Until now:

Supervised learning:

$$\vec{x}^{(i)} = (x_1^{(i)}, \dots, x_p^{(i)}) \; ; \; y^{(i)} \qquad i = 1, \dots, N$$

given $\vec{x}^{\,in}$ $\qquad f(\vec{x}^{\,in}) = \hat{y}$

inferred from $(\vec{x}^{(i)}, y^{(i)})$ $i = 1, \dots, N$

Switch now to

Unsupervised learning:

- There is no 'observable' y
- There is no ground truth.
- There are no examples.
- There is no training.

We have a series of samples:

$$\{\vec{x}^{(i)}\}_{i=1}^{N}$$

Data $\vec{x}^{(i)} = (x_1^{(i)}, \dots, x_p^{(i)})$ $i = 1, \dots, n$

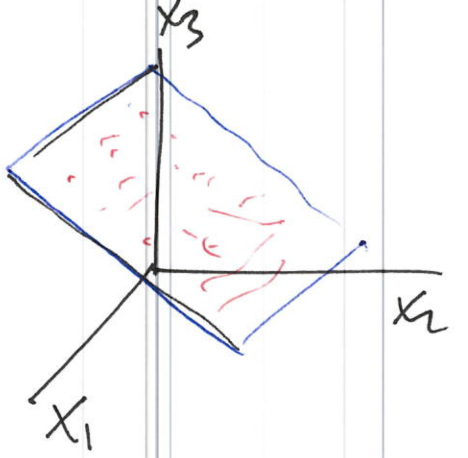
of different types $\begin{cases} \vec{x}^{(i)} \in \mathbb{R}^p \\ \vec{x}^{(i)} \in \{c_1, \dots, c_k\}^p \end{cases}$

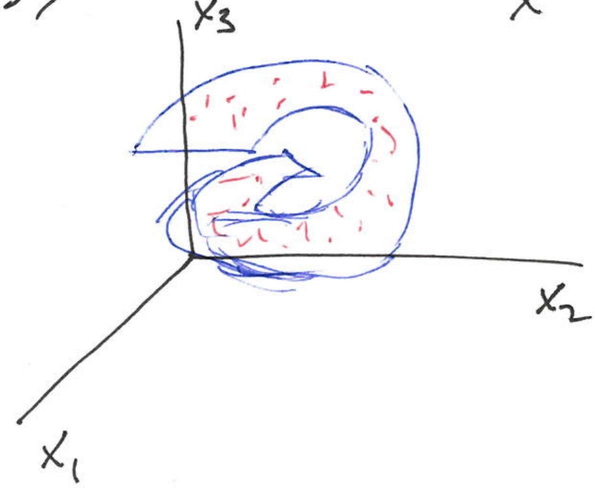
Types of tasks in unsupervised learning.

(1) Clustering : find groups of points that are more similar to each other within the group than to points outside the group.

(2) Dimensionality reduction :

$p >>$ $\vec{x}^{(i)} \in \mathbb{R}^3$





Visualization .

Clustering:

Key ingredient: Similarity vs Dissimilarity

$\updownarrow$

"Distance".

(Not always a true metric)

$$D(\vec{x}^{(i)}, \vec{x}^{(j)})$$

Typical: A If $\vec{x} \in \mathbb{R}^p$

$$D(\vec{x}^{(i)}, \vec{x}^{(j)}) = \| \vec{x}^{(i)} - \vec{x}^{(j)} \|^2$$

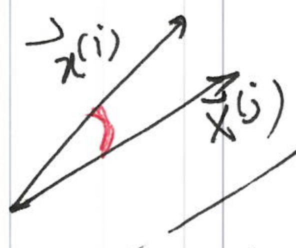
(i) dissimilarities:

$$= | \vec{x}^{(i)} - \vec{x}^{(j)} |$$

Distances.

(ii) similarities

cosine similarity. $S(\vec{x}^{(i)}, \vec{x}^{(j)}) = \dfrac{\vec{x}^{(i)} \cdot \vec{x}^{(j)}}{\|\vec{x}^{(i)}\| \, \|\vec{x}^{(j)}\|}$



statistical similarity:

$$\rho(\vec{x}^{(i)}, \vec{x}^{(j)}) = \frac{\mathrm{cov}(\vec{x}^{(i)}, \vec{x}^{(j)})}{\sqrt{\mathrm{cov}(\vec{x}^{(i)}, \vec{x}^{(i)}) \cdot \mathrm{cov}(\vec{x}^{(j)}, \vec{x}^{(j)})}}$$

B. If $\vec{x}$ are categorical:

x belonging to K classes

$$\begin{array}{c|ccc} & C1 & C2 & C3 \\ \hline C1 & 0 & D_{12} & D_{13} \\ C2 & & 0 & D_{23} \\ C3 & & & 0 \end{array} \qquad K = 3$$

Distance matrix: is based on distance assigned to feature classes.

C. Ordinal variables: ranked.

$$A \to B \to C \to D \qquad K = 4$$

$$i = \quad 1 \quad 2 \quad 3 \quad 4$$

$$j = \frac{i - \frac{1}{2}}{K} \begin{bmatrix} \frac{1}{8} & \frac{3}{8} & \frac{5}{8} & \frac{7}{8} \end{bmatrix}$$

~~Heuristics for clustering:~~

~~1. K-means~~

$$\cancel{\{\vec{x}^{(i)}\}_{i=1}^{N} \qquad \vec{x}^{(i)} \in \mathbb{R}^{p}}$$

$$\cancel{D(\vec{x}^{(i)}, \vec{x}^{(j)}) = \|\vec{x}^{(i)} - \vec{x}^{(j)}\|^{2}}$$

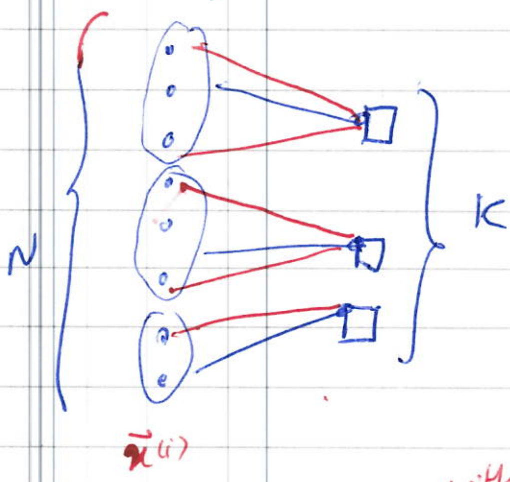
Unsupervised learning:

(1) Clustering problem

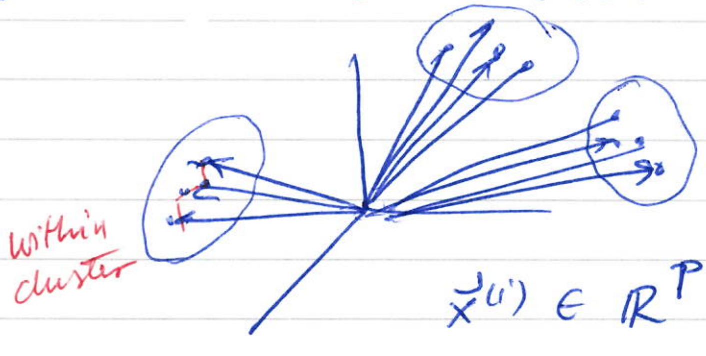
General statement: Given $\{\vec{x}^{(i)}\}_{i=1}^{N}$ our N samples and a dissimilarity ("distance") $D(\vec{x}^{(i)}, \vec{x}^{(j)})$,

the objective is to find a partition of the N samples into K ~~classes~~ clusters such that the dissimilarity is smaller within cluster ~~and~~ than across clusters.

In pictures: we have a mapping of the N samples to K clusters



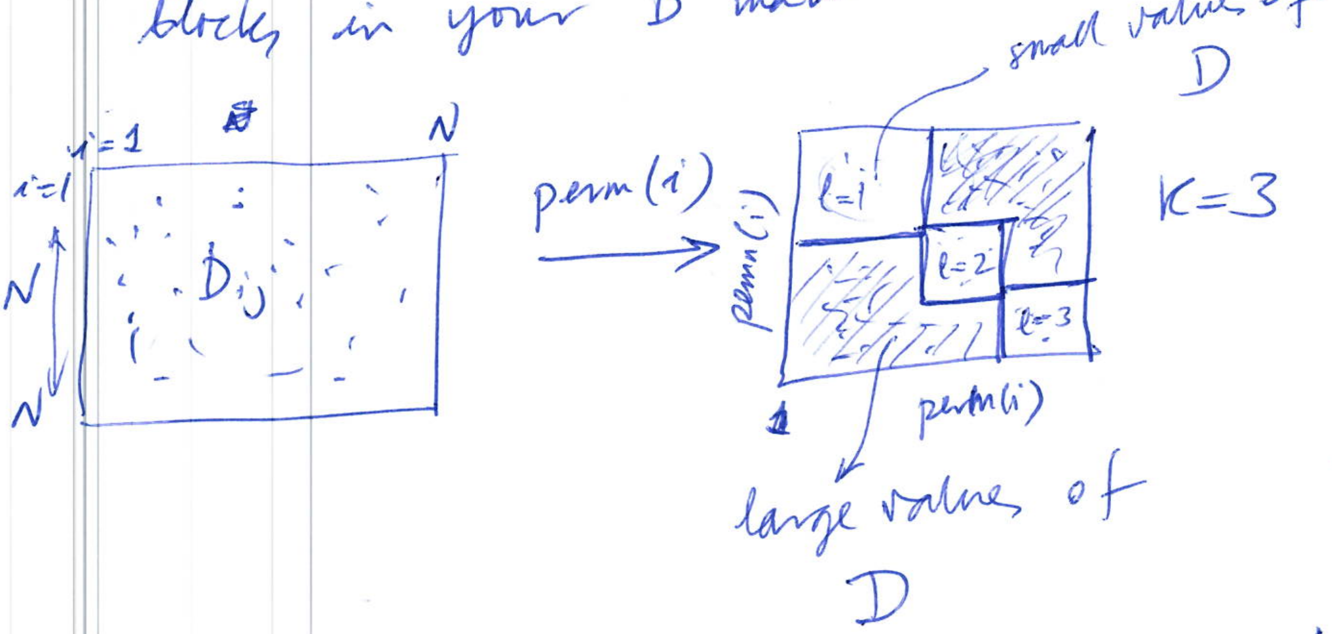
Geometrically: If the samples are continuous variables and D is a distance:



such that
$$\left\{ \begin{array}{l} \sum_{\text{within cluster}} D_{ij} \ll \\ \sum_{\text{across clusters}} D_{ij} \gg \end{array} \right\}$$

Another look at this problem:

We have a $D_{N \times N}$ matrix and we want to reorder the rows and columns (i.e. find the permutation) that makes blocks in your D matrix:



This will mean that for this clustering: C

Within cluster dissimilarity

$$W(C) = \sum_{l=1}^{K} \sum_{i,j \in C_l} d(\vec{x}^{(i)}, \vec{x}^{(j)}) \frac{1}{2}$$

is small

and

~~Between~~ Across cluster dissimilarity

$$B(C) = \underbrace{\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(\vec{x}^{(i)}, \vec{x}^{(j)})}_{\text{Total dissimilarity}} - W(C)$$

is large

Finding the best H (or the best clustering) is

a <u>combinatorial optimisation problem</u>:

- These are usually NP-hard, all possible arrangements have to be tried to find the global optimum.
- But the number of clusterings to try ~~[illegible]~~ explodes:

For N samples, K clusters.

$$S(N,K) = \frac{1}{K!} \sum_{l=1}^{K} (-1)^{K-l} \binom{K}{l} l^N$$

$$S(10,4) = 34,105$$

$$S(19,4) = 10^{10}$$

$$\vdots$$

Impractical to do this enumeration.

Therefore, we need to come up with heuristics to optimize.

K-means : What is the K-means heuristic?

We want to find an assignment between the N samples and K ~~means~~ clusters

First: ~~1)~~ Decide on K

Task: Find an assignment matrix:

$$
H_{N \times K} = \begin{array}{cc}
\begin{bmatrix}
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 1 & 0 & & \cdots & 0 \\
1 & 0 & 0 & & \cdots & 0 \\
 & & \vdots & & & \\
0 & \cdots & & \cdots & 0 & 1
\end{bmatrix} & \begin{array}{c} i=1 \\ \\ \\ \\ i=N \end{array} \\
c_1 \qquad\qquad\qquad\qquad c_K &
\end{array}
$$

This summarizes a hard assignment that is exhaustive.

$$\{C_\ell\}_{\ell=1}^{K} \qquad \text{(i)} \quad C_\ell \cap C_{\ell'} = \emptyset \quad \ell \neq \ell'$$

$$\text{(ii)} \quad \bigcup_{\ell=1}^{K} C_\ell = \{\vec{x}^{(i)}\}_{i=1}^{N}$$

As stated above,
This is a combinatorial problem : check all the permutations to see which one is best

K-means algorithm:

Given $\{\vec{x}^{(i)}\} \in \mathbb{R}^p$

Compute $D(\vec{x}^{(i)}, \vec{x}^{(j)}) = \| \vec{x}^{(i)} - \vec{x}^{(j)} \|^2$ $\boxed{\text{Distance matrix}}$

For a given clustering: $\{C_\ell\}_{\ell=1}^{K} \equiv C$

we have the within cluster normalized distance

$$W(C) = \frac{1}{2} \sum_{\ell=1}^{K} \frac{1}{|C_\ell|} \sum_{i,j \in C_\ell} \| \vec{x}^{(i)} - \vec{x}^{(j)} \|^2$$

~~equivalent to finding $H_{N \times K}$~~

where $|C_\ell|$ is the cardinality of C_ℓ

$W(C)$ can be rewritten in terms of H:

$$\mathrm{Tr}\left[\frac{1}{2} (H^T H)^{-1} \left[\underset{K\times N}{H^T} D_{N\times N} H_{N\times K} \right] \right] = W(C)$$

$$\mathrm{diag}(C_\ell) = H^T H_{K\times K} = \begin{bmatrix} \square & 0 & 0 \\ 0 & \square & 0 \\ 0 & 0 & \blacksquare \end{bmatrix} \qquad \begin{bmatrix} \boxdot & \blacksquare & \blacksquare \\ \blacksquare & \boxdot & \blacksquare \\ \blacksquare & \blacksquare & \boxdot \end{bmatrix}_{K\times K} \qquad K=3$$

$$|C_3| \qquad \sum_{i,j \in C_3} \| \vec{x}^{(i)} - \vec{x}^{(j)} \|^2$$

$$T = \frac{1}{2} \underset{1\times N}{\vec{1}^T} \underset{N\times N}{D} \underset{N\times 1}{\vec{1}} = \frac{1}{2} \underset{1\times K}{1^T} \underbrace{[H^T D H]}_{K\times K} \underset{K\times 1}{\vec{1}}$$

$$\vec{1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{N\times 1}$$

$$\underset{N\times K}{H} \vec{1}_{K\times 1} = \vec{1}_{N\times 1}$$

Objective : Minimise $W(C)$

$$\Downarrow$$

maximise $T - W(C)$ = between cluster distance.

$$\min_H \mathrm{Tr}\left[\frac{1}{2}(H^T H)^{-1}(H^T D H)\right]$$

$D_{N\times N}(\vec{x}^{(i)})$ is the distance matrix

Algorithm :

- Step 0 : Assign at random ~~the~~ every sample to a cluster.

$\boxed{\text{Initialisation}}$

- Step 1 : compute the centroid of each of the K clusters

$$\vec{m}_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \vec{x}^{(i)}$$

- Step 2 : Reassign each $\vec{x}^{(i)}$ to the closest centroid.

$\forall i$, ~~Find~~ Consider $l_i^{(t)}$

Evaluate

$$l_i^{(t+1)} = \arg\min_{l} \|\vec{x}^{(i)} - \vec{m}_l\|^2$$

Iterate step 1 and 2
Until convergence:

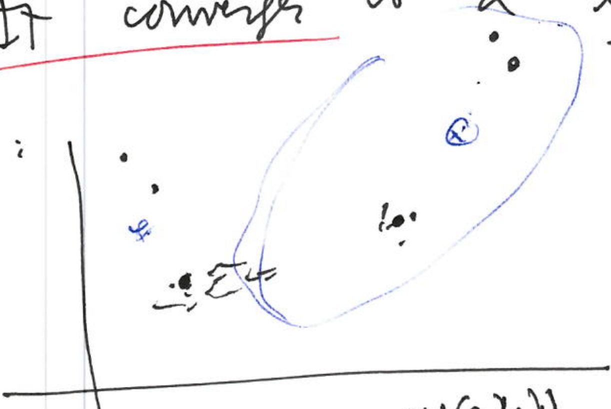
i.e., $W(C)$ does not improve much.
or assignments are not changing.

<u>Heuristic</u>: Gradient method
because at every step $W(C)$
is decreased.

It converges to a <u>local optimum</u>.

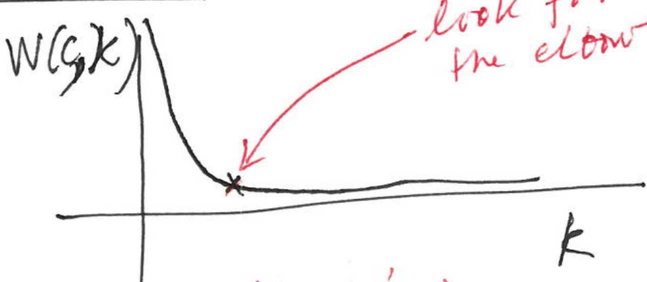
Comments:

(1) Outliers:



Instead: K-medoids.

(2) Try different K's:
How to choose K



or the 'kink' in
$W(C^*_{K+1}, K+1) - W(C^*_K, K)$