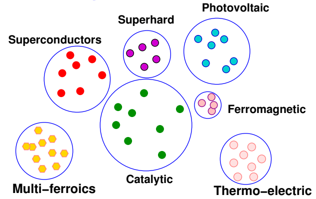
APPLICATIONS OF GRAPH LAPLACEANS: CLUSTERING

- **Details on clustering**
- **K-means**
- **Similarity graphs, KNN graphs**
- **Edge cuts, ratio cuts, etc.**
- **Application: segmentation**

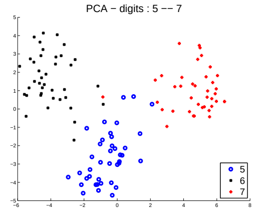
Clustering: Background

➤ Problem: we are given n data items: $x_1, x_2, \cdots, x_n$. Would like to *'cluster'* them, i.e., group them so that each group or cluster contains items that are similar in some sense.

➤ Example: materials



➤ Example: Digits



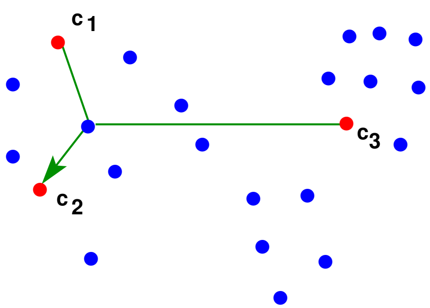
➤ Refer to each group as a 'cluster' or a 'class'

➤ A basic method: K-Means

A basic method: K-means

➤ A basic algorithm that uses Euclidean distance

1	Select p initial centers: $c_1, c_2, ..., c_p$ for classes $1, 2, \cdots, p$
2	For each x_i do: determine *class* of x_i as $\text{argmin}_k \|x_i - c_k\|$
3	Redefine each c_k to be the centroid of class k
4	Repeat until convergence



➤ Simple algorithm
➤ Works well (gives good results) but can be slow
➤ Performance depends on initialization

Methods based on similarity graphs

➤ Class of Methods that perform clustering by exploiting a graph that describes the similarities between any two items in the data.

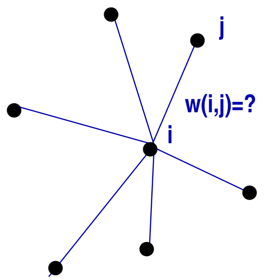
➤ Need to:

1. decide what nodes are in the neighborhood of a given node ?
2. quantify their similarities - by deciding on weights between any two 'similar' nodes.

Example: For text data: Can decide that any columns i and j with a cosine greater than 0.95 are 'similar' and assign that cosine value to w_{ij}

First task: build a 'similarity' graph

➤ Goal: to build a similarity graph, i.e., a graph that captures similarity between any two items



➤ Two methods: K-nearest Neighbor graphs or use Gaussian ('heat') kernel

K-nearest neighbor graphs

➤ Given: a set of n data points $X = \{x_1, \ldots, x_n\} \rightarrow$ vertices

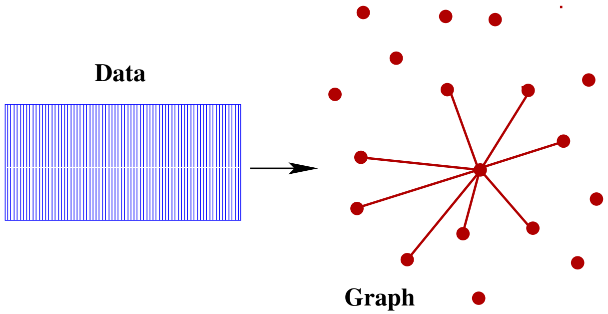
➤ Given: a proximity measure between two data points x_i and x_j – as measured by a quantity $dist(x_i, x_j)$

➤ Want: For each point x_i a list of the 'nearest neighbors' of x_i (edges between x_i and these nodes).

➤ Note: graph will usually be directed $\rightarrow$ need to symmetrize

Nearest neighbor graphs

➤ For each node, get a few of the nearest neighbors $\rightarrow$ Graph



➤ Problem: How to build a nearest-neighbor graph from given date

➤ We will revisit this later.

Two types of nearest neighbor graph often used:

ϵ-graph: Edges consist of pairs (x_i, x_j) such that $\rho(x_i, x_j) \leq \epsilon$

kNN graph: Nodes adjacent to x_i are those nodes x_ℓ with the k with smallest distances $\rho(x_i, x_\ell)$.

➤ ϵ-graph is undirected and is geometrically motivated. Issues: 1) may result in disconnected components 2) what ϵ?

➤ kNN graphs are directed in general (can be trivially fixed).

➤ kNN graphs especially useful in practice.

Similarity graphs: Using 'heat-kernels'

Define weight between i and j as:

$$w_{ij} = f_{ij} \times \begin{cases} e^{\frac{-\|x_i - x_j\|^2}{\sigma_X^2}} & \text{if } \|x_i - x_j\| < r \\ 0 & \text{if not} \end{cases}$$

- Note $\|x_i - x_j\|$ could be any measure of distance...
- f_{ij} = optional = some measure of similarity - other than distance
- Only nearby points kept.
- Sparsity depends on parameters

Edge cuts, ratio cuts, normalized cuts, ...

- Assume now that we have built a 'similarity graph'
- Setting is identical with that of graph partitioning.
- Need a Graph Laplacean: $L = D - W$ with $w_{ii} = 0, w_{ij} \geq 0$ and $D = diag(W * ones(n,1))$ [in matlab notation]
- Partition vertex set V in two sets A and B with

$$A \cup B = V, \quad A \cap B = \emptyset$$

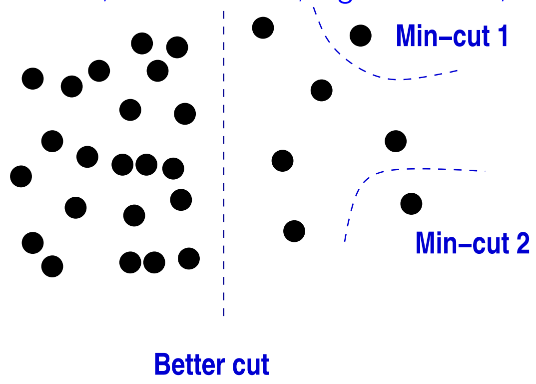
- Define

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

- First (naive) approach: use this measure to partition graph, i.e.,

... Find A and B that minimize $cut(A,B)$.

- Issue: Small sets, isolated nodes, big imbalances,



Ratio-cuts

- Standard Graph Partitioning approach: Find A, B by solving

Minimize $cut(A,B)$, subject to $|A| = |B|$

- Condition $|A| = |B|$ not too meaningful in some applications - too restrictive in others.
- Minimum Ratio Cut approach. Find A, B by solving:

Minimize $\frac{cut(A,B)}{|A|.|B|}$

- Difficult to find solution (original paper [Wei-Cheng '91] proposes several heuristics)
- Approximate solution : spectral .

Theorem [Hagen-Kahng, 91] If λ_2 is the 2nd smallest eigenvalue of L, then a lower bound for the cost c of the optimal ratio cut partition, is:

$$c \geq \frac{\lambda_2}{n}.$$

Proof: Consider an optimal partition A, B and let $p = |A|/n, q = |B|/n$. Note that $p+q = 1$. Let x be the vector with coordinates

$$x_i = \begin{cases} q & \text{if } i \in A \\ -p & \text{if } i \in B \end{cases}$$

Note that $x \perp \mathbf{1}$. Also if $(i,j) ==$ an edge-cut then $x_i - x_j = q - (-p) = q + p = 1$, otherwise $x_i - x_j = 0$. Therefore, $x^T L x = \sum_{(i,j)\in E}(x_i - x_j)^2 = w(A,B)$. In addition: $\|x\|^2 = pq^2 n + qp^2 n = pq(p+q)n = pqn = \frac{|A|.|B|}{n}$.

Therefore, by the Courant-Fischer theorem:

$$\lambda_2 \leq \frac{(Lx,x)}{(x,x)} = n \times \frac{w(A,B)}{|A|.|B|} = n \times c.$$

Hence result. ■

- Idea is to use this eigenvector to determine partition, e.g., based on sign of entries. Use the ratio-cut measure to actually determine where to split.

Normalized cuts [Shi-Malik,2000]

- Recall notation $w(X,Y) = \sum_{x\in X, y\in Y} w(x,y)$ - then define:

$$\text{ncut}(A,B) = \frac{cut(A,B)}{w(A,V)} + \frac{cut(A,B)}{w(B,V)}$$

- Goal is to avoid small sets A, B
- What is $w(A,V)$ in the case when $w_{ij} == 1$?
- Let x be an indicator vector:

$$x_i = \begin{cases} 1 \ if \ i \in A \\ 0 \ if \ i \in B \end{cases}$$

- Recall that: $x^T L x = \sum_{ij} w_{ij} |x_i - x_j|^2$

- Therefore:

$$cut(A,B) = \sum_{x_i=1, x_j=0} w_{ij} = x^T L x$$

$$w(A,V) = \sum_{x_i=1} d_i = x^T W \mathbf{1} = x^T D \mathbf{1}$$

$$w(B,V) = \sum_{x_j=0} d_j = (\mathbf{1} - x)^T W \mathbf{1} = (\mathbf{1} - x)^T D \mathbf{1}$$

- Goal now: to minimize ncut

$$\min_{A,B} \text{ncut}(A,B) = \min_{x_i \in \{0,1\}} \frac{x^T L x}{x^T D x} + \frac{x^T L x}{(\mathbf{1} - x)^T D x}$$

➤ Let

$$\beta = \frac{w(A,V)}{w(B,V)} = \frac{x^T D1}{(1-x)^T D1}$$
$$y = x - \beta(1-x)$$

➤ Then we need to solve:

$$\min_{y_i \ \{0,-\beta\}} \frac{y^T L y}{y^T D y}$$
Subject to $y^T D1 = 0$

➤ + Relax $\longrightarrow$ need to solve Generalized eigenvalue problem

$$Ly = \lambda D y$$

➤ $y_1 = 1$ is eigenvector associated with eigenvalue $\lambda_1 = 0$

➤ y_2 associated with second eigenvalue solves problem.

A few properties

Show that

$$ncut(A,B) = \sigma \times \frac{cut(A,B)}{w(A,V) \times w(B,V)}$$

where σ is a constant

How do ratio-cuts and normalized cuts compare when the graph is d-regular (same degree for each node).

Extension to more then 2 clusters

➤ Just like graph partitioning we can:

1. Apply the method recursively [Repeat clustering on the resulted parts]
2. or compute a few eigenvectors and run K-means clustering on these eigenvectors to get the clustering.

Application: Image segmentation

➤ First task: obtain a graph from pixels.

➤ Common idea: use "Heat kernels"

➤ Let F_j = feature value (e.g., brightness), and Let X_j = spatial position.

Then define

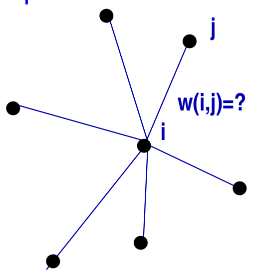
$$w_{ij} = e^{\frac{-\|F_i - F_j\|^2}{\sigma_I^2}} \times \begin{cases} e^{\frac{-\|X_i - X_j\|^2}{\sigma_X^2}} & \text{if} \|X_i - X_j\| < r \\ 0 & \text{else} \end{cases}$$

➤ Sparsity depends on parameters

Spectral clustering: General approach

1 Given: Collection of data samples $\{x_1, x_2, \cdots, x_n\}$

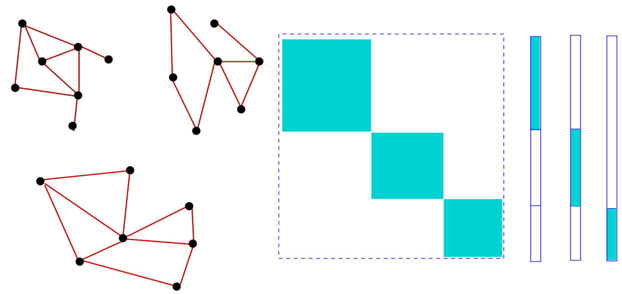
2 Build a similarity graph between items



3 Compute (smallest) eigenvector (s) of resulting graph Laplacean

4 Use k-means on eigenvector (s) of Laplacean

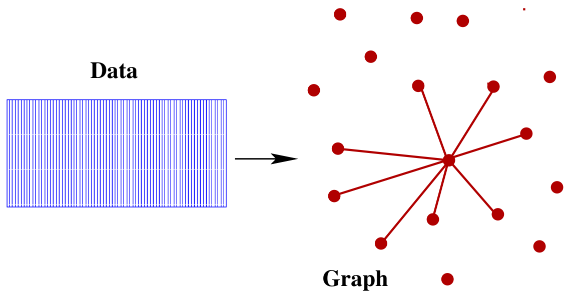
➤ For Normalized cuts solve generalized eigen problem.



➤ Alg. Multiplicity of eigenvalue zero = # connected components.

Building a nearest neighbor graph

➤ Question: How to build a nearest-neighbor graph from given data?



➤ Will demonstrate the power of a divide a conquer approach combined with the Lanczos algorithm.

➤ Note: The Lanczos algortithm will be covered in detail later

Recall: Two common types of nearest neighbor graphs

ϵ-graph: Edges consist of pairs (x_i, x_j) such that $\rho(x_i, x_j) \le \epsilon$

kNN graph: Nodes adjacent to x_i are those nodes x_ℓ with the k with smallest distances $\rho(x_i, x_\ell)$.

➤ ϵ-graph is undirected and is geometrically motivated. Issues: 1) may result in disconnected components 2) what ϵ?

➤ kNN graphs are directed in general (can be trivially fixed).

➤ kNN graphs especially useful in practice.

Divide and conquer KNN: key ingredient

- Key ingredient is *Spectral bisection*
- Let the data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$
- Each column == a data point.
- Center the data: $\hat{X} = [\hat{x}_1, \ldots, \hat{x}_n] = X - ce^T$
  where c == centroid; $e = ones(d, 1)$ (matlab)

Goal: Split $\hat{X}$ into halves using a hyperplane.

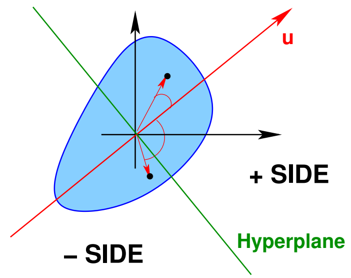
Method: Principal Direction Divisive Partitioning D. Boley, '98.

Idea: Use the (σ, u, v) = largest singular triplet of $\hat{X}$ with:

$$u^T \hat{X} = \sigma v^T.$$

- Hyperplane is defined as $\langle u, x \rangle = 0$, i.e., it splits the set of data points into two subsets:

$$X_+ = \{x_i \mid u^T \hat{x}_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid u^T \hat{x}_i < 0\}.$$



- Note that $u^T \hat{x}_i = u^T \hat{X} e_i = \sigma v^T e_i \rightarrow$

$$X_+ = \{x_i \mid v_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid v_i < 0\},$$

where v_i is the i-th entry of v.

- In practice: replace above criterion by

$$X_+ = \{x_i \mid v_i \geq \text{med}(v)\} \;\&\; X_- = \{x_i \mid v_i < \text{med}(v)\}$$

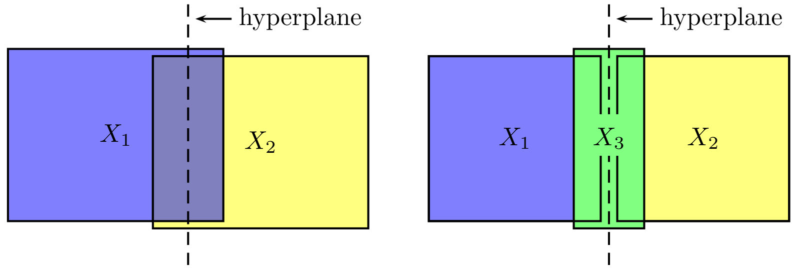
where *med*(v) == *median of the entries of* v.

- For largest singular triplet (σ, u, v) of $\hat{X}$: use Golub-Kahan-Lanczos algorithm or Lanczos applied to $\hat{X}\hat{X}^T$ or $\hat{X}^T\hat{X}$
- Cost (assuming s Lanczos steps) : $O(n \times d \times s)$; Usually: d very small

Two divide and conquer algorithms

Overlap method: divide current set into two overlapping subsets X_1, X_2

Glue method: divide current set into two disjoint subsets X_1, X_2 plus a third set X_3 called gluing set.



The Overlap Method

➤ Divide current set X into two overlapping subsets:

$$X_1 = \{x_i \mid v_i \geq -h_\alpha(S_v)\} \quad \text{and} \quad X_2 = \{x_i \mid v_i < h_\alpha(S_v)\},$$

- where $S_v = \{|v_i| \mid i = 1, 2, \ldots, n\}$.
- and $h_\alpha(\cdot)$ is a function that returns an element larger than $(100\alpha)\%$ of those in S_v.

➤ Rationale: to ensure that the two subsets overlap $(100\alpha)\%$ of the data, i.e.,

$$|X_1 \cap X_2| = \lceil \alpha |X| \rceil .$$

The Glue Method

Divide the set X into two disjoint subsets X_1 and X_2 with a gluing subset X_3:

$$X_1 \cup X_2 = X, \quad X_1 \cap X_2 = \emptyset, \quad X_1 \cap X_3 \neq \emptyset, \quad X_2 \cap X_3 \neq \emptyset.$$

Criterion used for splitting:

$$X_1 = \{x_i \mid v_i \geq 0\}, \quad X_2 = \{x_i \mid v_i < 0\},$$
$$X_3 = \{x_i \mid -h_\alpha(S_v) \leq v_i < h_\alpha(S_v)\}.$$

Note: gluing subset X_3 here is just the intersection of the sets X_1, X_2 of the overlap method.

Approximate kNN Graph Construction: The Overlap Method

```
function G = kNN-Overlap[X, k, α]
   if |X| < n_k
      G ← Call kNN-BruteForce[X, k]
   else
      (X1, X2) ← Call Divide-Overlap[X, α]
         G1 ← Call kNN-Overlap[X1, k, α]
         G2 ← Call kNN-Overlap[X2, k, α]
         G ← Call Conquer[G1, G2]
         Call Refine[G]
   EndIf
End
```


Approximate kNN Graph Construction: The Glue Method

```
G = kNN-Glue[X, k, α]
   if |X| < n_k
      G ← Call kNN-BruteForce[X, k]
   else
      (X1, X2, X3) ← Call Divide-GlueX, α
      G1 ← Call kNN-Glue[X1, k, α]
      G2 ← Call kNN-Glue[X2, k, α]
      G3 ← Call kNN-Glue[X3, k, α]
      G ← Call Conquer[G1, G2, G3]
      Call Refine[G]
   EndIf
End
```

Theorem The time complexity for the overlap method is

$$T_\circ(n) = \Theta(dn^{t_\circ}),$$

where: $$t_\circ = \log_{2/(1+\alpha)} 2 = \frac{1}{1 - \log_2(1+\alpha)}.$$

Theorem The time complexity for the glue method is

$$T_g(n) = \Theta(dn^{t_g}/\alpha),$$

where t_g is the solution to the equation: $\frac{2}{2^t} + \alpha^t = 1.$

Example: When $\alpha = 0.1$, then $t_\circ = 1.16$ while $t_g = 1.12$.

Reference:

Jie Chen, Haw-Ren Fang and YS, "Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection" JMLR, vol. 10, pp. 1989-2012 (2009).