CS2109S Tutorial 7

Unsupervised Learning

(AY 24/25 Semester 2)

March 27, 2025

(Prepared by Benson)

Contents

Q1. K-Means Algorithm

Extra Slide

Intuition:

- Lines 4-5: Minimize the distortion by reassigning clusters, keeping the centroids unchanged.
- Lines 6-7: Minimize the distortion by changing centroids, keeping the assignment unchanged.

Algorithm K-Means Clustering

1: **for** $k = 1$ **to** K **do**
2: $\quad \boldsymbol{\mu}_k \leftarrow$ random location
3: **while** not converged **do**
4: $\quad$ **for** $i = 1$ **to** m **do**
5: $\quad\quad c^{(i)} \leftarrow \text{argmin}_k \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
6: $\quad$ **for** $k = 1$ **to** K **do**
7: $\quad\quad \boldsymbol{\mu}_k \leftarrow \frac{1}{|\{\boldsymbol{x}^{(i)}|c^{(i)}=k\}|} \sum_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)}|c^{(i)}=k\}} \boldsymbol{x}$

$$\frac{d\mathcal{E}}{d\boldsymbol{\mu}_k} = 0 \quad \Rightarrow \quad \sum_{c^{(i)}=k} 2(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k) = 0 \quad \Rightarrow \quad \boldsymbol{\mu_k} = \frac{1}{|\{\boldsymbol{x}^{(i)}|c^{(i)} = k\}|} \sum_{c^{(i)}=k} \boldsymbol{x}^{(i)}$$

Q1. K-Means Algorithm

(a) Given that every iteration produces a partition with a lower distortion, prove that this algorithm always converges. Convergence is when the centroids/medoids do not change after an iteration of the algorithm.

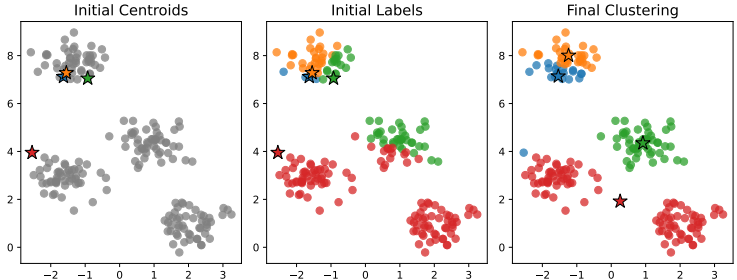
- There are not more than k^N possible partitions.
- Since every iteration of the algorithm produces a partition that has a lower distortion, it eventually terminates.

Algorithm K-Means Clustering

for $k = 1$ **to** K **do**
    $\boldsymbol{\mu}_k \leftarrow$ random location
while not converged **do**
    for $i = 1$ **to** m **do**
        $c^{(i)} \leftarrow \text{argmin}_k \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
    for $k = 1$ **to** K **do**
        $\boldsymbol{\mu}_k \leftarrow \frac{1}{|\{\boldsymbol{x}^{(i)}|c^{(i)}=k\}|} \sum_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)}|c^{(i)}=k\}} \boldsymbol{x}$

Q1. K-Means Algorithm

(b) Although k-means always converges, it may get stuck at a bad local minimum. As mentioned in the lecture, one method of circumventing this is to run the algorithm multiple times and choose the clusters with the minimum distortion. Suggest some other ways to help the algorithm get closer to the global minimum.



- Intuition: Pick centroids that are **as far as possible** during random initialization.

Q1. K-Means Algorithm

Determinstic method:

Algorithm Cluster initialization

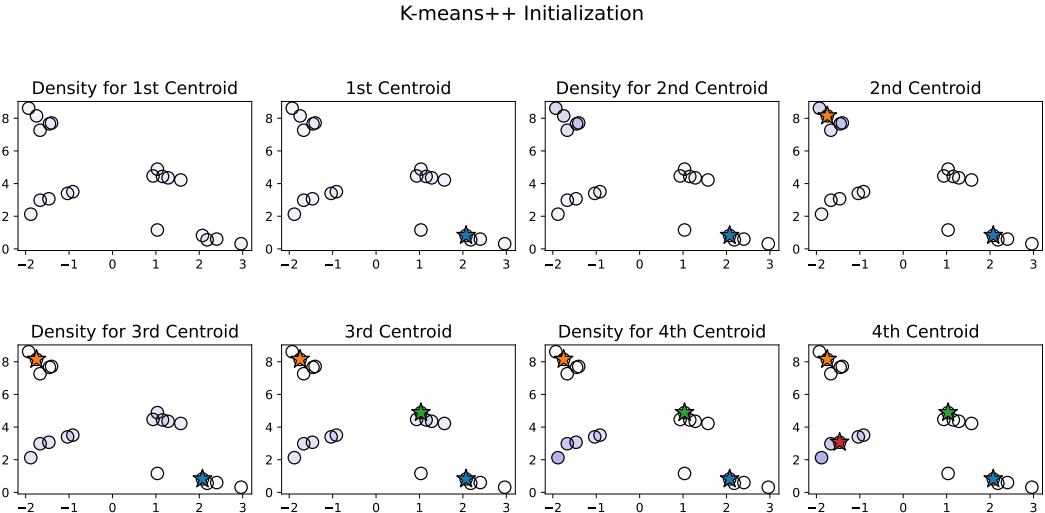
1: $\boldsymbol{\mu}_1 \leftarrow$ random location
2: **for** $k = 2$ **to** K **do**
3: $\quad \boldsymbol{\mu}_k \leftarrow$ point farthest possible from $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{k-1}$

Random method (K-means++):

Algorithm Cluster initialization

1: $\boldsymbol{\mu}_1 \leftarrow$ random location
2: **for** $k = 2$ **to** K **do**
3: $\quad D(x) \leftarrow$ distance from x to closest centroid in $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{k-1}$
4: $\quad \boldsymbol{\mu}_k \leftarrow$ random point with (weighted) probability distribution $D(x)^2$

Q1. K-Means Algorithm



Q1. K-Means Algorithm

(c) Cluster the 6 points in table 1 into two clusters using the K-means algorithm. The two initial centroids are $(0, 1)$ and $(2.5, 2)$.

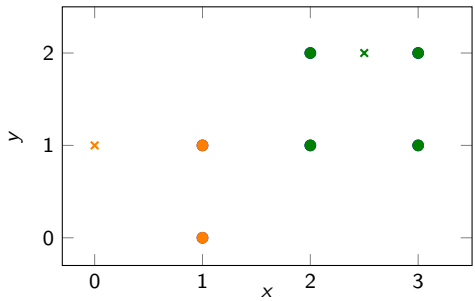
Iteration 1

i	1	2	3	4	5	6
x	1	1	2	2	3	3
y	0	1	1	2	1	2
c	1	1	2	2	2	2

- $\boldsymbol{\mu}_1 = \frac{1}{2}((1,0) + (1,1)) = (1, 0.5)$
- $\boldsymbol{\mu}_2 = \frac{1}{4}((2,1) + (2,2) + (3,1) + (3,2)) = (2.5, 1.5)$

Algorithm K-Means Clustering

for $k = 1$ **to** K **do**
  $\boldsymbol{\mu}_k \leftarrow$ random location
while not converged **do**
  for $i = 1$ **to** m **do**
    $c^{(i)} \leftarrow \text{argmin}_k \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
  for $k = 1$ **to** K **do**
    $\boldsymbol{\mu}_k \leftarrow \frac{1}{|\{\boldsymbol{x}^{(i)} | c^{(i)} = k\}|} \sum_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)} | c^{(i)} = k\}} \boldsymbol{x}$



Q1. K-Means Algorithm

(c) Cluster the 6 points in table 1 into two clusters using the K-means algorithm. The two initial centroids are $(0, 1)$ and $(2.5, 2)$.

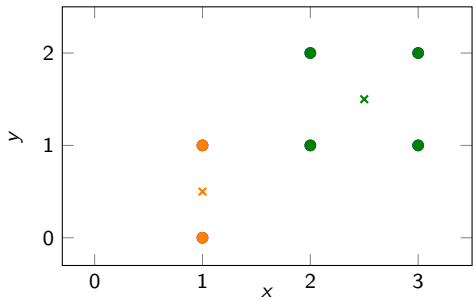
Iteration 2

i	1	2	3	4	5	6
x	1	1	2	2	3	3
y	0	1	1	2	1	2
c	1	1	2	2	2	2

- $\boldsymbol{\mu}_1 = \frac{1}{2}((1,0) + (1,1)) = (1, 0.5)$
- $\boldsymbol{\mu}_2 = \frac{1}{4}((2,1) + (2,2) + (3,1) + (3,2)) = (2.5, 1.5)$

Algorithm K-Means Clustering

for $k = 1$ **to** K **do**
  $\boldsymbol{\mu}_k \leftarrow$ random location
while not converged **do**
  for $i = 1$ **to** m **do**
    $c^{(i)} \leftarrow \text{argmin}_k \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
  for $k = 1$ **to** K **do**
    $\boldsymbol{\mu}_k \leftarrow \frac{1}{|\{\boldsymbol{x}^{(i)} | c^{(i)} = k\}|} \sum_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)} | c^{(i)} = k\}} \boldsymbol{x}$



Q1. K-Means Algorithm

(d) Cluster the 6 points in table 1 into two clusters using the K-medoids algorithm. The initial medoids are point 1 and point 3.

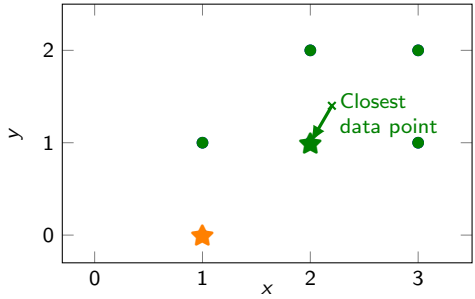
Iteration 1

i	1	2	3	4	5	6
x	1	1	2	2	3	3
y	0	1	1	2	1	2
c	1	2*	2	2	2	2

- $\boldsymbol{\mu}_1 = (1, 0)$
- $\boldsymbol{\mu}_2 = \frac{1}{5}((1,1) + (2,1) + (2,2) + (3,1) + (3,2)) = (2.2, 1.4) \Rightarrow (2, 1)$

Algorithm K-Medoids Clustering

for $k = 1$ **to** K **do**
  $\boldsymbol{\mu}_k \leftarrow$ random data point $\boldsymbol{x}^{(i)}$
while not converged **do**
  for $i = 1$ **to** m **do**
    $c^{(i)} \leftarrow \text{argmin}_k \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
  for $k = 1$ **to** K **do**
    $\boldsymbol{\mu}_k \leftarrow \frac{1}{|\{\boldsymbol{x}^{(i)} | c^{(i)} = k\}|} \sum_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)} | c^{(i)} = k\}} \boldsymbol{x}$
  for $k = 1$ **to** K **do** ▷ *Closest data point*
    $\boldsymbol{\mu}_k \leftarrow \text{argmin}_{\boldsymbol{x} \in \{\boldsymbol{x}^{(i)} | c^{(i)} = k\}} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|^2$



Recap: Kernel Trick

Important Property of a Kernel Function

$$K(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u}) \cdot \phi(\boldsymbol{v}) \text{ for some function } \phi$$

If we have a model that **ONLY** relies on the dot products of $\boldsymbol{x}^{(i)}$:

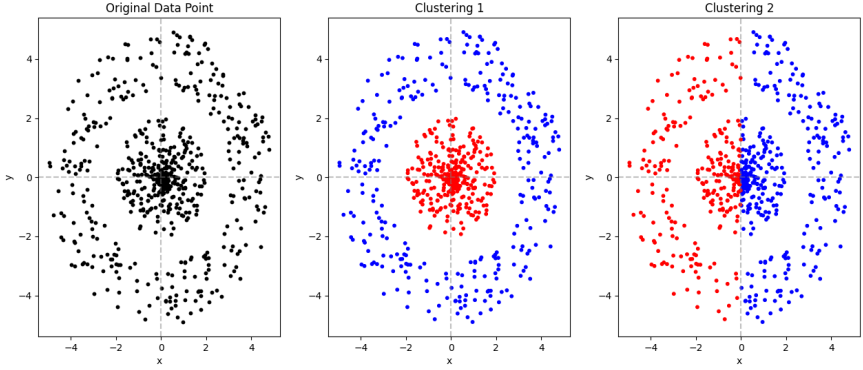
$$\text{SVM}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} (\boldsymbol{x}^{(i)} \cdot \boldsymbol{x})\right)$$

Applying the kernel function $\equiv$ applying the transformed features!

$$\text{SVM}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} (\phi(\boldsymbol{x}^{(i)}) \cdot \phi(\boldsymbol{x}))\right) = \text{sign}\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x})\right)$$

Q2. Kernel K-Means

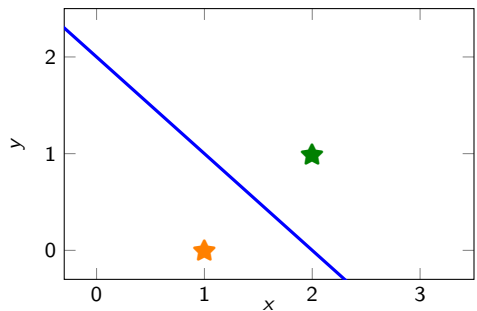
(a) Clustering 1 should be considered better than Clustering 2. Can the K-means algorithm achieve the better clustering? Why or why not?



Q2. Kernel K-Means

(a) Clustering 1 should be considered better than Clustering 2. Can the K-means algorithm achieve the better clustering? Why or why not?

- No. K-means uses **Euclidean distance**.
- Assignment to clusters will follow a **linear boundary**.



Q2. Kernel K-Means

(b) Write the squared Euclidean distance between two points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ using vector norms (length) and the dot product. How can we apply the kernel trick?

$$\begin{aligned}\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2 &= (\boldsymbol{p}_i - \boldsymbol{p}_j) \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j) \\ &= \boldsymbol{p}_i \cdot \boldsymbol{p}_i - \boldsymbol{p}_i \cdot \boldsymbol{p}_j - \boldsymbol{p}_j \cdot \boldsymbol{p}_i + \boldsymbol{p}_j \cdot \boldsymbol{p}_j \\ &= \boldsymbol{p}_i \cdot \boldsymbol{p}_i - 2(\boldsymbol{p}_i \cdot \boldsymbol{p}_j) + \boldsymbol{p}_j \cdot \boldsymbol{p}_j\end{aligned}$$

Naively applying transformed features:

$$\phi(\boldsymbol{p}_i) \cdot \phi(\boldsymbol{p}_i) - 2(\phi(\boldsymbol{p}_i) \cdot \phi(\boldsymbol{p}_j)) + \phi(\boldsymbol{p}_j) \cdot \phi(\boldsymbol{p}_j)$$

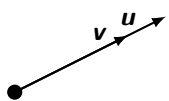
Kernel trick:

$$K(\boldsymbol{p}_i, \boldsymbol{p}_i) - 2K(\boldsymbol{p}_i, \boldsymbol{p}_j) + K(\boldsymbol{p}_j, \boldsymbol{p}_j)$$

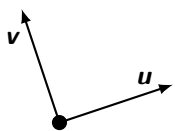
Q2. Kernel K-Means

Dot products can be viewed as a **similarity measure**.

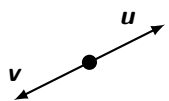
$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\|\|\boldsymbol{v}\| \cos\theta$$



$\theta = 0°, \cos\theta = 1$



$\theta = 90°, \cos\theta = 0$



$\theta = 180°, \cos\theta = -1$

Q2. Kernel K-Means

(c) Gaussian radial basis function (RBF) kernel:

distance $\Rightarrow$ similarity measure!

$$k_{rbf}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \exp\left(-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2}{2\sigma^2}\right)$$

Calculate $k_{rbf}(\boldsymbol{p}_i, \boldsymbol{p}_j)$ for all pairs of data points. Explain what the value represents.

$\boldsymbol{p}$	$k(\boldsymbol{p}_i, \boldsymbol{p}_1)$	$k(\boldsymbol{p}_i, \boldsymbol{p}_2)$	$k(\boldsymbol{p}_i, \boldsymbol{p}_3)$	$k(\boldsymbol{p}_i, \boldsymbol{p}_4)$	$k(\boldsymbol{p}_i, \boldsymbol{p}_5)$
$(0, 0)$	1	e^{-1}	e^{-1}	e^{-1}	e^{-1}
$(4, 4)$	e^{-1}	1	e^{-2}	e^{-4}	e^{-2}
$(-4, 4)$	e^{-1}	e^{-2}	1	e^{-2}	e^{-4}
$(-4, -4)$	e^{-1}	e^{-4}	e^{-2}	1	e^{-2}
$(4, -4)$	e^{-1}	e^{-2}	e^{-4}	e^{-2}	1

p	x	y
1	0	0
2	4	4
3	-4	4
4	-4	-4
5	4	-4

Q2. Kernel K-Means

Extra Slide

Intuition:

- Lines 4-5: Minimize the distortion by reassigning clusters, keeping the centroids unchanged.
- **Do not compute the centers** (We cannot compute $\phi(\boldsymbol{x})$...).

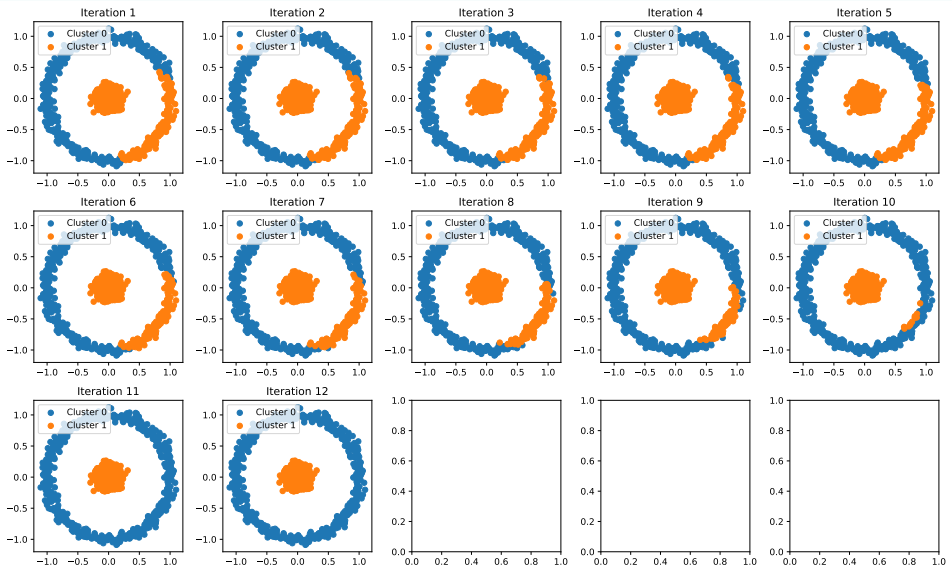
Algorithm Kernel K-Means Clustering

1: **for** $k = 1$ **to** K **do**
2: $\quad \boldsymbol{\mu}_k \leftarrow$ random location
3: **while** not converged **do**
4: $\quad$ **for** $i = 1$ **to** m **do**
5: $\quad\quad c^{(i)} \leftarrow \text{argmin}_k \; d^2_{rbf}(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}_k)$

$$
\begin{aligned}
& d^2_{rbf}(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}_c) \\
&= \phi(\boldsymbol{x}^{(i)}) \cdot \phi(\boldsymbol{x}^{(i)}) - 2 \cdot \left(\phi(\boldsymbol{x}^{(i)}) \cdot \frac{1}{\text{size}} \sum_{j \in \mathcal{C}} \phi(\boldsymbol{x}^{(j)}) \right) + \left(\left(\frac{1}{\text{size}} \sum_{j \in \mathcal{C}} \phi(\boldsymbol{x}^{(j)}) \right) \cdot \left(\frac{1}{\text{size}} \sum_{k \in \mathcal{C}} \phi(\boldsymbol{x}^{(k)}) \right) \right) \\
&= k_{rbf}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) + \frac{2}{\text{size}} \sum_{j \in \mathcal{C}} k_{rbf}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) + \frac{1}{\text{size}^2} \sum_{j,k \in \mathcal{C}} k_{rbf}(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(k)})
\end{aligned}
$$

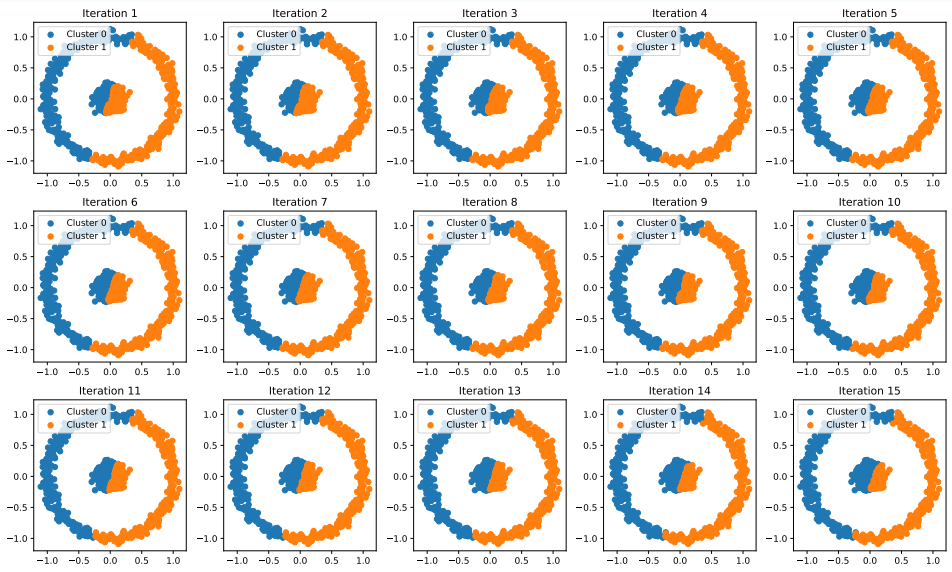
Q2. Kernel K-Means

Extra Slide



Q2. Kernel K-Means

Extra Slide



Recap: Principal Component Analysis (PCA)

$$\boldsymbol{X} = \begin{bmatrix} 4 & 2 & 5 & 1 \\ 1 & 3 & 4 & 0 \end{bmatrix}$$

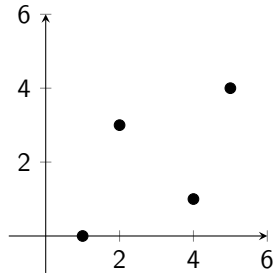
1. **Mean-center** the data (subtract each $\boldsymbol{x}$ by the mean $\bar{\boldsymbol{x}}$).

$$\bar{\boldsymbol{x}} = \frac{1}{4}\begin{bmatrix} 12 \\ 8 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\hat{\boldsymbol{X}} = \begin{bmatrix} 4 & 2 & 5 & 1 \\ 1 & 3 & 4 & 0 \end{bmatrix} - \begin{bmatrix} 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 2 & -2 \\ -1 & 1 & 2 & -2 \end{bmatrix}$$

2. Compute the **covariance matrix** $Cov(\boldsymbol{X}) = \frac{1}{m}\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^\top$.

$$Cov(\boldsymbol{X}) = \frac{1}{4}\begin{bmatrix} 1 & -1 & 2 & -2 \\ -1 & 1 & 2 & -2 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 2 \\ -2 & -2 \end{bmatrix} = \begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 2.5 \end{bmatrix}$$



Recap: Principal Component Analysis (PCA)

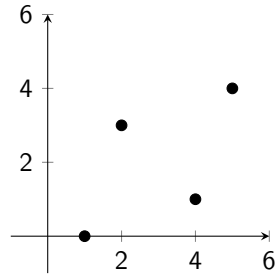
$$\boldsymbol{X} = \begin{bmatrix} 4 & 2 & 5 & 1 \\ 1 & 3 & 4 & 0 \end{bmatrix}, Cov(\boldsymbol{X}) = \begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 2.5 \end{bmatrix}$$

3. Compute the **singuar value decomposition** of $Cov(\boldsymbol{X})$ ($Cov(\boldsymbol{X}) = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$).

$$\begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 2.5 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

4. Reduce the basis to k components to obtain the new basis $\tilde{\boldsymbol{U}}$. Calculate the ratio of explained variance $\frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{m}\sigma_i^2}$.

$$\tilde{\boldsymbol{U}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{m}\sigma_i^2} = \frac{4}{4+1} = 80\%$$



Recap: Principal Component Analysis (PCA)

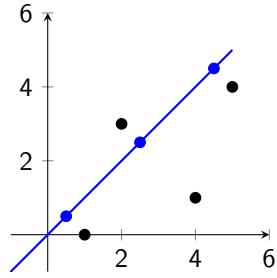
$$\boldsymbol{X} = \begin{bmatrix} 4 & 2 & 5 & 1 \\ 1 & 3 & 4 & 0 \end{bmatrix}, \tilde{\boldsymbol{U}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

- **Reduction**: $\boldsymbol{Z} = \tilde{\boldsymbol{U}}^\top \boldsymbol{X}$

$$Z = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 & 2 & 5 & 1 \\ 1 & 3 & 4 & 0 \end{bmatrix} = \begin{bmatrix} \frac{5}{\sqrt{2}} & \frac{5}{\sqrt{2}} & \frac{9}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- **Reconstruction**: $\boldsymbol{X} \approx \tilde{\boldsymbol{U}}\boldsymbol{Z}$

$$X \approx \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{2}} & \frac{5}{\sqrt{2}} & \frac{9}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 2.5 & 2.5 & 4.5 & 0.5 \\ 2.5 & 2.5 & 4.5 & 0.5 \end{bmatrix}$$



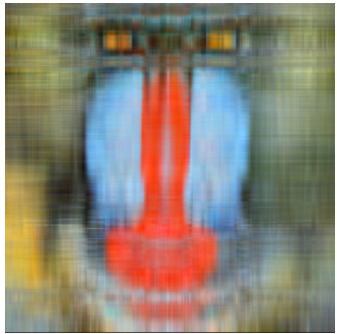
We are only required to store $\tilde{\boldsymbol{U}}$ and $\boldsymbol{Z}$ (instead of $\boldsymbol{X}$)!

Q3. Lossy Compression

(a) The current choice of $k = 9$ does not produce a very nice output. What is a good value for k? Justify your answer.

Before

After k=9



Q3. Lossy Compression

1. Compute the **covariance matrix** $Cov(\boldsymbol{X}) = \frac{1}{m}\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^\top$ *(already includes mean-centering)*.
2. Compute the **singuar value decomposition** of $Cov(\boldsymbol{X})$ ($Cov(\boldsymbol{X}) = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$).

```
cov = np.cov(X, bias=True)
U, s, VT = np.linalg.svd(cov)
```

Q3. Lossy Compression

Goal: Retain at least 99% of the explained variance.

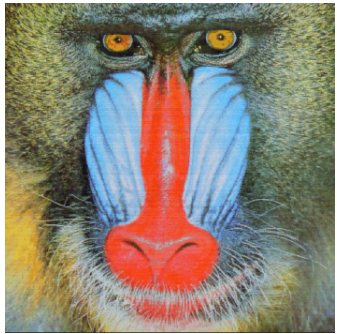
```
def explained_variance(s, k):
    """ returns the explained variance when we keep the first k cols """
    return sum(s[:k]) / sum(s)

minimum_retained_variance = 0.99
k = 0
var = 0
while var < minimum_retained_variance:
    k += 1
    var = explained_variance(s, k)
```

Q3. Lossy Compression

Answer: $k = 286$

Before

After



Q3. Lossy Compression

(b) For the value of k you select in (a), what is the space saved by doing this compression?

- Original: $\boldsymbol{X}$ (512×1536)
- New: $\tilde{\boldsymbol{U}}$ (512×286) and $\boldsymbol{Z}$ (286×1536).
- Compression ratio $= \dfrac{(512 \times 286) + (286 \times 1536)}{512 \times 1536} = 0.745.$
- 25.5% space saved!

Reduction: $\boldsymbol{Z} = \tilde{\boldsymbol{U}}^\top \boldsymbol{X}$
Reconstruction: $X \approx \tilde{\boldsymbol{U}} \boldsymbol{Z}$

Q3. Lossy Compression

(c) What are the drawbacks of this form of compression?

- This is a lossy compression as we do not regain 100% of the variance.
- $\tilde{\boldsymbol{U}}$ is only used for one image $\Rightarrow$ waste of space (if k is large, we might end up using more space).

Bonus. Composing Kernels

Extra Slide

Refer to the template code provided.

1. Let $K^{(1)}(\boldsymbol{u}, \boldsymbol{v})$ and $K^{(2)}(\boldsymbol{u}, \boldsymbol{v})$ be kernel functions.
   Work out the transformed features in the kernel $K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) + K^{(2)}(\boldsymbol{u}, \boldsymbol{v})$.
2. Let $K^{(1)}(\boldsymbol{u}, \boldsymbol{v})$ and $K^{(2)}(\boldsymbol{u}, \boldsymbol{v})$ be kernel functions.
   Work out the transformed features in the kernel $K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) \times K^{(2)}(\boldsymbol{u}, \boldsymbol{v})$.
3. Invent your own kernel!

Bonus. Composing Kernels

Extra Slide

Sample Solution:

1. We have

$$K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) = \phi^{(1)}(\boldsymbol{u}) \cdot \phi_1(\boldsymbol{v})$$
$$K^{(2)}(\boldsymbol{u}, \boldsymbol{v}) = \phi^{(2)}(\boldsymbol{u}) \cdot \phi_2(\boldsymbol{v})$$

Then, we get

$$K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) + K^{(2)}(\boldsymbol{u}, \boldsymbol{v}) = \begin{bmatrix} \phi^{(1)}(\boldsymbol{u}) \\ \phi^{(2)}(\boldsymbol{u}) \end{bmatrix} \cdot \begin{bmatrix} \phi^{(1)}(\boldsymbol{v}) \\ \phi^{(2)}(\boldsymbol{v}) \end{bmatrix}$$

which means the addition of two kernel functions essentially **concatenates** the transformed features together.

Bonus. Composing Kernels

Extra Slide

2. Similarly, we get

$$
\begin{aligned}
K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) \times K^{(2)}(\boldsymbol{u}, \boldsymbol{v}) &= (\phi^{(1)}(\boldsymbol{u}) \cdot \phi^{(1)}(\boldsymbol{v})) \times (\phi^{(2)}(\boldsymbol{u}) \cdot \phi^{(2)}(\boldsymbol{v})) \\
&= \left(\sum_i \phi_i^{(1)}(\boldsymbol{u}) \phi_i^{(1)}(\boldsymbol{v}) \right) \times \left(\sum_j \phi_j^{(2)}(\boldsymbol{u}) \phi_j^{(2)}(\boldsymbol{v}) \right) \\
&= \sum_{ij} \left(\phi_i^{(1)}(\boldsymbol{u}) \phi_j^{(2)}(\boldsymbol{u}) \right) \left(\phi_i^{(1)}(\boldsymbol{v}) \phi_j^{(2)}(\boldsymbol{v}) \right)
\end{aligned}
$$

Define another set of transformed features $\phi^*(\boldsymbol{u})$ which consists of $\phi_i^{(1)}(\boldsymbol{u})\phi_j^{(2)}(\boldsymbol{u})$ for all pairs (i, j), then

$$
K^{(1)}(\boldsymbol{u}, \boldsymbol{v}) \times K^{(2)}(\boldsymbol{u}, \boldsymbol{v}) = \phi^*(\boldsymbol{u}) \cdot \phi^*(\boldsymbol{v})
$$

which means the product of two kernel functions takes the **pairwise product** of between their transformed features.

Bonus. Composing Kernels

Extra Slide

1. Just a simple concatenation.

```
def transform_add(X1, X2):
    return np.hstack([X1, X2])
```

2. Broadcasting is your best friend!

```
def transform_multiply(X1, X2):
    n = X1.shape[0]
    pairwise_prod = X1.reshape(n, -1, 1) * X2.reshape(n, 1, -1)
    return pairwise_prod.reshape(n, -1)
```