BrainHubs

December 5, 2024

[2]:
```
# To find the Markov matrix

import numpy as np
import pandas as pd

# Load the neurons and connections data
neurons_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/
 ↪traced-neurons.csv')
conn_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/
 ↪traced-total-connections.csv')

# Create a dictionary mapping body IDs to neuron indices
bodyid_to_idx = {bodyid: idx for idx, bodyid in enumerate(neurons_df['bodyId'])}

# Compute the outdegree of each neuron
outdegree_dict = {bodyid: 0 for bodyid in bodyid_to_idx.keys()}
for from_bodyid in conn_df['bodyId_pre']:
    outdegree_dict[from_bodyid] += 1

# Create the markov matrix
n = len(bodyid_to_idx)

markov = np.zeros((n, n), dtype=np.float64)
for from_bodyid, to_bodyid in zip(conn_df['bodyId_pre'],␣
 ↪conn_df['bodyId_post']):
    from_idx, to_idx = bodyid_to_idx[from_bodyid], bodyid_to_idx[to_bodyid]
    markov[from_idx, to_idx] = 1/outdegree_dict[from_bodyid]


# Set the row and column labels
markov_df = pd.DataFrame(markov, columns=neurons_df['bodyId'],␣
 ↪index=neurons_df['bodyId'])

# Print the markov matrix
markov_df
```

[2]:
bodyId	200326126	202916528	203253072	203253253	203257652	\
bodyId						
200326126	0.0	0.000000	0.0	0.000000	0.000000	
202916528	0.0	0.000000	0.0	0.021739	0.021739	
203253072	0.0	0.000000	0.0	0.016393	0.000000	
203253253	0.0	0.000000	0.0	0.000000	0.000000	
203257652	0.0	0.029412	0.0	0.029412	0.000000	
…	…	…	…	…	…	
7112579856	0.0	0.000000	0.0	0.000000	0.000000	
7112615127	0.0	0.000000	0.0	0.000000	0.000000	
7112617294	0.0	0.000000	0.0	0.000000	0.000000	
7112622044	0.0	0.000000	0.0	0.001908	0.000000	
7112622236	0.0	0.000000	0.0	0.000000	0.000000	

bodyId	203594169	203594175	203598499	203598504	203598542	…	\
bodyId						…	
200326126	0.000000	0.000000	0.000000	0.000000	0.000000	…	
202916528	0.000000	0.000000	0.000000	0.000000	0.000000	…	
203253072	0.000000	0.016393	0.000000	0.016393	0.000000	…	
203253253	0.004651	0.000000	0.004651	0.000000	0.000000	…	
203257652	0.000000	0.029412	0.000000	0.000000	0.029412	…	
…	…	…	…	…	…	…	
7112579856	0.000000	0.000000	0.000000	0.000000	0.000000	…	
7112615127	0.000000	0.000000	0.000000	0.000000	0.000000	…	
7112617294	0.000000	0.000000	0.000000	0.000000	0.000000	…	
7112622044	0.000000	0.000000	0.000000	0.000000	0.000000	…	
7112622236	0.000000	0.000000	0.000000	0.000000	0.000000	…	

bodyId	5901231246	5901231325	5901232053	6400000773	7112579848	\
bodyId						
200326126	0.0	0.0	0.0	0.0	0.0	
202916528	0.0	0.0	0.0	0.0	0.0	
203253072	0.0	0.0	0.0	0.0	0.0	
203253253	0.0	0.0	0.0	0.0	0.0	
203257652	0.0	0.0	0.0	0.0	0.0	
…	…	…	…	…	…	
7112579856	0.0	0.0	0.0	0.0	0.0	
7112615127	0.0	0.0	0.0	0.0	0.0	
7112617294	0.0	0.0	0.0	0.0	0.0	
7112622044	0.0	0.0	0.0	0.0	0.0	
7112622236	0.0	0.0	0.0	0.0	0.0	

bodyId	7112579856	7112615127	7112617294	7112622044	7112622236
bodyId					
200326126	0.0	0.0	0.000000	0.0	0.0
202916528	0.0	0.0	0.000000	0.0	0.0
203253072	0.0	0.0	0.000000	0.0	0.0

```
203253253          0.0          0.0    0.000000          0.0          0.0
203257652          0.0          0.0    0.000000          0.0          0.0
…                  …            …      …                 …            …
7112579856         0.0          0.0    0.000000          0.0          0.0
7112615127         0.0          0.0    0.006623          0.0          0.0
7112617294         0.0          0.0    0.000000          0.0          0.0
7112622044         0.0          0.0    0.000000          0.0          0.0
7112622236         0.0          0.0    0.000000          0.0          0.0

[21739 rows x 21739 columns]
```

[ ]:
```python
# To find the Adjacency matrix

import numpy as np
import pandas as pd

import networkx as nx
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix

# Load the neurons and connections data
neurons_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/
 ↪traced-neurons.csv')
conn_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/
 ↪traced-total-connections.csv')

# Create a dictionary mapping body IDs to neuron indices
bodyid_to_id = {bodyid: idx for idx, bodyid in enumerate(neurons_df['bodyId'])}

# Create the adj matrix
n = len(bodyid_to_id)

adj = np.zeros((n, n), dtype=np.int32)
for from_bodyid, to_bodyid in zip(conn_df['bodyId_pre'],␣
 ↪conn_df['bodyId_post']):
    from_idx, to_idx = bodyid_to_id[from_bodyid], bodyid_to_id[to_bodyid]
    adj[from_idx, to_idx] = 1

# Set the row and column labels
adj_df = pd.DataFrame(adj, columns=neurons_df['bodyId'],␣
 ↪index=neurons_df['bodyId'])

# Print the adj matrix
adj_df
```

[3]:
```
#To find the stationary distribution

import numpy as np
import pandas as pd
from scipy.sparse.linalg import eigs

# Create the Markov matrix
markov = np.array(markov_df)
n = len(markov)

# Compute the eigenvectors and eigenvalues of the transpose of the Markov matrix
eigenvalues, eigenvectors = eigs(markov.T)

# Find the eigenvector corresponding to the eigenvalue closest to 1
idx = np.abs(eigenvalues - 1).argmin()
stationary = np.real(eigenvectors[:, idx].T / np.sum(eigenvectors[:, idx]))

# Convert the stationary distribution to a Pandas Series
stationary_df = pd.Series(stationary, index=markov_df.columns)

# Print the stationary distribution
print(stationary_df)
```

```
bodyId
200326126     0.000040
202916528     0.000069
203253072     0.000054
203253253     0.000204
203257652     0.000046
                ...
7112579856    0.000014
7112615127    0.000021
7112617294    0.000023
7112622044    0.000179
7112622236    0.000006
Length: 21739, dtype: float64
```

[5]:
```
# Find the top 2500 nodes with highest probabilities in the stationary␣
 ↪distribution
top_nodes = stationary_df.nlargest(2500)

# Print the top nodes
print("Nodes with highest probabilities in stationary distribution:")
print(top_nodes)
```

```
Nodes with highest probabilities in stationary distribution:
bodyId
```

```
329566174     0.000864
423101189     0.000832
425790257     0.000778
5813105172    0.000706
393766777     0.000698
                 …
5812982924    0.000099
800618162     0.000099
1068302710    0.000099
5813040190    0.000099
1072874511    0.000099
Length: 2500, dtype: float64
```

[ ]: