Lecture # 10 : 10-11-19

(*) = "Tutorial on Spectral Clustering" U. von Luxburg, 2007.

- A different family of approaches build a graph on data and attempt to learn clusters from the structure in the graph.
- Let's think of a graph G as a collection of vertices V and edges between the vertices E: $G=(V,E)$. $\{x_i\}_{i=1}^n \subset \mathbb{R}^D$
- We will associate V with the data points and E will "weight" between points.
- E will often be denoted as a $|V|\times|V|$ matrix W, where $W_{ij} \in [0,1]$, $i,j = 1,\dots,|V|$. $W_{ij} \sim 0 \Rightarrow$ weak connection between x_i, x_j
  $W_{ij} \sim 1 \Rightarrow$ strong connection between x_i, x_j.

ex: Fix some integer $k_{NN} \geq 1$. Let $W_{ij} = \begin{cases} 1, & x_i \text{ is among the } k_{NN}\text{-nearest neighbors of } x_j \\ 0, & \text{else} \end{cases}$

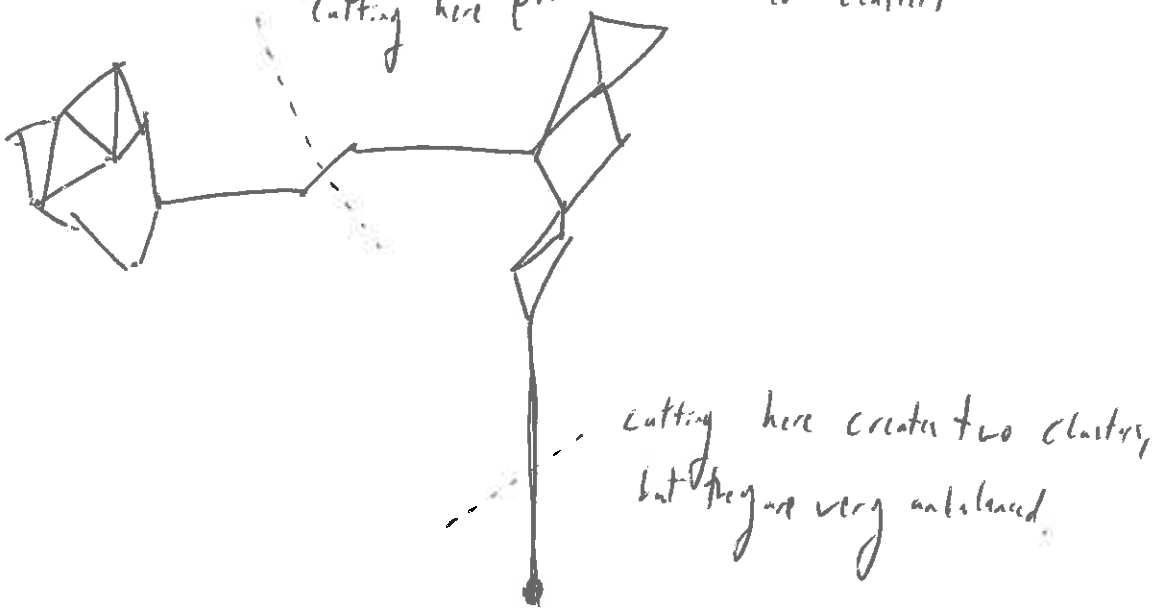
This is a k_{NN}-nearest neighbors graph, where x_i a k_{NN}-nearest neighbor of x_j means that when all pairwise distances $\{\|x_\ell - x_j\|_2\}_{\ell=1}^n$ are computed, $\|x_\ell - x_i\|_2$ is among the k_{NN} smallest.

Remark: If $k_{NN} \ll n$, this is a sparse graph. Computationally

advantageous in some cases.

e.g. $W_{ij} = e^{-\|x_i - x_j\|_2^2 / 2\sigma^2}$, $\sigma > 0$ fixed scale parameter. This connects all points since $e^{-\|x_i - x_j\|_2^2 / \sigma} > 0$ no matter what $\|x_i - x_j\|_2^2 / \sigma^2$ is, but it may be very small.

- Given these weights on the graph, we can try to break the graph into two pieces that "cut" as little structure as possible:



Attempt 1: Partition so that the weight of edges broken is as small as possible.

$$Cut(C_1, C_2) = \sum_{\substack{x_i \in C_1 \\ x_j \in C_2}} W_{ij}$$

Choose clusters to minimize this cut:

$$(C_1^*, C_2^*) = \underset{(C_1, C_2)}{\operatorname{argmin}} \; Cut(C_1, C_2) = \underset{(C_1, C_2)}{\operatorname{argmin}} \sum_{\substack{x_i \in C_1 \\ x_j \in C_2}} W_{ij}$$

- Seems reasonable... but could end up breaking a small number of edges connecting outliers to the main data bulk.

Attempt 2: Partition so that the weight of the edges broken is small and also so that the "size" of the clusters is larger.

$$Ncut(C_1, C_2) = \frac{\sum_{\substack{x_i \in C_1 \\ x_j \in C_2}} w_{ij}}{\sum_{\substack{x_i \in C_1 \\ x_j \in C_1}} w_{ij}} + \frac{\sum_{\substack{x_i \in C_1 \\ x_j \in C_2}}}{\sum_{\substack{x_i \in C_2 \\ x_j \in C_2}} w_{ij}}$$

$$= \sum_{k=1}^{2} \frac{\sum_{\substack{x_i \in C_k \\ x_j \notin C_k}} w_{ij}}{\sum_{x_i, x_j \in C_k} w_{ij}}$$

Setting $vol(C_l) = \sum_{\substack{x_i \in C_l \\ x_j \in C_l}} w_{ij}$, this simplifies to

$$Ncut(C_1, C_2) = \sum_{k=1}^{2} \left(\sum_{\substack{x_i \in C_k \\ x_j \notin C_k}} w_{ij} \Big/ vol(C_k) \right)$$

- Compared to Attempt 1, this is normalized by the volume factor $vol(C_k)$.
- In particular, if $vol(C_k)$ is very small, Ncut blows up... penalizing small clusters, thus enforcing balance.

$$(C_1^*, C_2^*) = \underset{(C_1,C_2)}{\arg\min}\ Ncut(C_1,C_2)$$

$$= \underset{(C_1,C_2)}{\arg\min} \sum_{k=1}^{2} \Big(\sum_{\substack{x_i \in C_k \\ x_j \notin C_k}} w_{ij} \Big) \Big/ vol(C_k) \quad (\bigstar)$$

Question: 1.) How to generalize to more than 2 clusters.
2.) How to compute.

Answer: 1.) Easy: if you want k clusters, extend $(\bigstar)$ to sum over k partition elements:

$$(C_1^*, C_2^*, \ldots, C_k^*) = \underset{(C_1,C_2)}{\arg\min} \sum_{\ell=1}^{K} \Big(\sum_{\substack{x_i \in C_\ell \\ x_j \notin C_\ell}} w_{ij} \Big) \Big/ vol(C_\ell).$$

2.) Hard: Computing the exact solution to $(\bigstar)$ is NP-hard. It is a combinatorial nightmare, and one can prove there is no efficient algorithm. Basically, you have to check all possibility in the worst case, which means exponentially many in n computation. Bad!

- Instead of calculating $(\bigstar)$ exactly, we can use linear algebra to relax $(\bigstar)$.
- Indeed, let us consider a partition (C_1, C_2). Let f_C be defined as:

$$\bar{C} = V \setminus C. \qquad (C, \bar{C})$$

$$f_i^C = \begin{cases} \sqrt{\frac{vol(\bar{C})}{vol(C)}}, & x_i \in C \\ -\sqrt{\frac{vol(C)}{vol(\bar{C})}}, & x_i \in \bar{C} \end{cases}$$

- This is $+$ for points in C, $-$ for points in $\bar{C}$, and moreover is closely related to the degree matrix: $D_{ij} = \begin{cases} \sum_{x_j \in V} w_{ij}, & i = l \\ 0, & \text{else} \end{cases}$

$$D = \begin{pmatrix} \sum_j w_{1j} & & 0 \\ & \sum_j w_{2j} & \\ 0 & & \ddots \\ & & & \sum_j w_{nj} \end{pmatrix}. \quad \text{Then } (Df^C)^T \mathbf{1} = 0,$$

(Df^C: column vectors)

where $\mathbf{1} = (1, \dots, 1)^T$.

- Moreover, $(f^C)^T D f^C = vol(V) = \sum_{x_i, x_j} w_{ij}$.

- Most importantly, f^C is intimately related to $Ncut(C, \bar{C})$ through the graph Laplacian $L = D - W$. Then $f^C L f = vol(V) Ncut(C, \bar{C})$.

- So, instead of handling ⛤ directly, we can use this Laplacian formulation:

$$c^* = \underset{c}{\operatorname{argmin}} \; (f^c)^T L f^c \quad \text{s.t.} \quad 1.)\ (Df^c)^T \mathbf{1} = 0$$
$$2.)\ (f^c)^T D f = \mathrm{vol}(V).$$

- Great... so what? Still hard to solve. The problem is f^c has special form.

- Relax: $\underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \; f^T L f \quad \text{s.t.} \quad 1.)\ (Df)^T \mathbf{1} = 0$
$$2.)\ f^T D f = \mathrm{vol}(V)$$

↓ lose some accuracy, gain ease of computation.

This can be attacked with linear algebra... next time.