erdos_reyni

May 17, 2022

[1]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import cvxpy as cp
import itertools as it
import Metrics as met
import binary_graph_learning as bgl
```

[2]:
```
# Number of Nodes
N = 20
# Number of Signals
M = 100
# Seed for reproducibility
seed = 123

# Ground truth graph
G = nx.fast_gnp_random_graph(N, .3, seed = seed)
np.random.seed(seed)

# Set random wieghts for the edges
for s,t in G.edges:
    G[s][t]['weight'] = np.random.uniform(0,1)

L = nx.laplacian_matrix(G).toarray()
A = nx.adjacency_matrix(G).toarray()

# Normalized adjacency matrix
A = (N/sum(np.diag(L)))*A
```

[3]:
```
# Create ground truth h
np.random.seed(seed)
h = np.random.uniform(-1,1,N)
alpha = .2
```

[4]:
```
# Generate synthetic signals
Y = []
for i in range(M):
    signal = bgl.generate_signal(h,A)
```

```
    Y.append(signal)
    temp = {n:signal[n] for n in G.nodes()}
    nx.set_node_attributes(G, temp, name = "Round " + str(i+1))
Y = np.transpose(Y)
Y.shape
```

[4]: (20, 100)

[68]:
```
# Create mesh for parameter optimization and run the optamization
alpha = np.linspace(0,1,1)
estimates = []
for a in alpha:
    est = bgl.binary_graph_learning(Y, a)
    estimates.append((est,a))
```

```
CP variables built.
```

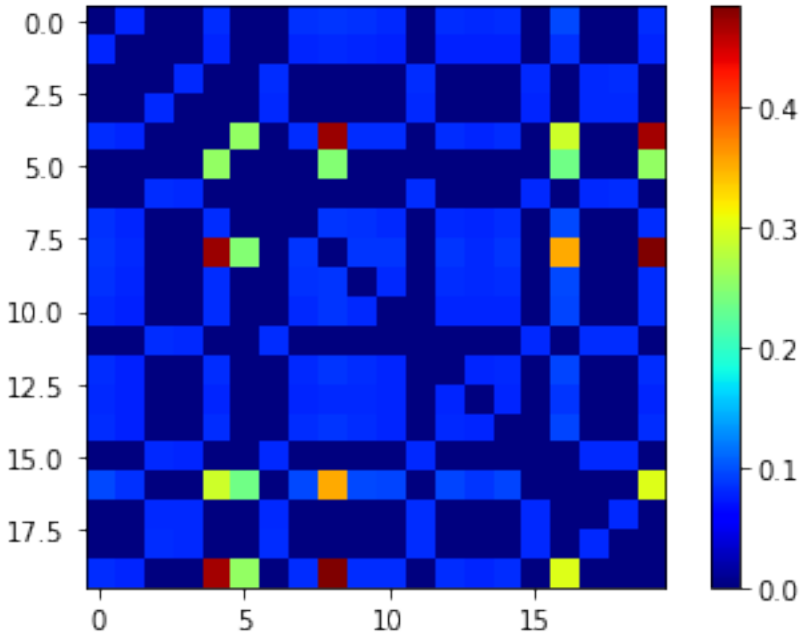
[76]:
```
mean = np.mean(estimates[0][0][9])
sd = np.std(estimates[0][0][9])
```

[77]:
```
print(mean, sd, np.median(estimates[0][0][9]))
```

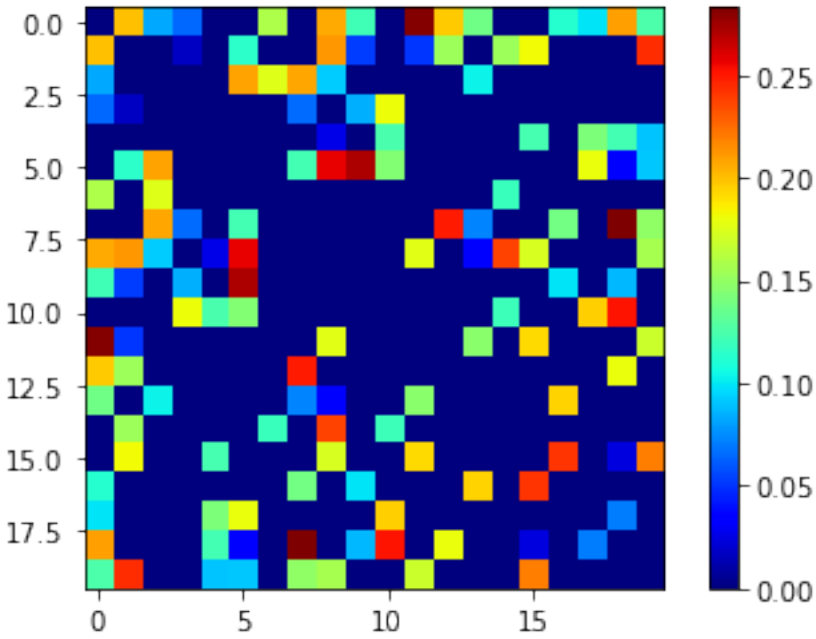
```
1.2943814498735278 1.8543558062609007 1.7684965253552725
```

[78]:
```
# Heat map of learned adjacency matrix
# Note this is a poor alpha value
est_A = estimates[0][0][9]
for i in range(len(est_A)):
    for j in range(len(est_A[i])):
        if est_A[i][j] < 1.8:
            est_A[i][j] = 0
print(np.mean(est_A))
est_A = ((N)/sum(est_A@np.ones(N)))*est_A
im1 = plt.imshow(est_A, cmap='jet', interpolation='nearest')
plt.colorbar()
plt.show()
```

```
1.1790152944705767
```



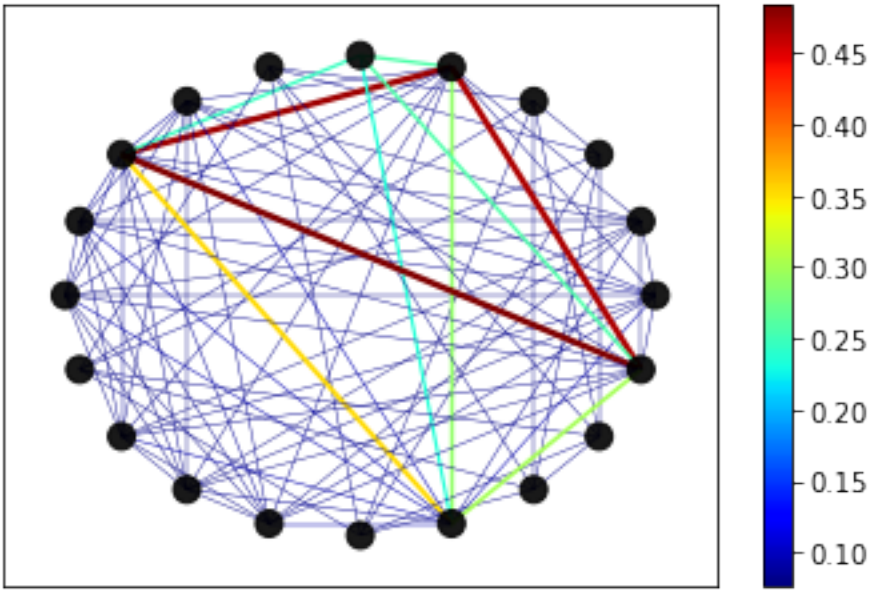
[9]:
```
# Heat map of ground truth
im1 = plt.imshow(A, cmap='jet', interpolation='nearest')
plt.colorbar()
plt.show()
```



[79]:
```
est_G = nx.from_numpy_matrix(est_A)
edges,weights = zip(*nx.get_edge_attributes(est_G,'weight').items())

nodes = est_G.nodes()
node_color = ['black']*len(signal)
size = [w*5 for w in weights]
pos = nx.circular_layout(est_G)
ec = nx.draw_networkx_edges(est_G, pos,
                            edgelist=edges,
                            edge_color=weights,
                            width=size,
                            edge_cmap=plt.cm.jet)
nc = nx.draw_networkx_nodes(est_G,pos,
                            nodelist=nodes,
                            node_color=node_color,
                            node_size = 100,
                            alpha = .9)
plt.colorbar(ec)
```

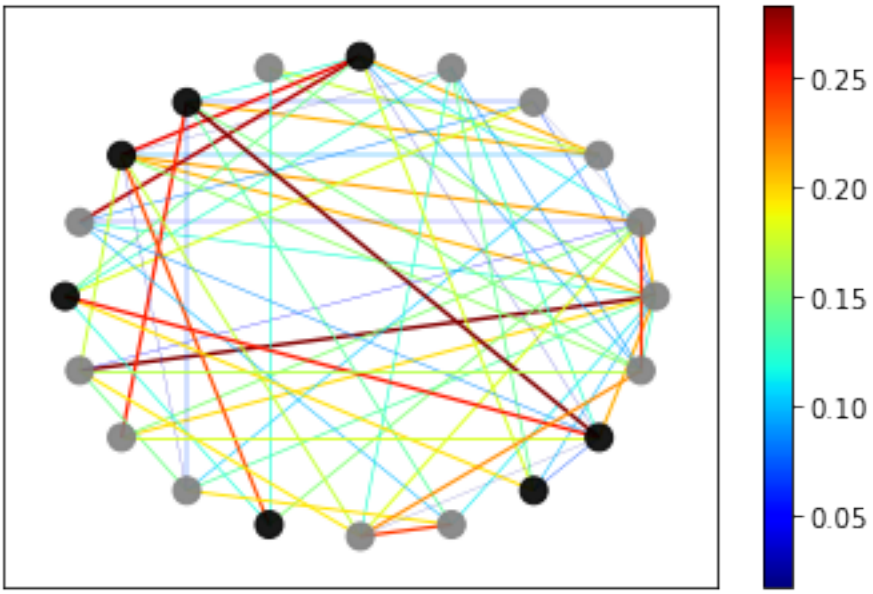
[79]:
```
<matplotlib.colorbar.Colorbar at 0x14ced4a60>
```



[75]:
```
G = nx.from_numpy_matrix(A)
edges,weights = zip(*nx.get_edge_attributes(G,'weight').items())
nodes = G.nodes()
node_color = ['grey']*len(signal)
for i in range(len(signal)):
    if signal[i] == 1:
```

```
        node_color[i] = 'black'
size = [w*5 for w in weights]
pos = nx.circular_layout(G)
ec = nx.draw_networkx_edges(G, pos,
                            edgelist=edges,
                            edge_color=weights,
                            width=size,
                            edge_cmap=plt.cm.jet)
nc = nx.draw_networkx_nodes(G,pos,
                            nodelist=nodes,
                            node_color=node_color,
                            node_size = 100,
                            alpha = .9)
plt.colorbar(ec)
```

[75]: <matplotlib.colorbar.Colorbar at 0x14d14e280>



[80]:
```
print(G.number_of_edges(), est_G.number_of_edges())
```

```
68 91
```