Unsupervised Learning

Machine Learning Tutorials: Exercises 3, 4 et 5
Clustering, Dimensionality Reduction, Dictionaries

Christian Heinrich, Aurélien Reutenauer, Antoine Deleforge, Sylvain Faisan
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
1. Introduction

Supervised Learning

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^{N} \]

Input

\[ [x_1, x_2, \ldots, x_N] \]

Labels/Targets

\[ [y_1, y_2, \ldots, y_N] \]
1. Introduction

Supervised Learning

Labeled training data

\[ \{ (x_n, y_n) \}_{n=1}^{N} \]

\[ [x_1, x_2, \ldots, x_N] \quad [y_1, y_2, \ldots, y_N] \]

Input \quad \text{Labels/Targets}

Training

Learned Model \( \mathcal{M}_\theta \)
1. Introduction

Supervised Learning

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^{N} \]

Input

\[ [x_1, x_2, \ldots, x_N] \]

Labels/Targets

\[ [y_1, y_2, \ldots, y_N] \]

Test

\[ \tilde{x} \]

Training

Learned Model

\[ \mathcal{M}_\theta \]

Estimate

\[ \tilde{y} \]
1. Introduction

Supervised Learning

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^{N} \]

\[ [x_1, x_2, \ldots, x_N] \]

\[ [y_1, y_2, \ldots, y_N] \]

Input

Labels/Targets

Test

\[ \tilde{x} \]

Training

Learned Model \( M_\theta \)

Estimate

\[ \tilde{y} \]

1. Discrete case: («one-hot»)
   - Classification
   - Ex. application: dog breed

\[ y = 1 \]

13: German Shepherd
1. Introduction

Supervised Learning

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^N \]

\[ [x_1, x_2, \ldots, x_N] \quad [y_1, y_2, \ldots, y_N] \]

Input
Labels/Targets

Test \(\tilde{x}\)

Training

Learned Model \(\mathcal{M}_\theta\)

Estimate \(\tilde{y}\)

1. **Discrete case:** («one-hot»)
   - Classification
     Ex. application: *dog breed*
     \[ y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \]

2. **Continuous case:**
   - Regression
     Ex. application: *head pose*

\[ y = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \]

13: German Shepherd
1. Introduction

Supervised Learning

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^N \]

Input

\[ [x_1, x_2, \ldots, x_N] \]

Labels/Targets

\[ [y_1, y_2, \ldots, y_N] \]

1. **Discrete case:** («one-hot»)
   - Classification
   - Ex. application: dog breed

2. **Continuous case:**
   - Regression
   - Ex. application: head pose

3. **Sparse case:**
   - Multi-classification
   - Ex. application: image labelling

\[ y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \]

\[ y = \begin{bmatrix} 0.1 \\ 0.6 \\ 0.3 \end{bmatrix} \]

Ex. application: dog breed

Ex. application: head pose

Ex. application: image labelling
1. Introduction

## Supervised Learning

Labeled training data

\[ \{ (x_n, y_n) \}_{n=1}^{N} \]

- **Input**
- **Labels/Targets**

![Diagram of Supervised Learning](image)

**Discrete case:** («one-hot»)
- Classification
- Ex. application: *dog breed*

1. German Shepherd

**Continuous case:**
- Regression
- Ex. application: *head pose*

**Sparse case:**
- Multi-classification
- Ex. application: *image labelling*

<table>
<thead>
<tr>
<th>label</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>0.1</td>
</tr>
<tr>
<td>phone</td>
<td>0.3</td>
</tr>
<tr>
<td>palm tree</td>
<td>0.6</td>
</tr>
</tbody>
</table>

## Unsupervised Learning

![Diagram of Unsupervised Learning](image)
1. Introduction

Supervised Learning

Labeled training data

\[
\{ (x_n, y_n) \}_{n=1}^N
\]

Unlabeled training data

\[
\{ x_n \}_{n=1}^N
\]

Test

\[ \tilde{x} \]

Training

\[
\text{Learned Model } \mathcal{M}_\theta
\]

Estimate

\[ \tilde{y} \]

1. Discrete case: («one-hot»)
   - Classification
   - Ex. application: dog breed

2. Continuous case:
   - Regression
   - Ex. application: head pose

3. Sparse case:
   - Multi-classification
   - Ex. application: image labelling

Unsupervised Learning

Labeled training data

\[
[x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_N]
\]

Test

\[ \tilde{x} \]

Training

\[
\text{Learned Model } \mathcal{M}_\theta
\]

Estimate

\[ \tilde{y} \]

1. Discrete case: («one-hot»)
   - Classification
   - Ex. application: dog breed

2. Continuous case:
   - Regression
   - Ex. application: head pose

3. Sparse case:
   - Multi-classification
   - Ex. application: image labelling
Supervised Learning

Labeled training data
\( \{ (x_n, y_n) \}_{n=1}^N \)

Unlabeled training data
\( \{ x_n \}_{n=1}^N \)

Input

Labels/Targets

1. Discrete case: («one-hot»)
   - Classification
     - Ex. application: dog breed
     - \( y = 1 \)
     - 13: German Shepherd

2. Continuous case:
   - Regression
     - Ex. application: head pose
     - \( y = \) [Diagram showing head pose regression]

3. Sparse case:
   - Multi-classification
     - Ex. application: image labelling
     - \( y = \) [Diagram showing label probabilities]

Unsupervised Learning

Test
\( x \)

Training

Learned Model \( \mathcal{M}_\theta \)

Estimate
\( \tilde{y} \)

Test
\( \tilde{x} \)

Training

Learned Model \( \mathcal{M}_\theta \)
1. Introduction

**Supervised Learning**

Labeled training data

\[ \{ (x_n, y_n) \}_{n=1}^N \]

\[ \left[ \begin{array}{cccc} x_1, & x_2, & \ldots, & x_N \\ y_1, & y_2, & \ldots, & y_N \end{array} \right] \]

Input → Labels/Targets

Test \( \tilde{x} \) → Training → Learned Model \( \mathcal{M}_\theta \) → Estimate \( \tilde{y} \)

- **Discrete case**: («one-hot»)
  - Classification
  - Ex. application: dog breed

- **Continuous case**:
  - Regression
  - Ex. application: head pose

- **Sparse case**:
  - Multi-classification
  - Ex. application: image labelling

**Unsupervised Learning**

Unlabeled training data

\[ \{ x_n \}_{n=1}^N \]

\[ \left[ \begin{array}{cccc} x_1, & x_2, & \ldots, & x_N \end{array} \right] \]

Input

Training → Learned Model \( \mathcal{M}_\theta \) → Representation

\[ \left[ \begin{array}{cccc} z_1, & z_2, & \ldots, & z_N \end{array} \right] \]
1. Introduction

## Supervised Learning

- **Labeled training data**
  \[
  \{(x_n, y_n)\}^N_{n=1} \hspace{1cm} \begin{bmatrix} x_1, x_2, \ldots, x_N \end{bmatrix} \hspace{1cm} \begin{bmatrix} y_1, y_2, \ldots, y_N \end{bmatrix}
  \]

- **Test**
  \[\tilde{x}\]

- **Train**
  Learned Model \( \mathcal{M}_\theta \)

- **Estimate**
  \[\tilde{y}\]

### 1. Discrete case: («one-hot»)
- Classification
- Ex. application: dog breed

### 2. Continuous case:
- Regression
- Ex. application: head pose

### 3. Sparse case:
- Multi-classification
- Ex. application: image labelling

## Unsupervised Learning

- **Unlabeled training data**
  \[\{x_n\}^N_{n=1} \hspace{1cm} [x_1, x_2, \ldots, x_N] \hspace{1cm} \]

- **Train**
  Learned Model \( \mathcal{M}_\theta \)

- **Represent**
  \[\{z_1, z_2, \ldots, z_N\} \hspace{1cm} [z_1, z_2, \ldots, z_N] \hspace{1cm} \]

### 1. Discrete case:
- Clustering

---

Antoine.Deleforge@inria.fr
1. Introduction

---

**Supervised Learning**

Labeled training data

\[ \{(x_n, y_n)\}_{n=1}^N \]

\[
\begin{bmatrix}
  x_1 & x_2 & \cdots & x_N \\
  y_1 & y_2 & \cdots & y_N \\
\end{bmatrix}
\]

\text{Input} \quad \leftrightarrow \quad \text{Labels/Targets}

Test

\[ \tilde{x} \]

\[ \rightarrow \]

Learned Model \[ M_\theta \]

\[ \rightarrow \]

Estimate \[ \tilde{y} \]

---

**Unsupervised Learning**

Unlabeled training data

\[ \{x_n\}_{n=1}^N \]

\text{Input}

Training

Learned Model \[ M_\theta \]

\[ \rightarrow \]

Representation

\[
\begin{bmatrix}
  z_1 & z_2 & \cdots & z_N \\
\end{bmatrix}
\]

---

1. **Discrete case:** («one-hot»)
   - Classification
     - Ex. application: dog breed

2. **Continuous case:**
   - Regression
     - Ex. application: head pose

3. **Sparse case:**
   - Multi-classification
     - Ex. application: image labelling

---

**1. Discrete case:**

\[ y = \begin{bmatrix} 1 \end{bmatrix} \]

Ex. application: dog breed

- Cluster: German Shepherd

**2. Continuous case:**

\[ y = \begin{bmatrix} 0.1 & 0.6 & 0.3 \end{bmatrix} \]

Ex. application: image labelling

\[ z = \begin{bmatrix} \end{bmatrix} \]

- Dimensionality Reduction

---

École d'ingénieurs
Télécom Physique
Université de Strasbourg

Antoine.Deleforge@inria.fr

Unsupervised Learning - 2021
1. Introduction

Supervised Learning

Labeled training data
\[ \{ (x_n, y_n) \}_{n=1}^N \]

Unlabeled training data
\[ \{ x_n \}_{n=1}^N \]

 Supervised Learning

Test
\[ \tilde{x} \]

Training
 Learned Model \[ M_\theta \]

Estimate
\[ \tilde{y} \]

Unsupervised Learning

 Supervised Learning

Test
\[ \tilde{x} \]

Training
 Learned Model \[ M_\theta \]

Estimate
\[ \tilde{y} \]

Unsupervised Learning

 Supervised Learning

Test
\[ \tilde{x} \]

Training
 Learned Model \[ M_\theta \]

Estimate
\[ \tilde{y} \]

1. Discrete case: («one-hot»)
   - Classification
     Ex. application: *dog breed*

2. Continuous case:
   - Regression
     Ex. application: *head pose*

3. Sparse case:
   - Multi-classification
     Ex. application: *image labelling*

1. Discrete case:
   - Clustering

2. Continuous case:
   - Dimensionality Reduction

3. Sparse case:
   - Dictionary Learning
1. Introduction

Applications of Unsupervised Learning

- Data Visualization
- Segmentation
- Denoising
- User Recommendation
- Feature Learning
- Data Generation
- Compression
- Data Completion
- Self-supervision

...
1. Introduction

Applications of Unsupervised Learning

- Data Visualization
- Segmentation
- Feature Learning
- Denoising
- User Recommendation
- Compression
- Data Generation
- Data Completion
- Self-supervision
- …

« If AI was a cake, reinforcement learning would be the cherry on the cake, supervised learning the icing, and unsupervised learning the génoise. »

-Yann Lecun (Facebook AI) at NIPS 2016
1. Introduction

Applications of Unsupervised Learning

Data Visualization
Segmentation
User Recommendation
Denoising
Feature Learning
Compression
Data Generation
Data Completion
Self-supervision

« If AI was a cake, reinforcement learning would be the cherry on the cake, supervised learning the icing, and unsupervised learning the génoise. »

-Yann Lecun (Facebook AI) at NIPS 2016

Potential to learn from massive amount of unlabeled data to generate even more
1. Introduction

How does it work?

\[
\begin{bmatrix}
  x_1, x_2, \ldots, x_N
\end{bmatrix}
\]

Input

\[
\begin{bmatrix}
  z_1, z_2, \ldots, z_N
\end{bmatrix}
\]

Representation

\[
\begin{bmatrix}
  0, 0, 0, \ldots, 0, 0
\end{bmatrix}
\]
How does it work?

1. Introduction

How does it work?

\[
\begin{bmatrix}
  x_1, x_2, \ldots, x_N
\end{bmatrix}
\]

\[
\begin{bmatrix}
  z_1, z_2, \ldots, z_N
\end{bmatrix}
\]

Input

Representation

\[
\begin{bmatrix}
  0, 0, 0, \ldots, 0, 0
\end{bmatrix}
\]
1. Introduction

How does it work?

• **Desirable properties**

  - **Preserve information from original data**
  - **Can be used to reconstruct it / generalize it / efficiently learn from it**
1. Introduction

How does it work?

- **Desirable properties**
  - Preserve information from original data
  - Can be used to reconstruct it / generalize it / efficiently learn from it

- **A unifying view**: Generative models
1. Introduction

How does it work?

- **Desirable properties**
  - Preserve information from original data
  - Can be used to reconstruct it / generalize it / efficiently learn from it

- **A unifying view:** Generative models

  Find \( p_\theta(z) \) (as simple as possible)

  \[ p_\theta(x|z) \] (typically, transformation of \( z \))

  so that \( p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz \)

  *fits* the data (e.g., maximum likelihood).
1. Introduction

How does it work?

**Desirable properties**
- Preserve information from original data
- Can be used to reconstruct it / generalize it / efficiently learn from it

**A unifying view**: Generative models

Find
\[
\begin{align*}
&\quad p_\theta(z) \quad \text{(as simple as possible)} \\
&\quad p_\theta(x|z) \quad \text{(typically, transformation of } z) \\
\end{align*}
\]

so that
\[
p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz
\]

*fits* the data (e.g., maximum likelihood).

<table>
<thead>
<tr>
<th></th>
<th>Generation</th>
<th>Compression</th>
<th>Reconstruction / Completion / Denoising</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_\theta(z) \rightarrow z \rightarrow x$</td>
<td>$x \rightarrow p_\theta(z</td>
<td>x) \rightarrow z$</td>
</tr>
</tbody>
</table>
## Content of this TP

- **Explore by practicing** the most fundamental techniques in these 3 categories:

<table>
<thead>
<tr>
<th>Clustering (Exo3)</th>
<th>Dim. Reduction (Exo4)</th>
<th>Dict. Learning (Exo5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>∏</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Techniques</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>K-means</strong>*</td>
<td>• <strong>PCA</strong>*</td>
<td>• Sparse Coding</td>
</tr>
<tr>
<td>• GMM EM</td>
<td>• Manifold Learning</td>
<td><strong>K-SVD</strong>*</td>
</tr>
</tbody>
</table>

*you will implement yourself!
## Content of this TP

- **Explore by practicing** the most fundamental techniques in these 3 categories:

<table>
<thead>
<tr>
<th>Clustering (Exo3)</th>
<th>Dim. Reduction (Exo4)</th>
<th>Dict. Learning (Exo5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>Discrete</td>
<td>Continuous</td>
</tr>
<tr>
<td>Techniques</td>
<td>K-means*</td>
<td>PCA*</td>
</tr>
<tr>
<td></td>
<td>GMM EM</td>
<td>Manifold Learning</td>
</tr>
</tbody>
</table>

*you will implement yourself!

- They form the basis of many advanced unsupervised deep learning techniques used today:
  - Variational Auto-Encoders (VAEs)
  - Deep Clustering
  - Self-Supervised Learning (DINO, …)
  - Conditional Neural Processes
  - Nonlinear Independent Component Estimation (NICE, FLOW, …)
  - Generative Adversarial Networks (GANs)
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
2. Clustering

How would you cluster/group/separate these 2D points?

\[ \{ \mathbf{x}_n \}_{n=1}^{N} \subset \mathbb{R}^2 \]
2. Clustering

How would you cluster/group/separate these 2D points?

$\{ x_n \}_{n=1}^N \subset \mathbb{R}^2$

1. Pick $K = 4$ points at random
   - The centroids $\{ c_k \}_{k=1}^K \subset \mathbb{R}^2$
2. Clustering

How would you cluster/group/separate these 2D points?

\[ \{ \mathbf{x}_n \}_{n=1}^{N} \subset \mathbb{R}^2 \]

1. Pick \( K = 4 \) points at random
   ▶ The centroids \( \{ c_k \}_{k=1}^{K} \subset \mathbb{R}^2 \)

2. For each point \( \mathbf{x}_n \), find its nearest centroid \( c_k \). Place \( n \) in cluster \( G_k \).
2. Clustering

**How would you cluster/group/separate these 2D points?**

1. Pick $K = 4$ points at random
   - The centroids $\{c_k\}_{k=1}^{K} \subset \mathbb{R}^2$

2. For each point $x_n$, find its nearest centroid $c_k$. Place $n$ in cluster $G_k$.

3. Update each $c_k$ as the mean of all points in $G_k$:
   - $c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n$
2. Clustering

How would you cluster/group/separate these 2D points?

\[ \{ x_n \}_{n=1}^{N} \subset \mathbb{R}^2 \]

1. Pick \( K = 4 \) points at random
   - The centroids \( \{ c_k \}_{k=1}^{K} \subset \mathbb{R}^2 \)

2. For each point \( x_n \), find its nearest centroid \( c_k \). Place \( n \) in cluster \( G_k \).

3. Update each \( c_k \) as the mean of all points in \( G_k \):
   - \( c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n \)

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

1. Pick $K = 4$ points at random
   - The centroids $\{c_k\}_{k=1}^{K} \subset \mathbb{R}^2$

2. For each point $x_n$, find its nearest centroid $c_k$. Place $n$ in cluster $G_k$.

3. Update each $c_k$ as the mean of all points in $G_k$:
   - $c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n$

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

\[ \{ \mathbf{x}_n \}_{n=1}^{N} \subset \mathbb{R}^2 \]

1. Pick \( K = 4 \) points at random
   - The centroids \( \{ c_k \}_{k=1}^{K} \subset \mathbb{R}^2 \)

2. For each point \( \mathbf{x}_n \), find its nearest centroid \( c_k \). Place \( n \) in cluster \( G_k \).

3. Update each \( c_k \) as the mean of all points in \( G_k \):
   - \[ c_k = \frac{1}{|G_k|} \sum_{n \in G_k} \mathbf{x}_n \]

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

1. Pick $K = 4$ points at random
   - The *centroids* $\{c_k\}_{k=1}^K \subset \mathbb{R}^2$

2. For each point $x_n$, find its nearest centroid $c_k$. Place $n$ in cluster $G_k$.

3. Update each $c_k$ as the mean of all points in $G_k$:
   - $c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n$

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

1. Pick $K = 4$ points at random
   ▶ The centroids $\{c_k\}_{k=1}^K \subset \mathbb{R}^2$

2. For each point $x_n$, find its nearest centroid $c_k$. Place $n$ in cluster $G_k$.

3. Update each $c_k$ as the mean of all points in $G_k$:
   ▶ $c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n$

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

\[ \{ x_n \}_{n=1}^{N} \subset \mathbb{R}^2 \]

1. Pick \( K = 4 \) points at random
   - The centroids \( \{ c_k \}_{k=1}^{K} \subset \mathbb{R}^2 \)

2. For each point \( x_n \), find its nearest centroid \( c_k \). Place \( n \) in cluster \( G_k \).

3. Update each \( c_k \) as the mean of all points in \( G_k \):
   - \( c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n \)

4. Repeat 2. and 3. until convergence
2. Clustering

How would you cluster/group/separate these 2D points?

1. Pick $K = 4$ points at random
   - The centroids $\{c_k\}_{k=1}^K \subset \mathbb{R}^2$

2. For each point $x_n$, find its nearest centroid $c_k$. Place $n$ in cluster $G_k$.

3. Update each $c_k$ as the mean of all points in $G_k$:
   - $c_k = \frac{1}{|G_k|} \sum_{n \in G_k} x_n$

4. Repeat 2. and 3. until convergence

The K-means Algorithm
2. Clustering

K-means: What does it do?

- It decreases this cost function at each iteration:

\[
C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \| x_n - c_k \|^2, \quad \theta = \{c_1, \ldots, c_K\}
\]
2. Clustering

**K-means: What does it do?**

- It decreases this cost function at each iteration:

\[
C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \|x_n - c_k\|^2, \quad \theta = \{c_1, \ldots, c_K\}
\]

- Non-Convex
  (Depends on init.)
2. Clustering

K-means: What does it do?

• It decreases this cost function at each iteration:

\[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \|x_n - c_k\|^2, \quad \theta = \{c_1, \ldots, c_K\} \]

▶ Non-Convex (Depends on init.)

• « Compression » interpretation: \( X \) is « summarized » by \( \theta \)
2. Clustering

K-means: What does it do?

- It decreases this cost function at each iteration:

\[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \| x_n - c_k \|^2, \quad \theta = \{ c_1, \ldots, c_K \} \]

- « Compression » interpretation: \( X \) is « summarized » by \( \theta \)

- Probabilistic / Generative interpretation (where is \( z \) ?)

- Non-Convex (Depends on init.)
2. Clustering

K-means: What does it do?

- It decreases this cost function at each iteration:

\[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \| x_n - c_k \|^2, \quad \theta = \{c_1, \ldots, c_K\} \]

- Non-Convex (Depends on init.)

- « Compression » interpretation: \( X \) is « summarized » by \( \theta \)

- Probabilistic / Generative interpretation (where is \( z \))?

\[
\begin{align*}
p(z_n, k = 1) &= \frac{1}{K} \\
&\text{where } z_n = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix} \in \{0, 1\}^K \\
p_\theta(x_n | z_n \equiv k) &= \mathcal{N}(x_n; c_k, I) \quad \text{(Gaussian centered on } c_k)\end{align*}
\]
2. Clustering

**K-means: What does it do?**

- It decreases this cost function at each iteration:

\[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \| x_n - c_k \|^2, \quad \theta = \{ c_1, \ldots, c_K \} \]  
- **Non-Convex** (Depends on init.)

- « Compression » interpretation: \( X \) is « summarized » by \( \theta \)

- **Probabilistic / Generative interpretation (where is \( z \))?**

\[
\begin{align*}
p(z_{n,k} = 1) &= \frac{1}{K} \quad \text{where} \quad z_n = \begin{cases} 
1 \\
0
\end{cases} \quad \in \{0, 1\}^K \\
\end{align*}
\]

\[
p_\theta(x_n | z_n = k) = N(x_n ; c_k , I) \quad \text{(Gaussian centered on} \ c_k \text{)}
\]

\[
\Rightarrow p_\theta(X, Z) = \prod_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} N(x_n ; c_k , I)
\]
2. Clustering

K-means: What does it do?

- It decreases this cost function at each iteration:

\[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \| x_n - c_k \|^2, \quad \theta = \{c_1, \ldots, c_K\} \]

  - Non-Convex
  - (Depends on init.)

- « Compression » interpretation: \( X \) is « summarized » by \( \theta \)

- Probabilistic / Generative interpretation (where is \( z \) ?)

\[
\begin{align*}
  p(z_{n,k} = 1) &= \frac{1}{K} \\
  &\text{where } z_n = \begin{pmatrix} \vdots \\ k \rightarrow 1 \end{pmatrix} \in \{0, 1\}^K
  \\
  p_\theta(x_n | z_n = k) &= \mathcal{N}(x_n; c_k, I) \quad (\text{Gaussian centered on } c_k)
\end{align*}
\]

\[ p_\theta(X, Z) = \prod_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \mathcal{N}(x_n; c_k, I) \]

\[ C(X, G, \theta) = -\log p_\theta(Z|X) \]
2. Clustering

K-means: What does it do?

- It decreases this cost function at each iteration:
  \[ C(X, G, \theta) = \sum_{k=1}^{K} \sum_{n \in G_k} \|x_n - c_k\|^2, \quad \theta = \{c_1, \ldots, c_K\} \]

  ▶ Non-Convex
  (Depends on init.)

- « Compression » interpretation: \(X\) is « summarized » by \(\theta\)

- Probabilistic / Generative interpretation (where is \(z\) ?)

  \[
  p(z_{n,k} = 1) = \frac{1}{K} \quad \text{where} \quad z_n = \begin{cases} 1 & \text{if} \quad k \rightarrow \begin{cases} \text{if} \quad k \rightarrow 1 \\ \text{if} \quad \text{if} \quad k \rightarrow 1 \end{cases} \\
  \end{cases} \in \{0, 1\}^K !
  \]

  \[
  p_{\theta}(x_n | z_n = k) = \mathcal{N}(x_n; c_k, I) \quad \text{(Gaussian centered on} \quad c_k \text{)}
  \]

  \[
  \Rightarrow p_{\theta}(X, Z) = \prod_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \mathcal{N}(x_n; c_k, I)
  \]

  \[ C(X, G, \theta) \overset{c}{=} - \log p_{\theta}(Z|X) \]

K-means can be seen as a maximum a posteriori (MAP) approach!
2. Clustering

Gaussian mixture models: a generalization

\[
\begin{align*}
p(z_{n,k} = 1) &= \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1 \\
p_\theta(x_n | z_n \equiv k) &= \mathcal{N}(x; c_k, \Sigma_k) \\
\Rightarrow p_\theta(x_n) &= \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; c_k, \Sigma_k)
\end{align*}
\]

\[
\theta = \{\pi_1, \ldots, \pi_K, \ c_1, \ldots, c_K, \ \Sigma_1, \ldots, \Sigma_K\}
\]
2. Clustering

Gaussian mixture models: a generalization

\[
p(z_{n,k} = 1) = \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1
\]

\[
p_{\theta}(x_n | z_n \equiv k) = N(x; c_k, \Sigma_k)
\]

\[
\Rightarrow p_{\theta}(x_n) = \sum_{k=1}^{K} \pi_k N(x_n; c_k, \Sigma_k)
\]

\[
\theta = \{\pi_1, \ldots, \pi_K, c_1, \ldots, c_K, \Sigma_1, \ldots, \Sigma_K\}
\]

- Find \( \theta \) that maximizes the observed data log-likelihood:

\[
\mathcal{L}_\theta(X) = \log p_{\theta}(X) = \sum_n \log p_{\theta}(x_n)
\]
2. Clustering

Gaussian mixture models: a generalization

\[
\begin{align*}
p(z_{n,k} = 1) &= \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1 \\
p_{\theta}(x_n | z_n \equiv k) &= \mathcal{N}(x; c_k, \Sigma_k) \\
p_{\theta}(x_n) &= \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; c_k, \Sigma_k)
\end{align*}
\]

\[\theta = \{\pi_1, \ldots, \pi_K, c_1, \ldots, c_K, \Sigma_1, \ldots, \Sigma_K\}\]

- Find \(\theta\) that maximizes the observed data log-likelihood:

\[\mathcal{L}_\theta(X) = \log p_{\theta}(X) = \sum_n \log p_{\theta}(x_n)\]

...very hard to solve directly.
2. Clustering

Gaussian mixture models: a generalization

\[ p(z_{n,k} = 1) = \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1 \]

\[ p_{\theta}(x_n | z_n = k) = \mathcal{N}(x; c_k, \Sigma_k) \]

\[ \Rightarrow p_{\theta}(x_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; c_k, \Sigma_k) \]

\[ \theta = \{ \pi_1, \ldots, \pi_K, c_1, \ldots, c_K, \Sigma_1, \ldots, \Sigma_K \} \]

- Find \( \theta \) that maximizes the observed data log-likelihood:

\[ \mathcal{L}_\theta(X) = \log p_{\theta}(X) = \sum_n \log p_{\theta}(x_n) \]

...very hard to solve directly.

- The Expectation-Maximization (EM) algorithm iteratively maximizes the expected complete-data log-likelihood instead:

\[ \theta^{(i+1)} = \arg\max_{\theta} \mathbb{E}_{p_{\theta^{(i)}}}(Z|X) \{ \log p_{\theta}(X, Z) \} \]
2. Clustering

Gaussian mixture models: a generalization

\[
p(z_n,k = 1) = \pi_k, \quad \sum_{k=1}^{K} \pi_k = 1
\]

\[
p_\theta(x_n | z_n \equiv k) = \mathcal{N}(x; c_k, \Sigma_k)
\]

\[\Rightarrow p_\theta(x_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; c_k, \Sigma_k)\]

• Find \( \theta \) that maximizes the observed data log-likelihood:

\[
\mathcal{L}_\theta(X) = \log p_\theta(X) = \sum_n \log p_\theta(x_n)
\]

...very hard to solve directly.

• The Expectation-Maximization (EM) algorithm iteratively maximizes the expected complete-data log-likelihood instead:

\[
\theta^{(i+1)} = \arg\max_{\theta} \mathbb{E}_{p_{\theta^{(i)}}(Z|X)} \{\log p_\theta(X, Z)\}
\]

Repeat until convergence:

• \( r_{n,k}^{(i)} = p_{\theta^{(i-1)}}(z_{n,k} = 1|x_n) \)

• \( \pi_k^{(i)} = \frac{1}{N} \sum_n r_{n,k}^{(i)} = \frac{\bar{r}_k}{N} \)

• \( c_k^{(i)} = \frac{1}{\bar{r}_k} \sum_n r_{n,k}^{(i)} x_n \)

• \( \Sigma_k^{(i)} = \frac{1}{\bar{r}_k} \sum_n r_{n,k}^{(i)} (x_n - c_k^{(i)}) (x_n - c_k^{(i)})^T \)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

Gaussian Mixture Model Expectation-Maximization (GMM EM)
2. Clustering

To go further…

• **Deep clustering**: trains a neural network to project data to a feature space where K-means can be optimally used.
2. Clustering

To go further...

- **Deep clustering**: trains a neural network to project data to a feature space where K-means can be optimally used

- Application to blind speech source separation:

2. Clustering

To go further…

- **Deep clustering:** trains a neural network to project data to a feature space where K-means can be optimally used

- Application to blind speech source separation:
  

- Application to feature learning from images:
  
2. Clustering

Typical applications of clustering

- **Image segmentation** (\( \{x_n\}_{n=1}^N \) are local descriptors)
- **Audio segmentation** (\( \{x_n\}_{n=1}^N \) are sound segment)
- **Data Quantification**
- **DNA sequence analysis**
- **Social network analysis**
- **Anti-spam filters**
- **Species classification** (biology)
- **Medical imaging**
2. Clustering

Typical applications of clustering

Image segmentation (\(\{x_n\}_{n=1}^N\) are local descriptors)

Audio segmentation (\(\{x_n\}_{n=1}^N\) are sound segment)

Data Quantification

DNA sequence analysis

Social network analysis

Anti-spam filters

Medical imaging

Species classification (biology)

Now let’s code! 😊
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times 3} \) be a 3D set of points.
3. Dimensionality Reduction

Let \( \mathbf{X} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times 3} \) be a 3D set of points

\[ \mathbf{Z} = [z_1, \ldots, z_N]^\top \in \mathbb{R}^{N \times 2} \]

How to reduce its dimensionality while preserving most of its information?
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times 3} \) be a 3D set of points. How to reduce its dimensionality while preserving most of its information? Projected it along axes of maximal variance.

\[
\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^T \in \mathbb{R}^{N \times 2}
\]
3. Dimensionality Reduction

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times 3}$ be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find $\mathbf{v}_1 \in \mathbb{R}^3$ such that $\text{Var}(\mathbf{X}\mathbf{v}_1) = \text{Var}(\mathbf{v}_1^\top \mathbf{x}_1, \ldots, \mathbf{v}_1^\top \mathbf{x}_n)^\top) = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{v}_1^\top \mathbf{x}_n|^2$ is largest.
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times 3} \) be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

Projected it along axes of **maximal variance**

1. Find \( \mathbf{v}_1 \in \mathbb{R}^3 \) such that \( \text{Var}(\mathbf{Xv}_1) = \text{Var}( [\mathbf{v}_1^\top \mathbf{x}_1, \ldots, \mathbf{v}_1^\top \mathbf{x}_n]^\top ) = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{v}_1^\top \mathbf{x}_n|^2 \) is largest
   
   The solution is given by the eigenvector associated to the largest eigenvalue \( \lambda_1 \) of the sample covariance matrix \( \mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{\mu})(\mathbf{x}_n - \mathbf{\mu})^\top \in \mathbb{R}^{3 \times 3} \).
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times 3} \) be a 3D set of points

\[
\mathbf{Z} = [z_1, \ldots, z_N]^\top \in \mathbb{R}^{N \times 2}
\]

How to reduce its dimensionality while preserving most of its information?

1. Find \( \mathbf{v}_1 \in \mathbb{R}^3 \) such that \( \text{Var}(\mathbf{X} \mathbf{v}_1) = \text{Var}(\mathbf{v}_1^\top \mathbf{x}_1, \ldots, \mathbf{v}_1^\top \mathbf{x}_n)^\top = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{v}_1^\top \mathbf{x}_n|^2 \) is largest

- The solution is given by the eigenvector associated to the largest eigenvalue \( \lambda_1 \) of the sample covariance matrix \( \mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^\top \in \mathbb{R}^{3 \times 3} \).
3. Dimensionality Reduction

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times 3}$ be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find $\mathbf{v}_1 \in \mathbb{R}^3$ such that $\text{Var}(\mathbf{X}\mathbf{v}_1) = \text{Var}([\mathbf{v}_1^\top \mathbf{x}_1, \ldots, \mathbf{v}_1^\top \mathbf{x}_n]^\top) = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{v}_1^\top \mathbf{x}_n|^2$ is largest

- The solution is given by the eigenvector associated to the largest eigenvalue $\lambda_1$ of the sample covariance matrix $\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\top \in \mathbb{R}^{3 \times 3}$.
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N\times3} \) be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find \( \mathbf{v}_1 \in \mathbb{R}^3 \) such that 
   \[
   \text{Var}(\mathbf{X}\mathbf{v}_1) = \text{Var}([v_1^\top \mathbf{x}_1, \ldots, v_1^\top \mathbf{x}_n]^\top) = \frac{1}{N} \sum_{n=1}^{N} |v_1^\top \mathbf{x}_n|^2
   \]
   is largest.
   - The solution is given by the eigenvector associated to the largest eigenvalue \( \lambda_1 \) of the sample covariance matrix \( \mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^\top \in \mathbb{R}^{3\times3} \).

2. Find \( \mathbf{v}_2 \perp \mathbf{v}_1 \) such that 
   \[
   \text{Var}(\mathbf{X}\mathbf{v}_2)
   \]
   is largest.
   - Second dominant eigenvector of \( \mathbf{C} \).
Let $X = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times 3}$ be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find $v_1 \in \mathbb{R}^3$ such that $\text{Var}(Xv_1) = \text{Var}([v_1^\top x_1, \ldots, v_1^\top x_n]^\top) = \frac{1}{N} \sum_{n=1}^N |v_1^\top x_n|^2$ is largest.
   - The solution is given by the eigenvector associated to the largest eigenvalue $\lambda_1$ of the sample covariance matrix $C = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)(x_n - \mu)^\top \in \mathbb{R}^{3 \times 3}$.

2. Find $v_2 \perp v_1$ such that $\text{Var}(Xv_2)$ is largest.
   - Second dominant eigenvector of $C$.

$Z = [z_1, \ldots, z_N]^\top \in \mathbb{R}^{N \times 2}$

$Z^{(1)} = Xv_1$
3. Dimensionality Reduction

Let $X = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times 3}$ be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find $v_1 \in \mathbb{R}^3$ such that $\text{Var}(Xv_1) = \text{Var}([v_1^\top x_1, \ldots, v_1^\top x_n]^\top) = \frac{1}{N} \sum_{n=1}^{N} |v_1^\top x_n|^2$ is largest.
   - The solution is given by the eigenvector associated to the largest eigenvalue $\lambda_1$ of the sample covariance matrix $C = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^\top \in \mathbb{R}^{3 \times 3}$.

2. Find $v_2 \perp v_1$ such that $\text{Var}(Xv_2)$ is largest. 
   - Second dominant eigenvector of $C$.

3. Find $v_3 \perp [v_1, v_2]$ ... etc.
Let $X = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times 3}$ be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find $v_1 \in \mathbb{R}^3$ such that $\text{Var}(Xv_1) = \text{Var}([v_1^\top x_1, \ldots, v_1^\top x_n]^\top) = \frac{1}{N} \sum_{n=1}^{N} |v_1^\top x_n|^2$ is largest.
   - The solution is given by the eigenvector associated to the largest eigenvalue $\lambda_1$ of the sample covariance matrix $C = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^\top \in \mathbb{R}^{3 \times 3}$.

2. Find $v_2 \perp v_1$ such that $\text{Var}(Xv_2)$ is largest.
   - Second dominant eigenvector of $C$.

3. Find $v_3 \perp [v_1, v_2]$ ... etc.
   - The Principal Axes of $X$ are the $P$ dominant eigenvectors of $C$.
   - The eigenvalues $\lambda$ are variances along these axes.
3. Dimensionality Reduction

Let \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times 3} \) be a 3D set of points.

How to reduce its dimensionality while preserving most of its information?

1. Find \( \mathbf{v}_1 \in \mathbb{R}^3 \) such that \( \text{Var}(\mathbf{Xv}_1) = \text{Var}([\mathbf{v}_1^\top \mathbf{x}_1, \ldots, \mathbf{v}_1^\top \mathbf{x}_n]^\top) = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{v}_1^\top \mathbf{x}_n|^2 \) is largest.
   ▶ The solution is given by the eigenvector associated to the largest eigenvalue \( \lambda_1 \) of the sample covariance matrix \( \mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{\mu})(\mathbf{x}_n - \mathbf{\mu})^\top \in \mathbb{R}^{3 \times 3} \).

2. Find \( \mathbf{v}_2 \perp \mathbf{v}_1 \) such that \( \text{Var}(\mathbf{Xv}_2) \) is largest. ▶ Second dominant eigenvector of \( \mathbf{C} \).

3. Find \( \mathbf{v}_3 \perp [\mathbf{v}_1, \mathbf{v}_2] \) ... etc.
   ▶ The **Principal Axes** of \( \mathbf{X} \) are the \( P \) dominant eigenvectors of \( \mathbf{C} \).
   ▶ The eigenvalues \( \lambda \) are **variances** along these axes.

\[ \mathbf{Z} = [z_1, \ldots, z_N]^\top \in \mathbb{R}^{N \times 2} \]

\[ \mathbf{Z}^{(1)} = \mathbf{Xv}_1 \]

Projected it along axes of **maximal variance**.
Principality Component Analysis

- Probabilistic / Generative interpretation:

\[
p_{\theta}(z_n) = \mathcal{N}(z_n; 0_P, \Lambda - \sigma^2 I_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P
\]
3. Dimensionality Reduction

Principal Component Analysis

- Probabilistic / Generative interpretation:

\[
\begin{align*}
    p_\theta(z_n) &= \mathcal{N}(z_n; 0_P, \Lambda - \sigma^2 I_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P \\
    p_\theta(x_n|z_n) &= \mathcal{N}(x_n; Vz_n + \mu, \sigma^2 I_D), \quad V = [v_1, \ldots, v_P] \in \mathbb{R}^{D \times P}, \\
    \theta &= \{\Lambda, \sigma^2, V\}
\end{align*}
\]
3. Dimensionality Reduction

### Principal Component Analysis

- Probabilistic / Generative interpretation:

\[
\begin{align*}
p_\theta(z_n) &= \mathcal{N}(z_n; \mathbf{0}_P, \Lambda - \sigma^2 \mathbf{I}_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P \\
p_\theta(x_n|z_n) &= \mathcal{N}(x_n; \mathbf{V}z_n + \mu, \sigma^2 \mathbf{I}_D), \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_P] \in \mathbb{R}^{D \times P}, \\
\theta &= \{\Lambda, \sigma^2, \mathbf{V}\}
\end{align*}
\]

PCA is equivalent to:

- \(\hat{\theta} = \arg\max_{\theta} \log p_\theta(\mathbf{X})\) (Maximum Likelihood)
- \(\hat{z}_n = \arg\max_{z_n} p_{\hat{\theta}}(z_n|x_n)\) (Maximum a posteriori)

---

3. Dimensionality Reduction

Principal Component Analysis

- Probabilistic / Generative interpretation:

\[
\begin{align*}
p_\theta(z_n) &= \mathcal{N}(z_n; 0_P, \Lambda - \sigma^2 I_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P \\
p_\theta(x_n|z_n) &= \mathcal{N}(x_n; V z_n + \mu, \sigma^2 I_D), \quad V = [v_1, \ldots, v_P] \in \mathbb{R}^{D \times P}, \quad \theta = \{\Lambda, \sigma^2, V\}
\end{align*}
\]

PCA is equivalent to:

- \( \hat{\theta} = \arg\max_{\theta} \log p_\theta(X) \) (Maximum Likelihood)
- \( \hat{z}_n = \arg\max_{z_n} p_\hat{\theta}(z_n|x_n) \) (Maximum a posteriori)

- Variational Auto-Encoders


3. Dimensionality Reduction

Principal Component Analysis

- Probabilistic / Generative interpretation:

\[
\begin{align*}
    p_\theta(z_n) &= \mathcal{N}(z_n; \mathbf{0}_P, \Lambda - \sigma^2 \mathbf{I}_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P \\
    p_\theta(x_n|z_n) &= \mathcal{N}(x_n; Vz_n + \mu, \sigma^2 \mathbf{I}_D), \quad V = [v_1, \ldots, v_P] \in \mathbb{R}^{D \times P},
\end{align*}
\]

PCA is equivalent to:

- \( \hat{\theta} = \arg\max_{\theta} \log p_\theta(X) \) (Maximum Likelihood)
- \( \hat{z}_n = \arg\max_{z_n} p_\hat{\theta}(z_n|x_n) \) (Maximum a posteriori)

- Variational Auto-Encoders
  - Generalize PCA by replacing this by a neural network (the decoder)


3. Dimensionality Reduction

Principal Component Analysis

- Probabilistic / Generative interpretation:
  \[
  \begin{align*}
  p_\theta(z_n) &= \mathcal{N}(z_n; 0_P, \Lambda - \sigma^2 I_P), \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_P), \quad \sigma^2 \leq \lambda_P \\
  p_\theta(x_n|z_n) &= \mathcal{N}(x_n; V z_n + \mu, \sigma^2 I_D), \quad V = [v_1, \ldots, v_P] \in \mathbb{R}^{D \times P}, \\
  \theta &= \{\Lambda, \sigma^2, V\}
  \end{align*}
  \]

PCA is equivalent to:
- \( \hat{\theta} = \arg\max_\theta \log p_\theta(X) \) (Maximum Likelihood)
- \( \hat{z}_n = \arg\max_{z_n} p_{\hat{\theta}}(z_n|x_n) \) (Maximum a posteriori)

- Variational Auto-Encoders
  - Generalize PCA by replacing this by a neural network (the decoder)
  - Optimized using a variational approximation of \( p_\theta(z|x) \) by another neural network (the encoder)

---


Non-Linear Dimension Reduction (Manifold Learning)

- Local Tangent Space Alignment (LTSA)

Non-Linear Dimension Reduction (Manifold Learning)

- Local Tangent Space Alignment (LTSA)

1. Builds local k-nearest neighborhoods on the data

3. Dimensionality Reduction

**Non-Linear Dimension Reduction** *(Manifold Learning)*

- **Local Tangent Space Alignment (LTSA)**

  1. Builds local k-nearest neighborhoods on the data
  2. Applies PCA to each neighborhood

---

3. Dimensionality Reduction

Non-Linear Dimension Reduction \textit{(Manifold Learning)}

- Local Tangent Space Alignment (LTSA)

\begin{itemize}
  \item 1. Builds local k-nearest neighborhoods on the data
  \item 2. Applies PCA to each neighborhood
  \item 3. Patch the local PCAs together
\end{itemize}

3. Dimensionality Reduction

Non-Linear Dimension Reduction (*Manifold Learning*)

- Graph-Based Methods: *Isomap, LLE, Laplacian Eigenmap,…*

---

3. Dimensionality Reduction

Non-Linear Dimension Reduction *(Manifold Learning)*

- Graph-Based Methods: *Isomap, LLE, Laplacian Eigenmap, …*
  - Build a *neighborhood graph* from the data
  - « *Unroll* » the graph to a lower dimensional space

---

3. Dimensionality Reduction

Non-Linear Dimension Reduction (*Manifold Learning*)

- Graph-Based Methods: *Isomap, LLE, Laplacian Eigenmap, ...*
  
  - Build a *neighborhood graph* from the data
  
  - « *Unroll* » the graph to a lower dimensional space
  
  - Ex: *Isomap*. Compute all *geodesic distances* (shortest paths) on the graph

---

3. Dimensionality Reduction

To go further...


+ extensions (Normalizing Flows, Glow…)
3. Dimensionality Reduction

To go further…


+ extensions (Normalizing Flows, Glow…)

Typical applications:

Dataset Visualization

Data generation (Glow)

Pre-processing to speed up learning

Compression
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)
Short introduction to sparsity

- A vector is said to be $S$-sparse if it has at most $S$ non-zero elements:

$$z = \begin{bmatrix} 1.7 \\ -216 \\ 5.4 \end{bmatrix} \in \mathbb{R}^9$$

is $3$-sparse.
Short introduction to sparsity

- A vector is said to be \textit{\(S\)-sparse} if it has at most \(S\) non-zero elements:

\[ z = \begin{bmatrix} 1.7 \\ -216 \\ 5.4 \end{bmatrix} \in \mathbb{R}^9 \text{ is } 3\text{-sparse.} \]

- A vector \( \mathbf{x} \in \mathbb{R}^D \) is said to be \textit{sparse in a basis} \( \mathbf{D} \in \mathbb{R}^{D \times K} \) if there exist a sparse vector \( \mathbf{z} \in \mathbb{R}^K \) such that \( \mathbf{x} = \mathbf{D}\mathbf{z} \).
4. Dictionary Learning

Short introduction to sparsity

- A vector is said to be $S$-sparse if it has at most $S$ non-zero elements:

$$z = \begin{bmatrix} 1.7 \\ -216 \\ 5.4 \end{bmatrix} \in \mathbb{R}^3$$ is $3$-sparse.

- A vector $x \in \mathbb{R}^D$ is said to be sparse in a basis $D \in \mathbb{R}^{D \times K}$ if there exist a sparse vector $z \in \mathbb{R}^K$ such that $x = Dz$.
Short introduction to sparsity

- A vector is said to be $S$-sparse if it has at most $S$ non-zero elements:

\[
\text{Example : } z = \begin{pmatrix} 1.7 \\ -216 \\ 5.4 \end{pmatrix} \in \mathbb{R}^9 \text{ is 3-sparse.}
\]

- A vector $x \in \mathbb{R}^D$ is said to be sparse in a basis $D \in \mathbb{R}^{D \times K}$ if there exist a sparse vector $z \in \mathbb{R}^K$ such that $x = Dz$:
4. Dictionary Learning

Short introduction to sparsity

- A vector is said to be \textit{S-sparse} if it has at most \( S \) non-zero elements:

\[
\text{Example: } z = \begin{bmatrix} 1.7 \\ -216 \\ 5.4 \end{bmatrix} \in \mathbb{R}^3 \text{ is 3-sparse.}
\]

- A vector \( x \in \mathbb{R}^D \) is said to be \textit{sparse in a basis} \( D \in \mathbb{R}^{D \times K} \) if there exist a sparse vector \( z \in \mathbb{R}^K \) such that \( x = Dz \):

\[
x = z_3 d_3 + z_7 d_7 + z_{10} d_{10} + z_{11} d_{11}
\]

- \( x \) is a \textit{linear combination} of a few columns (called \textit{atoms}) of \( D \).
Sparse Coding

- We call **sparse coding** the task of recovering a sparse \( z \) **given** a dictionary \( \mathbf{D} \) and an observed noisy signal \( x = \mathbf{D}z + e \).
4. Dictionary Learning

Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ given a dictionary $D$ and an observed noisy signal $x = Dz + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

  $\hat{z}_{\text{lasso}} = \arg\min_{z} \|x - Dz\|_2^2 \quad \text{s.t.} \quad \|z\|_1 \leq \epsilon$
Sparse Coding

• We call **sparse coding** the task of recovering a sparse $z$ **given** a dictionary $\mathbf{D}$ and an observed noisy signal $\mathbf{x} = \mathbf{D}z + e$.

• **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

$$
\hat{z}_{\text{lasso}} = \arg\min_{z} \| \mathbf{x} - \mathbf{D}z \|_2^2 \quad \text{s.t.} \quad \| z \|_1 \leq \epsilon
$$

The $L_1$-norm $\sum_{K=1}^{K} |z_k|$ is a convex proxy for the $L_0$-(non-zero-counting) norm.
Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ given a dictionary $D$ and an observed noisy signal $x = Dz + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

$\hat{z}_{\text{lasso}} = \arg\min_z \|x - Dz\|_2^2 \text{ s.t. } \|z\|_1 \leq \epsilon$

The $L_1$-norm $\sum_{k=1}^{K} |z_k|$ is a convex proxy for the $L_0$-(non-zero-counting) norm. Can be solved efficiently using coordinate descent or subgradient descent.
4. Dictionary Learning

Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ given a dictionary $D$ and an observed noisy signal $x = Dz + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

$$\hat{z}_{\text{lasso}} = \arg\min_{z} \|x - Dz\|_2^2 \text{ s.t. } \|z\|_1 \leq \epsilon$$

The $L_1$-norm $\sum_{k=1}^{K} |z_k|$ is a convex proxy for the $L_0$-(non-zero-counting) norm. Can be solved efficiently using coordinate descent or subgradient descent.

- **Example 2** of sparse-coding method: Matching Pursuit (A **greedy** method)

1. Find the $1$-sparse vector $\hat{z}^{(1)}$ minimizing $\|x - Dz^{(1)}\|_2^2$
Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ **given** a dictionary $\mathbf{D}$ and an observed noisy signal $\mathbf{x} = \mathbf{D}z + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

  $$\hat{z}_{\text{lasso}} = \arg\min_{\mathbf{z}} \| \mathbf{x} - \mathbf{D}\mathbf{z} \|_2^2 \quad \text{s.t.} \quad \| \mathbf{z} \|_1 \leq \epsilon$$

  The $L_1$-norm $\sum_{k=1}^{K} |z_k|$ is a convex proxy for the $L_0$-(non-zero-counting) norm.

  Can be solved efficiently using coordinate descent or subgradient descent.

- **Example 2** of sparse-coding method: Matching Pursuit (A greedy method)

  1. Find the $1$-sparse vector $\hat{z}^{(1)}$ minimizing $\| \mathbf{x} - \mathbf{D}z^{(1)} \|_2^2$

     ▪ It is proportional to the atom $\hat{d}^{(1)}$ of $\mathbf{D}$ maximizing $|\langle \mathbf{x}, \hat{d}^{(1)} \rangle|$. 

4. Dictionary Learning
Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ given a dictionary $D$ and an observed noisy signal $x = Dz + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

  \[
  \hat{z}_{\text{lasso}} = \underset{z}{\arg\min} \|x - Dz\|_2^2 \quad \text{s.t.} \quad \|z\|_1 \leq \epsilon
  \]

  The $L_1$-norm $\sum_{K=1}^{K} |z_k|$ is a convex proxy for the $L_0$-(non-zero-counting) norm. Can be solved efficiently using coordinate descent or subgradient descent.

- **Example 2** of sparse-coding method: Matching Pursuit (A greedy method)

  1. Find the $1$-sparse vector $\hat{z}^{(1)}$ minimizing $\|x - Dz^{(1)}\|_2^2$
     - It is proportional to the atom $\hat{d}^{(1)}$ of $D$ maximizing $|\langle x, d^{(1)} \rangle|$.  
  2. Remove $D\hat{z}^{(1)}$ from $x$. 

4. Dictionary Learning

Sparse Coding

- We call **sparse coding** the task of recovering a sparse $z$ given a dictionary $D$ and an observed noisy signal $x = Dz + e$.

- **Example 1** of sparse-coding method: LASSO (least absolute shrinkage and selection operator)

  $\hat{z}_{\text{lasso}} = \arg\min_z \|x - Dz\|_2^2 \text{ s.t. } \|z\|_1 \leq \epsilon$

  Can be solved efficiently using coordinate descent or subgradient descent.

- **Example 2** of sparse-coding method: Matching Pursuit (A greedy method)

  1. Find the $1$-sparse vector $\hat{z}^{(1)}$ minimizing $\|x - Dz^{(1)}\|_2^2$
     - It is proportional to the atom $\hat{d}^{(1)}$ of $D$ maximizing $|\langle x, d^{(1)} \rangle|$.

  2. Remove $D\hat{z}^{(1)}$ from $x$.

  3. Continue like this until a desired sparsity or reconstruction error is reached.
Typical applications of sparse coding

\[ x = Dz + e \]

Noisy observation in signal space \( \subseteq \mathbb{R}^D \)

Dictionary \( \in \mathbb{R}^{D \times K} \)

Observation noise & modeling error \( \in \mathbb{R}^D \)

- RGB Image
- Hyperspectral Image
- MRI, EEG, EMG,…
- Audio waveform
- Spectrogram
- …

- Discrete Fourier coeffs.
- Discrete Cosine coeffs.
- Wavelets
- Curvelets
- …
4. Dictionary Learning

Typical applications of sparse coding

\[ x = Dz + e \]

Noisy image
Typical applications of sparse coding

\[ x = Dz + e \]
Typical applications of sparse coding

\[ x = Dz + e \]
4. Dictionary Learning

Typical applications of sparse coding

\[ x = Dz + e \]

Noisy image

2D-DCT dictionary

+
Typical applications of sparse coding

\[ x = Dz + e \]
Typical applications of sparse coding

\[ x = Dz + e \]
4. Dictionary Learning

**Typical applications of sparse coding**

\[ x = Dz + e \]

Corrupted image = 2D-DCT dictionary + ?
4. Dictionary Learning

Typical applications of sparse coding

\[ x = Dz + e \]

Corrupted image

2D-DCT dictionary

Inpainted image
4. Dictionary Learning

Typical applications of sparse coding

\[ x = Dz + e \]

Where does it work well?
Typical applications of sparse coding

\[ x = Dz + e \]

Corrupted image \[\Rightarrow\] 2D-DCT dictionary \[\Rightarrow\] Inpainted image

Why does it work well?

**Intuition:** because “natural” images are known to have sparse representations in DCT basis
4. Dictionary Learning

More formally

• \( m \)-term approximation:
  • Let \( x \) be a signal in \( S \)
  • Apply a transform \( \mathcal{T} \) on \( x \)
  • Take the \( m \) leading coefficients
  • Take the inverse transform \( \mathcal{T}^{-1} \) to get \( \hat{x}_m \)
  • Ideally, we want \( \mathcal{T} \) such that
    \[
    f_{\mathcal{T}}(m) = \max_{\mathcal{X} \in S} \| x - \hat{x}_m \|_2^2
    \]
    is small
More formally

- \( m \)-term approximation:
  - Let \( x \) be a signal in \( S \)
  - Apply a transform \( T \) on \( x \)
  - Take the \( m \) leading coefficients
  - Take the inverse transform \( T^{-1} \) to get \( \hat{x}_m \)
  - Ideally, we want \( T \) such that \( f_T(m) = \max_{x \in S} \| x - \hat{x}_m \|_2^2 \) is small

- Some theoretical results for images [Candès and Donoho 2002]
  - \( S = \text{“piecewise } C^2 \text{ with } C^2 \text{ edges”} \)
  - Fourier (similar to DCT): \( f_T(m) \lesssim \frac{1}{\sqrt{m}} \)
  - Wavelet: \( f_T(m) \lesssim \frac{1}{m} \)
  - Curvelets: \( f_T(m) \lesssim \frac{\log^3(m)}{m^2} \) (but results not so good in practice)
4. Dictionary Learning

More formally

- \( m \)-term approximation:
  - Let \( x \) be a signal in \( S \)
  - Apply a transform \( \mathcal{T} \) on \( x \)
  - Take the \( m \) leading coefficients
  - Take the inverse transform \( \mathcal{T}^{-1} \) to get \( \hat{x}_m \)
  - Ideally, we want \( \mathcal{T} \) such that \( f_\mathcal{T}(m) = \max_{x \in S} \| x - \hat{x}_m \|_2^2 \) is small

- Some theoretical results for images [Candès and Donoho 2002]
  - \( S = \text{“piecewise } C^2 \text{ with } C^2 \text{ edges}” \)
  - **Fourier** (similar to DCT): \( f_\mathcal{T}(m) \lesssim \frac{1}{\sqrt{m}} \)
  - **Wavelet**: \( f_\mathcal{T}(m) \lesssim \frac{1}{m} \)
  - **Curvelets**: \( f_\mathcal{T}(m) \lesssim \frac{\log^3(m)}{m^2} \) (but results not so good in practice)
4. Dictionary Learning

More formally

- \( m \)-term approximation:
  - Let \( x \) be a signal in \( S \)
  - Apply a transform \( \mathcal{T} \) on \( x \)
  - Take the \( m \) leading coefficients
  - Take the inverse transform \( \mathcal{T}^{-1} \) to get \( \hat{x}_m \)
  - Ideally, we want \( \mathcal{T} \) such that \( f_{\mathcal{T}}(m) = \max_{x \in S} \| x - \hat{x}_m \|_2^2 \) is small

- Some theoretical results for images [Candès and Donoho 2002]
  - \( S = " \text{piecewise } C^2 \text{ with } C^2 \text{ edges}" \)
  - **Fourier** (similar to DCT): \( f_{\mathcal{T}}(m) \lesssim \frac{1}{\sqrt{m}} \)
  - **Wavelet**: \( f_{\mathcal{T}}(m) \lesssim \frac{1}{m^2} \)
  - **Curvelets**: \( f_{\mathcal{T}}(m) \lesssim \frac{m \log^3(m)}{m^2} \) (but results not so good in practice)

- Predefined dictionaries (DCT, wavelets,...) are used because they are quite good and fast to compute… but they are not necessarily optimal for the signal space considered

Best theoretically achievable

\[
f_{\mathcal{T}}(m) \lesssim \frac{1}{m^2}
\]
4. Dictionary Learning

More formally

- $m$-term approximation:
  - Let $x$ be a signal in $S$
  - Apply a transform $\mathcal{T}$ on $x$
  - Take the $m$ leading coefficients
  - Take the inverse transform $\mathcal{T}^{-1}$ to get $\hat{x}_m$
  - Ideally, we want $\mathcal{T}$ such that $f_\mathcal{T}(m) = \max_{x \in S} \|x - \hat{x}_m\|_2^2$ is small

- Some theoretical results for images [Candès and Donoho 2002]
  - $S = \text{"piecewise } C^2 \text{ with } C^2 \text{ edges}"
  - Fourier (similar to DCT): $f_\mathcal{T}(m) \lesssim \frac{1}{\sqrt{m}}$
  - Wavelet: $f_\mathcal{T}(m) \lesssim \frac{1}{m}$
  - Curvelets: $f_\mathcal{T}(m) \lesssim \frac{m \log^3(m)}{m^2}$ (but results not so good in practice)

- Predefined dictionaries (DCT, wavelets,...) are used because they are quite good and fast to compute... but they are not necessarily optimal for the signal space considered

Can we learn a good dictionary from a set of training signals?
4. Dictionary Learning

Image example

Training database
4. Dictionary Learning

Image example

Training database

Patch extraction
4. Dictionary Learning

Image example

Training database

Patch extraction

\[ x_n \]

\[ 1 \leq n \leq N \]
4. Dictionary Learning

Image example

$\mathbf{x}_n \approx \mathbf{Dz}_n \quad 1 \leq n \leq N$

Patch extraction

Training database
4. Dictionary Learning

Image example

Training database

Patch extraction

\[ x_n \approx Dz_n \quad 1 \leq n \leq N \]

Unknown dictionary

Unknown Sparse coding
4. Dictionary Learning

Image example

Training database

In matrix form: $X \approx D \times Z$

\[
x_n \approx Dz_n \quad 1 \leq n \leq N
\]

Patch extraction

Unknown dictionary & Unknown Sparse coding
4. Dictionary Learning

Image example

Training database

Patch extraction

\[ x_n \approx D z_n \quad 1 \leq n \leq N \]

In matrix form:

\[
\begin{align*}
X & \approx D \times Z \\
D \times N & \quad D \times K & \quad K \times N
\end{align*}
\]
4. Dictionary Learning

Image example

Training database

Patch extraction

\[ x_n \approx D z_n \quad 1 \leq n \leq N \]

Unknown dictionary & Unknown Sparse coding

In matrix form: \[ X \approx D \times Z \]

\[ X = \begin{pmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_N \end{pmatrix} \]
\[ D = \begin{pmatrix} \mathbf{d}_1 & \ldots & \mathbf{d}_K \end{pmatrix} \]
\[ Z = \begin{pmatrix} \mathbf{z}_1 \ldots \mathbf{z}_N \end{pmatrix} \]
4. Dictionary Learning

**Image example**

Training database

---

Patch extraction

\[ \mathbf{x}_n \approx \mathbf{D}\mathbf{z}_n \quad 1 \leq n \leq N \]

**In matrix form:**

\[ \mathbf{X} \approx \mathbf{D} \times \mathbf{Z} \]

\[ \begin{array}{c|c|c}
D & N & D \\
\hline
N & K & K \\
\hline
\end{array} \]

**Unknown dictionary & Unknown Sparse coding**
4. Dictionary Learning

Image example

In matrix form: \( X \approx D \times Z \)

Patch extraction

\[ x_n \approx D z_n \quad 1 \leq n \leq N \]

Unknown dictionary & Unknown Sparse coding
Fitting a union of subspaces

\[ X \approx D \times Z \]

\[ D \times N \quad D \times K \quad K \times N \]
4. Dictionary Learning

Fitting a union of subspaces

\[ X \approx D \times Z \]

\[ D \times N \quad D \times K \quad K \times N \]

\[ x_1 \approx \ldots \times z_1 \]
4. Dictionary Learning

Fitting a union of subspaces

\[ X \approx D \times Z \]

\[ D \times N \quad D \times K \quad K \times N \]
4. Dictionary Learning

Fitting a union of subspaces

\[ X \approx D \times Z \]

\[ D \times N \quad D \times K \quad K \times N \]
4. Dictionary Learning

Matrix factorization: a General Framework

$$\arg\min_{D,Z} \|X - DZ\|_F^2 \quad \text{s.t. } z_n \in \mathcal{C}_z \ \forall n, \quad D \in \mathcal{C}_D$$
Matrix factorization: a General Framework

\[ \arg\min_{D,Z} \|X - DZ\|_F^2 \quad \text{s.t.} \quad z_n \in \mathcal{C}_z \quad \forall n, \quad D \in \mathcal{C}_D \]

- **Dictionary Learning**
  - \( \mathcal{C}_z \): Sparse
  - \( \mathcal{C}_D \): Unit-norm columns (typically \( D < K \))
4. Dictionary Learning

Matrix factorization: a General Framework

\[
\arg\min_{D,Z} \|X - DZ\|_F^2 \quad \text{s.t.} \quad z_n \in \mathcal{C}_z \ \forall n, \quad D \in \mathcal{C}_D
\]

- **Dictionary Learning**
  - $\mathcal{C}_z$: Sparse
  - $\mathcal{C}_D$: Unit-norm columns (typically $D < K$)

- **K-mean clustering**
  - $\mathcal{C}_z$: Canonical basis vectors
  - $\mathcal{C}_D$: No constraint
4. Dictionary Learning

Matrix factorization: a General Framework

\[
\text{argmin}_{D,Z} \|X - DZ\|_F^2 \text{ s.t. } z_n \in C_z \forall n, \ D \in C_D
\]

• Dictionary Learning
  • \(C_z\): Sparse
  • \(C_D\): Unit-norm columns (typically \(D < K\))

• K-mean clustering
  • \(C_z\): Canonical basis vectors
  • \(C_D\): No constraint

• Principal component analysis
  • \(C_z\): No constraint
  • \(C_D\): Orthonormal columns
Matrix factorization: a General Framework

\[
\arg\min_{D,Z} \|X - DZ\|_F^2 \text{ s.t. } z_n \in \mathcal{C}_z \ \forall n, \ D \in \mathcal{C}_D
\]

- **Dictionary Learning**
  - \( \mathcal{C}_z \): Sparse
  - \( \mathcal{C}_D \): Unit-norm columns (typically \( D < K \))

- **K-mean clustering**
  - \( \mathcal{C}_z \): Canonical basis vectors
  - \( \mathcal{C}_D \): No constraint

- **Principal component analysis**
  - \( \mathcal{C}_z \): No constraint
  - \( \mathcal{C}_D \): Orthonormal columns

- **Non-negative matrix factorization**
  - \( \mathcal{C}_z \): Non-negative coefficients
  - \( \mathcal{C}_D \): Unit-norm non-negative coefficients
## The K-SVD Algorithm

<table>
<thead>
<tr>
<th>K-mean</th>
<th>K-SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### The K-SVD Algorithm

#### K-mean

**Objective:** 
\[
\forall C, g_1, \ldots, g_K \sum_{k=1}^{K} \sum_{x_n \in g_k} \| x_n - c_k \|_2^2
\]

#### K-SVD
# The K-SVD Algorithm

## K-mean vs. K-SVD

<table>
<thead>
<tr>
<th>Objective: $\arg\min_{C, g_1, \ldots, g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} |x_n - c_k|_2^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\iff \arg\min_{C, z_1, \ldots, z_N} \sum_{n=1}^{N} |x_n - Cz_n|_2^2$ s.t. $z_n \in \mathcal{E}$ $\forall n$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K-mean</th>
<th>K-SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\arg\min_{C, g_1, \ldots, g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} |x_n - c_k|_2^2$</td>
<td>$\arg\min_{C, z_1, \ldots, z_N} \sum_{n=1}^{N} |x_n - Cz_n|_2^2$ s.t. $z_n \in \mathcal{E}$ $\forall n$</td>
</tr>
</tbody>
</table>
## The K-SVD Algorithm

### K-mean

**Objective:**

\[
\arg\min_{C, g_1, \ldots, g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} \|x_n - c_k\|_2^2
\]

\[
\Leftrightarrow \arg\min_{C, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Cz_n\|_2^2 \quad \text{s.t.} \quad z_n \in \mathcal{E} \quad \forall n
\]

### K-SVD

![Diagram of K-SVD]

The K-SVD algorithm combines dictionary learning with K-means clustering, where the dictionary matrix \(C\) is updated to minimize the reconstruction error, and the cluster assignments \(z_n\) are determined based on the nearest dictionary atom for each data point \(x_n\). This iterative process leads to a compact representation of the data in terms of the learned dictionary.
The K-SVD Algorithm

### K-mean

**Objective:**
\[
\text{argmin}_{C,g_1,\ldots,g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} \|x_n - c_k\|_2^2
\]

\[
\Leftrightarrow \text{argmin}_{C,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - C z_n\|_2^2 \text{ s.t. } z_n \in \mathcal{E} \forall n
\]

**Algorithm:**
- Initialize \( C \)
- repeat until convergence:
  - Assign \( x_n \) to nearest centroid (Find support of each \( z_n \))
  - Given assignments (support of \( Z \)), update centroids in \( C \) (\( K \) means)
- end

### K-SVD
4. Dictionary Learning

The K-SVD Algorithm

K-mean

Objective: \[ \arg\min_{C,g_1,...,g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} \|x_n - c_k\|_2^2 \]

\[ \iff \quad \arg\min_{C,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Cz_n\|_2^2 \text{ s.t. } z_n \in E \forall n \]

Algorithm:
- Initialize \( C \)
- repeat until convergence:
  - Assign \( x_n \) to nearest centroid (Find support of each \( z_n \))
  - Given assignments (support of \( Z \)), update centroids in \( C \) (K means)
- end

K-SVD

Objective: \[ \arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\max} \forall n \]
4. Dictionary Learning

The K-SVD Algorithm

**K-mean**

Objective: \( \arg \min_{C, g_1, \ldots, g_K} \sum_{k=1}^{K} \sum_{x_n \in g_k} \|x_n - c_k\|_2^2 \)

\( \iff \arg \min_{C, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Cz_n\|_2^2 \) s.t. \( z_n \in \mathcal{E} \ \forall n \)

Algorithm:
- Initialize \( C \)
- repeat until convergence:
  - Assign \( x_n \) to nearest centroid (Find support of each \( z_n \))
  - Given assignments (support of \( Z \)), update centroids in \( C \) (K means)
- end

**K-SVD**

Objective: \( \arg \min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \) s.t. \( \|z_n\|_0 \leq S_{\text{max}} \ \forall n \)

Algorithm:
- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using *sparse coding*
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \) (K Singular Value Decompositions)
- end
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[
\text{argmin}_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \| x_n - D z_n \|_2^2 \text{ s.t. } \| z_n \|_0 \leq S_{\text{max}} \quad \forall n
\]

K-SVD algorithm:

- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[
X
\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \( \operatorname{argmin}_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \| x_n - D z_n \|_2^2 \) s.t. \( \| z_n \|_0 \leq S_{\text{max}} \forall n \)

K-SVD algorithm:
- **Initialize** \( D \)
- **repeat** until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- **end**
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\max} \forall n \]

K-SVD algorithm:

- **Initialize** \(D\)
- **repeat** until convergence:
  - Find support and coefficients of each \(z_n\) using **sparse coding**
  - Given support of \(Z\), update atoms in \(D\) and coefficients in \(Z\)
- **end**

\[ X \quad \begin{array}{c|c}
\hline
\text{X} & \text{D} \\
\hline
\end{array} \]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective:

$$\operatorname*{argmin}_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \left\| x_n - D z_n \right\|_2^2 \text{ s.t. } \| z_n \|_0 \leq S_{\text{max}} \forall n$$

K-SVD algorithm:

- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using **sparse coding**
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end

![Matrix X and D](image)
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\max} \forall n \]

K-SVD algorithm:

- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using \textit{sparse coding} 
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end
4. Dictionary Learning

The K-SVD Algorithm

**K-SVD objective:**
$$\arg\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \| x_n - D z_n \|_2^2 \quad \text{s.t.} \quad \| z_n \|_0 \leq S_{\text{max}} \quad \forall n$$

**K-SVD algorithm:**
- Initialize $D$
- **repeat** until convergence:
  - Find support and coefficients of each $z_n$ using **sparse coding**
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- **end**
The K-SVD Algorithm

K-SVD objective: \[
\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \quad \text{s.t.} \quad \|z_n\|_0 \leq S_{\text{max}} \quad \forall n
\]

K-SVD algorithm:
- Initialize \( D \)
- \textbf{repeat} until convergence:
  - Find support and coefficients of each \( z_n \) using \textit{sparse coding}
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- \textbf{end}
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[
\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n
\]

K-SVD algorithm:

- Initialize \(D\)
- \textbf{repeat} until convergence:
  - Find support and coefficients of each \(z_n\) using \textit{sparse coding}
  - Given support of \(Z\), update atoms in \(D\) and coefficients in \(Z\)
- \textbf{end}
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective:
\[
\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \ \forall n
\]

K-SVD algorithm:

- Initialize $D$
- **repeat** until convergence:
  - Find support and coefficients of each $z_n$ using *sparse coding*
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- **end**

\[ X \approx DZ \]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: $\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|^2_2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \ \forall n$

K-SVD algorithm:

- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using sparse coding
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end

$$X \approx DZ$$
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \| x_n - Dz_n \|_2^2 \quad \text{s.t.} \quad \| z_n \|_0 \leq S_{\text{max}} \quad \forall n \]

K-SVD algorithm:

- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - **Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)**
- end
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective:  \[ \arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n \]

K-SVD algorithm:
- Initialize \( D \)
- \textbf{repeat} until convergence:
  - Find support and coefficients of each \( z_n \) using \textit{sparse coding}
  - \textbf{Given support of} \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- \textbf{end}

\( X \) \( \approx \) \( D \) \( Z \)
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n \]

K-SVD algorithm:

- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[ X \approx \begin{array}{c} X_1 \end{array} \quad \begin{array}{c} D \end{array} \quad \begin{array}{c} Z \end{array} \]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: $\text{argmin}_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n$

K-SVD algorithm:
- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using **sparse coding**
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end
4. Dictionary Learning

The K-SVD Algorithm

**K-SVD objective:**

\[
\arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \quad \text{s.t.} \quad \|z_n\|_0 \leq S_{\max} \quad \forall n
\]

**K-SVD algorithm:**

- Initialize \(D\)
- **repeat** until convergence:
  - Find support and coefficients of each \(z_n\) using **sparse coding**
  - **Given support of** \(Z\), update atoms in \(D\) and coefficients in \(Z\)
- **end**

\[X \approx DD^T \approx X_1 \approx Z_1^T\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[
\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \quad \text{s.t.} \quad \|z_n\|_0 \leq S_{\max} \quad \forall n
\]

K-SVD algorithm:

- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using **sparse coding**
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end

\[
(d_1, \tilde{z}_1^\top) := \arg\min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{Z}_1\|_F^2
\]

![Diagram](image)
4. Dictionary Learning

The K-SVD Algorithm

**K-SVD objective:** \( \arg\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \) s.t. \( \|z_n\|_0 \leq S_{\max} \ \forall n \)

**K-SVD algorithm:**

- Initialize \( D \)
- **repeat** until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - **Given support of** \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- **end**

\[
(d_1, \tilde{z}_1^\top) := \arg\min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{Z}_1\|_F^2
\]
\[
:= \arg\min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{Z}_1 + d_1\tilde{z}_1^\top - d_1\tilde{z}_1^\top\|_F^2
\]
The K-SVD Algorithm

**K-SVD objective:**
\[
\arg \min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \quad \text{s.t.} \quad \|z_n\|_0 \leq S_{\max} \quad \forall n
\]

**K-SVD algorithm:**
- Initialize \( D \)
- \textbf{repeat} until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- \textbf{end}

\[
(d_1, \tilde{z}_1^\top) := \arg \min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{Z}_1\|_F^2
\]
\[
:= \arg \min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{Z}_1 + d_1\tilde{z}_1^\top - d_1\tilde{z}_1^\top\|_F^2
\]
\[
:= \arg \min_{d_1, \tilde{z}_1^\top} \|\tilde{E}_1 - d_1\tilde{z}_1^\top\|_F^2
\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg \min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \ \forall n \]

K-SVD algorithm:
- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using \textbf{sparse coding}
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[ (d_1, \tilde{z}_1^T) := \arg \min_{d_1, \tilde{z}_1^T} \|\tilde{X}_1 - D\tilde{Z}_1\|_F^2 \]
\[ := \arg \min_{d_1, \tilde{z}_1^T} \|\tilde{X}_1 - D\tilde{Z}_1 + d_1\tilde{z}_1^T - d_1\tilde{z}_1^T\|_F^2 \]
\[ := \arg \min_{d_1, \tilde{z}_1^T} \|\tilde{E}_1 - d_1\tilde{z}_1^T\|_F^2 \]
\[ (d_1, \tilde{z}_1^T) := \text{svds}(\tilde{E}_1, 1) \]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[
\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \quad \text{s.t.} \quad \|z_n\|_0 \leq S_{\text{max}} \quad \forall n
\]

K-SVD algorithm:

- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using \textit{sparse coding}
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[
(d_1, \tilde{z}_1^\top) := \min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{z}_1\|_F^2
\]
\[
 := \min_{d_1, \tilde{z}_1^\top} \|\tilde{X}_1 - D\tilde{z}_1 + d_1\tilde{z}_1^\top - d_1\tilde{z}_1^\top\|_F^2
\]
\[
 := \min_{d_1, \tilde{z}_1^\top} \|\tilde{E}_1 - d_1\tilde{z}_1^\top\|_F^2
\]

\[
(d_1, \tilde{z}_1^\top) := \text{svds}(\tilde{E}_1, 1)
\]
The K-SVD Algorithm

K-SVD objective: \[
\text{argmin}_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \| x_n - D z_n \|_2^2 \quad \text{s.t.} \quad \| z_n \|_0 \leq S_{\text{max}} \quad \forall n
\]

K-SVD algorithm:

- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using sparse coding
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end

\[\begin{align*}
X & \approx DZ
\end{align*}\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: 
\[
\arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n
\]

K-SVD algorithm:
- Initialize \(D\)
- \textbf{repeat} until convergence:
  - Find support and coefficients of each \(z_n\) using \textit{sparse coding}
  - Given support of \(Z\), update atoms in \(D\) and coefficients in \(Z\)
- \textbf{end}

\[
\begin{array}{ccc}
X \approx & D & Z \\
\end{array}
\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective:

$$\arg\min_{D,z_1,...,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n$$

K-SVD algorithm:

- Initialize $D$
- repeat until convergence:
  - Find support and coefficients of each $z_n$ using **sparse coding**
  - Given support of $Z$, update atoms in $D$ and coefficients in $Z$
- end
The K-SVD Algorithm

K-SVD objective:
\[ \arg\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n \]

K-SVD algorithm:

- Initialize D
- repeat until convergence:
  - Find support and coefficients of each \(z_n\) using sparse coding
  - Given support of \(Z\), update atoms in \(D\) and coefficients in \(Z\)
- end
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n \]

K-SVD algorithm:
- Initialize D
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using \textit{sparse coding}
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[ (d_2, \tilde{z}_2^\top) := \arg\min_{d_2, \tilde{z}_2^\top} \|\tilde{X}_2 - D\tilde{Z}_2\|_F^2 \]
\[ := \arg\min_{d_2, \tilde{z}_2^\top} \|\tilde{X}_2 - D\tilde{Z}_2 + d_2\tilde{z}_2^\top - d_2\tilde{z}_2^\top\|_F^2 \]
\[ := \arg\min_{d_2, \tilde{z}_2^\top} \|\tilde{E}_2 - d_2\tilde{z}_2^\top\|_F^2 \]
\[ (d_2, \tilde{z}_2^\top) := \text{svds}(\tilde{E}_2, 1) \]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[ \arg\min_{D, z_1, \ldots, z_N} \sum_{n=1}^{N} \| x_n - Dz_n \|_2^2 \text{ s.t. } \| z_n \|_0 \leq S_{\text{max}} \forall n \]

K-SVD algorithm:
- Initialize \( D \)
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using \textbf{sparse coding}
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[
(d_2, \tilde{z}_2^\top) := \arg\min_{d_2, \tilde{z}_2^\top} \| \tilde{X}_2 - D\tilde{Z}_2 \|_F^2
\]
\[
:= \arg\min_{d_2, \tilde{z}_2^\top} \| \tilde{X}_2 - D\tilde{Z}_2 + d_2\tilde{z}_2^\top - d_2\tilde{z}_2^\top \|_F^2
\]
\[
:= \arg\min_{d_2, \tilde{z}_2^\top} \| \tilde{E}_2 - d_2\tilde{z}_2^\top \|_F^2
\]

\[
(d_2, \tilde{z}_2^\top) := \text{svds}(\tilde{E}_2, 1)
\]
4. Dictionary Learning

The K-SVD Algorithm

K-SVD objective: \[
\arg\min_{D,z_1,\ldots,z_N} \sum_{n=1}^{N} \|x_n - Dz_n\|_2^2 \text{ s.t. } \|z_n\|_0 \leq S_{\text{max}} \forall n
\]

K-SVD algorithm:

- Initialize D
- repeat until convergence:
  - Find support and coefficients of each \( z_n \) using **sparse coding**
  - Given support of \( Z \), update atoms in \( D \) and coefficients in \( Z \)
- end

\[
(d_k, \tilde{z}_k^\top) := \arg\min_{d_k, \tilde{z}_k^\top} \|\tilde{X}_k - D\tilde{Z}_k\|_F^2
\]
\[
= \arg\min_{d_k, \tilde{z}_k^\top} \|\tilde{X}_k - D\tilde{Z}_k + d_k\tilde{z}_k^\top - d_k\tilde{z}_k^\top\|_F^2
\]
\[
= \arg\min_{d_k, \tilde{z}_k^\top} \|\tilde{E}_k - d_k\tilde{z}_k^\top\|_F^2
\]

\[
(d_k, \tilde{z}_k^\top) := \text{svds}(\tilde{E}_k, 1)
\]
The K-SVD Algorithm

Now let’s code it! 😊
4. Dictionary Learning

The K-SVD Algorithm

Now let’s code it! 😊

To go further...

4. Dictionary Learning

The K-SVD Algorithm

Now let’s code it! 😊

To go further...


Outline

1. Introduction
2. Clustering (Exo3)
3. Dimensionality Reduction (Exo4)
4. Dictionary Learning (Exo5)