Threshold based reconstruction in a network of randomly coupled chaotic electronic oscillators

Isak Ederlöv

April 20, 2023

1 Background and objectives

In this report, data from a network of randomly coupled chaotic electronic oscillators will be analyzed, with the goal to evaluate the effectiveness of reconstruction of the network based solely on measurement data. Threshold based reconstruction of a network is based on data of connection strengths between the nodes in the network. By calculating some decision metric between the different nodes one can infer the probability of coupling between them. In this report, the Pearson coefficient (ρ) will be used as the thresholding parameter.

1.1 The Pearson correlation coefficient

The Pearson correlation coefficient is a statistical measure that quantifies the degree of linear association between two variables. In network reconstruction, it can be used as a similarity measure to determine the strength of the relationship between pairs of nodes in a network. The range of the Pearson coefficient is $[-1, 1]$, where a higher score indicates a stronger positive correlation. The Pearson coefficient will be computed for all pairs of nodes in a network, and then a threshold will be set to determine which pairs of nodes should be connected based on their similarity.

Mathematically, the Pearson coefficient is in essence the cross-correlation with no lag. It is important that the data to be analyzed is normalized to zero mean and unit variance.

$$\rho = |C_{xy}(0)| = \frac{1}{N}\sum_{k=1}^{N} x(k)y(k) \tag{1}$$

2 Methodology

2.1 Representation of the network

The data, which was stored in an adjacency matrix was imported using the Python framework *Pandas* [pdt20]. The data frame was reshaped into separate collections of nodes to represent their connections. To visualise, the graph tool in the package *networkx* [HSSC08] was utilised.

```
adj_mat = pd.read_csv('./ex5_files/Topology.dat', delim_whitespace=True, header=None)
rows, cols = np.where(adj_mat == 1)
edges = zip(rows.tolist(), cols.tolist())

gr = nx.Graph()
gr.add_edges_from(edges)

plt.title("Network connections")
pos = nx.spring_layout(gr, k=2.2) # Increase k value to increase spacing between nodes
nx.draw(gr, node_size=400, with_labels=True, pos=pos)
plt.show()
```

2.2 Extraction and calculation of the Pearson coefficient

A function was defined to be used throughout the report. Equation 1 was implemented into the function as follows.

```
def pearson(s1, s2) -> float:
    N = len(s1)
    return 1/(N) * abs(sum([s1[k] * s2[k] for k in range(1, N)]))
```

To proceed, the coupling strength data for each measurement was to be extracted. Because of the size of the data set and the number of correlations, some optimization of the trivial algorithm was in place to reduce the computational load.

With the knowledge that the points on the main diagonal represent "self-coupling" ($\rho = 1$), together with the fact that equation 1 is commutative, the matrix must be symmetrical and 1 on the main diagonal. The size of the domain of the function can thus be reduced from n^2 to $\frac{n(n+1)}{2}$. While the time and space complexity of the algorithm remains $O(n^2)$, the amount of computations is reduced by at least 50% for the given input using the triangular method.

```
    def pearson_table_from_file(number: int):
    file_path = f"./ex5_files/TS_{str(number).zfill(2)}.dat"
    db = np.transpose(np.loadtxt(file_path))
    size = np.size(db, 0)
    table = np.empty((28,28))
    db1 = [(db[i] - np.mean(db[i]))/np.std(db[i]) for i in range(size)] #Standardizes the time
    ↪  series data column-wise
    column_indices = np.hstack([[i for i in range(0,j)] for j in range(1,28)])
    row_indices = np.hstack([[i]*i for i in range(1,28+1)])
    for row, col in zip(row_indices, column_indices):
        table[row, col] = pearson(db1[row], db1[col])
    return table
    #Combinatorically calculates the pearson coefficient and returns the lower triangle of a 28*28
    ↪  matrix
```

This function generates a data set that can be visualised in a heat map from one measurement. The full data set consisted of 25 different measurements, and thus the previous process was repeated for all measurements. Finally, all measurements were then merged using mean assimilation. This process is akin to calculating 25 x-y heat maps stacked on top of each other in the z-axis and then "flattening" this stack by averaging the cell values along the z-axis.

```
all_raw_data = [pearson_table_from_file(k) for k in range(25)]
heatmap_data = np.mean(all_raw_data, axis=0)
```

2.3 Thresholding and reconstruction of the network

Thresholding the data is mainly a question of deconstructing the computed data set into a sortable container, deciding a threshold, and reconstructing it back to its original shape. In this case, the triangular region of the heat map data is the target.

```
# Flatten the matrix and sort in descending order
flattened = heatmap_data.flatten()
sorted_indices = np.argsort(flattened)[::-1]  # Reverse order to get highest values first

num_elem = 27*28/2 # Equal to 1+2+3+...+26+27
num_top_values = int((num_elem * 0.1))

# Get the indices of the top 10% highest values
top_10p_indices = np.unravel_index(sorted_indices[:num_top_values], heatmap_data.shape)
```

With the new data set, the inverse of the previous snippet can be implemented.

```
removed_heatmap_data = np.copy(heatmap_data)
row_indices = np.hstack([[i]*i for i in range(1,28+1)])
column_indices = np.hstack([[i for i in range(0,j)] for j in range(1,28)])
for row, col in zip(row_indices, column_indices):
    if [row, col] not in list(map(list, zip(*top_10p_indices))):
        removed_heatmap_data[row][col] = 0

# Reshape the data to again fit the triangular region
shaped_data = (np.tril(removed_heatmap_data) + np.triu(adj_mat)*0.5)
```

3 Results

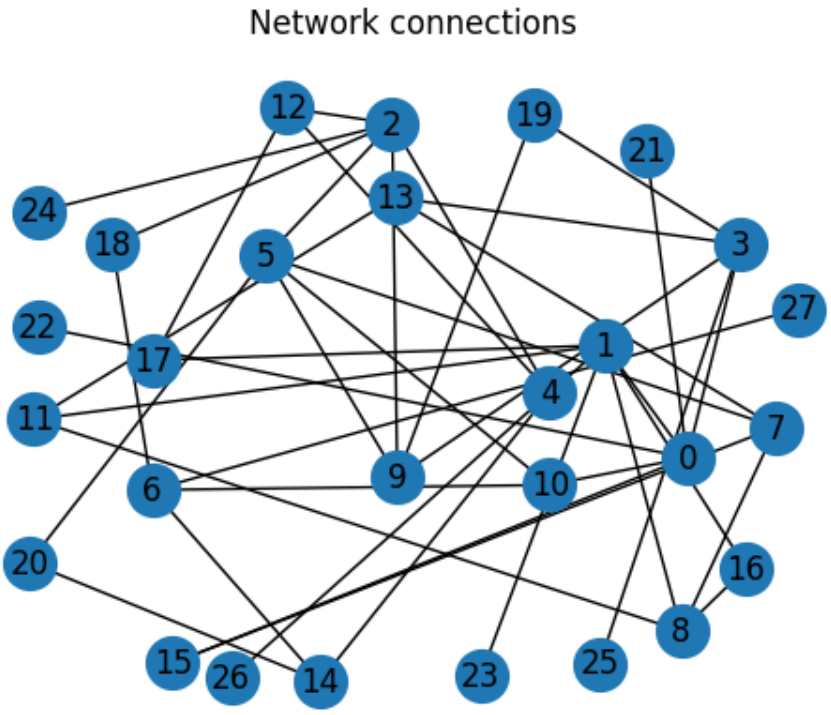

Figure 1: Network graph showing the connections of the nodes to each other.

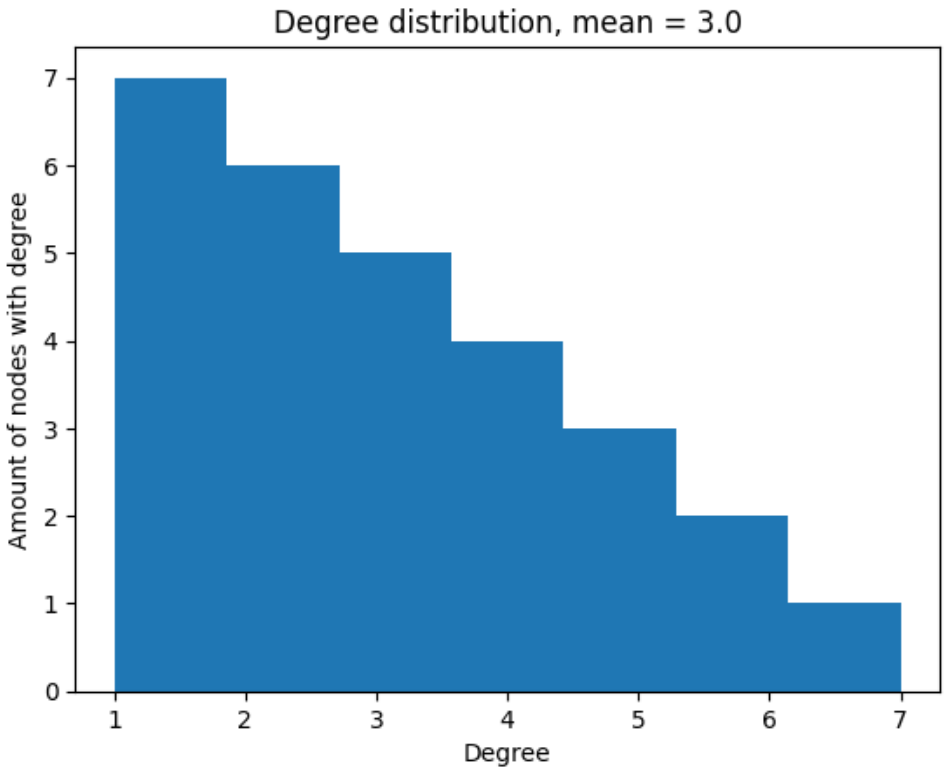

Figure 2: Shows the distribution of degrees.

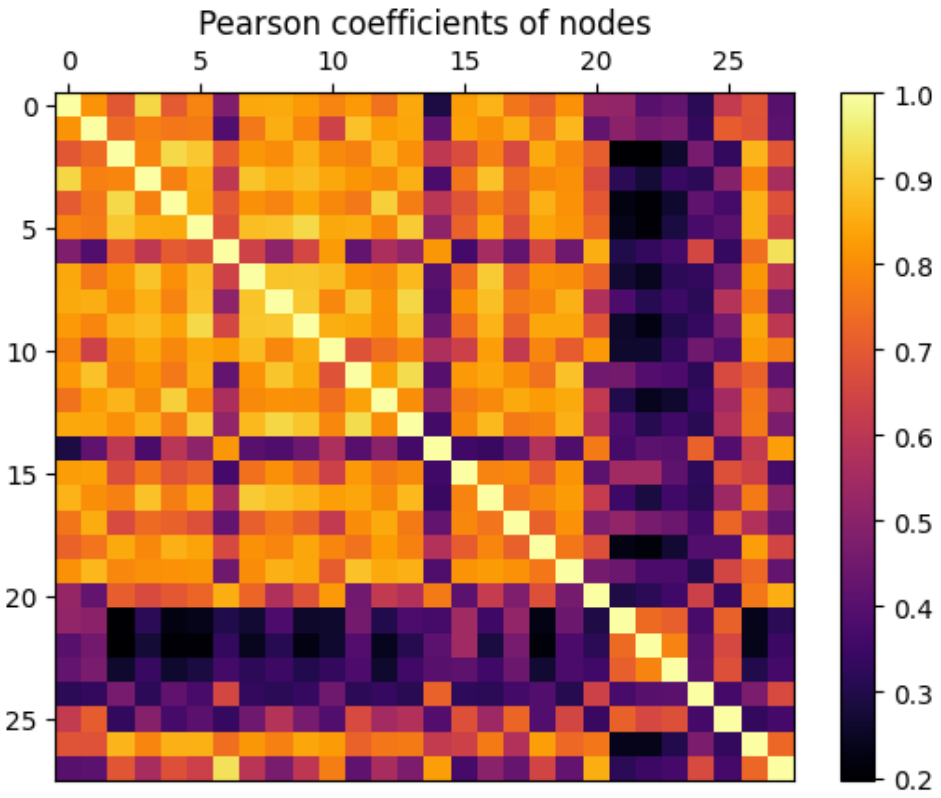

Figure 3: Heat map showing values of ρ between node pairs.

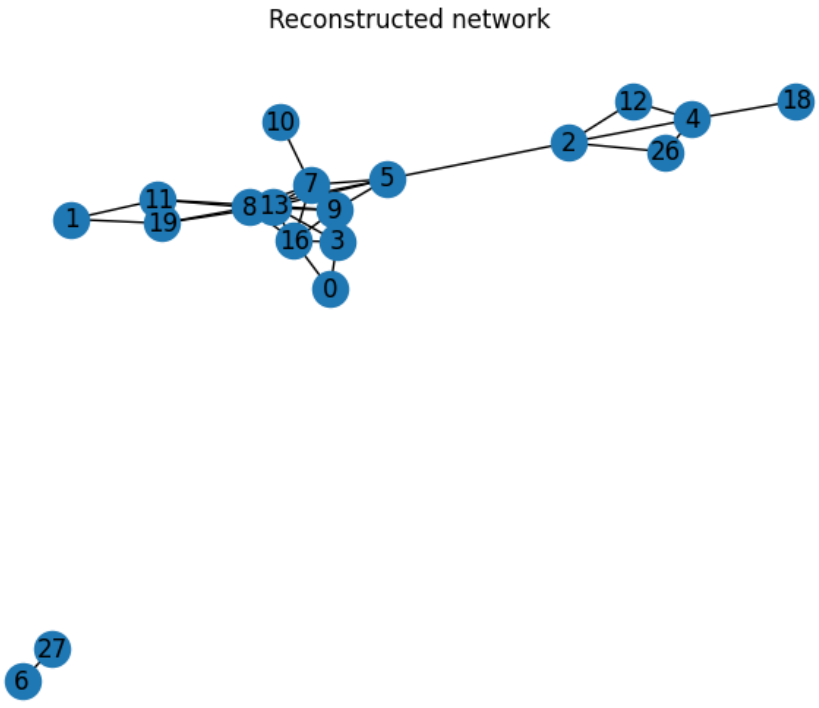

Figure 4: Shows the network reconstructed from the 10% strongest connections.

Figure 5: Heat map showing the reconstructed network in the bottom left half, and the actual coupling in the upper right.

By "folding" over figure 5 along the main diagonal and visualising different overlap with different colors, a "confusion heat map" is obtained.

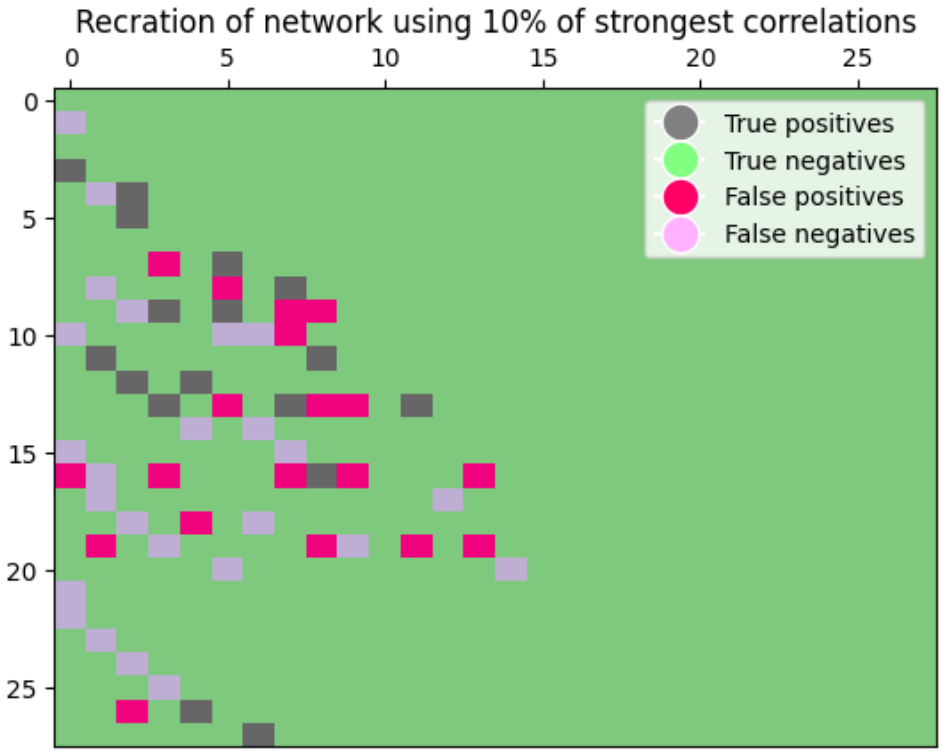

Figure 6: Showing a sample of the laser intensity time series. The upper right half is redundant and thus removed.

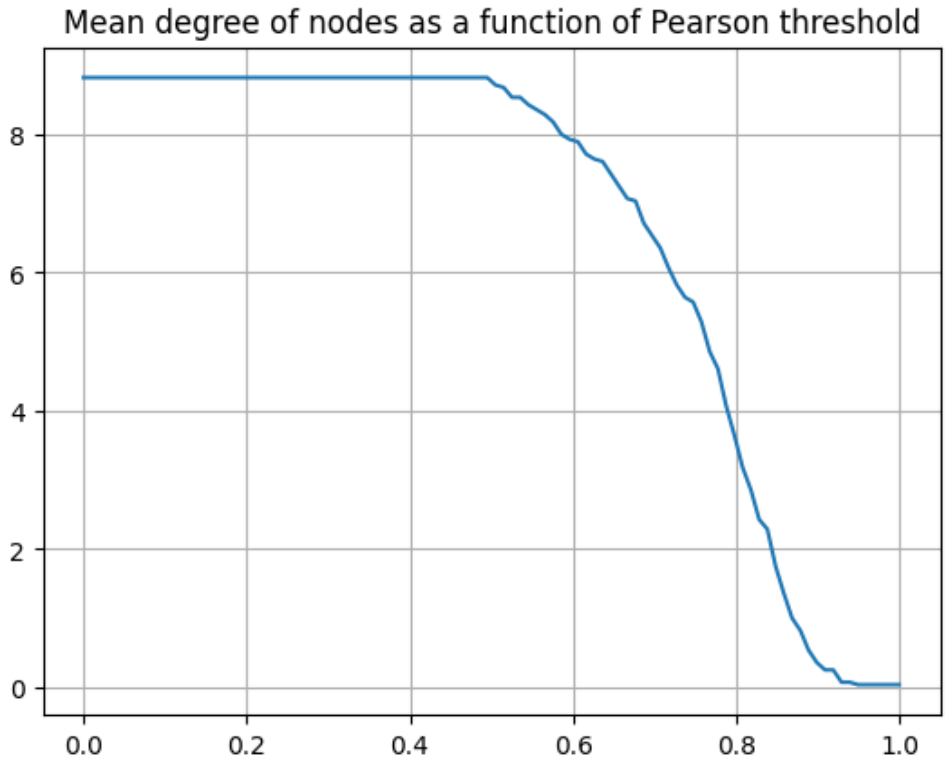

Figure 7: Showing a sample of the laser intensity time series. The upper right half is redundant and thus removed.

4 Discussion

The results seem to indicate that the Pearson correlation does not offer good predictive capability for connectivity in a network of randomly coupled electronic oscillators. Weak but significant links are missed and the hit rate is unsatisfactory even with more conservative thresholds. The problem may partly lie in the fact that the Pearson coefficient assumes jointly normally distributed data, that follows a bi-variate normal distribution. While not a lot is known about the source of the data, it is probable that a network of oscillators that are chaotic does not generate data that can be considered Gaussian.

Given how my understanding of what I was doing formed during the process of doing it, I did not have a lot of specific expectations. The validity of the analysis might be limited because of my lack of experience and knowledge in the field, which may impede the accuracy and reliability of the findings.

5 Summary

This report analyzes the effectiveness of network reconstruction using threshold-based reconstruction of a network of randomly coupled chaotic electronic oscillators based on data of connection strengths between nodes in the network. The Pearson correlation coefficient is used as a similarity measure to determine the strength of the relationship between pairs of nodes in a network. The report details the methodology employed, which includes the extraction and calculation of the Pearson coefficient and thresholding of the data. The results show how reconstruction of the network affect the connections, degrees of the nodes and sensitivity and specificity of the reconstructed network. On a brief evaluation of the data, it is found that the Pearson correlation coefficient may not posses good predictive capability for reconstruction of networks.

References

[HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los

Alamos, NM (United States), 2008.

[pdt20] The pandas development team. pandas-dev/pandas: Pandas, February 2020.