheme computes the lengths of each pattern as if we would actually compress the data.

`[Slide #13 - Pattern union]`

After we compute the gain for each candidate, we pick the candidate with the best gain. We use this candidate to expand the model by adding a new pattern. This new pattern is a union of the two candidate patterns. Pattern union is also the fundamental operation of the algorithm and because we use a greedy heuristic, patterns can only be merged and cannot be taken apart during a later step.

`[Slide #14 - Simplified algorithm]`

The algorithm stops if no candidates with a positive gain can be found, as we see here in a simplified overview of the algorithm. The final algorithm also contains a step that performs a local search to refine and optimize each pattern union, which is explained in more detail in the paper. This concludes my brief review of our method as I would like to take a moment to discuss the experiments.

0.0.5 Experiments

The paper describes a fairly simple experiment that mainly tests the quality of the compression we can achieve. Since, according to MDL, the amount of compression roughly equals the amount of information we can extract, we use this to evaluate the performance of the algorithm.

`[Slide #15 - Experiments]`

The general idea is that we use a synthetic dataset generator to create a set of random patterns. These patterns are distributed over a matrix that is otherwise filled with noise. We then compress the resulting matrix with our algorithm. Here we see a small example of a generated matrix and a resulting compression. Ideally we want to find the original patterns and nothing more. We test this by comparing the elements that the algorithm thinks are patterns with the actual ground-truth elements.

`[Slide #16 - Results]`

Let's take a look at some of the results for differently sized matrices. In this graph we plot the compression ratio against the signal-to-noise ratio of the input matrix. Ideally we want these figures to be more or less opposite. This means that if the signal ratio is low we get only a small amount of compression, because noise can, by definition, not be compressed. In the graph we see a nice linear relation, which means the compression algorithm works very well on this data. In the paper we also present precision/recall figures and performance in terms of speed.

`[Slide #17 - End]`

The next step for future work is to test in different use-cases. We think that VOUW can be used for clustering and classification of textures, images, geographical data and possibly also multi-dimensional data. We hope that this paper can serve as a basis for other work on geometric pattern mining and MDL-based approaches on matrices. We look forward to your questions and thank you for watching this presentation.