2-Dimensional Pattern Complexity

Josef Klafka

7/10/18

1 Introduction

To understand the algorithm in Kempe et al. (2015), we want to evaluate the complexity of a 2-dimensional 10×10 two-color block pattern. Evaluating pattern complexity is difficult for low-dimensional blocks. We could 'flatten' the block into a 1-dimensional strip, but this does not always provide accurate complexity measurements. Instead, we'll adapt an algorithm for measuring mathematical complexity of 1-dimensional patterns, which are known as 'strings', into two dimensions, and connect a pattern's complexity with its frequency. By using a pattern's frequency, we can obtain an approximation of the pattern's complexity.

2 Kolmogorov-Chaitin Complexity

Definition 2.1 (Complexity)**.** *Kolmogorov-Chaitin complexity* is the length of the shortest program p that outputs the string s running on a universal Turing machine T, or symbolically

$$K_T(s) = \min\{|p|; T(p) = s\}$$

where K_T is the Kolmogorov-Chaitin complexity and $|p|$ denotes the length of the program p taken as input by a Turing machine T.

A *Turing machine* is a theoretical simple type of computer that takes in a set of instructions and completes the tasks described on the instructions. Specifically, a Turing machine is programmed with a set of instructions, and then takes in strip of tape and alters the tape according to the instructions. An actual example of a Turing machine is a programming language, which takes in the set of instructions ('code') you give it and performs some task. It's important to note that a classic Turing machine reads input in one dimension, so it can't work with 2-dimensional block patterns.

Explanation : To consider Kolmogorov-Chaitin complexity in concrete terms: imagine you pick a programming language such as Python or C and pick an output s like 01101010101 or 202020. The programming language is T and the output is s. You want to find the shortest program in your programming language that gives you s as the output.

The shorter the shortest program that produces s is, the less complex s is. If the shortest program for 01010101 is 20 bytes long and the shortest program for 11010100 is 30 bytes long, then we can guess that 01010101 is less complex than 11010100. This makes sense, since we can describe 01010101 as "01 repeated four times" but we have to describe 11010100 by saying "1 then 1 then 0 then 1 then 0 then..." or a similar intensive individual explanation of each part of the string 11010100.

2.1 From Complexity to Frequency

Thus the Kolmogorov-Chaitin Complexity is a useful tool for describing how complex a 1-dimensional string is. But what if we want to describe the complexity of a block pattern in two dimensions?

For example, an 8×8 chess board, which has a clear alternating pattern to the black and white squares, has a simpler to describe pattern than an 8×8 board where the squares are randomly filled in with no discernible pattern.

For this, we have the coding theorem method. If you would like to read about the mathematics behind the coding theorem method, please skip to Appendix A.

Essentially, we run all possible 2-dimensional Turing machines which operate on 4×4 squares, producing two-color 4×4 patterns. We then calculate the probability of each of the patterns being generated by such a Turing machine, which after being put through a $-\log_2$ gives us an approximation of the Kolmogorov-Chaitin complexity.

Calculating the outcome of all such Turing machines took 12 days when run on a supercomputer in Spain! Since we need all of the outcomes to generate even one pattern complexity, this isn't something we can reasonably reproduce. Instead, we can access the results of the computation by the Spanish supercomputer for the Kempe algorithm.

3 Kempe Algorithm

For the block analysis described in Kempe et al. (2015), we need to compute the computational complexity of a 10×10 2-dimensional plot where each point can have one of two colors. To do so, we split the 10×10 graph into 16 different 4 x 4 squares. We draw these squares starting on every other row and every other column.

Each one of these 4×4 squares has a pattern denoted by p. We then find the number of times each pattern occurs, which is denoted by n_p. We then find i.e. lookup the Kolmogorov-Chaitin complexity for that pattern $K(p)$, and plug all these numbers into the following expression to obtain an approximation for the complexity of the 10×10 pattern:

$$(\star) \sum_p [\log_2(n_p) + K(p)]$$

$(\star)$ is the expression we've been looking for. Find all the 4×4 patterns p in a 10×10 grid, compute the n_p term and look up the $K(p)$ term for the 4×4 pattern. Then compute the sum over all patterns to find the final measurement for the complexity of the 10×10 pattern.

To find the $K(p)$ complexity term for each one of these patterns, you can use the applet at the following link: Applet. I had to download the applet and run it off my computer, but it allows you to select squares in a 4×4 grid and gives the $K(p)$ complexity for that pattern.

4 Conclusion

We know how to conceptualize the complexity for 1-dimensional patterns. How do we extend this to two dimensions? We extend a measure for the complexity of short 1-dimensional patterns to two dimensions, then use the frequency calculations for each pattern to find an approximation for the complexity of 4×4 patterns, which we can combine into an approximation of the Kempe 10×10 patterns.

5 Appendix A: Coding Theorem Mathematics

Definition 5.1 (Algorithmic Probability)**.** The *algorithmic probability* of a string s is a measure that describes the expected probability of a random program p running on a universal Turing machine T

producing s as output.

$$m(s) = \sum_{p:\,T(p)=s} \left(\frac{1}{2}\right)^{|p|}$$

Explanation : If we have a programming language that we use for T, and give it a program p to run, then what's the probability that the output of the program running on that programming language is s? Computers work with bits, which are made up of 0s and 1s. If each p has binary output, giving 0s and/or 1s so that the computer can work with it, then at each step of the program the probability of producing 0 or 1 is $\frac{1}{2}$. So we take the number of 0s and 1s produced by p, namely the length of the program p denoted as $|p|$, which has a probability of producing a given $|p|$-long output of 0s and 1s as $\frac{1}{2}^{|}p|$.

Theorem 5.2 (Coding Theorem)**.** Given a string s, then we have the following expression relating the Kolmogorov-Chaitin complexity of the string to its probability of occurring.

$$|-\log_2 m(s) - K(s)| < c$$

where c is a constant. This expression can be rewritten to give a nice approximation to the complexity measure.

$$K_m(s) = -\log_2 m(s) + O(1)$$

Explanation : The $O(1)$ term basically means that $K_m(s)$ and $-\log_2 m(s)$ only differ by an unknown constant, so $-\log_2 m(s)$ is a good approximation of $K_m(s)$. The $\log_2(n_p)$ in each term of the Kempe sum is necessary for adding the complexities of the smaller patterns together to get the complexity of the larger 10×10 pattern.

Therefore to find

6 References

1. ⋆ Zenil, Hector, Fernando Soler-Toscano, Jean-Paul Delahaye, and Nicolas Gauvrit. "Two-dimensional Kolmogorov complexity and an empirical validation of the Coding theorem method by compressibility." *PeerJ Computer Science* 1 (2015): e23.

2. Kempe, Vera, Nicolas Gauvrit, and Douglas Forsyth. "Structure emerges faster during cultural transmission in children than in adults." *Cognition* 136 (2015): 247-254.

3. Kolmogorov, Andrei Nikolaevich. "Three approaches to the quantitative definition of information." *International journal of computer mathematics* 2, no. 1-4 (1968): 157-168.

4. Zenil, Hector, Fernando Soler-Toscano, Kamaludin Dingle, and Ard A. Louis. "Correlation of automorphism group size and topological properties with program-size complexity evaluations of graphs and complex networks." *Physica A: Statistical Mechanics and its Applications* 404 (2014): 341-358.