MATHEMATICS FOR MACHINE LEARNING

Nagoya University, Fall 2023

Lecture 13

PCA II & Autoencoder

This week Tutorial: Thursday 25th Jan. 6th period

This is also the deadline of Homework 4 and the semester project

https://www.henrikbachmann.com/mml2023.html

Semester project

Objective: Choose a project topic related to Nagoya or Japan more broadly that can be addressed using machine learning algorithms. Your task is to develop a machine learning model to solve a specific problem or provide insights into an aspect of life, business, environment, culture, etc., in Nagoya/Japan.

Group Size: 1-3 members **A variation of Homework 2,3 or 4 is also ok!**

Code: Preferably a Google Colab notebook. Exceptions are possible; please provide full documentation for any different technology or package used. If you plan not to submit a Google Colab, please contact us in advance.

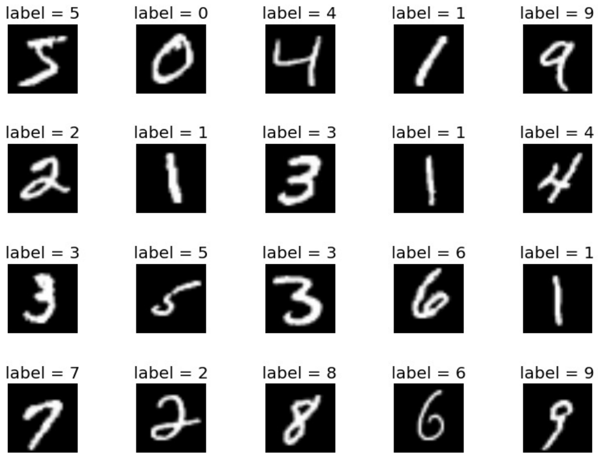
Documentation: 5-10 slides as if you were going to present the project.
Your slides **could** cover for example:

- Problem Statement
- Data Collection
- Data Exploration and Visualization
- Model Building and Evaluation
- Conclusion

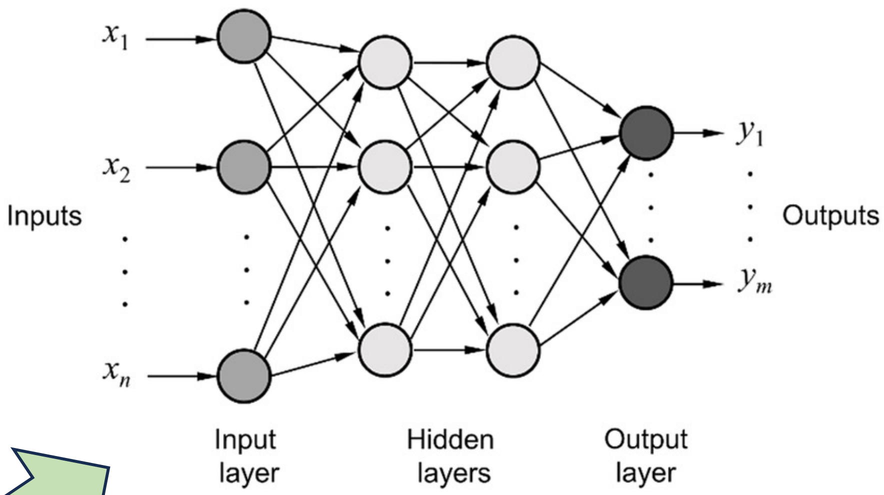
Unsupervised learning: Dimensionality reduction

Example: Digit recognition (MNIST Dataset)

Pictures of size 28 x 28 = 784 pixels



Neural network with input layer size X

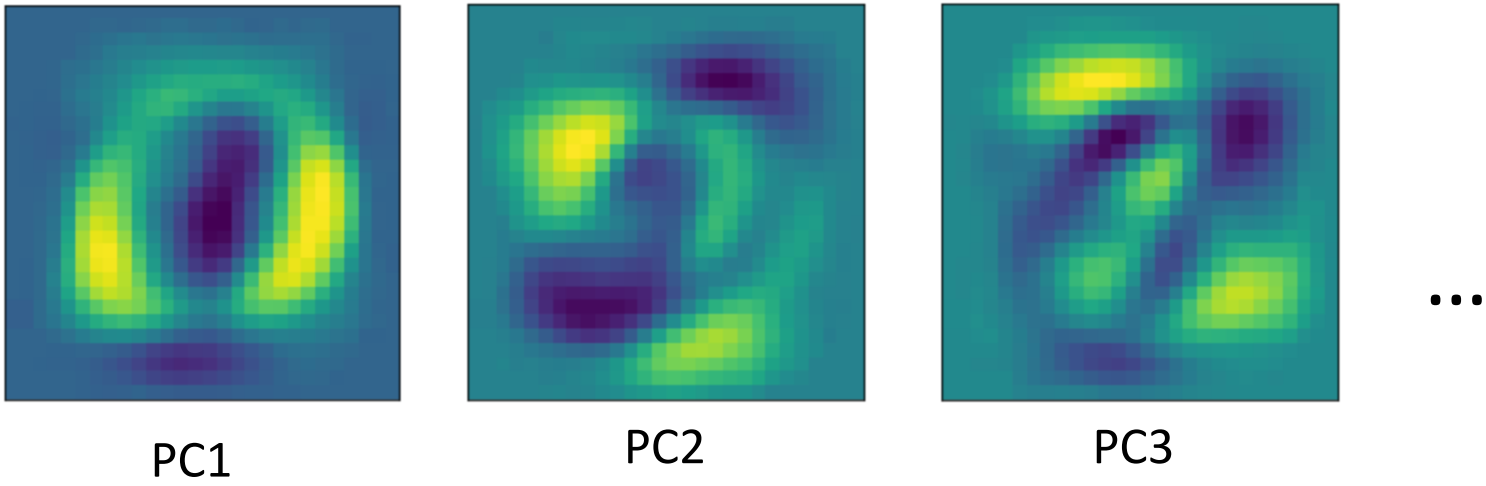


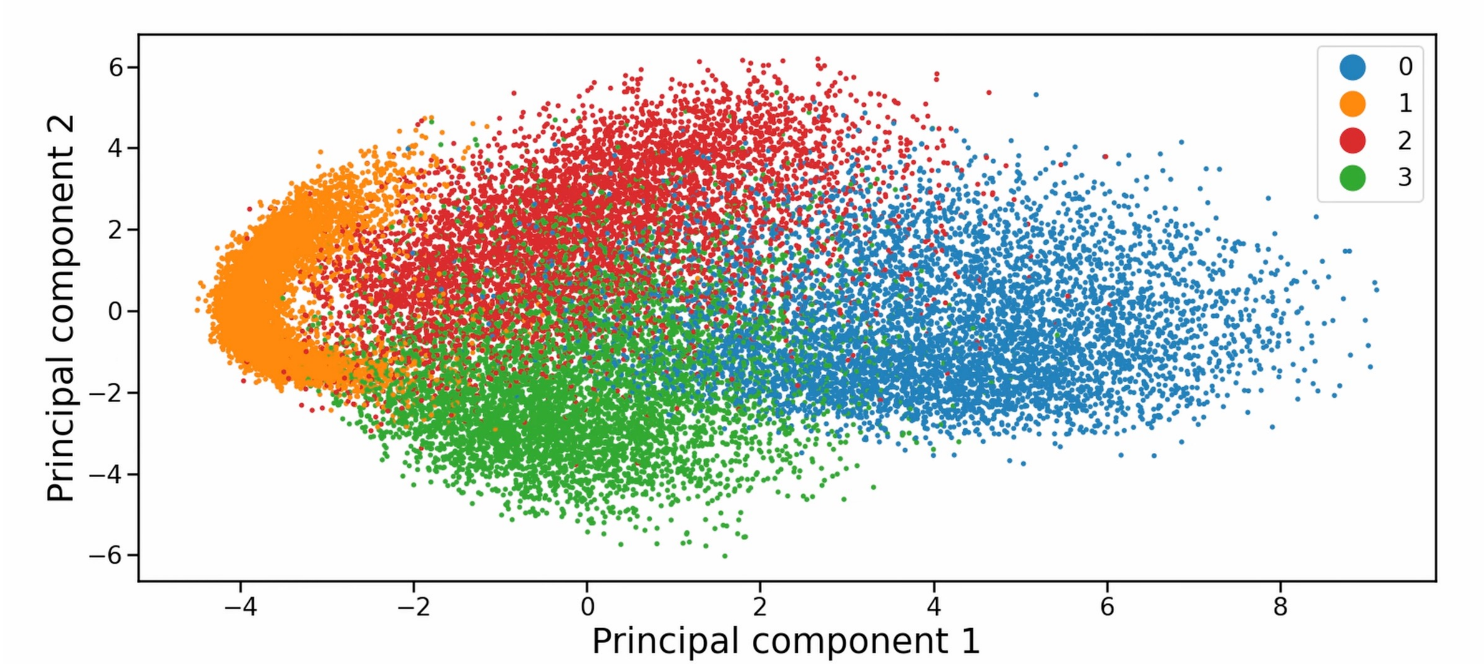
784 Datapoints ordered by "importance".
Choose the first X of them.

Better approach: Find a new representation of the picture into <u>principal components.</u>

Unsupervised learning: Dimensionality reduction

The first three principal components (PC) of the MNIST dataset

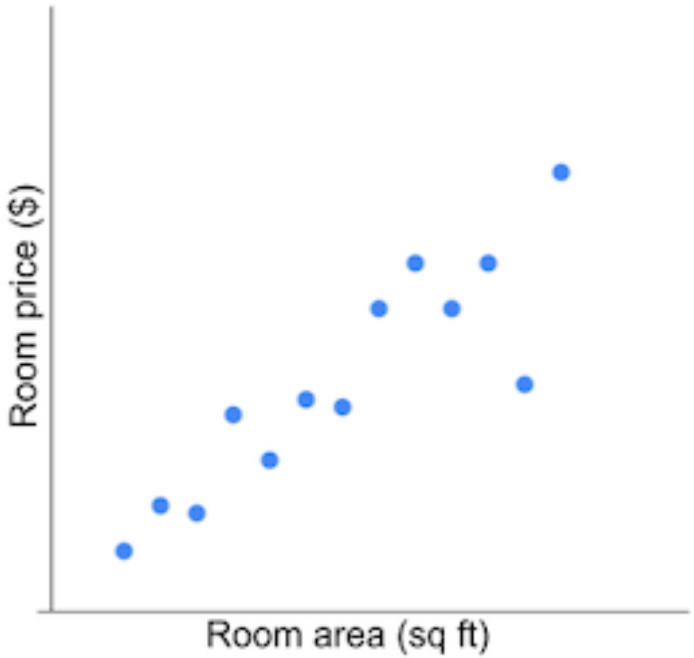




Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a method that reduces the dimensionality of data by transforming it into principal components, each representing unique variance, while retaining the most significant information.

“Finds a better coordinate system for given data, such that the axes are ordered by importance”



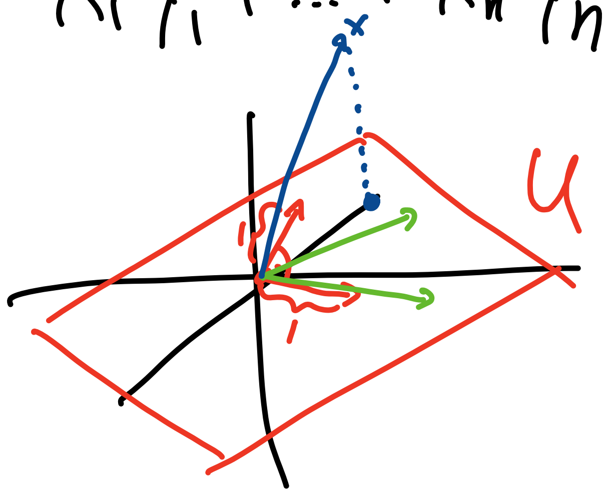
Recall: Orthonormal bases

Vectors $u_1, \ldots, u_l \in \mathbb{R}^n$ *are called* **orthonormal** *if for* $1 \le i, j \le l$,

$$\|x\| = \sqrt{x \cdot x} \qquad u_i \bullet u_j = \begin{cases} 1 & , \text{if } i = j \\ 0 & , \text{if } i \neq j \end{cases}.$$

A basis $B = (b_1, \ldots, b_m)$ of a subspace U is called an **orthonormal basis (ONB)** of U if $b_1, \ldots, b_m$ are orthonormal.

$$x^T y = \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}}_{= x} \bullet \underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}}_{= y} = x_1 y_1 + \ldots + x_n y_n$$



Recall: Orthonormal bases

Vectors $u_1, \ldots, u_l \in \mathbb{R}^n$ *are called* **orthonormal** *if for* $1 \leq i, j \leq l$,

$$u_i \bullet u_j = \begin{cases} 1 & , \textit{if } i = j \\ 0 & , \textit{if } i \neq j \end{cases} .$$

A basis $B = (b_1, \ldots, b_m)$ of a subspace U is called an **orthonormal basis (ONB)** of U if $b_1, \ldots, b_m$ are orthonormal.

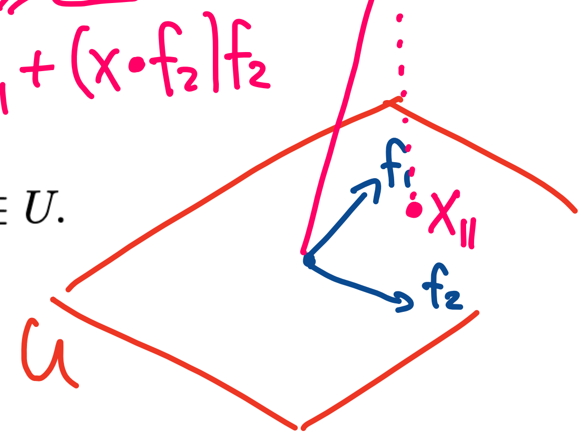
Lemma 2.11. *Let* $U \subset \mathbb{R}^n$ *be a subspace with ONB* $(f_1, \ldots, f_r)$*. Then any* $x \in \mathbb{R}^n$ *can be written as*

$$x = x_\parallel + x_\perp ,$$

where $x_\perp = x - x_\parallel \in U^\perp$ *and*

$$x_\parallel = \sum_{i=1}^{r} (x \bullet f_i) f_i \in U.$$

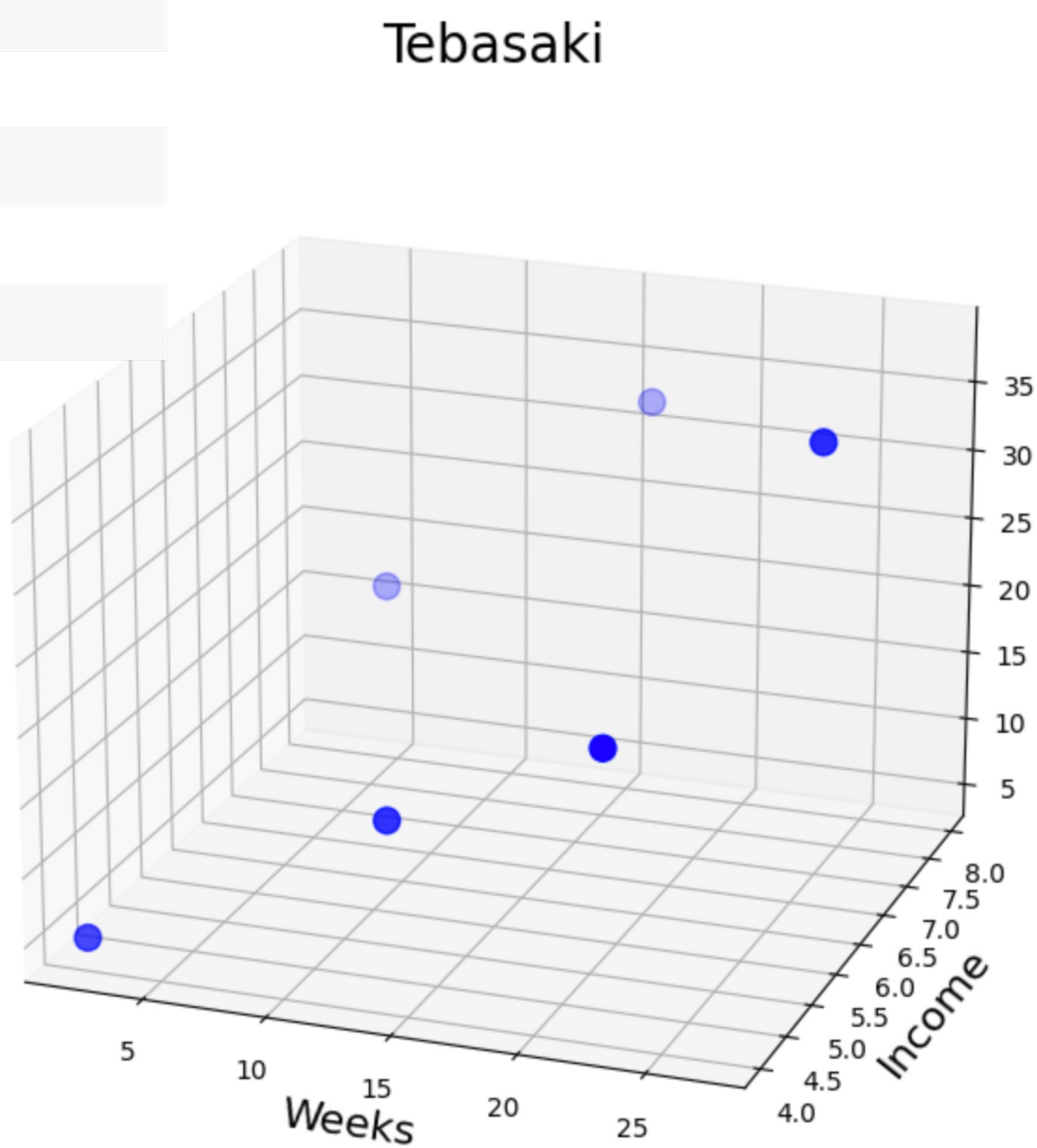
$$X_\parallel = (x \bullet f_1) f_1 + (x \bullet f_2) f_2 = \underbrace{x^T f_1}\, f_1 + x^T f_2 f_2$$



The $x_\parallel$ *is the orthogonal projection of* x *onto* U.

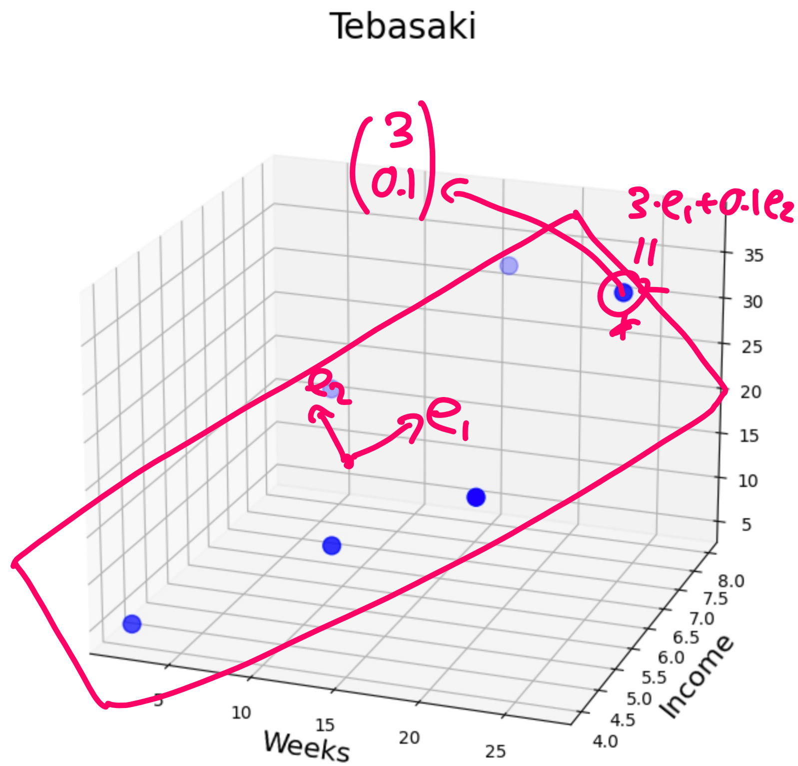
PCA – Tebasaki Example

Weeks living in Nagoya	Monthly income (万円)	Tebasaki eaten
2	4	5
7	7	20
13	4.5	14
16	8	32
22	4.3	22
27	6	38



PCA – Tebasaki Example

Weeks living in Nagoya	Monthly income (万円)	Tebasaki eaten
2	4	5
7	7	20
13	4.5	14
16	8	32
22	4.3	22
27	6	38
W	I	T



Goal:

- Represent these datapoints by 2 (PC1,PC2) values instead of 3 (W,I,T)
- Make it possible to get back (a good approximation of) (W,I,T) from (PC1,PC2)

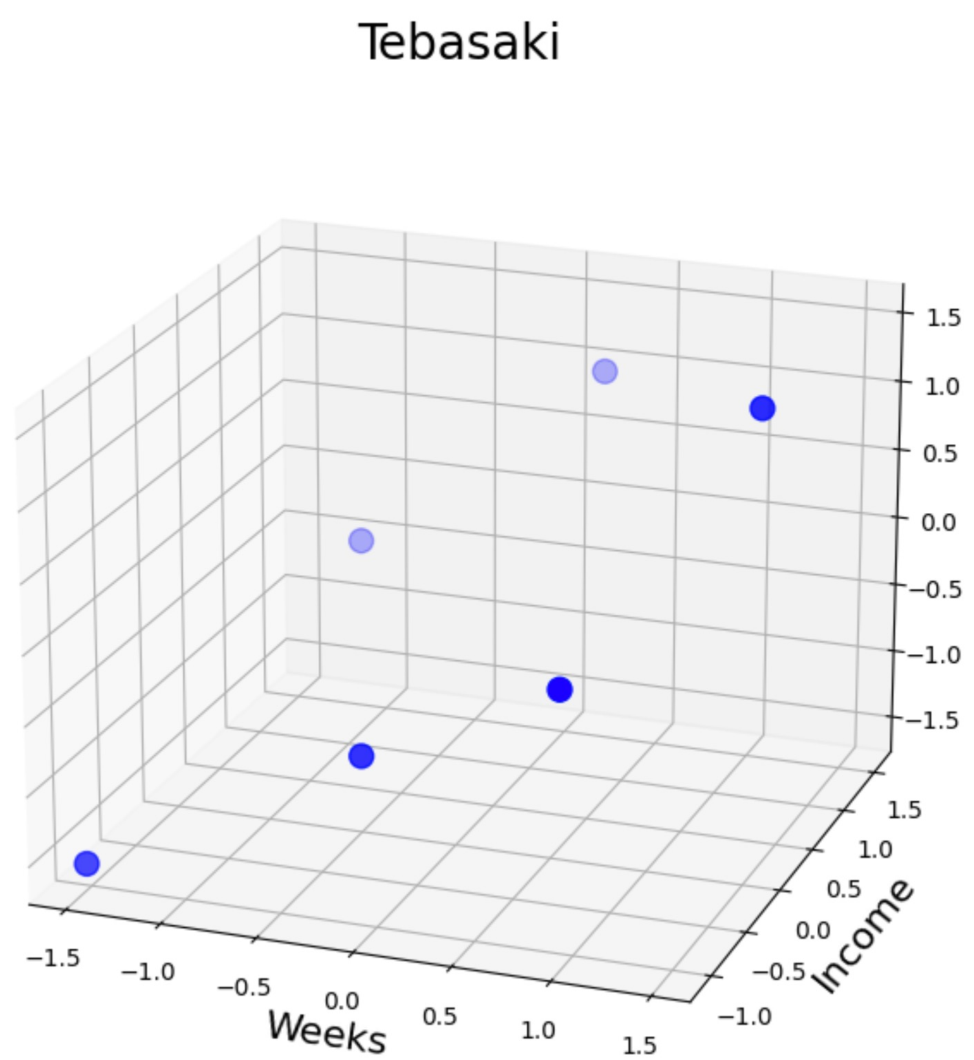
PCA/Tebasaki Example: Step 1 - Normalize

We first normalize the data and give them a mean 0 and standard deviation of 1

$$\begin{pmatrix} 2 & 4 & 5 \\ 7 & 7 & 20 \\ 13 & 4.5 & 14 \\ 16 & 8 & 32 \\ 22 & 4.3 & 22 \\ 27 & 6 & 38 \end{pmatrix}$$

Step 1
mean = 0
std = 1

```
[[-1.47742027 -1.09567331 -1.54437139]
 [-0.88645216  0.91678787 -0.16819886]
 [-0.17729043 -0.76026311 -0.71866788]
 [ 0.17729043  1.58760826  0.93273916]
 [ 0.88645216 -0.89442719  0.01529081]
 [ 1.47742027  0.24596748  1.48320817]]
```



PCA/Tebasaki Example: Step 2 – Covariance matrix

Last lecture: The directions of the biggest variations are given (ordered by the eigenvalue) by the eigenvectors of the **covariance matrix**.

$$\widetilde{X} = \begin{pmatrix} 2 & 4 & 5 \\ 7 & 7 & 20 \\ 13 & 4.5 & 14 \\ 16 & 8 & 32 \\ 22 & 4.3 & 22 \\ 27 & 6 & 38 \end{pmatrix}$$

Step 1
mean = 0
std = 1

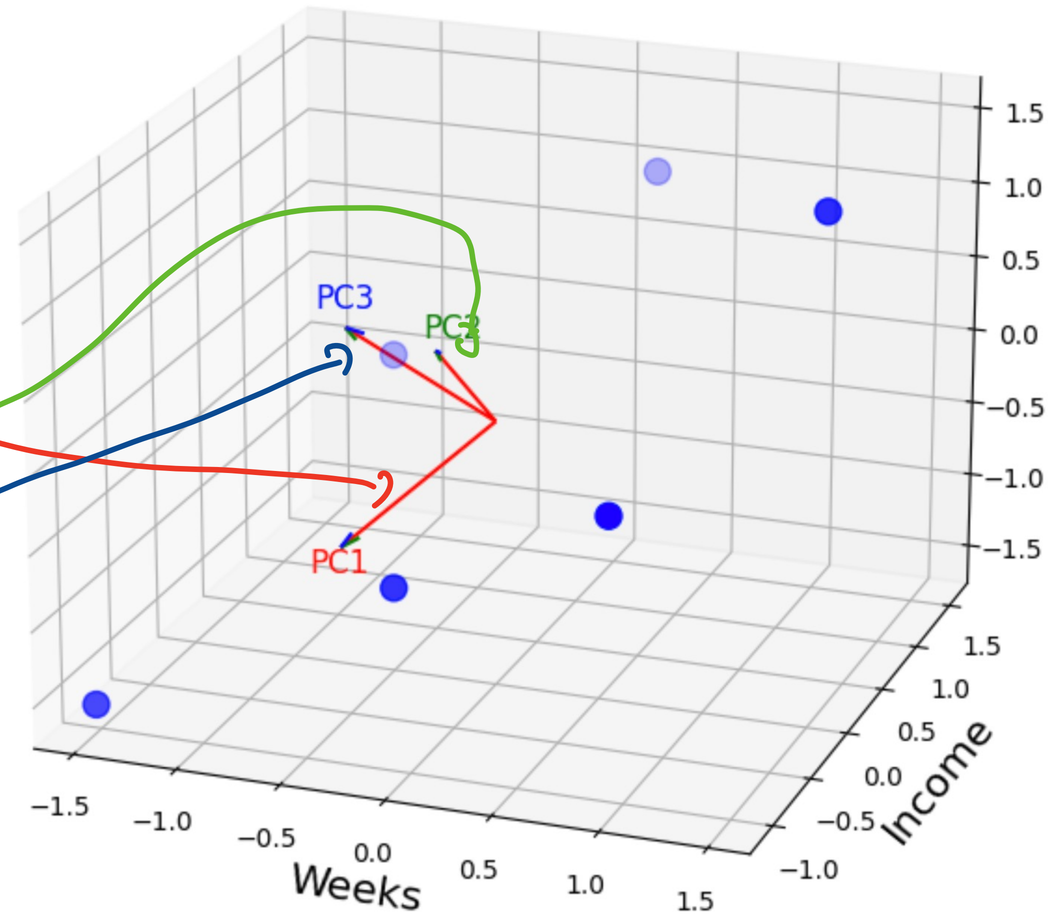
X =
```
[[-1.47742027 -1.09567331 -1.54437139]
 [-0.88645216  0.91678787 -0.16819886]
 [-0.17729043 -0.76026311 -0.71866788]
 [ 0.17729043  1.58760826  0.93273916]
 [ 0.88645216 -0.89442719  0.01529081]
 [ 1.47742027  0.24596748  1.48320817]]
```

$$\Sigma = \frac{1}{n} X^T X.$$

```
Covariance Matrix:
 [[1.2        0.15857338 0.98568818]
 [0.15857338 1.2        0.78325387]
 [0.98568818 0.78325387 1.2       ]]

Eigenvalues:
 [2.53905459 1.04565515 0.01529026]

Eigenvectors:
 [[-0.55943729 -0.61904027 -0.55119784]
 [-0.4668611   0.78481159 -0.40756775]
 [-0.68488731 -0.02932423  0.72805869]]
```

PCA/Tebasaki Example: Step 2 – Eigenvectors

```
Covariance Matrix:
 [[1.2        0.15857338 0.98568818]
 [0.15857338 1.2        0.78325387]
 [0.98568818 0.78325387 1.2       ]]

Eigenvalues:
 [2.53905459 1.04565515 0.01529026]

Eigenvectors:
 [[-0.55943729 -0.61904027 -0.55119784]
 [-0.4668611   0.78481159 -0.40756775]
 [-0.68488731 -0.02932423  0.72805869]]
```



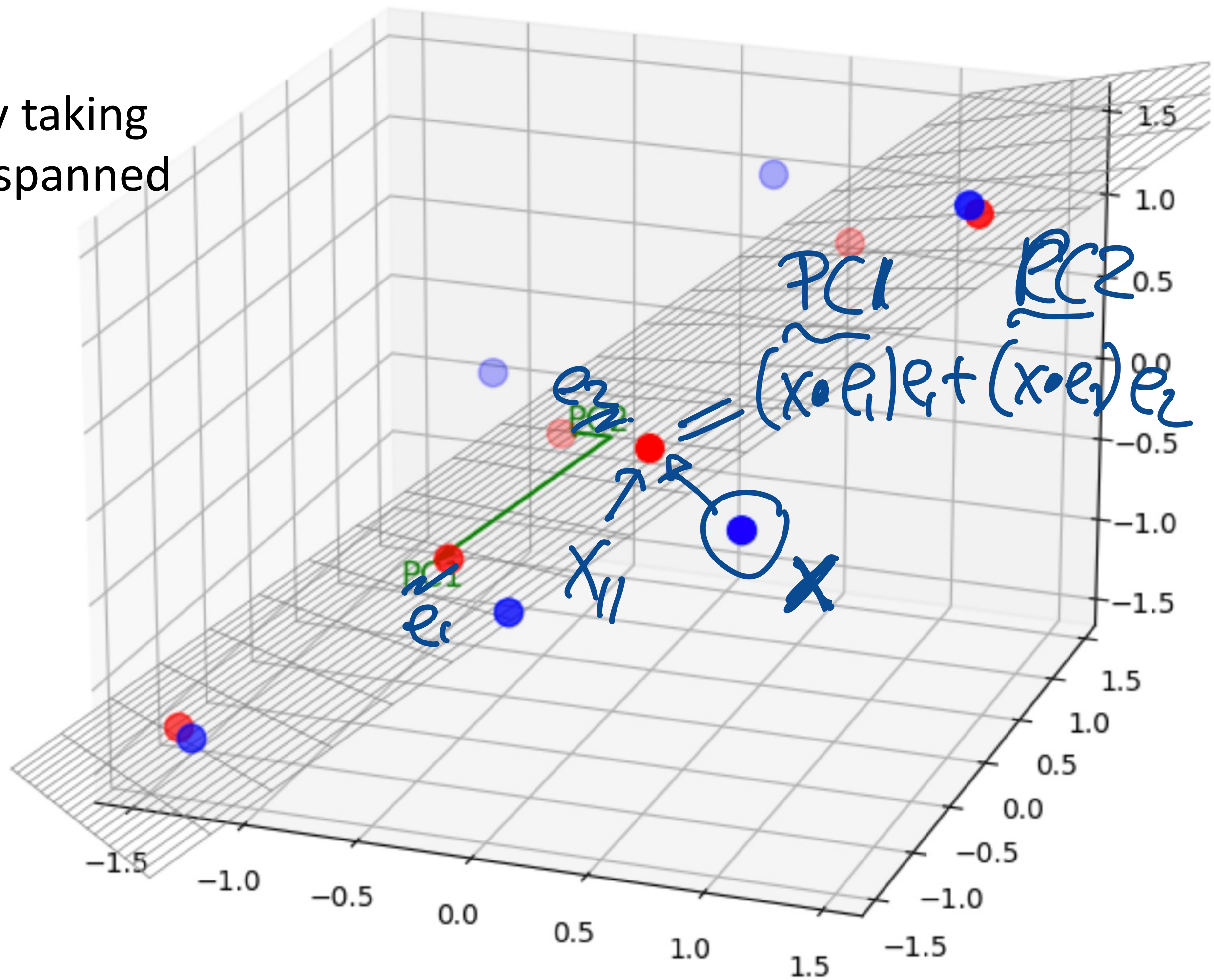
Linear algebra fact (Spectral theorem): One can always find an ONB of eigenvectors for the covariance matrix (since it is symmetric).

PCA/Tebasaki Example: Step 3 – Getting PCs

There are 3 principal components (eigenvectors). Taking all of them would not reduce the number of values we use to describe our point. We could try to just use the first two.

Consider the projection (given by taking the dot product) onto the plane spanned by the first two eigenvectors.

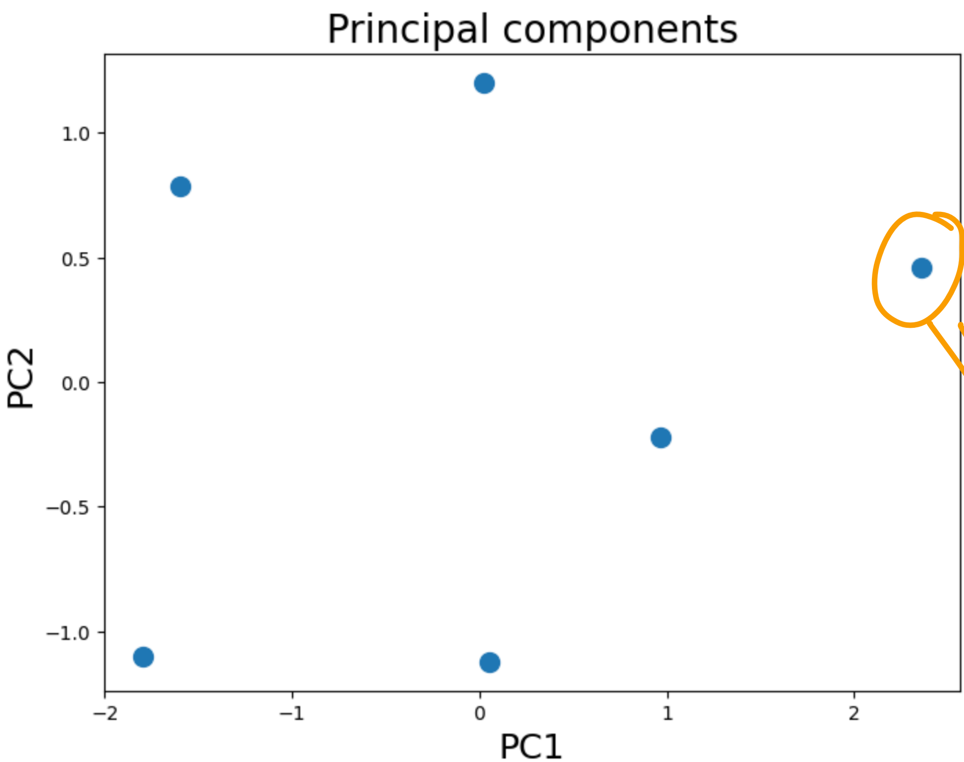
$$\begin{pmatrix} PC1 \\ PC2 \end{pmatrix} = \begin{pmatrix} x \cdot e_1 \\ x \cdot e_2 \end{pmatrix} = \underbrace{\overbrace{\begin{pmatrix} - & e_1 & - \\ - & e_2 & - \end{pmatrix}}^{3}}_{= (e_1 \; e_2)^T} X$$



PCA/Tebasaki Example: Step 3 – Plotting PC1,2

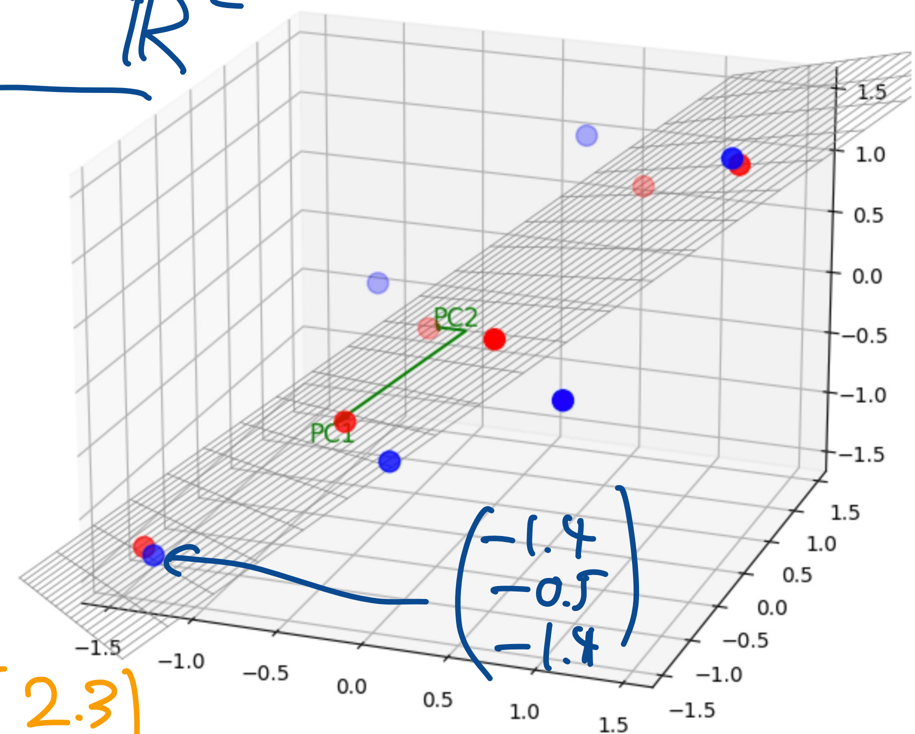
$$\begin{pmatrix} | & | & | \\ e_1 & e_2 & e_3 \\ | & | & | \end{pmatrix}\begin{pmatrix} - e_1 - \\ - e_2 - \\ - e_3 - \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} - e_1 - \\ - e_2 - \end{pmatrix}$$

$\mathbb{R}^2$ $\qquad$ $\mathbb{R}^2$



2 dimensional representation of our data

$\begin{pmatrix} 2.3 \\ 0.5 \end{pmatrix}$ $\qquad$ p''



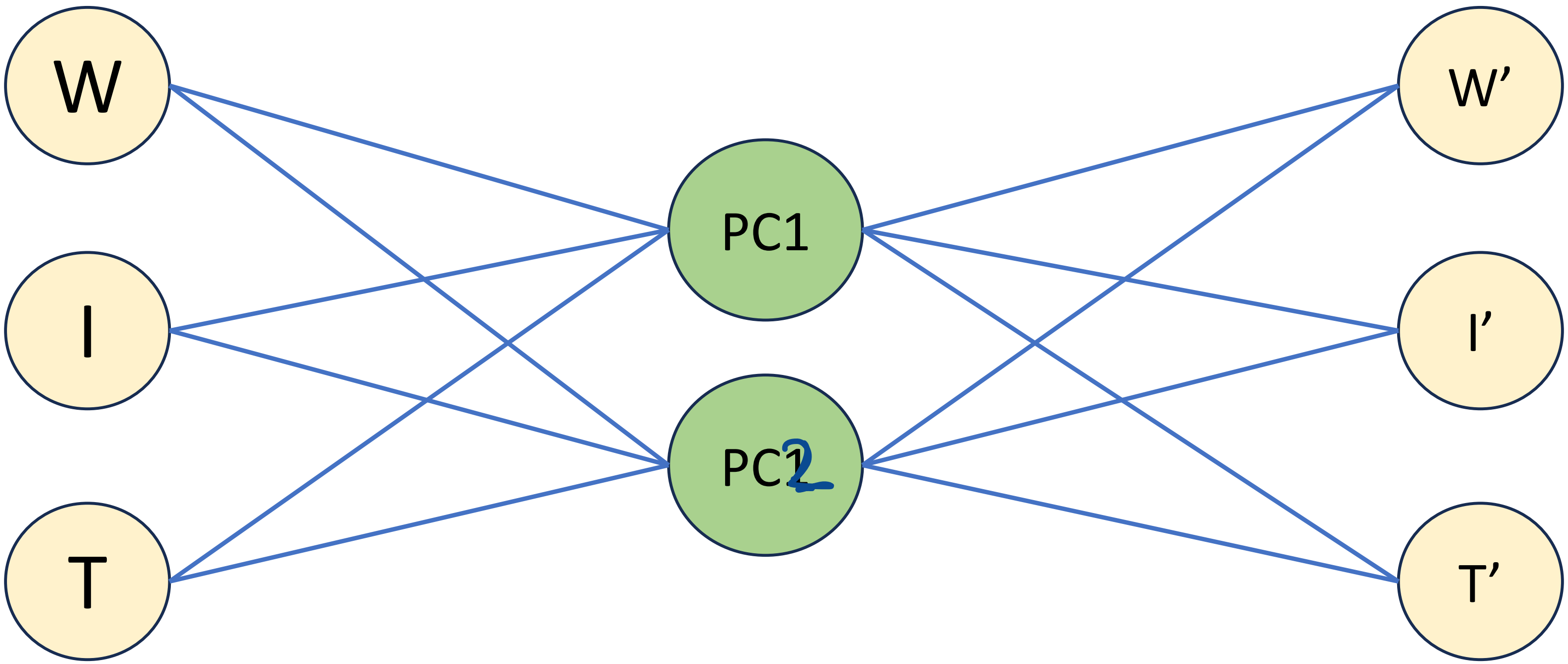
$\begin{pmatrix} -1.4 \\ -0.5 \\ -1.4 \end{pmatrix}$

$$\begin{array}{c c} & 2 \\ 3 & \begin{pmatrix} | & | \\ e_1 & e_2 \\ | & | \end{pmatrix} p \end{array}$$

Now Python examples!

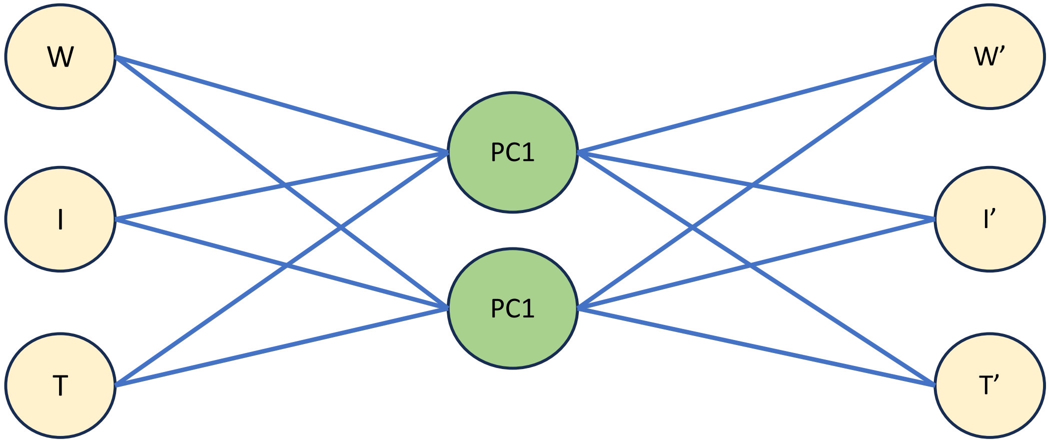
PCA – different interpretation

What we did: Have data in 3 dimensions, reduce it to 2 dimensions such that (if we go back to 3) we do not lose a lot of information.



PCA – different interpretation

What we did: Have data in 3 dimensions, reduce it to 2 dimensions such that (if we go back to 3) we do not lose a lot of information.



This can be seen as a neural network without bias and with the identity function as an activation function.

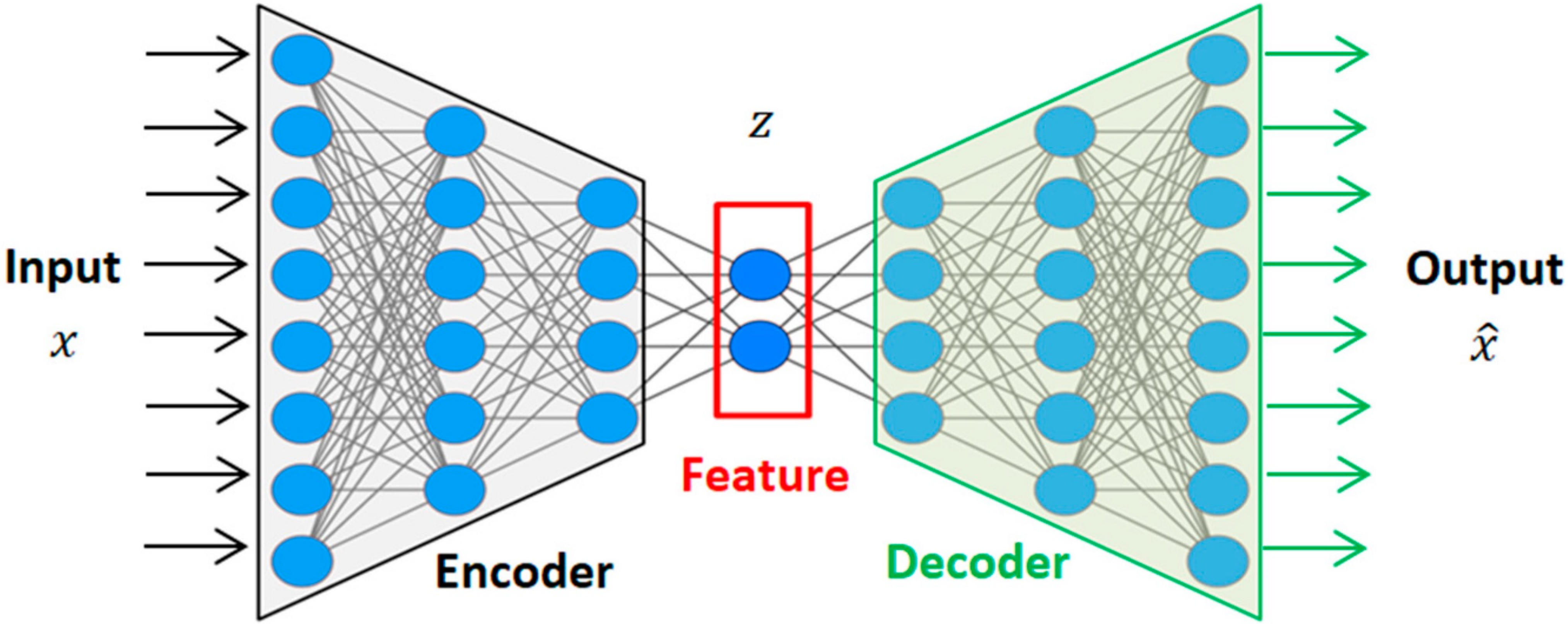
This is the basic idea of an autoencoder.

Autoencoder

An autoencoder is defined by the following components:

Two sets: the space of decoded messages $\mathcal{X}$; the space of encoded messages $\mathcal{Z}$. Almost always, both $\mathcal{X}$ and $\mathcal{Z}$ are Euclidean spaces, that is, $\mathcal{X} = \mathbb{R}^m$, $\mathcal{Z} = \mathbb{R}^n$ for some m, n.

Two parametrized families of functions: the encoder family $E_\phi : \mathcal{X} \rightarrow \mathcal{Z}$, parametrized by ϕ; the decoder family $D_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, parametrized by θ.



Autoencoder Application: Deep fakes