Bioinformatics III

Prof. Dr. Volkhard Helms
Markus Hollander, Thorsten Will, Nicolas Künzel,
Andreas Denger, Dr. Pratiti Bhadra
Winter Semester 2019/2020

Saarland University
Chair for Computational Biology

Exercise Sheet 2

Due: November 1, 2019 13:15

Submit your solutions on paper, hand–written or printed at the *beginning* of the lecture or in building E2.1 room 309. Alternatively, you can send an email with a single PDF attachment to markus-hollander@web.de. Your PDF should include commented code listings for programming exercises. Additionally, hand in a .zip file with your source code via email.

2 Scale–Free and Interaction Networks, Fourier Transform

We continue to evolve the classes from the first assignment. The assignment of this week deals with scale–free networks, characterising network structure and real data on protein–protein interaction networks. Additionally, we will have a closer look at the Fourier transform convolution theorem.

Exercise 2.1: Scale–Free Network (45 Points)

First, we construct a scale–free network according to the Barabási–Albert model. Then, we examine the degree distribution of such networks and determine some characteristics in comparison to random networks. Finally, we try to fit the degree distribution to a theoretical distribution.

(a) Implement the Barabási–Albert algorithm for setting up scale–free networks in the initialisation method of the **ScaleFreeNetwork**–class that inherits basic functionality from the **AbstractNetwork**–class.

Given the number of nodes in the network n and the edge parameter m, start by adding three nodes and fully connecting them. Iteratively add the remaining nodes to the network:

i. Add the new node to the network.

ii. Establish m undirected edges from the new node to nodes that are already in the network. Existing nodes i with current degree k_i are selected for the new edges with probability:

$$p_i = \frac{k_i}{\sum_i k_i}.$$

To obtain a much faster implementation and full points, think of a method that avoids recomputing the node probabilities from scratch every time you want to add a new edge.

(b) First, create two scale–free networks with $n = 10,000$ and $n = 100,000$ nodes, respectively. In both networks add $m = 2$ edges for each new node. Plot the degree distribution of both networks with logarithmic axes by using a new, pre–implemented plotting function in **tools.py**.

Next, create a random network with $n = 10,000$ nodes and $m = 20,000$ edges. Plot its degree distribution together with the degree distribution of the first scale–free network, again with logarithmic scale.

Implement your solution in the function **exercise_1b()** in **main.py**.

Questions: What differences can you observe between the degree distributions of the two scale–free networks? What are the major differences between degree distributions of the scale–free and the random network?

(c) The degree distribution of a scale–free network follows a power law, which has the form $P(k) \sim k^{-\gamma}$, where k is the node degree and γ the slope. Try to fit this theoretical distribution to the degree distribution of a random network using the Kolmogorov–Smirnov distance as outlined below:

i. Implement the function **scale_free_distribution(**max**,** γ**)** in **tools.py** that first computes the power–law histogram $h_k = k^{-\gamma}$ for $k \in \{0, \ldots, max\}$ and then normalises it by dividing each histogram entry by $c = \sum_k h_k$.

ii. Implement the function **cumulative(**$dist$**)** in **tools.py** that computes the cumulative distribution of a given probabilistic distribution $dist$.

iii. Implement the Kolmogorov–Smirnov distance between two histograms A and B of length n in the function **KS_dist(**A**,** B**)** in **tools.py**. The KS distance of two distributions is the maximal distance between their respective cumulative distributions F:

$$D = \max_i |F_A(i) - F_B(i)|$$

Thus, first compute the cumulative distributions of the input histograms, then find the position where the cumulative distributions deviate the most and return this distance.

iv. Use the KS distance to determine a γ between 1.0 and 3.0 (advance in 0.1 steps) that fits best to the degree distribution of a scale–free network with $n = 10,000$ nodes and $m = 2$ new edges per iteration. Plot the empirical scale–free distribution and the theoretical power law distribution with optimal γ, again with logarithmic scale.

Implement your solution in the function **exercise_1c()** in **main.py**.

Question: How good is your fit? How could it be improved?

Exercise 2.2: Real Interaction Networks (45 Points)

In this exercise we will implement a **BioGRIDReader** which will help us process real interaction data. BioGRID (Biological General Repository for Interaction Datasets) is a protein interaction database which, in version 3.4.159 (March 2018), contains data of 1,548,143 raw protein and genetic interactions from major model organism species compiled from 64,826 publications.

The supplement contains this release as a tab–separated file (**BIOGRID-ALL-3.4.159.tsv**). The beginning of the file contains some information about the format. Additionally, you can find a mapping of NCBI taxon IDs to organism names in **mapping.txt**.

(a) Implement the initialisation method of the **BioGRIDReader**–class in **reader.py**. It should parse the input file and store the necessary data in a data structure that simplifies later queries. For every organism found in the file as **NCBI taxon identifiers**, one should be able to easily retrieve all interactions as pairs of **official gene symbols**. Skip entries where the organism is inconsistent or where the interacting proteins are the same. Avoid interaction duplicates.

(b) Implement the function **network_size(**id**)** that returns the number of interactions in the network of a give NCBI taxon ID.

Question: How big is the human interaction network?

(c) Implement the function **most_abundant_taxon_ids(**n**)** that returns the n organism taxon IDs with the most interactions and the respective number of interactions.

Question: What are the $n = 5$ most abundant organisms and why is that not surprising?

(d) Implement the **GenericNetwork**–class in **generic_network.py** that imports networks from files and inherits basic network functionality from the **AbstractNetwork**–class. Then implement the function **export_network(**id**,** $file$**)** in the **BioGRIDReader**–class that creates an organism–specific network file that can be used by the **GenericNetwork**–class.

(e) Compute the 10 proteins with the highest degree in the human interaction network. Implement your solution in the function **exericse_2e()** in **main.py**.

Question: What are the names and degress of these proteins? Take one of them as an example and briefly explain the biology behind the high connectivity.

(f) Build a network for human interaction data, and then determine and plot the corresponding degree distribution. Discuss if the distribution behaves more like a scale–free or a random network. Implement your solution in the function **exericse_2f()** in **main.py**.

Exercise 2.3: Fourier Transform (10 Points)

As shown in lecture 3 (slides 17+), the convolution $(f \circledast g)$ of a high–resolution atomic structure f and a blurring kernel g can be used to obtain a low–resolution representation of the structure that can be compared to experimental, low–resolution EM images. The convolution is defined as follows:

$$(f \circledast g)(x) = \int f(y)g(x-y)\ dy$$

Fast Fourier transform can speed up the search for the best fit of atomic structure and EM image. The continuous Fourier transform FT of a function $f(x)$ is defined as:

$$FT[f(x)] = F(k) = \int f(x)e^{-ikx}dx$$

Prove the convolution theorem given in the lecture (slide 23):

$$FT[(f \circledast g)(x)] = FT\left[f(x)\right] \cdot FT\left[g(x)\right]$$

Give a brief explanation of what you are doing in each step of your proof.

Have fun!