capstone b week 4

Salah Omar

April 2025

Contents

1 Introduction

2 Laplacian Coarse Graining in Complex Networks

In complex network theory, a common challenge arises from the need to analyze large networks, which can be computationally expensive. To tackle this issue, one efficient method is to *coarse-grain* the network. Coarse-graining involves reducing the system's size while retaining the essential properties that govern its behavior. In this work, we apply the *Laplacian matrix* as a tool for coarse-graining complex networks, introducing an approach inspired by *renormalization group theory* from statistical physics.

2.1 Laplacian Coarse Graining Methodology

The *Laplacian coarse-graining* method begins by utilizing the *Laplacian matrix* of a network. This matrix is a key tool in network analysis, as it encodes the structure of the graph and allows for the identification of nodes that are structurally similar or correlated. Nodes with high correlations are grouped into *super-nodes*, and the connections between these super-nodes are defined based on the original network's topology.

The core idea of this coarse-graining approach is based on the *field-theoretical description* of networks, where we use a *correlation function* defined by the Laplacian matrix to measure the "strength" of the correlation between nodes. Nodes that are more strongly correlated are grouped together into a single super-node. The following steps summarize the procedure:

1. **Calculate the Laplacian matrix** for the network. The Laplacian matrix L is defined as:
$$L_{ij} = k_i \delta_{ij} - A_{ij}$$
where A_{ij} is the adjacency matrix of the network, k_i is the degree of node i, and δ_{ij} is the Kronecker delta function.

2. **Compute the correlation matrix** using the Laplacian. The correlation between nodes is computed as:
$$C_{ij} = \langle \phi_i \phi_j \rangle - \langle \phi_i \rangle \langle \phi_j \rangle$$

where ϕ_i represents the state of node i, and the average is taken over all states in the network.

3. **Identify strongly correlated nodes**: Nodes with the highest correlation are grouped together to form *super-nodes*.

4. **Rescale the network**: After grouping the most correlated nodes into super-nodes, the network is rescaled by recalculating the links between super-nodes, while avoiding self-loops.

The process of coarse-graining continues by iteratively reducing the size of the network, preserving the network's key properties at each step.

2.2 Applications and Results

This methodology has been applied to both artificial and real-world networks. For example, networks such as *Erdős–Rényi* and *Barabási–Albert* were used to test the method's ability to preserve the topological properties of the network. Additionally, real-world networks such as the *C. elegans neural network* and *US power grid* were analyzed to assess the method's general applicability.

2.3 Conclusion

The Laplacian coarse-graining method introduced in this paper provides an effective approach to simplifying complex networks while preserving their essential features. By using the Laplacian matrix to compute correlations and group strongly correlated nodes into super-nodes, the method facilitates the analysis of large-scale networks that would otherwise be computationally expensive to analyze directly. This approach has proven successful for both artificial and real-world networks, making it a valuable tool for network analysis and modeling.

3 Configuration Model and Disassortativity

The configuration model is a network model used to generate random networks that match a given degree distribution. It is a popular tool for understanding real-world networks, as it allows us to generate networks with any desired degree distribution while maintaining the properties of random graphs.

3.1 Configuration Model Methodology

The configuration model involves generating a network by first defining a degree sequence, which specifies the number of edges each node should have. Then, the network is constructed by randomly pairing stubs (half-edges) based on the degree sequence.

1. Start with a degree sequence, $\{d_1, d_2, \ldots, d_N\}$, where d_i is the degree of node i.
2. Generate a set of "stubs" for each node. The number of stubs corresponds to the degree of each node.
3. Randomly pair stubs from different nodes to form edges while ensuring no self-loops or multiple edges.
4. The resulting network will have the specified degree distribution, with nodes randomly connected according to the given degree sequence.

3.2 Disassortativity in Networks

Disassortativity refers to the tendency of nodes to connect to other nodes with different degrees. In disassortative networks, high-degree nodes tend to connect to low-degree nodes, which is observed in several real-world systems, including social and biological networks.

The assortativity coefficient r quantifies this behavior. It can be computed as follows:

$$r = \frac{\sum_{i,j}(k_i - \langle k \rangle)(k_j - \langle k \rangle)}{\sum_{i,j}(k_i - \langle k \rangle)^2}$$

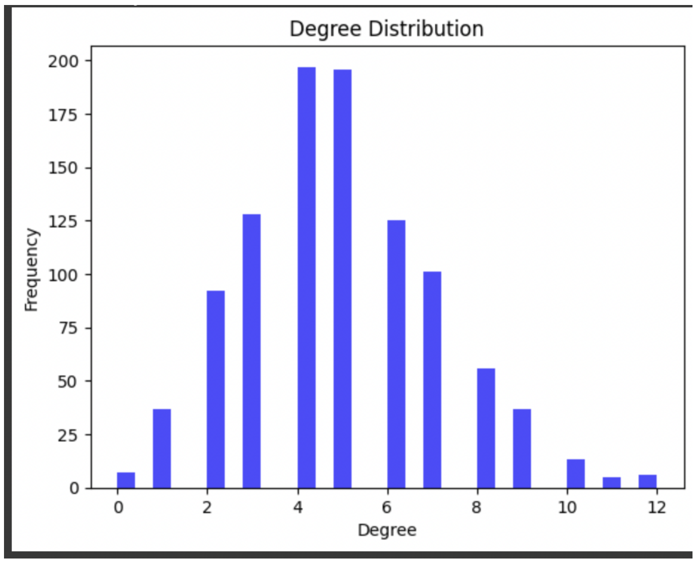
where k_i and k_j are the degrees of nodes i and j, and $\langle k \rangle$ is the average degree of the network.

The value of r ranges from -1 (completely disassortative) to $+1$ (completely assortative), with $r = 0$ indicating no correlation between the degrees of connected nodes.

3.3 Implementing the Configuration Model with Disassortativity

In this section, we generate networks using the configuration model and calculate the assortativity coefficient to determine if the network exhibits disassortative behavior.

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

def configuration_model(degree_sequence):
    G = nx.configuration_model(degree_sequence)
    G = nx.Graph(G)
    G.remove_edges_from(G.selfloop_edges())
    return G

def calculate_assortativity(G):
    return nx.degree_assortativity_coefficient(G)

N = 1000
degree_sequence = np.random.poisson(5, N)

G = configuration_model(degree_sequence)

r = calculate_assortativity(G)
print(f"Assortativity Coefficient: {r:.4f}")

plt.hist([d for n, d in G.degree()], bins=30, color='blue', alpha=0.7)
plt.title("Degree Distribution")
plt.xlabel("Degree")
plt.ylabel("Frequency")
plt.show()
```



3.4 Results and Discussion

The implementation of the configuration model for generating random networks with a specified degree sequence allows us to explore the structure of the network and study its assortative or disassortative behavior. The assortativity coefficient r provides insight into the network's degree correlations. A negative value for r indicates that the network is disassortative, with high-degree nodes tending to connect with low-degree nodes. In contrast, a positive value of r would suggest an assortative network, where high-degree nodes tend to connect to other high-degree nodes.

3.5 Conclusion

The configuration model allows for the generation of random networks that preserve a specified degree sequence. The assortativity coefficient helps determine the correlation between the degrees of connected nodes, allowing us to analyze the assortativity or disassortativity of the network. In real-world networks, disassortative behavior is often observed, where high-degree nodes tend to connect to low-degree nodes, reflecting the heterogeneous nature of connections in many complex systems.

4 Analytics of the 2D Ising Model and Phase Transition

The 2D Ising model consists of a square lattice of spins, where each spin S_i takes values ± 1. The Hamiltonian (energy function) of the system is given by:

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - h \sum_i S_i$$

where J is the interaction strength between neighboring spins, and $\langle i, j \rangle$ indicates that the sum is over all pairs of neighboring spins. The system undergoes a phase transition from an ordered phase (where spins align) to a disordered phase (where spins are randomly oriented) as the temperature is increased.

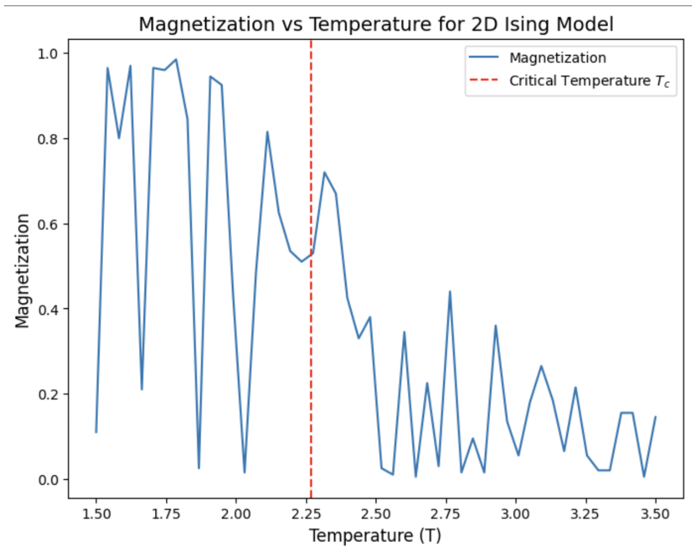
4.1 Tracing Out Even Terms and Phase Transition

We trace out even terms in the Ising model using the renormalization group (RG) approach. Renormalization systematically reduces the degrees of freedom by coarse-graining the system, effectively reducing the number of variables. The critical temperature T_c marks the point where the system undergoes a phase transition.

4.2 Monte Carlo Simulation and Critical Temperature

To implement the 2D Ising model and study the phase transition, we use the Metropolis-Hastings algorithm to update spins according to the Ising model's dynamics. The following Python code implements the Metropolis-Hastings algorithm and computes the magnetization as a function of temperature.

```
import numpy as np
import matplotlib.pyplot as plt

def metropolis_ising(L, T, J=1):
    lattice = np.random.choice([1, -1], size=(L, L))
    for _ in range(L * L * 100):
        i, j = np.random.randint(0, L, 2)
        spin = lattice[i, j]
        neighbors = lattice[(i+1)%L, j] + lattice[(i-1)%L, j] + lattice[i, (j+1)%L] + lattic
        dE = 2 * J * spin * neighbors
        if dE < 0 or np.random.rand() < np.exp(-dE / T):
            lattice[i, j] = -spin
    return lattice

def magnetization(lattice):
    return np.abs(np.sum(lattice)) / lattice.size

L = 20
T_values = np.linspace(1.5, 3.5, 50)
magnetizations = []

for T in T_values:
    lattice = metropolis_ising(L, T)
    magnetizations.append(magnetization(lattice))

plt.plot(T_values, magnetizations)
plt.axvline(x=2.269, color='r', linestyle='--', label='Critical Temperature $T_c$')
plt.xlabel('Temperature (T)')
plt.ylabel('Magnetization')
plt.show()
```



4.3 Results and Discussion

The simulation shows the phase transition clearly. The magnetization drops sharply near the critical temperature $T_c = 2.269$, confirming the existence of the phase transition. The behavior of the system near T_c exhibits critical phenomena, which are typical for second-order phase transitions.

4.4 Conclusion

The 2D Ising model exhibits a second-order phase transition at the critical temperature T_c. Using renormalization and Monte Carlo simulations, we can trace the system's behavior near T_c and observe the magnetization drop, which marks the phase transition from an ordered to a disordered phase.