CMLC Simple Example

H. Robert Frost

This vignette illustrates the use of the CMLC R package to compute constrained multilayer centrality values for a simple 3 layer network where each layer is a undirected and unweighted 5 node network. The topology of the 3 layers is shown in Figure 1. Note that this is the same example network detailed in Section 4 of the associated paper (Eigenvector centrality for multilayer networks with dependent node importance. https://doi.org/10.48550/arXiv.2205.01478).

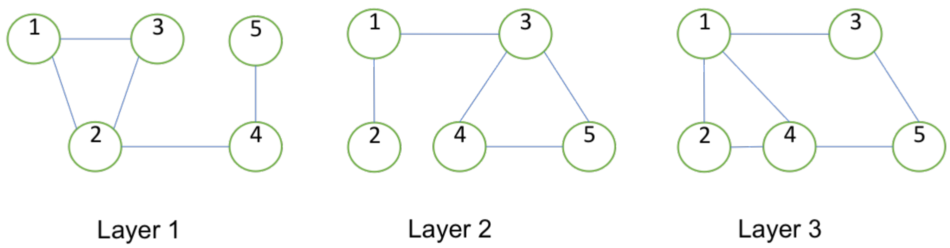

Figure 1: Simple undirected and unweighted 3 layer network.

1 Define adjacency matrices

For this example network, the symmetric adjacency matrices for the three layers can be specified in R as:

```
> layer1 = matrix(
+ c(0,1,1,0,0,
+   1,0,1,1,0,
+   1,1,0,0,0,
+   0,1,0,0,1,
+   0,0,0,1,0),
+ nrow=5, byrow=TRUE)
> layer2 = matrix(
+ c(0,1,1,0,0,
+   1,0,0,0,0,
+   1,0,0,1,1,
+   0,0,1,0,1,
+   0,0,1,1,0),
+ nrow=5, byrow=TRUE)
> layer3 = matrix(
+ c(0,1,1,1,0,
+   1,0,0,1,0,
+   1,0,0,0,1,
+   1,1,0,0,1,
+   0,0,1,1,0),
+ nrow=5, byrow=TRUE)
```

Aggregate all of the layer adjacency matrices into a list:

2 No dependency scenario

First, load the CMLC library:

```
> library(CMLC)
```

If no dependencies exist between the layers (i.e., $\tilde{\mathbf{A}} = \mathbf{I}$), the eigenvector centrality values computed by the constrained method are equivalent to those computed separately for each layer using eigen() (note that there will likely be small deferences depending on the threshold used for power iteration with constrainedMultiplePowerIteration()):

```
> cmlc.out = constrainedMultiplePowerIteration(X, A=diag(c(1,1,1)))
> cmlc.out$num.iter

[1] 13

> cmlc.out$V1[,1]

[1] 0.4969149 0.6043330 0.4969149 0.3419073 0.1550232

> abs(eigen(layer1)$vectors[,1])

[1] 0.4971537 0.6037035 0.4971537 0.3424853 0.1546684

> cmlc.out$V1[,2]

[1] 0.3419073 0.1550232 0.6043330 0.4969149 0.4969149

> abs(eigen(layer2)$vectors[,1])

[1] 0.3424853 0.1546684 0.6037035 0.4971537 0.4971537

> cmlc.out$V1[,3]

[1] 0.5299022 0.4271271 0.3577498 0.5299022 0.3577498

> abs(eigen(layer3)$vectors[,1])

[1] 0.5298991 0.4271323 0.3577512 0.5298991 0.3577512
```

As expected given the structure of layer 1, node 2 has the largest eigenvector centrality, followed by nodes 1 and 3 with node 5 having the lowest. Similarly for layer 2, node 3 has the largest centrality, followed by nodes 4 and 5 with node 2 having the lowest centrality. For layer 3, nodes 1 and 4 are tied for the largest centrality with nodes 3 and 5 tied for the lowest.

3 Mixture of layer dependency cases A and B

If layer 1 is independent, layer 2 is dependent on just layer 1 and layer 3 is dependent on layer 2, the $\tilde{\mathbf{A}}$ matrix takes the form:

```
> A = matrix(
+ c(1,0,0,
+    1,0,0,
+    0,1,0),
+ nrow=3, byrow=TRUE)
```

The constrained eigenvector centrality values for this scenario are:

```
> cmlc.out = constrainedMultiplePowerIteration(X=X, A=A)
> cmlc.out$num.iter

[1] 32

> cmlc.out$V1

          [,1]      [,2]      [,3]
[1,] 0.4971549 0.5820177 0.4782621
[2,] 0.6037004 0.2628438 0.3910999
[3,] 0.4971549 0.5256875 0.4330112
[4,] 0.3424882 0.3446154 0.5439485
[5,] 0.1546667 0.4439159 0.3673249
```

Since layer 1 is still independent in this scenario, it has the same centrality values as the prior case. For layer 2, we see the expected increase in the centrality of node 1 relative to node 3 given the importance of their adjacent nodes in layer 1 (i.e, node 1 is adjacent to node 2, which has the largest centrality value in layer 1; node 3 is adjacent to nodes 4 and 5, which have the lowest centrality values in layer 1). For layer 3, the centrality for node 3 has the largest change (an increase) relative to the independent scenario, which is expected given that it is adjacent to the node with the largest centrality value in layer 2 (node 1).

4 Mixture of layer dependency cases A and B with dependencies modeled by inter-layer edges

Most existing approaches for multilayer eigenvector centrality represent dependencies between layers using inter-layer edges. For the dependency scenario outlined above, this is equivalent to all of the nodes in layer 2 have directed edges of weight 1 to the same nodes in layer 1. After introduction of these edges, the entire multilayer network can be represented by a single network with pk nodes and the following $pk \times pk$ adjacency matrix:

```
> (merged.adjacency = createMergedAdjacencyMatrix(X=X, A=A))

15 x 15 sparse Matrix of class "dgCMatrix"

 [1,] . 1 1 . . . . . . . . . . . .
 [2,] 1 . 1 1 . . . . . . . . . . .
 [3,] 1 1 . . . . . . . . . . . . .
 [4,] . 1 . . 1 . . . . . . . . . .
 [5,] . . . 1 . . . . . . . . . . .
```

```
 [6,] 1 . . . . . 1 1 . . . . . . .
 [7,] . 1 . . . 1 . . . . . . . . .
 [8,] . . 1 . . 1 . . 1 1 . . . . .
 [9,] . . . 1 . . . 1 . 1 . . . . .
[10,] . . . . 1 . . 1 1 . . . . . .
[11,] . . . . . 1 . . . . . 1 1 1 .
[12,] . . . . . . 1 . . . 1 . . 1 .
[13,] . . . . . . . 1 . . 1 . . . 1
[14,] . . . . . . . . 1 . 1 1 . . 1
[15,] . . . . . . . . . 1 . . 1 1 .
```

Multilayer eigenvector centrality values can then be computed usign the standard eigenvector centrality formulation on the merged network:

```
> interlayerEdgeCentrality(X=X, A=A, normalize.per.layer=TRUE)
```

```
          [,1]      [,2]      [,3]
[1,] 0.5410003 0.3889868 0.5298991
[2,] 0.5032057 0.2366012 0.4271323
[3,] 0.4750426 0.6028442 0.3577512
[4,] 0.4578841 0.4678182 0.5298991
[5,] 0.1370376 0.4587311 0.3577512
```

This type of approach obviously has a very distinct mathematical interpretation from an approach which uses adjacent node importance to capture inter-layer dependencies and, as expected, the constrained centrality values are very different from those generated according to the adjacent node importance technique.

5 Mixture of layer dependency cases A, B and C

If layer 1 is independent, layer 2 is dependent on just layer 1 and layer 3 is equally dependent on both layer 2 and itself, the $\tilde{\mathbf{A}}$ matrix takes the form:

```
> A = matrix(
+ c(1,0,0,
+    1,0,0,
+    0,0.5,0.5),
+ nrow=3, byrow=TRUE)
```

The constrained eigenvector centrality values for this scenario are:

```
> cmlc.out = constrainedMultiplePowerIteration(X, A=A)
> cmlc.out$num.iter
```

```
[1] 32
```

```
> cmlc.out$V1
```

```
          [,1]      [,2]      [,3]
[1,] 0.4971549 0.5820177 0.5091828
[2,] 0.6037004 0.2628438 0.4064337
[3,] 0.4971549 0.5256875 0.3936928
[4,] 0.3424882 0.3446154 0.5319309
[5,] 0.1546667 0.4439159 0.3709447
```

Since layers 1 and 2 have the same dependency structure as the prior scenario, the centrality values are unchanged. As expected, the equally divided dependency structure for layer 3 yields centrality values that are between those computed in the first two scenarios.

6 Mixture of layer dependency cases A, B and C with negative dependency

If layer 1 is independent, layer 2 is dependent on just layer 1 and layer 3 has a positive dependence on itself and a small negative dependency on layer 2, the $\tilde{\mathbf{A}}$ matrix takes the form:

```
> A = matrix(
+ c(1,0,0,
+    1,0,0,
+    0,-0.2,1.2),
+ nrow=3, byrow=TRUE)
```

The constrained eigenvector centrality values for this scenario are (note that users may want to override the default maximum number of iterations to ensure convergence when using negative weights):

```
> cmlc.out = constrainedMultiplePowerIteration(X, A=A)
> cmlc.out$num.iter

[1] 96

> cmlc.out$V1

          [,1]      [,2]      [,3]
[1,] 0.4971537 0.5820194 0.5369981
[2,] 0.6037035 0.2628434 0.4373771
[3,] 0.4971537 0.5256869 0.3422854
[4,] 0.3424853 0.3446161 0.5307609
[5,] 0.1546684 0.4439142 0.3485225
```

Since layers 1 and 2 have the same dependency structure as the prior scenario, the centrality values are unchanged. While the results for layer 3 are not dramatically different relative to the prior example and the impact of negative dependencies is not necessarily intuitive, the increase in the centrality of node 1 in layer 3 is consistent with the fact that it now has a negative association with the smallest centrality node in layer 2.

7 All layers are dependency case B with a cycle

If layer 1 is dependent on layer 3, layer 2 dependent on layer 1 and layer 3 dependent on layer 2, a cycle is introduced in the layer dependency graph, the $\tilde{\mathbf{A}}$ matrix takes the form:

```
> A = matrix(
+ c(0,0,1,
+    1,0,0,
+    0,1,0),
+ nrow=3, byrow=TRUE)
```

The constrained eigenvector centrality values for this scenario are:

```
> cmlc.out = constrainedMultiplePowerIteration(X, A=A)
> cmlc.out$num.iter

[1] 8

> cmlc.out$V1

          [,1]      [,2]      [,3]
[1,] 0.3975710 0.5888889 0.4766597
[2,] 0.6811267 0.2129040 0.4036389
[3,] 0.4185482 0.5471731 0.4325420
[4,] 0.3753689 0.3573921 0.5233568
[5,] 0.2488360 0.4251520 0.3858449
```

If the inter-layer dependencies are instead represented by inter-layer edges, the centrality values for this scenario are

```
> merged.adjacency = createMergedAdjacencyMatrix(X=X, A=A)
> interlayerEdgeCentrality(X=X, A=A, normalize.per.layer=TRUE)

          [,1]      [,2]      [,3]
[1,] 0.4861158 0.4506616 0.5160089
[2,] 0.5547537 0.3363244 0.3972156
[3,] 0.4599116 0.5650744 0.4141969
[4,] 0.4214381 0.4485059 0.5081169
[5,] 0.2584783 0.4041329 0.3823778
```