Data Science Training

November 2017

Graph Science

Xavier Bresson

Data Science and AI Research Centre
NTU, Singapore

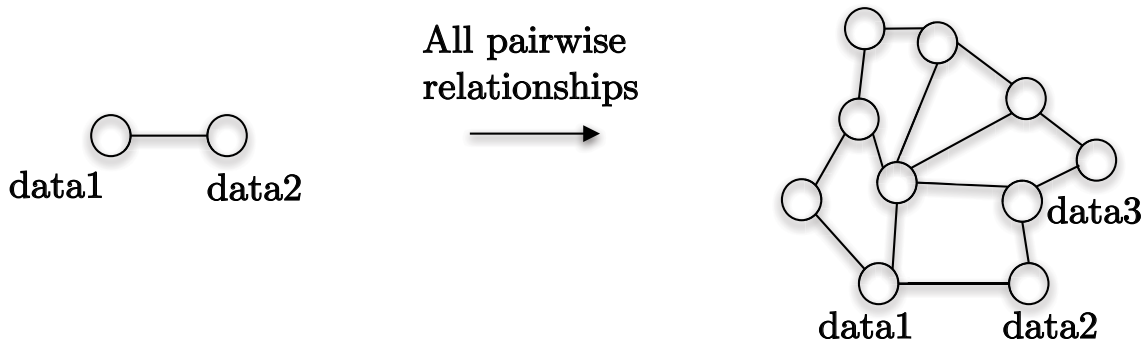
http://data-science-optum17.tk

Outline

- **Graph Science and Graph Theory**
- Classes of Networks
- Basic Definitions
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- Spectral Graph Theory
- Construct Graphs from Data
- Conclusion

Graph/Network Science

- Definition of graph/network: mathematical models representing **pairwise relations between objects/data:**



Q: Why are they useful?

Graphs offers a global view of data structure

⇒ *Extract global meaningful patterns, insights about data*

⇒ *Increase performances, ex: classification from 5-20% (later discussed)*

⇒ *Easy to use, slight increase of computational time*

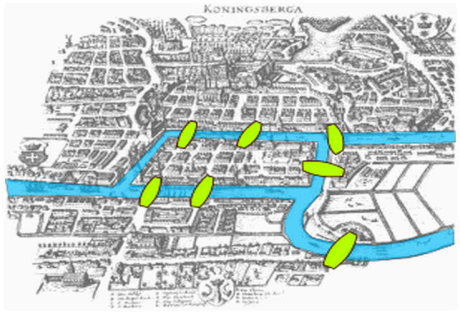
⇒ *Some tasks are only designed on networks, e.g. Google pagerank*

Graph Science = Graph Theory

Q: When did it start?

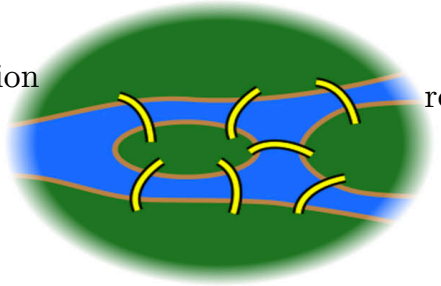
- **History of graph theory:** Graphs have been studied since 1736, starting with *Leonhard Euler* and the famous problem of "Seven Bridges of Königsberg":

  Q: Can we find a path through the city that cross each bridge once and only once? Yes if the graph has even degree.

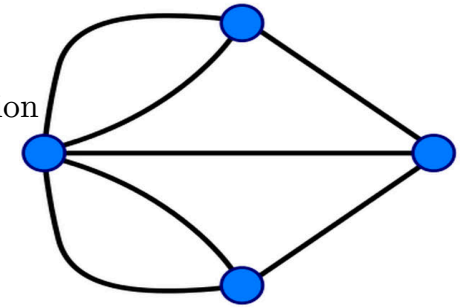


Königsberg City

Simplification →



Graph representation →



Source: Wikipedia.

A: Not possible. Needs cycles ⇒ All vertices must have even degree.

- **Graph theory offers many analysis tools** to use networks for all kind of applications: from clustering to classification, visualization, recommendation, deep learning, etc.

Outline

- Graph Science and Graph Theory
- **Classes of Networks**
- Basic Definitions
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- Spectral Graph Theory
- Construct Graphs from Data
- Conclusion

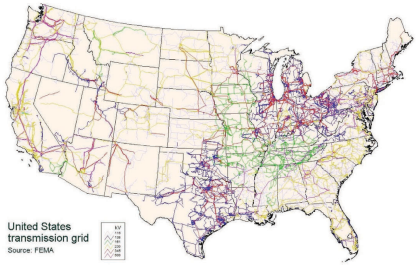
Classes of Graphs/Networks

Q: Can you cite a few networks?

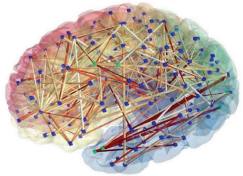
- **Natural Graphs:**
  - *(1) Social networks:* Facebook, LinkedIn, Twitter
  - *(2) Biological networks:* Brain connectivity and functionality, Gene regulatory networks
  - *(3) Communication networks:* Internet, Networking Devices
  - *(4) Transportation networks:* Trains, Cars, Airplanes, Pedestrians
  - *(5) Power networks:* Electricity, Water

Minnesota Road Network

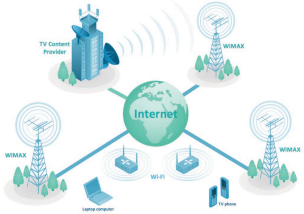
Facebook



US Electrical Network



Brain Connectivity



Telecommunication Network

=

Graphs/networks

- **Essential data $$ lie on network structures**, like medical, social, communication data.

Terminology: Natural graphs means **no** graph construction.

Classes of Graphs/Networks

- **Constructed Graphs (from Data)**

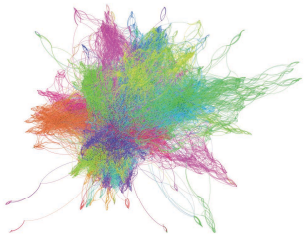
  Examples:

  (1) MNIST image network

  (2) GTZAN music network

  (3) 20NEWS text document network

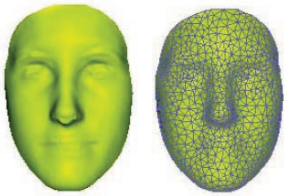
  (4) 3D mesh points



Graph of Text Documents
20newsgroups

GTZAN Music Graph

MNIST Image Graph



3D mesh points

- **Graph Construction: No universal recipe** but good common practices (later discussed) and domain expertise knowledge.

Q: How much time you need to construct a network from data?

- **Computational time:** May be time consuming $O(n^2)$, n=#data.

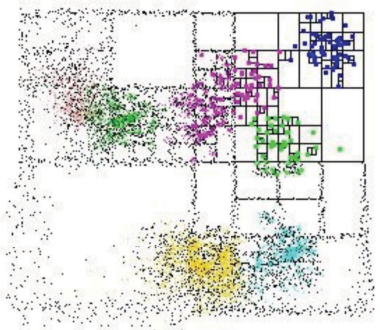
  Exs: d=1K, n=1K $\Rightarrow$ time $<$ 1sec

  n=100K $\Rightarrow$ time ~ 1 min

  n=1M $\Rightarrow$ time $>$ 1 hour

- **Approximate technique:** FLANN is a library for performing fast *approximate* nearest neighbor searches in high dimensional spaces.

FLANN: http://www.cs.ubc.ca/research/flann



kd-tree

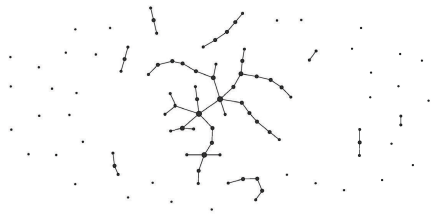
Classes of Graphs/Networks

- **Mathematical/Simulated Graphs:**

  (1) Erdos-Renyi graphs (1959)

  (2) Stochastic blockmodels [Faust-Wasserman '92]

  (3) Lancichinetti-Fortunato-Radicchi (LFR) graphs (2008)



Erdos-Renyi Network
Source: Wikipedia.

Q: Why using artificial math networks?

- **Mathematical Models:**
  - *Advantages:* Precise control of your data analysis model (best performances, data assumptions). No need to perform extensive experiments! (big issue with deep learning)
  - *Limitations:* Most data assumptions are too restrictive, and it may be hard to check if your data follow the model assumptions.

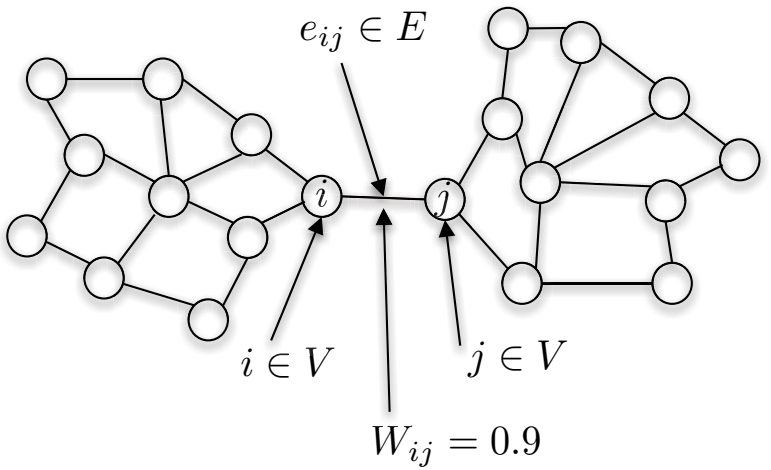
Outline

- Graph Science and Graph Theory
- Classes of Networks
- **Basic Definitions**
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- Spectral Graph Theory
- Construct Graphs from Data
- Conclusion

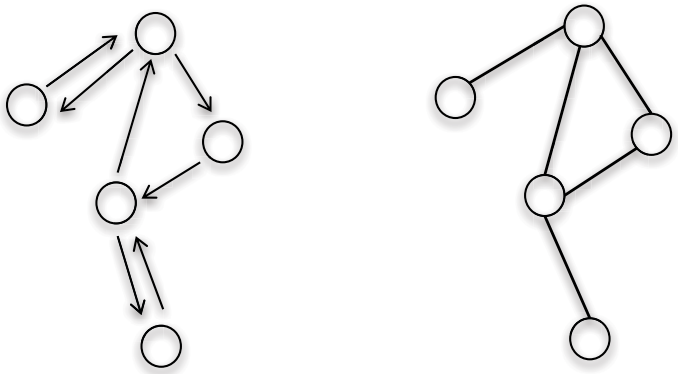
Basic Definitions

- **Graphs: Fully defined** by $G=(V,E,W)$:
  - V set of vertices,
  - E set of edges, and $|V|=n$,
  - W similarity matrix.

G



- **Directed/undirected graphs:**



directed **undirected**

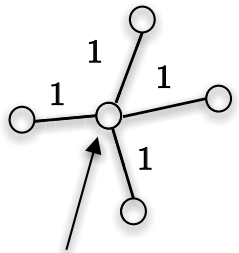
Note: In this workshop, we will mostly talk about *undirected* networks.

Basic Definitions

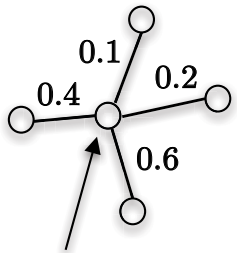
- **Vertex Degree:**

  (1) Binary graphs ($W_{ij}=\{0,1\}$): degree= #edges connected to a vertex

  (2) Generic graphs (W_{ij} in $[0,1]$): degree= $d_i = \sum_{j \in V} W_{ij}$



Binary graph



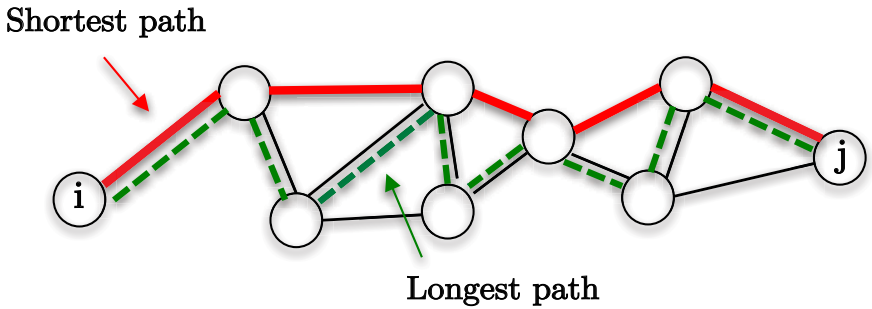
Generic graph

- **Shortest path:** A path on a graph with the smallest possible distance.

  Fast Algorithm: Dijkstra's algorithm (1956).

Q: Do you know a popular application?

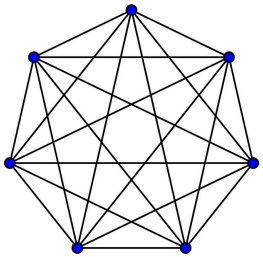
A: *Road navigator,* e.g. Lausanne to Venezia.

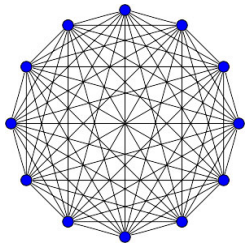


Full vs. Sparse Graphs

- **Full/Complete Graphs:** Each vertex is connected to *all* other vertices.

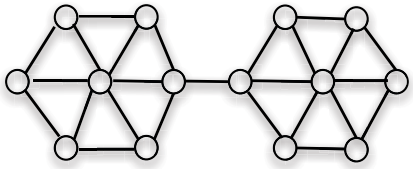
$$|E| = \frac{n(n-1)}{2} = O(n^2)$$





- **Sparse Graphs:** Each vertex is connected to *a few* other vertices (k=10-50).

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



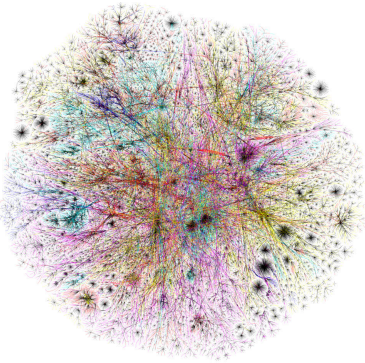
Q: Full or sparse?

- A: Sparse networks are highly **desirable** for *memory* and *computational* efficiency.

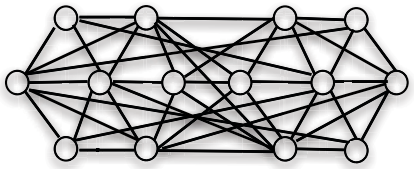
  Ex: **Internet**, n= 4.73 billion pages (August 2016)

  $|E| = n^2 = 10^{18}$ if it was full.

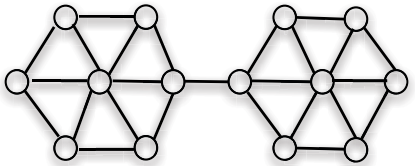
  $|E| = k.n = 10^{11}$ as it is (very) sparse.



- **Good news:** most natural/real-world networks (Facebook, Brain, Communication) are sparse. Besides, sparsity ⇔ structure:



Full graph



Sparse graph

Adjacency/Similarity Matrix W

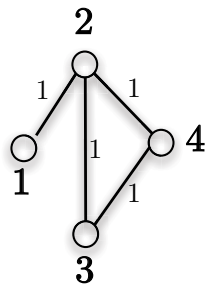
- **Definition:** Matrix W in $G=(V,E,W)$ actually contains **all information about your network**. There are two choices of W:

  (1) Binary W: W_{ij} in {0,1}

  (2) Weighted W: W_{ij} in [0,1] (commonly normalized to 1)

(1) Binary W:

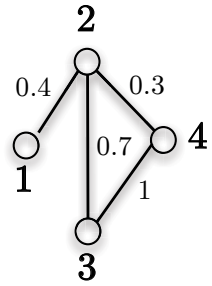
$$W_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

$G=$ 

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

(2) Weighted W:

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

$G=$ 

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

Demo: Synthetize Social Networks

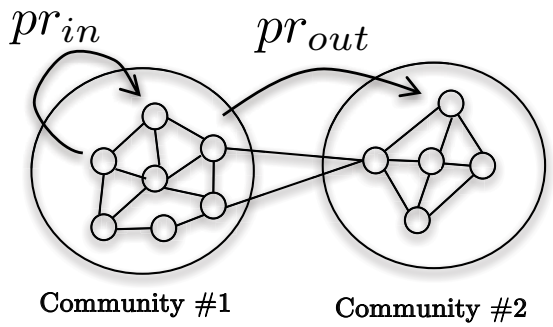
- Run **code01.ipynb**

  Synthesize LFR social networks: Play with the mixing parameter μ:

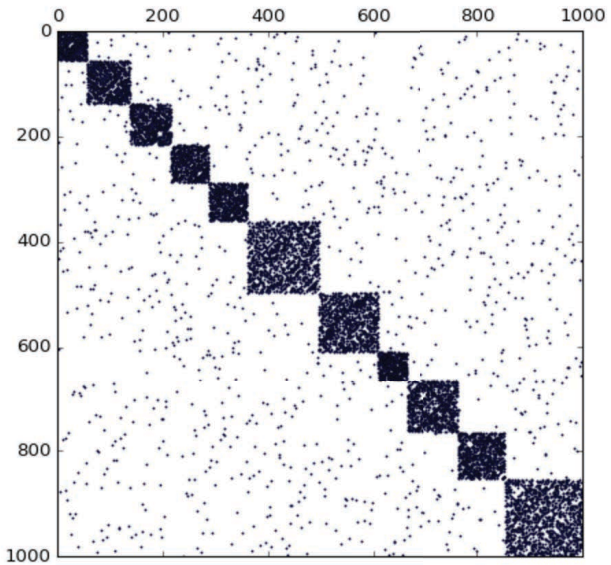
  (1) μ small: communities well separated.

  (2) μ large: communities are mixed up.

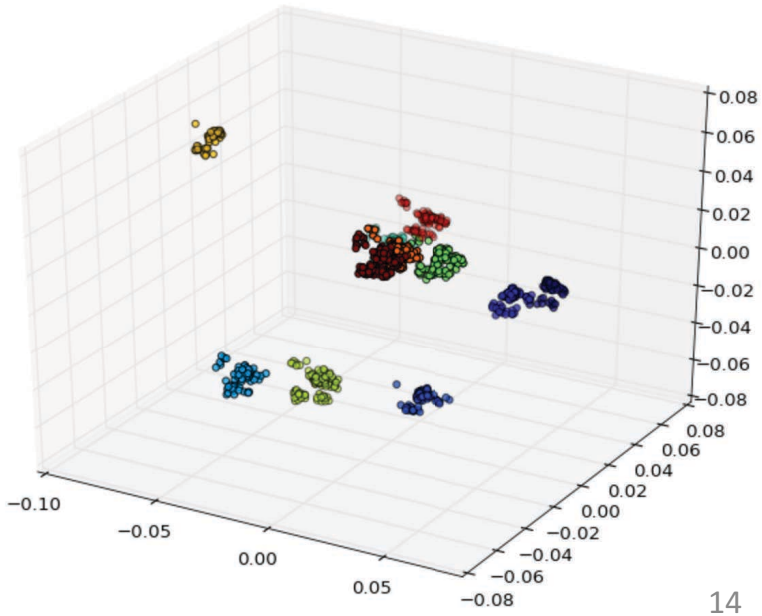
$$\mu = \frac{pr_{out}}{pr_{in}}$$




```
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 Science and Graph Theory
- Classes of Networks
- Basic Definitions
- **Curse of Dimensionality and Structure**
- Manifolds and Graphs
- Spectral Graph Theory
- Construct Graphs from Data
- Conclusion

Curse of Dimensionality

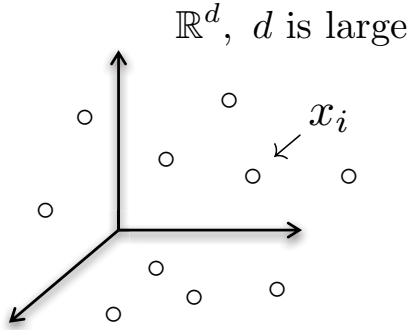
Q: What is the curse of dimensionality?

- A: In **high** dimensions, (Euclidean) distances between data is **meaningless** ⇒ all data are close to each other!

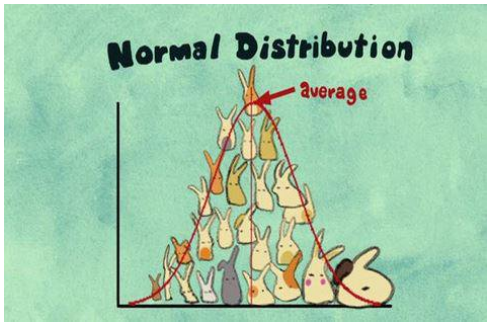
  Result [Beyer'98]: Suppose data are **uniformly** distributed in R^{d},

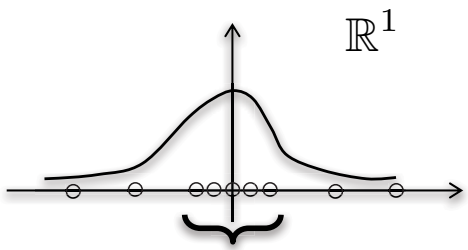
  Pick any data x_i then we have:

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

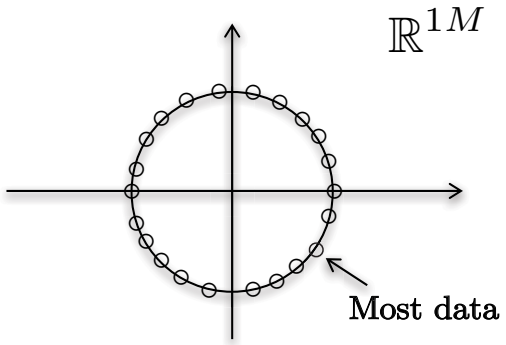


- **Loss of intuition:** Gaussian distribution of data.
  - In low-dim, most data are concentrated at the **center**.
  - In high-dim, most data are concentrated at the **surface**.





1-D Gaussian

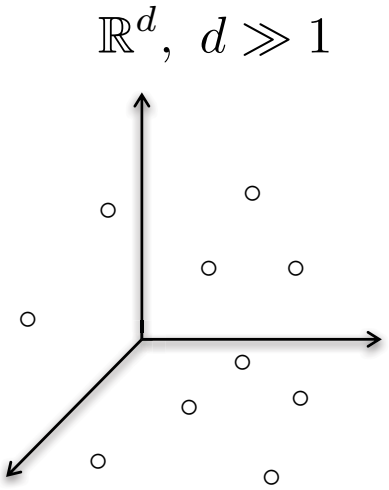


1,000,000-D Gaussian

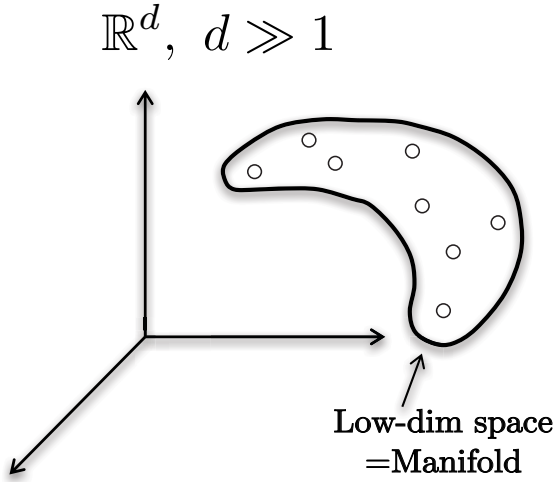
Blessing of Structure

Q: What is the blessing of structure?

- **Good news:** Assumption "data are uniformly" distributed is **not** true for real-world data. Data have always some structures in the sense that they belong to **a low-dimensional space called manifold** ⇒ distances on this surface are **meaningful!**



Uniform distribution
of data
No structure
Randomness



Non-Uniform distribution
of data
Structureness

Outline

- Graph Science and Graph Theory
- Classes of Networks
- Basic Definitions
- Curse of Dimensionality and Structure
- **Manifolds and Graphs**
- Spectral Graph Theory
- Construct Graphs from Data
- Conclusion

Manifold Learning

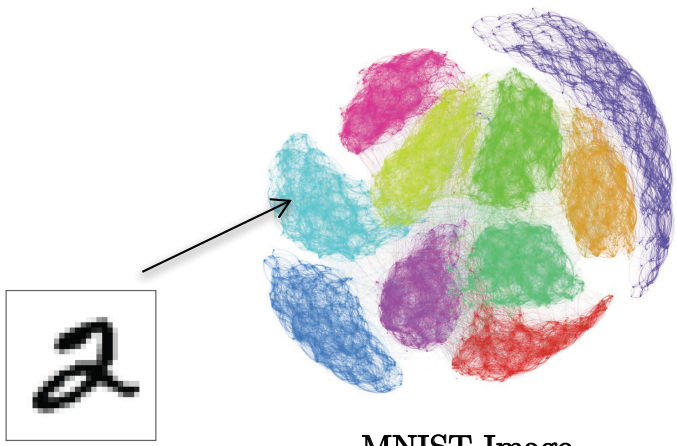
- **Big challenge:** It is difficult to discover the structures hidden in the data because:

  (1) High-dimensional data

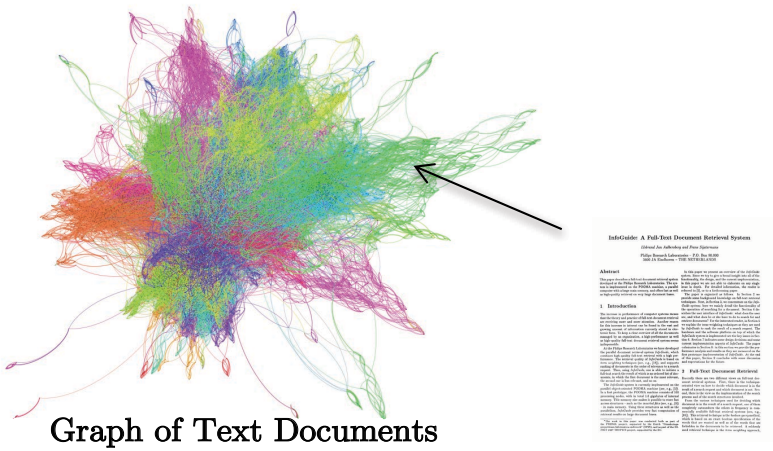
  (2) Large-scale data

  $\Rightarrow$ A class of algorithms exists and is called **manifold learning techniques** (later discussed).

- Some data have **clear structures** (some others less):



MNIST Image Graph

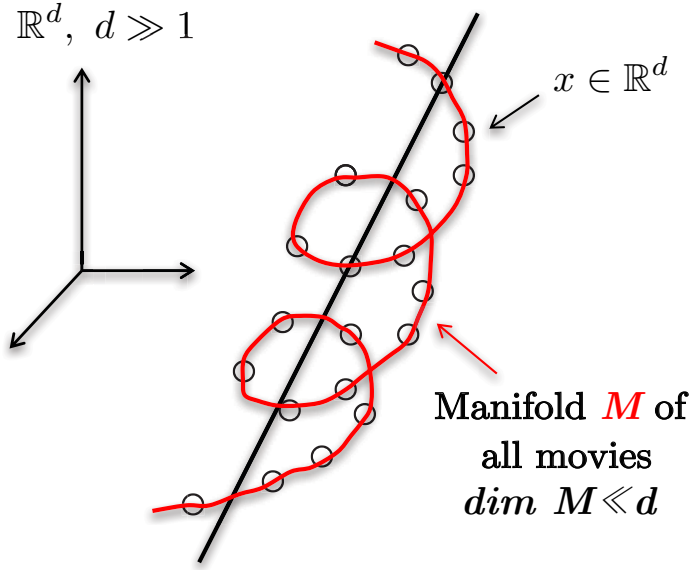


Graph of Text Documents 20newsgroups

From Manifolds to Graphs

- **Manifold assumption:** High-dim data are **sampled** from a low-dim manifold.

  Ex: 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. Then we can make the assumption that **all movies form a manifold** in R^{d}.



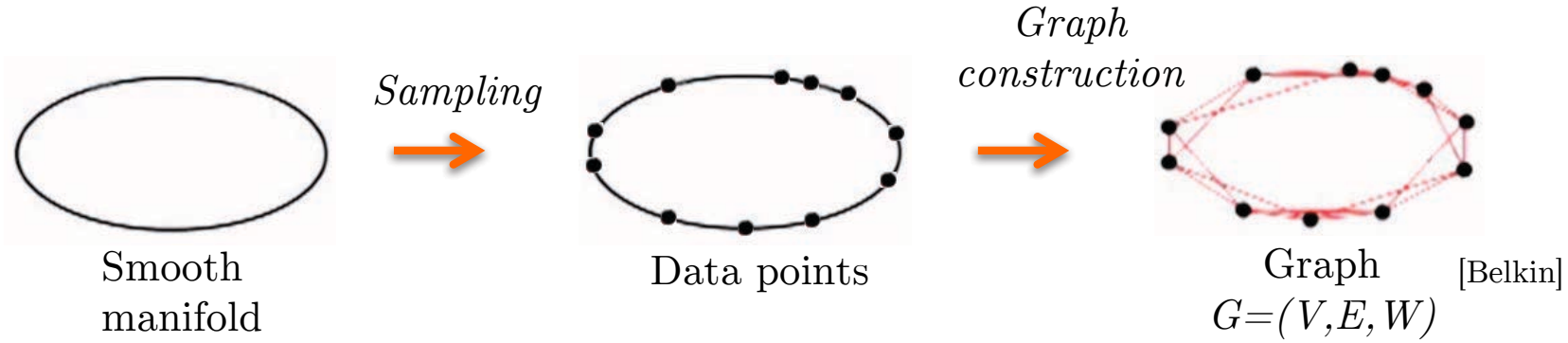
- **Assumption validity:** It is a *good working hypothesis* for:

  (1) Several types of data (images, text documents, music, etc)

  (2) Most data science tasks (classification, visualization, recommendation, etc)

From Manifolds to Graphs

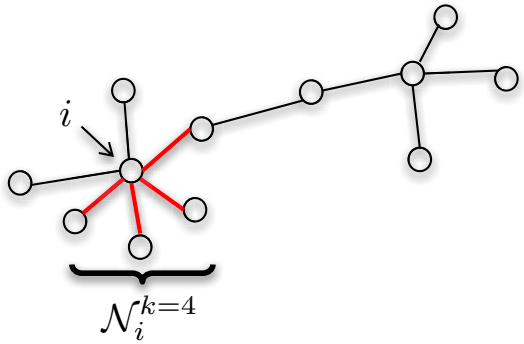
- **Graphs=Manifold sampling:** The manifold information is encoded by neighborhood graphs (we **never** form/see the manifold):



- **Neighborhood Graphs:**

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

where $dist(x_i,x_j)$ is the distance between x_i and x_j (to be defined), σ is the scale parameter (value depends on data), N_i^k is the neighborhood of data x_i:



Outline

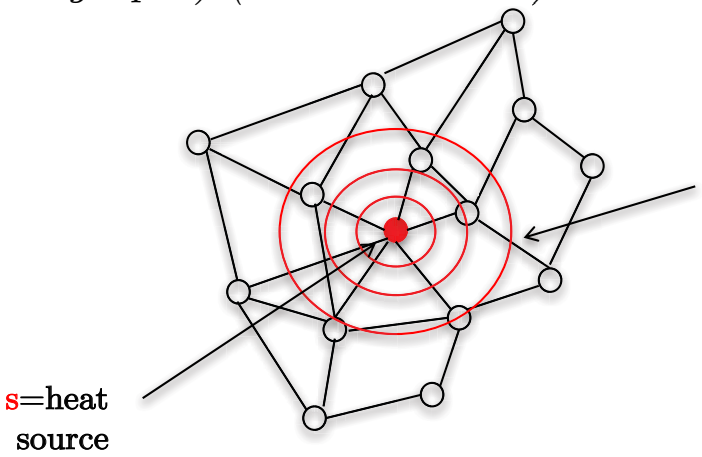
- Graph Science and Graph Theory
- Classes of Networks
- Basic Definitions
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- **Spectral Graph Theory**
- Construct Graphs from Data
- Conclusion

Graph Analysis=Spectral Graph Theory

Q: How to benefit from graphs?

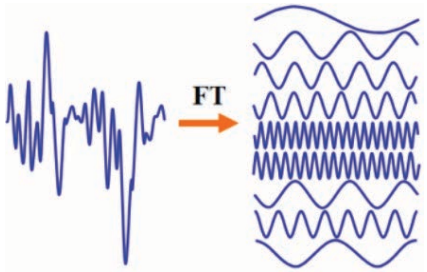
- **A:** Given a data graph $G=(V,E,W)$, use **spectral graph theory (SGT)** to:
  - *(1) Find meaningful patterns (multi-scale data structures)*
  - *(2) Analyze data on top of the network (Facebook users with messages, videos, etc)*
  - *(3) Design data science tasks (clustering, classification, recommendation, etc)*

- **Graph Laplacian Operator *L*:** The **most powerful tool** in SGT to analyze and process networks! What is L?
  - *(1) Heat diffusion operator (on graphs).*
  - *(2) Basis functions of this operator are the well-known **Fourier modes** (on graphs) (later discussed).*



Heat propagation with Laplacian:

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



Graph Laplacian Definitions

- **Unnormalized/combinatorial graph Laplacian:** (historical)

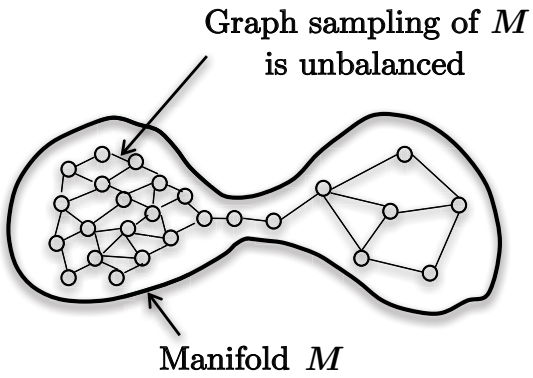
$$\underset{\substack{n \times n \\ n = |V|}}{L_{un}} = D - W \qquad \text{with } D \text{ is the degree matrix: } \quad D = \text{diag}(d_1, ..., d_n),$$

$$d_i = \sum_j W_{ij}$$

- **Normalized Laplacian:** (most popular)

$$L = D^{-1/2} L_{un} D^{-1/2} = I_n - D^{-1/2} W D^{-1/2}$$

$\Rightarrow$ Nice math properties s.a. robustness w.r.t. *unbalanced sampling*:



- **Random Walk Laplacian:** (for Google PageRank)

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

- **Note:** All Laplacian are *diffusion* operators, but different diffusion properties.

Graph Spectrum

- **Motivation:** Study the **modes of variation** of the graph system.
  Q: *How?* A: **Eigenvalue Decomposition (EVD) of Laplacian *L*:**

$$L = U \Lambda U^T \qquad U = [u_1, ..., u_n] \qquad \Lambda = \text{diag}(\lambda_1, ..., \lambda_n)$$

$$\langle u_k, u_{k'} \rangle = \begin{cases} 1 \text{ if } k = k' \\ 0 \text{ otherwise} \end{cases}$$

$$Lu_k = \lambda_k u_k$$

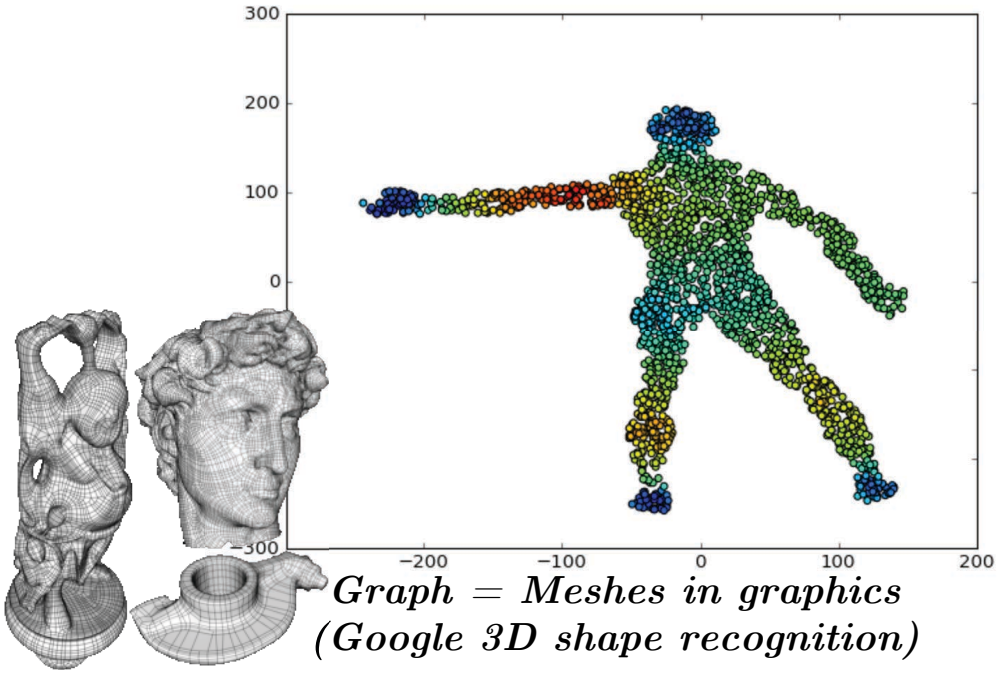
$$0 = \lambda_{\min} = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n = \lambda_{\max} \leq 2$$

- **Interpretation:**
  (1) u_k*: Fourier modes*, i.e. vibration modes of the graph.
  (2) λ_k*: Frequencies of the Fourier modes* u_k, i.e. how fast they vibrate.

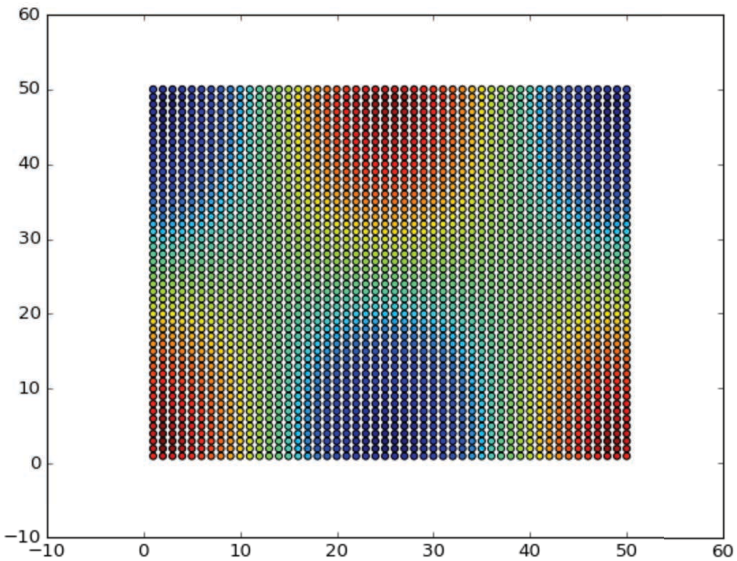
Demo: Modes of Variations of the Graph System

- Run **code02.ipynb**

```
In [5]: # Plot mode of variations
        for i in range(1,10):
            plt.figure(str(10+i))
            plt.scatter(X[:,0], X[:,1], c=U[:,i-1], s=size_vertex_plot*np.ones(X.shape[0]))
            plt.show()
```



Graph = Meshes in graphics (Google 3D shape recognition)



Graph = Regular grids (JPEG image compression, most used worldwide)

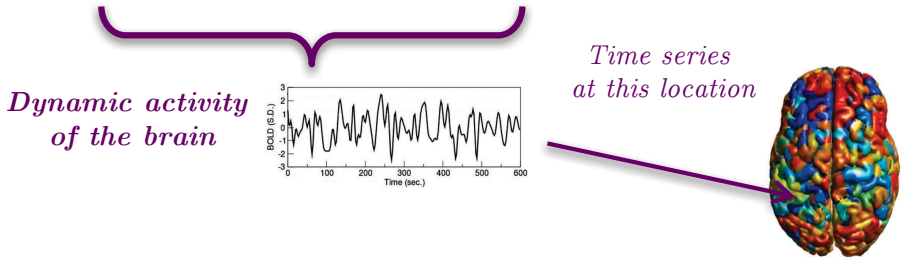
Q: What is the main property of the first and last of eigenvectors?

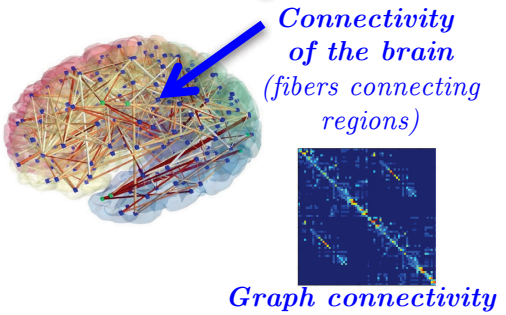
- **A:** *First* eigenvectors = *smoothest* modes of vibration of the graph

  Last eigenvectors = *highest* frequencies of the graph

Neuroscience

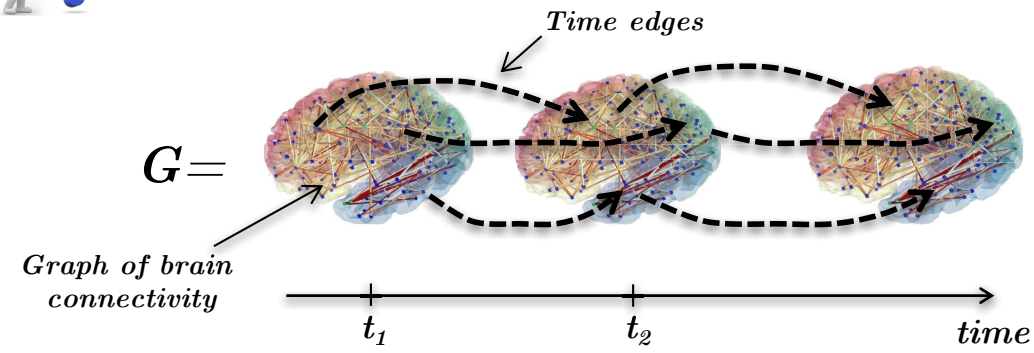
- **Goal:** Find meaningful **activation patterns** in brain using Structural MRI and Functional MRI.

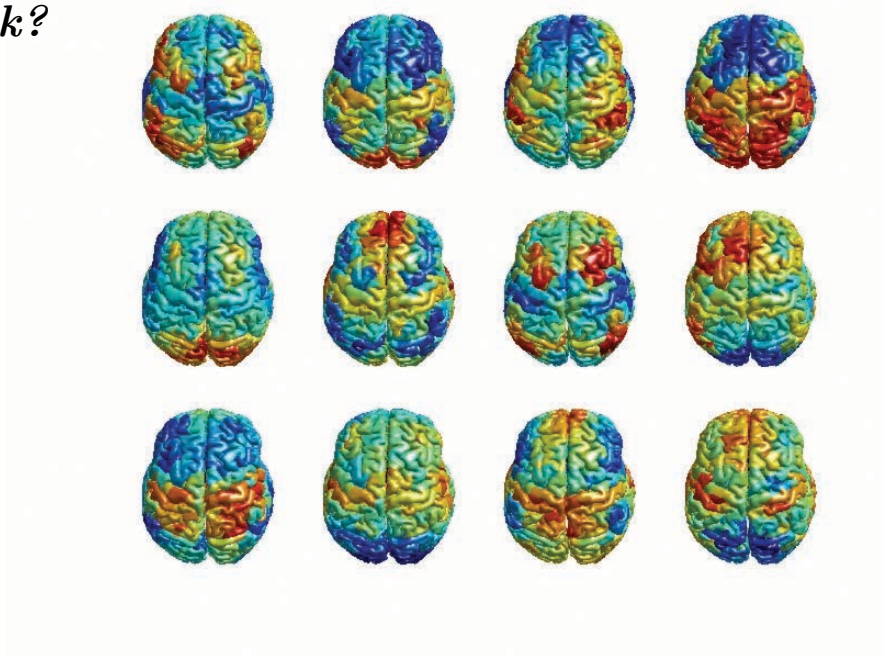




- **Methodology:** $G \Rightarrow L \Rightarrow u_k \Rightarrow$ dynamic activation patterns:

Q: How to construct the dynamic network?





- **Results:** (Re)discover *dynamic patterns* related to **basic functional tasks**: vision, body motor, language, etc

Outline

- Graph Science and Graph Theory
- Classes of Networks
- Basic Definitions
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- Spectral Graph Theory
- **Construct Graphs from Data**
- Conclusion

How to Construct Graphs from Data?

- **Three fundamental questions:**
  (1) What type of graphs?
  (2) What distances between data?
  (3) What data features?

- **Answers:** Optimal graph construction is an **open problem** – depends on data and analysis tasks. However, they exist **good practices and domain expertise** are useful.

What Type of Graphs?

- **Neighborhood graphs:** k-NN graphs (there exist *ε-graphs* but they are dense)

  Parameters: (1) $\boldsymbol{k}$ = #nearest neighbors

  (2) $\boldsymbol{\sigma}$ = scale parameter

- **k-value:** *10-50*

- **σ-value:** Two strategies:

  (1) Global scale: σ = mean distance of *all* k$^{\text{th}}$ neighbors

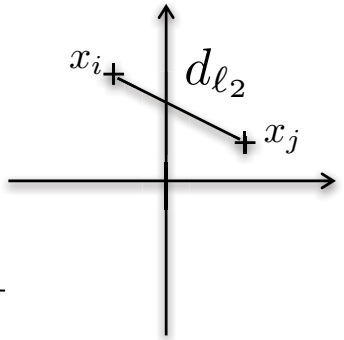
$$W_{ij} = \begin{cases} e^{-\frac{\text{dist}(x_i,x_j)^2}{\sigma^2}} & \text{if } j \in \mathcal{N}_i^k \\ 0 \text{ otherwise} \end{cases}$$

  (2) Local scale [Zelnik-Perona'04]: σ_i = distance of the k$^{\text{th}}$ neighbor of vertex i.

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

What Distances?

Q: What Distances do you know?

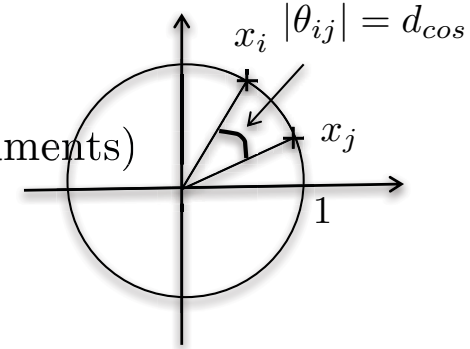


- **(1) Euclidean distance:**

Good for low-dim data $d<10$

Good for high-dim data with *clear structures* (MNIST)

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

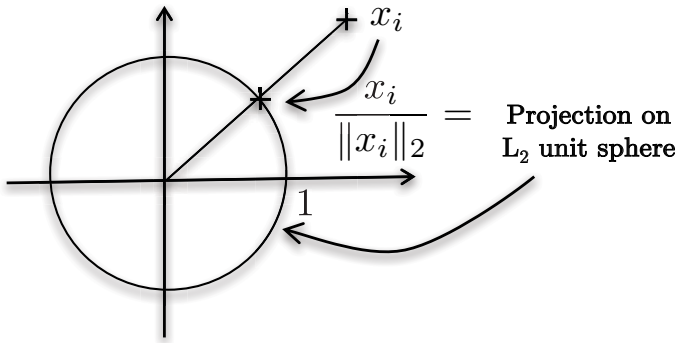


- **(2) Cosine distance:**

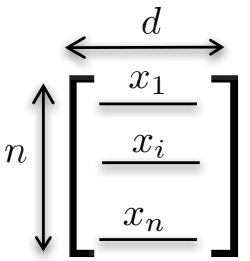
Good for high-dim and sparse data (text documents)

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

Note:



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



- **(3) Other distances:** Kullback-Leibler (information theory), Wasserstein (earth's mover) (PDEs theory), etc.

What Data Features?

- **Three types of data features:** $x_i \in \mathbb{R}^d$

  (1) Natural features (e.g. movie features s.a. genre, actors, year, etc)

  (2) Hand-crafted features (e.g. SIFT in computer vision)

  (3) Learned features (PCA, NMF, sparse coding, deep learning)

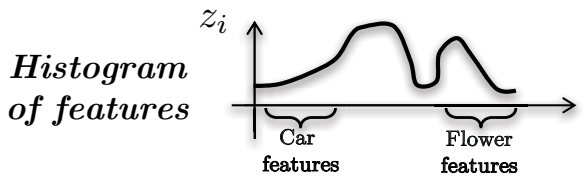
- **Bad approach:** It is usually a bad idea to use directly **raw** data as features.

- **Good approach:** *Transform raw data* into meaningful data representation by *extracting features* (later discussed) and use them for graph construction (trick known as *bag of words*):

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

Raw data
Ex: image

Features
Ex: car features
flower features



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

$$\Downarrow$$

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

Data Pre-Processing

- **Center data** (along each dimension): zero-mean property (very common)

$$x_i \leftarrow x_i - mean(\{x_i\})$$

- **Normalize data variance** (along each dimension): z-scoring property (w/ zero-mean)

$$x_i \leftarrow x_i / std(\{x_i\})$$

$$std(\{x_i\}) = \sqrt{\sum_j |x_j - mean(\{x_i\}|^2}$$

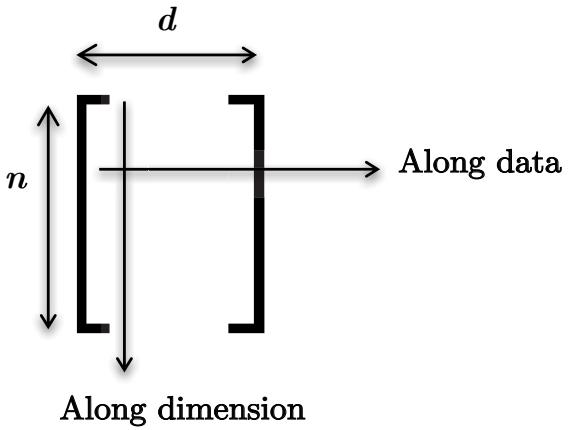
- **Projection on l2-sphere** (along each data):

$$x_i \leftarrow x_i / \|x_i\|_2$$

- **Normalize max and min value:**

$$x_i \in [0, 1] \leftarrow x_i$$

$$X = \underset{n \text{ x } d}{}$$



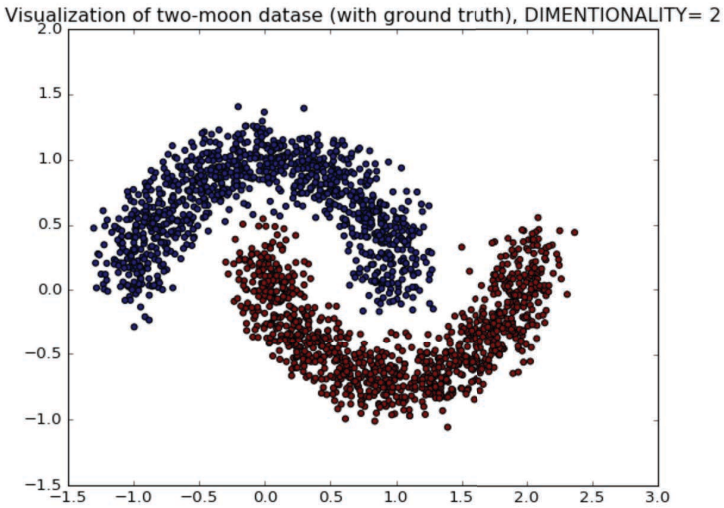
Demo: Graph Construction and Pre-Processing

- Run **code03.ipynb**

  Let us test: *Pre-processing, Construct k-NN graphs, Visualize distances, Visualize W, Test graph quality with clustering accuracy.*

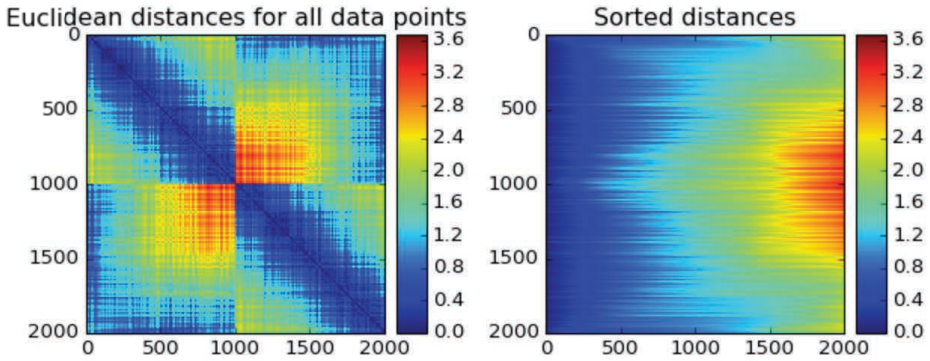
```
# Visualize in 2D
plt.figure(30)
size_vertex_plot = 20.
plt.scatter(X[:,0], X[:,1], s=size_vertex_plot*np.ones(n), c=C)
plt.title('Visualization of two-moon datase (with ground truth), DIMENTIONALITY= ' + str(dim))
plt.show()
```

```
(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)
```

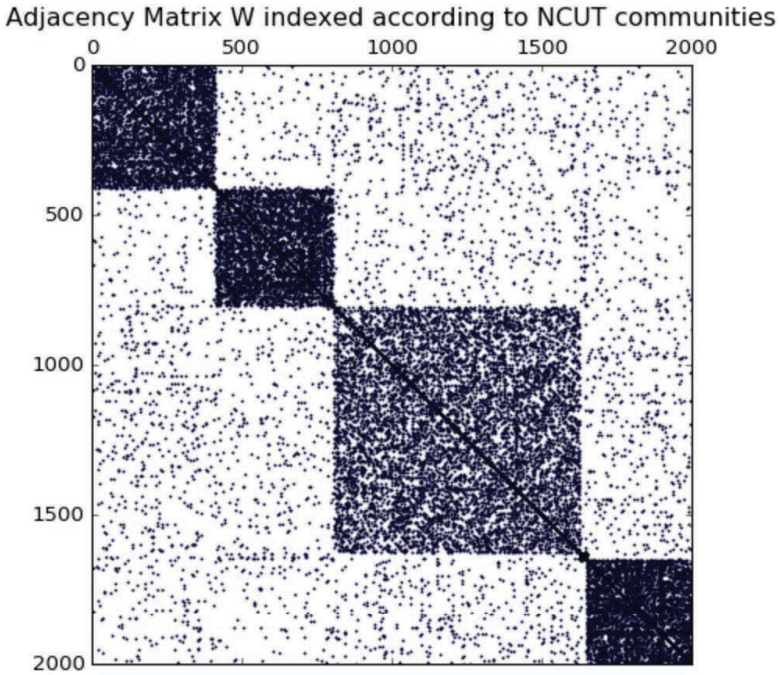


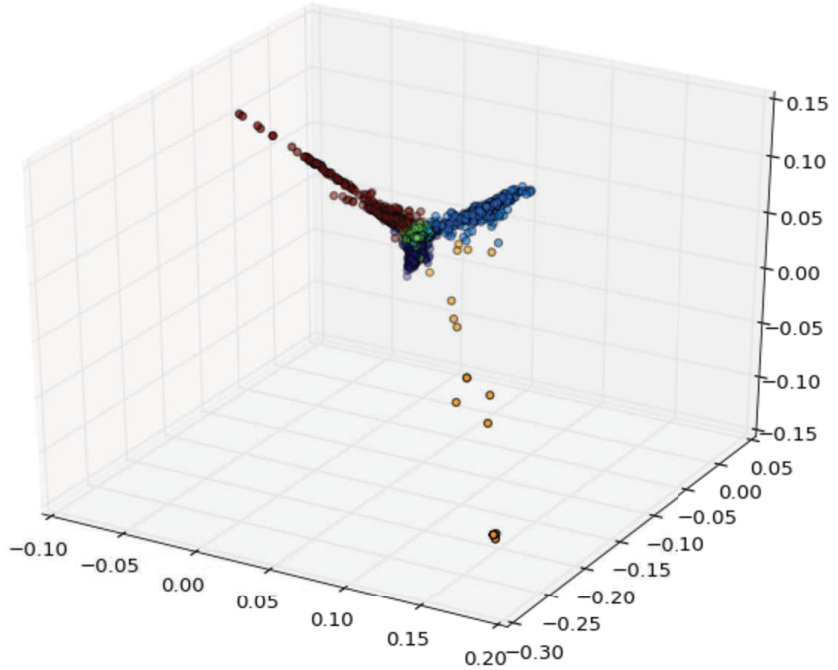
Demo: Construct Network of Text Documents

- Run **code04.ipynb**

```
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 Science and Graph Theory
- Classes of Networks
- Basic Definitions
- Curse of Dimensionality and Structure
- Manifolds and Graphs
- Spectral Graph Theory
- Construct Graphs from Data
- **Conclusion**

Summary

- **Graph is a superior representation of data:**

$$\text{Data} \Rightarrow \text{Graph } G=(V,E,W)$$

- **1st fundamental tool: Adjacency Matrix W**

  (1) It reveals structures hidden on data.

  (2) It allows to visualize graphs.

  (3) It is used for analysis tasks (later discussed).

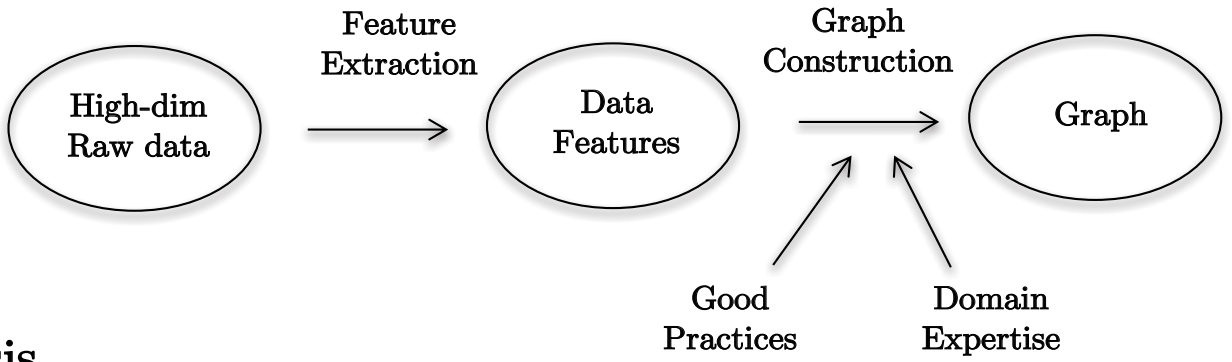
- **2nd fundamental tool: graph Laplacian Matrix L**

  (1) It represents the modes of variations of the graph.

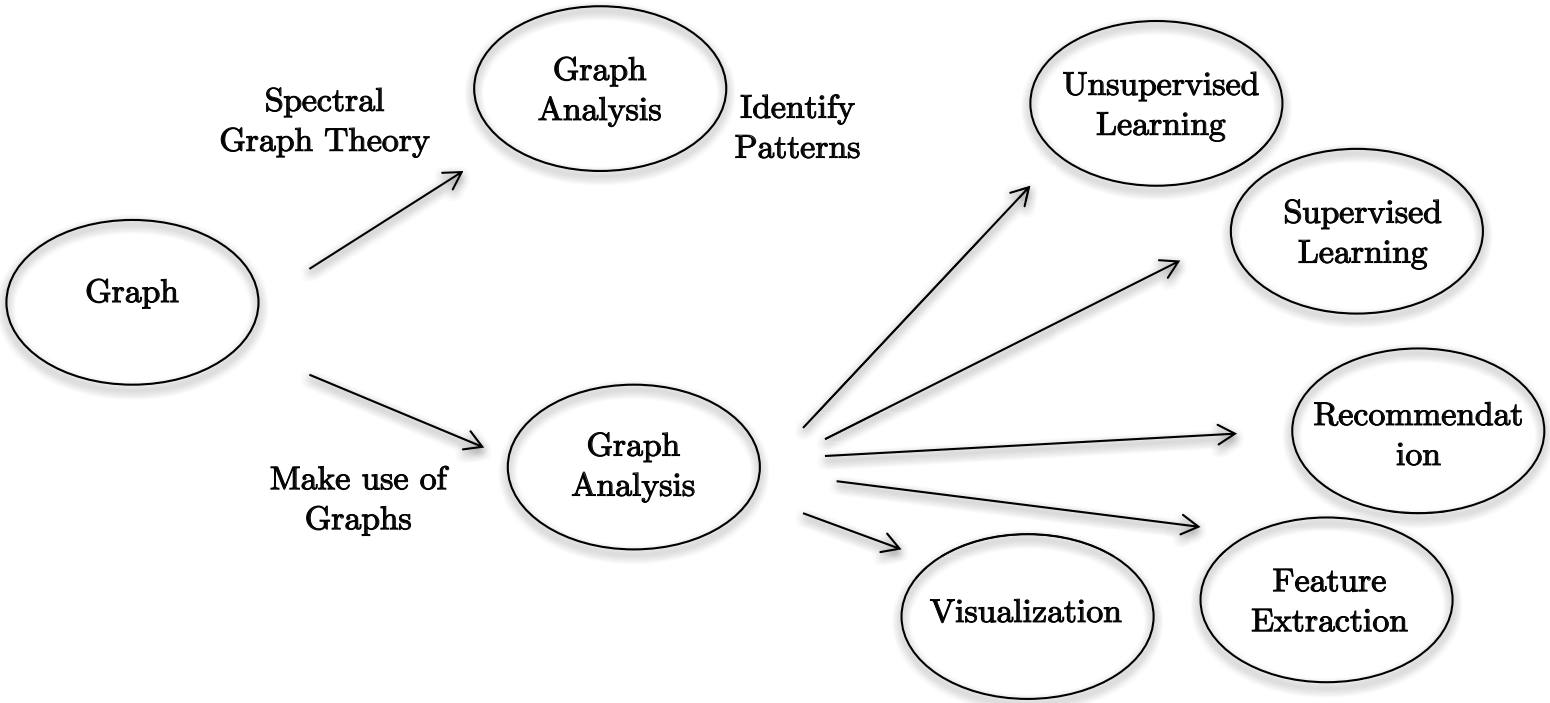
  (2) Used for image compression (jpeg), neuroscience, etc.

Pipeline of Graph Science

- Step 1: Data ⇒ Graph



- Step 2: Graph ⇒ Analysis



Questions?