CS6208 : Advanced Topics in Artificial Intelligence
Graph Machine Learning

Lecture 2 : Introduction to Graph Science

Semester 2 2022/23

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Course lectures

- Introduction to Graph Machine Learning
- Part 1: GML without feature learning (before 2014)
  - Introduction to Graph Science
  - Graph Analysis Techniques without Feature Learning
    - Graph clustering
    - Classification
    - Recommendation
    - Dimensionality reduction
- Part 2 : GML with shallow feature learning (2014-2016)
  - Shallow graph feature learning
- Part 3 : GML with deep feature learning, a.k.a. GNNs (after 2016)
  - Graph Convolutional Networks (spectral and spatial)
  - Weisfeiler-Lehman GNNs
  - Graph Transformer & Graph ViT/MLP-Mixer
  - Benchmarking GNNs
  - Molecular science and generative GNNs
  - GNNs for combinatorial optimization
  - GNNs for recommendation
  - GNNs for knowledge graphs
  - Integrating GNNs and LLMs

Outline

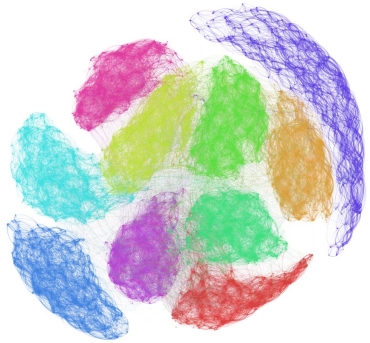
- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Outline

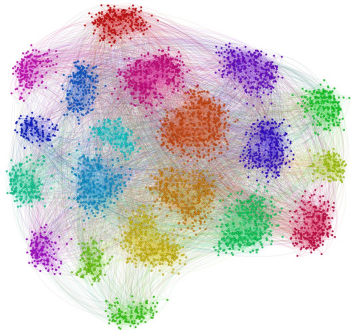
- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Graphs

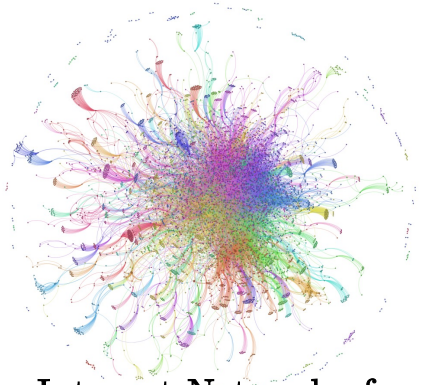
- Graphs encode complex data structures.
  - They are everywhere! Internet, social networks, customer-product relationships, etc



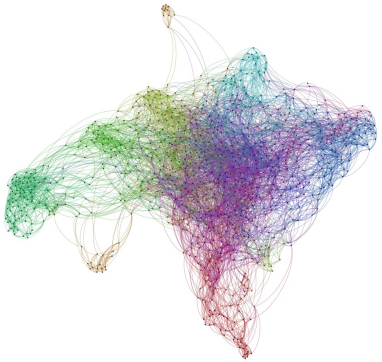
MNIST Image Network



Social Network

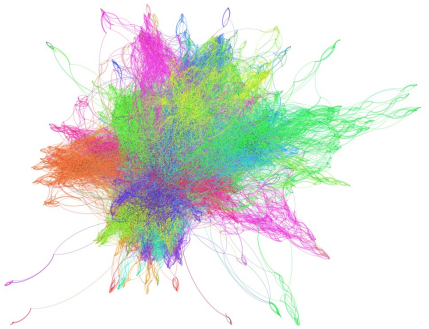


Internet Network of California



GTZAN Music Network

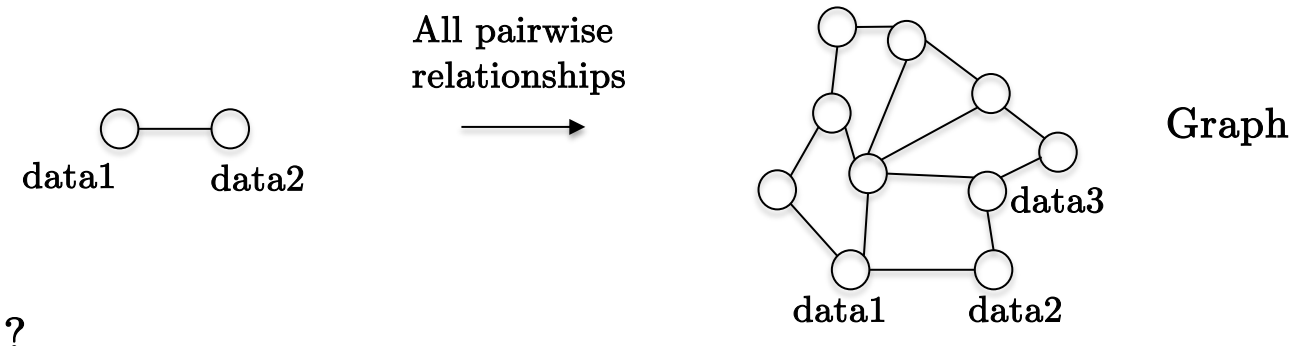
- "Graphs are the most important discrete models in the world." - Gil Strang (MIT)



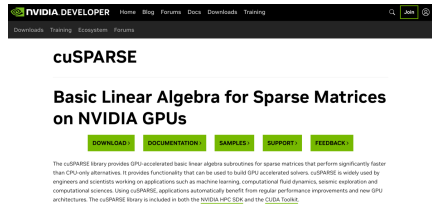
Network of Text Documents 20newsgroups

Graphs

- Definition : (Simple) mathematical model representing pairwise relationships between data.



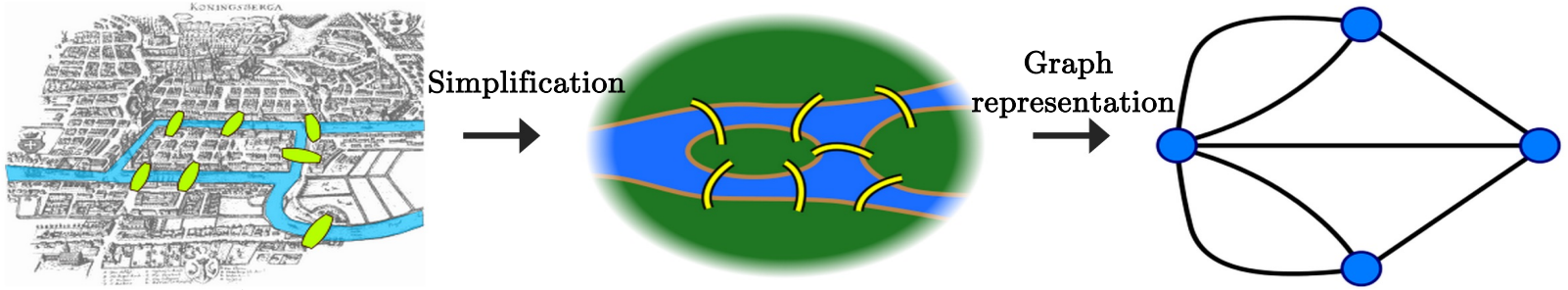
- Why are graphs useful?
  - Graphs offer a global view and analysis of data structures.
    - They possess meaningful patterns, i.e. insights about data properties.
  - Some tasks are exclusively designed for graphs, e.g. Google PageRank recommendation.
  - They can boost performance with additional priors, a.k.a. inductive bias.
  - They can benefit from GPUs as graphs are (sparse) matrices.



Graph theory

- When did it start?
  - History of graph theory : Graphs have been formally studied since 1736, starting with Mathematician Leonhard Euler and the famous problem of "Seven Bridges of Königsberg" :

    Q: Can we find a path through the city (starting from any place) that crosses each bridge once and only once? Euler proved that it is not possible (it is only feasible if the graph has even degree that allows cycles).



Königsberg
City

Source: Wikipedia.

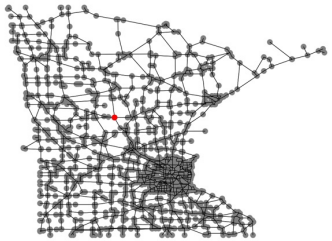
- Graph theory have since developed tools to analyze and process networks for all sort of applications : clustering, classification, visualization, recommendation, etc.

Outline

- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

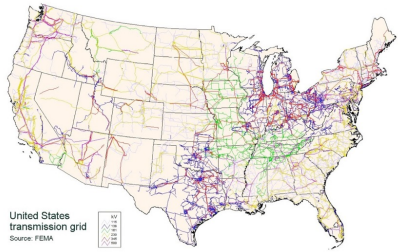
Graph categories

- Natural Graphs
  - Social networks : Meta, LinkedIn, Twitter
  - Biological networks : Brain connectivity & functionality, gene regulatory networks
  - Communication networks : Internet, networking devices
  - Transportation networks : Trains, cars, airplanes, pedestrians
  - Power networks : Electricity, water
- Natural graphs mean graphs that are not artificially hand-crafted/constructed.



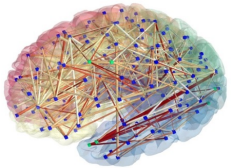
Minnesota Road Network

Telecommunication Network



US Electrical Network

Facebook



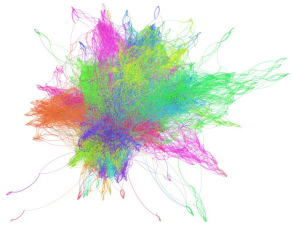
Brain Connectivity

=

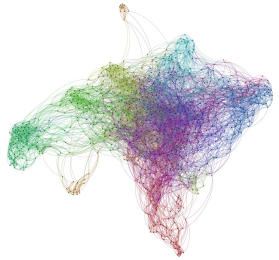
Graphs

Graph categories

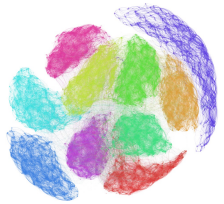
- Graphs constructed from data :
  - MNIST image network
  - GTZAN music network
  - 20NEWS text document network
  - 3D mesh points



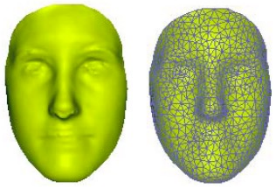
Graph of Text Documents
20newsgroups



GTZAN Music Graph



MNIST Image Graph



3D mesh points

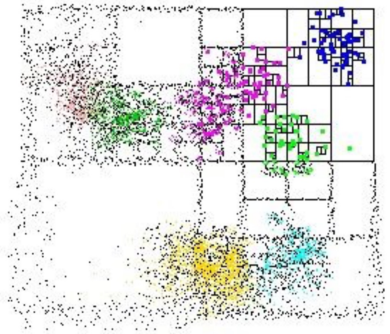
- Optimal graph construction : Unfortunately, no theoretical approach is available – it is empiricial and depends mostly on domain expertise knowledge and good common practice (later discussed).

- What is the computational time needed to construct a graph from data features ?
  - Exact construction : $O(n^2.d)$, n = num data, d = num features.

    for d = 1K, n = 1K ⇒ time < 1sec

    n = 100K ⇒ time = 1 min

    n = 1M ⇒ time > 1 hour



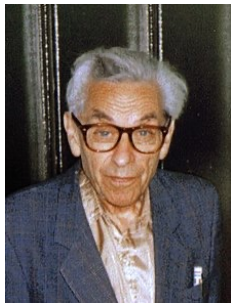
  - Approximate technique : kd-trees O(nlogn.d) with e.g. FLANN[1], a library for fast approximate nearest neighbor search in high-dimensional spaces.

[1] https://github.com/flann-lib/flann

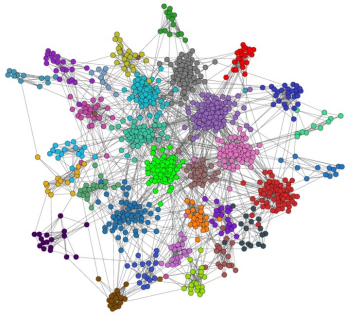
Graph categories

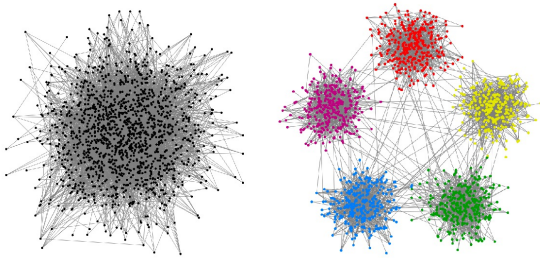
- Mathematical/simulated graphs
  - Erdos-Renyi graphs[1]
  - Stochastic block models (SBM)[2]
  - Lancichinetti-Fortunato-Radicchi (LFR) graphs[3]

Erdos-Renyi Network
Source: Wikipedia.



Paul Erdős
1913 – 1996





LFR
Source: Kojaku, 2018

SBM
Source: Abbe, JMLR'17

- Why using artificial networks?
  - Mathematical Modeling
    - Advantage : Precise control of your data model (estimate best performance given some data assumptions). No need to perform extensive experiments.
    - Limitation : Most data assumptions are often too restrictive, and it is not guaranteed that real-world data follow the given model assumptions.

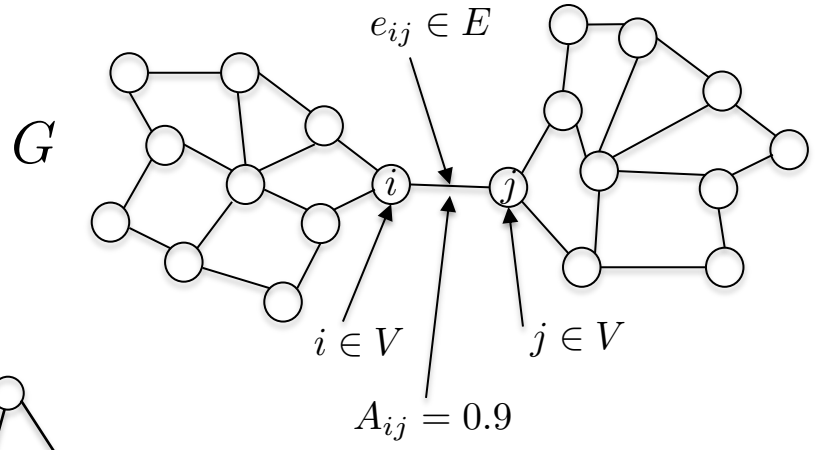
[1] Erdos, Renyi, On random graph, 1959
[2] Anderson, Wasserman, Faust, Building stochastic blockmodels, 1992
[3] Lancichinetti, Fortunato, Filippo, Benchmark graphs for testing community detection algorithms, 2008

Outline

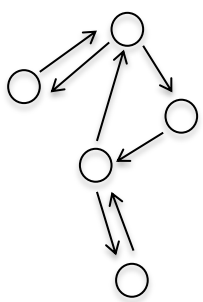
- Graph theory
- Graph categories
- **Basic definitions**
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Basic definitions

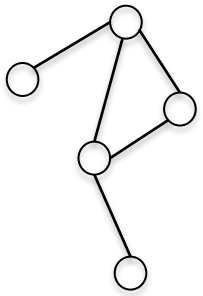
- Graphs are defined as G = (V,E,A) where
  - V is the set of vertices (or nodes) w/ |V| = n.
  - E is the set of edges.
  - A is the adjacency similarity matrix.



- Directed or undirected graphs :

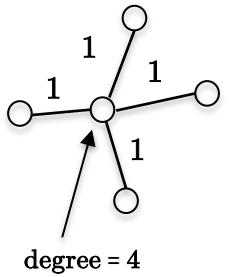


Directed graph

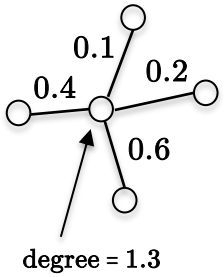


Undirected graph

- Node degree :
  - For binary graphs, $A_{ij} = \{0,1\}$ ⇒ degree = num of edges connected to a node
  - For weighted graphs, A_{ij} in [0,1] ⇒ degree is defined as $d_i = \sum_{j \in V} Aij$



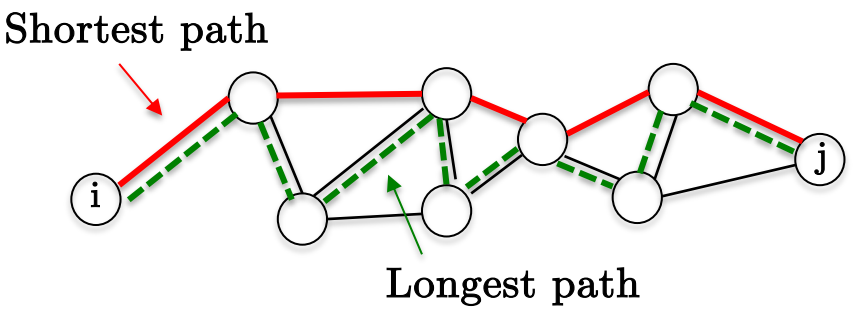
Binary graph

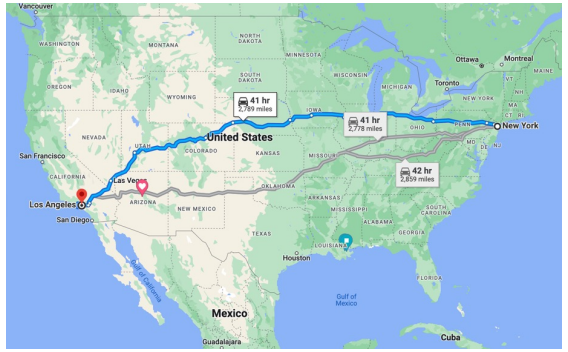


Weighted graph

Algorithms

- Standard graph algorithms (a.k.a. software engineering, no learning) :
  - Breadth-first search, depth-first search, minimum spanning tree, topological sorting, strongly connected components, graph colouring, maximum flow/minimum cut, graph matching, etc.
- Shortest path algorithm : Find a path on a graph with the smallest possible length.
  - Fast Algorithm : Dijkstra's algorithm[1]
  - Popular application is the road navigator product, e.g. from New York to Los Angeles.



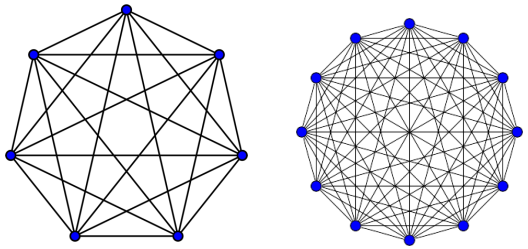


[1] Dijkstra, A note on two problems in connexion with graphs, 1959

Dense vs. sparse graphs

- Dense/complete/full graphs : Each vertex is connected to all other vertices.

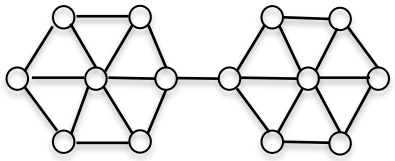
$$|E| = \frac{n(n-1)}{2} = O(n^2)$$ ☹



Dense graphs

- Sparse graphs : Each vertex is connected to a few other k≪n vertices.

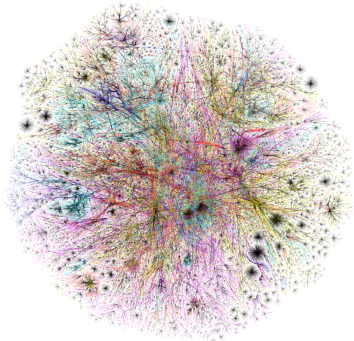
$$|E| = O(kn) = O(n)$$ ☺



Sparse graph

- Which graph is better -- dense or sparse?
  - Sparse networks are highly desirable for memory and computational efficiency.

    For instance, Internet has n = 4.73 billion pages (as of August 2016)
    - $|E| = n^2 = 10^{18}$ if Internet was full.
    - $|E| = k.n = 10^{11}$ as it is sparse with $k \approx 100$ (mean degree).



Internet

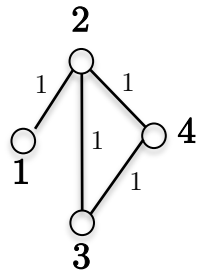
- Good news : Most real-world networks (e.g. social, brain, communication, etc) are sparse !
  - Because sparsity induces structure (a fully connected graph has no structure).

Adjacency matrix

- **Definition : Matrix A in G = (V,E,A) represents structural/topological information about the network. We have two classes of A :**
  - **Binary matrix : $A_{ij} \in \{0,1\}$**

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

G =

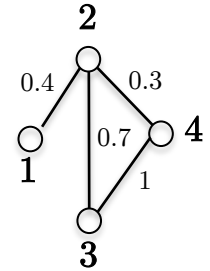


$$\Rightarrow A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

  - **Weighted matrix : $A_{ij} \in [0,1]$ (commonly normalized to 1)**

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

G =

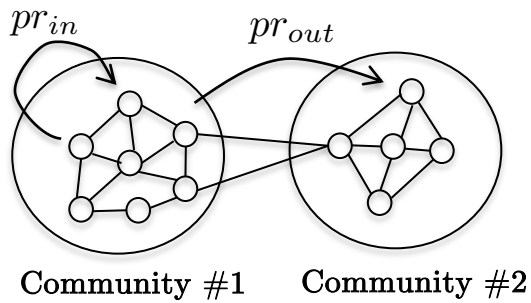


$$\Rightarrow A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0.4 & 0 & 0 \\ 0.4 & 0 & 0.7 & 0.3 \\ 0 & 0.7 & 0 & 1 \\ 0 & 0.3 & 1 & 0 \end{bmatrix} \end{array}$$

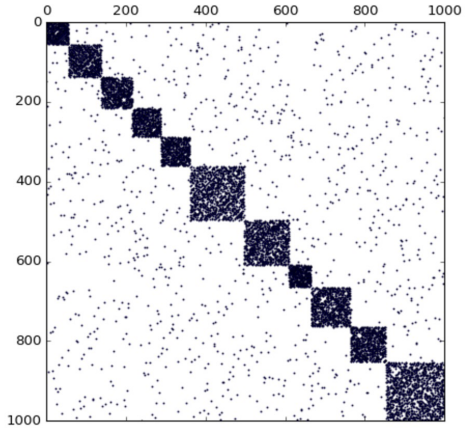
Lab 1 : LFR social networks

- Run code01.ipynb and synthesize LFR social networks.
  - Play with the mixing parameter μ :
    - μ small : Communities are well separated.
    - μ large : Communities are mixed.

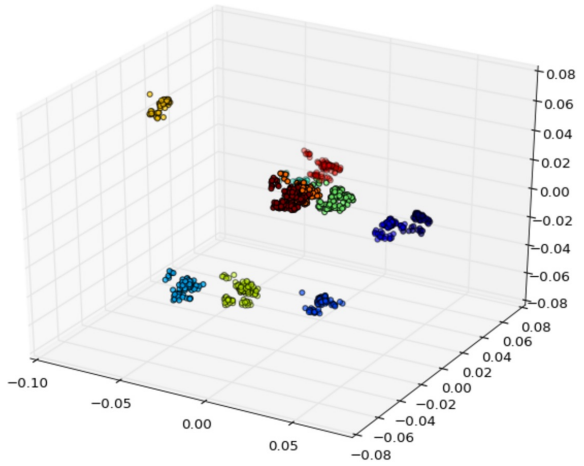
$$\mu = \frac{pr_{out}}{pr_{in}}$$




```
In [15]: # Plot same W but according to communities
         # Any structure?
         plt.figure(2)
         plt.spy(W,precision=0.01, markersize=1)
         plt.show()
```



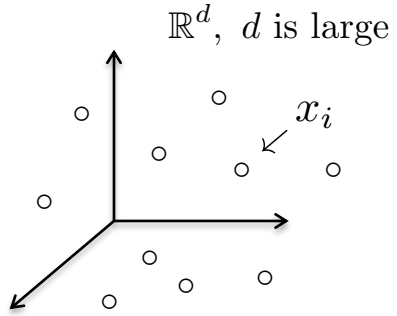

```
In [19]: # Visualize the social network in 3D
         fig = pylab.figure(4)
         ax = Axes3D(fig)
         ax.scatter(X, Y, Z, c=C)
         pyplot.show()
```



Outline

- Graph theory
- Graph categories
- Basic definitions
- **Curse of dimensionality and structure**
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Curse of dimensionality



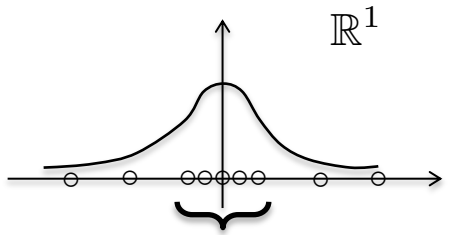
- What is the curse of dimensionality ?
  - In high dimensions, Euclidean distance between data becomes meaningless.
  - Theorem[1] : Suppose data are uniformly distributed in R^d,

    pick any data $x_i \in V$, we have :

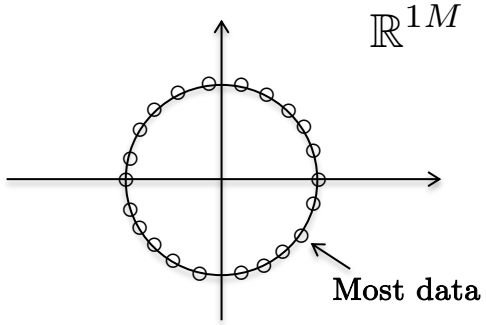
$$\lim_{d\to\infty} \mathbb{E}_{x_i}\left(\frac{\mathrm{d}^{\ell_2}_{\max}(x_i, V\setminus x_i) - \mathrm{d}^{\ell_2}_{\min}(x_i, V\setminus x_i)}{\mathrm{d}^{\ell_2}_{\min}(x_i, V\setminus x_i)}\right) = 0$$

Interpretation : All data are far away to each other with the same distance value !

- Besides, loss of intuition in high dimensions, e.g. with the Normal distribution :
  - In low-dim, most data are concentrated at the center.
  - In high-dim, most data are concentrated on the surface.



1-D Gaussian

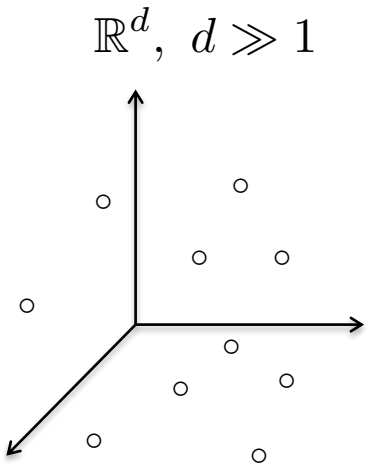


1,000,000-D Gaussian

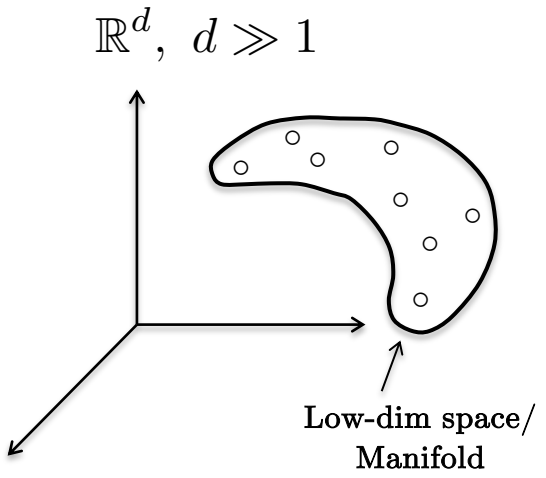
[1] Beyer, Goldstein, Ramakrishnan, Shaft, When is "nearest neighbor" meaningful? 1999

Blessing of structure

- What is the blessing of structure ?
  - Previously, the assumption "data are uniformly" distributed is actually not true for real-world data. Data have always properties, i.e. structures or invariances, such that they belong to a low-dimensional space called manifold where (geodesic) distances are meaningful.



Uniform distribution of data
No structure/randomness



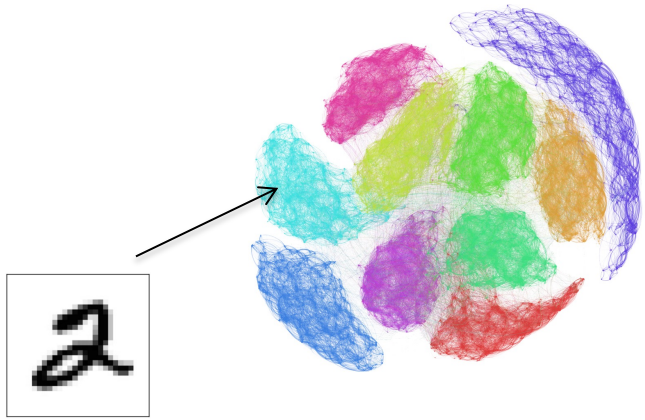
Non-Uniform distribution of data
Structure/inductive bias

Outline

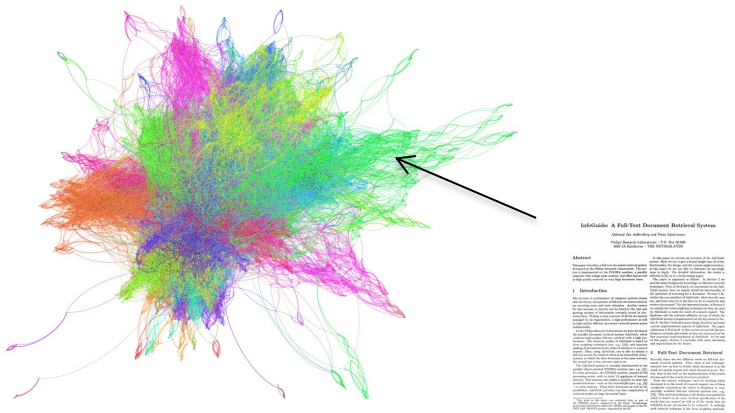
- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- **Manifolds and graphs**
- Spectral graph theory
- Graph construction
- Conclusion

Manifold learning

- It can be challenging to identify structures hidden in data because of
  - The curse of dimensionality (i.e. high-dimensional data).
  - Some data have easy structures, but most have complex ones.
- A class of algorithms that extracts low-dimensional patterns is manifold learning (later discussed).



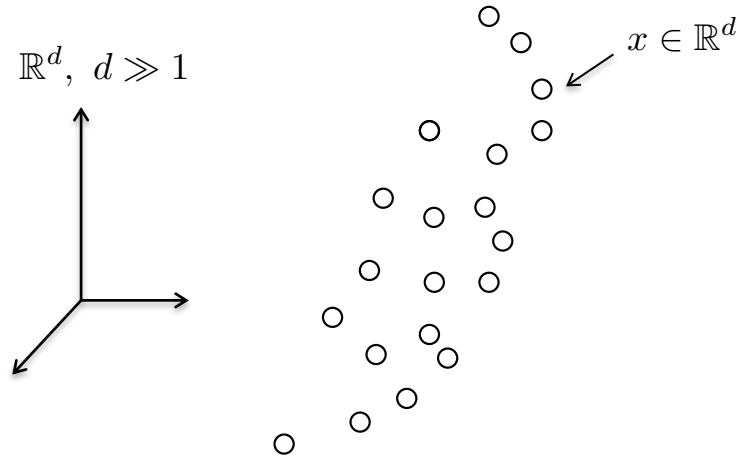
MNIST Image
Graph



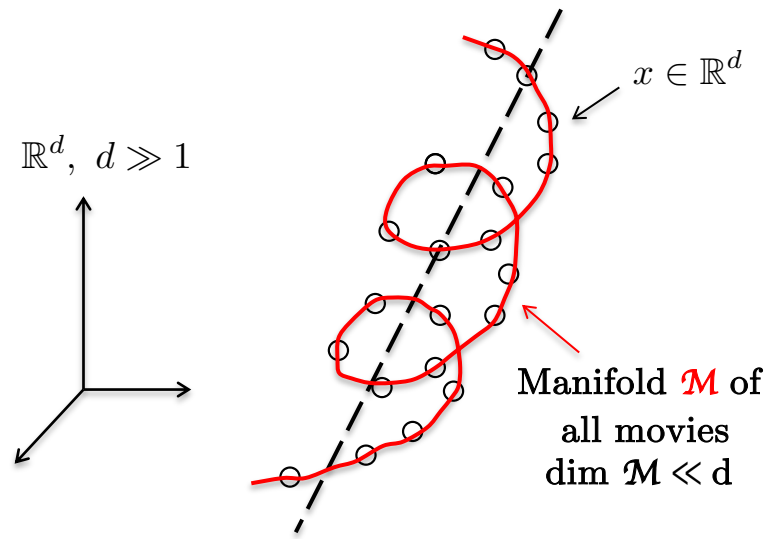
Graph of Text Documents
20newsgroups

From manifolds to graphs

- Manifold assumption : High-dimensional data are sampled from a low-dimensional manifold.
  - Example : Let x be a movie, each movie is defined by d features/attributes like genre, actors, release year, origin country, etc such that x in R^d. We can decide to make the assumption that all movies form a manifold $\mathcal{M}$ in R^d.

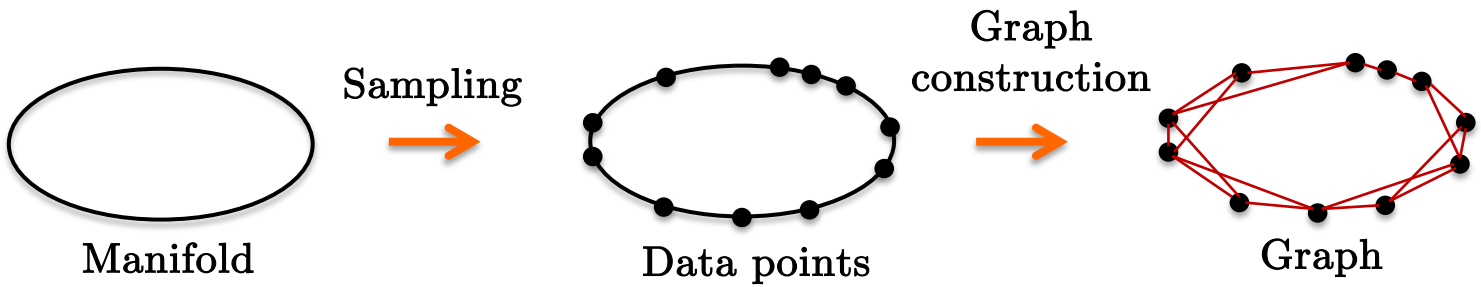


- Assumption validity : The manifold is a good working hypothesis for
  - Several types of data, including images, text documents, music, etc.
  - Most machine learning tasks e.g. classification, visualization, recommendation.
  - However, it can also be limited as it can be a too crude approximation of the (true) data distribution.



From manifolds to graphs

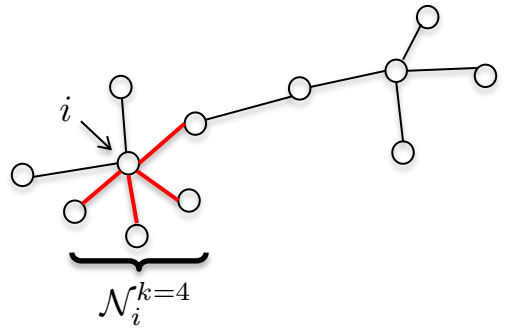
- Graphs can be regarded as a manifold sampling process.
  - The manifold information is represented by a neighborhood graph (observe that the manifold is never directly observed or constructed).



- Neighborhood graphs :
  - Most populars are k-NN graphs defined as :

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

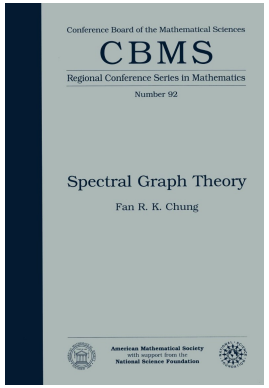
where $\text{dist}(x_i,x_j)$ is a distance (to be decided) between x_i and x_j, σ is the scale parameter (value depends on the dataset), and $\mathcal{N}_i^k$ is the neighborhood of data x_i.



Outline

- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- **Spectral graph theory**
- Graph construction
- Conclusion

Spectral graph theory

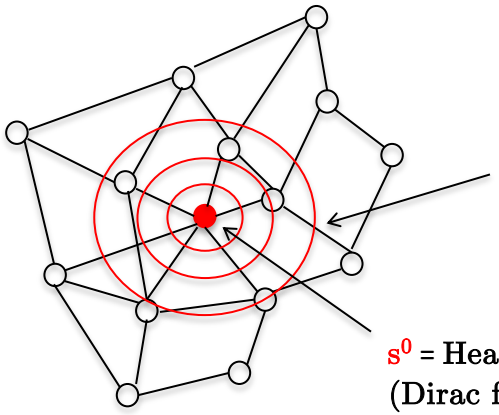


Fan Chung
SGT book
1997

- Given a graph G = (V,E,A), spectral graph theory (SGT) can
  - Find meaningful patterns that reveal multi-scale graph structures.
  - Process data defined on the graph domain (a.k.a. graph signal processing).
  - Boost performance of learning tasks s.a. clustering, classification, recommendation, etc.

- The most fundamental tool in SGT is the graph Laplacian operator L.
  - Why is the Laplacian useful?
    - It is the central operator of diffusion processes (on graphs).
    - Basis functions of this operator are the well-known Fourier modes (later discussed).

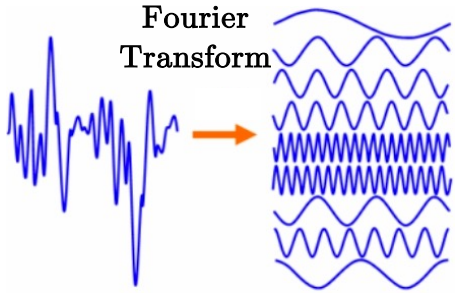
Pierre-Simon Laplace
(1749–1827)



Heat propagation using
Laplacian operator :

$$s^{t+1} = Ls^t$$

s^0 = Heat source
(Dirac function)



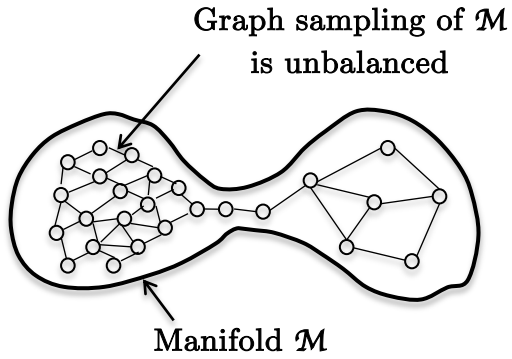
Graph Laplacian

- Un-normalized/combinatorial graph Laplacian :

$$L_{un} = D - A \quad \text{with D is the degree matrix :} \quad D = \text{diag}(d_1, ..., d_n),$$
$$n \times n$$
$$n = |V| \qquad d_i = \sum_j A_{ij}$$

- Normalized Laplacian (most popular) :
  - Robust w.r.t. unbalanced sampling

$$L = D^{-1/2} L_{un} D^{-1/2} = I_n - D^{-1/2} A \; D^{-1/2}$$



- Random Walk Laplacian (for Google PageRank, later discussed) :

$$L = D^{-1} L_{un} = I_n - D^{-1} A$$

- All Laplacians are diffusion operators.

Graph spectrum

- Spectral graph theory can extract the modes of variation of the graph system.
  - How? With the eigenvalue decomposition (EVD) of the Laplacian operator L :

$$L = U\Lambda U^T \in \mathbb{R}^{n\times n} \text{ with}$$
$$U = [u_1, ..., u_n] \in \mathbb{R}^{n\times n},$$
$$U^T U = I_n, \text{ i.e. } \langle u_k, u_{k'} \rangle = \begin{cases} 1 & k = k' \\ 0 & \text{otherwise} \end{cases},$$
$$\Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \in \mathbb{R}^{n\times n},$$
$$0 = \lambda_{\min} = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_1 \leq 2$$

  - Interpretation:
    - 图的震动模式 u_k : Laplacian eigenvectors a.k.a. Fourier functions, i.e. vibration modes of the graph.
    - 图的震动速度 λ_k : Laplacian eigenvalues a.k.a. frequencies of the Fourier functions, i.e. how fast u_k vibrate.
  - EVD answers the famous question 'Can One Hear the Shape of a Drum'?[1]

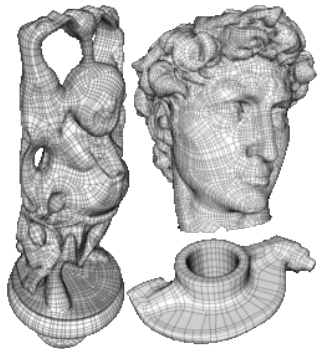
[1] Kac, Can One Hear the Shape of a Drum, 1966

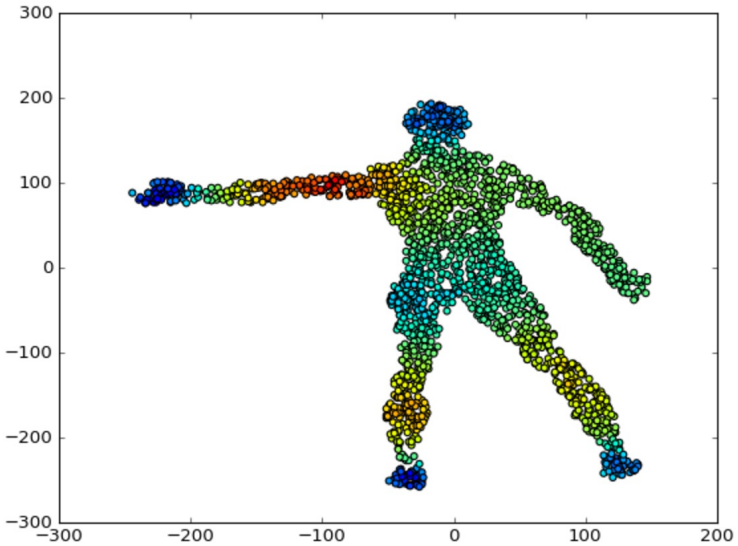
Lab 2 : Spectrum of point cloud and grid

- Run code02.ipynb and visualize the Fourier functions of a human body graph and a regular grid.
- What is the main property of the smallest and largest eigenvectors?
  - Smallest eigenvectors ⇒ Smoothest modes of vibration, i.e. low-frequency information.
  - Largest eigenvectors ⇒ Highest frequencies of the graph, i.e. details or noise.

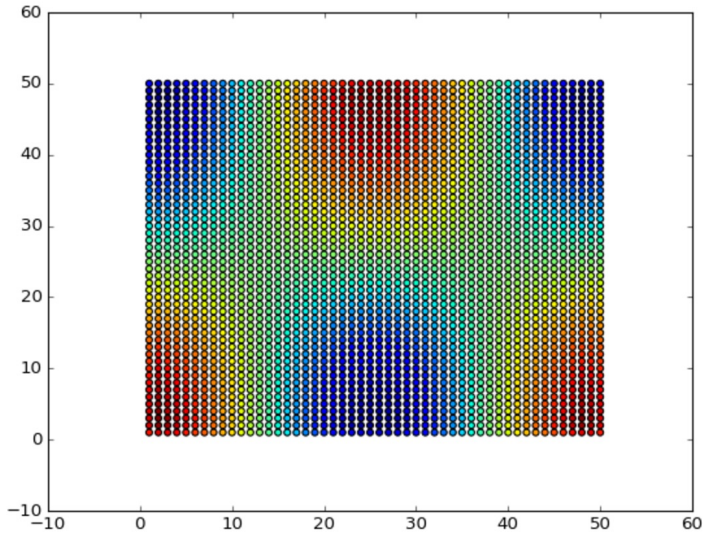
In [5]:
```
# Compute graph Laplacian
L = graph_laplacian(W)

# Compute modes of variations of graph system = Fourier functions
lamb, U = scipy.sparse.linalg.eigsh(L, k=9, which='SM')
```



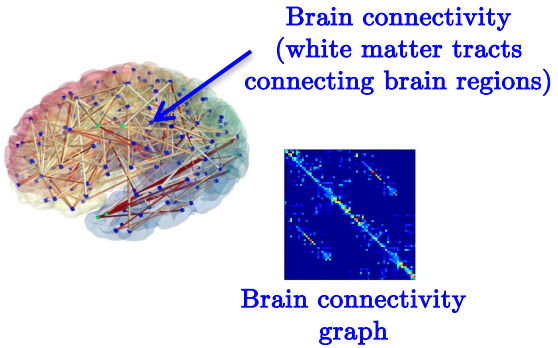


Fourier mode from a Graph =
Set of points or meshes in graphics
(e.g. 2D/3D shape recognition)

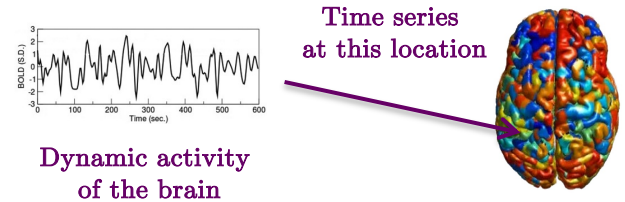


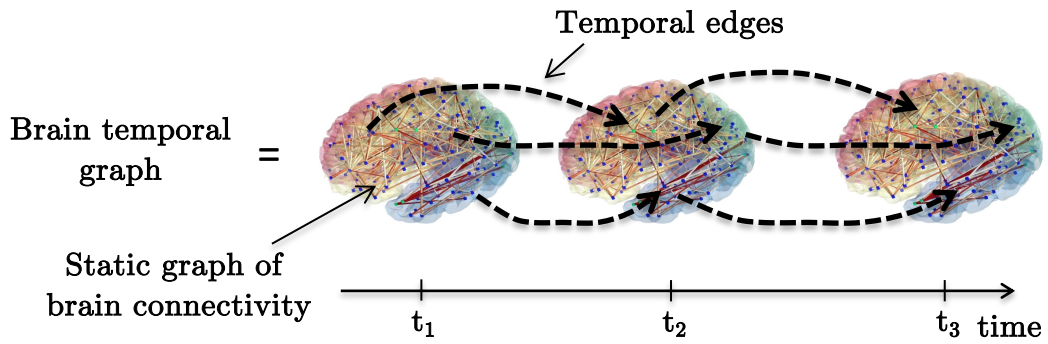
Fourier mode from a Graph =
Regular grid
(e.g. JPEG image compression,
most used technique)

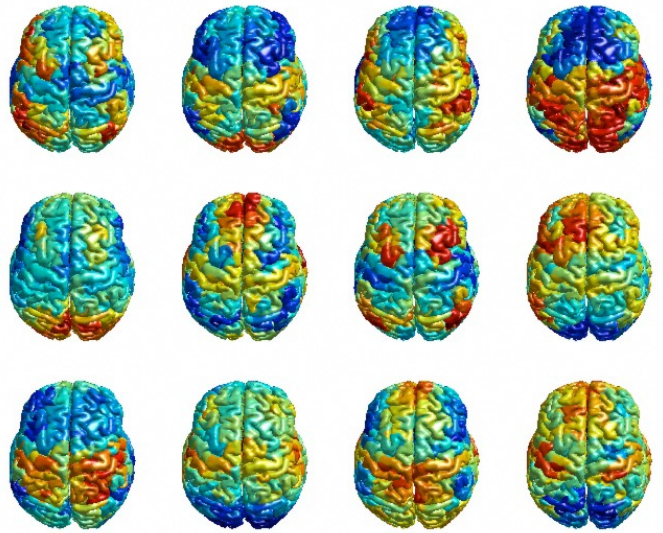
Spectrum of human brain



- Goal : Find brain activation patterns from structural MRI, a.k.a. brain resting states (brain structure implies brain function).
- Methodology : Given G (brain connectivity graph) ⇒ design a temporal graph ⇒ compute Laplacian L ⇒ compute eigenvectors u_k ⇒ visualize brain temporal activation patterns.
- Result : u_k represent the dynamic patterns related to basic functional brain tasks s.a. vision, body motor, language, etc.
- How to construct the brain temporal graph?







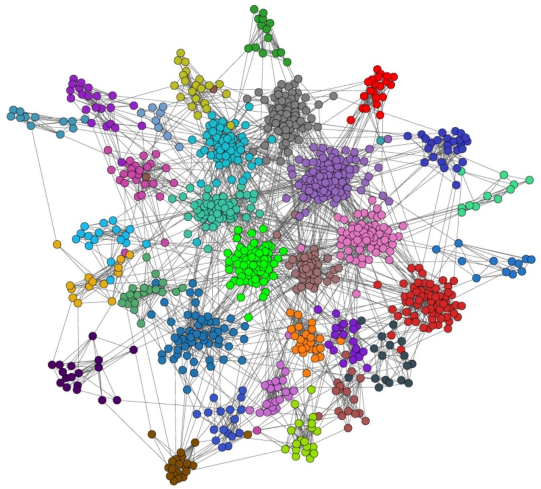
Laplacian eigenvectors u_k
a.k.a. brain resting states
(video)

Outline

- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- **Graph construction**
- Conclusion

How to construct graphs from data?

- Three basic questions
  - Which type of graphs?
  - Which data distance?
  - Which data features?



- Optimal graph construction
  - No universal recipe is available (no theory).
  - It depends on data and analysis task (empirical).
  - Domain expertise and good practice are essential.

Type of constructed graphs

- ε-graphs : Fully connected graphs, not practical.
- Neighborhood graphs
  - k-NN graphs, i.e. sparse graphs by design.
  - Parameters
    - k : number of nearest neighbors, a common value is between {5,50}.
    - σ : positive scale parameter
- Two options to compute scale σ
  - Global scale : σ = mean distance of all k^{th} neighbors.
  - Local scale[1] : σ_i = distance of the k^{th} neighbor for node i.

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

Adjacency matrix with global scale

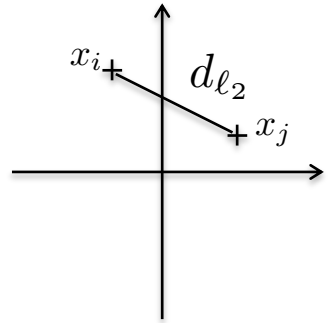
$$A_{ij} = \begin{cases} e^{-\frac{\text{dist}(x_i,x_j)^2}{\sigma_i \sigma_j}} & \text{if } j \in \mathcal{N}_i^k \\ 0 \text{ otherwise} \end{cases}$$

Adjacency matrix with local scale

[1] Zelnik-Manor, Perona, Self-tuning spectral clustering, 2004

Distance definition

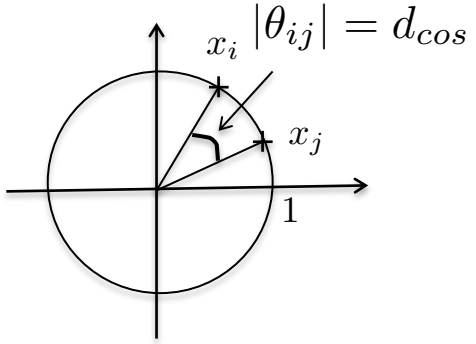
- Euclidean/L2 distance :
  - Good distance for low-dim data, e.g. d < 10.
  - Good distance for high-dim data with linearly separable data (e.g. MNIST).

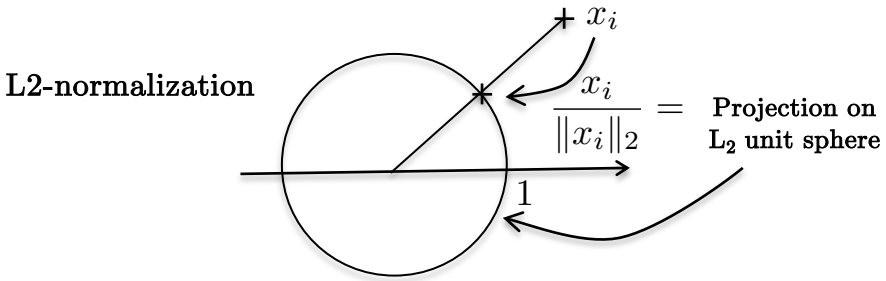


$$d_{\ell_2}(x_i, x_j) = \|x_i - x_j\|_2 = \sqrt{\sum_{m=1}^{d} |x_{i,m} - x_{j,m}|^2}$$

- Cosine/dot product distance :
  - Good distance for high-dim sparse data (e.g. text documents)

$$d_{cos}(x_i, x_j) = \left| \cos^{-1}\left(\frac{\langle x_i, x_j \rangle}{\|x_i\|_2 \|x_j\|_2} \right) \right| = |\theta_{ij}|$$





$$X = \underset{n \times d}{\begin{bmatrix} x_1 \\ x_i \\ x_n \end{bmatrix}} \quad \|x_i\|_2 = 1$$

- Other analytical distances : Kullback-Leibler distance (information theory), Wasserstein distance (optimal transport), etc.

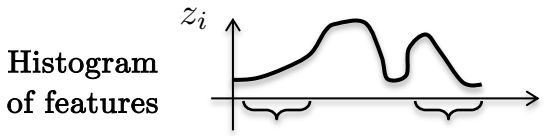
Data features

- Types of data features
  - Raw features (e.g. movie features such as genre, actors, year, etc)
  - Hand-crafted features (e.g. SIFT in computer vision, etc)
  - Learned features (PCA, NMF, sparse coding, deep learning, etc)
- It is generally unsuccessful to directly use the raw features for graph construction.
  - Issues are noise, unbalanced scaling, lack of expressiveness, curse of dimentionality, etc
- For successful graph construction, raw data should be transformed into meaningful data representation by designing new features, s.a. handcrafted or learned features.

$$x_i \in \mathbb{R}^d \xrightarrow{\text{Feature Extraction}} z_i \in \mathbb{R}^e$$

Raw data
Ex: Image pixels

New features
Ex: Car features
Flower features



$$A_{ij} = e^{-\frac{\text{dist}(x_i, x_j)^2}{\sigma^2}}$$

$$\Downarrow$$

$$A_{ij} = e^{-\frac{\text{dist}(z_i, z_j)^2}{\sigma^2}}$$

Standard pre-processing

- Center data (along feature dimension) : zero-mean property

$$x_i \leftarrow x_i - \text{mean}(\{x_i\}) \in \mathbb{R}^d$$

- Normalize data variance (along feature dimension) : z-scoring property

$$x_i \leftarrow x_i \ / \ \text{std}(\{x_i\}) \in \mathbb{R}^d$$

$$\text{with } \text{std}(\{x_i\})^2 = \text{mean}(\{\left(x_j - \text{mean}(\{x_i\})\right)^2\})$$

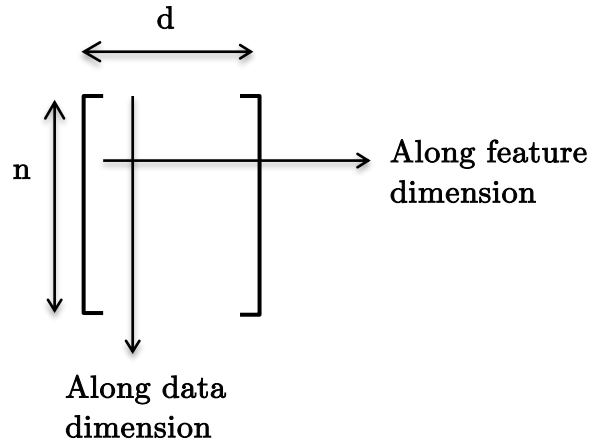
- Project data on L2-sphere (along feature dimension or data dimension) :

$$x_i \leftarrow x_i \ / \ \|x_i\|_2 \in \mathbb{R}^d$$

- Normalize max and min of feature value :

$$x_i \leftarrow \frac{x_i - \min(\{x_i\})}{\max(\{x_i\}) - \min(\{x_i\})} \in [0,1]^d$$

X =

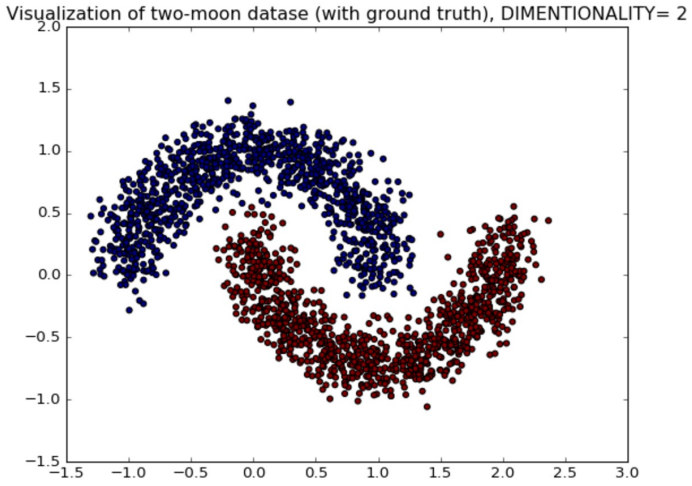


Lab 3 : Graph construction for two-moon

- Run code03.ipynb and study data pre-processing, construction of k-NN graphs, visualization of distances, adjacency matrix, and graph quality with clustering accuracy.

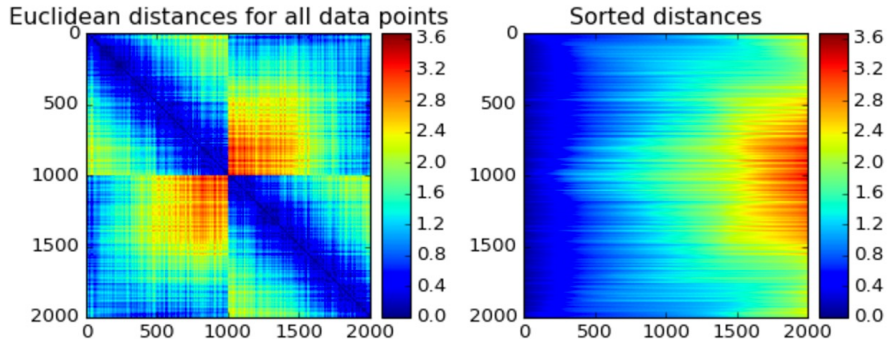
```
# Visualize in 2D
plt.figure(30)
size_vertex_plot = 20.
plt.scatter(X[:,0], X[:,1], s=size_vertex_plot*np.ones(n), c=C)
plt.title('Visualization of two-moon datase (with ground truth), DIMENTIONALITY= ' + str(dim))
plt.show()
```

```
(2000, 2) (2000,)
```




```
In [9]: # Visualize distances
        fig, (ax1, ax2) = plt.subplots(1,2)
        #fig.suptitle('Title of figure 2', fontsize=15)

        ax1.set_title('Euclidean distances for all data points')
        im1 = ax1.imshow(Dnot_sorted, interpolation='nearest')
        divider1 = make_axes_locatable(ax1)
        cax1 = divider1.append_axes("right", size="10%", pad=0.1)
        ax1.get_figure().colorbar(im1, cax=cax1)
```

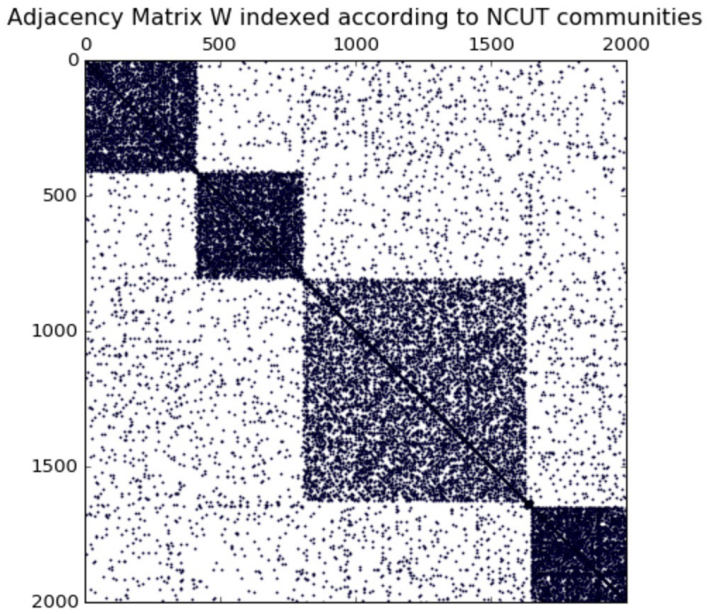


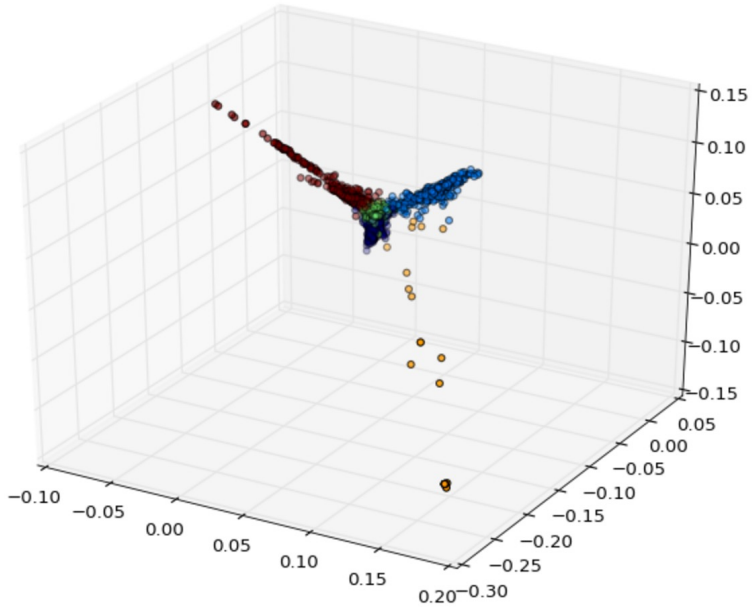
Lab 4 : Graph construction for text documents

- Run code04.ipynb and construct graph text documents.

```
In [9]: # Compute the k-NN graph with Cosine distance
        W_cosine = construct_knn_graph(X,10,'cosine')
```

```
k-NN graph with cosine distance
```





Outline

- Graph theory
- Graph categories
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- **Conclusion**

Summary

- Graphs can represent complex and heterogenous relationships between data.
  - They help to go beyond the standard assumption that data are i.i.d. (independent and identically distributed) as they explicitly leverage the connections between pairs of data.
  - Any dataset and task that use explicit relations between data is a graph-based task.
    - When we begin looking for graphs, we discover they are ubiquitous! ☺
- Graphs offer an augmented representation of data.
  - With data feature X and data relationship depicted by G = (V,E,A).

Summary

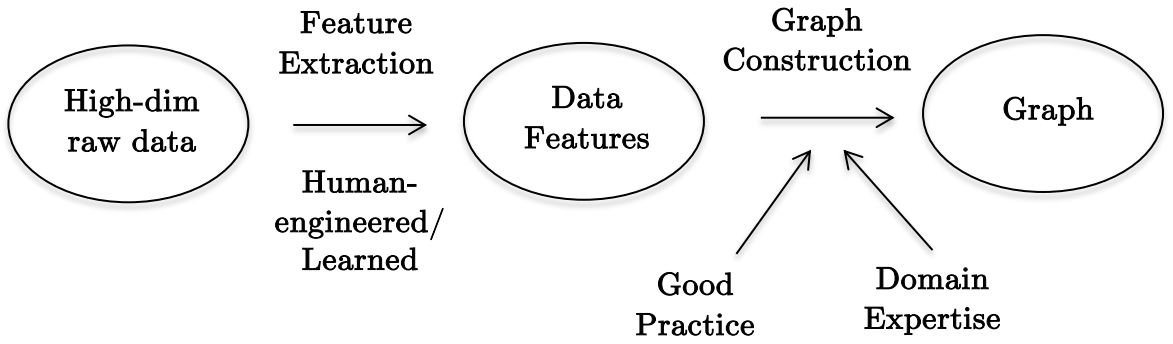
- First fundamental tool of graph science :
  - Adjacency matrix A
    - It reveals global structures in data relationship (graph spectrum).
    - It can visualize graphs in 2D or 3D Euclidean spaces (later discussed).
    - It can boost performance of machine learning techniques (later discussed).
- Second fundamental tool of graph science :
  - Graph Laplacian matrix L
    - It is a diffusion operator -- It propagates information on graphs (parabolic PDEs).
    - It is used to reveal modes of variation of the graph system.
    - It is used for image compression (jpeg), neuroscience (brain activity), positional encoding (later discussed), etc.

Pipeline

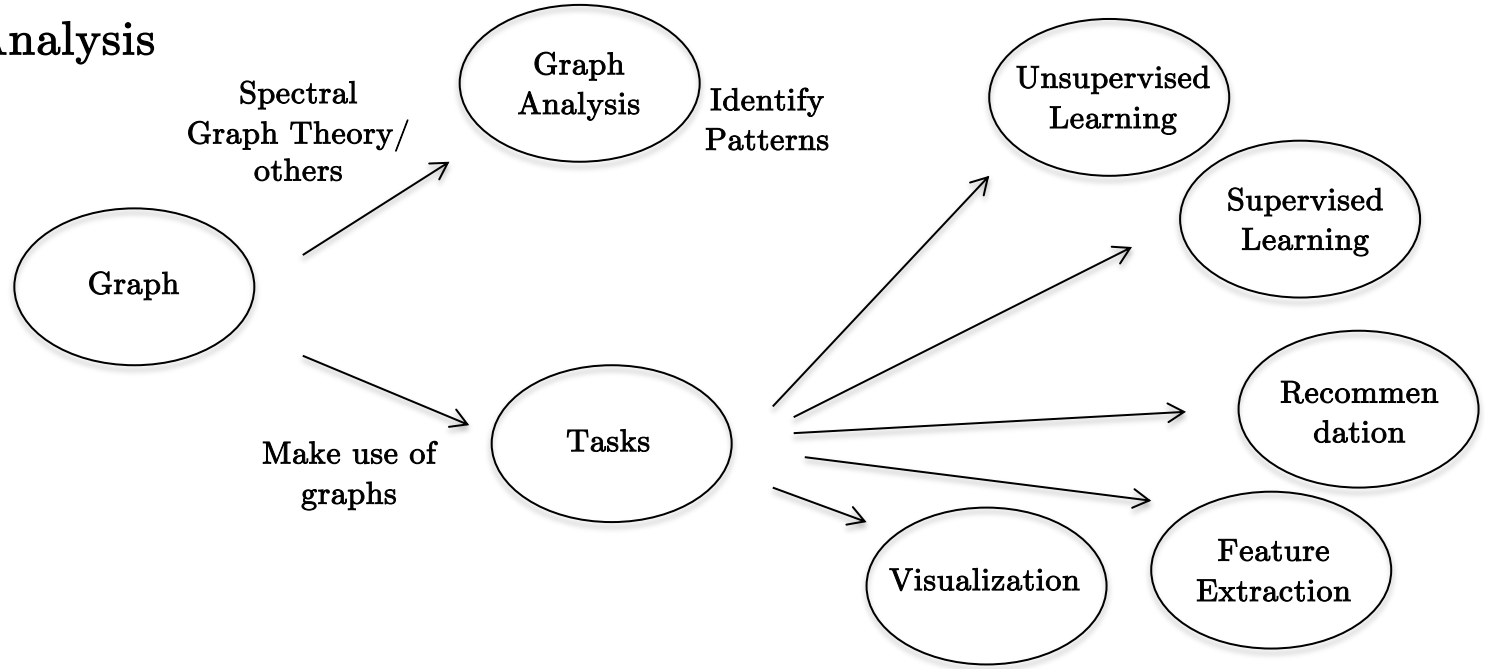
- Step #1: Data ⇒ Graph

  Skip this step if graph is given,

  e.g. molecule, social network, etc



- Step #2: Graph ⇒ Analysis



Questions?