Scan the QR to record your attendance

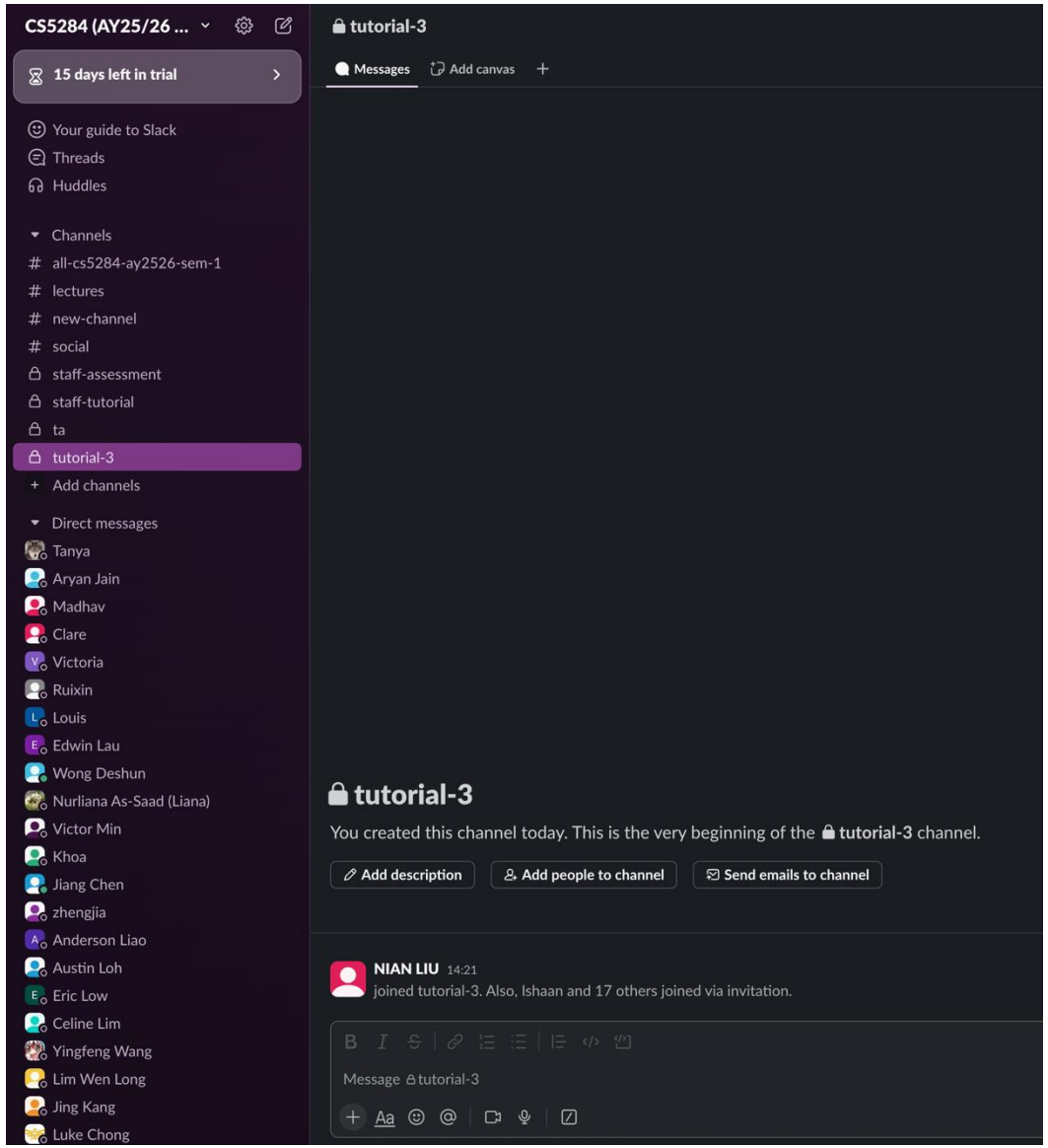
Something about tutorial

- Time: Wed 8:30pm-9:20pm … 50 mins
- Organization
  - Review the lecture last week
  - Tutorial Notebook, GitHub Code
    - Core API → Logic of code
  - Q&A
- Way of attendance
  - Come here, Zoom link, Self-study
  - Scan the QR code every week
- Contact
  - Slack, Email (nianliu@u.nus.edu)

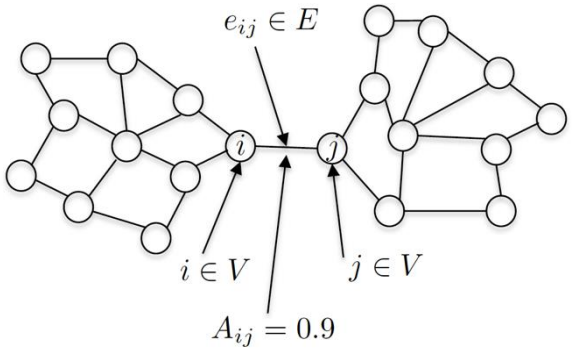
Slack Channel

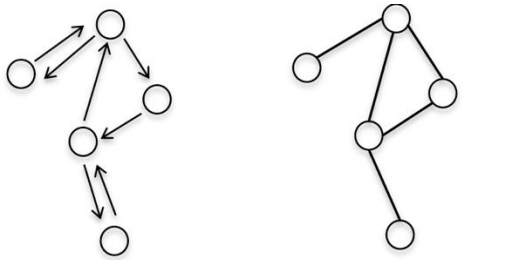
Join in this course

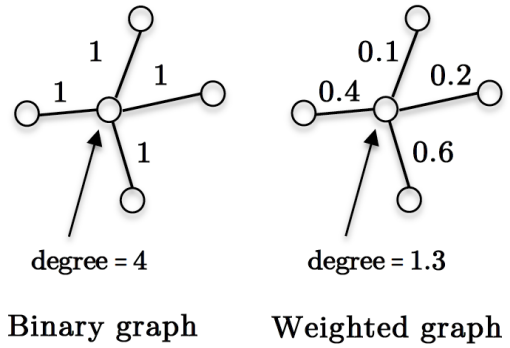
Invite you



Quick review -- Introduction to Graph Science





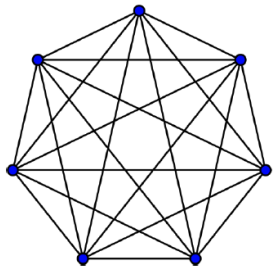


Node, Edge, Adjacency Matrix

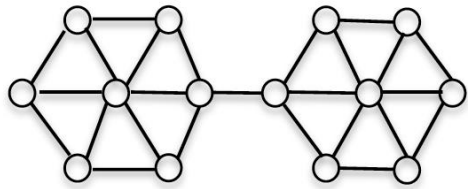
Direction

Weight, Degree

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \qquad A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0.4 & 0 & 0 \\ 0.4 & 0 & 0.7 & 0.3 \\ 0 & 0.7 & 0 & 1 \\ 0 & 0.3 & 1 & 0 \end{bmatrix} \end{array}$$



Dense



Sparse

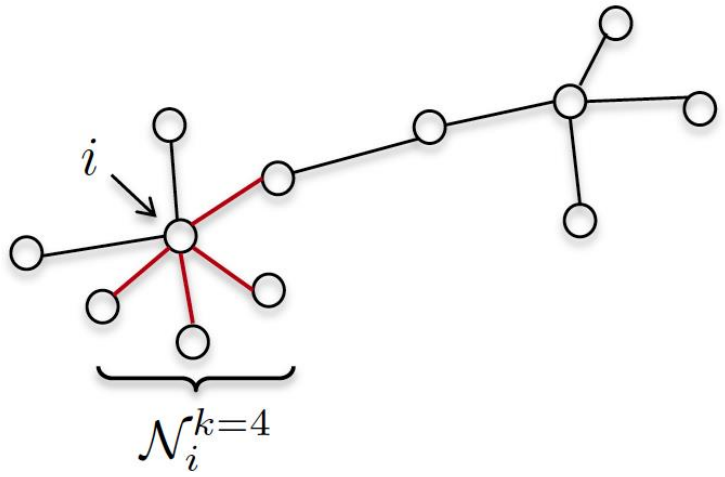
$$|E| = \frac{n(n-1)}{2} = O(n^2)$$

$$|E| = O(kn) = O(n)$$

Quick review -- Introduction to Graph Science

- Curse of dimensionality: High-dimension space, meaningless distance
- Extract low-dimension pattern: manifold learning
- Manifold is encoded in neighbor graph, KNN graph

$$A_{ij} = \begin{cases} e^{-\frac{\mathrm{dist}(x_i, x_j)^2}{\sigma^2}} & \text{if } j \in \mathcal{N}_i^k \\ 0 \text{ otherwise} \end{cases}$$



Core API

- Construct a sparse graph
  - W = scipy.sparse.csr_matrix((data, (row_ind, col_ind)), shape)

```
import numpy as np
from scipy.sparse import csr_matrix

row = np.array([0, 0, 1, 2, 2, 2])
col = np.array([0, 2, 2, 0, 1, 2])
data = np.array([1, 2, 3, 4, 5, 6])
W = csr_matrix((data, (row, col)), shape=(3, 3))
print(W)
```

✓ 0.2s

```
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 6 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (0, 2)        2
  (1, 2)        3
  (2, 0)        4
  (2, 1)        5
  (2, 2)        6
```

```
print(W.toarray())
```

✓ 0.0s

```
[[1 0 2]
 [0 0 3]
 [4 5 6]]
```

Core API

- Eigenvalue decomposition
  - lamb, U = scipy.sparse.linalg.eigsh(L, k=4, which='SM')

k : ***int, optional***

The number of eigenvalues and eigenvectors desired. *k* must be smaller than N. It is not possible to compute all eigenvectors of a matrix.

which : ***str [‘LM’ | ‘SM’ | ‘LA’ | ‘SA’ | ‘BE’]***

If A is a complex Hermitian matrix, ‘BE’ is invalid. Which *k* eigenvectors and eigenvalues to find:

‘LM’ : Largest (in magnitude) eigenvalues.
‘SM’ : Smallest (in magnitude) eigenvalues.
‘LA’ : Largest (algebraic) eigenvalues.
‘SA’ : Smallest (algebraic) eigenvalues.
‘BE’ : Half (k/2) from each end of the spectrum.

Core API

- Plot a graph -- matplotlib.pyplot.spy

```
import matplotlib.pyplot as plt

plt.spy(W, precision=0.01, markersize=1)
```

  - Plot the sparsity pattern of a 2D array.