Step-by-Step Guide to Implementing the Barabási-Albert Model

Goal

The goal is to simulate a Barabási-Albert (BA) network, starting with a small connected network and gradually adding nodes. Each new node will attach to a fixed number of existing nodes based on the principle of *preferential attachment*, meaning it's more likely to connect to nodes with higher degrees.

Overview of Steps

1. **Set up parameters**: Decide on the total number of nodes `n` and the number of edges `m` each new node will create when added.
2. **Initialize the network**: Begin with `m+1` nodes that are fully connected, as these provide the initial structure.
3. **Add new nodes with preferential attachment**: For each new node, connect it to `m` existing nodes based on their current degree (more edges will be formed with higher-degree nodes).
4. **Use an adjacency matrix**: Track all connections in an adjacency matrix, where `matrix[i][j] = 1` indicates an edge between nodes `i` and `j`.

Step 1: Initial Setup

Choose:

- `n`, the total number of nodes you want in your network.
- `m`, the number of connections each new node will form (this should be smaller than or equal to `m+1`, the number of initial nodes).
- you can experiment with different values of `m`

Step 2: Initialize a Fully Connected Graph for the First `m+1` Nodes

1. Create an empty adjacency matrix of size `n x n` to represent all possible edges.
2. Connect the initial `m+1` nodes by setting `matrix[i][j] = 1` for each pair `(i, j)` among these nodes.
3. Track the degree of each node in a list called `degrees`, where each entry corresponds to the total number of connections for each node. Start by setting the degree of each of the `m+1` nodes to `m`.

Step 3: Add Each New Node with Preferential Attachment

For each new node (from index `m+1` onward):

1. **Compute the Probability Distribution**: To implement preferential attachment, calculate the probability of connecting to each existing node in proportion to its degree. See page 3 for more details on this step.

    - Use the cumulative sum of degrees. If you normalize this cumulative sum (dividing by the total sum), each entry will represent the cumulative probability up to that node.
    - For example, if `degrees = [1, 2, 1, 3]`, compute the cumulative sum and normalize it to get a cumulative probability distribution.
2. **Select `m` Nodes for Connection**:
    - Randomly generate a value between 0 and 1 to represent a point within the cumulative probability distribution. Identify the node associated with that value by finding where it falls in the cumulative sum array.
    - Repeat this `m` times to select `m` unique nodes to connect to. Avoid duplicates by keeping track of already chosen nodes.

Step 4: Update the Adjacency Matrix

1. After selecting the `m` target nodes, add edges between the new node and each target node by setting `matrix[new_node][target] = 1` and `matrix[target][new_node] = 1`.
2. Update the `degrees` list to account for the new connections formed with each target node.
3. Append the new node's degree (equal to `m`) to the `degrees` list.

Step 5: Finalize the Adjacency Matrix

Once all nodes are added, your adjacency matrix will represent the network structure generated by the Barabási-Albert model. The matrix will be symmetric since the network is undirected.

Step 6: Visualize the Network

After constructing the network's adjacency matrix, visualize the Barabási-Albert network to gain further insights into its structure. You'll use NetworkX (`nx`), as introduced in the previous coding session, to create the visualization.

1. **Convert the Adjacency Matrix to a NetworkX Graph**:
    - Use NetworkX's `from_numpy_matrix()` function to create a graph from the adjacency matrix.
2. **Set Up a Spring Layout**:
    - Apply a spring layout (`nx.spring_layout`), which arranges nodes to highlight community structure by using attractive and repulsive forces between nodes. This layout is well-suited for visualizing complex networks like the Barabási-Albert model.
3. **Define Node Properties for Visualization**:
    - **Node Color by Age**: Use the order in which nodes were added (i.e., their "age") to determine their color.You could use a color gradient from older to newer nodes.
    - **Node Size by Degree**: Set the size of each node based on its degree.
4. **Plot the Network**:

- Use `nx.draw()` with the spring layout positions, node colors, and node sizes.
- Include a color bar (if using a gradient for age).

Key Concept: Using Cumulative Sum for Preferential Attachment

The cumulative sum helps simulate preferential attachment by creating a cumulative probability distribution. Here's how to use it:

1. **Calculate the cumulative sum**: For example, if degrees are `[1, 3, 2, 4]`, the cumulative sum becomes `[1, 4, 6, 10]`.
2. **Normalize the cumulative sum**: Divide each element by the total degree sum (in this case, 10) to get `[0.1, 0.4, 0.6, 1.0]`. Each entry now represents the cumulative probability up to that point.
3. **Select a node based on a random value**: Generate a random number between 0 and 1. Identify where this random number would "fit" in the cumulative distribution to select the target node.
    - For instance, a random number of 0.35 falls within the range of `0.1 - 0.4`, meaning you would choose the node with degree 3 as the target.

This cumulative sum approach ensures that nodes with higher degrees are more likely to be chosen, achieving the preferential attachment effect.

Analysis of the Barabási-Albert Network

After building the BA model, analyze its properties with the following steps.

Step 1: Degree Distribution

The degree distribution of a network tells us the probability of finding a node with a given degree. In the Barabási-Albert model, the degree distribution follows a power-law decay, meaning a few nodes have high degrees while most have low degrees.

1. **Plot the Degree Distribution with Linear Binning**:
    - Create a histogram of the node degrees using regular intervals.
    - Plot this histogram on a log-log scale to visualize the power-law nature of the degree distribution.
2. **Plot the Degree Distribution with Logarithmic Binning**:
    - Group the degrees into bins that increase logarithmically (e.g., [1, 2-4, 5-8, etc.]). Logarithmic binning smoothens the plot and is especially useful for networks with a long-tailed distribution.
    - Normalize the counts in each bin dividing the counts by the bin width.

   - Plot this histogram on a log-log scale alongside the linear-binned histogram.
3. **Compare with Theoretical Scaling**:
   - Overlay the theoretical power-law scaling (exponent -3) on the plots to observe how closely the simulated data matches the expected distribution.

Step 2: Clustering and Average Path Length Scaling as the Network Grows

The clustering coefficient and average path length provide insight into the structure of the network. In the Barabási-Albert model, we observe that clustering decreases with the network size, and average path length grows slowly as the network size increases. To examine this, we'll track these properties at select intervals as the network grows.

1. Choose Logarithmically Spaced Intervals:
   - Instead of calculating clustering and path length at each step, choose a few network sizes that are spaced logarithmically. For instance, you might record these properties when the network reaches sizes like 100, 200, 500, 1000, and so on.
   - This approach allows you to see the overall scaling without the computational cost of measuring at every step.
2. Calculate Clustering Coefficient at Each Interval:
   - For each chosen network size, calculate the clustering coefficient for the network in its current state.
   - You can use a NetworkX function to compute the average clustering.
3. Calculate Average Path Length at Each Interval:
   - At each chosen interval, compute the average shortest path length for the network in its current state.
   - Also in this case you can use a NetworkX function.
4. Plot the Scaling of Clustering and Path Length with Network Size:
   - After recording clustering and path length for the selected network sizes, plot these values against network size on a log-log scale.
   - Include the theoretical scaling for random networks and a scale-free network with exponent γ=3 (see lecture slides).

Step 3: Implement Nonlinear Preferential Attachment and Repeat the Analysis

In the standard BA model, preferential attachment is linear, meaning nodes with higher degrees are proportionally more likely to attract new connections. Introducing a nonlinearity allows you to explore how different attachment dynamics affect the network's properties.

1. **Introduce a Nonlinear Attachment Factor**:
   - Instead of making the probability proportional to the degree k, use k□□, where α is a constant. Experiment with values of α (e.g., 0.5, 1.5, etc.).
   - For α=1, you get the original BA model; values greater than 1 favor high-degree nodes more, while values less than 1 reduce this preference.
2. **Simulate the Network with Nonlinear Preferential Attachment**:
   - Repeat the steps to build the network using nonlinear attachment.
   - Compute the degree distribution, clustering, and average path length.