

# Dimensional reduction

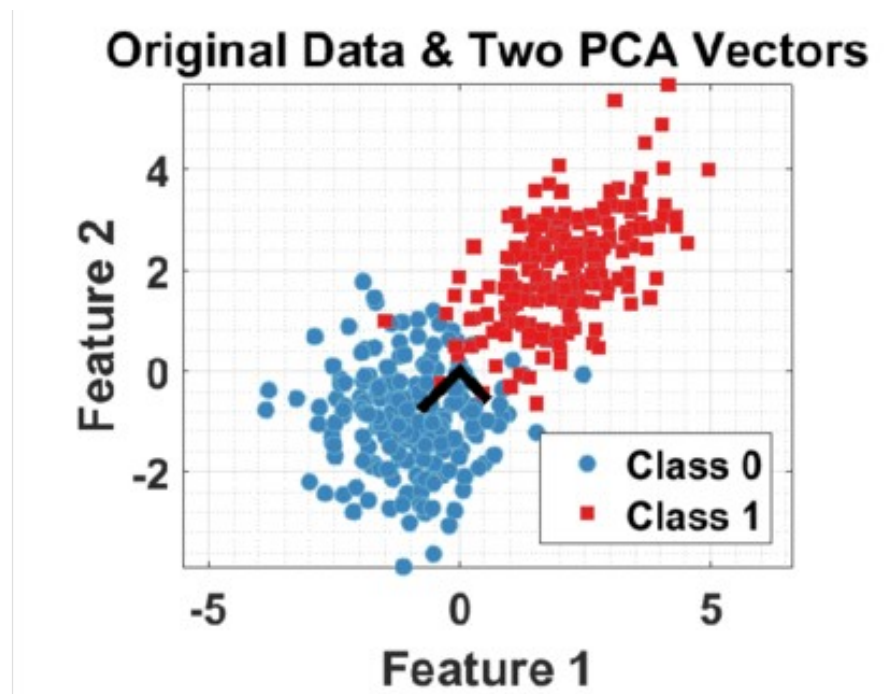
## Generalities

- Transform input features with a non-injective application :  $\phi : \mathbb{R}^D \longrightarrow \mathbb{R}^p, p < D$
- Why would we do that?
- - **eliminate redundancy** in the input information
  - reduce “noise” ( $\sim$  useless information)
  - Avoid the « ***Curse of dimensionality*** » (large  $D$  )
  - helps the learning:
    - \* **speed**: helps a lot
    - \* **performance**: it depends
  - visualization (but there's better, e.g. *t-SNE*)
- It's a form of ***data compression***

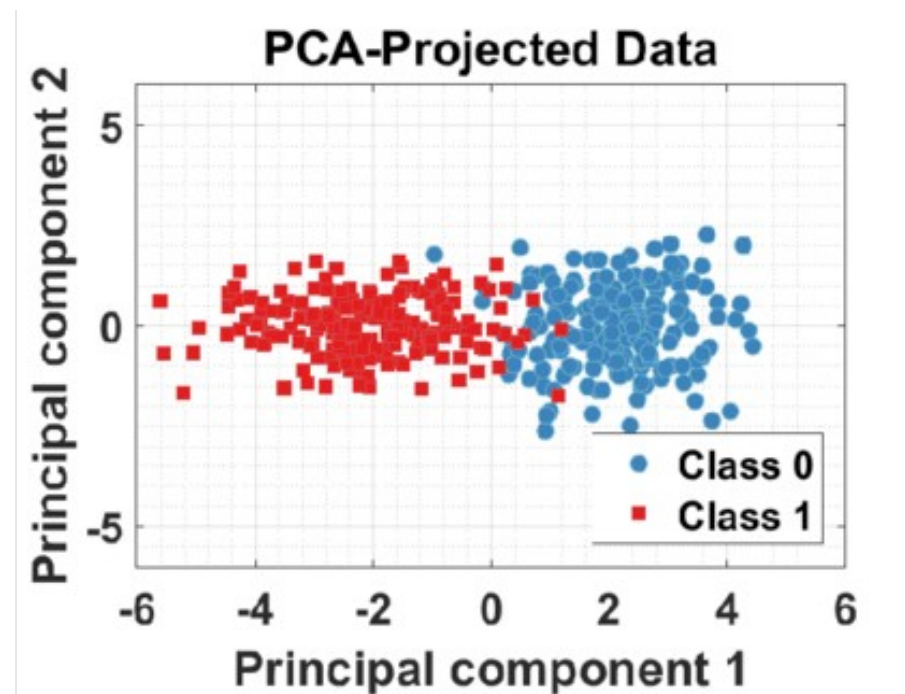
# Dimensional reduction

## *Principal Component Analysis*

- Linear application towards a smaller subspace  
→ projection on a hyper-plane



(3 dimensions, or more)



(2 dimensions shown)

# PCA - first perspective: Maximize the Variance

- **Maximize** the **variance** of projected data along direction  $\mathbf{u}_i$  :  $\text{Var} = \mathbf{u}_i^T \mathbf{C} \mathbf{u}_i$  with  $\mathbf{C}$  the **covariance**

$$\mathbf{C} = \frac{1}{N} \sum_n (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T$$

under the constraint :  $\mathbf{u}_i^T \mathbf{u}_i = 1$  , we find:

$$\Rightarrow \mathbf{C} \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

→ take the  $p$  first **eigenvectors** of the covariance matrix  $\mathbf{C}$

- Full proof: *Bishop book*, sec. 12.1.1, page 561-563<sub>3</sub>

# Proof

# PCA

## The “algorithm”

- Compute **covariance** matrix (centered data)

$$\mathbf{C} = \frac{1}{N} \sum_n^N (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T$$

- Diagonalize  $C$ :  $C = V \Lambda V^{-1}$
- Keep only the first  $p$  eigenmodes:  $P = V[:, p]$   
(keep  $p$  columns.. be careful with *np.linalg* )

$$P = \left( \begin{pmatrix} \cdot \\ v_1 \\ \cdot \end{pmatrix} \quad \begin{pmatrix} \cdot \\ v_2 \\ \cdot \end{pmatrix} \quad \dots \quad \begin{pmatrix} \cdot \\ v_p \\ \cdot \end{pmatrix} \right)$$

- transform:  $\vec{x}_{n,transformed} = P \cdot (\vec{x}_n - \langle \vec{x} \rangle)$

# PCA

## inverse transform

- Transformation forward:

$$\vec{x}_{n,transformed} = P \cdot (\vec{x}_n - \langle \vec{x} \rangle)$$

- Backwards:

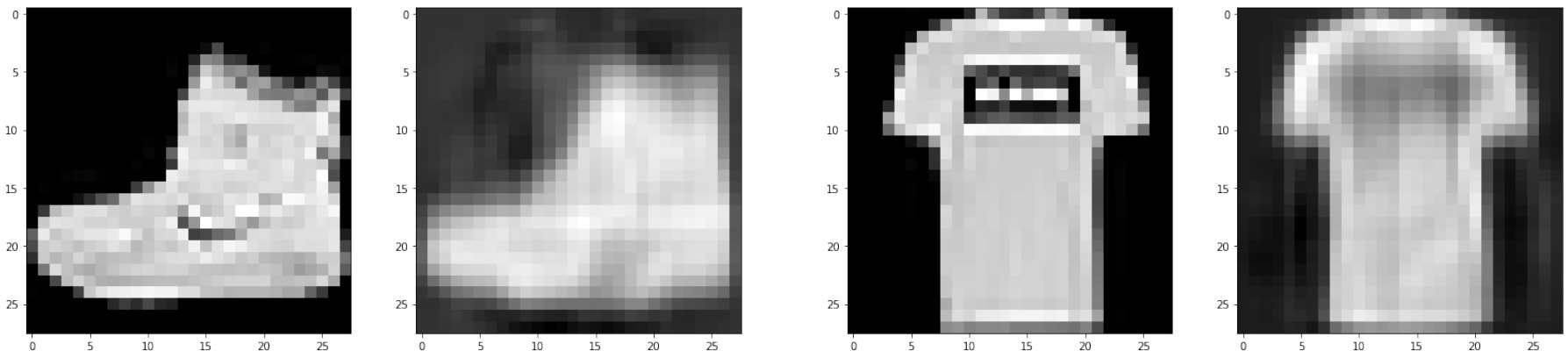
$$\vec{x}_{n,decompressed} = P^T \cdot \vec{x}_{n,transformed} - \langle \vec{x} \rangle$$

- Hence, reconstruction error is defined as:  
(quadratic reconstruction loss)

# PCA

## Concretely

- Full diagonalization: takes time  $O(D^3)$ , but iterative solution in time  $O(pD^2)$   
**Approximate solutions** are faster, but slightly random
- Fashion-MNIST ( $D=784$ ,  $D'=30$ ) :



→ TP5.2-PCA-from-scratch+overfitting.ipynb

- Interactive PCA, to build your intuition:  
<http://setosa.io/ev/principal-component-analysis/>

# Dimensional reduction

- PCA: **minimizes reconstruction error** (can be proven)
- PCA limitation: reduces the representation dimensionality, **independently from the labels** (class or target value) of examples
- *Independent Component Analysis (ICA)*: deals with correlations of order  $> 2$
- ***Auto Encoders (AE) and Variational Auto Encoders (VAE)***  
→ also compress, and make generative models



# Key concepts

- Curse of dimensionality
- PCA, variance, covariance, projection, reconstruction error
- Limits of PCA

**New ML definition: building a *model* of the data**