Learning from Data
Lecture 8: Unsupervised Learning I

Yang Li **yangli@sz.tsinghua.edu.cn**

TBSI

November 19, 2021

Today's Lecture

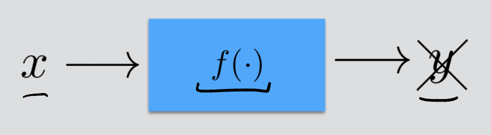
Unsupervised Learning (Part I)

- Overview: the representation learning problem
- K-means clustering
- Spectral clustering

Project Introduction

Unsupervised Learning Overview

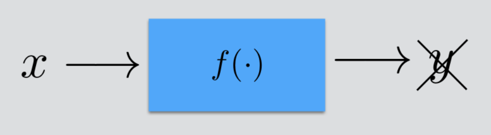
Unsupervised Learning



Similar to supervised learning, but without labels.

- Still want to learn the machine f
- Significantly harder in general

Unsupervised Learning



Similar to supervised learning, but without labels.

- Still want to learn the machine f
- Significantly harder in general

Unsupervised learning goal

Find **representations** of input feature x that can be used for reasoning, decision making, predicting things, comminicating etc.

The representation learning problem

(Y Bengio et. al. *Representation Learning: A Review and New Perspectives*, 2014)

Given input features x, find "simpler" features z that **preserve the same information** as x.

Example: Face recognition

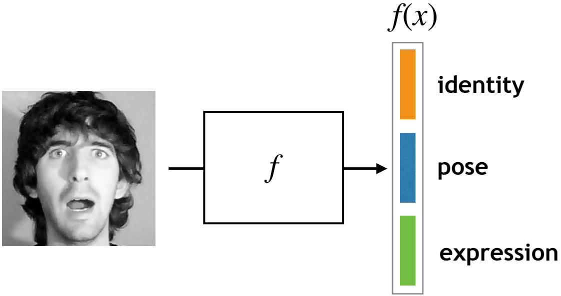
100×100

$$\rightarrow x = \begin{bmatrix} 0.5 \\ 0 \\ \vdots \\ 0.3 \\ 1.0 \end{bmatrix} \Big\} 10^4 \rightarrow \quad z = [\vdots]$$

What information is in this picture? *identity, facial attributes, gender, age, sentiment, etc*

Characteristics of a good representation

- low dimensional: compress information to a smaller size → *reduce data size*
- sparse representation: most entries are zero for most data → *better interpretability*
- independent representations: disentangle the source of variations

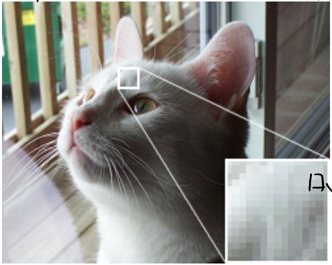


Uses of representation learning

- Data compression

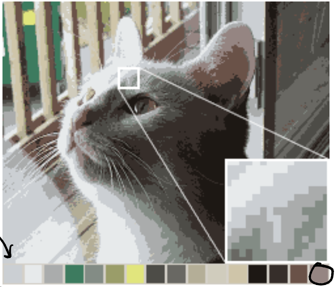
Example: Color image quantization. Each 24bit RGB color is reduced to a palette of 16 colors.

Original



(0-255,0-255,0-255)
24bit x 300 x 400

Compressed

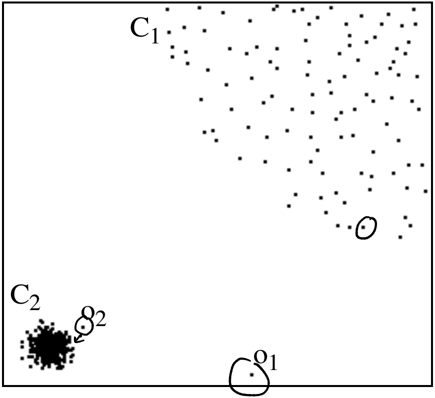


0-15
4bit x 300 x 400 + 16 x 24bit
6 times smaller

Uses of representation learning

- Abnormality (outlier, novelty) detection

Example: local density-based outlier detection



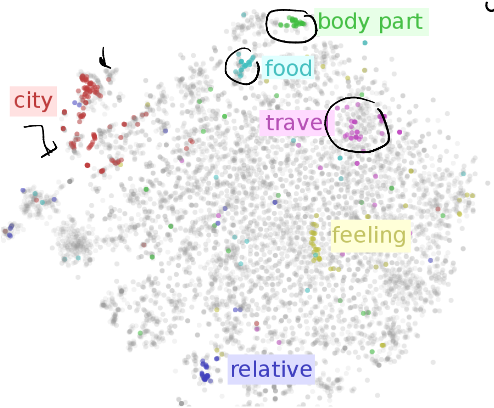
o_1 and o_2 are the detected outliers

Uses of representation learning

- Knowledge representation based on human perception

Example: word embedding

word2vec
word embedding



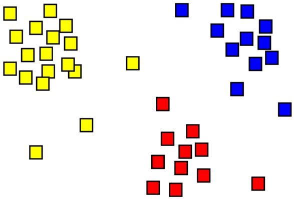
http://ruder.io/word-embeddings-1/

Each word is represented by a 2D vector. Words in the same semantic category are grouped together

K-Means Clustering

Clustering analysis

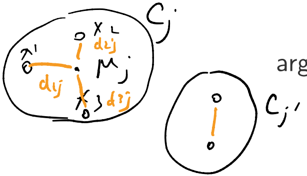
Given input features $\{x^{(1)}, \ldots, x^{(m)}\}$, group the data into a few *cohesive* "clusters".



- Objects in the same cluster are more similar to each other than to those in other clusters

The k-means clustering problem

Given input data $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$, **k-means clustering** partition the input into $k \leq m$ sets $C_1, \ldots, C_k$ to minimize the within-cluster sum of squares (WCSS).



$$\underset{C}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

The k-means clustering problem

Given input data $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$, **k-means clustering** partition the input into $k \leq m$ sets $C_1, \ldots, C_k$ to minimize the within-cluster sum of squares (WCSS).

$$\operatorname*{argmin}_{C} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

$$\mathrm{Var}(C_j) = \frac{1}{|C_j|} \sum_{x \in C_j} \|x - \mu_j\|^2$$

Equivalent definitions:

- minimizing the within-cluster variance: $\sum_{j=1}^{k} |C_j| \, \mathrm{Var}(C_j)$

$$= \sum_{j=1}^{k} |C_j| \frac{1}{|C_j|} \sum_{x \in C_j} \|x - \mu_j\|^2.$$

The k-means clustering problem

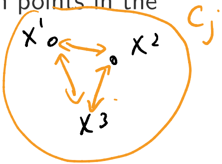
Given input data $\{x^{(1)}, \dots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$, **k-means clustering** partition the input into $k \leq m$ sets $C_1, \dots, C_k$ to minimize the within-cluster sum of squares (WCSS).

$$\underset{C}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

Equivalent definitions:

- minimizing the within-cluster variance: $\sum_{j=1}^{k} |C_j| \operatorname{Var}(C_j)$
- minimizing the pairwise squared deviation between points in the same cluster: *(homework)*

$$\sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{x, x' \in C_i} \|x - x'\|^2$$



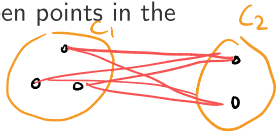
The k-means clustering problem

Given input data $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$, **k-means clustering** partition the input into $k \leq m$ sets $C_1, \ldots, C_k$ to minimize the within-cluster sum of squares (WCSS).

$$\underset{C}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

Equivalent definitions:

- minimizing the within-cluster variance: $\sum_{j=1}^{k} |C_j| \operatorname{Var}(C_j)$
- minimizing the pairwise squared deviation between points in the same cluster: *(homework)*



$$\sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{x, x' \in C_i} \|x - x'\|^2$$

- maximizing between-cluster sum of squares (BCSS) *(homework)*

K-Means Clustering Algorithm

- Optimal k-means clustering is NP-hard in Euclidean space.
- Often solved via a heuristic, iterative algorithm

K-Means Clustering Algorithm

- Optimal k-means clustering is NP-hard in Euclidean space.
- Often solved via a heuristic, iterative algorithm

$c^{(1)} = j \quad j = 1, \ldots, k.$

Lloyd's Algorithm (1957,1982)

Let $c^{(i)} \in \{1, \ldots, k\}$ be the cluster label for $x^{(i)}$

Initialize cluster centroids $\mu_1, \ldots \mu_k \in R^n$ randomly

Repeat until convergence{

For every i,

$$c^{(i)} := \text{argmin}_j \|x^{(i)} - \mu_j\|^2$$

update assignment

For each j

$$\mu_j := \frac{\sum_{i=1}^m \mathbf{1}\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{c^{(i)}=j\}}$$

update cluster centroid.

}

Demo:http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

Lloyd, Stuart P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory

K-Means Clustering Algorithm

- Optimal k-means clustering is NP-hard in Euclidean space.
- Often solved via a heuristic, iterative algorithm

Lloyd's Algorithm (1957,1982)

Let $c^{(i)} \in \{1, \dots, k\}$ be the cluster label for $x^{(i)}$

Initialize cluster centroids $\mu_1, \dots \mu_k \in R^n$ randomly
Repeat until convergence{
  For every i,
    $c^{(i)} := \text{argmin}_j \|x^{(i)} - \mu_j\|^2$ ← assign $x^{(i)}$ to the cluster with the closest centroid
  For each j

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\}}$$

}

Demo:http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

Lloyd, Stuart P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory

K-Means Clustering Algorithm

- Optimal k-means clustering is NP-hard in Euclidean space.
- Often solved via a heuristic, iterative algorithm

Lloyd's Algorithm (1957,1982)

Let $c^{(i)} \in \{1, \ldots, k\}$ be the cluster label for $x^{(i)}$

Initialize cluster centroids $\mu_1, \ldots \mu_k \in R^n$ randomly

Repeat until convergence{

For every i,

$c^{(i)} := \text{argmin}_j \|x^{(i)} - \mu_j\|^2$ ← assign $x^{(i)}$ to the cluster with the closest centroid

For each j

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\}}$$ ← update centroid

}

Demo:http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

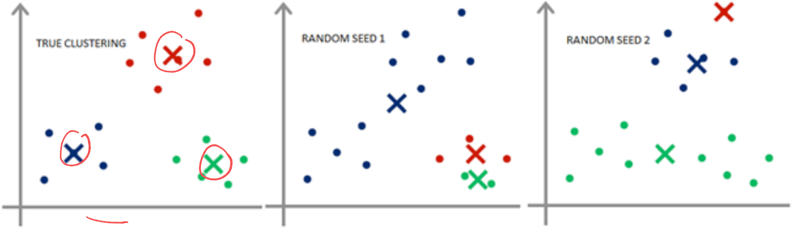
Lloyd, Stuart P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory

K-Means clustering discussion

- K-Means learns a k-dimensional *sparse* representation.
  i.e. $x^{(i)}$ is transformed into a "one-hot" vector $z^{(i)} \in \mathbb{R}^k$:

$$z_j^{(i)} = \begin{cases} 1 & \text{if } c^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

- Only converges to a local minimum: initialization matters!



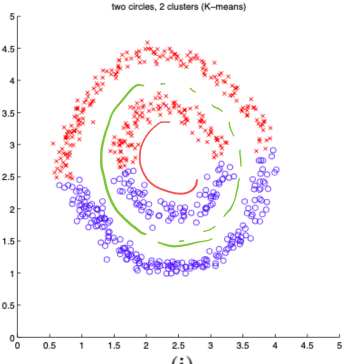
Practical considerations

- Replicate clustering trails and choose the result with the smallest WCSS
- How to initialize centroids μ_j's ?
  - Uniformly random sampling ☹
  - Distance-based sampling e.g. kmeans++ [Arthur & Vassilvitskii SODA 2007] ☺
- How to choose k?
  - Cross validation (later lecture)
  - G-Means [Hamerly & Elkan, NIPS 2004]
- How to improve k-means efficiency?
  - Elkan's algorithm [Elkan, ICML 2003]
  - Mini-batch k-means [D. Sculley, WWW 2010]

Spectral Graph Theory

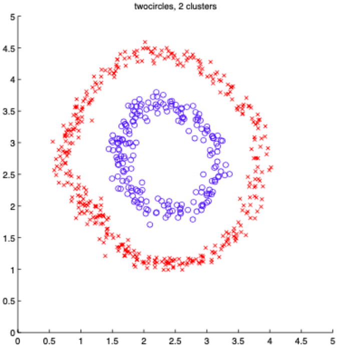
K-Means vs Spectral Clustering

K-Means



(i)

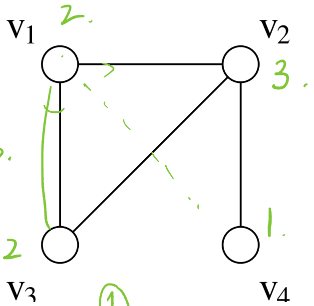
Spectral Clustering



(e)

[Shi & Malik 00; Ng, Jordan, Weiss NIPS 01]

Graph Terminologies



$$W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- An **undirect graph** $G = (V, E)$ consists of nodes $V = \{v_1, \ldots, v_n\}$ and edges $E = \{e_1, \ldots, e_m\}$
- Edge e_{ij} connects v_i and v_j if they are **adjacent** or neighbors.
- Adjacency matrix

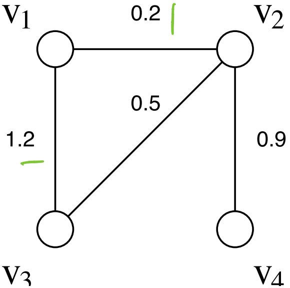
$$W_{ij} = \begin{cases} 1 & \text{if there is an edge } e_{ij} \\ 0 & \text{otherwise} \end{cases}$$

- **Degree** d_i of node v_i is the number of neighbors of v_i.

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Graph Terminologies

deg(v1) = 1.2 + 0.2 = 1.4



- **Weigthed undirected graph** $G = (V, E, W)$
- Edge weight $w_{ij} \in \mathbb{R}_{\geq 0}$ between v_i and v_j
  edge (v_i, v_j) *exists iff* $w_{ij} > 0$
- **Weighted adjacency matrix** $W = [w_{ij}]$
- Vertex degree $d_i = \sum_{j=1}^{n} w_{ij}$
- **Degree matrix** $D = diag(d_1, \ldots, d_n)$

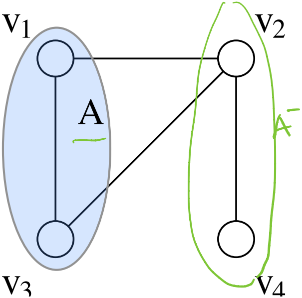
$$W = \begin{bmatrix} 0 & 0.2 & 1.2 & 0 \\ 0.2 & 0 & 0.5 & 0.9 \\ 1.2 & 0.5 & 0 & 0 \\ 0 & 0.9 & 0 & 0 \end{bmatrix}$$

$\mathbb{1}_n = \begin{bmatrix} 1.4 \\ 1.6 \\ 1.7 \\ 0.9 \end{bmatrix} \leftarrow \begin{matrix} d_1 \\ \vdots \\ d_n \end{matrix}$

$\begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{bmatrix}$

$W\mathbb{1}_n$ $\quad \mathbb{1}_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \Big\} n$

Graph Terminologies



- Given vertex subset $A \subset V$, let $\bar{A} = V \backslash A$ be the complement of A in the graph
- Subset indicator function $\mathbf{1}_A \in \mathbb{R}^n$:

$$\mathbf{1}_A\{i\} = \begin{cases} 1 & \text{if } v_i \in A \\ 0 & \text{if } v_i \notin A \end{cases}$$

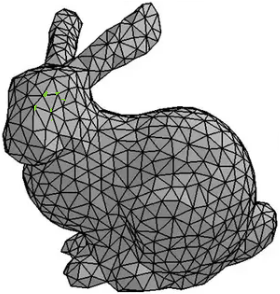
- Sets $A_1, \ldots, A_k$ form a **partition** of the graph if $A_i \cap A_j = \varnothing$ for all $i \neq j$ and $A_1 \cup \ldots \cup A_k = V$

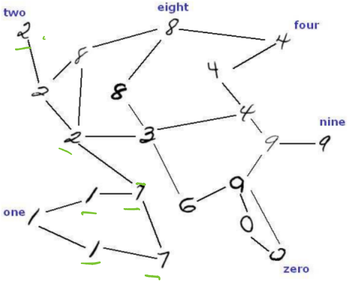
$V \backslash A$

K=2. for any $A \subset V$.
$A, \bar{A}$ forms a partition of V.

Represent data using a graph

Some data are naturally represented by a graph e.g. social networks, 3D mesh etc





Use graph to represent similarity in data

Clustering from a graph point of view

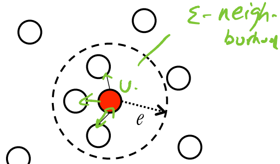
$s_{ij} = \|x^i - x^j\|^2$

- Given data points $x^{(1)}, \ldots, x^{(n)}$ and **similarity measure** $s_{ij} \geq 0$ for all $x^{(i)}, x^{(j)}$
- A typical **similarity graph** $G = (V, E)$ is
  - $v_i \leftrightarrow x^{(i)}$
  - v_i and v_j are connected if $s_{ij} \geq \delta$ for some threshold δ
- **Clustering**: Divide data into groups such that points in the same group are similar and points in different groups are dissimilar
- **Spectral Clustering (informal)**: *Find a partition of G such that edges between the same group have high weight and edges between different groups have very low weight.*

Building similarity graphs from data

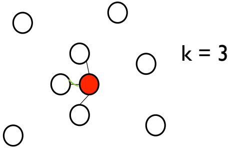
ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v



k-Nearest Neighbors

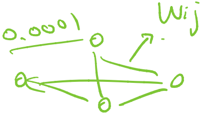
Add edges between v's k-nearest neighbors.



Fully connected graph

Often, Gaussian similarity is used

$$W_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

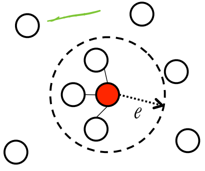


Building similarity graphs from data

ϵ-neighborhood

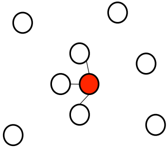
Add edges to all points inside a ball of radius f centered at v

Drawbacks: sensitive to ϵ, edge weights are on similar scale



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.



k = 3

Fully connected graph

Often, Gaussian similarity is used

$$W_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$

Building similarity graphs from data

ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v

Drawbacks: sensitive to e, edge weights are on similar scale

k-Nearest Neighbors

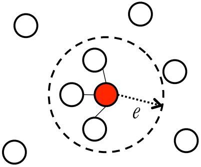
Add edges between v's k-nearest neighbors.

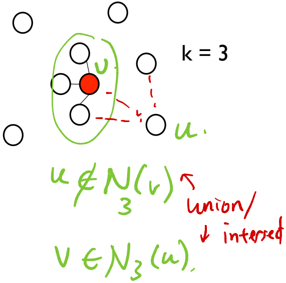
Drawbacks: may result in asymmetric and irregular graph

Fully connected graph

Often, Gaussian similarity is used

$$W_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{2\sigma^2}\right) \text{ for } i, j = 1, \ldots, m$$



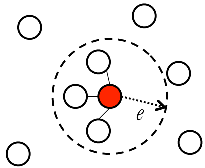


Building similarity graphs from data

ϵ-neighborhood

Add edges to all points inside a ball of radius f centered at v

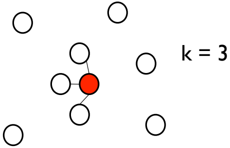
Drawbacks: sensitive to e, edge weights are on similar scale



k-Nearest Neighbors

Add edges between v's k-nearest neighbors.

Drawbacks: may result in asymmetric and irregular graph



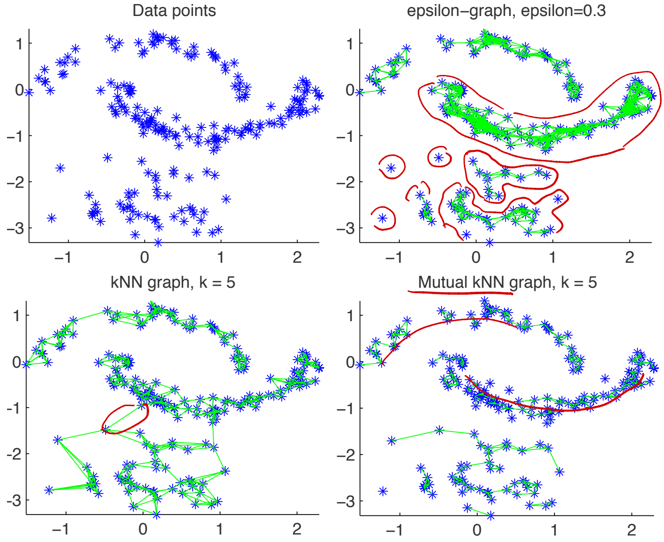
Fully connected graph

Often, Gaussian similarity is used

$$W_{i,j} = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{2\sigma^2}\right) \text{ for } i,j = 1, \ldots, m$$

Drawbacks: W is not sparse

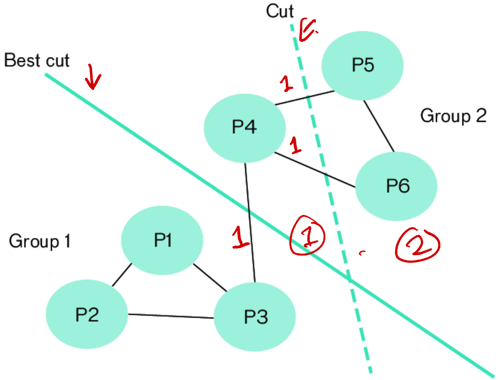
Similarity graphs examples



Spectral Clustering as Graph Partitioning

Find a partition of the graph such that

- Edges between groups have a low weight
- Edges within each group have a high weight



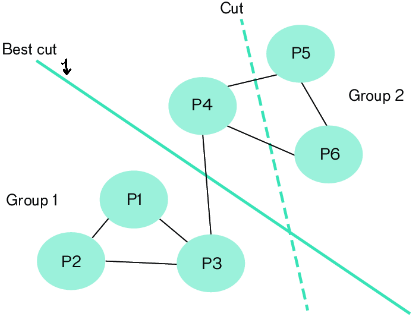
Graph Cut Formulation

Case $k = 2$:

- Given partition $A, \bar{A}$, define a cut as the total weight of edges weights between groups:

$$cut(A, \bar{A}) := \sum_{i \in A, j \in \bar{A}} w_{ij}$$

- Example: $cut(\{p_1, p_2, p_3\}, \{p_4, p_5, p_6\}) = 1$, $cut(\{p_1, p_2, p_3, p_4\}, \{p_5, p_6\}) = 2$



Graph Cut Formulations

Case $k > 2$:

- Given partition $A_1, \ldots, A_k$, define a cut as the total weight of edges weights between groups:

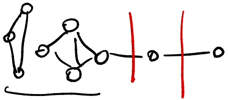
$$cut(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} cut(A_i, \bar{A}_i)$$

Graph Cut Formulations

Case $k > 2$:

- Given partition $A_1, \ldots, A_k$, define a cut as the total weight of edges weights between groups:

$$cut(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} cut(A_i, \bar{A}_i)$$



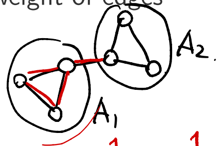
Minimizing cut directly tends to unbalanced partitions. Alternative solutions:

Graph Cut Formulations

Case $k > 2$:

- Given partition $A_1, \ldots, A_k$, define a cut as the total weight of edges weights between groups:

$$cut(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} cut(A_i, \bar{A}_i)$$



Minimizing cut directly tends to unbalanced partitions. Alternative solutions:

$$RatioCut(A_1, A_2) = \frac{1}{2}\left(\frac{cut(A_1, A_2)}{|A_1|} + \frac{cut(A_2, A_1)}{|A_2|}\right)$$

RatioCut and NCut

Find a k-way partition of graph G ($A_i \cup \ldots \cup A_k = V, A_i \cap A_j = \emptyset$) that minimizes:

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|}$$

Normalized

$$NCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}, vol(A_i) = \sum_{i \in A, j \in V} w_{ij}$$

$$NCut(A_1, A_2) = \frac{1}{2}\left(\frac{1}{vol(A_1)} + \frac{1}{vol(A_2)}\right)$$

Graph Cut Formulations

Case $k > 2$:

- Given partition $A_1, \ldots, A_k$, define a cut as the total weight of edges weights between groups:

$$cut(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} cut(A_i, \bar{A}_i)$$

Minimizing cut directly tends to unbalanced partitions. Alternative solutions:

RatioCut and NCut

Find a k-way partition of graph G ($A_i \cup \ldots \cup A_k = V, A_i \cap A_j = \varnothing$) that minimizes:

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|}$$

$$NCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}, vol(A_i) = \sum_{i \in A, j \in V} w_{ij}$$

Both RatioCut and NormalizeCut can be ***approximated*** *by spectral method.*