Manipulating Two-Mode (Bipartite) Networks

Simone Santoni

2024-11-26

Synopsis This notebook shows how to read and transforms a two-mode network (e.g., a person-to-team affiliation network) into two one-mode networks (a person-person network - in which individuals are connected when they belong to the same team - and team-team network - in which individuals are connected when they share at least a member)

1 Notebook setup

We are using `NumPy` for data simulation, `NetworkX` for network manipulation, and `Pandas` for data wrangling.

```python
import os
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
from networkx.algorithms import bipartite as bp
import pandas as pd
```

2 Fake data

We simulate an incidence matrix with 100 mode-1/bottom nodes (e.g., individuals) and 10 mode-2/top nodes (e.g., products).

```python
# numpy simulation
n, k = 100, 10
bottom_nodes = np.arange(0, n)
top_nodes = np.arange(n, n+k)
edges = []
for i in bottom_nodes:
    # random number of ties from a poisson distribution
    degree = np.random.poisson(lam=3, size=1)
    # alters
    alters = np.random.choice(top_nodes, size=degree)
    # add edges
    for alter in alters:
        edges.append((i, alter))
```

3 Graph creation

We initialize a graph object and add nodes and edges as per the simulated incidence matrix. As you can see from the below-displayed graph, we pass the nodes in two chunks, one for each mode. Specifically, we distinguish between mode-1/bottom nodes and mode-2/top nodes by setting the `bipartite` attribute to 0 and 1, respectively. Then, we check whether the graph is bipartite, i.e., there are no edges connecting nodes of the same mode.

```
# empty graph
bg = nx.Graph()
# add nodes
bg.add_nodes_from(bottom_nodes, bipartite=0)
bg.add_nodes_from(top_nodes, bipartite=1)
# get nx object
bg.add_edges_from(edges)
# `is bipartite` check
is_bip = nx.is_bipartite(bg)
```

4 Getting network projections

To get the projections of a bipartite network X (N x K) means to compute the following:

$$Y = X \times X^T$$

and

$$Z = X^T \times X$$

where Y is the N x N matrix of the mode-1 nodes and Z is the K x K matrix of the mode-2 nodes. The projections are weighted if the edges of the bipartite network are weighted. In this case, the weights are the number of shared neighbors between two nodes. Let us see how to get the projections of the bipartite network we have just created with `NetworkX`.

4.1 Unweighted projections of the two-mode networks

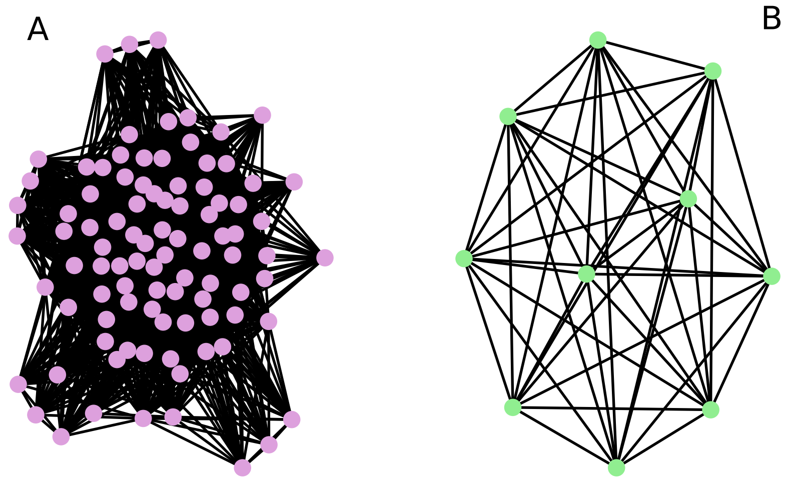
The unweighted projected graph is the projection of the bipartite network `bg` onto the specified nodes. The nodes retain their attributes and are connected in the resulting graph if they have an edge to a common node in the original graph.

```
g_b = bp.projected_graph(bg, bottom_nodes)
g_t = bp.projected_graph(bg, top_nodes)
```

Figure 1 reports both X's projections, Y (panel A) and Z (panel B).

```
fig = plt.figure()
# bottom nodes network
cc = max(nx.connected_components(g_b), key=len)
# filter in the nodes in the largest connected component
g_b = g_b.subgraph(cc)
ax0 = fig.add_subplot(121)
pos = nx.spring_layout(g_b)
nx.draw(g_b, with_labels=False, node_color='plum', node_size=30, ax=ax0)
ax0.text(-0.7, 1, 'A', fontsize=12, ha='center')
# top nodes network
ax1 = fig.add_subplot(122)
nx.draw(g_t, with_labels=False, node_color='lightgreen', node_size=30, ax=ax1)
ax1.text(1, 1, 'B', fontsize=12, ha='center')
# show plot
plt.show()
```

Figure 1: Unweighted projections of the two-mode networks



4.2 Weighted projections of the two-mode networks

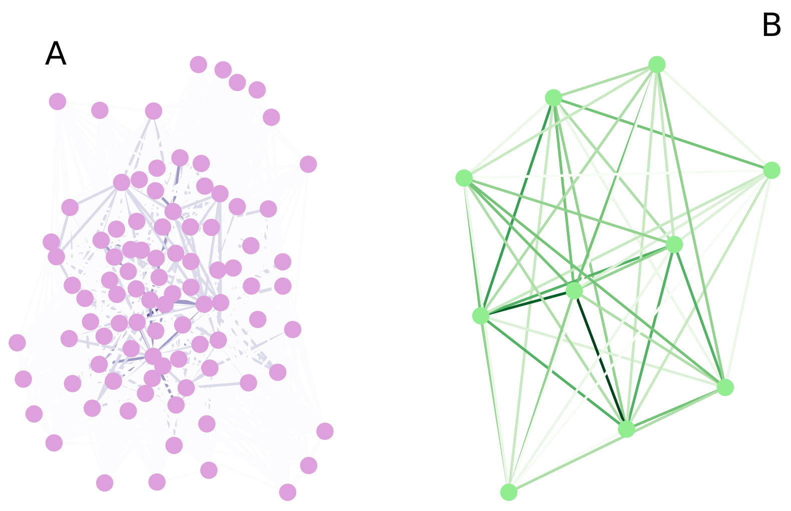
The weighted projected graph is the projection of the bipartite network `bg` onto the specified nodes with weights representing the number of shared neighbors or the ratio between actual shared neighbors and possible shared neighbors if ratio is True. The nodes retain their attributes and are connected in the resulting graph if they have an edge to a common node in the original graph.

```
g_b_w = bp.weighted_projected_graph(bg, bottom_nodes, ratio=True)
g_t_w = bp.weighted_projected_graph(bg, top_nodes, ratio=True)
```

Figure 2 reports both X's projections, Y (panel A) and Z (panel B).

```python
fig = plt.figure()
# bottom nodes network
cc = max(nx.connected_components(g_b_w), key=len)
# filter in the nodes in the largest connected component
g_b_w = g_b_w.subgraph(cc)
ax0 = fig.add_subplot(121)
pos = nx.spring_layout(g_b_w)
## draw the graph color coding the edges by weight
edges = g_b_w.edges(data=True)
weights = [w["weight"] for u, v, w in edges]
nx.draw(
    g_b_w,
    with_labels=False,
    node_color="plum",
    node_size=30,
    edge_color=weights,
    edge_cmap=plt.cm.Purples,
    ax=ax0,
)
ax0.text(-0.7, 1, "A", fontsize=12, ha="center")
# top nodes network
ax1 = fig.add_subplot(122)
## draw the graph color coding the edges by weight
edges = g_t_w.edges(data=True)
weights = [w["weight"] for u, v, w in edges]
nx.draw(
    g_t_w,
    with_labels=False,
    node_color="lightgreen",
    node_size=30,
    edge_color=weights,
    edge_cmap=plt.cm.Greens,
    ax=ax1,
)
ax1.text(1, 1, "B", fontsize=12, ha="center")
# show plot
plt.show()
```

Figure 2: Weighted projections of the two-mode networks



5 Writing projections to files

Finally, we write the projections to files.

```
path = 'data'
f0 = 'event_event_graph.csv'
nx.write_edgelist(g_t, open(os.path.join(".", f0), 'wb'))
f1 = 'user_user_graph.csv'
nx.write_edgelist(g_t_w, open(os.path.join(".", f1), 'wb'))
f2 = 'event_event_weighted_graph.csv'
nx.write_weighted_edgelist(g_b, open(os.path.join(".", f2), 'wb'))
f3 = 'user_user_weighted_graph.csv'
nx.write_weighted_edgelist(g_b_w, open(os.path.join(".", f3), 'wb'))
f4 =  'bipartite_graph.csv'
nx.write_edgelist(bg, open(os.path.join(".", f4), 'wb'))
```