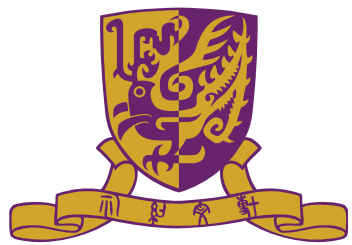
DDA5001 Machine Learning

Unsupervised Learning: Dimensionality Reduction (PCA) & Clustering (k-means)

Xiao Li

School of Data Science
The Chinese University of Hong Kong, Shenzhen



Recap: PCA

PCA modeling:

$$\boxed{\boldsymbol{x}_i = \boldsymbol{A}\boldsymbol{\theta}_i + \boldsymbol{\mu}}$$

Interpretation: Find the closest k-dimensional data point $\tilde{x}_i = \boldsymbol{A}\boldsymbol{\theta}_i = \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i$ to $\boldsymbol{x}_i$ (projecting $\boldsymbol{x}_i$ onto the k-dimensional subspace spanned by the columns of $\boldsymbol{A}$).

Process of applying PCA:

- Remove the mean: $\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$, where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$.
- Compute SVD of the data matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ (recommended) or eigen decomposition of the covariance matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$.
- Basis $\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$, principle component $\boldsymbol{\theta}_i = \boldsymbol{A}^\top \boldsymbol{x}_i$, and k-dimensional sample is $\tilde{\boldsymbol{x}}_i = \boldsymbol{A}\boldsymbol{\theta}_i = \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i$.

PCA — Continued

Clustering: k-means

PCA From the Matrix Factorization Perspective

Recall the (first form) PCA (we assume without loss of generality that $\boldsymbol{\mu} = 0$)

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \{\boldsymbol{\theta}_i\}}{\text{minimize}} \ \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{\theta}_i\|_2^2.$$

It can be written in a matrix form:

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times k}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{k \times n}$, and $\|\cdot\|_F$ is the Frobenius norm.

- The above problem is also called low-rank matrix factorization.
- Interpretation: Factorize $\boldsymbol{X}$ into two factors' multiplication, where the latter is a low-rank matrix.

PCA From the Matrix Factorization Perspective

Low-rank matrix factorization

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I},\boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

- Calculate the SVD of $\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$.

Solution from PCA

- One optimal solution to the above (low-rank) matrix factorization problem is given by

$$\boldsymbol{A}=[\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k], \quad \boldsymbol{\Theta}=[\sigma_1\boldsymbol{v}_1,\ldots,\sigma_k\boldsymbol{v}_k]^\top$$

- It has infinitely many equivalent optimal solutions.

It is a closed-form solution to a nonconvex optimization problem.

More General Matrix Factorization

We can also remove the orthogonal constraint on $\boldsymbol{A}$ to allow more flexibility

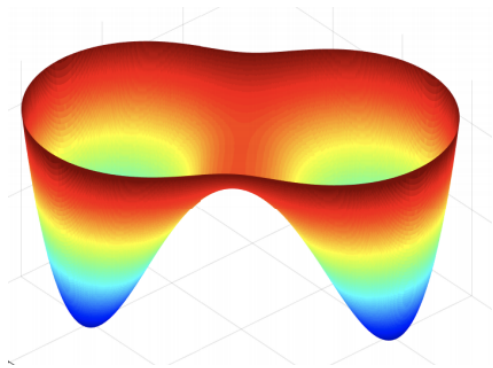
$$\underset{\boldsymbol{A}\in\mathbb{R}^{d\times k},\boldsymbol{\Theta}\in\mathbb{R}^{k\times n}}{\text{minimize}} \quad \|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{\Theta}\|_F^2.$$

One optimal solution to the above (low-rank) matrix factorization problem is given by

$$\boldsymbol{A}=[\sqrt{\sigma_1}\boldsymbol{u}_1,\ldots,\sqrt{\sigma_k}\boldsymbol{u}_k], \quad \boldsymbol{\Theta}=[\sqrt{\sigma}_1\boldsymbol{v}_1,\ldots,\sqrt{\sigma}_k\boldsymbol{v}_k]^\top,$$

and it has infinitely many equivalent optimal solution.

- A nonconvex optimization problem



- Fortunately, closed-form solution exists.

LoRA: Low-rank Adaptation

In the post-training stage of large models (like large language models), we often need to learn an incremental to the learned model to incorporate new knowledge. That is

$$\underset{\Delta\boldsymbol{\Theta}\in\mathbb{R}^{m\times n}}{\text{minimize}} \ \mathcal{L}(\widehat{\boldsymbol{\Theta}}+\Delta\boldsymbol{\Theta})$$

Low-rank Adaptation (LoRA) uses the simple idea of PCA. It does not aim to learn the full $\Delta\boldsymbol{\Theta}$, it instead learns a PCA decomposition of $\Delta\boldsymbol{\Theta}$.

$$\underset{\boldsymbol{A}\in\mathbb{R}^{m\times r},\boldsymbol{B}\in\mathbb{R}^{r\times n}}{\text{minimize}} \ \mathcal{L}(\widehat{\boldsymbol{\Theta}}+\boldsymbol{AB})$$

where $r \ll \min\{m,n\}$.

- LoRA approach can save computation memory.
- It is now widely utilized in large language models.
- We will use LoRA in our final project.

Apply PCA to Real Image Dataset

The ORL Database of Faces

- 40 persons.
- Each has 10 distinct face images.
- Each image is of size 92×112.
- The images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses).

Form the Data Matrix

92x112

vectorize

$$\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{x}_i[1] \\ \vdots \\ \boldsymbol{x}_i[92 \times 112] \end{bmatrix} \in \mathbb{R}^{10304}$$

- Do the same thing for all the face images, we get

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n},$$

where $d = 10304$, $n = 10 \times 40 = 400$.

Apply PCA to $\boldsymbol{X}$ with $k = 40$

Perform PCA, i.e., solving

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

We get the extracted features (i,e., $\boldsymbol{A} \in \mathbb{R}^{d \times k}$)

where we resize each column of $\boldsymbol{A}$ (of dimension $d = 10304$) to a 92×112 image and show it.

Reconstructing the Face Image

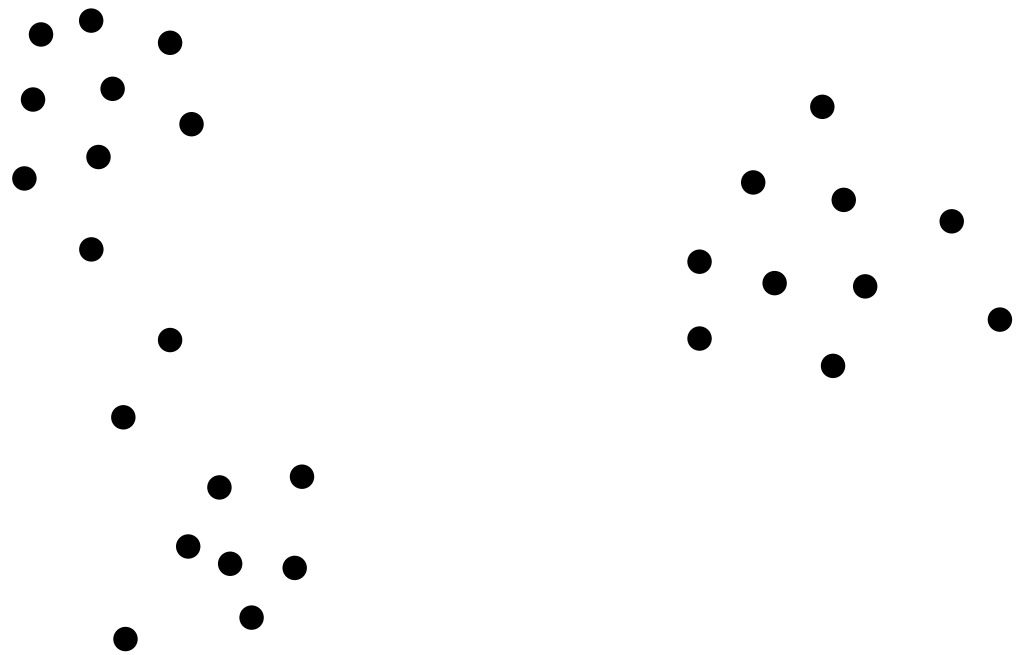
$$\boldsymbol{x}_i \approx \boldsymbol{A}\boldsymbol{\theta}_i$$

- Each face image $\boldsymbol{x}_i$ can be interpreted as a linear combination of the columns in $\boldsymbol{A}$.
- The importance of each feature is implied in $\boldsymbol{\theta}_i$.
- $\boldsymbol{x}_i \in \mathbb{R}^d$ ($d = 10304$ here) has been nicely represented by a k-dimensional vector $\boldsymbol{A}\boldsymbol{\theta}_i$ ($k = 40$ here) — Dimensionality reduction.

PCA — Continued

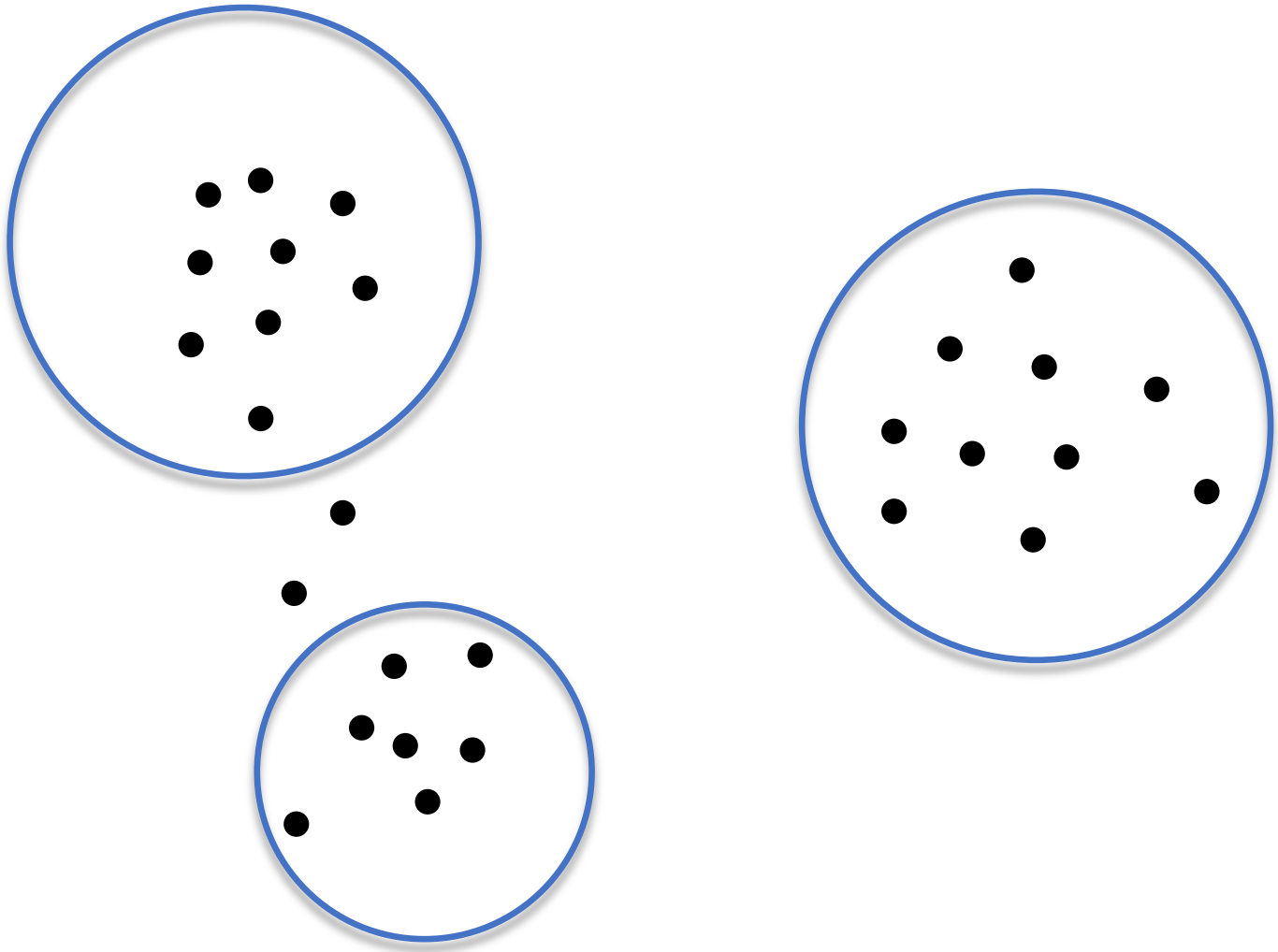
Clustering: k-means

The Unlabeled Samples



- Unlabeled samples.
- **Task:** Assign data to several groups / clusters.

Clustering



- This task is very different from classification.

Clustering: Formal Definition

- Given unlabeled data samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$.

Clustering: Assign the unlabeled data to disjoint subsets called clusters. The principle is that points in the same cluster are more similar to each other than points in different clusters.

A clustering algorithm can be represented by a cluster mapping, which is a function

$$\mathcal{C} : \{1, \ldots, n\} \to \{1, \ldots, k\}$$

where k is the number of clusters.

Intuition for performing clustering:

- Define certain distance functions to measure similarity.
- Compute pair-wise distances and put a threshold to assign clusters.

Clustering vs. Classification

- Clustering: Unsupervised, unlabeled data, assign data into clusters without recognizing them. For example, we just cluster dog and cat without recognizing whether x is dog or cat.

- Classification: Supervised, labeled data, recognize the given data, i.e., tell whether x is dog or cat.

k-means Clustering

k-means

One natural method for clustering is k-means, which is to cluster by determining a center of each group.

k-means criterion: Choosing $\mathcal{C}$ to minimize

$$\min_{\mathcal{C}} W(\mathcal{C}) = \sum_{j=1}^{k} \sum_{i:\mathcal{C}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2$$

where

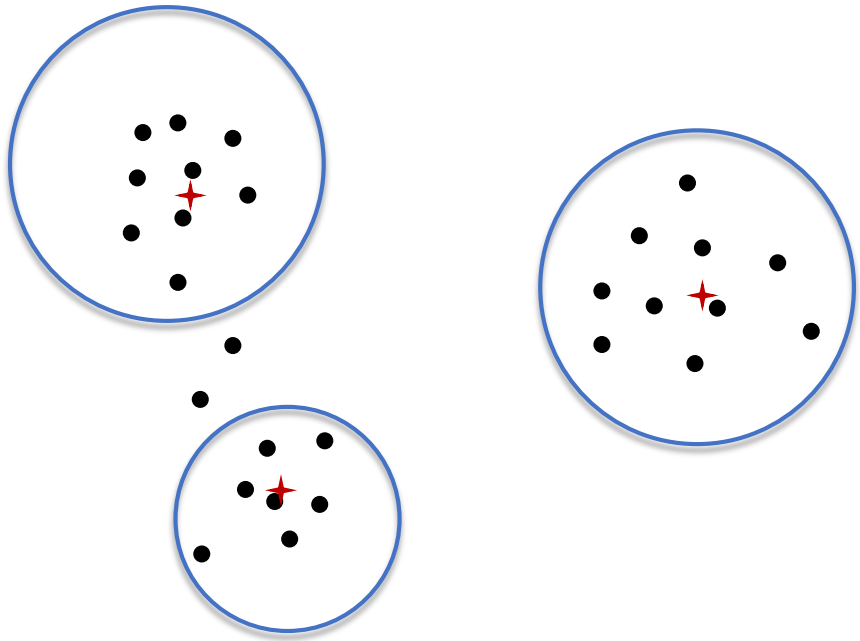
$$\boldsymbol{\mu}_j := \frac{1}{n_j} \sum_{i:\mathcal{C}(i)=j} \boldsymbol{x}_i \qquad n_j = |\{i : \mathcal{C}(i) = j\}|$$

- In k-means, the number of clusters k is assumed to be known as a prior.

Interpretation:

- k-means: Find k means of the k clusters.
- Calculate the squared Euclidean distance (ℓ_2-norm) to measure similarity and dissimilarity.
- Clustering by finding the assignment that provides small within-cluster distance, i.e., minimizing $W(\mathcal{C})$.

k-means: Interpretation



- Intuition: Find k balls (due to Euclidean distance) to enclose each cluster.

How many possible clusters?

How many cluster maps $\mathcal{C}$ do we need to consider?

$$S(n,k) = \#\text{ of possible clusterings of } n \text{ objects into } k \text{ clusters}$$

$$= \frac{1}{k!}\sum_{j=1}^{k}(-1)^{k-j}C_k^j j^n$$

where $C_k^j = \frac{k(k-1)\cdots(k-j+1)}{j(j-1)\cdots 1}$.

Examples:

- $S(10,4) = 34105$.
- $S(19,4) \approx 10^{10}$.

Thus, solving exactly a k-means problem is almost impossible.

Minimizing The k-means Criterion

- No existing (provably) efficient method for solving it.
- It can be solved exactly (by enumerating) in time $\mathcal{O}\left(n^{dk+1}\log n\right)$, which is exponential time.
- This is completely impractical unless both d and k are extremely small.
  - E.g., $d = 2$, $k = 3$ already results in $\mathcal{O}(n^7 \log n)$ solving time.

Minimizing the k-means criterion is a combinatorial optimization problem, which is NP-hard.

In practice, we resort to iterative, suboptimal algorithms—k-means algorithm.

Reformulation of k-means

Recall k-means is to

$$\mathcal{C}^\star = \operatorname*{argmin}_{\mathcal{C}} \sum_{j=1}^{k} \sum_{i:\mathcal{C}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2.$$

For fixed $\mathcal{C}$

$$\boldsymbol{\mu}_j = \operatorname*{argmin}_{\boldsymbol{m}} \sum_{i:\mathcal{C}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{m}\|_2^2.$$

Thus, we can write k-means equivalently

$$\mathcal{C}^\star = \operatorname*{argmin}_{\mathcal{C},\{\boldsymbol{m}_j\}_{j=1}^k} \sum_{j=1}^{k} \sum_{i:\mathcal{C}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2.$$

- We now have two types of decision variables: 1) The clustering map. 2) The means/centers/centroids $\{\boldsymbol{m}_j\}$.
- This reformulation gives us an alternating minimization framework.

k-means Algorithm

$$\mathcal{C}^\star = \underset{\mathcal{C},\{\boldsymbol{m}_j\}_{j=1}^k}{\operatorname{argmin}} \left\{ W(\mathcal{C}, \{\boldsymbol{m}_j\}_{j=1}^k) := \sum_{j=1}^{k} \sum_{i:\mathcal{C}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2 \right\}.$$

- In the formulation above, we have two types of decision variables.
- Recall that optimization algorithm almost always follows the idea: Dividing the original problem into a set of simpler subproblems.
- Solving the k-means problem for both the clustering map $\mathcal{C}$ and the centers $\{\boldsymbol{m}_j\}$ is hard. Can we just solve one of them at one time? It could be much simpler than solving the two variables simultaneously.

Alternating minimization algorithm:

- Fix $\mathcal{C}$, update $\{\boldsymbol{m}_j\}_{j=1}^k$ to minimize $W(\mathcal{C}, \{\boldsymbol{m}_j\}_{j=1}^k)$.
- Fix $\{\boldsymbol{m}_j\}_{j=1}^k$, update $\mathcal{C}$ to minimize $W(\mathcal{C}, \{\boldsymbol{m}_j\}_{j=1}^k)$.

k-means Algorithm

Key advantage: Each subproblem can be solved very easily.

Use t to denote the iteration number.

k-means algorithm

- Fix $\mathcal{C} = \mathcal{C}^{(t)}$, the update of $\{\boldsymbol{m}_j\}_{j=1}^k$ is

$$\boldsymbol{m}_j^{(t+1)} = \frac{1}{n_j} \sum_{i:\mathcal{C}^{(t)}(i)=j} \boldsymbol{x}_i, \qquad j = 1, \ldots, k.$$

- Fix $\{\boldsymbol{m}_j\}_{j=1}^k = \{\boldsymbol{m}_j^{(t+1)}\}_{j=1}^k$, the update of $\mathcal{C}$ is

$$\mathcal{C}^{(t+1)}(i) = \underset{1 \le j \le k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{m}_j^{(t+1)}\|_2^2, \qquad i = 1, \ldots, n.$$

- Iterate the above two steps until convergence.

In the update of $\mathcal{C}^{(t+1)}(i)$, each data sample $\boldsymbol{x}_i$ is assigned to exactly one cluster, even though it can assigned to more than one clusters.

See `https://www.youtube.com/watch?v=5I3Ei69I40s` for an illustration of the process.

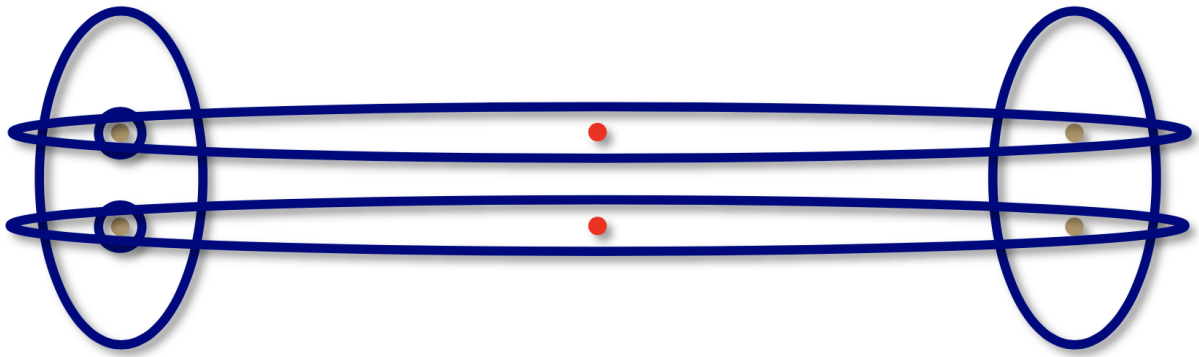
k-means Algorithm: Properties

- Since each step optimizes $W(\mathcal{C})$ and there only exists finite number of partitioning, so the algorithm usually converges to a (local) optimum in finite number of iterations.
- No guarantee for convergence to global optimum.
- Thus, initialization can be important.

Initialization

The algorithm is typically initialized by setting each $\boldsymbol{m}_j$ to be a random point in the dataset, i.e., random initialization.

However, depending on the initialization, the algorithm can get stuck in a local minimum.

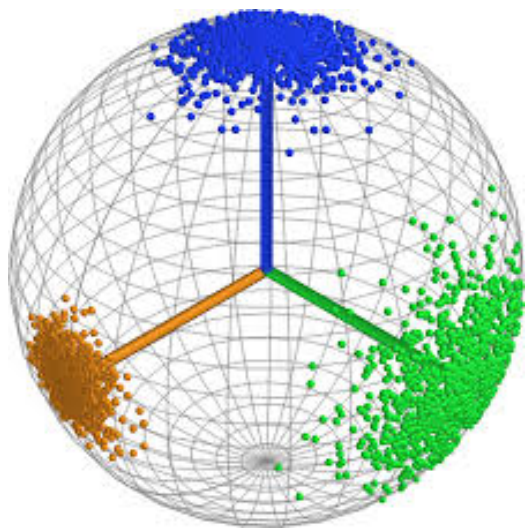


One can avoid this by:

- Repeating for several random initializations, then check which $W(\mathcal{C})$ is the smallest (very common choice in practice).
- Initialize by sequentially selecting random points in the dataset, but with a probability depending on how far the point is from the already selected $\boldsymbol{m}_j$. This technique leads to the k-means $++$.

Remarks

- k-means algorithm was originally developed at Bell Labs as an approach to vector quantization in signal processing.
- If we replace the squared ℓ_2-norm with the ℓ_1-norm distance in the definition of $W(\mathcal{C})$:
  - The center of each region is actually calculated via the median.
  - Results in k-medians clustering, which can be more robust.
- If we change the Euclidean squared ℓ_2-norm distance to the cosine similarity, we get the spherical k-means.

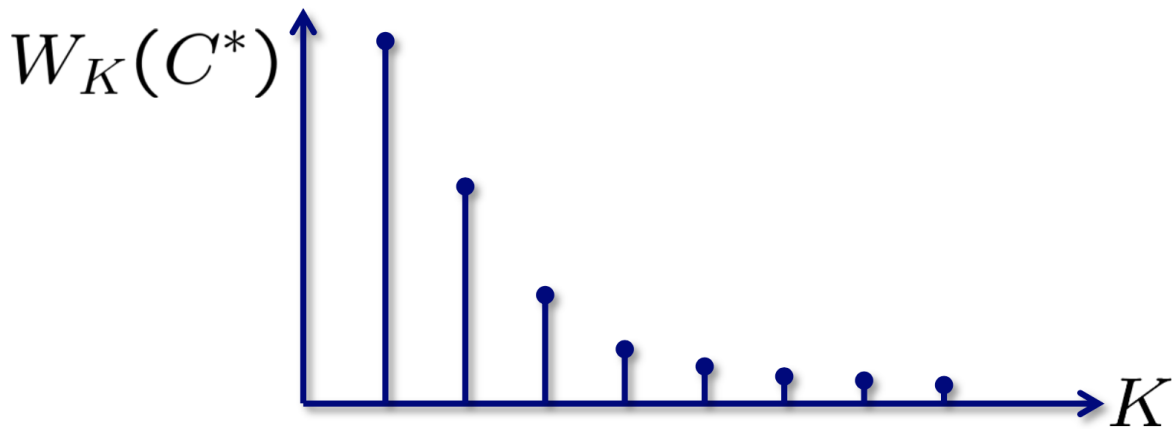


Model Selection for k-means

What is model selection used for? Selecting hyper-parameters.

Sometimes in practice, k is not known as a prior, then it becomes a hyper-parameter.

Let $W_k(C^\star)$ be the optimal loss based on $\#$ of clusters k.



Suppose the "correct" number of clusters is $k^\star$, we expect

- for $k < k^\star$, $W_k(C^\star) - W_{k-1}(C^\star)$ will be large.
- for $k > k^\star$, $W_k(C^\star) - W_{k-1}(C^\star)$ will be small.

This suggests choosing k to be near the "knee" of the curve.

Matrix Factorization Perspective for k-means

k-means can also be viewed as a special matrix factorization:

$$\boldsymbol{X} \approx \boldsymbol{B}\boldsymbol{C},$$

where

$$\boldsymbol{B} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k]$$

and $\boldsymbol{C}$ has exactly one "1" per column, i.e., unit vector.

- Interpretation: Assign each sample, column of $\boldsymbol{X}$, to one cluster, represented by the "1"" element in the corresponding column in $\boldsymbol{C}$.
- Learning problem:

$$\min_{\boldsymbol{B},\boldsymbol{C}} \|\boldsymbol{X}-\boldsymbol{B}\boldsymbol{C}\|_F^2, \quad \text{subject to} \quad \boldsymbol{C} \text{ has exactly one "1" per column.}$$

$\rightsquigarrow$ This is the end of unsupervised learning