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)

Outline

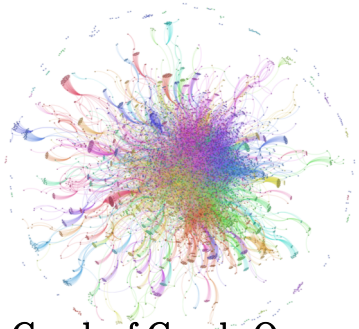
- Graph theory
- Classes of graphs
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Outline

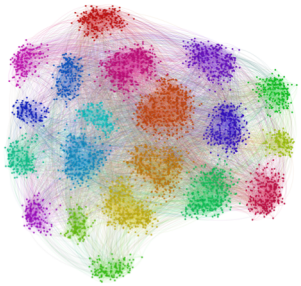
- Graph theory
- Classes of graphs
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Graphs

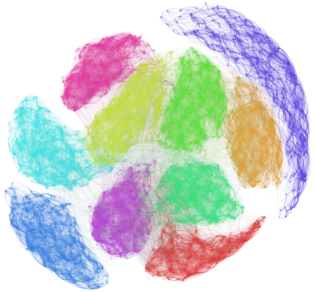
- Graphs encode complex data structures. They are everywhere: WWW, Facebook, Amazon, etc



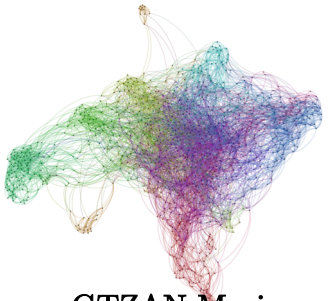
Graph of Google Query "California"



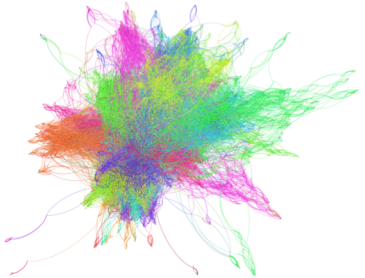
Social Network



MNIST Image Network



GTZAN Music Network

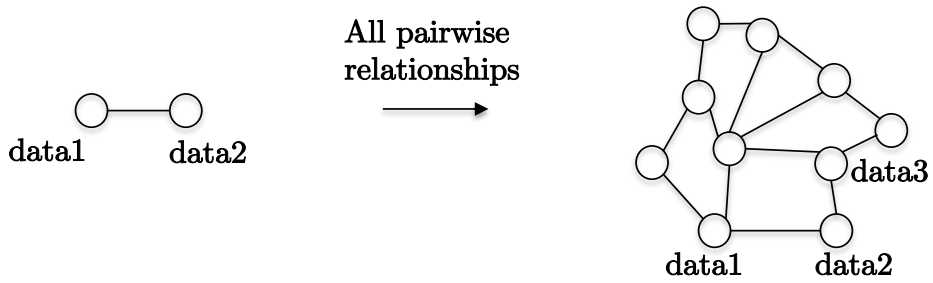


Network of Text Documents 20newsgroups

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

Graphs

- Definition of graphs : Mathematical model representing pairwise relations between objects/data

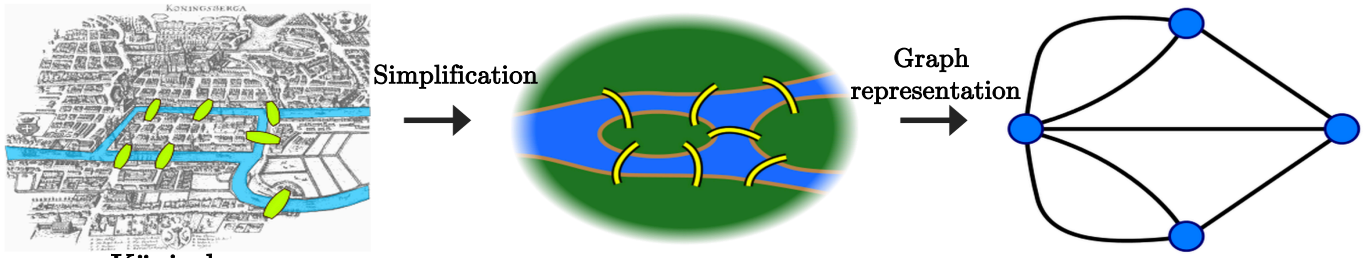


- Why are they useful?
  - Graphs offers a global view of data structure
    - Extract global meaningful patterns, insights about data.
    - Some tasks are purely designed on graphs, e.g. Google PageRank recommendation.
    - Increase task performances thanks to prior information/inductive bias.
    - Benefit (but not optimized) from GPUs (graphs are sparse matrices).

Graph Theory

- 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 (starting from any place) that crosses each bridge once and only once? Euler proved it is not possible (it is only possible if the graph has even degree that allows cycles).



Königsberg
City

Source: Wikipedia.

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

Outline

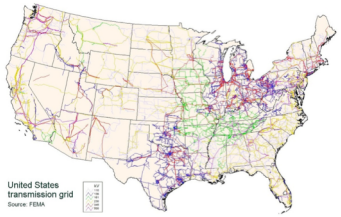
- Graph theory
- Classes of graphs
- Basic definitions
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Classes of graphs

- Natural Graphs
  - Social networks: Facebook, LinkedIn, Twitter
  - Biological networks: Brain connectivity and functionality, gene regulatory networks
  - Communication networks: Internet, networking devices
  - Transportation networks: Trains, cars, airplanes, pedestrians
  - Power networks: Electricity, water

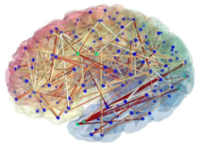
Minnesota Road Network

Telecommunication Network



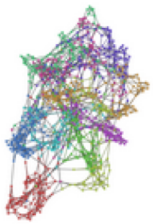
US Electrical Network

Facebook



Brain Connectivity

=

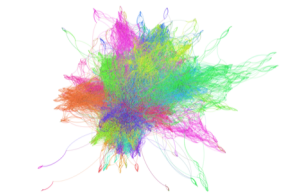


Graphs/networks

- Terminology : Natural graph means graph was not hand-crafted/constructed.

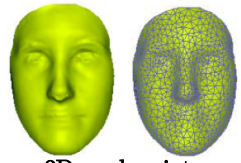
Classes of graphs

- Constructed Graphs (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

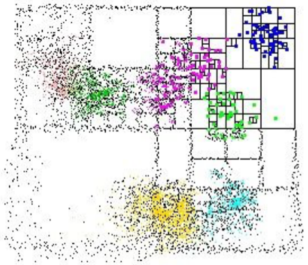

3D mesh points

- Graph Construction : No universal recipe but domain expertise knowledge and common practice.
- How much time do we need to construct a graph?
  - Computational time: Exact (/brute force) is $O(n^2d)$, n=#data, d=#features.

    for d=1K n=1K ⇒ time < 1sec

    n=100K ⇒ time = 1 min

    n=1M ⇒ time > 1 hour

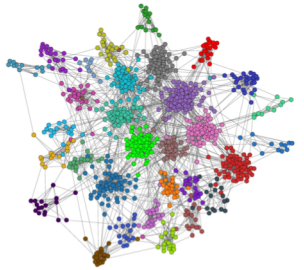


  - Approximate technique: FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces, https://github.com/flann-lib/flann

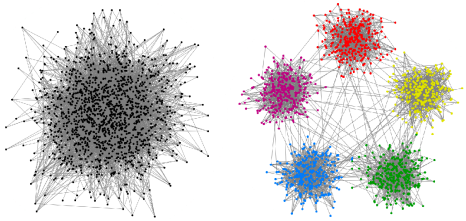
Classes of graphs

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

Erdos-Renyi Network
Source: Wikipedia.



LFR
Source: Kojaku, 2018



SBM
Source: Abbe, JMLR'17

- Why using artificial networks?
  - Mathematical Modeling
    - Advantages : Precise control of your data model (estimate best performances given data assumptions). No need to perform extensive experiments! (limitation with deep learning)
    - Limitations : Most data assumptions are too restrictive, no guarantee that real-world data follow (closely) the model assumptions.

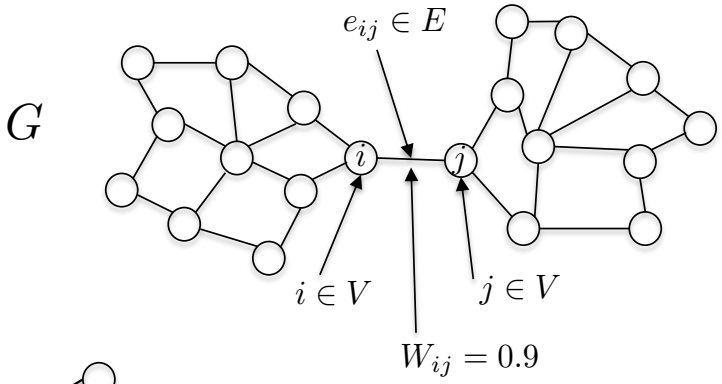
[1] Erdos, Renyi, On random graph, 1959
[2] Anderson, Wasserman, Faust, Building stochastic blockmodels, 1992
[3] Lancichinetti, Fortunato, Filippo, Benchmark graphs for testing community detection algorithms, 2008

Outline

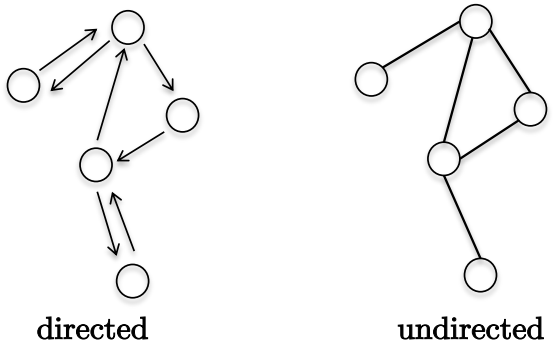
- Graph theory
- Classes of graphs
- **Basic definitions**
- Curse of dimensionality and structure
- Manifolds and graphs
- Spectral graph theory
- Graph construction
- Conclusion

Basic definitions[1]

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



- Directed/undirected graphs:



- Note: We will work mostly with undirected networks.

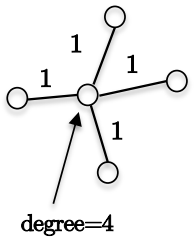
[1] Chung, Spectral graph theory, 1997

Basic definitions

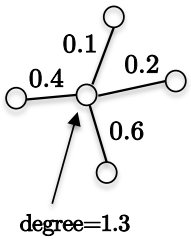
- Vertex degree

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

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



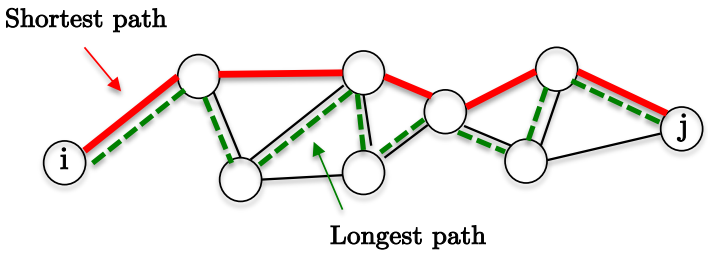
Binary graph

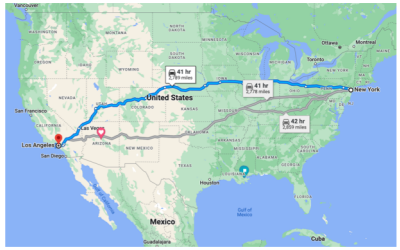


Weighted graph

- Shortest path: A path on a graph with the smallest possible distance.
  - Fast Algorithm: Dijkstra's algorithm[1]
  - Do you know a popular application?

    Road navigator, e.g. New York to Los Angeles.





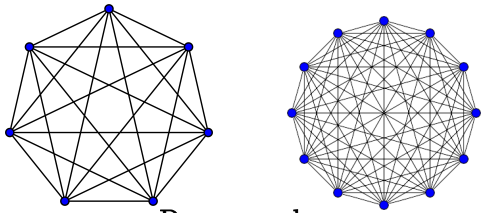
- Classical graph algorithms (no feature learning) : Breadth-first search, depth-first search, minimum spanning tree, topological sorting, strongly connected components, graph colouring, maximum flow/minimum cut, graph matching, etc.

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

Dense vs. sparse graphs

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

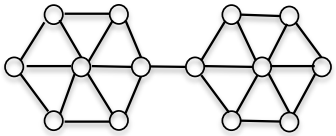
$$|E| = \frac{n(n-1)}{2} = O(n^2)$$ 



Dense graphs

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

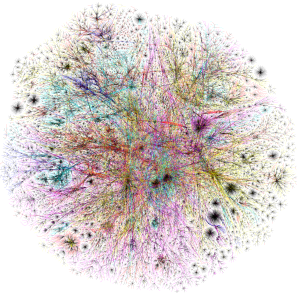
$$|E| = O(kn) = O(n)$$ 



Sparse graph

- What is better; dense or sparse?
  - 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/dense.
    - $|E| = k.n = 10^{11}$ as it is (very) sparse.



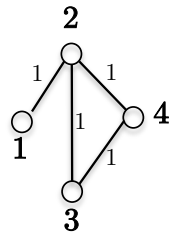
- Good news: Most natural/real-world networks (e.g. Facebook, brain, communication) are sparse.
  - Sparsity ⇔ structure/inductive bias

Adjacency/similarity matrix

- **Definition: Matrix W in G=(V,E,W) actually contains all structural/topological information about the network. There are two choices for W:**
  (1) Binary W: W_{ij} in {0,1}
  (2) Weighted W: W_{ij} in [0,1] (commonly normalized to 1)

(1) Binary W :

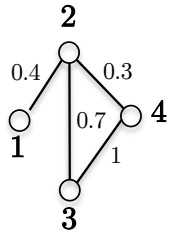
$$W_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

$G=$ 

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

(2) Weighted W :

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

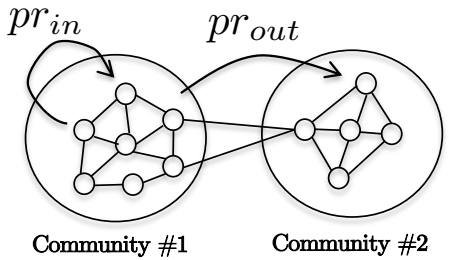
$G=$ 

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

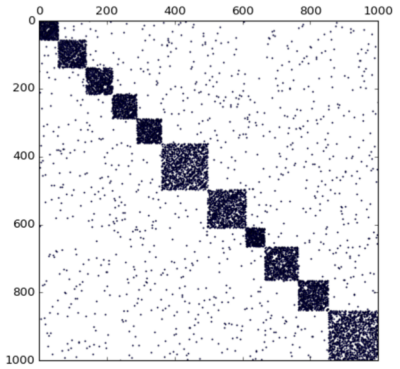
Lab 1: Generate artificial LFR social networks

- Run code01.ipynb
- Synthesize LFR social networks : Play with the mixing parameter μ
  - μ small : communities well separated.
  - μ large : communities are mixed up.

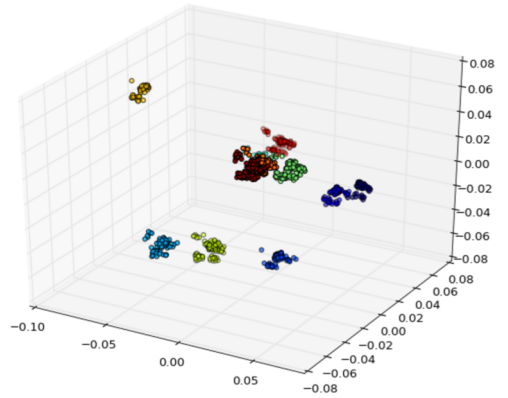
$$\mu = \frac{pr_{out}}{pr_{in}}$$




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



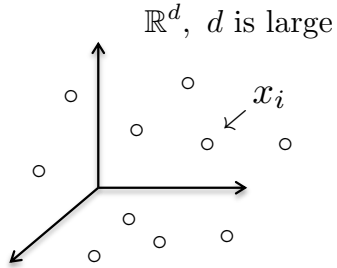

```
In [19]: # Visualize the social network in 3D
         fig = pylab.figure(4)
         ax = Axes3D(fig)
         ax.scatter(X, Y, Z, c=C)
         pyplot.show()
```



Outline

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

Curse of dimensionality



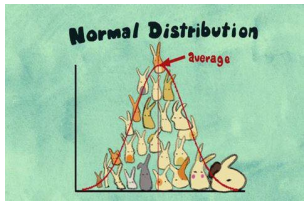
- What is the curse of dimensionality?
  - In high dimensions, (Euclidean) distance between data is meaningless.
  - Theorem[1] : Suppose data are uniformly distributed in R^d,

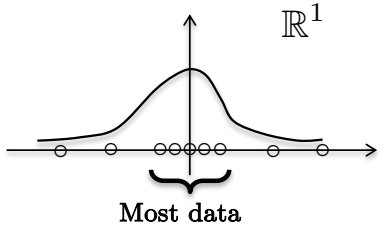
    pick any data x_i then we have :

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

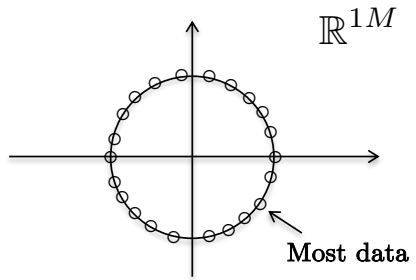
    All data are close/far away to each other!

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





1-D Gaussian

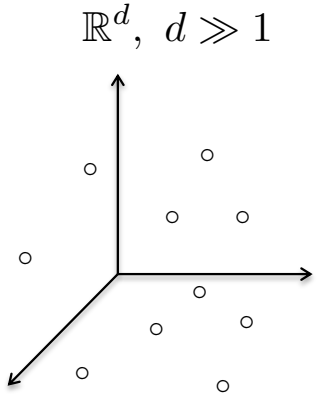


1,000,000-D Gaussian

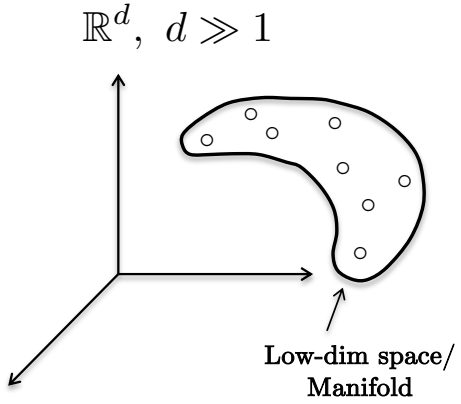
[1] Beyer, Goldstein, Ramakrishnan, Shaft, When is "nearest neighbor" meaningful? 1999

Blessing of structure

- What is the blessing of structure?
  - Assumption "data are uniformly" distributed is not true for real-world data. Data have always properties/structures such that they belong to a low-dimensional space called manifold where (some) distances are meaningful.



Uniform distribution of data
No structure/randomness



Non-Uniform distribution of data
Structure/inductive bias

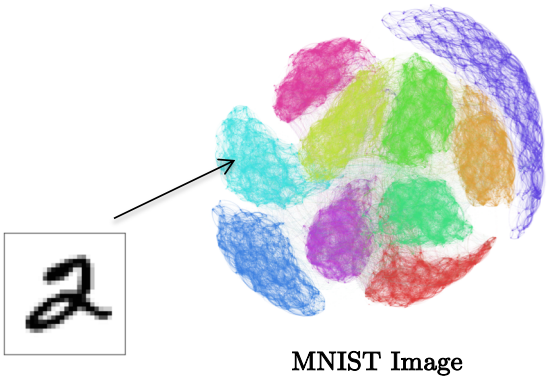
Outline

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

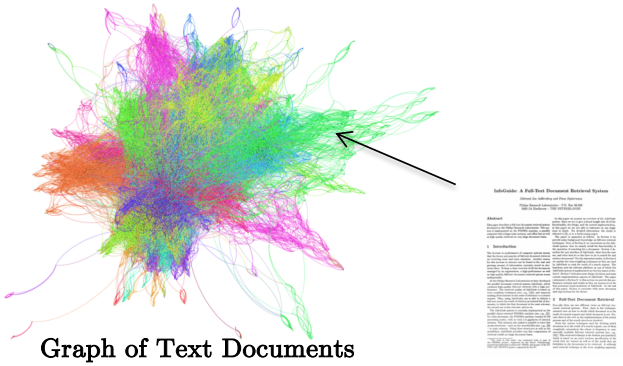
Manifold learning

- It is challenging to identify structures hidden in data because of
  (1) High-dimensional data
  (2) Large-scale dataset

- Some data have clear structures (some less):



MNIST Image
Graph



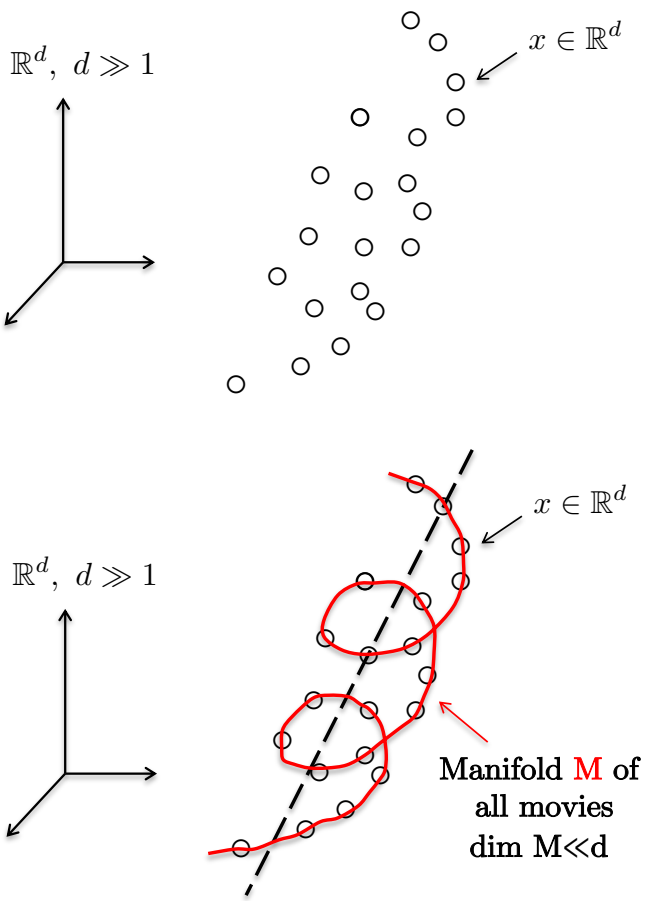
Graph of Text Documents
20newsgroups

- A class of algorithms that studies this question is manifold learning (later discussed).

From manifolds to graphs

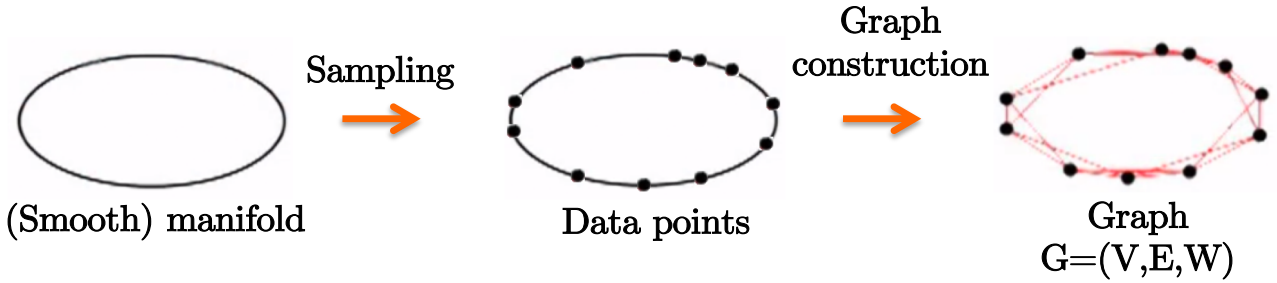
- Manifold assumption : High-dim data are sampled from a low-dimensional manifold.
  - Example : Let x be a movie, each movie is defined by d features/attributes like genre, actors, release year, origin country, etc such that x in R^d. Then we can make the assumption that all movies form a manifold M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

- Assumption validity : It is good working hypothesis for
  - Several types of data (images, text documents, music, etc)
  - Most data science tasks (classification, visualization, recommendation, etc)



From manifolds to graphs

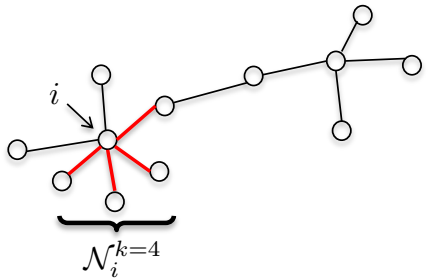
- Graphs can be seen as manifold sampling : The manifold information is encoded by neighborhood graphs (we never see/construct explicitly the manifold).



- 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 :

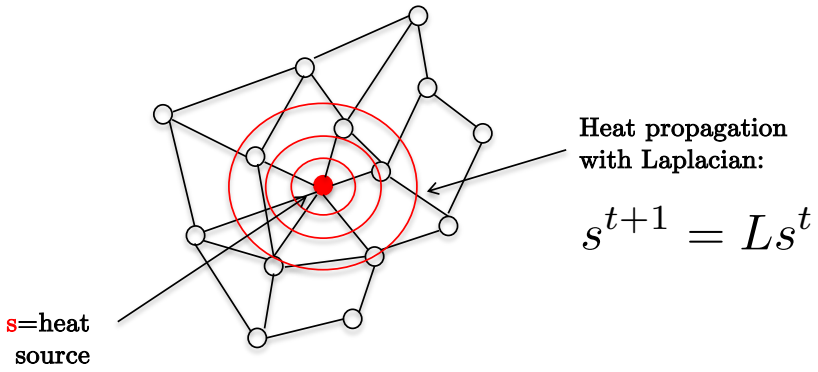


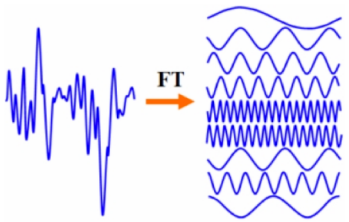
Outline

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

Spectral graph theory

- How to benefit from graphs?
  - Given a data graph G=(V,E,W), use spectral graph theory (SGT) to
    - Find meaningful patterns (multi-scale data structures)
    - Analyze data on the network (Facebook users with messages, videos, etc)
    - Design data analysis tasks (clustering, classification, recommendation, etc)
- Graph Laplacian Operator L : The most powerful tool in SGT to analyze and process networks!
  - Why is L useful?
    - Heat diffusion operator (on graphs).
    - Basis functions of this operator are the well-known Fourier modes (later discussed).





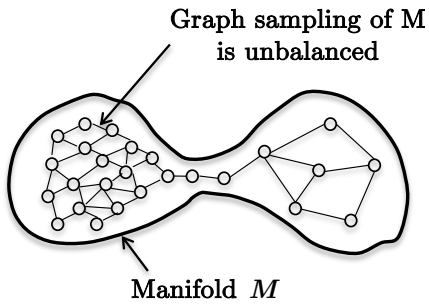
Graph Laplacian definitions

- Unnormalized/combinatorial graph Laplacian (historical):

$$\underset{\substack{n \times n \\ n = |V|}}{L} = D - W \qquad \text{with } \boldsymbol{D} \text{ is the degree matrix : } D = \text{diag}(d_1, ..., d_n),$$
$$d_i = \sum_j W_{ij}$$

- Normalized Laplacian (most popular):
  - Nice math properties s.a. robustness w.r.t. unbalanced sampling:

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



- Random Walk Laplacian (Google PageRank):

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

- All Laplacians are diffusion operators, but with different diffusion properties.

Graph spectrum

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

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

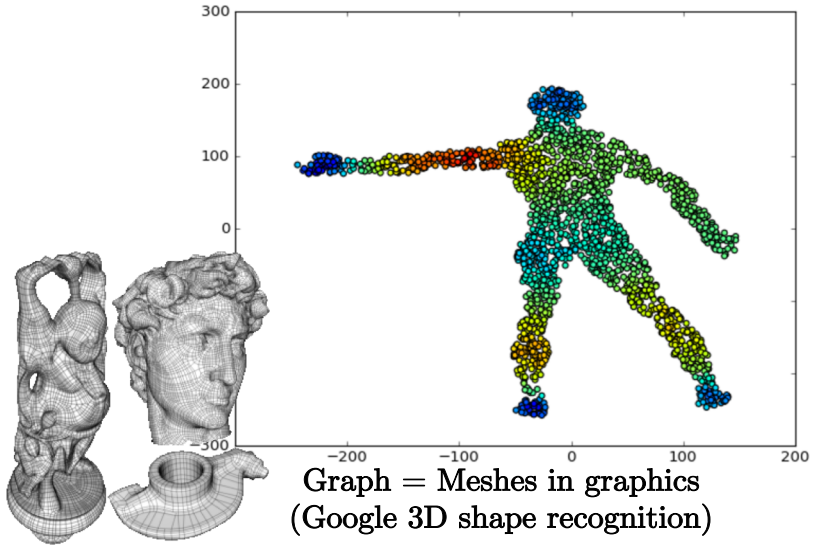
$$L u_k = \lambda_k u_k \qquad \langle u_k, u_{k'} \rangle = \begin{cases} 1 \text{ if } k = k' \\ 0 \text{ otherwise} \end{cases}$$

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

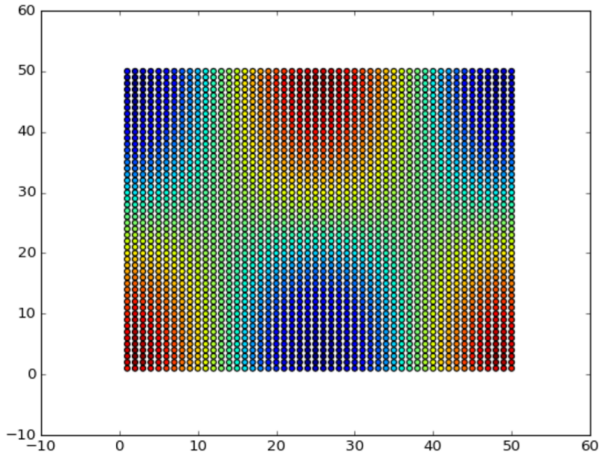
- Interpretation:
  - u_k : Fourier modes, i.e. vibration modes of the graph.
  - λ_k : Frequencies of the Fourier modes u_k, i.e. how fast they vibrate.

Lab 2: Modes of variations of a graph system

- Run code02.ipynp
- What is the main property of the first and last of eigenvectors?
  - First eigenvectors = Smoothest modes of vibration of the graph / low-frequency information
  - Last eigenvectors = Highest frequencies of the graph / noise or meaningful details



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



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

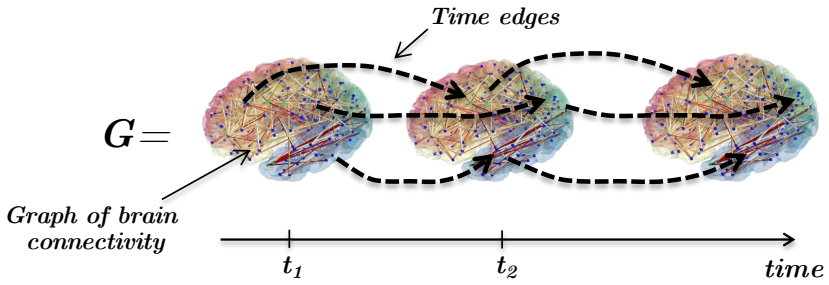
Neuroscience

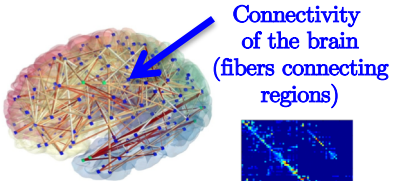
- Goal : Find meaningful brain activation patterns using structural MRI and functional MRI.

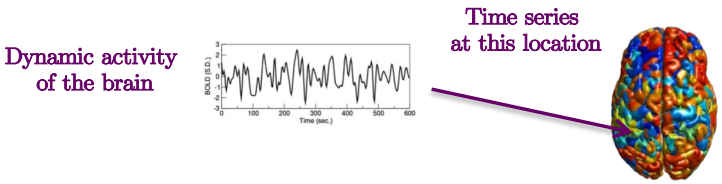
- Methodology: G ⇒ L ⇒ u_k ⇒ dynamic activation patterns!

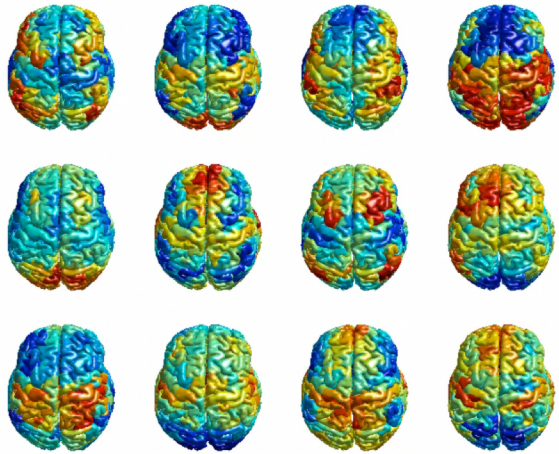
  Results : (Re)discover dynamic patterns related to basic functional tasks: vision, body motor, language, etc..

- How to construct the dynamic network?









Outline

- Graph theory
- Classes of graphs
- 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 :
  (1) What type of graphs?
  (2) What distance between data?
  (3) What data features?

- Answer : Optimal graph construction is an open problem – it depends on data and analysis task.
  - However, domain expertise and good practice are useful.

What type of graphs?

- Neighborhood graphs : k-NN graphs (there exist ε-graphs but they are dense, not practical)

  Parameters: (1) k = #nearest neighbors

  (2) σ = scale parameter

- Typical k-value is 10-50.

- σ-value: Two strategies:

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

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

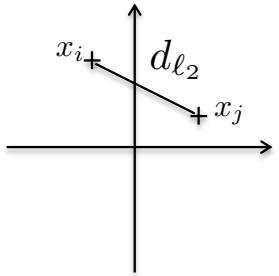
  (2) Local scale[1] : σ_i = distance of the k^{th} neighbor of vertex i.

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

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

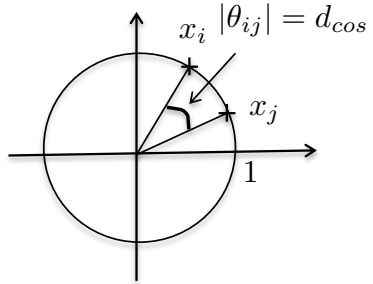
What distances?

- Euclidean distance
  - Good for low-dim data d<10
  - Good for high-dim data with clear structure (e.g. MNIST)



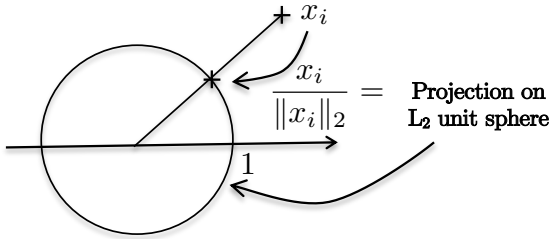
$$d_{\ell_2}(x_i, x_j) = \|x_i - x_j\|_2 = \sqrt{\sum_{m=1}^{d} |x_{i,m} - x_{j,m}|^2}$$

- Cosine distance
  - Good for high-dim and sparse data (e.g. text documents)

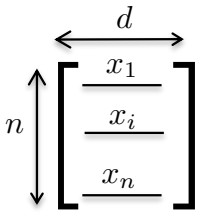


$$d_{cos}(x_i, x_j) = \left| \cos^{-1}\left(\frac{\langle x_i, x_j \rangle}{\|x_i\|_2 \|x_j\|_2} \right) \right| = |\theta_{ij}|$$

Note:



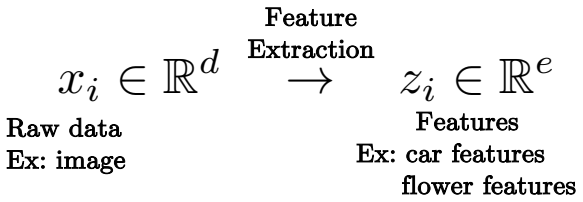
$$\underset{n \times d}{X} = \begin{bmatrix} x_1 \\ x_i \\ x_n \end{bmatrix} \quad \|x_i\|_2 = 1$$



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

What data features?

- Three types of data features
  - Natural features (e.g. movie features such as genre, actors, year, etc)
  - Hand-crafted features (e.g. SIFT in computer vision)
  - Learned features (PCA, NMF, sparse coding, deep learning)
- It is generally bad to directly use the raw data as features for graph construction.
- It is better to transform raw data into meaningful data representation by extracting features (handcrafted bag of words or learned features) and use them for graph construction.



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

$$\Downarrow$$

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

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)

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

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

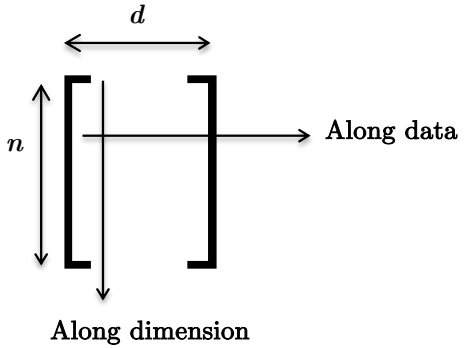
- Projection on L2-sphere (along each data):

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

- Normalize max and min value:

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

$X =$

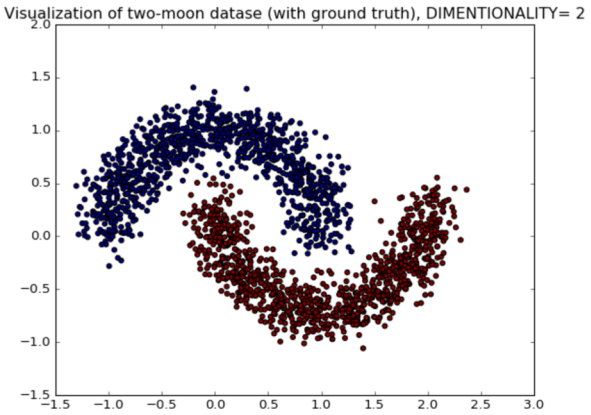


Lab 3: Graph construction with pre-processing

- Run code03.ipynb
- Study pre-processing, construct k-NN graphs, visualize distances, visualize W, test graph quality with clustering accuracy.

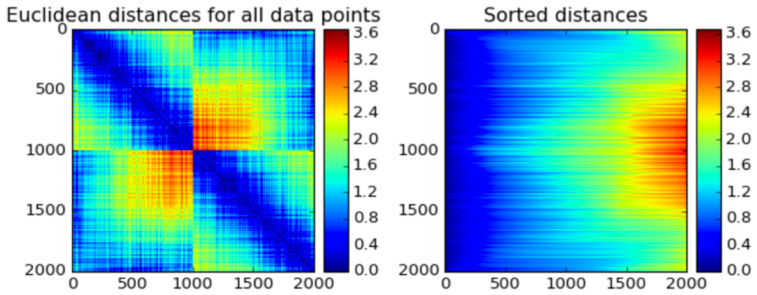
```
# Visualize in 2D
plt.figure(30)
size_vertex_plot = 20.
plt.scatter(X[:,0], X[:,1], s=size_vertex_plot*np.ones(n), c=C)
plt.title('Visualization of two-moon datase (with ground truth), DIMENTIONALITY= ' + str(dim))
plt.show()
```

```
(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)
```

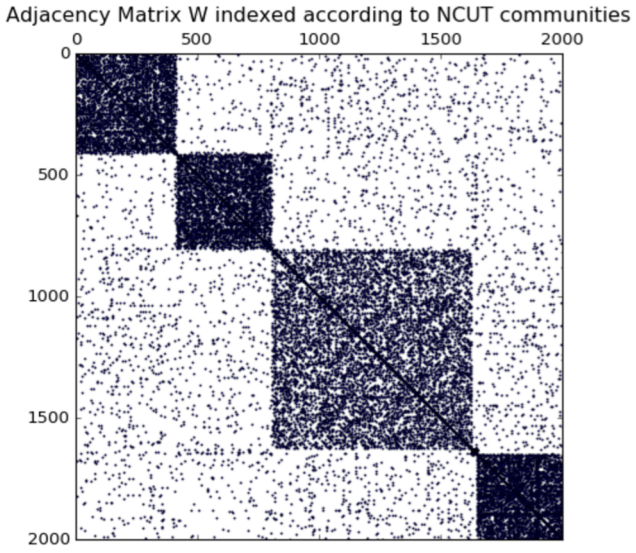


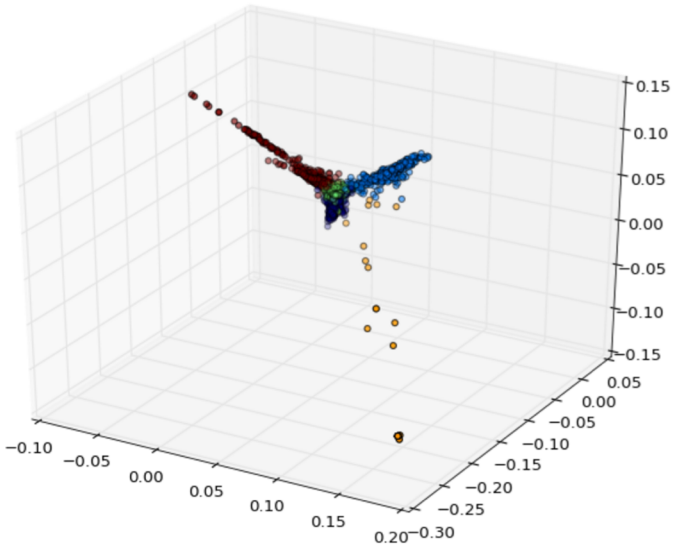
Lab 4: Construct a network of text documents

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

Summary

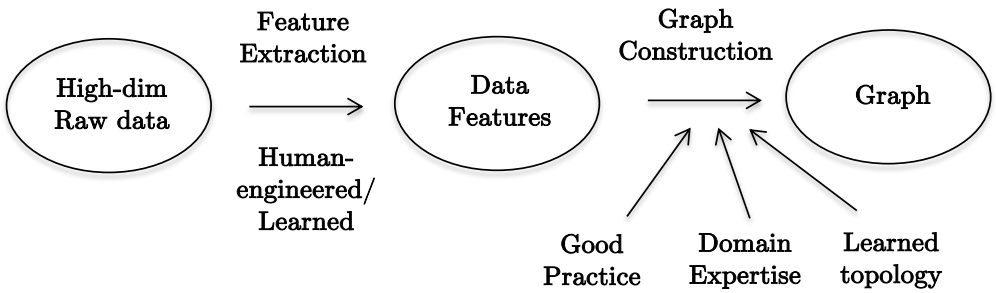
- Graph is an augmented representation of data with their relationship or connectivity.
  - Data ⇒ Graph G=(V,E,W)

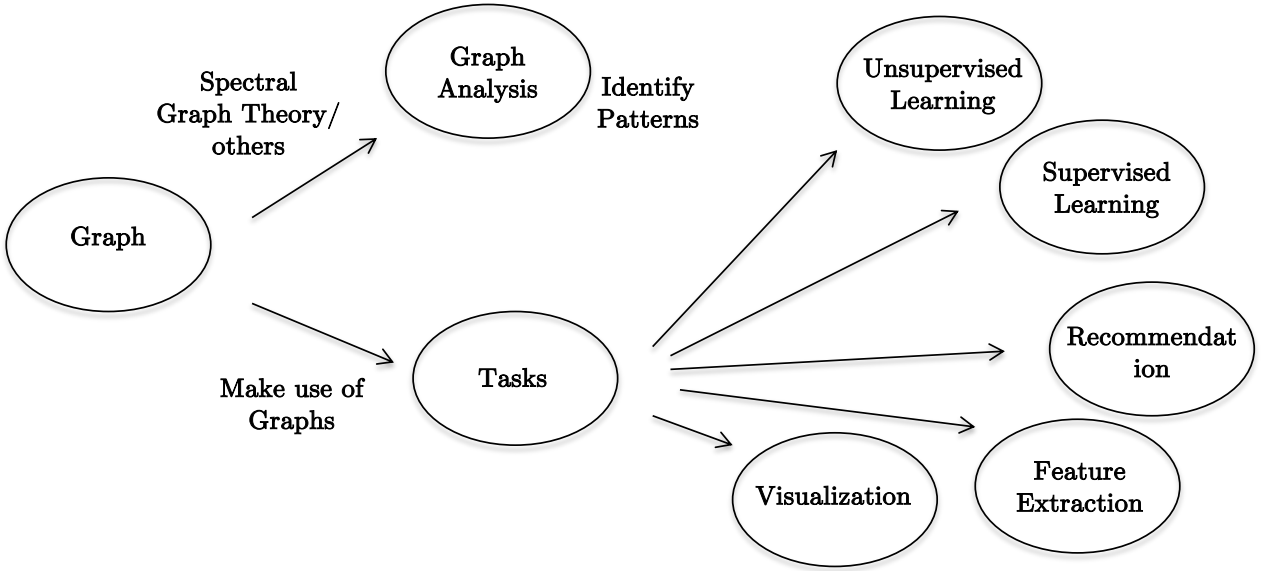
- 1st fundamental tool : Adjacency matrix W
  - It reveals structures hidden in data.
  - It allows to visualize graphs.
  - It is used for analysis tasks (later discussed).

- 2nd fundamental tool : Graph Laplacian matrix L
  - It represents the modes of variations of the graph.
  - Used for image compression (jpeg), neuroscience, positional encoding, etc.

Standard pipeline for graph science

- Step 1: Data ⇒ Graph
- Step 2: Graph ⇒ Analysis





Questions?