Machine Learning

Dmytro Fishman (dmytro.fishman@ut.ee)
# Deadlines

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Date of assignment</th>
<th>Deadline (midnight 23:59)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>Sep 7</td>
<td>Sep 20</td>
</tr>
<tr>
<td>HW2</td>
<td>Sep 21</td>
<td>Oct 4</td>
</tr>
<tr>
<td>HW3</td>
<td>Oct 5</td>
<td>Oct 18</td>
</tr>
<tr>
<td>Paper summary</td>
<td>Oct 12</td>
<td>Oct 26</td>
</tr>
<tr>
<td>HW4</td>
<td>Oct 19</td>
<td>Nov 1</td>
</tr>
<tr>
<td>HW5</td>
<td>Nov 9</td>
<td>Nov 22</td>
</tr>
<tr>
<td>HW6</td>
<td>Nov 23</td>
<td>Dec 6</td>
</tr>
<tr>
<td>Project</td>
<td>TBA</td>
<td>Dec 14</td>
</tr>
</tbody>
</table>
Recap
Supervised Learning

Unsupervised Learning

Reinforcement Learning

Machine Learning
Machine Learning

Classification

Supervised Learning

Regression

Unsupervised Learning

Reinforcement Learning

Machine Learning
K-Nearest Neighbour Classifier

Let's assume $K = 5$
Linear Regression

\[ \hat{y} = w_0 + w_1 x \]

minimises the sum of errors with respect to \( w_0 \) and \( w_1 \)

\[ \arg\min_{w_0, w_1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \]
Decision Tree Algorithm

We choose the **split** that minimises total MSE
Decision Tree Algorithm

Thus, the resulting tree:

```
<table>
<thead>
<tr>
<th>distance</th>
<th>fare amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>True</td>
</tr>
</tbody>
</table>
```

```
MSE = 0.2
Is distance > 1.5
```

```
<table>
<thead>
<tr>
<th>X</th>
<th>fare amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4.5</td>
</tr>
</tbody>
</table>
```
Train/val split

Initial dataset

Simple, but imperfect

Randomly select 60%

MSE = 0.0

MSE = 1.0

Complicated, but ideal

Randomly select 40%

MSE = 2.5

MSE = 0.5

Train dataset

Only true for your data

Validation (val) dataset

Reality
Cross Validation (CV) Algorithm

Simple, but imperfect

Complicated, but ideal

Before each CV loop we shuffle the data to get new random folds

Average MSE = 0.65

Average MSE = 1.4
Supervised Learning pipeline

1. Acquire Data
2. Preprocessing
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set

Safe place

Profit
Machine Learning

- Supervised Learning
- Regression
- Classification
- Reinforcement Learning
- Clustering
- Unsupervised Learning
- Dimensionality reduction
High Dimensional Space

Low Dimensional Space

Dimensionality reduction

Unsupervised Learning

Supervised Learning

Classification

Regression

Reinforcement Learning

Machine Learning

Torus image credit: https://mathematica.stackexchange.com/questions/39879/create-a-torus-with-a-hexagonal-mesh-for-3d-printing
What are the examples of high-dimensional objects?
What are the examples of high-dimensional objects?

Object from 7D

A taxi ride from NYC data

3.5$
What are the examples of high-dimensional objects?

A taxi ride from NYC data

Object from 7D

MNIST digit

3.5$
What are the examples of high-dimensional objects?

A taxi ride from NYC data

Object from 7D

A taxi ride from NYC data

3.5$

Object from 7D

MNIST digit

Object from 784D
What are the examples of high-dimensional objects?

- A taxi ride from NYC data
- Object from **7D**
- MNIST digit
  - Object from **784D**
- ImageNet cat
  - Object from **196608D**

물체의 예를 들어 보자.
What are the examples of high-dimensional objects?

- A taxi ride from NYC data
- LiDAR recording
- MNIST digit
- ImageNet cat

Object from 784D
Object from 7D
Object from 196608D

few million D
3.5$
What are the examples of high-dimensional objects?

- **Two-dimensional objects:**
  - A taxi ride from NYC data
  - MNIST digit
  - Object from 7D

- **High-dimensional objects:**
  - LiDAR recording
  - ImageNet cat
  - Human DNA
  - Object from 784D
  - Object from 196608D

- **Examples of high-dimensional data:**
  - Few million D
  - A few dollars ($3.50)
What are the examples of high-dimensional objects?

- A taxi ride from NYC data
- Object from 7D
- MNIST digit
- Object from 784D
- ImageNet cat
- Object from 196608D
- LiDAR recording
- Few million D
- Human DNA
- Object from 3.6 billion D
What is the problem with high-dimensional things?
What is the problem with high-dimensional things?

**Hard to visualise**
What is the problem with **high-dimensional** things?

**Hard to visualise**
What is the problem with high-dimensional things?

- Hard to visualise
- Algorithms tend to get slow
What is the problem with high-dimensional things?

Hard to visualise

Algorithms tend to get slow

Nearest Neighbour Classifier

Euclidean distance

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Nearest neighbour is found by calculating distances to all existing examples
What is the problem with high-dimensional things?

Nearest Neighbour Classifier

Hard to visualise

Algorithms tend to get slow

Euclidean distance

\[ d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \]

nearest neighbour is found by calculating distances to all existing examples
What is the problem with high-dimensional things?

Hard to visualise

Algorithms tend to get slow

Nearest Neighbour Classifier

Euclidean distance

\[ d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \ldots + (n_2 - n_1)^2} \]

Nearest Neighbour is \( O(n^2) \), but with number of dimensions approaching the number of samples it is \( O(n^3) \)

**nearest neighbour** is found by calculating distances to **all existing examples**
What is the problem with **high-dimensional** things?

- **Hard** to visualise
- **Algorithms** tend to get slow
What is the problem with high-dimensional things?

- Hard to visualise
- Algorithms tend to get slow
- Methods trained on high-dimensional data suffer from the curse of dimensionality
The Curse of Dimensionality
What is the **curse** of **dimensionality**?
What is the **curse** of **dimensionality**?
What is the **curse** of **dimensionality**?
What is the **curse** of **dimensionality**?

![Diagram](image)

- **Blue (75%)**
  - False: 4 points
  - True: 6 points

- **Red (75%)**
  - False: 4 points
  - True: 6 points

**X > 2**
- False (Blue, 75%)
- True (Red, 75%)
What is the **curse of dimensionality**?
What is the **curse** of **dimensionality**?
What is the **curse** of **dimensionality**?
What is the **curse of dimensionality**?
What is the **curse** of **dimensionality**?
What is the **curse of dimensionality**?

On average **55.5%** of cells will be either **empty** or **singletons**.
What is the **curse of dimensionality**?

On average **55.5%** of cells will be either **empty** or **singletons**

On average **92.5%** of cells will be either **empty** or **singletons**
What is the **curse of dimensionality**?

In order to keep **high-dimensional** space reasonably covered you need **a lot more data**.

On average **55.5%** of cells will be either **empty** or **singletons**.

On average **92.5%** of cells will be either **empty** or **singletons**.
What is the **curse of dimensionality**? (part II)
What is the **curse of dimensionality**? (part II)

**Distances** become **similar** in high-dimensional space
Pixel-based distances on high-dimensional data (and images especially) can be very unintuitive. An original image (left) and three other images next to it that are all equally far away from it based on L2 pixel distance. Clearly, the pixel-wise distance does not correspond at all to perceptual or semantic similarity.
What is the problem with high-dimensional things?

- **Hard to visualise**
- **Methods trained** on high-dimensional data suffer from the **curse of dimensionality**
- **Algorithms** tend to get slow
What is the problem with high-dimensional things?

- **Hard to visualise**
- **Algorithms tend to get slow**
- You need **more data** and **objects become closer** in high-dimensional space
Is there a way to break the curse?
Feature **extraction** vs feature **elimination**
Feature extraction vs feature elimination

Keeping only few original features
Feature extraction vs feature elimination

Remove all the rest

Keeping only few original features
Feature **extraction** vs feature **elimination**
Feature extraction vs feature elimination
Feature extraction vs feature elimination
Principle Component Analysis
Principle Component Analysis
1-Dimensional data

\[ x \]

1, 2, 3, 4, 5

\[ X \]
2-Dimensional data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
3-Dimensional data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.5</td>
</tr>
</tbody>
</table>
200-Dimensional data?
Are all of these dimensions equally useful?
2-D example revisited
2-D example revisited

Main variation is from left to right
2-D example revisited

Main variation is from **left** to **right**
2-D example revisited

Main variation is from **left** to **right**

Not so much from **top** to **bottom**
2-D example revisited

Main variation is from **left to right**

Not so much from **top to bottom**

We can **keep only one dimension**
2-D example revisited

- Main variation is from **left** to **right**

- Not so much from **top** to **bottom**

- We can keep only **one dimension**

- Projected data does not seem to lose much **information**
2-D example revisited
2-D example revisited
2-D example revisited
Data seem to be spread more equally along X and y axes.
2-D example revisited
2-D example revisited

Data is **mostly spread** along this line.
2-D example revisited

Data is **mostly spread** along this line. And a **little bit** along this line.
2-D example revisited

How about we make new axes from these lines?

Data is mostly spread along this line

And a little bit along this line
2-D example revisited

How about we make new axes from these lines?

Data is mostly spread along this line

And a little bit along this line
2-D example revisited

How about we make new axes from these lines?

Data is mostly spread along this line. And a little bit along this line.
2-D example revisited

How about we make **new axes** from **these lines**?

Data is **mostly spread** along this **line**

And a **little bit** along this **line**
2-D example revisited

How about we make new axes from these lines?

Data is **mostly spread** along this line

And a **little bit** along this line
2-D example revisited

These **new axes** are called principle components.

Data is mostly spread along this line.
And a little bit along this line.

These new axes are called principle components.
2-D example revisited

These **new axes** are called principle components

Data is mostly spread along this line

And a little bit along this line

PC #1 is a new vector which spans along **most of the variation** in data.
2-D example revisited

These **new axes** are called principle components

Data is mostly spread along this line

And a little bit along this line

PC #1 is a new vector which spans along most of the variation in data

PC #2 is another new vector which spans along the direction of the second most variation
Principle components are **not additional axes/dimensions**
Principle components are **not additional axes/dimensions**

They are **old dimensions** rearranged
Principle components are not additional axes/dimensions

They are old dimensions rearranged

Such that the first axis now spans along most variation, the second the second most variation etc.
Principle components are **not additional axes/dimensions**

**How many** PCs will be in **3D** space?
Principle components are **not additional axes/dimensions**

**How many** PCs will be in 3D space?

As many as there were original dimensions, hence **3** PCs
Principle components are not additional axes/dimensions

How many PCs will be formed in 200D space?
Principle components are **not** additional axes/dimensions

**How many** PCs will be formed in **200D** space?

No exceptions, **200 PCs**
Principle components are not additional axes/dimensions

How many PCs will be formed in 200D space?

No exceptions, 200 PCs
But what is the benefit of having PCs?
Data is **mostly spread** along this line.

And a **little bit** along this **line**.
Data is **mostly spread** along this line.

And a **little bit** along this line.
Data is **mostly spread** along this line

And a **little bit** along this line

From **2D** to **1D** without losing much information
Principle components are **not additional axes/dimensions**

**How many** PCs will be formed in **200D** space?

No exceptions, **200 PCs**

But what is **the benefit** of having PCs?
Principle components are **not additional axes/dimensions**

**How many** PCs will be formed in **200D** space?

First few PCs would be **enough** to capture important information
Computational example of PCA
\[ x̄ = 3 \quad \bar{y} = 4 \]
Transpose the matrix of coordinates

\[
\begin{align*}
Z &= \begin{bmatrix}
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1
\end{bmatrix}
\end{align*}
\]
**Transpose** the matrix of coordinates

\[
\begin{array}{cc}
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1 \\
\end{array}
\]

What are the **dimensions** of the transposed matrix?
**Transpose** the matrix of coordinates

\[
\begin{array}{ll}
Z & \quad Z^T \\
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1 \\
\end{array}
\]
**Transpose** the matrix of coordinates

\[
Z = \begin{array}{cc}
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1 \\
\end{array}
\]

\[
Z^T = \begin{array}{cccc}
-2 & -1 & 0 & 1 \\
-2 & 0 & 1 & 0 \\
\end{array}
\]
\[ Z^\top \times Z = S \]
Matrix multiplication beautifully animated http://matrixmultiplication.xyz/

\[
\begin{align*}
Z^T & = \\
\begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{bmatrix} \\
& \times \\
\begin{bmatrix}
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1
\end{bmatrix} \\
& = \\
S
\end{align*}
\]
\[
Z^T \times Z = S
\]
\[
\begin{align*}
Z^T & = \begin{pmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1 \\
\end{pmatrix} \\
\end{align*}
\]

\[
Z = \begin{pmatrix}
-2 & -2 \\
-1 & 0 \\
0 & 1 \\
1 & 0 \\
2 & 1 \\
\end{pmatrix}
\]

\[
S = \begin{pmatrix}
10 & 6 \\
6 & 6 \\
\end{pmatrix}
\]
\[ Z^T \times \begin{bmatrix} -2 & -2 \\ -1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 1.5 \end{bmatrix} \times 4 \]
The equation depicted in the image shows the calculation of a covariance matrix. The covariance matrix $S$ is obtained by multiplying a matrix $Z^T$ with another matrix $Z$. Both matrices are composed of columns labeled with values from -2 to 2. The final product is scaled by 4.

Specifically, the covariance matrix $S$ is defined as:

$$S = Z^T \times Z \times 4$$

where $Z^T$ is the transpose of the matrix $Z$. Each element in the covariance matrix $S$ is calculated as the product of corresponding elements from $Z^T$ and $Z$, and then multiplied by 4.
How to interpret values in covariance matrix?
How to **interpret** values in **covariance** matrix?
How to **interpret** values in **covariance** matrix?

Collect all projected onto **X axis** values

\[-2, -1, 0, 1, 2\]

\[S\]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>
How to **interpret** values in **covariance** matrix?

\[
[-2, -1, 0, 1, 2]
\]

**mean**([-2, -1, 0, 1, 2]) = ?

Collect all projected onto **X axis** values

\[
\begin{array}{cc}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{array}
\]

**Covariance** matrix
How to interpret values in covariance matrix?

\[ [-2, -1, 0, 1, 2] \]

mean\(([-2, -1, 0, 1, 2]) = 0\)

Collect all projected onto X axis values

\[
\begin{array}{c|c|c}
\text{S} & 2.5 & 1.5 \\
1.5 & 1.5 \\
\end{array}
\]

Covariance matrix
How to interpret values in covariance matrix?

\[
[-2, -1, 0, 1, 2] \quad \bar{x} = 0
\]

Collect all projected onto X axis values:

\[
\begin{array}{cc}
  2.5 & 1.5 \\
  1.5 & 1.5 \\
\end{array}
\]

Covariance matrix
How to interpret values in covariance matrix?

\[ [-2, -1, 0, 1, 2] \quad \bar{x} = 0 \quad \sigma = \frac{\sum (x_i - \bar{x})^2}{n - 1} \]

Covariance matrix

\[
S = \begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{bmatrix}
\]

Collect all projected onto X axis values
How to interpret values in covariance matrix?

\[-2, -1, 0, 1, 2\] \hspace{0.5cm} \bar{x} = 0

\[
\sigma = \frac{\sum (x_i - \bar{x})^2}{n - 1}
\]
How to interpret values in covariance matrix?

\[ \text{Variance} \rightarrow \sigma = \frac{\sum (x_i - \bar{x})^2}{n - 1} \]
How to interpret values in covariance matrix?

**Variance**  \[ \sigma = \frac{\sum (x_i - \bar{x})^2}{n - 1} \]

Collect all projected onto X axis values

number of points

Covariance matrix
How to interpret values in covariance matrix?

**Variance**

$$\sigma = \frac{\sum (x_i - \bar{x})^2}{5 - 1}$$

Collect all projected onto X axis values

Number of points: \([-2, -1, 0, 1, 2]\)
How to interpret values in covariance matrix?

Variance \[ \sigma = \frac{1}{n} \sum (x_i - \bar{x})^2 \]

Collect all projected onto X axis values.

Number of points
How to interpret values in covariance matrix?

Collect all projected onto X axis values

Variance \( \sigma = \frac{\sum (x_i - \bar{x})^2}{n} \)

\( \bar{x} = 0 \)

mean of all points

number of points
How to interpret values in covariance matrix?

Variance \( \sigma = \frac{\sum (x_i - 0)^2}{n} \)

Collect all projected onto X axis values

\( \bar{x} = 0 \)
How to **interpret** values in **covariance** matrix?

**Variance**

\[ \sigma = \frac{\sum (x_i)^2}{n} \]

\[ \bar{x} = 0 \] mean of all points

Collect all projected onto X axis values

number of points

**Covariance** matrix

Collect all projected onto X axis values
How to interpret values in covariance matrix?

Variance $\rightarrow \sigma = \frac{\sum (x_i)^2}{n}$

Collect all projected onto X axis values

\[ \bar{x} = 0 \text{ mean of all points} \]

\[ \text{value of each point} \]

\[ \text{number of points} \]

\[ \text{number of points} \]

\[ \text{number of points} \]
How to interpret values in covariance matrix?

Collect all projected onto X axis values

\[ \bar{x} = 0 \] mean of all points

Variance \( \sigma = \frac{\sum (x_i)^2}{n} \)

number of points

\([-2, -1, 0, 1, 2]\) value of each point

Covariance matrix
How to interpret values in covariance matrix?

Collect all projected onto X axis values

Variance: \( \sigma = \frac{(-2)^2 + (-1)^2 + (0)^2 + (1)^2 + (2)^2}{4} \)

\([-2, -1, 0, 1, 2]\) value of each point

\( \bar{x} = 0 \) mean of all points

Number of points: 5

\( \sum_{i=1}^{n} x_i = -2, -1, 0, 1, 2 \)
How to interpret values in covariance matrix?

Collect all projected onto X axis values

Variance → $\sigma = \frac{10}{4}$
How to interpret values in covariance matrix?

Collect all projected onto X axis values

Variance $\rightarrow \sigma = 2.5$

$[-2, -1, 0, 1, 2]$ value of each point

$\bar{x} = 0$ mean of all points

Covariance matrix

number of points

Collect all projected onto X axis values
How to interpret values in covariance matrix?

Collect all projected onto X axis values

Variance → $\sigma = 2.5$

Variance is an expected value of the squared deviation from the mean

How to interpret values in covariance matrix?

Collect all projected onto X axis values
How to interpret values in covariance matrix?

\[
[-2, -1, 0, 1, 2] \quad \bar{x} = 0 \quad \sigma = 2.5
\]
How to **interpret** values in **covariance** matrix?

\[
[-2, -1, 0, 1, 2] \quad \bar{x} = 0 \quad \sigma = 2.5
\]
How to **interpret** values in **covariance** matrix?

![Covariance matrix and graph]

**Variance** along first axis

\[
S = \begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5 
\end{bmatrix}
\]

**Covariance** matrix
How to **interpret** values in **covariance** matrix?

### Variance along first axis

\[ S = \begin{pmatrix} 2.5 & 1.5 \\ 1.5 & 1.5 \end{pmatrix}\]
How to **interpret** values in **covariance** matrix?

- **Variance** along first axis:
  - $2.5$
  - $1.5$

- **Variance** along second axis:
  - $1.5$

**Covariance** matrix:

\[
\begin{array}{cc}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{array}
\]
How to **interpret** values in **covariance** matrix?

- **Value** of each point
- **Number** of points
- Mean of all points

\[
\sigma = \frac{\sum (y_i - \bar{y})^2}{n - 1}
\]

Variance along **first** axis

Variance along **second** axis

**Covariance** matrix
How to interpret values in covariance matrix?

Value of each point: 
[-2, 0, 0, 1, 1]

Number of points: 5

Mean of all points: \( \bar{y} = 0 \)

Variance along first axis:
\[ \sigma = \frac{\sum (y_i - \bar{y})^2}{n - 1} \]

Variance along second axis:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Covariance matrix
How to **interpret** values in **covariance** matrix?

<table>
<thead>
<tr>
<th>value of each point</th>
<th>number of points</th>
<th>mean of all points</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-2, 0, 0, 1, 1]</td>
<td>5</td>
<td>$\bar{y} = 0$</td>
</tr>
</tbody>
</table>

Variance along first axis

Variance along second axis

Covariance matrix

$$
\begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{bmatrix}
$$
How to interpret values in covariance matrix?

Variance along first axis

\[ S \]

\[
\begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{bmatrix}
\]

Variance along second axis

Covariance matrix
How to interpret values in covariance matrix?
How to **interpret** values in **covariance** matrix?
How to interpret values in covariance matrix?

Covariance indicates how two variables are related. A positive covariance means the variables are positively related, while a negative covariance means the variables are inversely related.
Positive covariance
How to **interpret** values in **covariance** matrix?

![Graph showing data points and covariance matrix](image)
How to interpret values in covariance matrix?

\[ \begin{bmatrix} -2, -1, 0, 1, 2 \end{bmatrix} \begin{bmatrix} -2, 0, 0, 1, 1 \end{bmatrix} \]
\[ \bar{x} = 0 \quad \bar{y} = 0 \]

\[ \text{cov}(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n - 1} \]

\[ S = \begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 1.5 \end{bmatrix} \]

Covariance matrix
How to interpret values in covariance matrix?

\[ \begin{bmatrix} -2, -1, 0, 1, 2 \end{bmatrix} \quad \begin{bmatrix} -2, 0, 0, 1, 1 \end{bmatrix} \]

\[ \bar{x} = 0 \quad \bar{y} = 0 \]

\[ \text{cov}(x, y) = \frac{6}{4} \]

\[
\begin{array}{c|c}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{array}
\]

Covariance matrix
How to interpret values in covariance matrix?

\[ \begin{bmatrix} -2, -1, 0, 1, 2 \end{bmatrix}, \begin{bmatrix} -2, 0, 0, 1, 1 \end{bmatrix} \]

\[ \bar{x} = 0 \quad \bar{y} = 0 \quad \text{cov}(x, y) = 1.5 \]
How to interpret values in covariance matrix?
How to interpret values in covariance matrix?

\[ \text{cov}(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n - 1} \]

\[ \begin{bmatrix} 2.5 & ? \\ ? & 1.5 \end{bmatrix} \]

Covariance matrix
How to **interpret** values in **covariance** matrix?

\[
\text{cov}(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n - 1}
\]

\[
\bar{x} = 0 \quad \bar{y} = 0
\]

\[
\begin{bmatrix}
-2, & -1, & 0, & 1, & 2 \\
0, & 0, & 1, & -2, & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
2.5 & ? \\
? & 1.5 \\
\end{bmatrix}
\]

Covariance matrix
How to **interpret** values in **covariance** matrix?

\[
\begin{bmatrix}
-2, -1, 0, 1, 2 \\
0, 0, 1,-2, 1
\end{bmatrix}
\]

\[
\bar{x} = 0 \quad \bar{y} = 0
\]

\[
\text{cov}(x, y) = \frac{\sum (x_i)(y_i)}{4}
\]

Covariance matrix

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>1.5</td>
</tr>
</tbody>
</table>
How to interpret values in covariance matrix?

\[ [\bar{x}, \bar{y}] = \left[ \begin{array}{c} \bar{x} = 0 \\ \bar{y} = 0 \end{array} \right] \]

\[ \text{cov}(x, y) = \frac{(-2)(0) + (-1)(0) + (0)(1) + (1)(-2) + (2)(1)}{4} \]

\[ \begin{array}{c|c|c} & \text{2.5} & \text{?} \\
\hline \text{?} & 1.5 & \text{?} \\
\end{array} \]

\textbf{Covariance matrix}
How to **interpret** values in **covariance** matrix?

\[
\begin{bmatrix}
-2, -1, 0, 1, 2
\end{bmatrix}
\begin{bmatrix}
0, 0, 1, -2, 1
\end{bmatrix}
\]

\[
\bar{x} = 0 \quad \bar{y} = 0
\]

\[
cov(x, y) = \frac{0}{4} = 0
\]

\[
S
\]

\[
\begin{array}{cc}
2.5 & ? \\
? & 1.5
\end{array}
\]

**Covariance** matrix
How to **interpret** values in **covariance** matrix?

\[ [\begin{bmatrix} -2, -1, 0, 1, 2 \end{bmatrix}, \begin{bmatrix} 0, 0, 1, -2, 1 \end{bmatrix}] \]

\[ \bar{x} = 0 \quad \bar{y} = 0 \]

\[ \text{cov}(x, y) = \frac{0}{4} = 0 \]

Covariance matrix:

\[ S = \begin{bmatrix} 2.5 & 0 \\ 0 & 1.5 \end{bmatrix} \]
How to **interpret** values in **covariance** matrix?

\[
\begin{bmatrix}
-2, -1, 0, 1, 2 \\
0, 0, 1, -2, 1
\end{bmatrix}
\]

\[\bar{x} = 0 \quad \bar{y} = 0\]

\[
cov(x, y) = \frac{0}{4} = 0
\]

Covariance 0 means that there is **no relationship** between two variables. Knowing something about the value of one does not say anything about the value of the other.
How to interpret values in covariance matrix?
How to **interpret** values in **covariance** matrix?

![Graph and covariance matrix](image)
How to interpret values in covariance matrix?

Why these values are > 1? Aren’t they supposed to be between [-1, 1]?
How to interpret values in covariance matrix?

Why these values are > 1? Aren’t they supposed to be between [-1,1]?

Covariance vs Correlation: https://en.wikipedia.org/wiki/Covariance_and_correlation
\[ Z^T \times Z \times 4 \]
\[
\begin{array}{cc}
S & = \begin{pmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{pmatrix}
\end{array}
\]
\[
S = PDP^T
\]

\[
\begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{bmatrix}
= 
\begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{bmatrix}
\]
\[ S = \begin{bmatrix} 2.5 & 1.5 \\ 1.5 & 1.5 \end{bmatrix} \quad PD = \begin{bmatrix} -2.9 & 0.24 \\ -2.1 & -0.33 \end{bmatrix} \quad \begin{bmatrix} 0.58 \\ -0.81 \end{bmatrix} \]
For an example: https://www.scss.tcd.ie/Rozenn.Dahyot/CS1BA1/SolutionEigen.pdf
Eigen decomposition

\[
S = P \times D \times P^T
\]

For an example: https://www.scss.tcd.ie/Rozenn.Dahyot/CS1BA1/SolutionEigen.pdf
Eigen decomposition

\[
S = P D P^T
\]

Eigen vectors

\[
\begin{pmatrix}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{pmatrix}
= \begin{pmatrix}
-0.81 & 0.58 \\
-0.58 & -0.81 \\
\end{pmatrix}
\times
\begin{pmatrix}
3.58 & 0 \\
0 & 0.41 \\
\end{pmatrix}
\times
\begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81 \\
\end{pmatrix}
\]

Eigen values

For an example: https://www.scss.tcd.ie/Rozenn.Dahyot/CS1BA1/SolutionEigen.pdf
Eigen\textbf{vectors}

\begin{tabular}{cc}
-0.81 & 0.58 \\
-0.58 & -0.81 \\
\end{tabular}
Eigenvalues

-0.81  0.58  
-0.58 -0.81

(0,0)
Eigen\textbf{vectors}

\begin{array}{cc}
-0.81 & 0.58 \\
-0.58 & -0.81 \\
\end{array}

\begin{tikzpicture}
\begin{axis}[
    axis lines=middle,
    x label style={at={(axis description cs:1,0)},anchor=north},
    y label style={at={(axis description cs:0,1)},anchor=south},
    xlabel={X},
    ylabel={Y},
    grid=both,
    grid style={line width=.2pt, draw=gray!30},
    major grid style={line width=.4pt,draw=gray!50},
]
\addplot[blue,mark=*] coordinates {(0,0)} node[above right,black] {(0,0)};
\end{axis}
\end{tikzpicture}
Eigenvectors

<table>
<thead>
<tr>
<th>-0.81</th>
<th>0.58</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.58</td>
<td>-0.81</td>
</tr>
</tbody>
</table>

(0,0)
Eigenvectors

\[
\begin{array}{cc}
-0.81 & 0.58 \\
-0.58 & -0.81 \\
\end{array}
\]
Eigen\textbf{vectors}

\begin{array}{|cc|}
\hline
-0.81 & 0.58 \\
\hline
-0.58 & -0.81 \\
\hline
\end{array}
Eigen\textbf{vectors}

\begin{center}
\begin{tabular}{|c|c|}
\hline
-0.81 & 0.58 \\
\hline
-0.58 & -0.81 \\
\hline
\end{tabular}
\end{center}

\textbf{Old} coordinate system

![Graph showing two eigenvectors. The eigenvector #1 is a blue line and the eigenvector #2 is a red line.
Eigen\textbf{vectors}

\begin{tabular}{|c|c|}
\hline
-0.81 & 0.58 \\
\hline
-0.58 & -0.81 \\
\hline
\end{tabular}

Old coordinate system

eigenvector \#1

eigenvector \#2

New coordinate system

eigenvector \#1

eigenvector \#2

old coordinate system

new coordinate system
Eigenvalues

\[
\begin{pmatrix}
-0.81 & 0.58 \\
-0.58 & -0.81
\end{pmatrix}
= \begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{pmatrix}
\]

Old coordinate system

New coordinate system
Eigen\textit{vectors}^T

\begin{table}
\begin{tabular}{cc}
-0.81 & -0.58 \\
0.58 & -0.81 \\
\end{tabular}
\end{table}

Old coordinate system

New coordinate system
Eigenvectors\textsuperscript{T}

\[
\begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{pmatrix}
\]

\[Z^T\]

\[
\begin{pmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{pmatrix}
\]
Eigen\textbf{vectors}^{T} \\
\begin{array}{cc}
-0.81 & -0.58 \\
0.58 & -0.81
\end{array}

\quad \times \quad 

\begin{array}{ccccc}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{array}

\textbf{Old} coordinate system

\textbf{New} coordinate system
Let $\mathbf{Z}_T$ be a matrix containing the eigenvectors of $\mathbf{E}_T$, i.e.,

$$
\mathbf{Z}_T = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & 0 & 1 & 0 & 1 \end{bmatrix}
$$

The eigenvectors are given by $\mathbf{e}_1 = [-0.81, -0.58]^T$ and $\mathbf{e}_2 = [0.58, -0.81]^T$. The transformation from the old coordinate system to the new coordinate system can be represented by the matrix multiplication

$$
\mathbf{Z}_T \times \begin{bmatrix} -0.81 & -0.58 \\ 0.58 & -0.81 \end{bmatrix} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & 0 & 1 & 0 & 1 \end{bmatrix}
$$

The old coordinate system is $x$-axis and $y$-axis, and the new coordinate system is defined by the eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. The points in the old coordinate system are transformed to the new coordinate system.
The eigenvectors are given by:

\[
\begin{bmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{bmatrix}^T
\]

And the transformation matrix is:

\[
\begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{bmatrix}
\]

The product of the two matrices is calculated as:

\[
\begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{bmatrix}^T
\]

\[
\begin{align*}
-2 \times (-0.81) + (-2) \times (-0.58) &= 2.78 \\
-2 \times (0.58) + (-2) \times (-0.81) &= 0.46
\end{align*}
\]

The old coordinate system is transformed to the new coordinate system.
Eigenvectors^T

\[
\begin{bmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{bmatrix}
\]

\[
\begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{bmatrix}
\]

\[
\begin{align*}
-2*(-0.81) + (-2)*(-0.58) &= 2.78 \\
-2*(0.58) + (-2)*(-0.81) &= 0.46
\end{align*}
\]

Old coordinate system

New coordinate system
The eigenvectors are:

\[
\begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{pmatrix}
\]

The matrix \( Z^T \) is:

\[
\begin{pmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{pmatrix}
\]

The multiplication results in:

\[
\begin{align*}
-2 \times (-0.81) + (-2) \times (-0.58) &= 2.78 \\
-2 \times (0.58) + (-2) \times (-0.81) &= 0.46
\end{align*}
\]
$\text{Eigenvectors}^T$

$$\begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{pmatrix} \times 
\begin{pmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{pmatrix} = 
\begin{pmatrix}
2.78 & 0.46
\end{pmatrix}$$

**Old coordinate system**

**New coordinate system**
\[ \begin{pmatrix} -0.81 & -0.58 \\ 0.58 & -0.81 \end{pmatrix} \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2.78 & 0.46 \end{pmatrix} \]

-1\((-0.81) + 0\)(-0.58) = 0.81
-1\((0.58) + 0\)(-0.81) = -0.58
Eigen\textbf{vectors}^T

\[
\begin{pmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{pmatrix}
\times
\begin{pmatrix}
\begin{array}{ccccc}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{array}
\end{pmatrix}
= \begin{pmatrix}
2.78 & 0.46
\end{pmatrix}
\]

\[-1 \times (-0.81) + 0 \times (-0.58) = 0.81\]
\[-1 \times (0.58) + 0 \times (-0.81) = -0.58\]

\text{Old coordinate system}

\text{New coordinate system}
$$\text{Eigenvectors}^T = \begin{bmatrix} -0.81 & -0.58 \\ 0.58 & -0.81 \end{bmatrix}$$

$$Z^T = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{align*}
-1 \times (-0.81) + (0) \times (-0.58) &= 0.81 \\
-1 \times (0.58) + (0) \times (-0.81) &= -0.58
\end{align*}$$

### Old coordinate system

### New coordinate system
The diagram illustrates the transformation from the old coordinate system to the new coordinate system using eigenvectors.

The eigenvectors are given as:

\[ \text{Eigen} \text{vectors}^T = \begin{bmatrix} -0.81 & -0.58 \\ 0.58 & -0.81 \end{bmatrix} \]

And the transformation matrix \( Z^T \) is:

\[ Z^T = \begin{bmatrix} 2.78 & 0.46 \\ 0.81 & -0.58 \end{bmatrix} \]

The graphs show the old and new coordinate systems, with eigenvectors indicated.

- **Old coordinate system**:
  - Eigenvector #1 is represented by the blue line.
  - Eigenvector #2 is represented by the red line.

- **New coordinate system**:
  - Eigenvector #1 is still represented by the blue line, but it points in a different direction.
  - Eigenvector #2 is also represented differently, now pointing in a new direction.

The transformation shows how the points in the old system are aligned along the eigenvectors and then scaled according to the transformation matrix.
Eigen\textbf{vectors}^T

\begin{array}{cc}
-0.81 & -0.58 \\
0.58 & -0.81 \\
\end{array}

\times

\begin{array}{ccccc}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1 \\
\end{array}

= \begin{array}{cc}
2.78 & 0.46 \\
0.81 & -0.58 \\
\end{array}

\begin{align*}
0^*(-0.81) + (1)^*(-0.58) &= -0.58 \\
0^*(0.58) + (1)^*(-0.81) &= -0.81
\end{align*}

\begin{array}{cc}
\text{Old coordinate system} & \text{New coordinate system} \\
\end{array}

\text{Old coordinate system}

\text{New coordinate system}
Eigen\textit{vectors}^T

\begin{align*}
\begin{bmatrix}
-0.81 & -0.58 \\
0.58 & -0.81 \\
\end{bmatrix} \times \begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1 \\
\end{bmatrix} &= \begin{bmatrix}
2.78 & 0.46 \\
0.81 & -0.58 \\
-0.58 & -0.81 \\
\end{bmatrix}
\end{align*}

0^*(-0.81) + 1^*(-0.58) = -0.58
0^*(0.58) + 1^*(-0.81) = -0.81
Eigen\textbf{vectors}^T 

\begin{align*}
\begin{array}{cc}
-0.81 & -0.58 \\
0.58 & -0.81 \\
\end{array}
\end{align*}

\begin{align*}
\begin{array}{cccc}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1 \\
\end{array}
\end{align*}

\begin{array}{cc}
2.78 & 0.46 \\
0.81 & -0.58 \\
-0.58 & -0.81 \\
\end{array}

\begin{align*}
= 
\end{align*}

\textbf{Old coordinate system} \\
\textbf{New coordinate system}
**Eigen**\textit{vectors}^T

\begin{bmatrix}
-0.81 & -0.58 \\
0.58 & -0.81
\end{bmatrix}

\times

\begin{bmatrix}
-2 & -1 & 0 & 1 & 2 \\
-2 & 0 & 1 & 0 & 1
\end{bmatrix}

= \begin{bmatrix}
2.78 & 0.46 \\
0.81 & -0.58 \\
-0.58 & -0.81 \\
-0.81 & 0.58 \\
-2.2 & 0.35
\end{bmatrix}

**Old coordinate system**

**New coordinate system**
These eigenvectors are called **Principle Components**.

Eigenvector #1 is called **PC1**.
Eigenvector #2 is called **PC2**.

\[
\begin{pmatrix}
0.58 & -0.81 \\
-0.81 & -0.58 \\
\end{pmatrix}
\]
These eigenvectors are called **Principle Components**

- eigenvector #1 is called **PC1**
- eigenvector #2 is called **PC2**
Compute covariance matrix

Perform eigen decomposition

Compute new coordinates
Do you still remember what was it all about?
Do you still remember what was it all about?
We want to **reduce the dimensionality**!
Do you still remember what was it all about?
We want to **reduce the dimensionality**!

**Old** coordinate system

**New** coordinate system
We can **ignore** the **second eigenvector** because it does not contain much information.

**Old** coordinate system

**New** coordinate system
We can **ignore** the **second eigenvector** because it does not contain much information. How much information the second eigenvector contains?
Old coordinate system

New coordinate system
**Old coordinate system**

**New coordinate system**

**Eigenvalues**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.58</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.42</td>
</tr>
</tbody>
</table>
Covariance matrix

\[
S = \begin{bmatrix}
2.5 & 1.5 \\
1.5 & 1.5 
\end{bmatrix}
\]

Eigenvalues

\[
D = \begin{bmatrix}
3.58 & 0 \\
0 & 0.42 
\end{bmatrix}
\]

Old coordinate system

New coordinate system
Old coordinate system

Covariance matrix

Variance along X axis

Variance along Y axis

Eigenvalues

D

3.58  0

0   0.42

S

2.5  1.5

1.5  1.5

New coordinate system

Variance along X axis

Variance along Y axis

PC1

PC2
Old coordinate system

Covariance matrix:

<table>
<thead>
<tr>
<th></th>
<th>2.5</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

Variance along X axis: 3.58
Variance along Y axis: 0.42

Eigenvalues:

<table>
<thead>
<tr>
<th></th>
<th>3.58</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.42</td>
<td></td>
</tr>
</tbody>
</table>

New coordinate system

Variance along PC1: 2.5
Variance along PC2: 1.5

Old coordinate system

Variance along PC1: 2.5
Variance along PC2: 1.5

New coordinate system

Variance along PC1: 2.5
Variance along PC2: 1.5

PC1

PC2
Covariance matrix

\[
S = \begin{pmatrix}
2.5 & 1.5 \\
1.5 & 1.5 
\end{pmatrix}
\]

Variance along X axis
Variance along Y axis

Variance along PC1
D
\[
\begin{pmatrix}
3.58 & 0 \\
0 & 0.42 
\end{pmatrix}
\]

Variance along PC2

Old coordinate system

New coordinate system
**Old coordinate system**

**Covariance matrix**

\[
S = \begin{pmatrix}
2.5 & 1.5 \\
1.5 & 1.5 \\
\end{pmatrix}
\]

**Eigenvalues**

\[
D = \begin{pmatrix}
3.58 & 0 \\
0 & 0.42 \\
\end{pmatrix}
\]

**New coordinate system**

**PC1**

**PC2**
Old coordinate system

Covariance matrix


definition of values

New covariance matrix

New coordinate system

PC1

PC2
Both **old** and **new** axes explain 4 units of variance

\[
\begin{pmatrix}
2.5 & 1.5 \\
1.5 & 1.5
\end{pmatrix}
\]

Covariance matrix

\[
\begin{pmatrix}
3.58 & 0 \\
0 & 0.42
\end{pmatrix}
\]

New covariance matrix

\[2.5 + 1.5 = 4\]

\[3.58 + 0.42 = 4\]
Both **old** and **new** axes explain 4 units of variance

\[ 2.5 + 1.5 = 4 \]

Out of these 4, **X** axis explains:

\[ \frac{2.5}{4} = 62.5\% \]

And **Y** axis explains:

\[ \frac{1.5}{4} = 37.5\% \]
Both **old** and **new** axes explain **4 units of variance**

\[ 2.5 + 1.5 = 4 \]

Out of these 4, **X** axis explains:
\[ \frac{2.5}{4} = 62.5\% \]
And **Y** axis explains:
\[ \frac{1.5}{4} = 37.5\% \]

\[ 3.58 + 0.42 = 4 \]

Out of these 4, **PC1** axis explains:
\[ \frac{3.58}{4} = 89.5\% \]
And **PC2** axis explains:
\[ \frac{0.42}{4} = 10.5\% \]
We can **ignore** the **second PC** because it does not contain much information.

**How much information the second PC contains?**

**Old** coordinate system

**New** coordinate system
We can **ignore** the **second PC** because it explains only **10.5%** of variation.
How many PCs will be formed in 200D space?
How many PCs will be formed in 200D space?

No exceptions, 200 PCs
How many PCs will be formed in 200D space?

No exceptions, 200 PCs
How many PCs should we keep?
**Variance explained** is a good criteria for choosing the total number of PCs to keep.

How many PCs will be formed in 200D space?

How many PCs should we keep?
Variance explained is a good criteria for choosing the total number of PCs to keep.

You should keep as many PCs as it takes to explain 90% of total variance.

How many PCs should we keep?
Variance explained is a good criteria for choosing the total number of PCs to keep

You should keep as many PCs as it takes to explain 90% of total variance

How many PCs should we keep?
Principle Component Analysis (PCA)

Can be used as part of supervised learning pipeline
Supervised Learning pipeline

1. Acquire Data
2. Preprocessing
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set

Safe place
Profit
Supervised Learning pipeline

1. 200D raw data
2. Preprocessing
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set

Safe place
Profit
Supervised Learning pipeline

1. 200D raw data
2. Normalisation (subtract mean)
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set

Safe place

Profit
Supervised Learning pipeline

1. 200D raw data
2. Normalisation (subtract mean)
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set

Safe place

Profit
Supervised Learning pipeline

1. 200D raw data
2. Normalisation (subtract mean)
3. Train/test split
4. Find the best model using CV
5. Evaluate final model on the test set
6. Safe place

Profit
Test
Supervised Learning pipeline

1. 200D raw data

2. Normalisation (subtract mean)

3. PCA

4. Train/test split

5. Evaluate final model on the test set

Find the best model using CV

Safe place

Profit
Supervised Learning pipeline

1. 200D raw data

2. Normalisation (subtract mean)

3. PCA
   - 200 PCs
   - Keep few PCs (90% variance)

4. Train/test split
   - Safe place

5. Evaluate final model on the test set

Find the best model using CV

Profit
Supervised Learning pipeline

1. 200D raw data
2. Normalisation (subtract mean)
3. Train/test split
4. PCA
   - Keep few PCs (90% variance)
5. Find the best model using CV

Evaluate final model on the test set
Safe place
Supervised Learning pipeline

1. 200D raw data
2. Normalisation (subtract mean)
3. PCA
   - 200 PCs
   - Keep few PCs (90% variance)
4. Train/test split
5. Find the best model using CV
6. Evaluate final model on the test set
   - Profit
Supervised Learning pipeline

Here, **PCA** is performed only on **training data**

1. **200D raw data**
2. Normalisation (subtract mean)
3. **PCA** 200 PCs
4. Keep few PCs (90% variance)
5. Find the best model using CV
6. Evaluate final model on the test set

Test

Safe place

Profit

Train/test split
Supervised Learning pipeline

Here, **PCA** is performed only on **training data**

- **1** 200D raw data
- **2** Normalisation (subtract mean)
- **3** Train/test split
- **4** Keep few PCs (90% variance)
- **5** Find the best model using CV
- **6** Evaluate final model on the test set

**When we use the test set, it is transformed using the same eigenvectors as the training data.**

**Safe place**
PCA has an "undo" button

You can recover the original features back!
PCA has an "undo" button

Conventional PCA

\[ \text{eigenvalues} \times Z_{\text{original}}^T = Z_{\text{transformed}} \]

Reversed PCA

\[ \text{eigenvalues}^T \times Z_{\text{transformed}} = Z_{\text{original}} \]

You can recover the original features back!

Principle components are linear combinations of original features.
Principle components are linear combinations of original features.

So if you predict anything based on PCs, the meaning of original features is not preserved after the transformation.
Principle components are linear combinations of original features. Which means PCs can capture only linear relationships between original features.
t-Distributed Stochastic Neighbor Embedding (t-SNE)
&
Uniform Manifold Approximation and Projection (UMAP)
**t-SNE** iteratively tries to make distances in low-dimensional space to be similar to distances in high-dimensional space.

A bit more about t-SNE: https://distill.pub/2016/misread-tsne/
**t-SNE** iteratively tries to make distances in low-dimensional space to be similar to distances in high-dimensional space.

**UMAP** ultimately tries to achieve similar things, using slightly different mechanisms.
t-SNE iteratively tries to make distances in low-dimensional space to be similar to distances in high-dimensional space.

Both t-SNE and UMAP cannot “undo” transformations.

UMAP ultimately tries to achieve similar things, using slightly different mechanisms.
t-SNE iteratively tries to make distances in low-dimensional space to be similar to distances in high-dimensional space.

Both t-SNE and UMAP cannot “undo” transformations.

Both t-SNE and UMAP are slower than PCA.

UMAP ultimately tries to achieve similar things, using slightly different mechanisms.
Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.

UMAP explained: https://pair-code.github.io/understanding-umap/
Figure 6: A comparison between UMAP and t-SNE projections of a 3D woolly mammoth skeleton (50,000 points) into 2 dimensions, with various settings for parameters. Notice how much more global structure is preserved with UMAP, particularly with larger values of n_neighbors.

UMAP explained and compared to t-SNE: https://pair-code.github.io/understanding-umap/
References

• Machine Learning by Andrew Ng (https://www.coursera.org/learn/machine-learning)

• Curse of dimensionality: http://cleverowl.uk/2016/02/06/curse-of-dimensionality-explained/

• Visualisation of PCA: http://setosa.io/ev/principal-component-analysis/


• Stanford CS class: Convolutional Neural Networks for Visual Recognition by Andrej Karpathy (http://cs231n.github.io/)

• Data Mining Course by Jaak Vilo at University of Tartu (https://courses.cs.ut.ee/MTAT.03.183/2017_spring/uploads/Main/DM_05_Clustering.pdf)
Happy end!
That's all Folks!