Complex Systems — Problems for LAB

Prepared by Katarzyna Sznajd-Weron

1 Zipf law - till 17.10.2023

Visit the Project Gutenberg website (`https://www.gutenberg.org/`), which is a library of over 70,000 free eBooks. Download four eBooks of your choice and conduct the Zipf analysis, and then:

1. Save the result of the analysis in the text file. The name of the file should include the title of the book and the number of words it contains. The file should contain four columns: (1) rank, (2) word, (3) number of a given word in the text, (4) frequency of a given word in the text. Why do I need both columns (3) and (4)? Think about it.
2. Compare the empirical distribution from your analysis with the theoretical Zipf distribution by plotting both on the same graph in log-log scale (one graph for each book). Try to do the same in linear scale. What do you observe?
3. (extra 1) Try to fit 'a' and 'b' constants in Zipf-Mandelbrot law freq. $\propto 1/(\text{rank} + b)^a$ for two different languages. Is it possible to distinguish languages using a and b parameters only?
4. (extra 2) Check if LLM (large language models) generate text following Zipf-Mandelbrot law for a given language.

2 Self-organized criticality: the Oslo model - till 14.11.2023

The Oslo rice-pile model is a theoretical model of self-organized criticality (SOC) used to study the behavior of granular materials and avalanche-like phenomena in a simple, discrete system. It was introduced as a variation of the sandpile model, and it's named after the city of Oslo, Norway, where it was developed.

1. Implement the Oslo model using the following algorithm focusing on slopes z_i:

   1) Initialize the system in arbitrary stable configuration $z_i \leqslant z_i^T$, where z_i^T is i-th slope threshold $\in [1, 2]$; 2) Drive the system by adding a grain at left-most site; 3) relax the system for all sites $i = 1 : z_1 \to z_1 - 2, z_2 \to z_2 + 1 | i = 2 \dots L - 1 : z_i = z_i - 2, z_{i\pm1} \to z_{i\pm1} + 1 | i = L : z_L \to z_L - 1, z_{L-1} \to z_{L-1} + 1$. During relaxation do not forget about choosing randomly new threshold $z_i^T \in [1, 2]$ for relaxed site. Continue relaxation until $z_i \leqslant z_i^T$ for all i; 4) return to point 2).
2. Plot scaled avalanche size s/s_{max} in function of time t (measured in terms of grain additions). Does it makes sense to analyze data for small t? Which condition should be satisfied in avalanche size statistical analysis?
3. Plot in log-log scale avalanche size probability $P(s, L)$ with respect to avalanche size s for several lengths of the system (L should be at least 64). Do you observe power law behavior? Why this power law breaks for large s?

3 Percolation - till 28.11.2023

Before you start working on this task, I recommend checking Introduction to Computational Physics (ICP). All of the algorithms listed below are described in ICP.

Write a program for the site percolation problem on the square lattice $L \times L$. Just to remind you, in the site percolation problem, each site of a lattice is occupied independently with probability p. Therefore a single Monte Carlo trial consists of initiating a lattice: for each site you choose a random number $r \sim U(0,1)$ and if $r < p$ then put $A[i,j] = 1$ at a given site, otherwise put $A[i,j] = 0$. After this step you are ready to:

1. check if the path connecting the first and the last row exists (use The Burning Method, check e.g. pages 24-25 in ICP)
2. the maximum cluster size s_{max}
3. distribution of clusters $n(s,p,L)$, where s is a size of a cluster (to find clusters use the Hoshen-Kopelman (HK) algorithm, check e.g. pages 28-31 in ICP)

Using Monte Carlo simulations, which should consist of T trials generate the output files for:

1. The probability P_{flow} that the path connecting the first and the last row exists as a function of p, where $p = p_0 : dp : p_k$.
2. The average size of the maximum cluster $\langle s_{max} \rangle$ as a function of p.
3. Distribution of clusters $n(s,p,L)$ for a given p.

The output file:

- Data described in points 1-2 should be saved in a single file for each set of input parameters with an automatically generated name according to the following scheme $'Ave\_L' + L + 'T' + T + '.txt'$ and should consists of 3 columns separated by double space in the following order: $p \;\; P_{flow} \;\; \langle s_{max} \rangle$, something like (artificial values are given here):

  0.1 0 7
  0.2 0 8
  0.3 0 7
  0.4 0 9
  0.5 0 10
  0.6 1 100

- The output file with the distribution of clusters $n(s,p,L)$ should have an automatically generated name according to the following scheme $'Dist\_p' + p + 'L' + L + 'T' + T + '.txt'$ and should consists of 2 columns separated by double space: $s \;\; n(s,p,L)$.

To collect output data you will need to introduce input data:

1. L: the linear size of the system,
2. T: number of trials,
3. p_0: minimum value of p in the loop,
4. p_k: maximum value of p in the loop,
5. dp: step with which p changes inside of the loop.

All this data should be saved in *perc_ini.txt* and used to run the program (program should load this data as an input). User changes data inside the file. This is much more convenient for the user than introducing each time data from the keyboard, especially if the user wants to change only one parameter. Example of such a file:

```
100     % L
1000    % T
0.01    % p_0
1       % p_k
0.01    % dp
```

3.1 Presentation of results

After collecting appropriate data you should present them properly.

1. Using output files create appropriate figures (in Matlab, Python, Gnuplot - you can choose yourself). Remember to label axis! Use different symbols to distinguish between series of data and add appropriate legend. Remember all figures should be readable also in Black&White that is why except of colors we use also different symbols. This is recommendation of all prestigious journals - ask yourself what is the reason for this. Here is the list of figures you should prepare. For points (b)-(d) use $T = 10^4$, if it is not possible use $T = 10^3$ (-2 points) or $T = 10^2$ (-4 points).
   (a) Visualize sample configurations for $L = 10$ and 3 values of $p = 0.4, 0.6, 0.8$ within two methods: (1) use the burning algorithm and describe each site by the number, as shown during the lecture, (2) use the HK algorithm and color each cluster with the different color.
   (b) Probability P_{flow} that the path connecting the first and the last row exists as a function of p for $L = 10, 50, 100$ (use legend).
   (c) The average size of the maximum cluster $\langle s_{max} \rangle$ as a function of p for $L = 10, 50, 100$ (use legend).
   (d) Distribution of clusters $n(s, p, L)$ for a given $p = 0.2, 0.3, 0.4, 0.5, p_c = 0.592746, 0.6, 0.7, 0.8$ (use legend). You can also prepare a figure with 3 subplots: for $p < p_c$, $p = p_c$ and $p > p_c$ (see Figure 2.10 in ICP).
2. Include all figures with appropriate captions in a single pdf file. The file, named using date and your name, e.g. "20210315KatarzynaWeron", should contain the following information:
   (a) title, e.g. "Site percolation on the square lattice",
   (b) name of the author, e.g. "Katarzyna Weron",
   (c) date, e.g. 15.03.2021,
   (d) all figures with captions,
   (e) please remember to write concise introduction, present results and discuss them.
3. Please send the report, code and input/output files from which figures were generated.
4. (Extra 1:) Use LaTex (e.g. Overleaf) to prepare the report.
5. (Extra 2:) Analyze numerically cluster size distributions for 3 different p: 1) $p < p_c$ in which we have scaling law given by $n_{p<p_c}(s) = A_1 s^{-A_2} \exp(A_3 s)$; 2) $p \approx p_c$ for which we should have $n_{p \approx p_c}(s) = A_4 s^{-A_5}$; 3) $p > p_c$ where scaling law is $n_{p>p_c}(s) = A_6 \exp\left(-A_7 s^{\left(1-\frac{1}{A_8}\right)}\right)$. Find numerical fits for parameters A_1 to A_8.

4 Complex networks - till 22.12.2023

1. Visit the following sites and check what kind of data can you acquire and what kind of visualization can you make:

   - `http://networkrepository.com/`
   - `http://snap.stanford.edu`
   - `http://www-personal.umich.edu/~mejn/netdata/`

   Download a chosen data set and draw as a network.

2. Go to Stanford Large Network Dataset Collection by Jure Leskovec (`https://snap.stanford.edu/data/index.html`) and download data for the social circles from Facebook (ego-Facebook). Calculate for this network:

   (a) Degree distribution $P(k)$ and an average degree $< k >$.

   (b) Distribution of clustering coefficients and an average clustering coefficient.

   (c) Distribution of the shortest paths, the diameter and the average path length.

3. Create the following random graphs. Implement at least one by yourself, for the rest you can use some libraries (e.g. NetworkX). Check online `http://networksciencebook.com`:

   (a) **Erdös-Rényi model** $G(N, L)$, in which N vertices are connected with L randomly placed edges.

   (b) **Erdös-Rényi-Gilbert model** $G(N, p)$, in which each pair $(i, j), i, j = 1, ..., N$ of vertices is connected (there is an edges between them) with probability p.

   (c) **Watts and Strogatz model** $WS(N, k, \beta)$. Within this model we start from the regular graph of size N, usually a ring of nodes, i.e. one-dimensional lattice with periodic boundary conditions. Each node is initially linked to its k neighbors, i.e. for the ring and $k = 2$ only to the nearest neighbors (nn), for k to the nn and next nearest neighbors (nnn), and so on. The same can be done for any other regular graph like the square lattice, but here use the most typical structure, namely the ring with $k = 4$. With probability β each link is rewired to a randomly chosen node.

   Calculate and plot for these graphs the following properties and compare them between the models:

   (a) Degree distribution $P(k)$ and an average degree $< k >$.

   (b) Distribution of clustering coefficients and an average clustering coefficient.

   (c) Distribution of the shortest paths, the diameter and the average path length.

   For what values of parameters does it make sense to compare these models?

In the report: 1) Present network from point 1; 2) Show network from point 2, describe method used for calculations and show results from points 2 (a-c); 3) Implement ONE model of random network in Python and show results from points 3 (a-c). (Extra 1): Explain and implement algorithm for shortest path (e.g. Dijkstra algorithm) (Extra 2): Study all 3 types of random graphs from point 3.

5 Boids flocking model - till ...

The aim of this project is to create a visualization of the Boids flocking model, developed by computer scientist Craig Reynolds in 1986. This model simulates the coordinated movement of birds in a flock and has been utilized in various computer games and films. We will not be gathering data or performing any computations. We will simply create a visually appealing animation.

The model is based on three simple rules:

1. Collision Avoidance: avoid collisions with nearby flockmates

2. Velocity Matching: attempt to match velocity with nearby flockmates

3. Flock Centering: attempt to stay close to nearby flockmates

You can learn more about the model from the original paper https://team.inria.fr/imagine/files/2014/10/flocks-hers-and-schools.pdf. You can find several implementations of this model on GitHub. The model is also implemented in NetLogo - check "Flocking" in Models Library.

Implement the model yourself and check the following:

1. What happens if you change the order of 3 rules given above. What is the "correct order"? Is it possible to answer this question?

2. What happens if birds can only see in front of them and what if they see all around?

6 Simulations of the lattice - till ...

1. Implement a square lattice $L \times L$ – the most convenient is to represent it just by an array $A[i, j], i, j = 1, \ldots, L$. Fill up an array: '1' (dog) with probability p or '0' (empty) with probability $1 - p$. Save it to file and then draw - denote dog by the gray color and empty cell by the white color. Parameters p, L should be loaded from the initialization file ini.txt.

2. Use the array from the previous task and now set a flea on the very left dog in the first row on the lattice – you can set '2' for a cell visited by the flea. Flea starts to jump randomly from dog to dog but it is able to jump only a distance of 1, so to one of four adjacent cells occupied with dogs. Each cell visited by the flea is marked with '2' and stays '2' for ever. Introduce parameter t denoting time – one unit time denotes one jump. After time t save it to file and then draw - denote dog by the gray color, empty cell by the white color and flea by the black color. Parameters p, L should be loaded from the initialization file ini.txt.

   (a) Above description of the jump is not precise and it can be implemented in at least two ways. Do you see it? Think about it, discuss it and choose one option to implement.

   (b) Play with it for a different values of parameters. You can even implement simple animation to see how the flea jumps. Can you draw any conclusions from this task?

3. Implement one of two cellular automata: (1) one-dimensional Wolfram Cellular Automata or (2) Game of Life.