Introduction to Machine Learning
Lecture 4 - PCA and K-Means

1 The Covariance Matrix

1.1 One-dimensional random variable

Let $X \in \mathbb{R}$ be a one-dimensional random variable and let $\{x_i\}_{i=1}^N$ be N i.i.d realizations from X.

- The first moment of X is given by:

$$\mu_x = \mathbb{E}[X] \approx \frac{1}{N}\sum_{i=1}^{N} x_i$$

- The variance of X is given by:

$$\sigma_x^2 = \mathbb{E}\left[(X-\mu_x)^2\right] \approx \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_x)^2$$

1.2 High-dimensional random vector

Let $X \in \mathbb{R}^d$ be a high-dimensional random variable and let $\{\boldsymbol{x}_i\}_{i=1}^N$ be N i.i.d realizations from X.

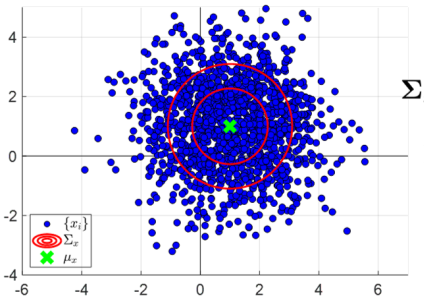
- The first moment of X is given by:

$$\boldsymbol{\mu}_x = \mathbb{E}[X] \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$$

- The covariance matrix of X is given by:

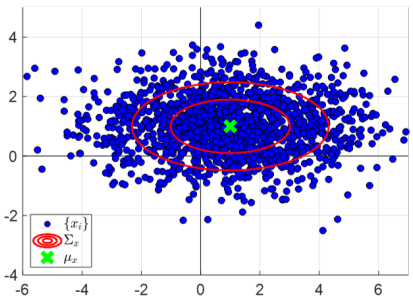
$$\boldsymbol{\Sigma}_x = \mathbb{E}\left[(X-\boldsymbol{\mu}_x)(X-\boldsymbol{\mu}_x)^T\right] \approx \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu}_x)(\boldsymbol{x}_i-\boldsymbol{\mu}_x)^T$$

The covariance matrix is the generalization of the one-dimensional variance.



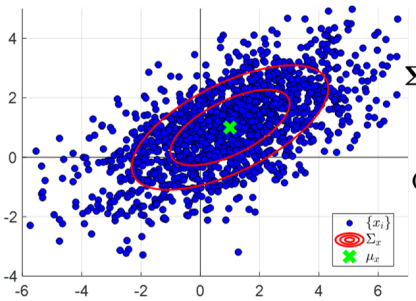
$\boldsymbol{\Sigma}_x = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

Isotropic



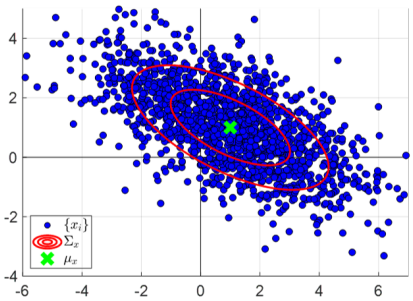
$\boldsymbol{\Sigma}_x = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$

Uncorrelated



$\boldsymbol{\Sigma}_x = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix}$

Positive Correlation



$\boldsymbol{\Sigma}_x = \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix}$

Negative Correlation

2 Principle Component Analysis (PCA)

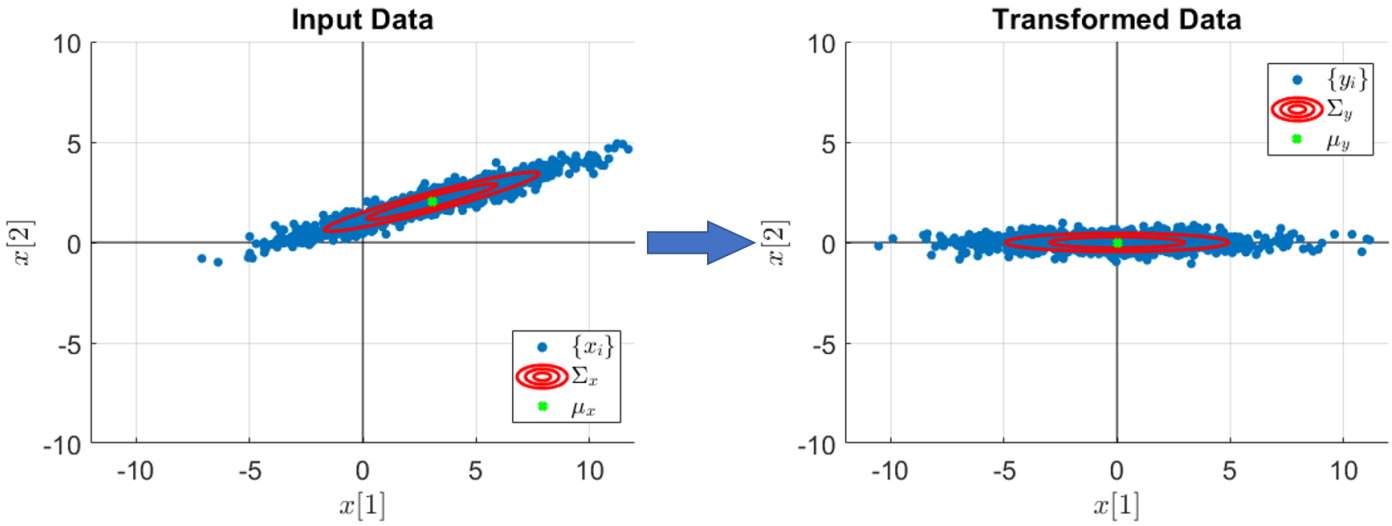
Let $\{\boldsymbol{x}_i\}_{i=1}^N$ be a set of vectors such that $\boldsymbol{x}_i \in \mathbb{R}^D$.
The **Sample (or empirical) Mean** and **Covariance** are given by:

$$\boxed{\boldsymbol{\mu}_x = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \triangleq \overline{\boldsymbol{x}}} \quad \boxed{\boldsymbol{\Sigma}_x = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu}_x)(\boldsymbol{x}_i - \boldsymbol{\mu}_x)^T}$$

- To have an unbiased estimator for $\boldsymbol{\Sigma}_x$, one can use $\frac{1}{N-1}$ instead of $\frac{1}{N}$.
- $\boldsymbol{\Sigma}_x$ is a Positive Semi-Definite (PSD),
  that is, $\boldsymbol{\Sigma}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ where :
  - $\boldsymbol{U}$ is unitary, namely, $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$
  - $\boldsymbol{\Lambda}$ is a diagonal matrix with non-negative elements, namely $\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_D \end{bmatrix}$, and $\lambda_i \geq 0$ for all i.

2.1 Change of basis (axis rotation)

Consider the set $\{\boldsymbol{x}_i\}_{i=1}^N$, where $N = 1000$, and $\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{x}_i[1] \\ \boldsymbol{x}_i[2] \end{bmatrix} \in \mathbb{R}^2$ (left figure).



A common preprocess approach is to apply (right figure):

$$\boldsymbol{y}_i = \boldsymbol{\phi}(\boldsymbol{x}_i)$$

such that:

1. $\{\boldsymbol{y}_i\}_i$ is centered around zero, namely, $\boldsymbol{\mu}_y = \boldsymbol{0}$.
2. $\boldsymbol{\Sigma}_y$ is a diagonal matrix.
3. $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2$ for all i, j.

The following Lemma suggests such map $\boldsymbol{\phi}$.

Lemma 1. *The following transformation:*

$$\boldsymbol{y}_i = \boldsymbol{\phi}\left(\boldsymbol{x}_i\right) = \boldsymbol{U}^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)$$

where the columns of $\boldsymbol{U}$ *are the eigenvectors of* $\boldsymbol{\Sigma}_x$ *satisfies (1), (2) and (3).*

Proof.

1. By definition:

$$\boldsymbol{\mu}_y = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{U}^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right) = \boldsymbol{U}^T\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right) = \boldsymbol{U}^T\left(\overbrace{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i}^{\boldsymbol{\mu}_x} - \boldsymbol{\mu}_x\right) = 0$$

2. Consider the eigen-value decomposition: $\boldsymbol{\Sigma}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ ($\boldsymbol{U}$ is unitary) Hence,

$$\begin{aligned}\boldsymbol{\Sigma}_y &= \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i - \boldsymbol{\mu}_y\right)\left(\boldsymbol{y}_i - \boldsymbol{\mu}_y\right)^T = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i\boldsymbol{y}_i^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{U}^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)^T\boldsymbol{U} = \boldsymbol{U}^T\underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)^T}_{=\boldsymbol{\Sigma}_x}\boldsymbol{U} \\ &= \boldsymbol{U}^T\boldsymbol{\Sigma}_x\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{\Lambda}\end{aligned}$$

3. Since $\boldsymbol{U}^T$ is unitary, wa have $\left\|\boldsymbol{U}^T\boldsymbol{v}\right\|_2 = \|\boldsymbol{v}\|_2$, thus:

$$\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|_2 = \left\|\boldsymbol{U}^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right) - \boldsymbol{U}^T\left(\boldsymbol{x}_j - \boldsymbol{\mu}_x\right)\right\|_2 = \left\|\boldsymbol{U}^T\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right\|_2 = \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2$$

□

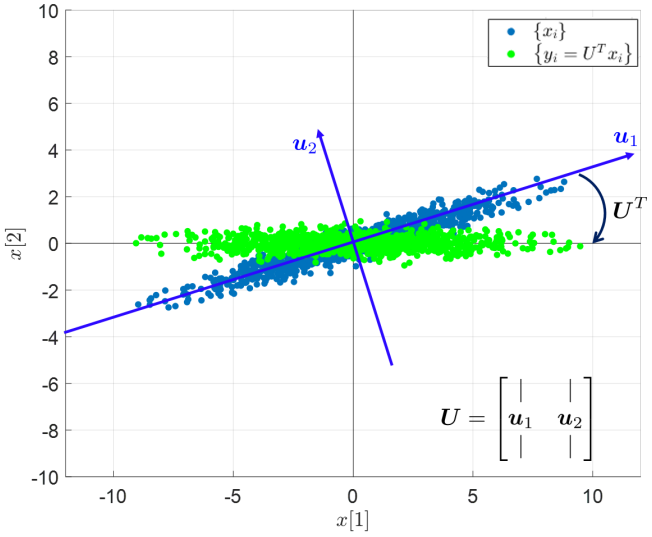
- It is common to arrange the order of $\boldsymbol{\Sigma}_y = \boldsymbol{\Lambda}$ from the largest to the lowest eigenvalue; and respectively, the order of the eigen-vectors $\boldsymbol{U}$. Namely, if:

  $$\boldsymbol{\Sigma}_y = \operatorname{diag}\left(\left[\lambda_1, \lambda_2, \ldots, \lambda_D\right]\right)$$

  Then:

  $$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$$

- Note that after the mean subtraction $\boldsymbol{\mu}_x$, the matrix $\boldsymbol{U}^T$ only rotates the data to its main principal components.



- Note that $\boldsymbol{x}_i$ can be recovered from $\boldsymbol{y}_i$ by:

  $$\boldsymbol{x}_i = \boldsymbol{U}\boldsymbol{y}_i + \boldsymbol{\mu}_x$$

2.2 Linear (affine) dimensionality reduction (using PCA)

- Let D be the dimension of the input data, which also known as the **features vector**: $\boldsymbol{x}_i \in \mathbb{R}^D$.
- D might be a vary large number.
  - For example, D can be the number of pixels in a High Definition (HD) image, $D = 1920 \times 1080$.
- To reduce the length of the features vector we can apply PCA but consider only the first $d < D$ components of $\boldsymbol{y}_i$.

2.2.1 Example 1

- Let $\{\boldsymbol{x}_i\}_i$ be a $2D$ data set, namely $\boldsymbol{x}_i \in \mathbb{R}^2$.
  For simplicity we set $\boldsymbol{\mu}_x = \mathbf{0}$ and let $\boldsymbol{\Sigma}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$.
- The full PCA (namely, only a rotation) is given by:

$$\boldsymbol{y}_i = \boldsymbol{U}^T\boldsymbol{x}_i = \begin{bmatrix} -\boldsymbol{u}_1^T- \\ -\boldsymbol{u}_2^T- \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{x}_i \\ | \end{bmatrix} \in \mathbb{R}^2$$

  and we can perfectly reconstruct $\boldsymbol{x}_i$ by:

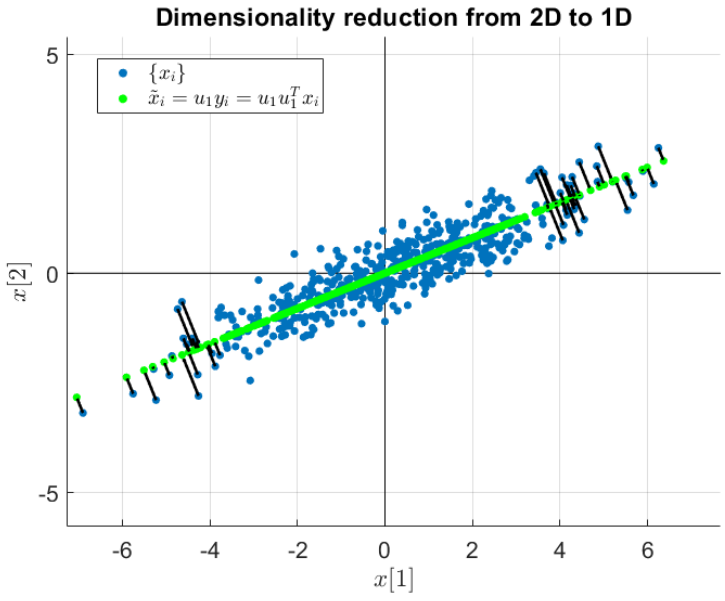
$$\boldsymbol{x}_i = \boldsymbol{U}\boldsymbol{y}_i = \begin{bmatrix} | & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 \\ | & | \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{y}_i \\ | \end{bmatrix} \in \mathbb{R}^2$$

- One can reduce the dimension of $\boldsymbol{y}_i \in \mathbb{R}^2$ from 2 to 1 by consider only the first eigenvector:

$$y_i = \boldsymbol{u}_1^T\boldsymbol{x}_i = \begin{bmatrix} -\boldsymbol{u}_1^T- \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{x}_i \\ | \end{bmatrix} \in \mathbb{R}$$

  where $\boldsymbol{u}_1$ is the first leading eigen-vector of $\boldsymbol{\Sigma}_x$ (corresponding to the largest eigen-vlaue λ_1)
- The imperfect reconstruction is given by:

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{u}_1 y_i = \begin{bmatrix} | \\ \boldsymbol{u}_1 \\ | \end{bmatrix} y_i = \begin{bmatrix} | \\ \boldsymbol{u}_1 \\ | \end{bmatrix} \begin{bmatrix} -\boldsymbol{u}_1^T- \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{x}_i \\ | \end{bmatrix} \in \mathbb{R}^2$$



Reconstruction error Let us compute the error between an original $\boldsymbol{x}_i$ and its reconstructed version $\tilde{\boldsymbol{x}}_i$

$$\epsilon_i^2 \triangleq \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2 = ?$$

$$\begin{aligned}
\epsilon_i^2 &= \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2 \\
&= \left\|\boldsymbol{U}^T\boldsymbol{x}_i - \boldsymbol{U}^T\tilde{\boldsymbol{x}}_i\right\|_2^2 \\
&= \left\|\begin{bmatrix}\boldsymbol{u}_1^T\boldsymbol{x}_i \\ \boldsymbol{u}_2^T\boldsymbol{x}_i\end{bmatrix} - \boldsymbol{U}^T\boldsymbol{u}_1\boldsymbol{u}_1^T\boldsymbol{x}_i\right\|_2^2 \\
&= \left\|\begin{bmatrix}\boldsymbol{u}_1^T\boldsymbol{x}_i \\ \boldsymbol{u}_2^T\boldsymbol{x}_i\end{bmatrix} - \begin{bmatrix}1 \\ 0\end{bmatrix}\boldsymbol{u}_1^T\boldsymbol{x}_i\right\|_2^2 \\
&= \left\|\begin{bmatrix}0 \\ \boldsymbol{u}_2^T\boldsymbol{x}_i\end{bmatrix}\right\|_2^2 \\
&= \left(\boldsymbol{u}_2^T\boldsymbol{x}_i\right)^2
\end{aligned}$$

Using the Pythagorean theorem:

$$\begin{aligned}
\|\boldsymbol{x}_i\|^2 &= \left(\boldsymbol{u}_1^T\boldsymbol{x}_i\right)^2 + \left(\boldsymbol{u}_2^T\boldsymbol{x}_i\right)^2 \\
\Rightarrow \epsilon_i^2 &= \left(\boldsymbol{u}_2^T\boldsymbol{x}_i\right)^2 = \|\boldsymbol{x}_i\|^2 - \left(\boldsymbol{u}_1^T\boldsymbol{x}_i\right)^2
\end{aligned}$$

The overall MSE is obtained by averaging over all points in $\{\boldsymbol{x}_i\}$:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\epsilon_i^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\|\boldsymbol{x}_i\|^2 - \left(\boldsymbol{u}_1^T\boldsymbol{x}_i\right)^2\right)$$

Exercise: Prove that the first leading eigen-vector $\boldsymbol{u}_1$ is indeed the minimizer of the MSE.
Formally, show that

$$\boldsymbol{u}_1 = \arg\min_{\|\boldsymbol{u}\|=1}\text{MSE}\left(\boldsymbol{u}\right)$$

Solution: Using Lagrangian multipliers ($\|\boldsymbol{u}\| = \|\boldsymbol{u}\|^2 = \boldsymbol{u}^T\boldsymbol{u} = 1$), the Lagrangian is:

$$\begin{aligned}
\mathcal{L}\left(\boldsymbol{u},\lambda\right) &= \text{MSE}\left(\boldsymbol{u}\right) + \lambda\left(\boldsymbol{u}^T\boldsymbol{u} - 1\right) \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\|\boldsymbol{x}_i\|^2 - \left(\boldsymbol{u}^T\boldsymbol{x}_i\right)^2\right) + \lambda\left(\boldsymbol{u}^T\boldsymbol{u} - 1\right)
\end{aligned}$$

$$\begin{aligned}
&\Rightarrow \nabla_{\boldsymbol{u}}\mathcal{L}\left(\boldsymbol{u},\lambda\right) = 0 \\
&-\frac{1}{N}\sum_{i=1}^{N}2\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{u} + 2\lambda\boldsymbol{u} = 0 \\
&\underbrace{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T}_{\boldsymbol{\Sigma}_x}\boldsymbol{u} = \lambda\boldsymbol{u} \\
&\boldsymbol{\Sigma}_x\boldsymbol{u} = \lambda\boldsymbol{u}
\end{aligned}$$

$$\Rightarrow \boxed{\boldsymbol{u} \text{ is an eigen-vector of } \boldsymbol{\Sigma}_x}$$

Now:

$$\begin{aligned}
\text{MSE}(\boldsymbol{u}) &= \frac{1}{N}\sum_{i=1}^{N}\left(\|\boldsymbol{x}_i\|_2^2 - \left\|\boldsymbol{x}_i^T\boldsymbol{u}\right\|_2^2\right) \\
&= C - \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{x}_i^T\boldsymbol{u}\right\|_2^2 \\
&= C - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{u}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{u} \\
&= C - \boldsymbol{u}^T\underbrace{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T}_{=\boldsymbol{\Sigma}_x}\boldsymbol{u} \\
&= C - \boldsymbol{u}^T\boldsymbol{\Sigma}_x\boldsymbol{u} \\
&= C - \lambda\underbrace{\boldsymbol{u}^T\boldsymbol{u}}_{=1} \\
&= C - \lambda
\end{aligned}$$

Thus, to minimize the MSE, we should choose $\boldsymbol{u}$ which corresponds to the largest eigen-value.

■

Note: C is in fact the trace of $\boldsymbol{\Sigma}_x$ (which is the sum of the eigen-values):

$$C = \text{tr}\left\{\boldsymbol{\Sigma}_x\right\} = \sum_{i=1}^{D}\lambda_i\left(\boldsymbol{\Sigma}_x\right)$$

2.2.2 Example 2

- Consider the set $\{\boldsymbol{x}_i\}_{i=1}^N$ of $N = 5,000$ images of faces.
- Each image has 32×32 pixels.
- Each $\boldsymbol{x}_i$ is a column stack of all pixels, that is, $\boldsymbol{x}_i \in \mathbb{R}^{1,024}$.

We can apply PCA by:

$$\boldsymbol{y}_i = \boldsymbol{U}_1^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right) \in \mathbb{R}^d$$

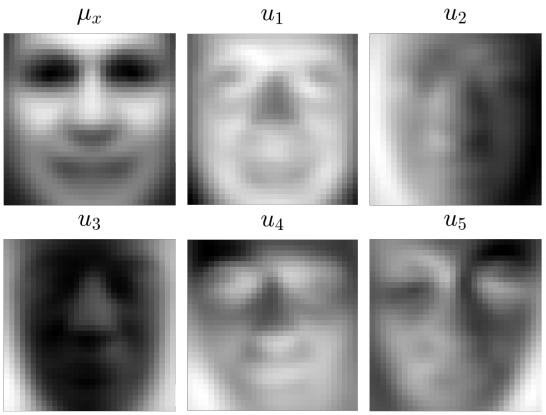
where $\boldsymbol{U}_1$ are the first d eigen-vectors of $\boldsymbol{\Sigma}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$:

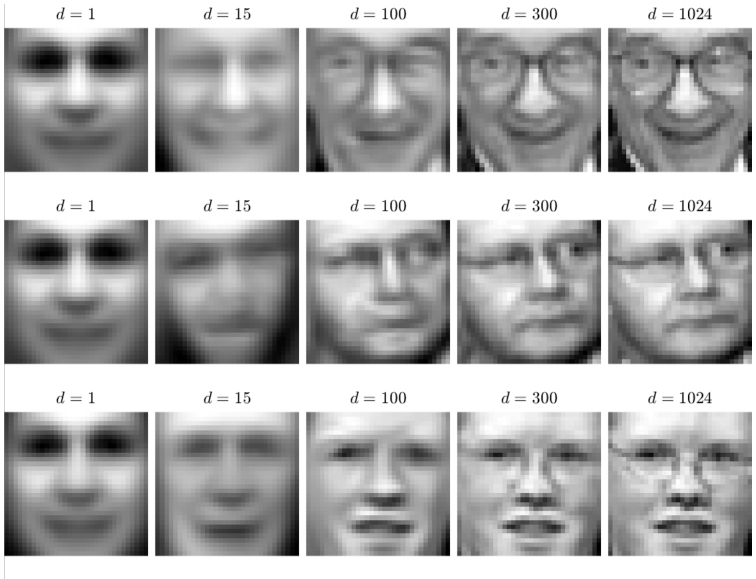
$$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix}, \qquad \boldsymbol{U}_1 \in \mathbb{R}^{D \times d}$$

The reconstruction is given by:

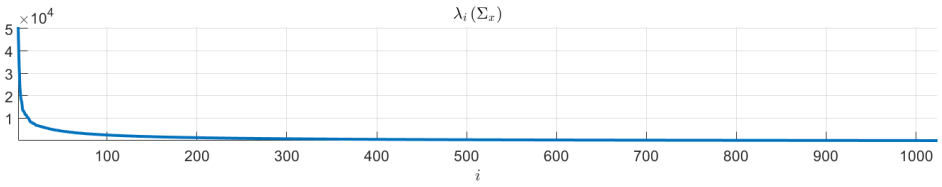
$$\tilde{\boldsymbol{x}}_i = \boldsymbol{U}_1 \boldsymbol{y}_i + \boldsymbol{\mu}_x$$

We present the reconstruction $\tilde{\boldsymbol{x}}_i$ from the low dimensionality representation $\boldsymbol{y}_i$ for several values of d.





The eigenvalues $\{\lambda_i(\boldsymbol{\Sigma}_x)\}_{i=1}^D$ of $\boldsymbol{\Sigma}_x$ can tell us about the reconstruction error:



Notice that most of the data variance is captured by the first 200 eigen-vectors.

2.3 Comments

1. $\boldsymbol{\Sigma}_x$ is a Positive Semi Definite (PSD), namely:
$$\boldsymbol{v}^T\boldsymbol{\Sigma}_x\boldsymbol{v} \geq 0, \qquad \forall \boldsymbol{v}$$

2. Any PSD matrix has non-negative eigenvalues and orthogonal eigen-vectors:
$$\boldsymbol{\Sigma}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T, \qquad \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$$

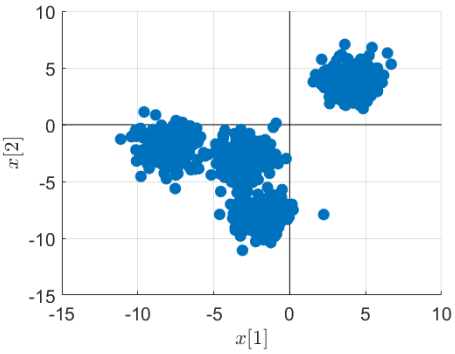
3. Let the columns of $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ contain the set $\{\boldsymbol{x}_i\}_i^D$:
$$\boldsymbol{X} \triangleq \begin{bmatrix} | & & | \\ \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N \\ | & & | \end{bmatrix} \in \mathbb{R}^{D\times N}$$
assume zero mean ($\boldsymbol{\mu}_x = \boldsymbol{0}$).
   (a) The eigen-vectors of $\boldsymbol{\Sigma}_x$ are equal to the left singular vectors of $\boldsymbol{X}$.
   (b) The eigen-values of $\boldsymbol{\Sigma}_x$ are equal to the singular values (squared) of $\boldsymbol{X}$, namely:
$$\sigma_i^2(\boldsymbol{X}) = \lambda_i(\boldsymbol{\Sigma}_x)$$

4. The reconstruction error as a function of d (the number of dimensions) is given by the eigen-vlaues:
$$\text{err}(d) = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\| = \sum_{i=d+1}^{D}\lambda_i$$
where
$$\tilde{\boldsymbol{x}}_i = \boldsymbol{U}_1\boldsymbol{U}_1^T\boldsymbol{x}_i, \qquad \boldsymbol{U}_1 \in \mathbb{R}^{D\times d}$$

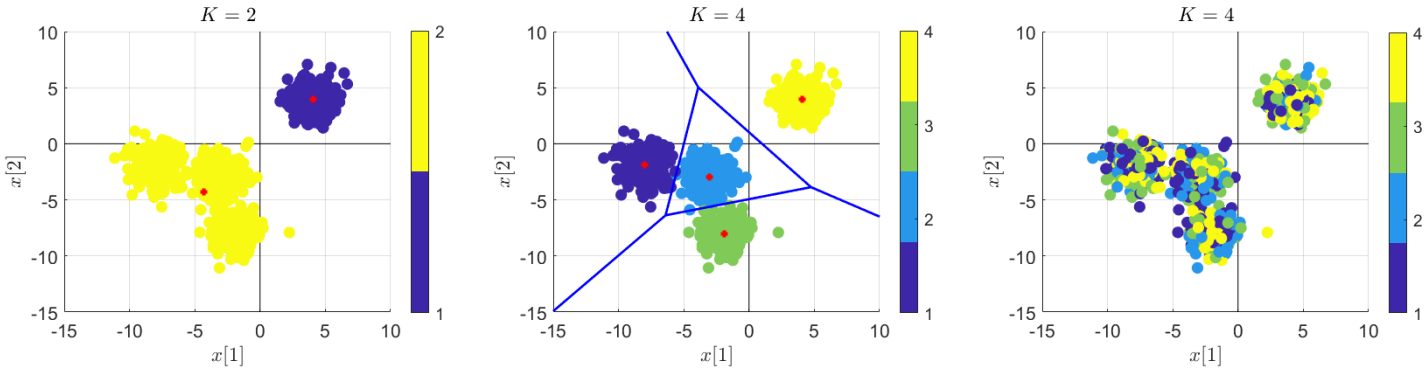
3 K-Means (Clustering)

- Consider the set $\{\boldsymbol{x}_i\}_{i=1}^N$.
- We want to cluster (split) the set $\{\boldsymbol{x}_i\}_i$ into $K > 1$ subsets.

Consider, for example, the following set:



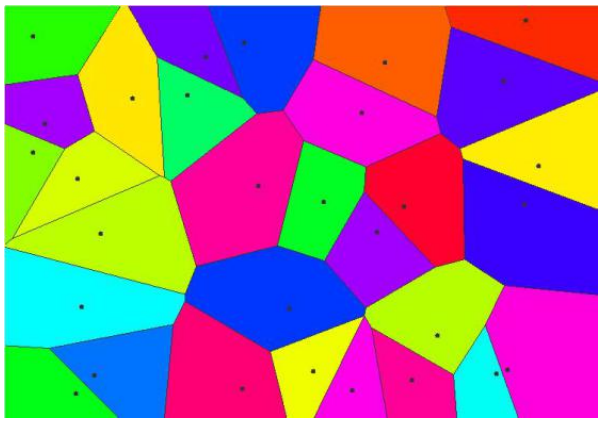
We can split it into $K = 2$ subsets or $K = 4$ subsets like so:



- The red markers are the cluster **centroid.**
- The last clustering (right) makes less sense (but it is a valid clustering into $K = 4$ subsets).
- In order to have a meaningful clustering we need a measure of similarity (or dissimilarity).

3.1 Voronoi

Given a set of centroids $\{\boldsymbol{\mu}_k\}_{k=1}^K$,
the Voronoi diagram plots the regions consisting of all points closer to their centroid than to any other.



3.2 Metric Function

A function $d : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ is called a metric function if it satisfies the following:

1. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$ (Symmetry).
2. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ if and only if $\boldsymbol{x}_i = \boldsymbol{x}_j$.
3. $d(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_j) + d(\boldsymbol{x}_j, \boldsymbol{x}_k)$ (triangle inequality).

3.2.1 Some Examples

1. L_2 (Euclidean):
$$d_E(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 = \sqrt{\sum_{i=1}^{D} (\boldsymbol{x}_1[i] - \boldsymbol{x}_2[i])^2}$$
2. L_1:
$$d_M(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1 = \sum_{i=1}^{D} |\boldsymbol{x}_1[i] - \boldsymbol{x}_2[i]|$$
3. log-Euclidean:
$$d(x_1, x_2) = |\log(x_1) - \log(x_2)|, \qquad x_1, x_2 > 0$$

Exercise: Prove that any metric function d, satisfies:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

for all $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{x}_j \in \mathcal{X}$.

Solution:

$$0 = d(\boldsymbol{x}_i, \boldsymbol{x}_i) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_j) + d(\boldsymbol{x}_j, \boldsymbol{x}_i) = 2d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\Rightarrow \boxed{d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0}$$

■

3.2.2 Dissimilarity Function

Dissimilarity function $d : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ is usually symmetric and non-negative but does not have to be a metric function. For example, the squared Euclidean:

$$d_E^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{i=1}^{D} (\boldsymbol{x}_1[i] - \boldsymbol{x}_2[i])^2$$

does not satisfy the triangle inequality:

$$100 = d_E^2(10, 0) \not\leq d_E^2(10, 5) + d_E^2(5, 0) = 25 + 25 = 50$$

Another example is the cosine dissimilarity:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2}$$

Formulation Using a dissimilarity function d, we write the clustering problem as follows:

$$\min_{\{\mathcal{C}_k\},\{\boldsymbol{\mu}_k\}} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_k\right)$$

where $\mathcal{C}_k \subseteq \{\boldsymbol{x}_i\}$ are the **disjoint** subsets such that $\bigsqcup_k \mathcal{C}_k = \{\boldsymbol{x}_i\}$, and $\{\boldsymbol{\mu}_k\}$ are the clusters centroids.

- Choosing the dissimilarity function will affect the obtained clustering.
- This problem is not convex, hence, finding the global minimum is not guaranteed.

3.3 K-Means Using Squared Euclidean Dissimilarity

Using the squared Euclidean measure we have:

$$\{\mathcal{C}_k\}, \{\boldsymbol{\mu}_k\} = \arg \min_{\{\mathcal{C}_k\},\{\boldsymbol{\mu}_k\}} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2^2 \qquad (1)$$

3.3.1

For fixed centroids $\{\boldsymbol{\mu}_k\}$.
What are the optimal $\{\mathcal{C}_k\}$?

$$\min_{\{\mathcal{C}_k\}} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2^2$$

Note that each point $\boldsymbol{x}_i$ is assigned only to a **single** centroid (one subset).
Thus, to minimize this problem, one should set:

$$\boxed{\mathcal{C}_s = \left\{ \boldsymbol{x}_i \;\middle|\; \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_s\right\|_2^2 \leq \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2^2 \right\}, \qquad k \in \{1, 2, \ldots, K\}}$$

- In case of equality $\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2 = \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_{k'}\right\|_2$ we can assign $\boldsymbol{x}_i$ to the cluster with the smaller index $\min\left(k, k'\right)$.

In other words, we can write:

$$\boxed{\boldsymbol{x}_i \in \mathcal{C}_s \iff s = \arg\min_k \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2^2}$$

3.3.2

For fixed clusters $\{\mathcal{C}_k\}$..
What are the optimal $\{\boldsymbol{\mu}_k\}$?

We can compare the gradient with respect to $\boldsymbol{\mu}_s$ to zero:

$$\begin{aligned} \nabla_{\mu_s} \left(\sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|_2^2 \right) &= \mathbf{0} \\ -2 \sum_{\boldsymbol{x}_i \in \mathcal{C}_s} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_s\right) &= \mathbf{0} \end{aligned}$$

$$\Rightarrow \boxed{\boldsymbol{\mu}_s = \frac{1}{|\mathcal{C}_s|} \sum_{\boldsymbol{x}_i \in \mathcal{C}_s} \boldsymbol{x}_i}$$

In words, the optimal centroid $\boldsymbol{\mu}_k$ of the kth cluster $\mathcal{C}_k$ is the mean of the cluster.

3.3.3 Algorithm

We can find a local minimum in Eq. (1) using the following alternating iterative approach:

Algorithm 1 K-Means

1. Set initial centroids $\{\boldsymbol{\mu}_k\}$
2. For $s = 1, 2, \ldots, K$, find the clusters $\mathcal{C}_s$ by:

$$\mathcal{C}_s = \left\{ \boldsymbol{x}_i \;\middle|\; \|\boldsymbol{x}_i - \boldsymbol{\mu}_s\|_2^2 \leq \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2 \,, \qquad \forall k \right\}$$

3. Update the centroids:

$$\boldsymbol{\mu}_s = \frac{1}{|\mathcal{C}_s|} \sum_{\boldsymbol{x}_i \in \mathcal{C}_s} \boldsymbol{x}_i$$

4. Repeat 2 - 3 until convergence.

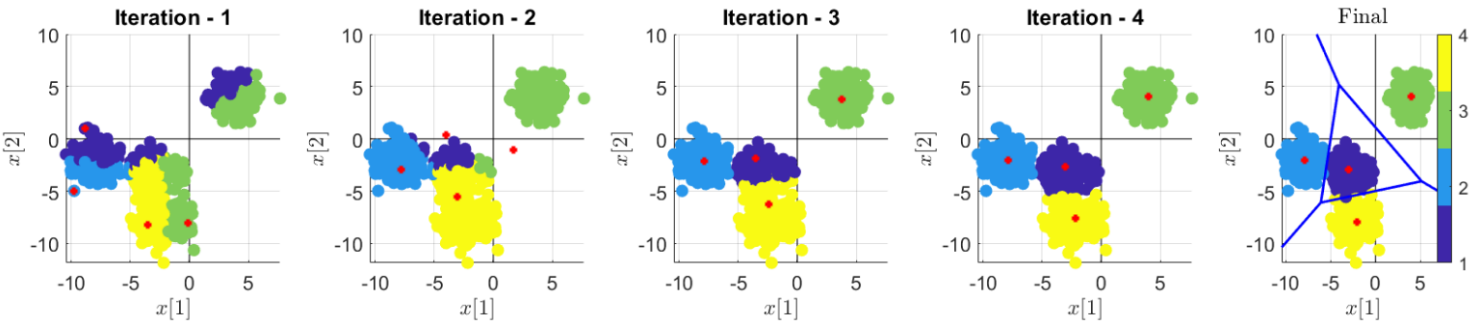
Notes:

1. Choosing the number of clusters K is an important issue.
2. Different initial conditions might lead to different clustering.

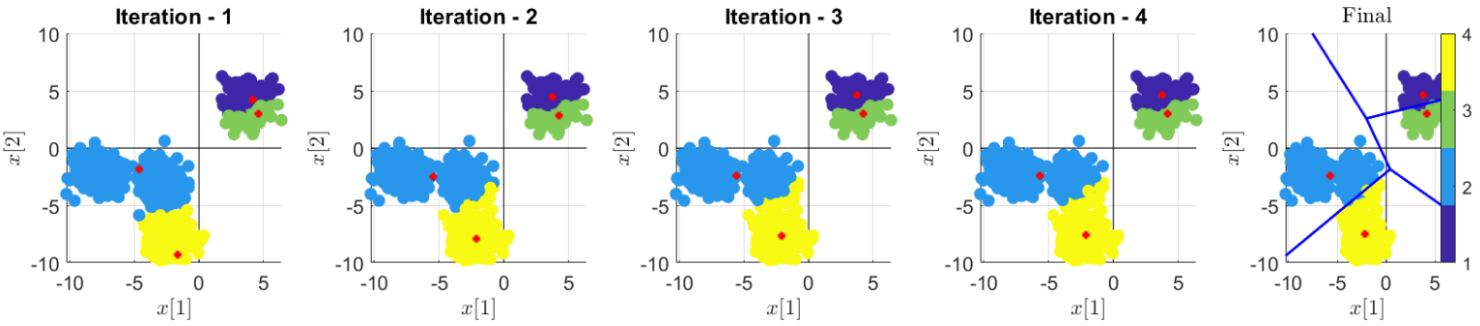
3.3.4 Example 1 - sensitivity to initial conditions

Given the set $\{\boldsymbol{x}_i\}_{i=1}^N$, we set the initial conditions $\{\boldsymbol{\mu}_k\}$ twice:

- First run:



- Second run:

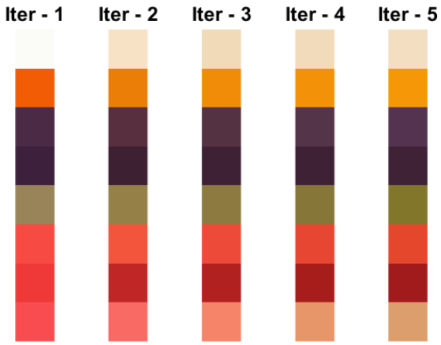


3.3.5 Example 2

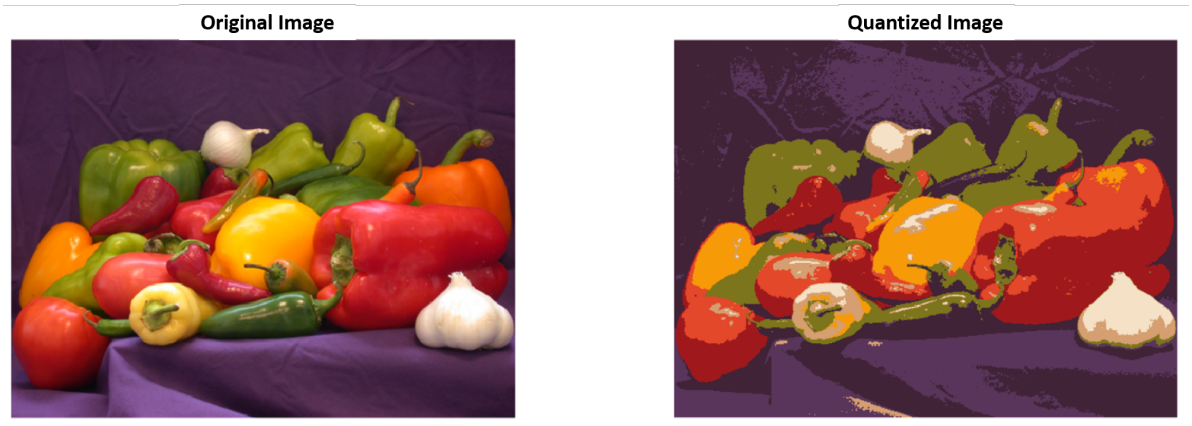
Consider the following image $\boldsymbol{I} \in \mathbb{R}^{384 \times 512 \times 3}$:

- The color image is a combination of 3 images: a red level image, a blue level image, and a green level image.
- Each color (red, blue and green) is represented by $2^8 = 256$ levels.
- Each pixel $\boldsymbol{x}_i \in \mathbb{R}^3$ contains the values of the three colors (Red, Green and Blue).
- Each pixel $\boldsymbol{x}_i$ can represents $2^8 \cdot 2^8 \cdot 2^8 = 2^{24}$ different colors.
- Thus, every pixel require 24 bits in the memory.

We can cluster the 2^{24} colors into $2^3 = 8$ centroids using the K-Means algorithm:



We can represent each pixel by its associate centroid:



The clustered (quantized) image require only 3 bits per pixel (instead of 24).