Lecture 2: Introduction to Low Rank Approximation, Dimension Reduction, and Clustering

Haesun Park

School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta GA, U.S.A.

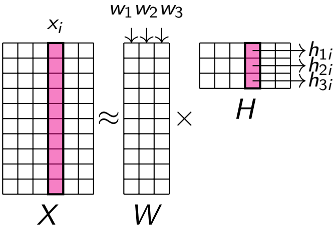
SIAM Gene Golub Summer School, Aussois France,
June 18, 2019

This work was supported in part by NSF, Homeland Security, DARPA

Overview

Linear Algebra is an important foundation in data analytics.

- Constrained low-rank approximations (CLRA) for modeling and algorithm/software development for scalable data analytics



PCA, SVD, LSI, pLSI, K-means
Topic/trend/video tracking
Community structure discovery

Recommendation system, ...

Lecture 2 Outline:

- Introduction to Low Rank Approximation
- Dimension Reduction
- Clustering of data represented in a feature-object matrix (attribute/content): data clustering, topic modeling, ...

Constrained Low Rank Approximations for Scalable Data Analytics

Objectives: Using CLRA,

- Model text and graph analytics problems
- Design, verify, and deploy scalable numerical alg.

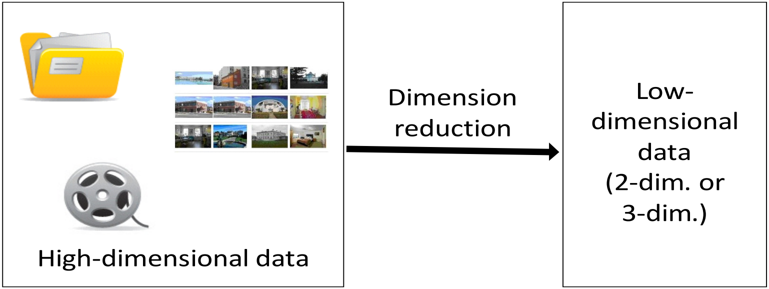
Goal: Orders of magnitude improvements over existing data analytics methods and solutions of higher quality

Why CLRA ?

- Utilize advances in numerical linear algebra and optimization
- Exploit software such as BLAS and LAPACK
- Behavior of algorithms easier to analyze
- Adaptive algorithms for streaming data
- Facilitates design of MPI based algorithms for scalable solutions
- Can easily be modified for various problem demands

Dimension Reduction

- Goal: Represent high-dimensional data in a lower dimension in order to visualize it or to make subsequent computation manageable.
- Input: Data $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^m$, reduced dimension k
- Output: Reduced-dimensional representation of data $y_1, y_2 \ldots, y_n \in \mathbb{R}^k$
  - Local, Global
  - Linear, Nonlinear
  - Unsupervised, Semi-supervised, Supervised



Curse of dimensionality

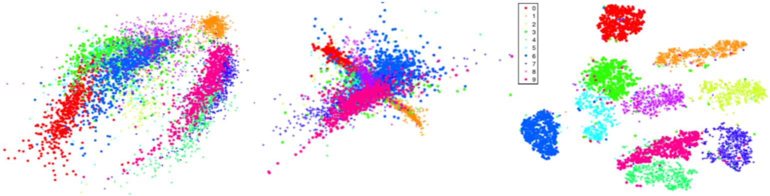
- When dimensionality increases, data bacomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful

Purpose and Techniques

- Purpose
  - Avoid curse of dimensionality
  - Reduce computational time and memory for algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise
- Techniques
  - Feature selection: Finding a subset of the original variables, e.g., features or attributes, which represent the original data.
  - Feature extraction: Transforms the data to a space of fewer dimensions. Different from feature selection, the features does not have to be the features in the original data.
    - Multidimensional scaling (MDS)
    - Principal component analysis (PCA)
    - Latent semantic analysis (LSA)
    - Non-negative matrix factorization (NMF)
    - Linear discriminant analysis (LDA)
    - Isometric Feature Mapping (ISOMAP)
    - Locally Linear Embedding (LLE)
    - Laplacian Eigenmaps
    - T-SNE (T-Distributed Stochastic Neighborhood Embedding) ...

Linear and Nonlinear Methods

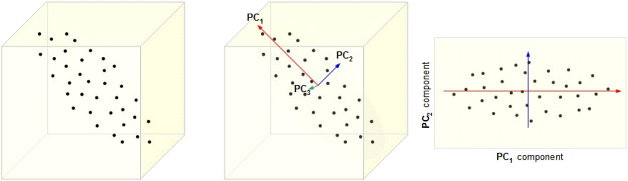
- PCA and SVD are mainly concerned with larger distances
  - Euclidian distance does not reflect similarity in high dim space very well when considering the structure of the data
  - Need methods preserving local structure focusing more on small pairwise distances
- ISOMAP: Geodesic distance in the data space, embedding is a little better
- LLE: Similar to t-SNE, focuses more on small pairwise distances, collapses many points to the center and lets outliers to satisfy the constraints



ISOMAP LLE t-SNE
MNIST vis (Handwritten digits) (van der Maaten and Hilton, JMLR08)

PCA (Principal Component Analysis)

- High level view
  - Seeks the most accurate data representation in a lower dimensional space
  - Preserves as much varience as possible
  - Not suitable for clustered data



Feature Extraction: PCA

- Algorithm
  - Can solve it using SVD
  - Let $X \in \mathbb{R}^{m \times n}$ be the data matrix where $m = \#$ of features, $n = \#$ of data samples.
  - Mean centering: Substract the centroid from every column
  - SVD: $X_c = U \Sigma V^T$ where U and V are orthogonal matrices and Σ is a diagonal matrix containing singular values.
  - Here, the columns of U can be seen as eigenvector of the empirical covarience matrix. ($X_c X_c^T = U \Sigma^2 U^T$)
  - Projection to k-dim. space: $Y = U_k^T X_c = \Sigma_k V_k^T$ (U_k, Σ_k, and V_k are the first k columns of U, Σ, and V respectively)

Feature Extraction: LSA (Latent Semantic Analysis)

- LSA exploits co-occurences of terms in documents to produce a mapping into a latent semantic space
- The new semantic space can find similar documents and relations between terms
- Term-Document Matrix
  - Let's assume we have three documents as following
    - D1: "I like visual analytics"
    - D2: "Visual representations and visual interactions"
    - D3: "Analytical reasoning and visualization"
  - TF+IDF, stop list removal, ...

$$
\begin{array}{c c c}
\text{Terms} & & \begin{array}{ccc} \text{D1} & \text{D2} & \text{D3} \end{array} \\
\begin{array}{c}
\text{analytical} \\ \text{analytics} \\ \text{and} \\ \text{I} \\ \text{interactions} \\ \text{like} \\ \text{reasoning} \\ \text{representations} \\ \text{visual} \\ \text{visualization}
\end{array}
& X = &
\left[\begin{array}{ccc}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 2 & 0 \\
0 & 0 & 1
\end{array}\right]
\end{array}
$$

Feature Extraction: LSA (continued)

Algorithm

- Applies SVD on the term-document matrix X
  $X = U_r S_r V_r^T$ where $X \in \mathbb{R}^{m \times n}$, $U_r \in \mathbb{R}^{m \times r}$, $S_r \in \mathbb{R}^{r \times r}$, and $V_r \in \mathbb{R}^{n \times r}$ (r is rank of X)
- Pick the top k largest singular values (in matrix S_r) and its corresponding singular vectors (in matrices U_r and V_r)

Feature Extraction: NMF (Non-negative Matrix Factorization)

Given the a non-negative matrix $X \in \mathbb{R}^{m \times n}$ and an integer $k \ll min\{m, n\}$, find non-negative matrices $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$, which minimizes $\|X - WH\|_F^2 = \sum_i \sum_j (X_{ij} - [WH]_{ij})^2$.

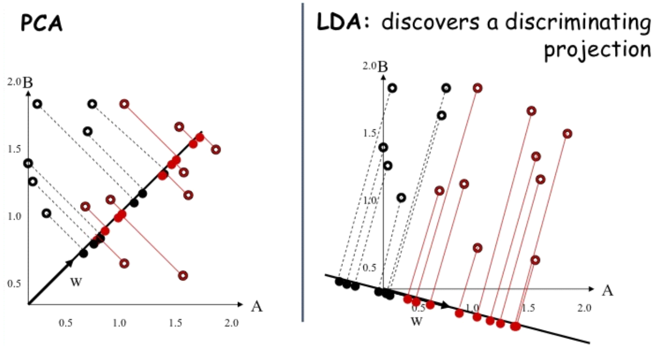
- W: basis for a k-dim space, the ith col of H: k-dim representation of the ith col of X
- Maintaining non-negativity prevents one factor from removing content that another factor contributed. NMF can uncover latent factors with better interpretability.
- Algorithm
  - Nonconvex
  - The factors are not unique
    (e.g., If we have a nonsingular matrix P, where $WP \geq 0$ and $P^{-1}H \geq 0$, then $(\tilde{W} = WP,\ \tilde{H} = P^{-1}H)$ is another solution, e.g. P is a diagonal matrix with positive diagonal elts.

Feature Extraction: LDA(Linear Discriminant Analysis)

P. Howland and HP, TPAMI 2004

- A supervised dimension reduction method for clustered data
- Maximizes between class separation while minimizing data separation within each class
- Algorithm
  - Want a linear transformation $G^T : x \in R^{m\times 1} \to y \in R^{l\times 1}$ assuming X is already clustered into k clusters $X = [X_1, \cdots, X_k]$
  - Two scatter matrices:
    Between-cluster scatter matrix
    $S_b = \sum_{j=1}^{k} \sum_{i\in C_j} (c^{(j)} - c)(c^{(j)} - c)^T$
    Within-cluster scatter matrix
    $S_w = \sum_{j=1}^{k} \sum_{i\in C_j} (x_i - c^{(j)})(x_i - c^{(j)})^T$
    $c^{(j)}$: centroid for the jth cluster, c: global centroid
  - Find G that
    maximizes trace$(G^T S_b G)$ while minimzes trace$(G^T S_w G)$
  - Find G that maximizes trace$((G^T S_w G)^{-1} G^T S_b G)$
  - Solved by a generalized eigenvalue problem

Graphical Example of PCA vs. LDA



(Figure from http://stuff.ttoy.net/cs591o/)

Generalized Singular Value Decomposition (GSVD)

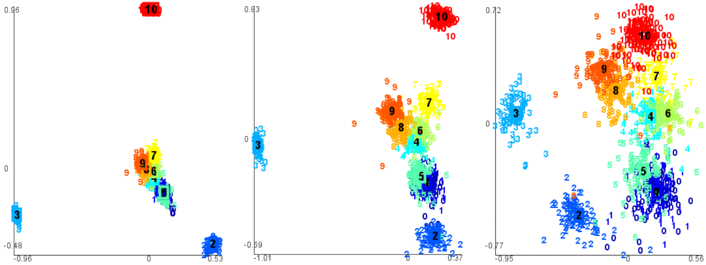
- GSVD: For $K_A \in R^{m \times n}$ with $m \geq n$ and $K_B \in R^{p \times n}$,
  there are $U \in R^{m \times m}$ and $V \in R^{p \times p}$
  with $U^T U = I$ and $V^T V = I$,
  and a nonsingular $X \in R^{n \times n}$ such that
  $U^T K_A X = diag(\alpha_1, \cdots, \alpha_n)$ and $V^T K_B X = diag(\beta, \cdots, \beta_q)$
  where $q = min(p, n)$, $\alpha_i \geq 0$, and $\beta_i \geq 0$.
- Through GSVD, we can solve generalized EVD:
  $\beta_i^2 K_A^T K_A x_i = \alpha_i^2 K_B^T K_B x_i$
- No nonsingularity condition on $K_B^T K_B$ is needed

LDA via GSVD

- $S_t = S_w + S_b$: total scatter matrix = covariance matrix
- PCA solves maximize $trace(G^T S_t G)$
- Letting
  $H_w = [X_1 - c^{(1)} {e^{(1)}}^T, \cdots, X_k - c^{(k)} {e^{(k)}}^T]$ and
  $H_b = [(c^{(1)} - c){e^{(1)}}^T, \cdots, (c^{(k)} - c){e^{(k)}}^T]$,
  we have $S_w = H_w H_w^T$ and $S_b = H_b H_b^T$
- GEVD of $\beta^2 S_b x = \alpha^2 S_w x$ can be solved via GSVD of H_w^T and H_b^T, regardless of nonsingularity of S_w

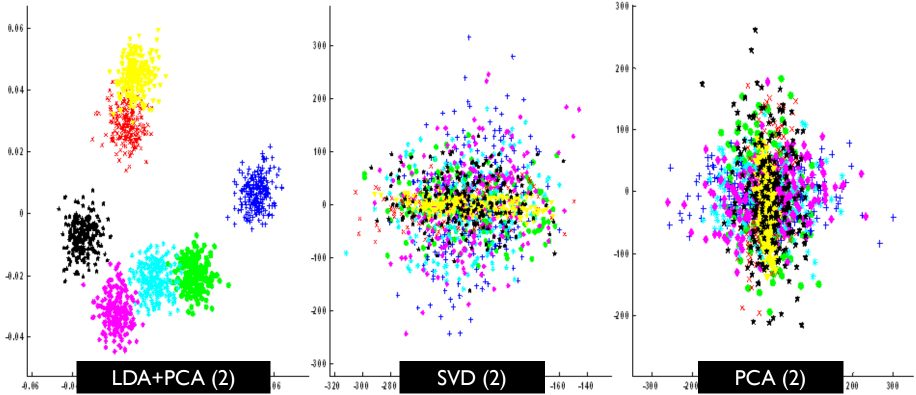
- LDA/GSVD algorithm
- Another way to handle singular S_w: Regularized LDA
  - Used when the data is undersampled, i.e., when # of dim > # of samples.
  - Regularized LDA max $trace((G^T S_w G + \gamma I)^{-1} G^T S_b G)$, where $G^T S_w G + \gamma I$ is guaranteed to be nonsingular for $\gamma > 0$

Application of LDA for 2D vis of clustered data



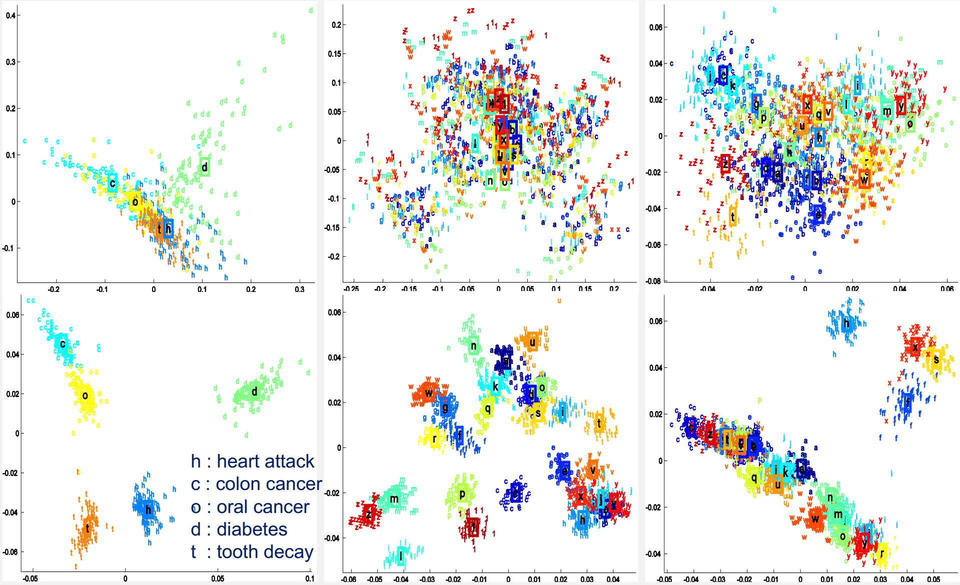
$\gamma = 10^0$ $\gamma = 10^1$ $\gamma = 10^2$

LDA/GSVD FOR 2D REPRESENTATION OF HIGH DIMENSIONAL CLUSTERED DATA



2D representation of 700x1000 data with 7 clusters: LDA vs. SVD vs. PCA
Want to represent data/cluster/outlier info even after a severe dim. reduction

2D VISUALIZATION (PCA VS LDA) OF CLUSTERED TEXT, IMAGE, AUDIO DATA



Medline Data *(Text)* | Facial Data *(Image)* | Spoken Letters *(Audio)*

Clustering

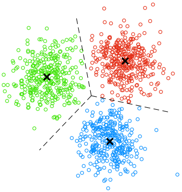
Given an unlabeled data set (no prior knowledge), how can we find grouping structures/patterns hidden in the data?

- Goal: Group *similar* objects together.
- Input: Data $X = [x_1, x_2, \ldots, x_n]$, number of clusters k
- Output: Data partitioning $X_1, X_2, \ldots, X_k$
- It provides an overview of large-scale data, dimension reduced representations, makes subsequent data analytics tasks more efficient.
- Core problem: k basis vectors where each represents a cluster well ? E.g. news articles:

Clustering

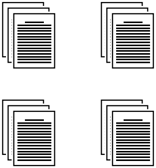
(J. Kim and H. Park, SISC 11; J.Kim, Y. He, and H. Park, JOGO 14, D. Kuang, S. Yun, and H. Park, JOGO 15)

Vector space



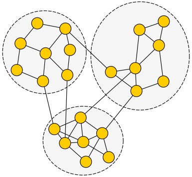
$$\min_{\substack{\mathbf{1}_k^T H = \mathbf{1}_n^T \\ H \in \{0,1\}^{k\times n}}} \|X - WH\|_F$$

Text (Topic Modeling)



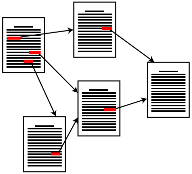
$$\min_{W\geq 0, H\geq 0} \|X - WH\|_F$$

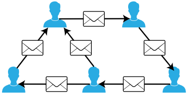
Graph (Community Detection)



$$\min_{H\geq 0} \|S - H^T H\|_F$$

Hybrid





$$\min_{W\geq 0, H\geq 0} \|X - WH\|_F^2 + \alpha \|S - H^T H\|_F^2$$

Clustering

Two most commonly used methods:

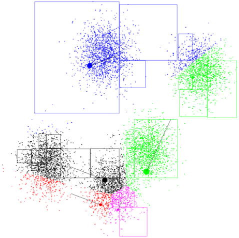
- For feature-data relationship: K-means
- For data-data relatinship: Spectral clustering

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0



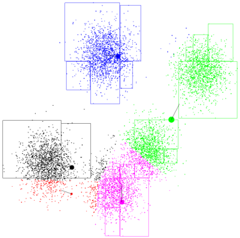
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1



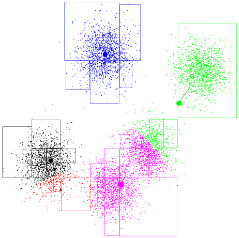
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2



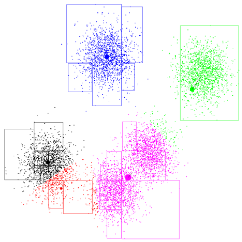
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3



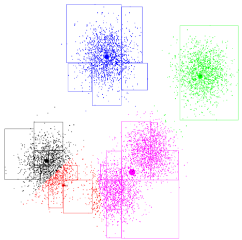
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4



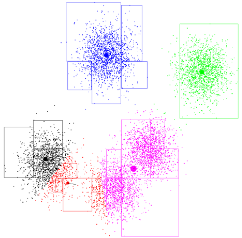
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5



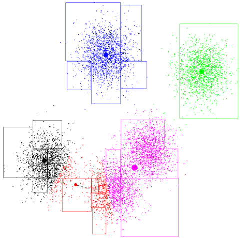
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6



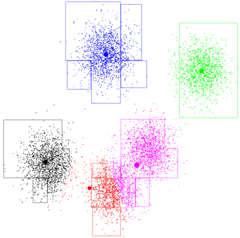
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6
- Step 7



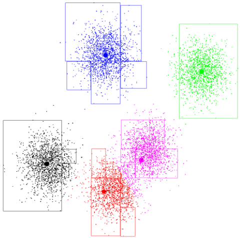
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6
- Step 7
- Step 8



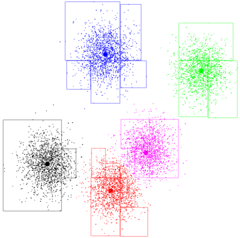
K-means algorithm for data matrix $X = [x_1, \cdots, x_n]$

Given k initial clustering centroids, K-means iteratively:

- Assigns each data point x_i to the nearest centroid in terms of Euclidean distance (or cosine value for spherical K-means)
- Recomputes k centroids

There are many versions of K-means and other variants such as K-median and K-medoids methods

- Step 0
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6
- Step 7
- Step 8



Given a notion of similarity $s_{ij} \geq 0$ between all pairs of data points, we form a similarity graph $G = (V, E)$.

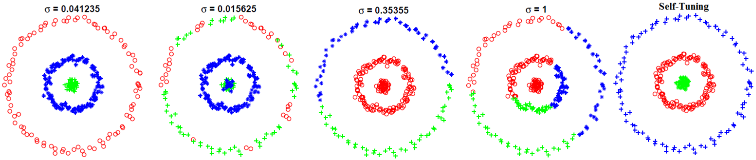
- Similarity Graph
  - The ε-neighborhood graph
    - Connect points whose pairwise distance are smaller than ε.
  - k-nearest neighbor graph (KNN graph)
    - Connect node x_i and x_j if x_j is among the k_n-nearest neighbors of x_i.
  - Mutual k-nearest neighbor graph (mutual KNN graph)
    - Connect node x_i and x_j if x_j is among the k_n-nearest neighbors of x_i and x_i is among the k-nearest neighbors of x_j.
  - Fully connected graph
    - Connect all points with an edge and assign the weight as pairwise positive similarity value.

[Von Luxburg, 2007]

A *self-tuned* way to define edge weight: [Zelnik-Manor and Penora, NIPS, 2004]

- $$s_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma_i \sigma_j}\right)$$

  where the *local scale* σ_i is the distance between x_i and its $\hat{k}$-th neighbor.



Spectral Clustering

Which similarity graph to choose? How to choose parameters in the similarity graph?

- A safe way is to build a *connected* graph.
- Choose ϵ, k_n, etc. so that the graph is connected.
- KNN graph is often a good starting point.
- The edge weight can be defined as the RBF kernel:

$$s_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$$

- But how to choose σ?
- Note: There is no theoretically principled way to choose the type of graph and parameters. And the clustering result is very sensitive to k and σ.

[Von Luxburg, 2007]

Which eigenvalues to compute?

Ratio cut: $\min \sum_{p=1}^{k} \frac{\sum_{i\in V_p, j\in V-V_p} W_{ij}}{\lvert V_p\rvert}$
k smallest eigenvalues of L

Ratio association: $\max \sum_{p=1}^{k} \frac{\sum_{i\in V_p, j\in V_p} W_{ij}}{\lvert V_p\rvert}$
k largest eigenvalues of W

Normalized cut: $\min \sum_{p=1}^{k} \frac{\sum_{i\in V_p, j\in V-V_p} W_{ij}}{\sum_{i\in V_p, j\in V} W_{ij}}$ $\Leftrightarrow \max \sum_{p=1}^{k} \frac{\sum_{i\in V_p, j\in V_p} W_{ij}}{\sum_{i\in V_p, j\in V} W_{ij}}$
k smallest eigenvalues of $D^{-1/2}LD^{-1/2}$ $\Leftrightarrow$ k largest eigenvalues of $D^{-1/2}WD^{-1/2}$

Spectral Clustering

The main tool for spectral clustering is the affinity matrix S and the graph Laplacian matrix L (n: number of nodes in graph)

- $S \in \mathbb{R}^{n \times n}$ contains edge weights between all connected pairs
- $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with degrees $d_1, d_2, \ldots, d_n$ on the diagonal: $d_i = \sum_{j=1}^{n} s_{ij}$
- L is the graph Laplacian matrix: $L = D - S$
  - Symmetric Positive Semi-definite
  - For every vector $f \in \mathbb{R}^n$, $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} s_{ij}(f_i - f_j)^2$
  - The smallest eigenvalue is 0 and its eigenvector has all ones $\vec{1}$
  - All eigenvalues are real and nonnegative $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
  - The multiplicity of the eigenvalue 0 is equal to the number of connected components in graph. Ex. $L = \begin{pmatrix} L_1 & 0 & 0 \\ 0 & L_2 & 0 \\ 0 & 0 & L_3 \end{pmatrix}$

Spectral Clustering

(One possible) Algorithm:

1. Input: Data points $x_1, \cdots, x_n$, number of clusters k
2. Construct a similarity graph and compute matrices S and L.
3. Compute the smallest k eigenvectors $u_1, u_2, \ldots, u_k$ of L.
4. Let $U = \begin{bmatrix} u_1 & u_2 & \ldots & u_k \end{bmatrix} \in \mathbb{R}^{n \times k}$.
5. For $i = 1, 2, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
6. Cluster the points $(y_i)_{i=1,2,\ldots,n}$ with the k-means algorithms to clusters $C_1, C_2, \ldots, C_k$.
7. Output: Partitioned data $X_j = \{i : y_i \in C_j\}$.

From objective functions to eigenvalues: Spectral relaxation

- Normalized cut objective: $\min \sum_{p=1}^{k} \frac{\sum_{i\in V_p, j\in V-V_p} S_{ij}}{\sum_{i\in V_p, j\in V} S_{ij}}$
- Define cluster indicator vector:
  $h_p = D^{1/2}[0,\cdots,0,1,\cdots,1,0,\cdots,0]^T$ with n_p 1's
- Define normalized indicator $y_p = h_p/\|h_p\|_2$, and rewrite J:

$$J = \sum_{p=1}^{k} y_p^T L y_p = \mathrm{trace}(Y^T D^{-1/2} L D^{-1/2} Y)$$

  where $Y = [y_1, \cdots, y_k]$
- Relax the constraints of Y to be $Y^T Y = I$:

$$\min_{Y^T Y = I} \mathrm{trace}(Y^T D^{-1/2} L D^{-1/2} Y)$$

- By *Ky Fan* theorem, the optimal Y is the eigenvectors corresponding to the smallest eigenvalues of $D^{-1/2} L D^{-1/2}$