CSCI 5922 - NEURAL NETWORKS AND DEEP LEARNING

GRAPHS AND GRAPH CONVOLUTIONAL NETWORKS

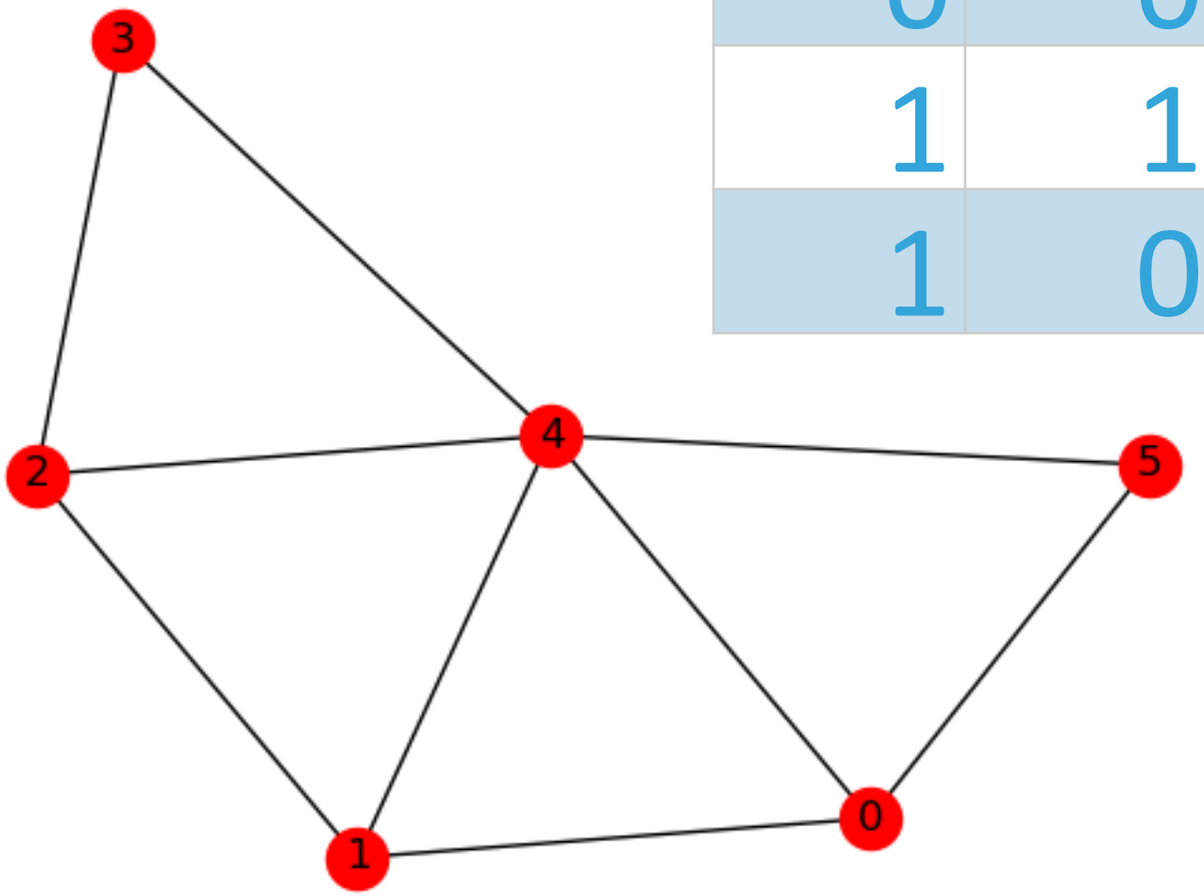
TASKS WITH GRAPHS

- Ranking (usually of vertices, can be of edges)
  - Find x_i in R for each vertex i
  - Degree centrality, eigenvector centrality, Katz centrality, PageRank
  - *Very* loosely: unsupervised regression
- Graph Partitioning
  - Partition vertices into two disjoint sets
  - Example: Spectral partitioning (Fiedler method)
- Graph Clustering or Community Detection
  - Separate vertices into multiple disjoint sets
  - Example: Spectral modularity maximization
  - *Very* loosely: unsupervised classification

ADJACENCY MATRIX

$$a_{ij} = \begin{cases} 1 \ if \ (i,j) \in E \\ 0 \ otherwise \end{cases}$$

0	1	0	0	1	1
1	0	1	0	1	0
0	1	0	1	1	0
0	0	1	0	1	0
1	1	1	1	0	1
1	0	0	0	1	0

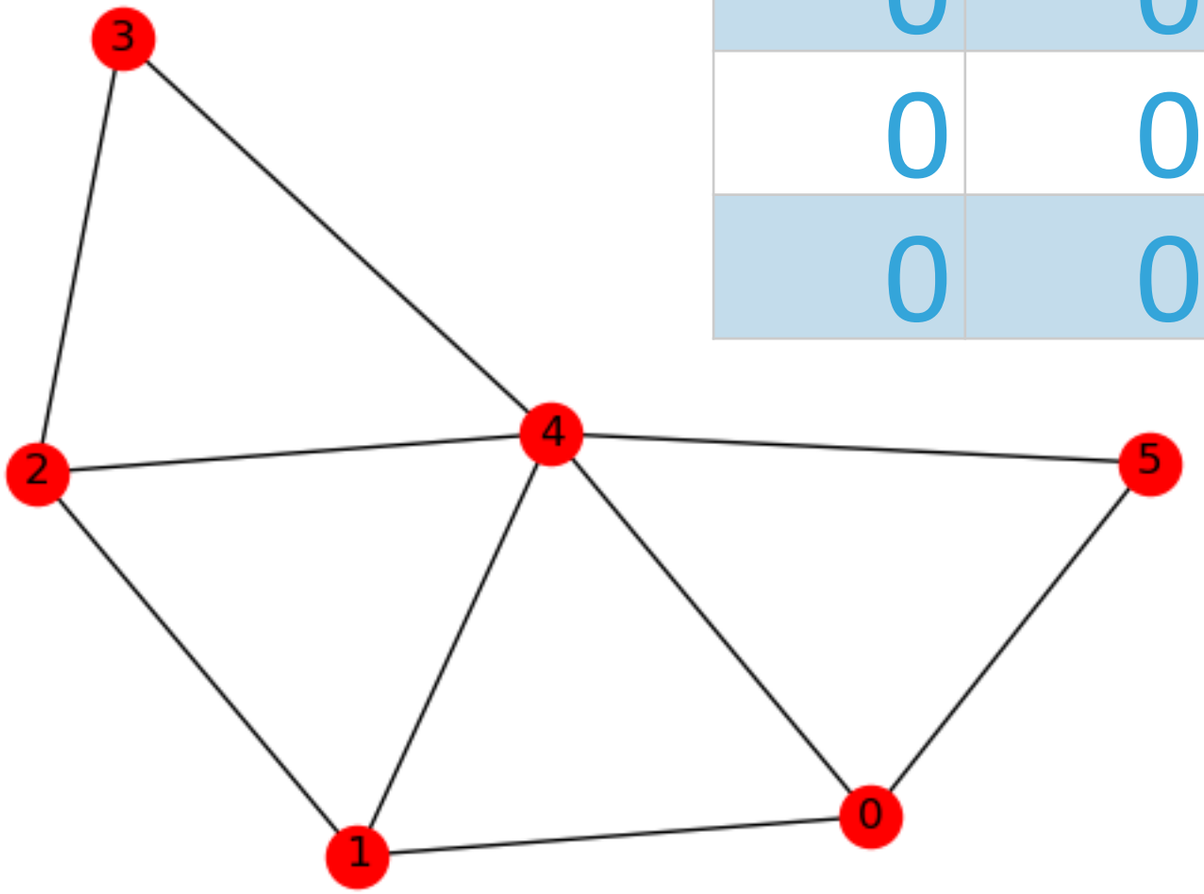


Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

DEGREE MATRIX

$$d_{ij} = \begin{cases} \sum_{k=0}^{N} a_{ik} \ if \ i = j \\ 0 \ otherwise \end{cases}$$

3	0	0	0	0	0
0	3	0	0	0	0
0	0	3	0	0	0
0	0	0	2	0	0
0	0	0	0	5	0
0	0	0	0	0	2

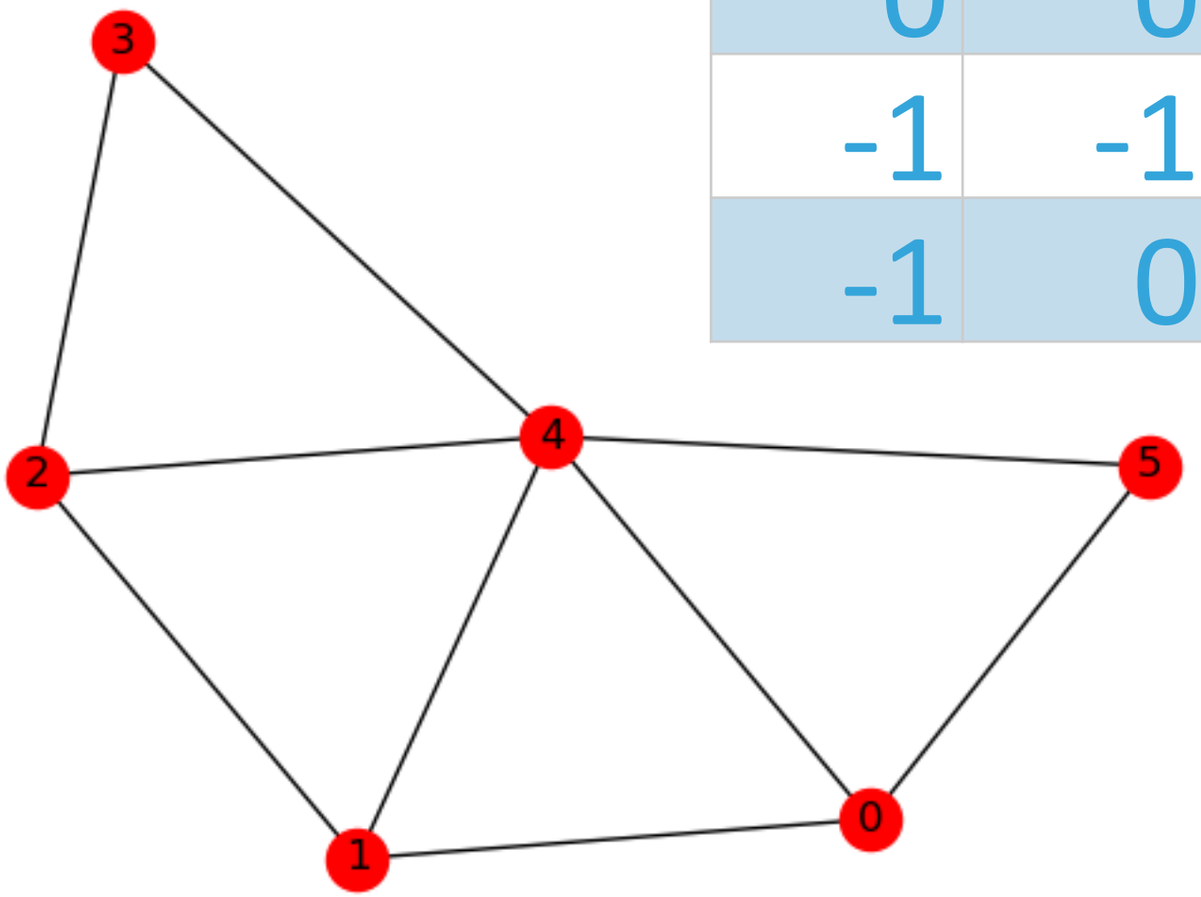


Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

LAPLACIAN MATRIX

$$L = D - A$$

3	-1	0	0	-1	-1
-1	3	-1	0	-1	0
0	-1	3	-1	-1	0
0	0	-1	2	-1	0
-1	-1	-1	-1	5	-1
-1	0	0	0	-1	2



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

SPECTRAL PARTITIONING

THE GOAL IS TO PARTITION A NETWORK WITH N NODES INTO TWO (CONNECTED) COMPONENTS WITH N_1 AND N_2 NODES WITH THE SMALLEST NUMBER OF EDGES BETWEEN THEM OF ANY PARTITION. WE WANT TO MINIMIZE THE CUT SIZE BETWEEN THE COMPONENTS.

NEWMAN SHOWS (BASED ON FIEDLER) THAT THE CUT SIZE IS A FUNCTION OF THE PRODUCT OF THE COMPONENT SIZES AND THE EIGENVALUE CORRESPONDING TO SOME EIGENVECTOR OF THE GRAPH LAPLACIAN.

$$R = \frac{n_1 n_2}{n} \lambda$$

WE WANT TO MINIMIZE THE THIS VALUE. SINCE N, N_1, AND N_2 ARE FIXED, WE MUST MINIMIZE LAMBDA. THE SMALLEST EIGENVALUE IS 0, AND THE CORRESPONDING EIGENVECTOR IS 1S, SO WE CHOOSE THE SECOND-SMALLEST.

Networks: An Introduction, M.E.J. Newman, Section 11.5, p. 368

SPECTRAL PARTITIONING

1. Calculate the eigenvector v_2 corresponding to the second-smallest eigenvector lambda_2 of the graph Laplacian.

2. Sort the elements of the eigenvector in order for largest to smallest.

3. Put the vertices corresponding to the n_1 largest elements in group 1, the rest in group 2, and calculate the cut size.

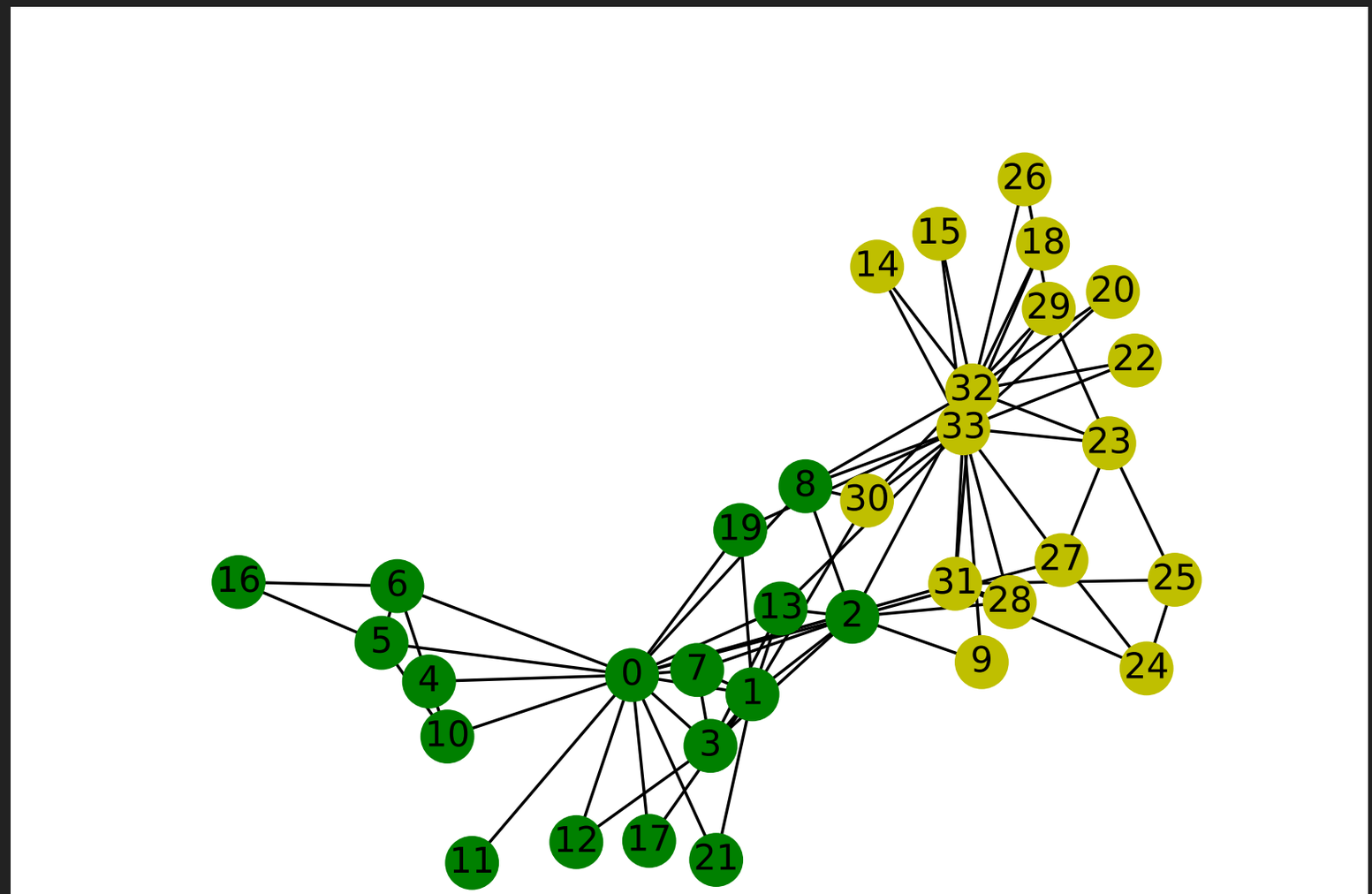
4. Then put the vertices corresponding to the n_1 smallest elements in group 1, the rest in group 2, and recalculate the cut size.

5. Between these two divisions of the network, choose the one that gives the smaller cut size.

Networks: An Introduction, M.E.J. Newman, Section 11.5, p. 369

SPECTRAL PARTITIONING

```
# coding: utf-8

import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

G = nx.karate_club_graph()
L = nx.laplacian_matrix(G).toarray()
w, v = np.linalg.eigh(L)
fiedler_value = w[1]
fiedler_vector = v[:, 1]

# Karate club has an even number of nodes,
# so n1 and n2 are the same.
n1 = len(G.nodes) // 2
n2 = len(G.nodes) - n1

indices = np.argsort(fiedler_vector)
smallest = indices[:n1]
colors = ['g' if i in smallest else 'y' for i in G]

nx.draw_networkx(
    G, pos=nx.spring_layout(G, seed=2), node_color=colors)
plt.title("Spectral partition of Zachary's karate club graph", y=-0.1)
plt.axis('off')
plt.show(block=False)
```



Spectral partition of Zachary's karate club graph

SPECTRAL MODULARITY MAXIMIZATION

THE QUANTITY Q IS THE MODULARITY OF A NETWORK. IT MEASURES THE ASSORTATIVITY OF THE NODES IN A NETWORK. IN AN ASSORTATIVE NETWORK, LIKE NODES ARE CONNECTED TO LIKE NODES. IN A DISASSORTATIVE NETWORK, NODES TEND TO CONNECT TO THOSE DISSIMILAR FROM THEM.

MODULARITY IS POSITIVE FOR ASSORTATIVE NETWORKS, NEGATIVE FOR DISASSORTATIVE NETWORKS.

Kronecker delta

Group or community of node i

$$Q = \frac{1}{2m} \sum_{ij} \underbrace{\left(A_{ij} - \frac{k_i k_j}{2m} \right)}_{B_{ij}} \delta(c_i, c_j) = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j)$$

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

NULL MODEL DEDUCTS FROM ANY EDGE THE PROBABILITY THAT IT WAS CREATED BY CHANCE.

SPECTRAL MODULARITY MAXIMIZATION

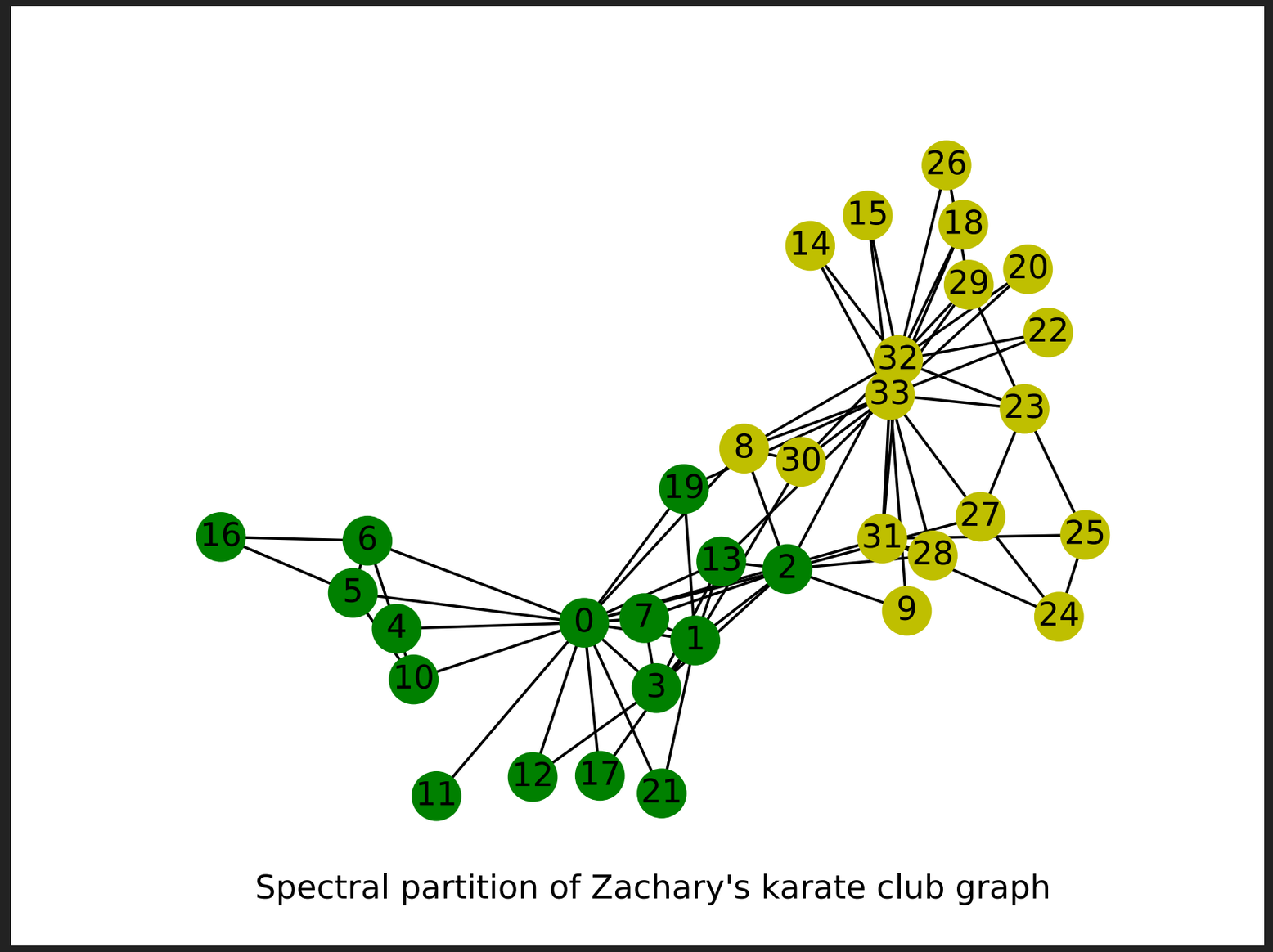
1. Calculate the eigenvector v_1 corresponding to the largest eigenvector lambda_1 of the modularity matrix **B**.

2. Put vertex i into group 1 if v_1_i is positive; otherwise put it into group 2.

3. Repeatedly apply steps 1 and 2 to the subgroups until the change in modularity \delta Q is not positive.

Networks: An Introduction, M.E.J. Newman, Section 11.5, p. 377-80

SPECTRAL MODULARITY MAXIMIZATION



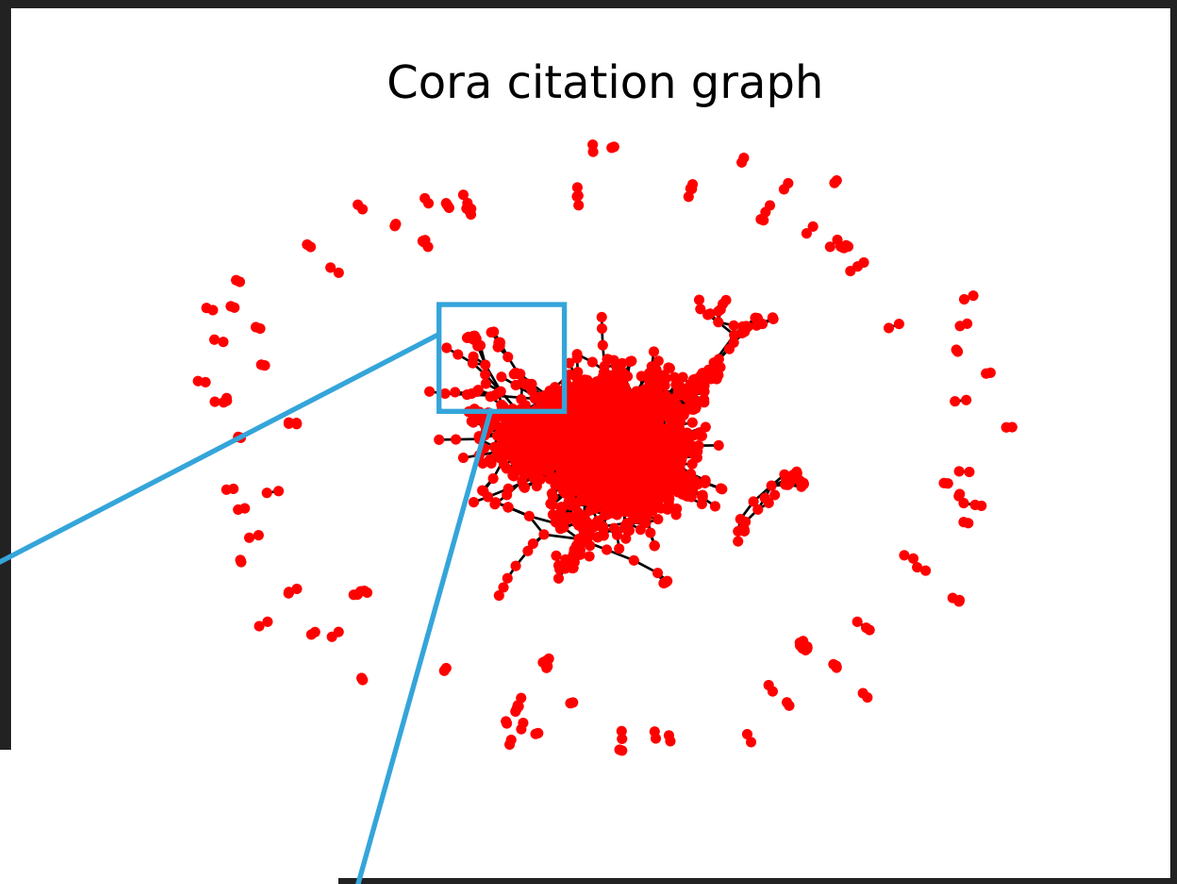
Spectral partition of Zachary's karate club graph

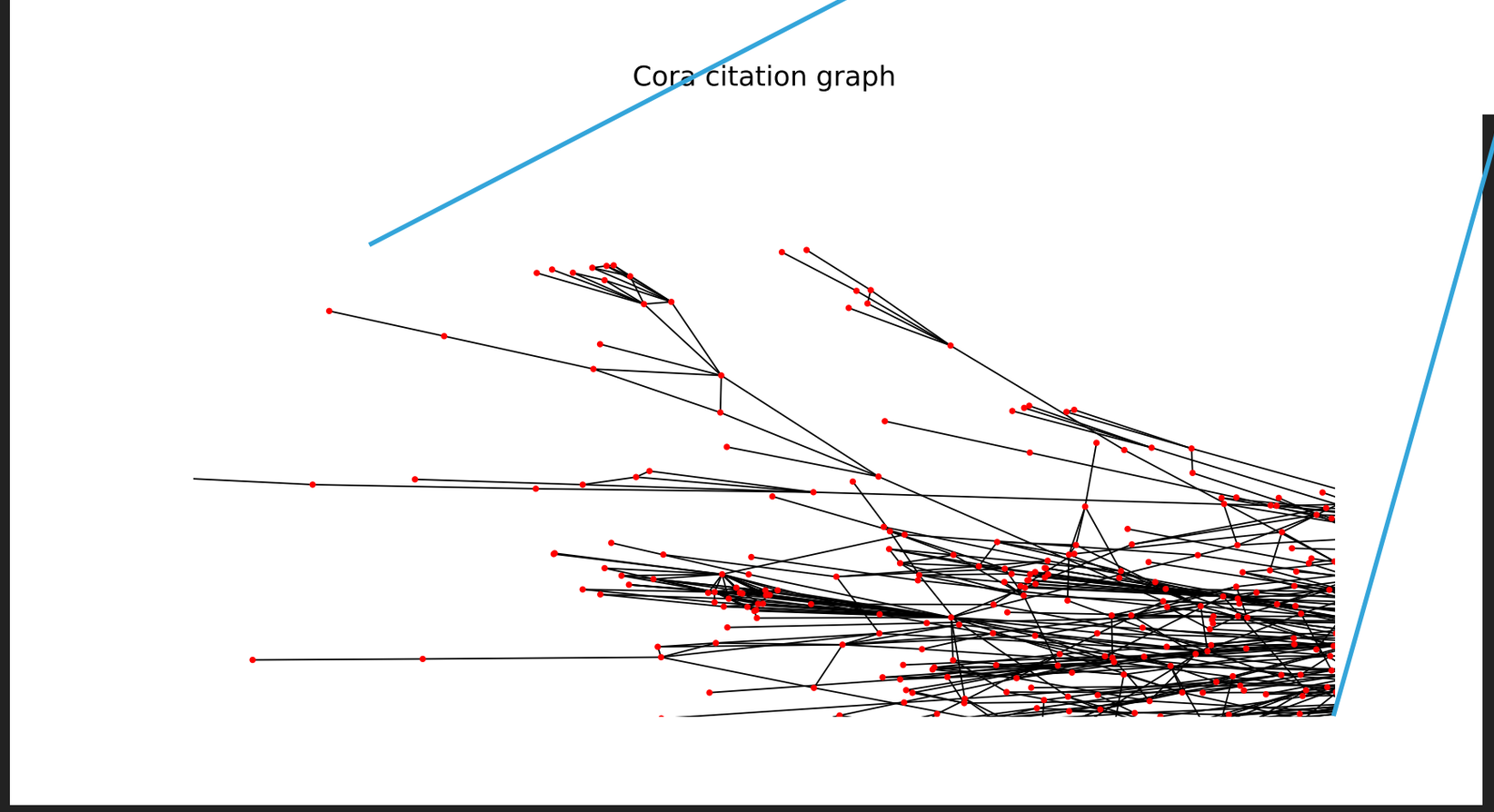
```python
# coding: utf-8

import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

G = nx.karate_club_graph()

B = nx.modularity_matrix(G)
w, v = np.linalg.eigh(B)
modularity_vector = v[:, -1]

groups = (modularity_vector > 0).astype(np.int)
colors = ['g' if groups[i] > 0 else 'y' for i in G]

nx.draw_networkx(
    G, pos=nx.spring_layout(G, seed=2), node_color=colors)
plt.title("Spectral modularity maximization of of Zachary's karate club graph", y=-0.1)
plt.axis('off')
plt.show(block=False)
```

CITATION NETWORKS

A CITATION NETWORK IS A DIRECTED ACYCLIC GRAPH (DAG) DERIVED FROM THE BIBLIOGRAPHIES OF SCIENTIFIC ARTICLES.

EACH NODE IN THIS GRAPH HAS TWO ATTRIBUTES: THE SUBTOPIC OF MACHINE LEARNING THAT THE PAPER IS ABOUT AND A FEATURE VECTOR DERIVED FROM THE ABSTRACT.

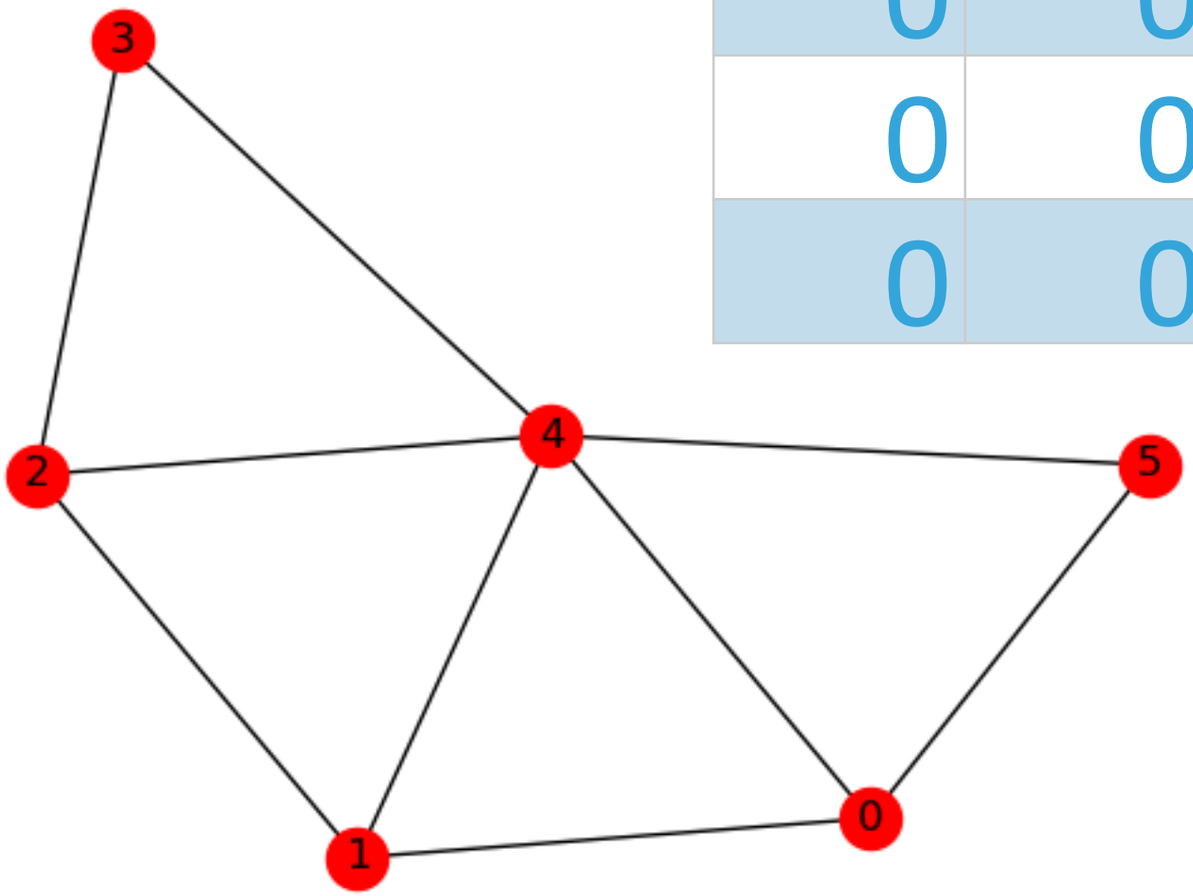




INVERSE DEGREE MATRIX

$$D^{-1}$$

0.33	0	0	0	0	0
0	0.33	0	0	0	0
0	0	0.33	0	0	0
0	0		0	0.50	0
0	0		0	0	0.20	0
0	0		0	0	0	0.50

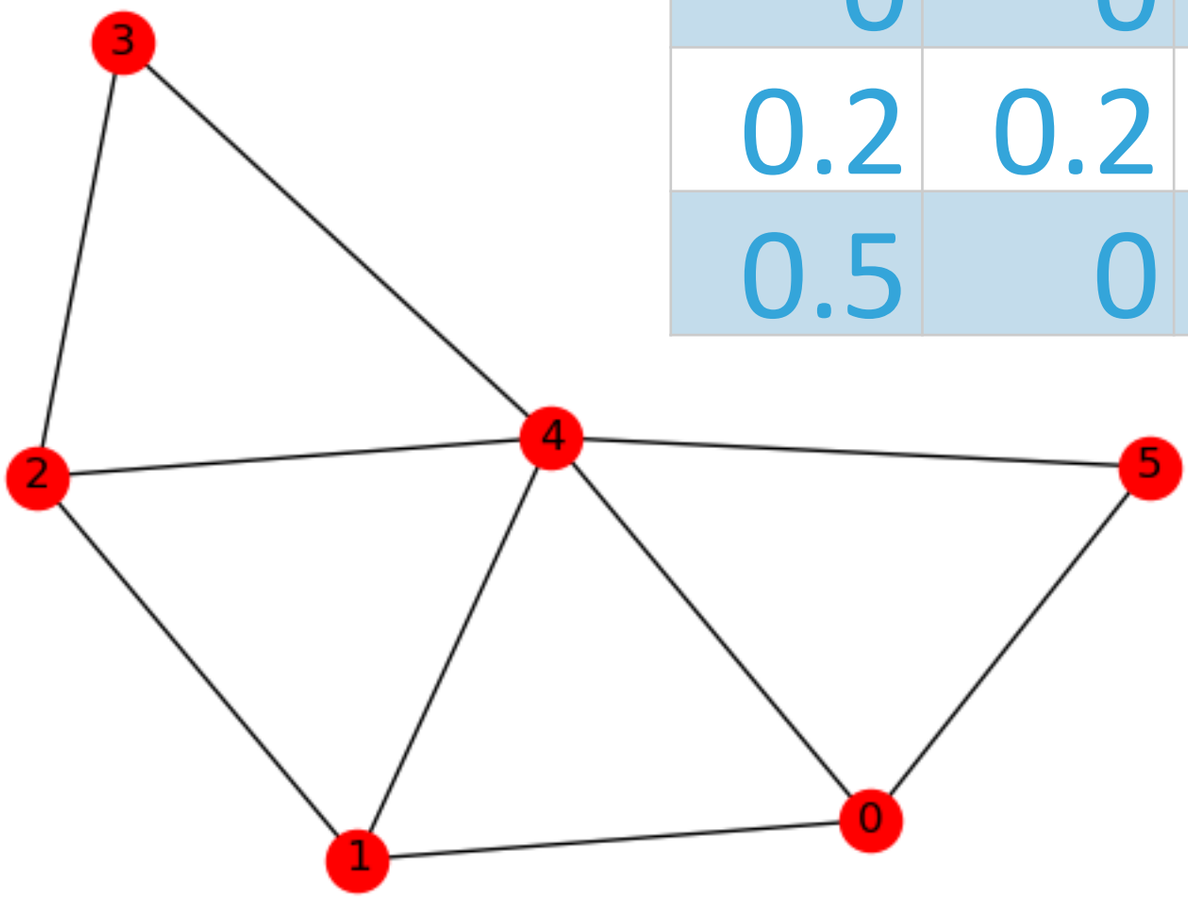


Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

NORMALIZED ADJACENCY MATRIX

$$D^{-1}A$$

0	0.33	0	0	0.33	0.33
0.33	0	0.33	0	0.33	0
0	0.33	0	0.33	0.33	0
0	0	0.5	0	0.5	0
0.2	0.2	0.2	0.2	0	0.2
0.5	0	0	0	0.5	0



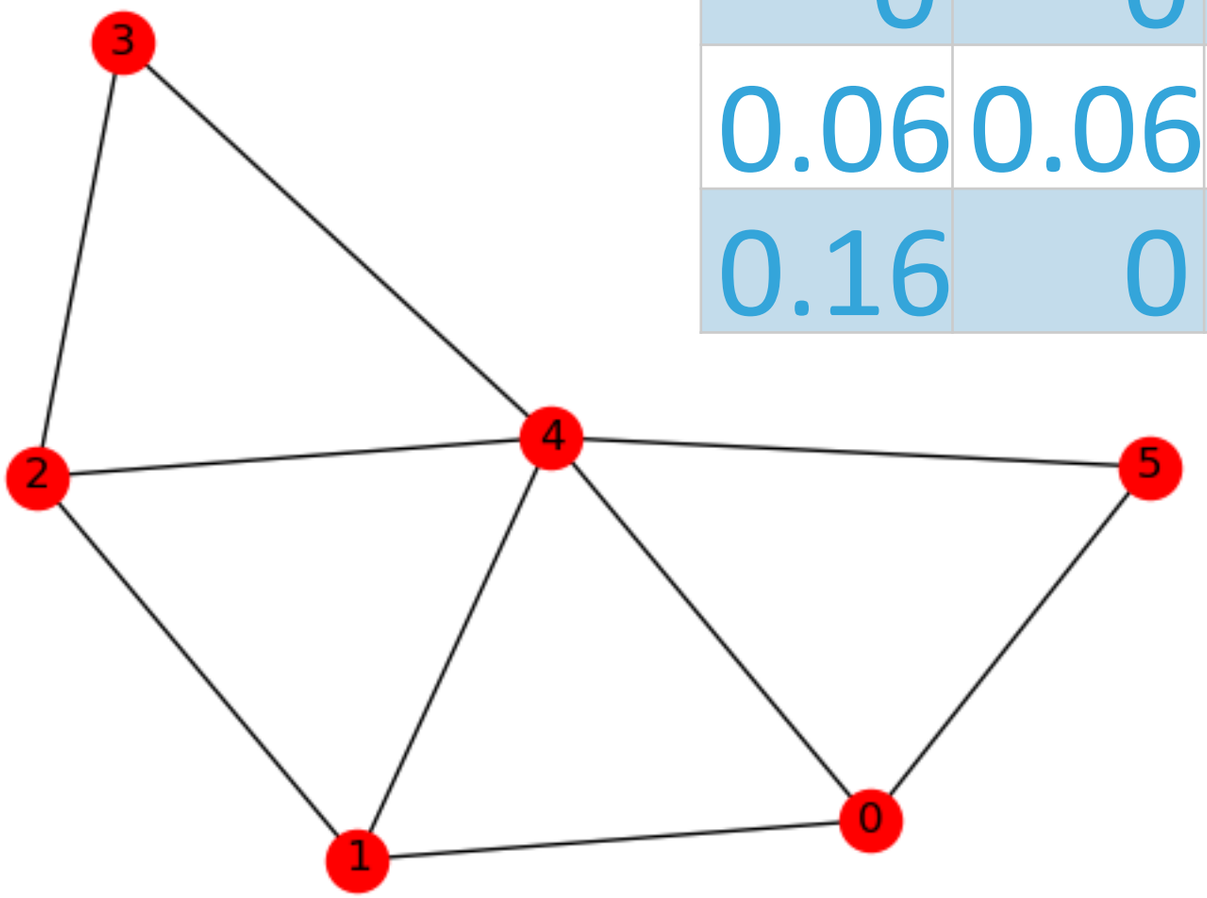
Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

LEFT NORMALIZATION ENSURES THAT ALL ROWS SUM TO ONE.

SYMMETRICALLY NORMALIZED ADJACENCY MATRIX

$$D^{-1}AD^{-1}$$

0	0.11	0	0	0.06	0.16
0.11	0	0.11	0	0.06	0
0	0.11	0	0.16	0.06	0
0	0	0.16	0	0.1	0
0.06	0.06	0.06	0.1	0	0.1
0.16	0	0	0	0.1	0



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

GRAPH CONVOLUTIONAL NETWORKS

Given a graph $G = (V, E)$, a graph convolutional network takes as input

$\mathbf{X} \in \mathbb{X}^{N \times D}$, a feature matrix with $N = |V|$ and $D =$ number of features

$\mathbf{A} \in \mathbb{A}^{N \times N}$, the adjacency matrix of G

The general propagation rule for a layer in a graph convolutional network is given by

$$\mathbf{H}^{(l+1)} = f(\mathbf{H}^{(l)}, \mathbf{A})$$

with L layers and $H^{(0)} = \mathbf{X}$

Semi-Supervised Classification with Graph Convolutional Networks, Kipf & Welling, 2016

GRAPH CONVOLUTIONAL NETWORKS

Consider now the following propagation rule with a non-linearity and a separate weight matrix per layer.

$$f(\mathbf{H}^{(l)}, \mathbf{A}) = \sigma\left(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

In this rule, the neighbors of a given node are summed, but not the node itself. Adding the identity matrix to the adjacency matrix includes the node itself in outputs related to it. Normalizing the adjacency matrix (similar to mean centering and scaling images) should aid in optimization. From which we obtain

$$f(\mathbf{H}^{(l)}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

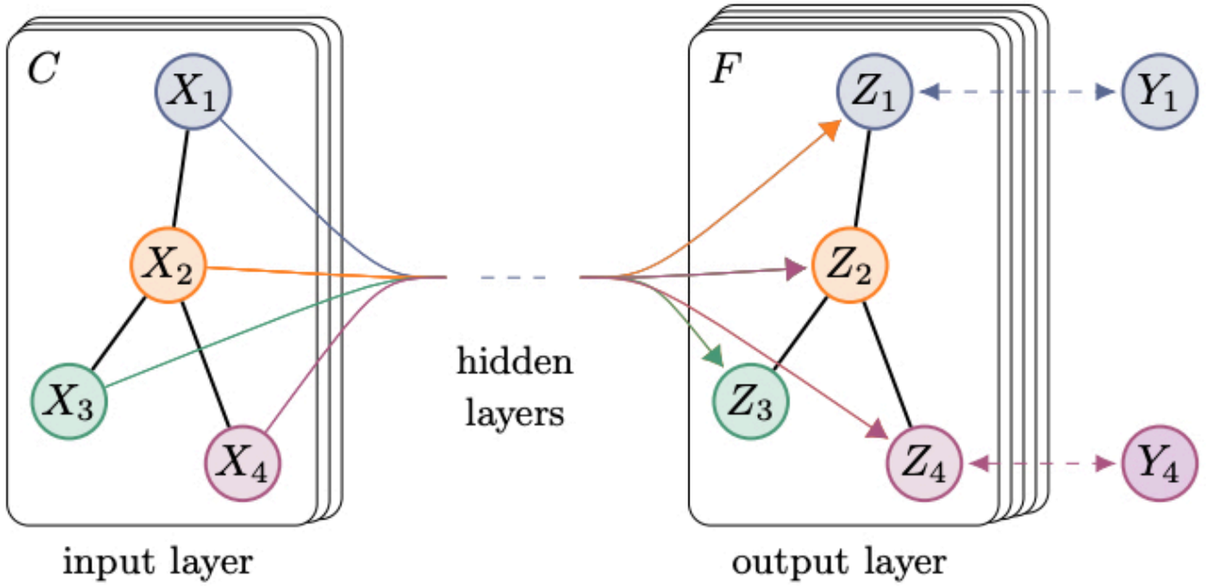
Semi-Supervised Classification with Graph Convolutional Networks, Kipf & Welling, 2016

TRANSDUCTIVE LEARNING

- Inductive learning
  - Supervised learning
    - Build a model from a training set
    - Use the model to predict target of unseen (and possibly unlabeled) examples
  - Semi-supervised learning
    - Build a model from a training set and additional unlabeled examples
    - Use the model to predict target of unseen (and possibly unlabeled) examples
- Transductive learning
  - Build a model from a training set and any (possibly unlabeled) examples you care about
  - Use the model only to obtain “predictions” of target of (possibly unlabeled) non-training examples
  - The model **cannot** be used in a predictive fashion on new, unseen examples

GRAPH CONVOLUTIONAL NETWORKS

https://github.com/tkipf/pygcn



(a) Graph Convolutional Network

(b) Hidden layer activations

Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

Semi-Supervised Classification with Graph Convolutional Networks, Kipf & Welling, 2016

GRAPH CONVOLUTIONAL NETWORKS

CODE WALKTHROUGH, TIME PERMITTING

https://github.com/tkipf/pygcn