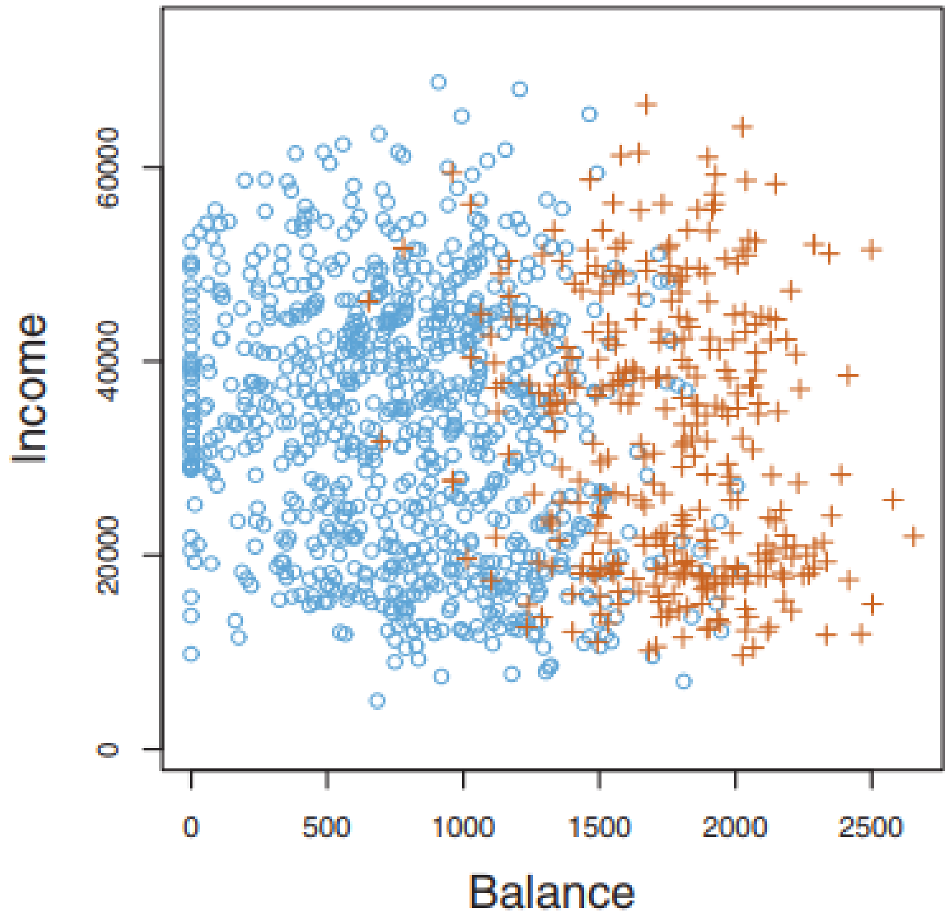
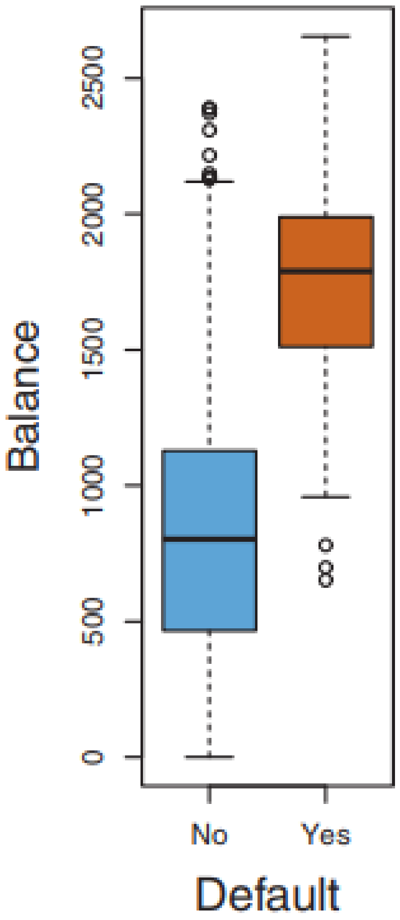
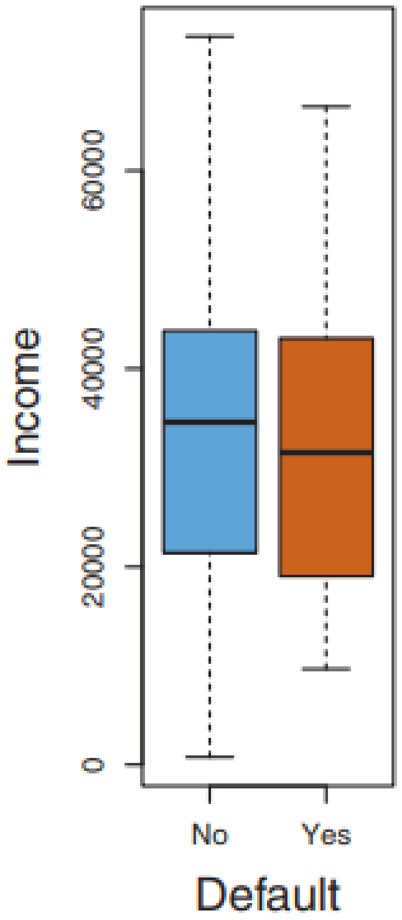
Dimensionality Reduction
Feature Selection & Extraction

Jing Sun

Is “Income” an informative feature?





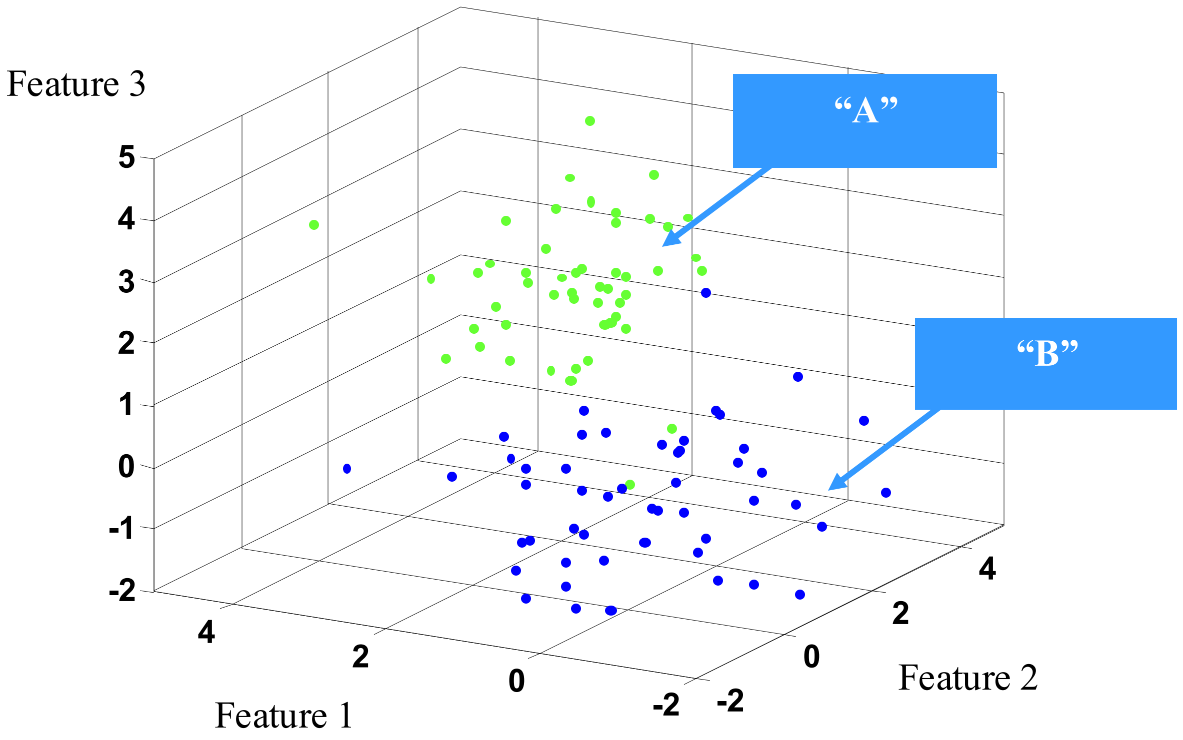


TU Delft

Feature Space

- A p-dimensional space,

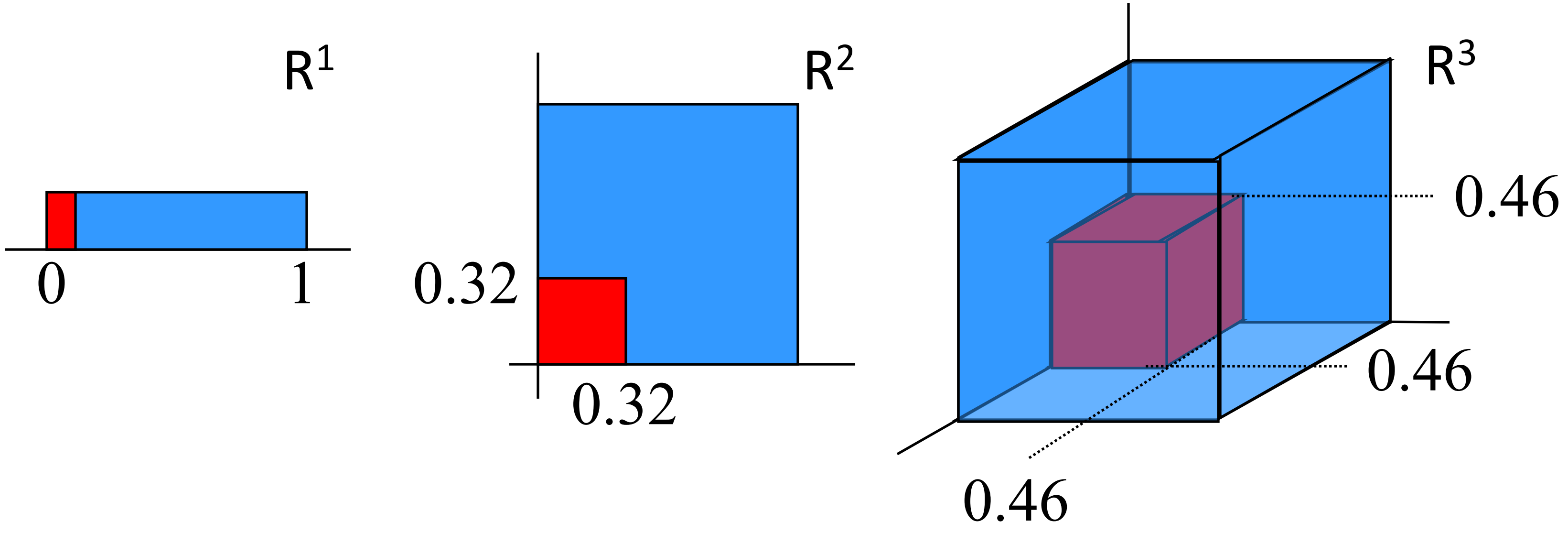
in which each dimension is a feature
containing n [labeled] samples [objects]



- What will happen if p is very large?
- **[the curse of dimensionality]**

- In high-dimensional spaces, our 2D/3D intuition does not work anymore...

High-Dimensional Spaces



- Example:
- Neighborhood capturing 10% of uniformly distributed data in hypercube
- E.g. in $\mathbb{R}^{20}$ side length of $\sqrt[20]{.1} \approx 0.89$

- So, not a small block anymore…

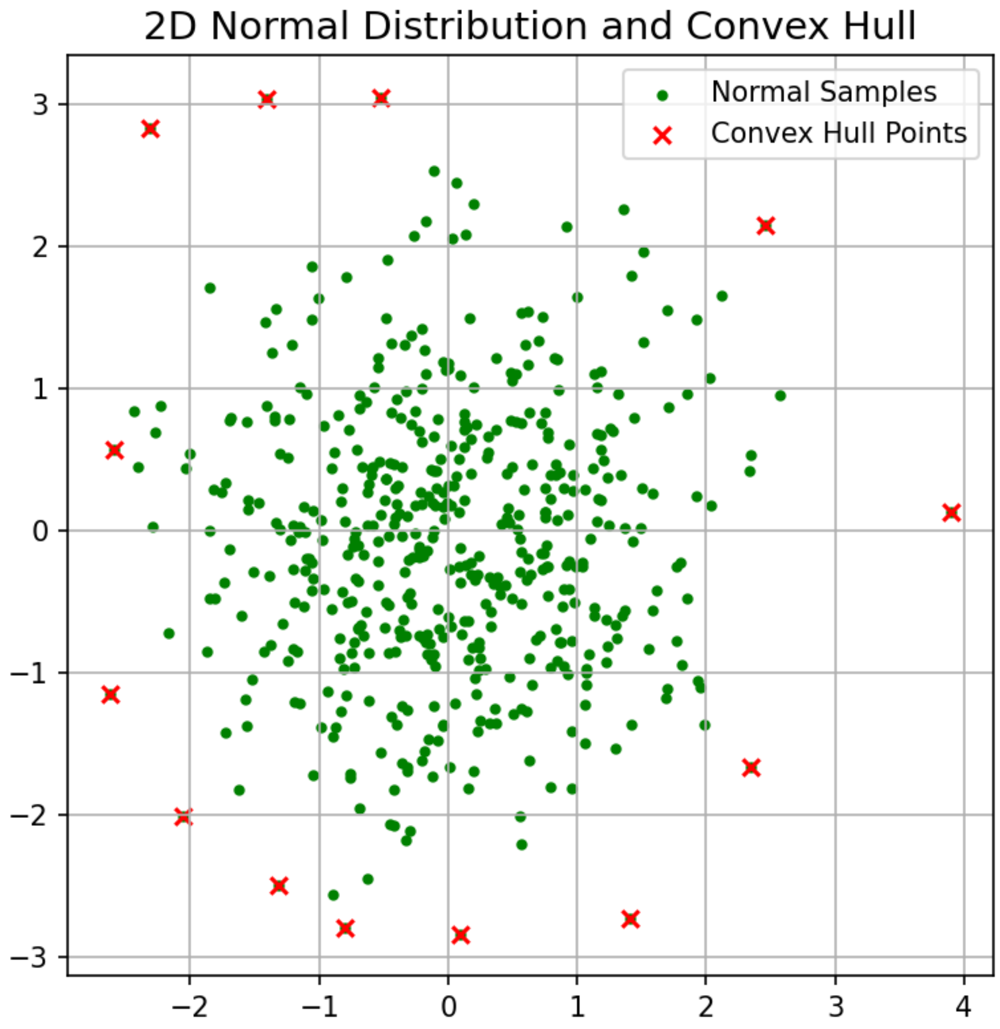
High-Dimensional Spaces

- Example: Boundary points ?

500 samples from normal distribution

In a 2-D space,

only 2% are on the convex hull



TU Delft

High-Dimensional Spaces

- Example: Boundary points ?

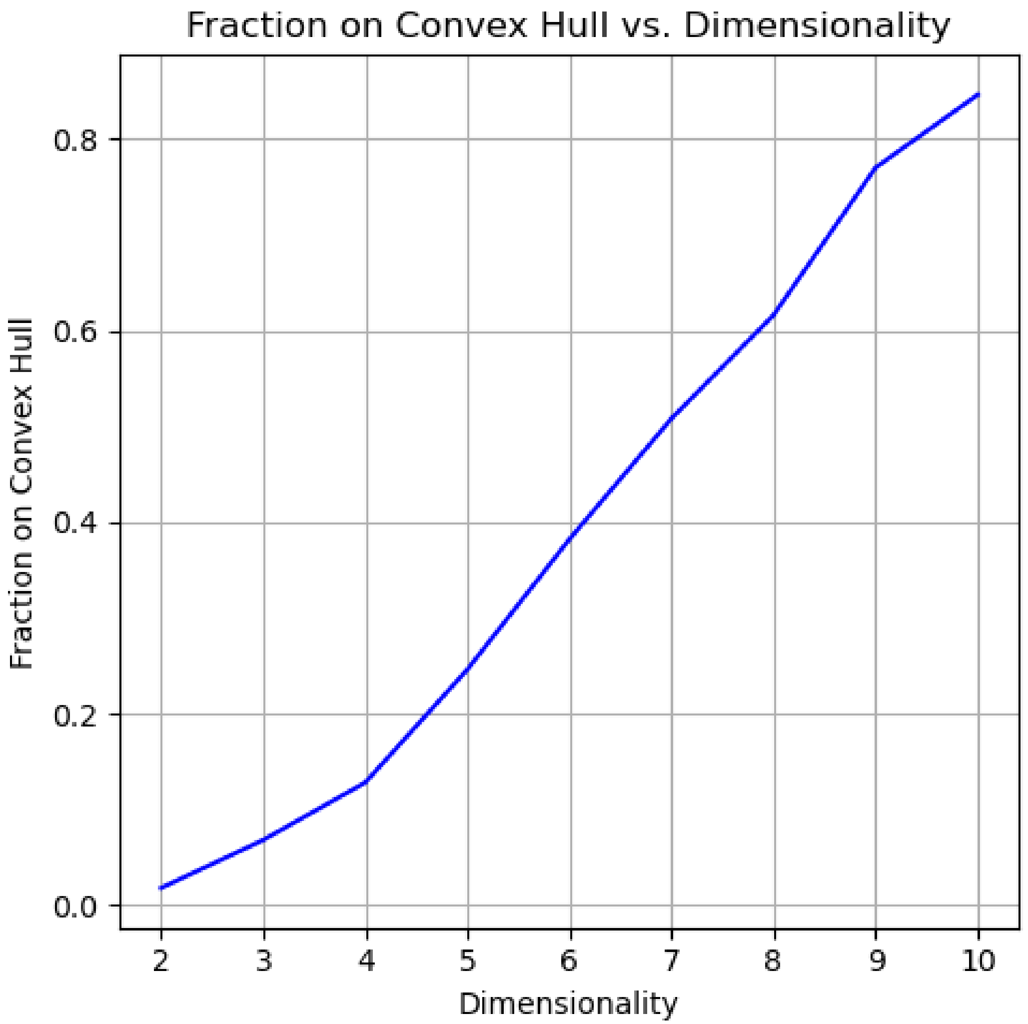
500 samples from normal distribution

In a 2-D space,

only 2% are on the convex hull

In a 20-D space,

95% are on the convex hull !



High-Dimensional Spaces

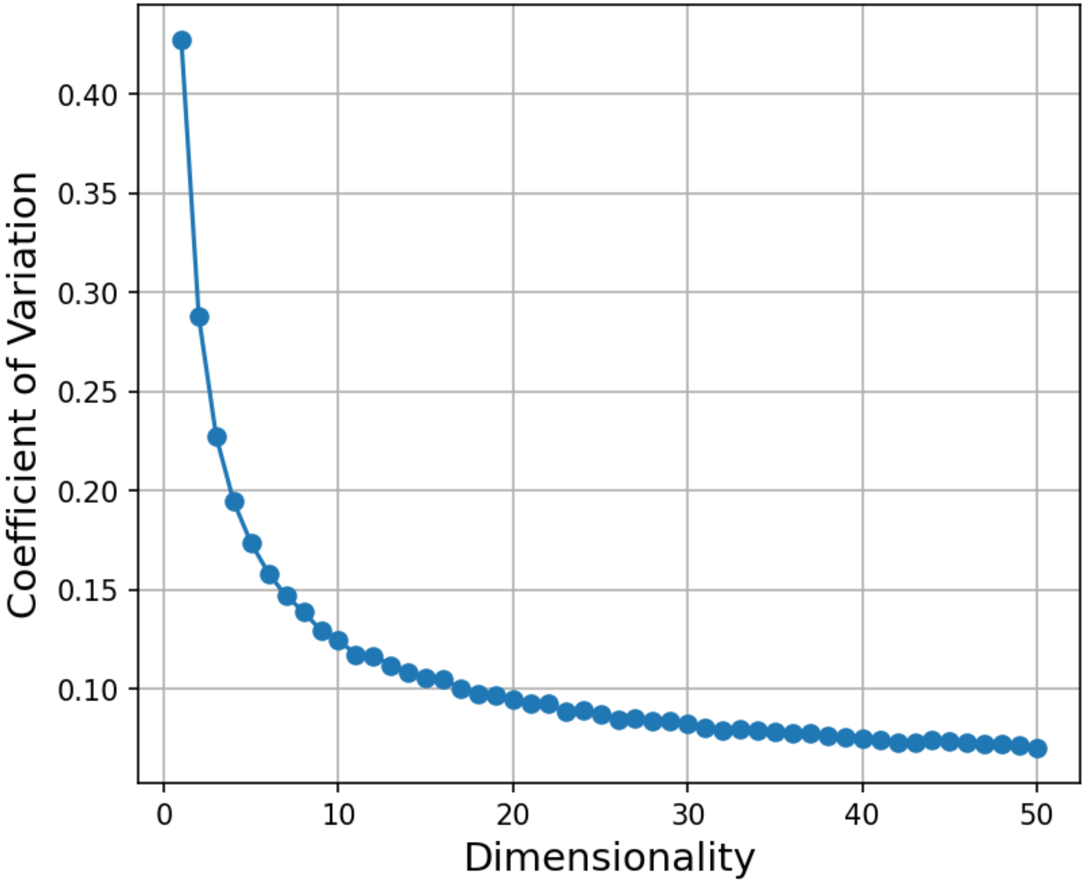
- Example: Points tend to have equal distances

200 samples from normal distribution

$N(2000, 8000)$

In a $\mathbb{R}^1$ to $\mathbb{R}^{1000}$ space

Consider $\frac{\text{std}(d^2)}{\text{mean}(d^2)}$ for squared distance d^2



TU Delft

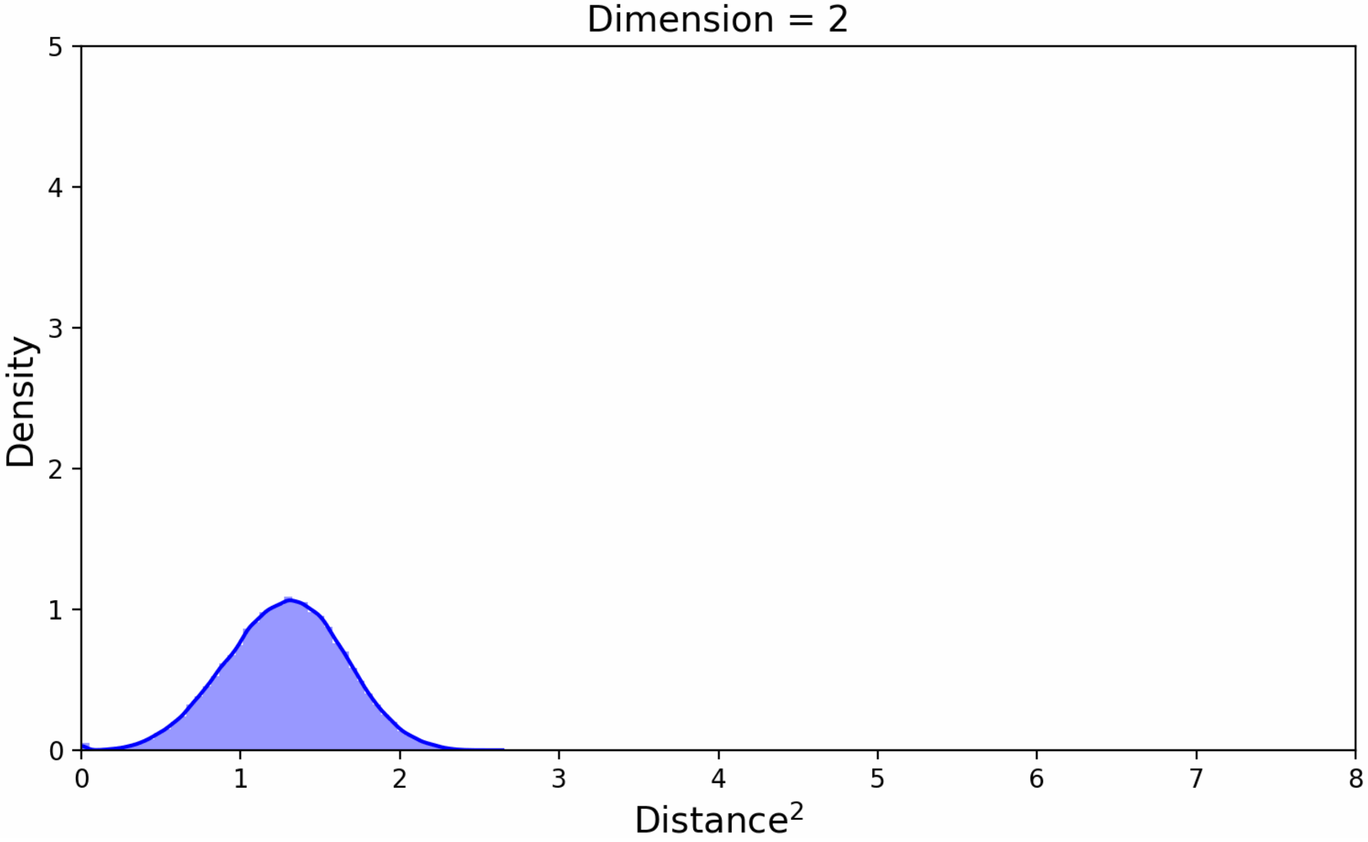
High-Dimensional Spaces

- Example: Points tend to have equal distances

200 samples from normal distribution

$N(2000, 8000)$

In a $\mathbb{R}^1$ to $\mathbb{R}^{1000}$ space

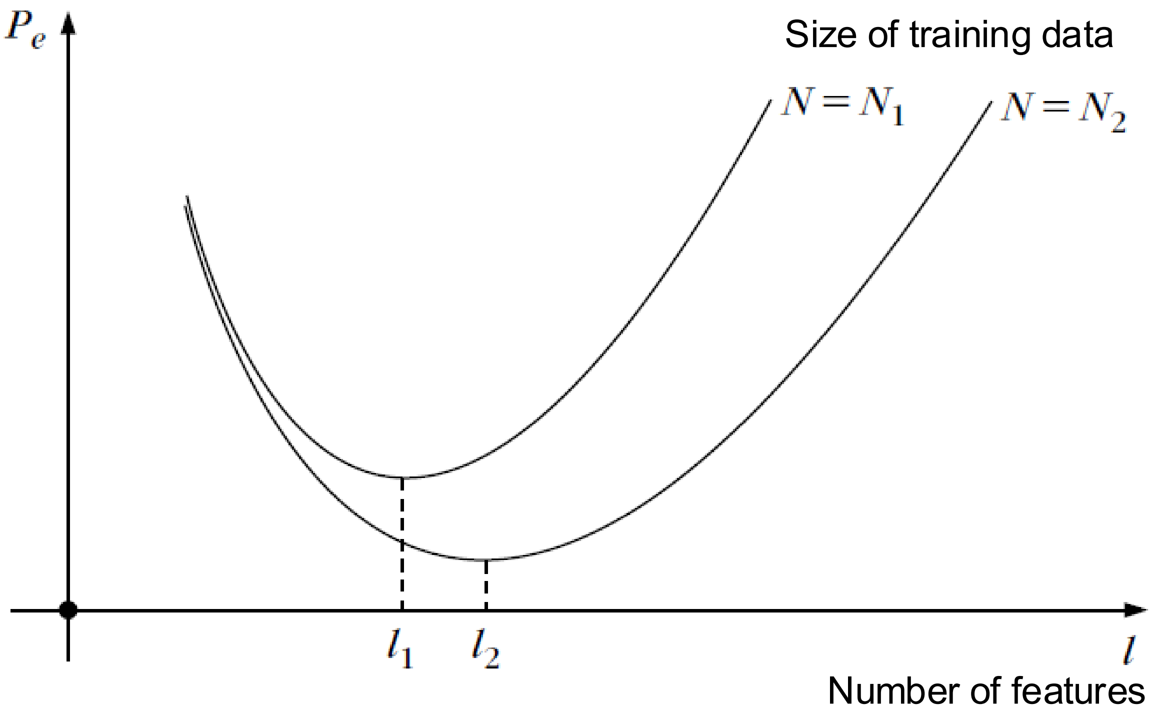


TU Delft

Dimensionality Reduction

- Problem: **too few samples in too many dimensions**
  [**the curse of dimensionality**]

- Solution: drop dimensions / features
  - Feature selection
  - Feature extraction



TU Delft

Dimensionality Reduction

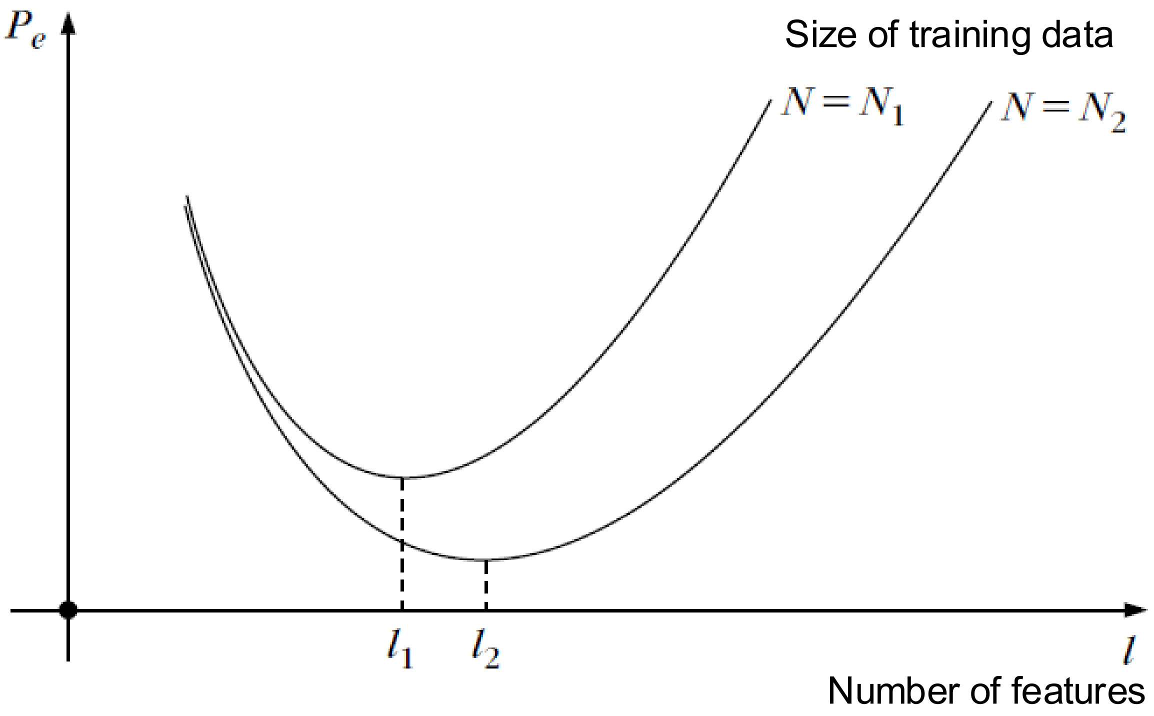
- Uses/Benefits :
  - Fewer parameters give **faster** algorithms and parameters are **easier** to estimate
  - **Explaining** which measurements are useful and which are not [**reducing redundancy**]
  - **Visualization of data** can be a powerful tool when designing pattern recognition systems

Dimensionality Reduction

- Problem: **too few samples in too many dimensions**
  [the curse of dimensionality]

- Solution: drop dimensions / features
  - Feature selection
  - Feature extraction

- **Questions:**
  - **Which dimensions to drop?**
  - **What feature subset to keep?**



TU Delft

Dimensionality Reduction by Selection or Extraction

- Overview – **Feature Selection** vs **Feature Extraction**

- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches
  - Sequential **feature selection** (individual, forward, backward, etc.)
  - Principal Component Analysis & Recall LDA (∈ linear **feature extraction**)

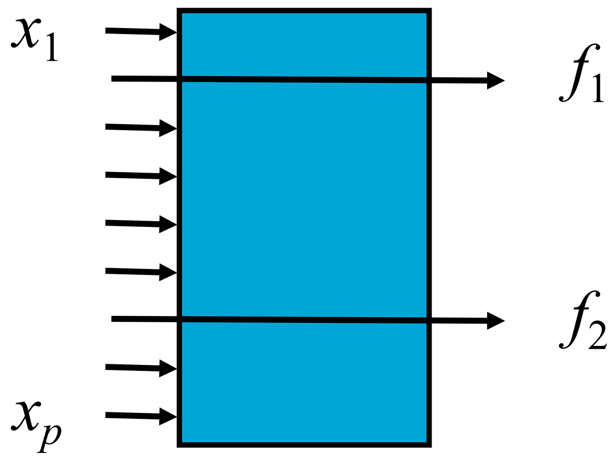
Dimensionality Reduction by Selection or Extraction

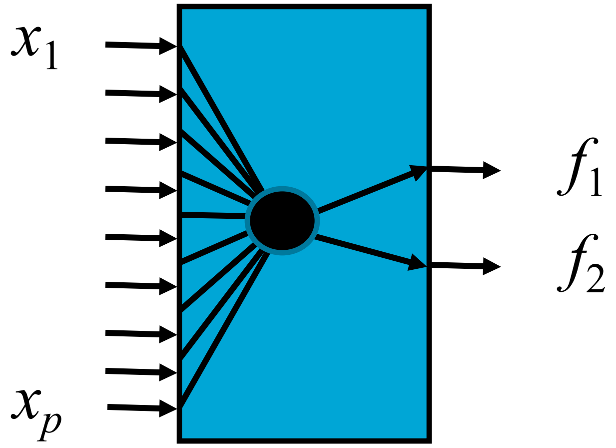
- Overview – **Feature Selection** vs **Feature Extraction**

- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches
  - Sequential **feature selection** (individual, forward, backward, etc.)
  - Principal Component Analysis & Recall LDA ($\in$ linear **feature extraction**)

Feature Selection vs Extraction





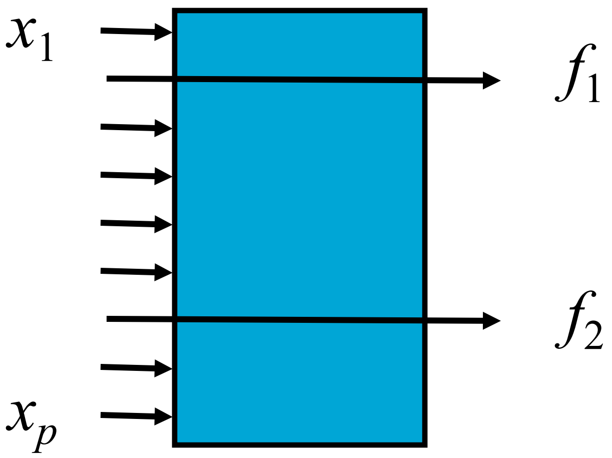
Feature Selection vs Extraction

- Feature selection :

  Select d **out of** p measurements

  Only a subset of the original features are selected.

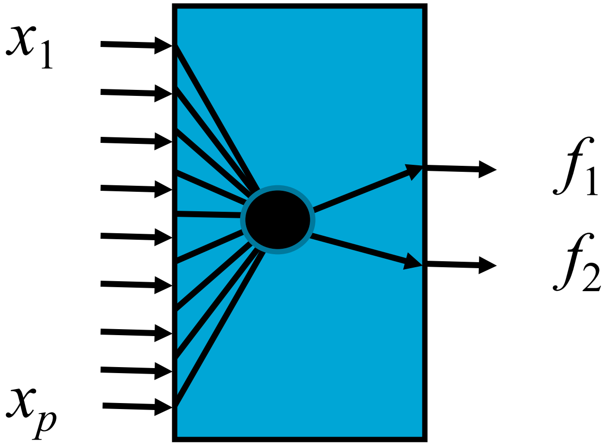
  There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ subsets.



- Feature extraction :

  Map p measurements **to** d measurements

  All original features are used (they are transferred)



Dimensionality Reduction by Selection or Extraction

- Overview – **Feature Selection** vs **Feature Extraction**

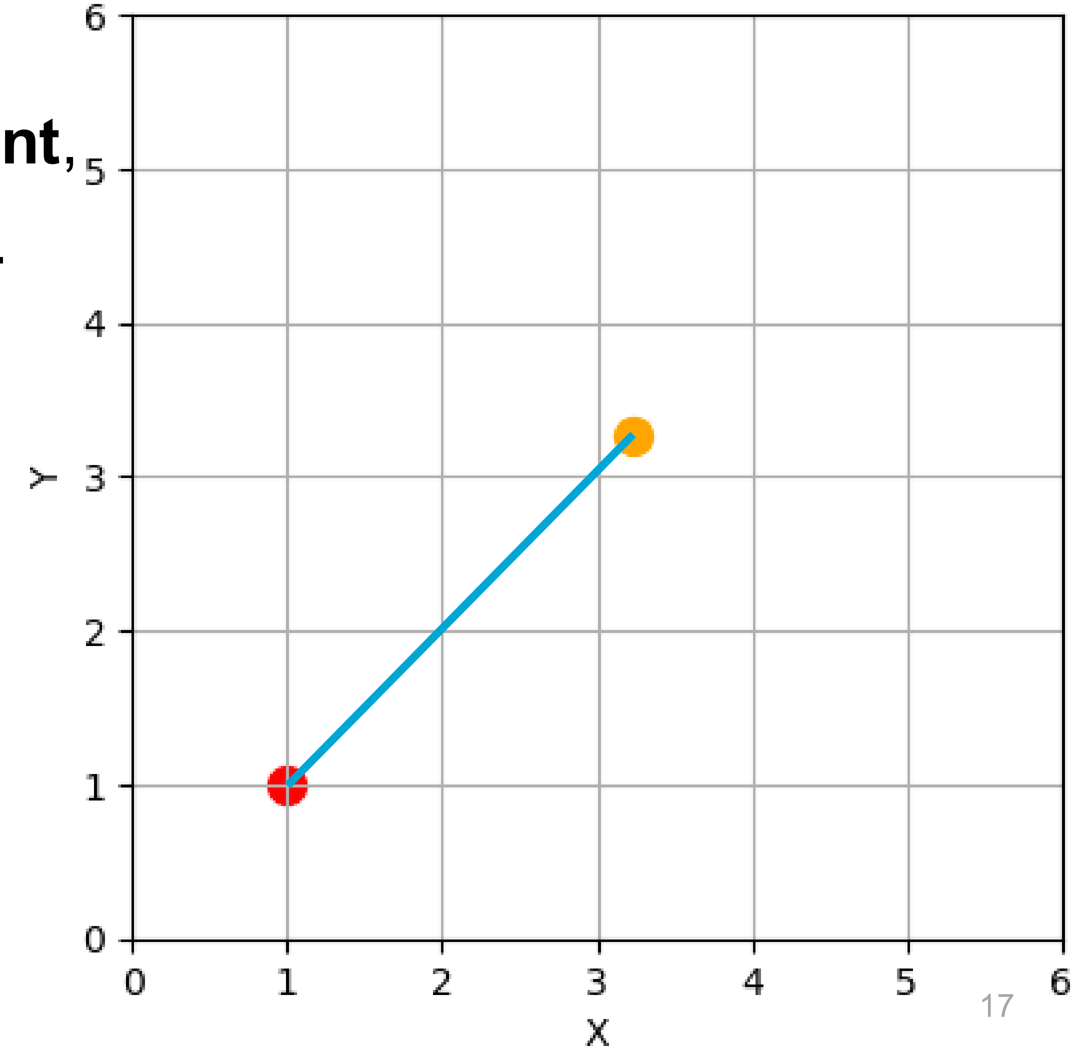
- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches
  - Sequential **feature selection** (individual, forward, backward, etc.)
  - Principal Component Analysis & Recall LDA ($\in$ linear **feature extraction**)

Why Mahalanobis distance?

- When measuring the distance
  from **a single point** to **another single point**,
  using (squared) Euclidean distance is fine.

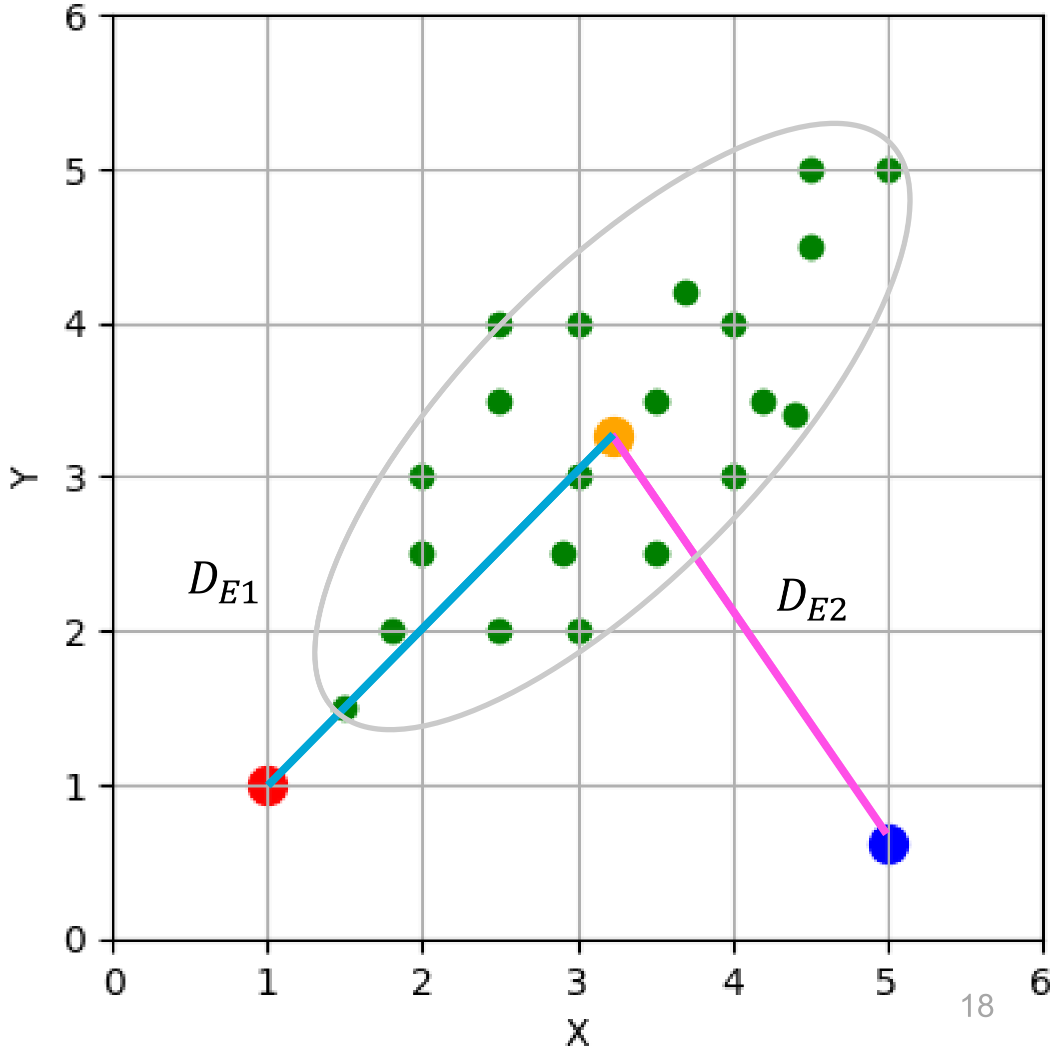
$$D_E = (x_{red} - x_{yellow})^2 + (y_{red} - y_{yellow})^2$$



TU Delft

Why Mahalanobis distance?

- However,
- when there is a group of data points:
- ***Centroid (mean vector)*** = $\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}$
- Euclidean distances $D_{E1} = D_{E2}$



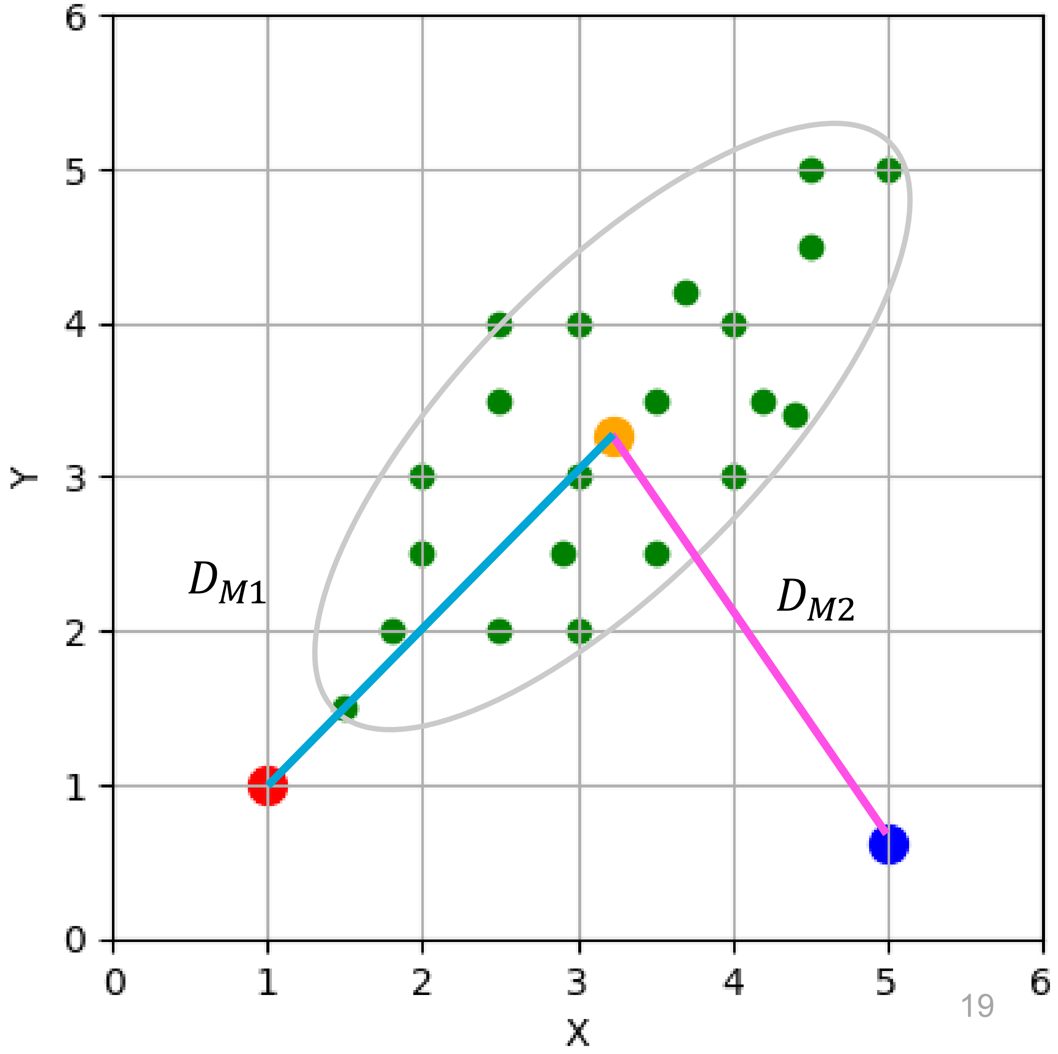
TU Delft

Mahalanobis distance

- Takes the **variance** into account.
- It is a distance measure
  between **a point** and **a distribution**.
- For red and blue points,

$$D_M = \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix}$$

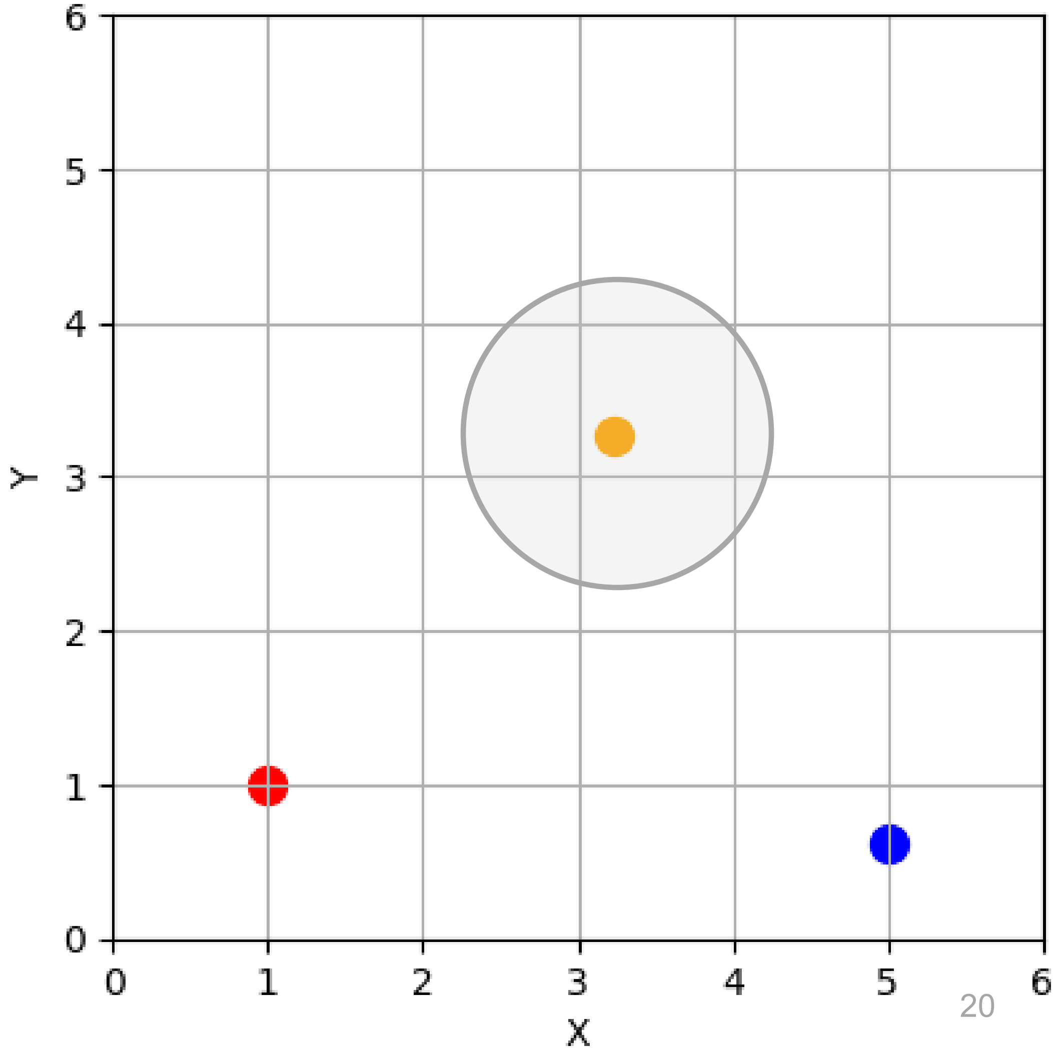
- You will see $D_{M2} > D_{M1}$



Mahalanobis distance

- Think about:
- What if $\boldsymbol{\Sigma}$ is an identity matrix?

$$D_M = \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix}^T \boldsymbol{I} \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix} = D_E$$



TU Delft

Mahalanobis distance

- Mahalanobis distance between two classes:
  - Assumes Gaussian distributions with equal covariance matrix
  - $D_M = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T {\boldsymbol{S}_W}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$
  - E.g., Exercise 6.21
  - What is this $\boldsymbol{S}_W$?

Dimensionality Reduction by Selection or Extraction

- Overview – **Feature Selection** vs **Feature Extraction**

- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches
  - Sequential **feature selection** (individual, forward, backward, etc.)
  - Principal Component Analysis & Recall LDA (∈ linear **feature extraction**)

Scatter Matrices

- Within-class scatter matrix:

$$\boldsymbol{S}_W = \sum_{i=1}^{M} \frac{n_i}{N} \boldsymbol{\Sigma}_i \,,$$

$\boldsymbol{\Sigma}_i$ is the **covariance matrix of class** w_i; M is the **number of classes**; n_i is the **number of samples in class** w_i, out of a **total** of N samples.

- Between-class scatter matrix:

$$\boldsymbol{S}_B = \sum_{i=1}^{M} \frac{n_i}{N} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T,$$

$\boldsymbol{\mu}_i$ is the **mean of class** w_i, $\boldsymbol{\mu}$ is the **global mean**.

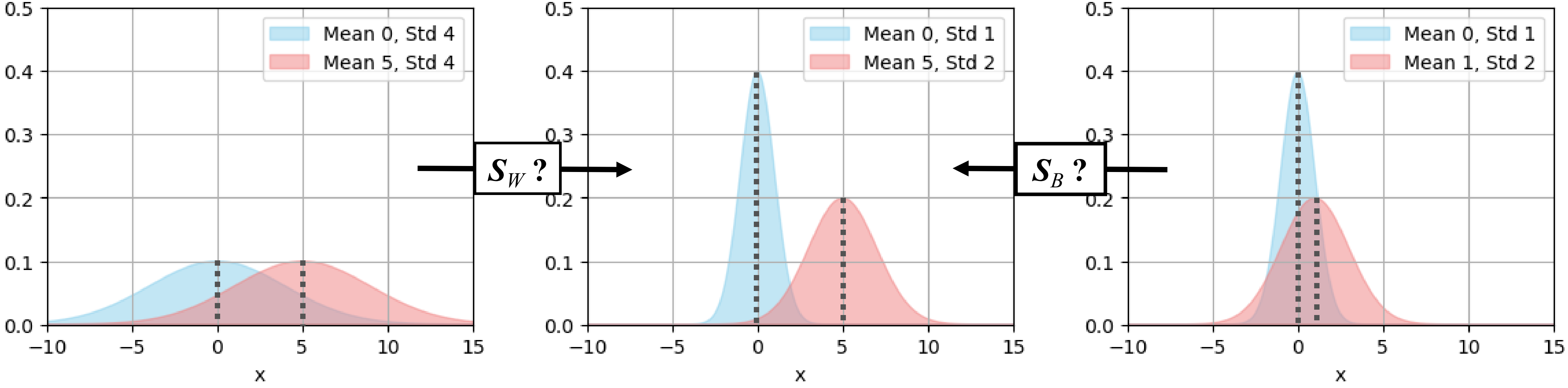
$$\boldsymbol{\mu} = \sum_{i=1}^{M} \frac{n_i}{N} \boldsymbol{\mu}_i$$

- Total scatter matrix: $\boldsymbol{S}_T = \boldsymbol{S}_W + \boldsymbol{S}_B$

Scatter Matrices

For a classification task

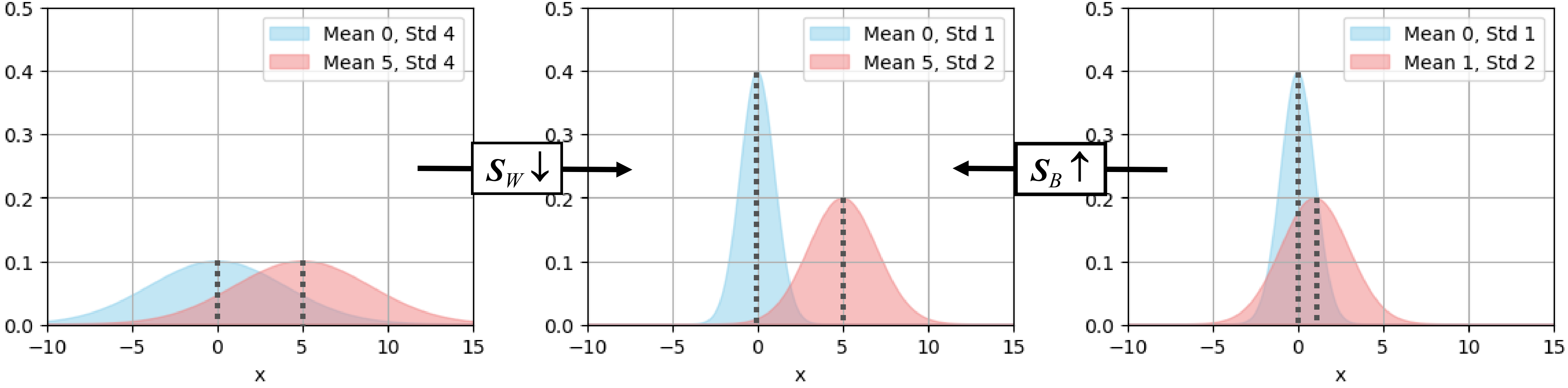
- $\boldsymbol{S}_W$ = "average class width"; ***the smaller, the better***
- $\boldsymbol{S}_B$ = "average distance between class means"; ***the larger, the better***
- $\boldsymbol{S}_T$ = "overall width"



Scatter Matrices

For a classification task

- $\boldsymbol{S}_W$ = "average class width"; ***the smaller, the better***
- $\boldsymbol{S}_B$ = "average distance between class means"; ***the larger, the better***
- $\boldsymbol{S}_T$ = "overall width"



Scatter-based Criteria

- $J_1 = \frac{trace\{\boldsymbol{S}_T\}}{trace\{\boldsymbol{S}_W\}}$
- $J_2 = \frac{|\boldsymbol{S}_T|}{|\boldsymbol{S}_W|}$
- etc.
- by using **various** **combinations** of $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$ in a “trace” or “determinant” formulation…

- PS: The “trace” is equal to the sum of the eigenvalues; the “determinant” is equal to their product.

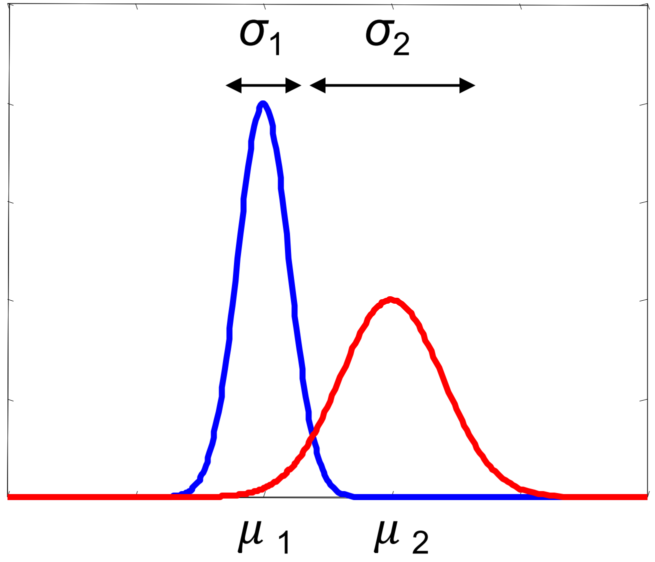
TU Delft

FDR: Fisher Discriminant Ratio

- 1-D, two-class problem

- $S_W \propto (\sigma_1^2 + \sigma_2^2)$, $S_B \propto (\mu_1 - \mu_2)^2$,
- Combining S_W and S_B, you get Fisher's criterion

$$J_F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$



- It is often used to quantify the separability capabilities of individual features.

TUDelft

Dimensionality Reduction by Selection or Extraction

- Overview – **Feature Selection** vs **Feature Extraction**

- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches
  - Sequential **feature selection** (individual, forward, backward, etc.)
  - Principal Component Analysis & Recall LDA ($\in$ linear **feature extraction**)

Which method would guarantee optimal performance?

- I have p features (let's say $p = 40$).
- I think this is too many to handle…
- I want to select d features from p.
- But what should $d =?$ Well, I'm not sure…

- What can I do?

Which method would guarantee optimal performance?

- Trying all possible feature combinations

$$\binom{p}{d} = \frac{p!}{d!\,(p-d)!}$$

- **Exhaustive** feature selection

$$\sum_{i=1}^{p} \binom{p}{i}$$ combinations

- If originally there are 4 features, we will end up with 15 combinations.

- $\binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 15$

- But, what if there are 40 features…? **-- over a billion! :-(**

Sub-optimal Strategies

- Trying all possible feature combinations

$$\binom{p}{d} = \frac{p!}{d!\,(p-d)!}$$

- **Exhaustive** feature selection

$$\sum_{i=1}^{p} \binom{p}{i} \quad \text{combinations}$$

- It can be super Expensive! And Exhaustive!! :-(
- Let’s use **Sequential Feature Selection**!

Feature Selection Methods

Filter approach

Wrapper approach

(Sequential Feature Selection Family)

- Individual Selection
- **Forward Selection (FS)**
- **Backward Selection (BS)**
- Bidirectional Selection
- Plus-L Take-away-R Selection
- Floating Selection

(***combine the idea*** *of forward and backward selections)*

Forward Selection (FS)

- Start with **empty feature set**

Forward Selection (FS)

- Start with empty feature set

- Compute the **criterion value** for **each feature individually** and **select the best one**,

Forward Selection (FS)

- Start with empty feature set

- Compute the **criterion value** for **each feature individually** and **select the best one**,

$x_2 > x_4 > x_2 > x_3 \ => x_2$

Forward Selection (FS)

- Start with empty feature set

- Compute the criterion value for each feature individually and select the best one,

$x_2 > x_4 > x_2 > x_3 \;\; => x_2$

- **Keep the winner** and compute the criterion value for **all two-feature combinations** that **include it**.

Forward Selection (FS)

- Start with empty feature set

- Compute the criterion value for each feature individually and select the best one,

$x_2 > x_4 > x_2 > x_3 \;\; => x_2$

- **Keep the winner** and compute the criterion value for **all two-feature combinations** that **include it**.

  $[x_2, x_1] > [x_2, x_4] > [x_2, x_3]$

Forward Selection (FS)

- Start with empty feature set

- Compute the criterion value for each feature individually and select the best one,

$x_2 > x_4 > x_2 > x_3 \;\; => x_2$

- Keep the winner and compute the criterion value for all two-feature combinations that include it.

$[x_2, x_1] > [x_2, x_4] > [x_2, x_3]$

… … until a **predefined number** of features are left.

TUDelft

Backward Selection (BS)

- Start with **all originally available features**

Backward Selection (BS)

- Start with all originally available features

- Compute the criterion value for **all possible combinations after eliminating one feature**,

Backward Selection (BS)

- Start with all originally available features

- Compute the criterion value for **all possible combinations after eliminating one feature**,

  $[x_1, x_2, x_4] > [x_1, x_2, x_3] > [x_2, x_3, x_4] > [x_1, x_3, x_4]$

  Keep the winner combination (**i.e.,** **remove one feature**);

Backward Selection (BS)

- Start with all originally available features

- Compute the criterion value for all possible combinations after eliminating one feature,

  $[x_1, x_2, x_4] > [x_1, x_2, x_3] > [x_2, x_3, x_4] > [x_1, x_3, x_4]$

  Keep the winner combination (i.e., remove one feature);
- Repeat step above: **from the winner vector**, **eliminate one feature**, and for **each of the resulting combinations**, compute the criterion value… …

Backward Selection (BS)

- Start with all originally available features

- Compute the criterion value for all possible combinations after eliminating one feature,

  $[x_1, x_2, x_4] > [x_1, x_2, x_3] > [x_2, x_3, x_4] > [x_1, x_3, x_4]$

  Keep the winner combination (i.e., remove one feature);

- Repeat step above: **from the winner vector**, **eliminate one feature**, and for **each of the resulting combinations**, compute the criterion value… …

  $[x_1, x_2] > [x_2, x_4] > [x_1, x_4]$

  … … until a **predefined number** of features are left.

TU Delft

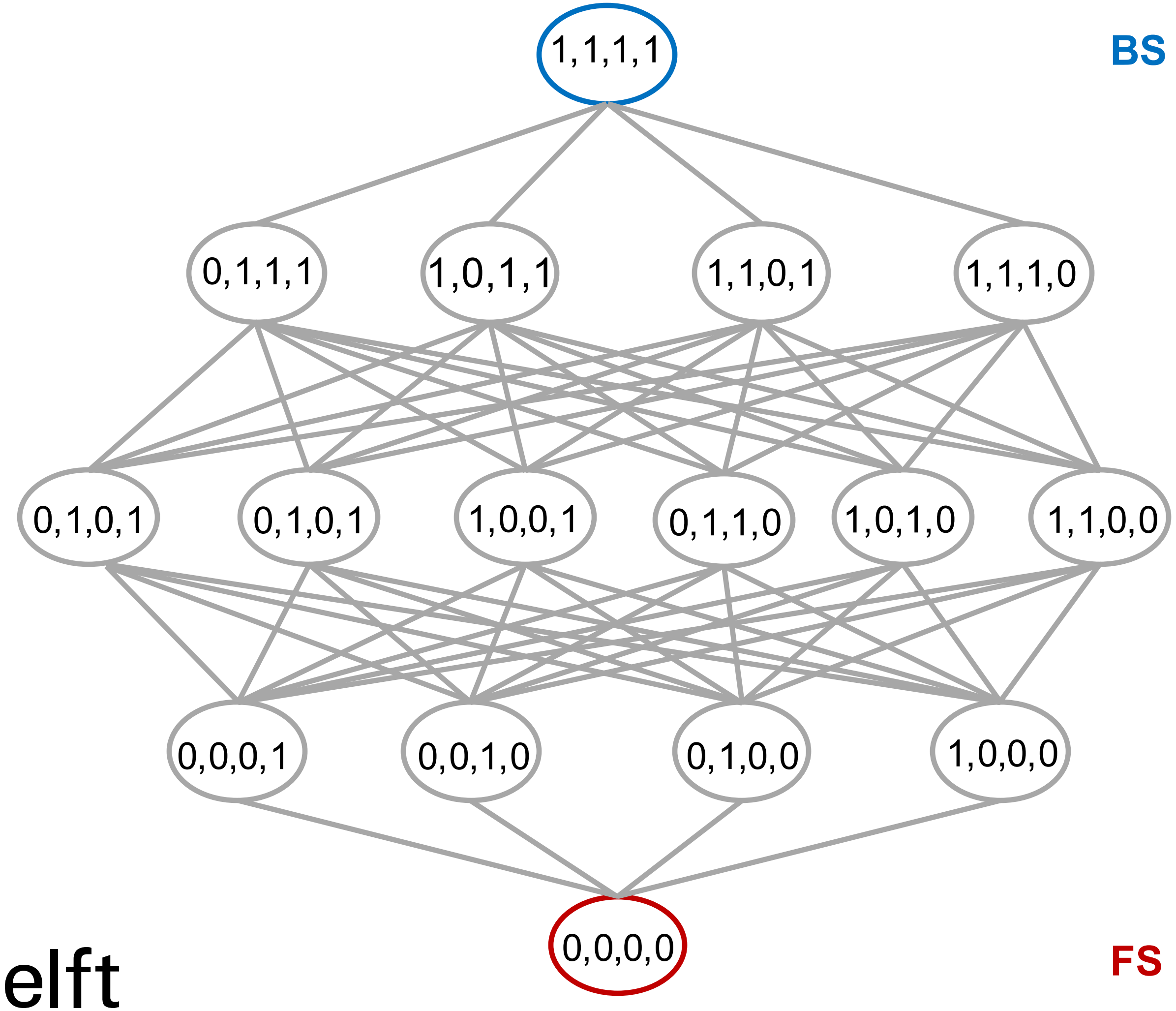
Bidirectional Selection

- It **applies FS and BS simultaneously**:
  - FS starts from the empty feature set.
  - BS starts from the full set of all originally available features.

- To make sure they **converge to the same solution:**
  - Features already selected by FS are **not removed** by BS.
  - Features already removed by BS are **not selected** by FS.

Bidirectional Selection

Four features in order of x_1, x_2, x_3, x_4,
1 means selected, 0 means not selected,
e.g., (0,0,0,1) means only x_4 is selected.

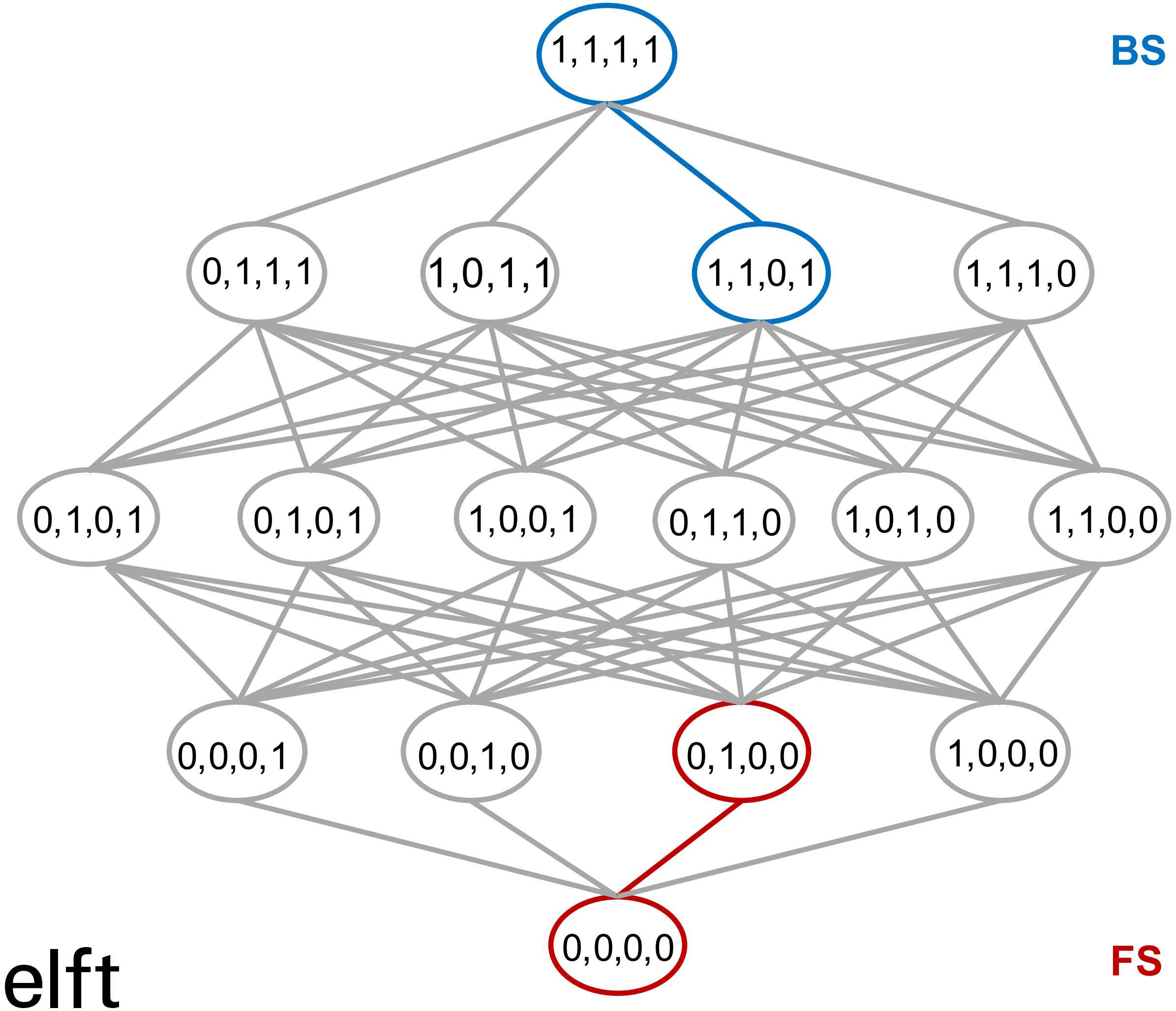


BS Full set of all originally available features

FS Empty feature set

Bidirectional Selection

Four features in order of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, $\boldsymbol{x}_4$,
1 means selected, 0 means not selected,
e.g., (0,0,0,1) means only x_4 is selected.



BS Full set of all originally available features

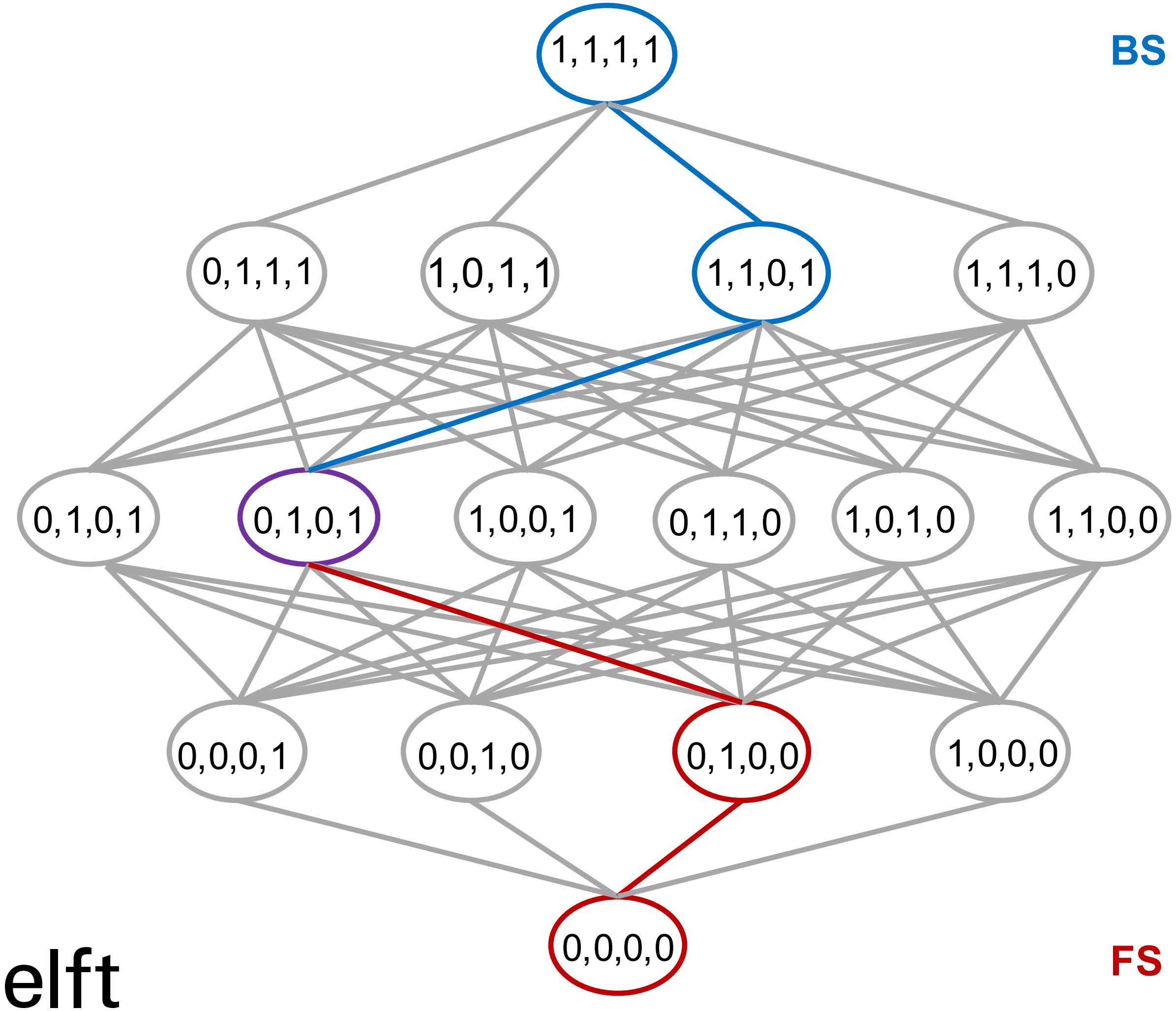
$\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_4$

$\boldsymbol{x}_2$

FS Empty feature set

Bidirectional Selection

Four features in order of x_1, x_2, x_3, x_4,
1 means selected, 0 means not selected,
e.g., (0,0,0,1) means only x_4 is selected.



BS Full set of all originally available features

x_1, x_2, x_4

x_2, x_4

x_2

FS Empty feature set

Plus-L Take-away-R Selection

- **Also based on the ideas of FS and BS**. It has two forms.
- If $L > R$, it starts from the **empty set** and

  -- repeatedly add L features

  -- repeatedly remove R features
- If $L < R$, it starts from the **full set** of all available features and

  -- repeatedly remove R features

  -- repeatedly add L features

- There is no way of foreseeing the best values of L and R. :-(

TUDelft

Floating Selection

- FS and BS suffer from the so-called **nesting effect**. That is,
  - For FS, once a feature is chosen, there is **no way** for it to be discarded later on.
  - For BS, once a feature is discarded, there is **no way** for it to be reconsidered again.

- Plus-L Take-away-R Selection doesn't have a flexible backtracking capability.
  - Every round, we **have to** plus L and **have to** take away R.

- **Floating Selection** allows flexible backtracking:
  - The **dimensionality of the subset** during the search can be **"floating" up and down**.

Floating Selection

- There are two floating methods:
  - Floating **forward** selection & Floating **backward** selection

- Floating **forward** selection starts from the **empty set,**
  - after each forward step, it performs backward steps as long as the criterion function increases.
- Floating **backward** selection starts from the **full set,**
  - after each backward step, it performs forward steps as long as the criterion function increases.

Koutroumbas, K., & Theodoridis, S. (2008). Pattern recognition. Elsevier Science & Technology. Chapter 5.7

Dimensionality Reduction by Selection or Extraction

- Overview – **Feature Selection** vs **Feature Extraction**

- Criteria

  - Mahalanobis distance (vs Euclidean distance)

  - Scatter matrices (what are $\boldsymbol{S}_W$, $\boldsymbol{S}_B$, $\boldsymbol{S}_T$?)

- Approaches

  - Sequential **feature selection** (individual, forward, backward, etc.)

  - Principal Component Analysis & Recall LDA ($\in$ linear **feature extraction**)

TU Delft

Interesting facts about PCA

- PCA is widely recognized as the most classical method for dimensionality reduction, having been invented in 1901.

- However, it **doesn't automatically reduce the dimensionality!**

- Rather, it **transforms the data into a new coordinate system** where **the choice to retain fewer** principal components effectively reduces dimensionality.
  - **Retain the variance as much as possible**
    - **i.e., Minimize the reconstruction error**

PCA: offers different view of your data

- Data:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_p \end{bmatrix}$$

mean-centered data (the mean of each feature is 0);
p is number of features

- (Variance-) Covariance matrix:

$$\underset{(p \times p)}{\boldsymbol{\Sigma}} = \boldsymbol{X}\boldsymbol{X}^T = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_p^2 \end{bmatrix}$$

PCA: offers different view of your data

- **Eigen-decomposition** of the covariance matrix:

$$\boldsymbol{\Sigma v} = \boldsymbol{v}\lambda,\ \|\boldsymbol{v}\|^2 = 1 \longrightarrow \underset{(p\times 1)}{\boldsymbol{v}_i} = \begin{bmatrix} v_{1i} \\ v_{2i} \\ \vdots \\ v_{pi} \end{bmatrix}, \lambda_i, i = 1,2,\dots p$$

- Transform the data to a new space, in which the coordinate system is defined by the principal components.

$$\underset{(p\times p)}{\boldsymbol{V}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \dots \boldsymbol{v}_k \dots \boldsymbol{v}_p \end{bmatrix}$$

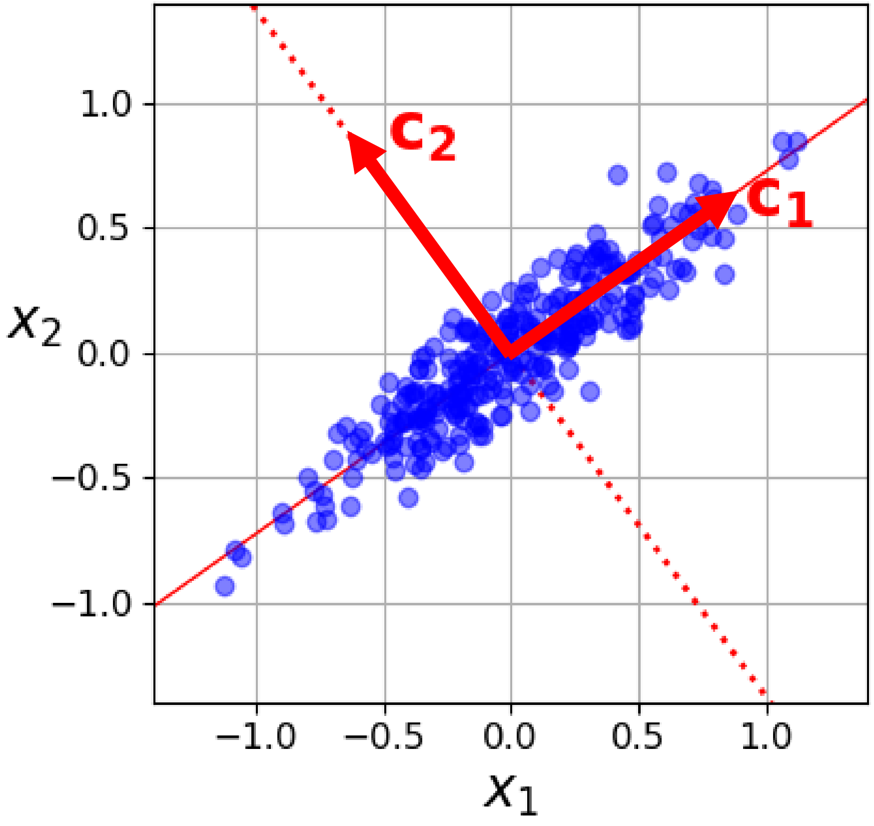
Each column of $\boldsymbol{V}$ is a principal component
ORDERED by the value of λ,
λ_1 is the largest eigenvalue

$$\boldsymbol{T} = \boldsymbol{V}^T\boldsymbol{X} = \begin{bmatrix} \boldsymbol{v}_1^T\boldsymbol{X} \\ \boldsymbol{v}_2^T\boldsymbol{X} \\ \vdots \\ \boldsymbol{v}_k^T\boldsymbol{X} \\ \vdots \\ \boldsymbol{v}_p^T\boldsymbol{X} \end{bmatrix}$$

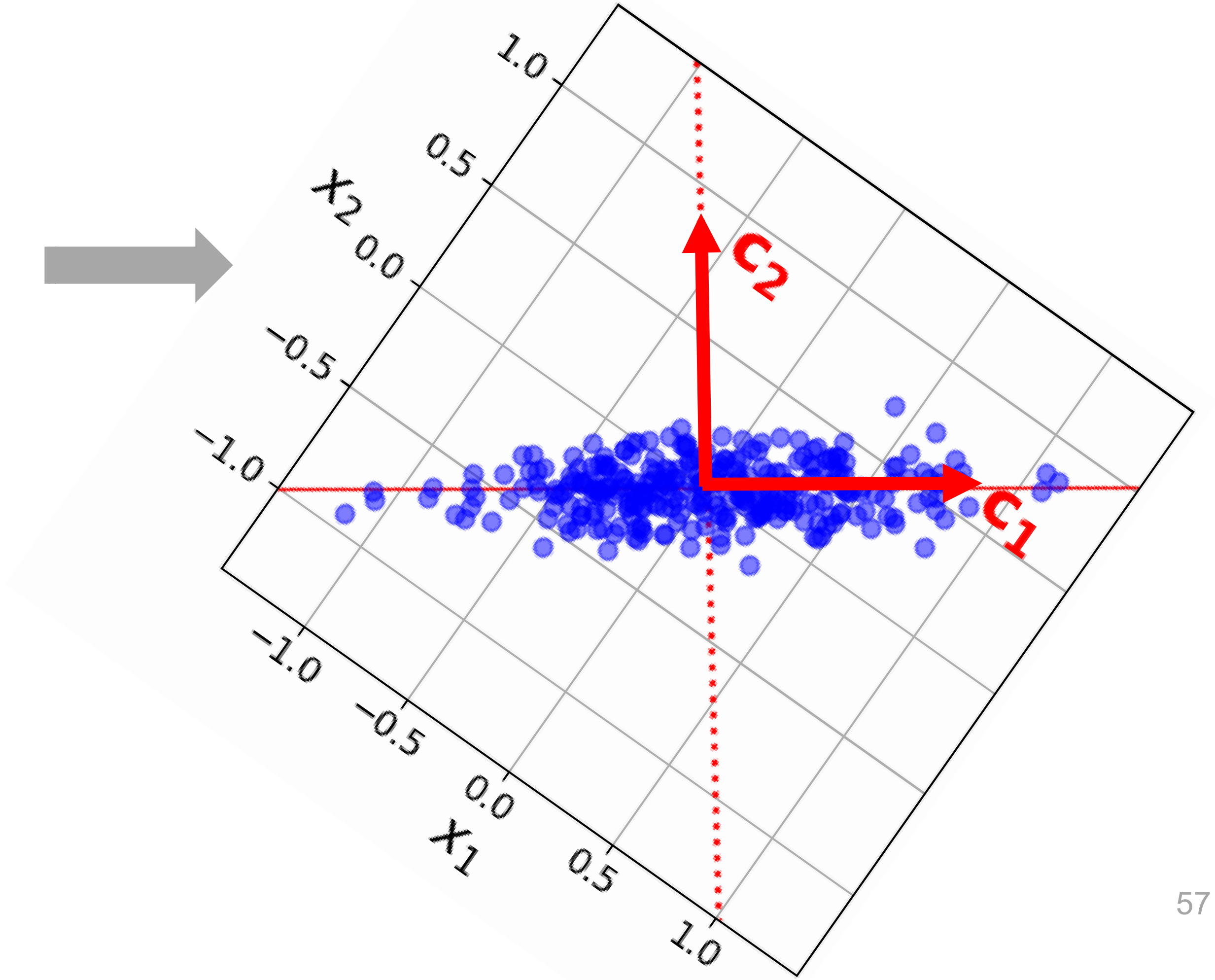
Quiz:
What is the dimensionality of $\boldsymbol{T}$?
Same with $\boldsymbol{X}$?

PCA: offers different view of your data

Original Space (2D)



PCA Space (2D)



$n = 300$

PCA: choose to reduce dimensionality

- Again, PCA doesn't automatically reduce the dimensionality.

$$\boldsymbol{T} = \boldsymbol{V}^T \boldsymbol{X}$$

- **Choose to retain** the first k principal components because e.g., 95% variance is captured

$$\boldsymbol{V} = [\underbrace{\boldsymbol{v}_1 \ \boldsymbol{v}_2 \dots \boldsymbol{v}_k}_{\text{keep}} \mid \underbrace{\dots \boldsymbol{v}_p}_{\text{drop}}]$$

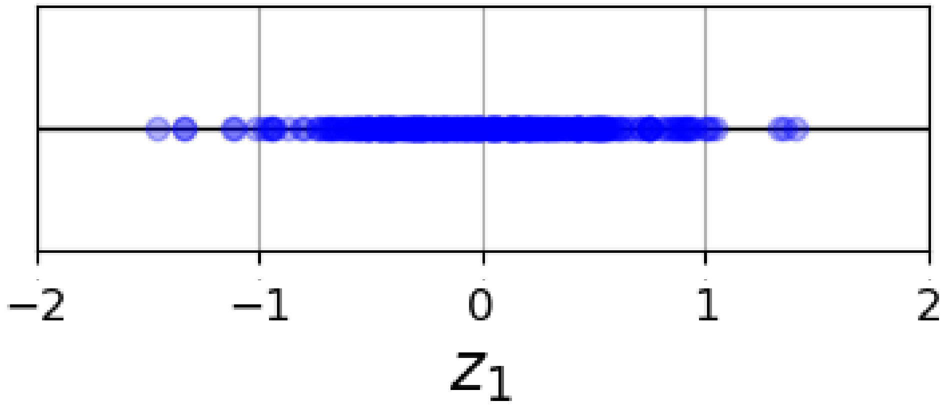
What is the dimensionality of $\boldsymbol{T}_k$?

$$\boldsymbol{T}_k = \boldsymbol{V}_k^T \boldsymbol{X} \qquad \underset{(k \times p)}{\boldsymbol{V}_k^T} = \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_k^T \end{bmatrix}$$

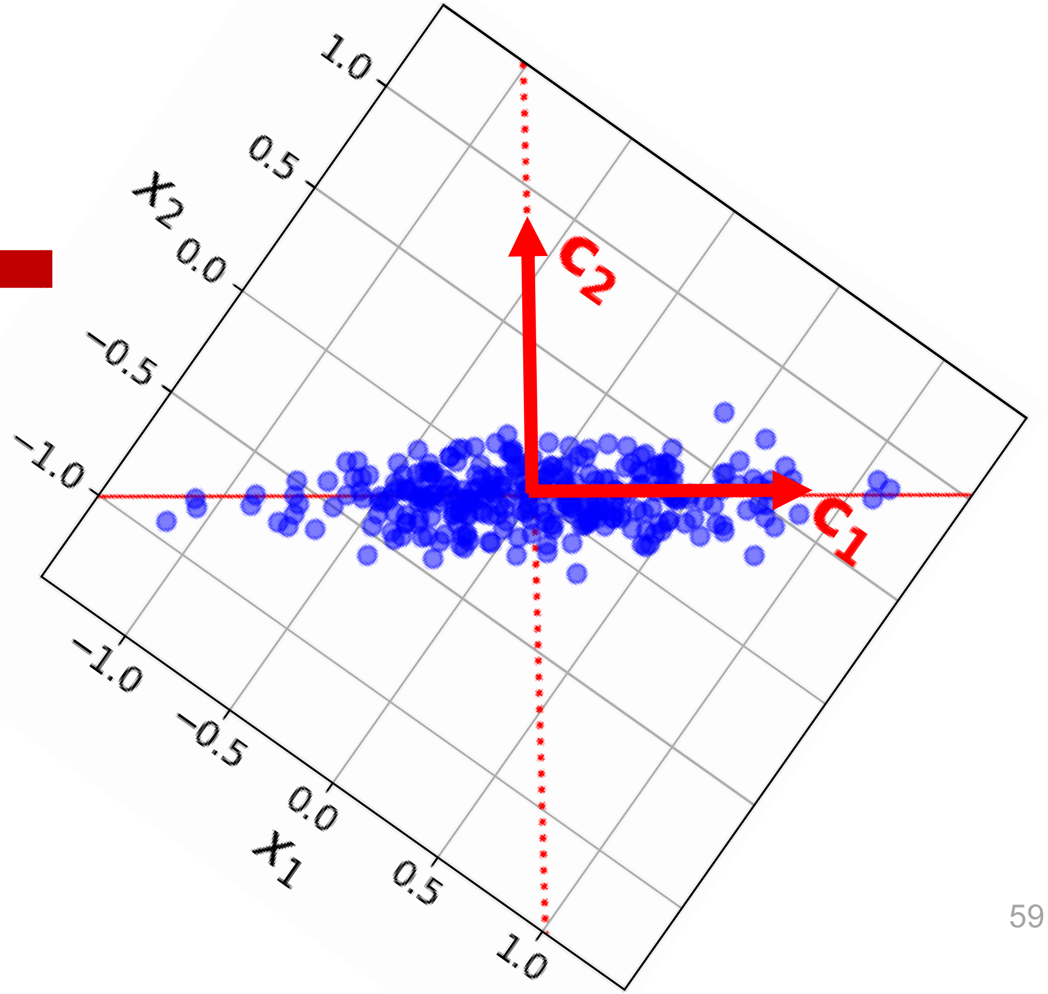
TUDelft

PCA: choose to reduce dimensionality

Project data to C1
Choose to drop C2 (1D)



PCA Space (2D)



Quiz

$$\underset{(p\ \times\ p)}{\boldsymbol{V}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \dots \boldsymbol{v}_k \dots \boldsymbol{v}_p \end{bmatrix}$$

$$\underset{(k \leq p)}{\boldsymbol{T}_k} = \boldsymbol{V}_k^T \boldsymbol{X} = \begin{bmatrix} \boldsymbol{v}_1^T \boldsymbol{X} \\ \boldsymbol{v}_2^T \boldsymbol{X} \\ \vdots \\ \boldsymbol{v}_k^T \boldsymbol{X} \end{bmatrix}$$

- When $k = p$, $\boldsymbol{T}_k$ contain **exactly the same amount** of information as the original data $\boldsymbol{X}$.
  True or False?
- What does $\boldsymbol{v}_1^T \boldsymbol{X}$ in $\boldsymbol{T}_k$ represent?
- What does $\boldsymbol{v}_1^T \boldsymbol{\Sigma} \boldsymbol{v}_1$ represent?

Quiz

$$\underset{(p \times p)}{\boldsymbol{V}} = \left[\boldsymbol{v}_1 \;\; \boldsymbol{v}_2 \ldots \boldsymbol{v}_k \ldots \boldsymbol{v}_p\right] \qquad \underset{(k \le p)}{\boldsymbol{T}_k} = \boldsymbol{V}_k^T \boldsymbol{X} = \begin{bmatrix} \boldsymbol{v}_1^T \boldsymbol{X} \\ \boldsymbol{v}_2^T \boldsymbol{X} \\ \vdots \\ \boldsymbol{v}_k^T \boldsymbol{X} \end{bmatrix}$$

- When $k = p$, $\boldsymbol{T}_k$ contain **exactly the same amount** of information as the original data $\boldsymbol{X}$.

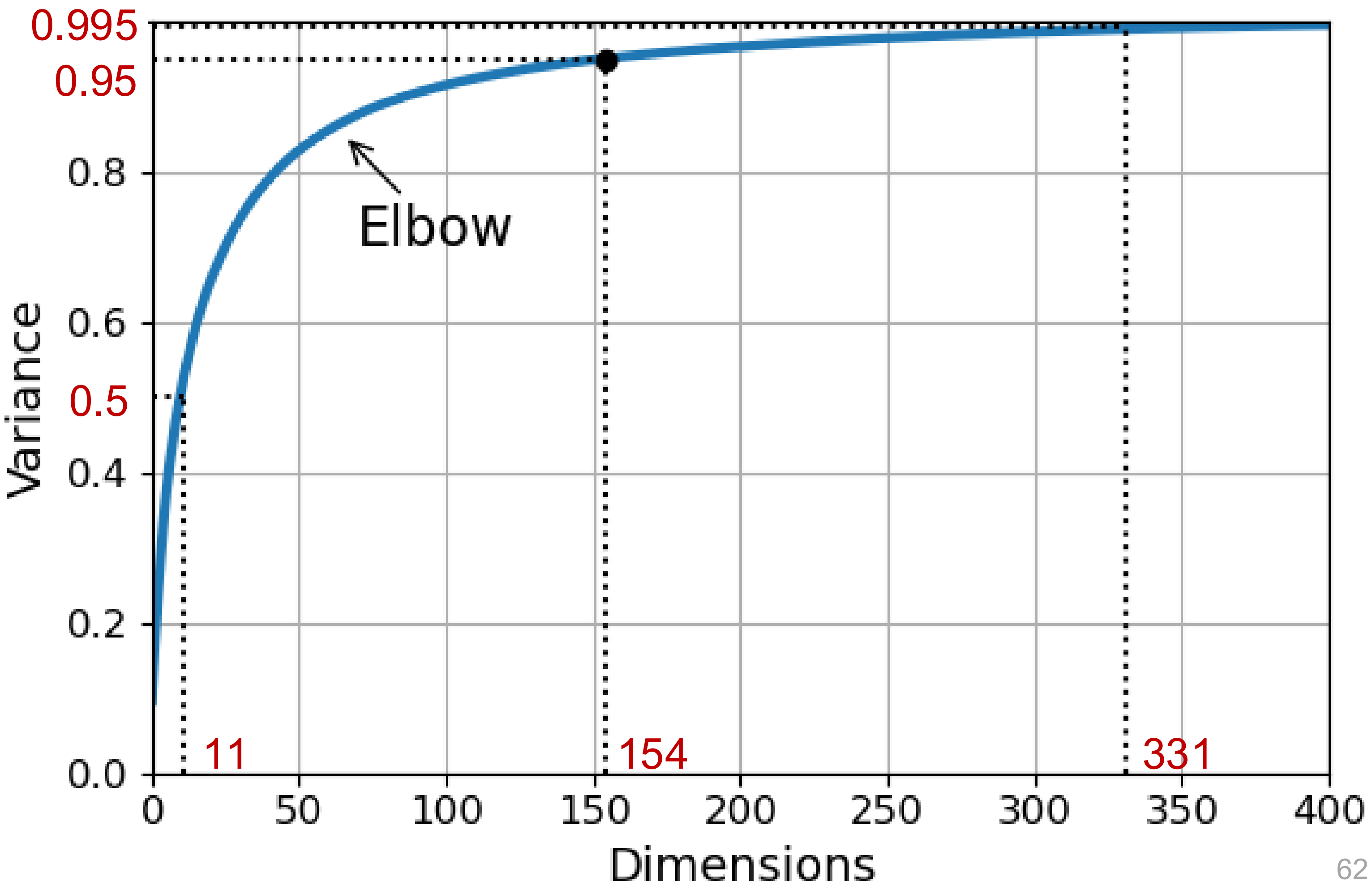
  True!!!
- What does $\boldsymbol{v}_1^T \boldsymbol{X}$ in $\boldsymbol{T}_k$ represent?

  Projection onto the first principal component!
- What does $\boldsymbol{v}_1^T \boldsymbol{\Sigma} \boldsymbol{v}_1$ represent?

  Variance!

PCA on MNIST data

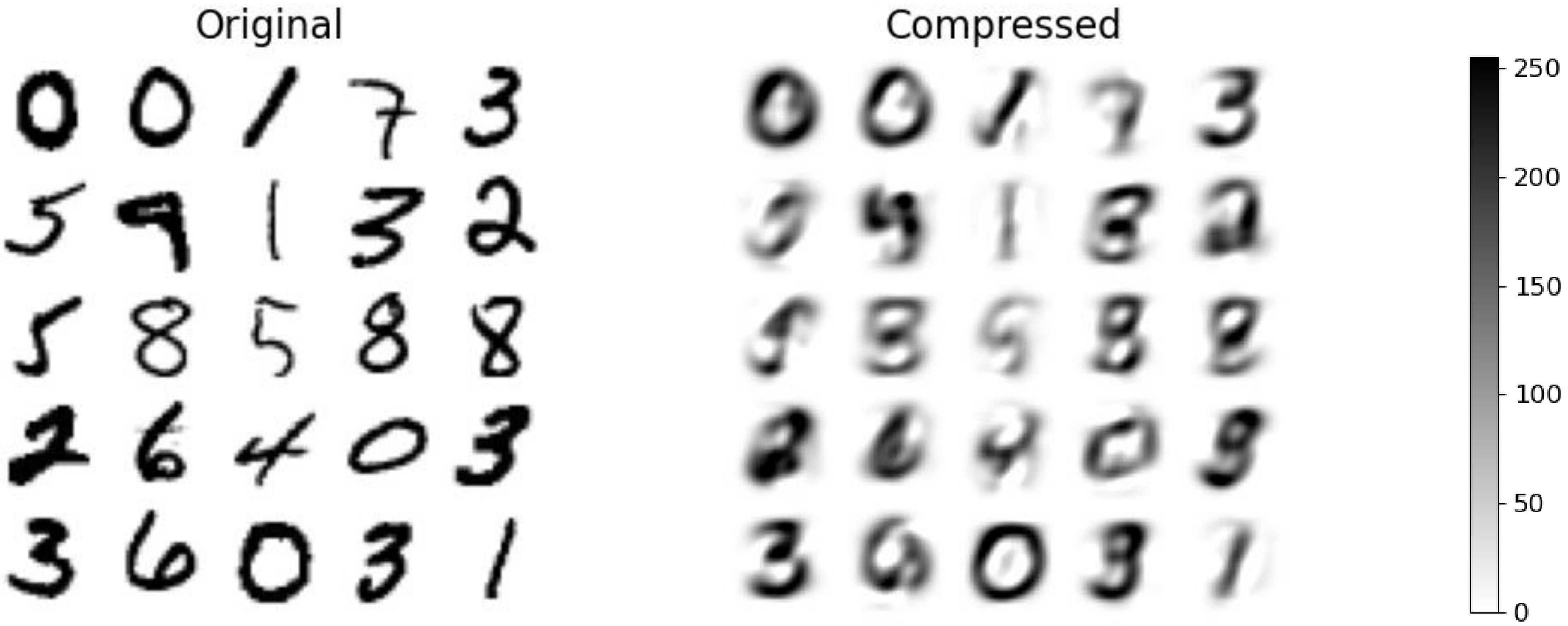
- PCA reconstructions
- Original space (784 D)



TU Delft

PCA on MNIST data

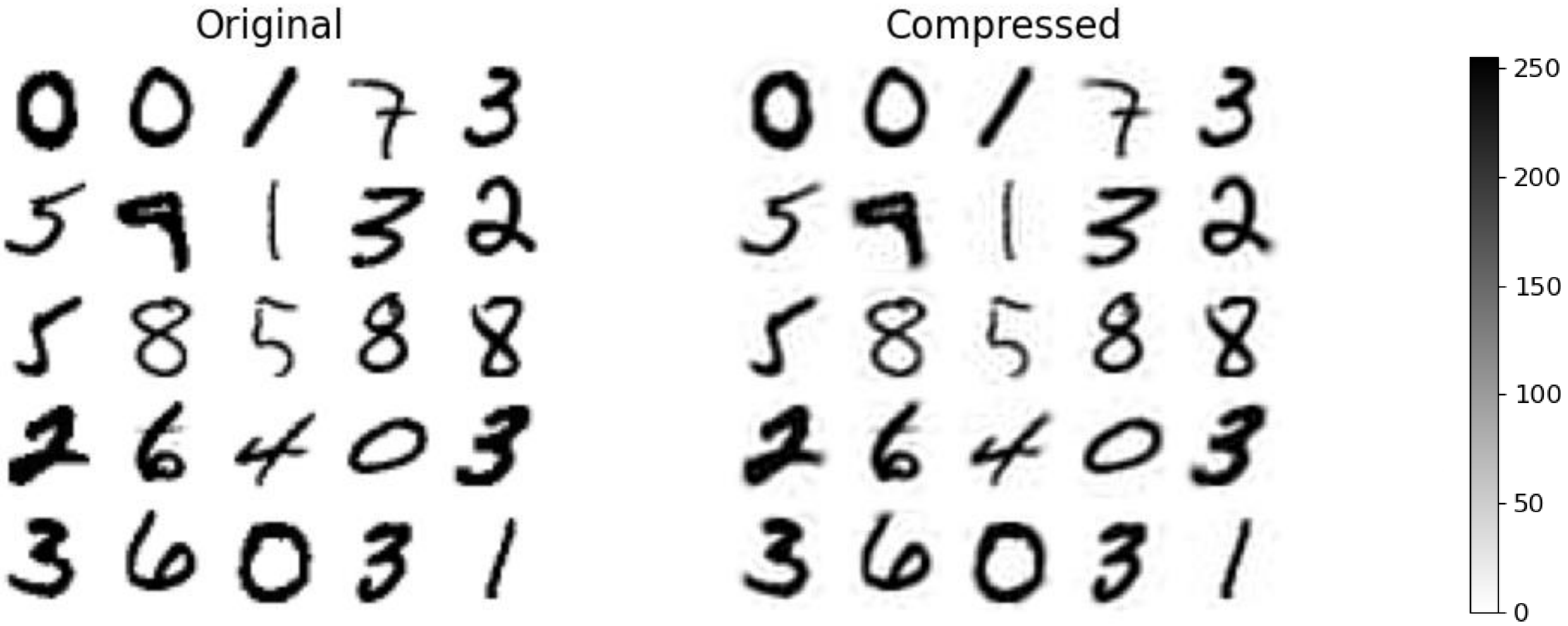
- 50% Variance: Dim = 11



- The more PCs we retain, the smaller the reconstruction error becomes.

PCA on MNIST data

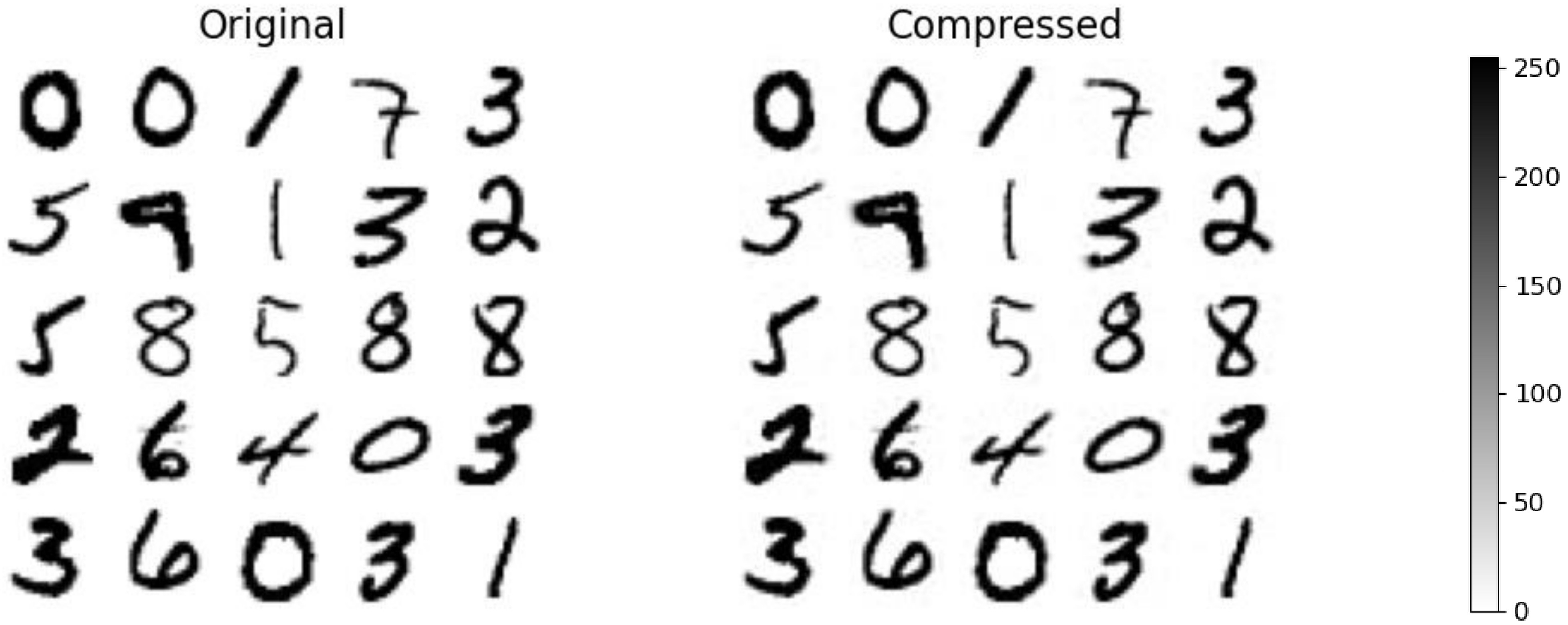
- 95% Variance: Dim = 154



- The more PCs we retain, the smaller the reconstruction error becomes.

PCA on MNIST data

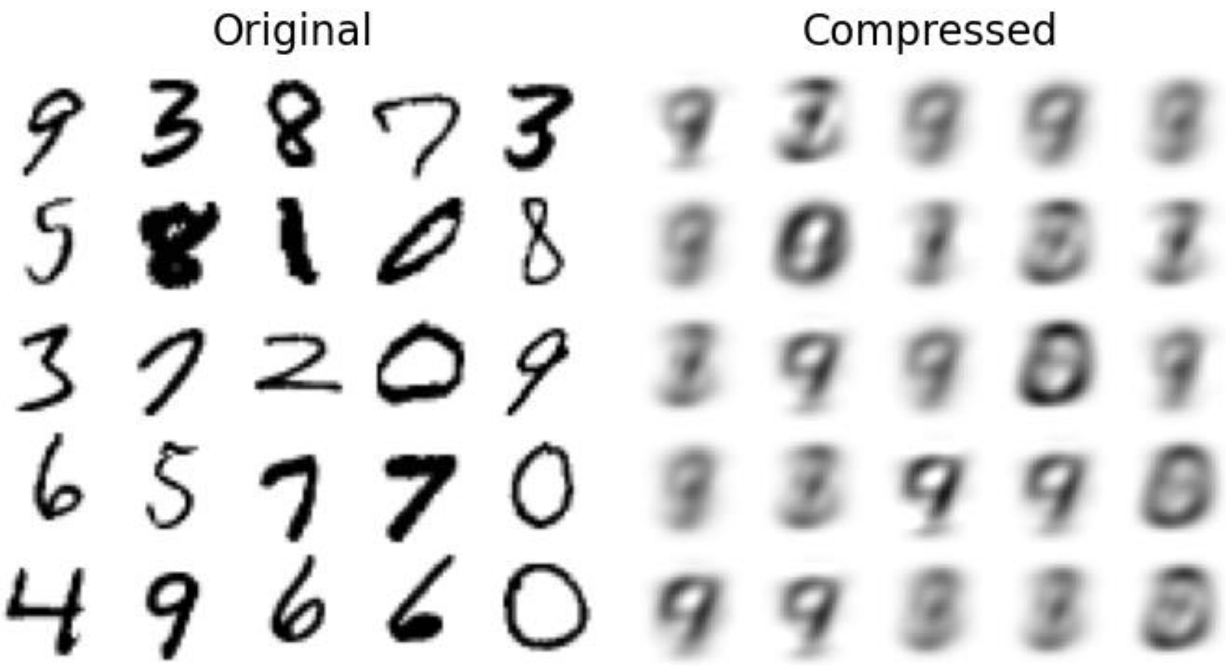
- 99.5% Variance: Dim = 331



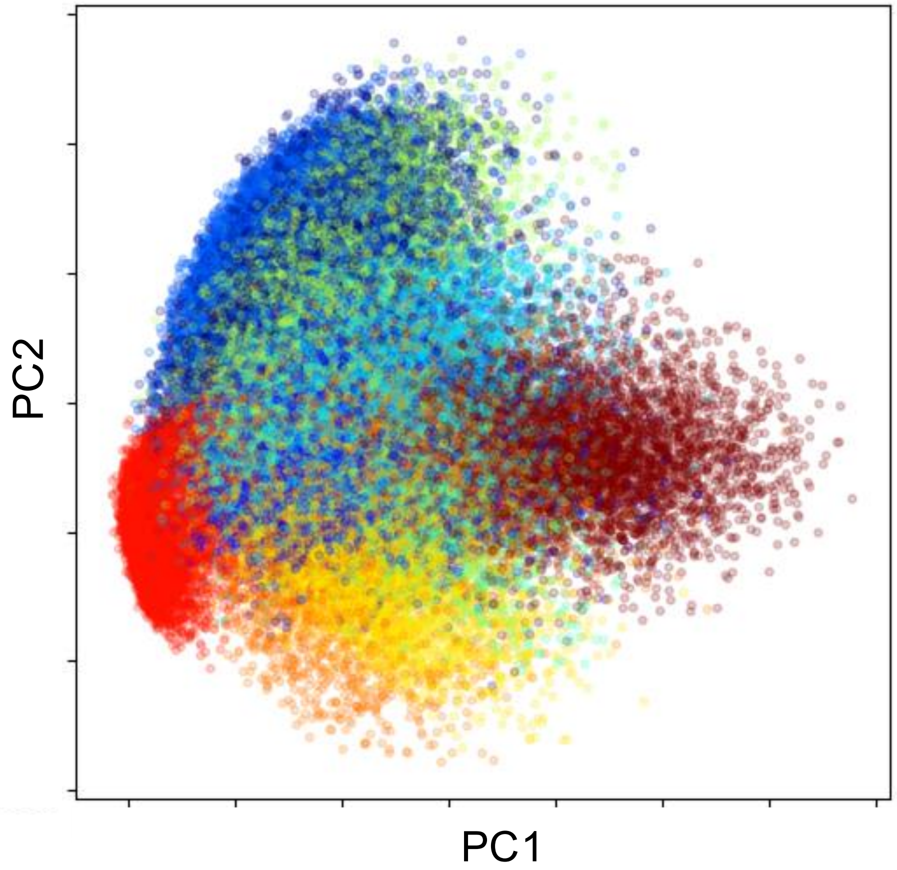
- The more PCs we retain, the smaller the reconstruction error becomes.

PCA on MNIST data

784 PCs **2 PCs**



PCA SPACE



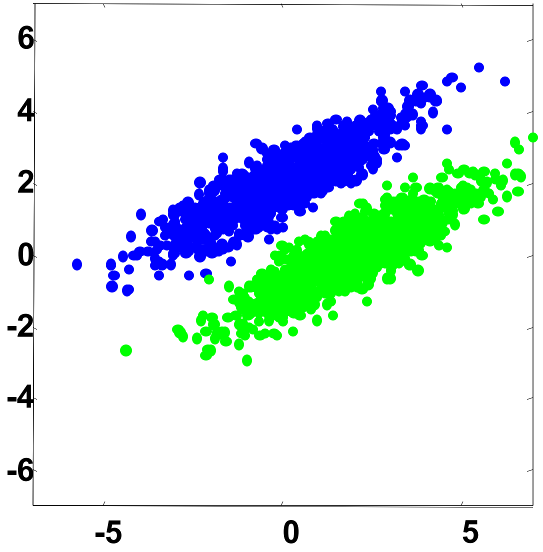
Colours indicate the class of the object

Two classical linear feature extractors

- LDA (or Fisher mapping):

  Supervised

  -- *Find projection vector* $\boldsymbol{a}$ *that captures the greatest* ***separability between the classes****, i.e., choose* $\boldsymbol{a}$ *to maximize Fisher criterion:*



$$J_F(\boldsymbol{a}) = \frac{\boldsymbol{a}^T \boldsymbol{S}_B \boldsymbol{a}}{\boldsymbol{a}^T \boldsymbol{S}_W \boldsymbol{a}}$$

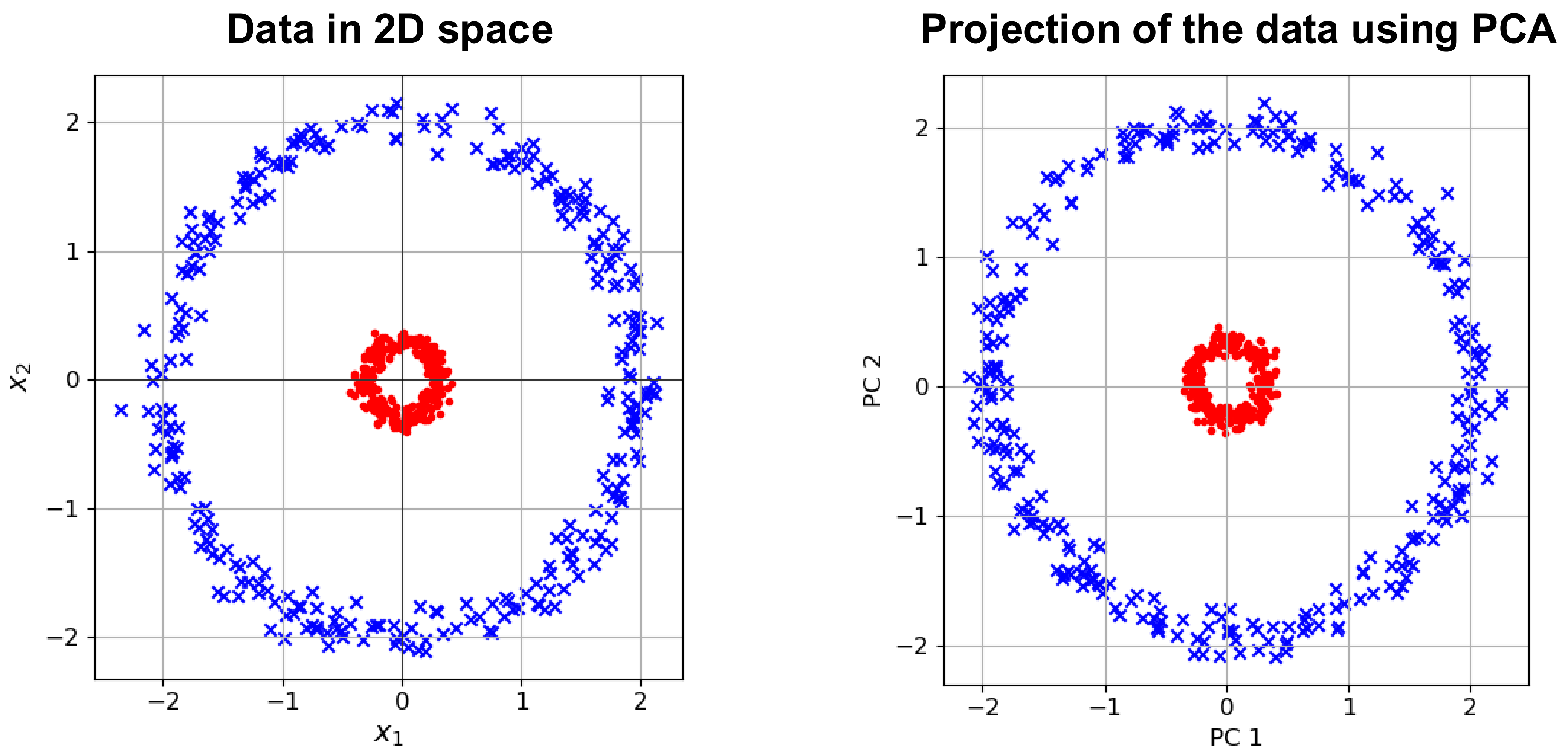
- PCA:

  Unsupervised

  -- *Capture the greatest* ***variance in the total data***

Kernel PCA – Non-linear Dimensionality Reduction.

- Linear subspaces may be inefficient for some cases.
- The data are not linearly separable in the original dimension.

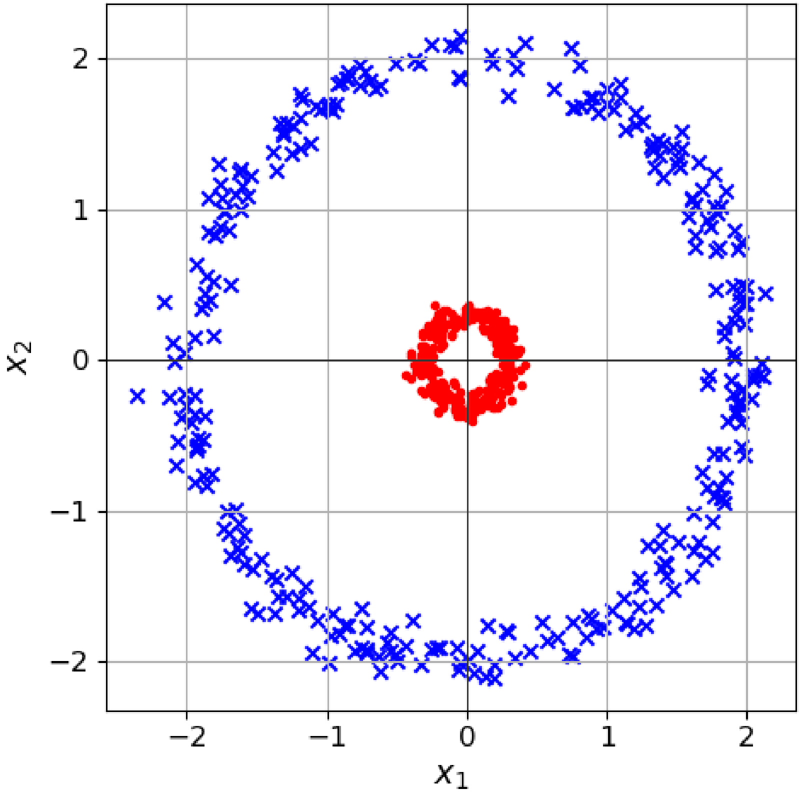


TUDelft

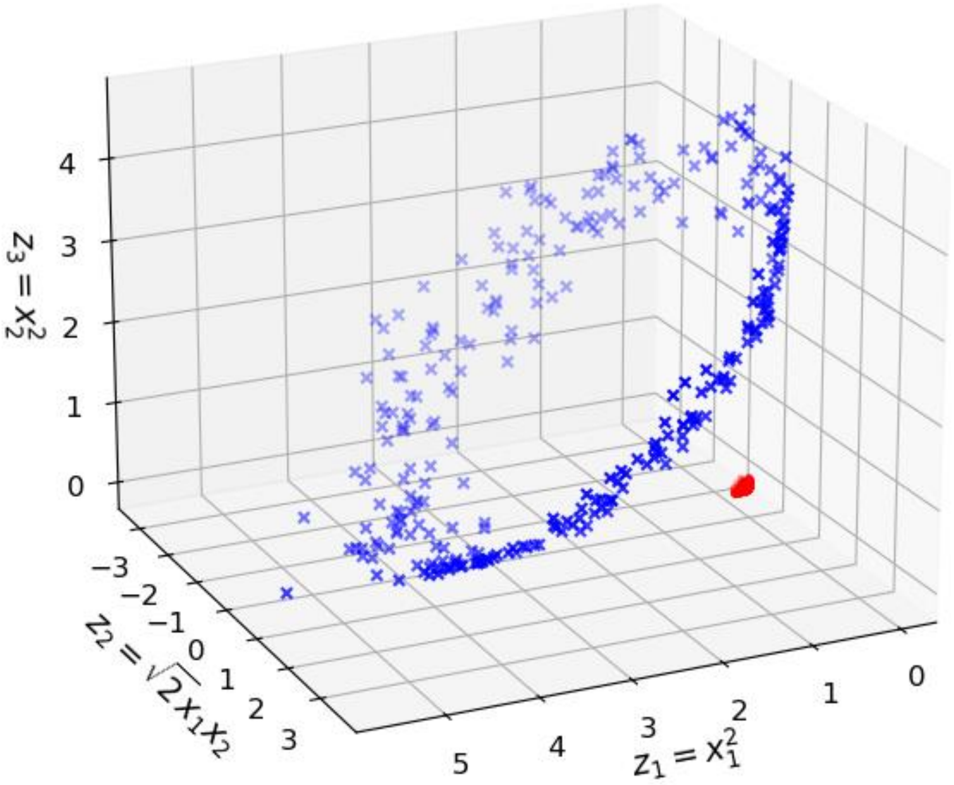
Kernel PCA – What it does

- Use a kernel function to project data into a higher-dim. space where they are linearly separable.
- $\phi = R^2 \rightarrow R^3$ $\qquad (x_1, x_2) \longrightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

Data in 2D space

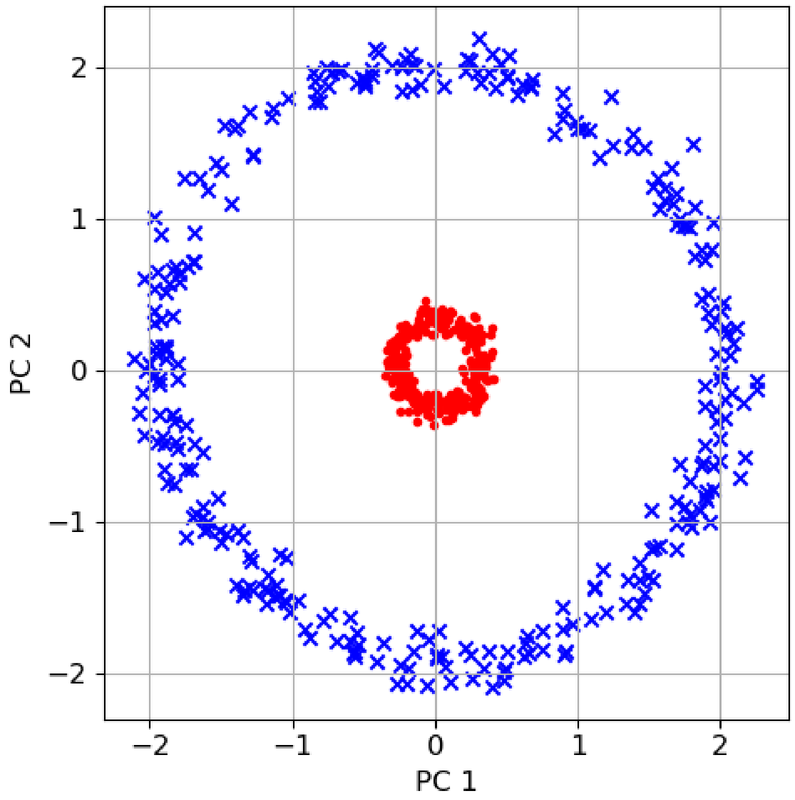


Data mapped to 3D space

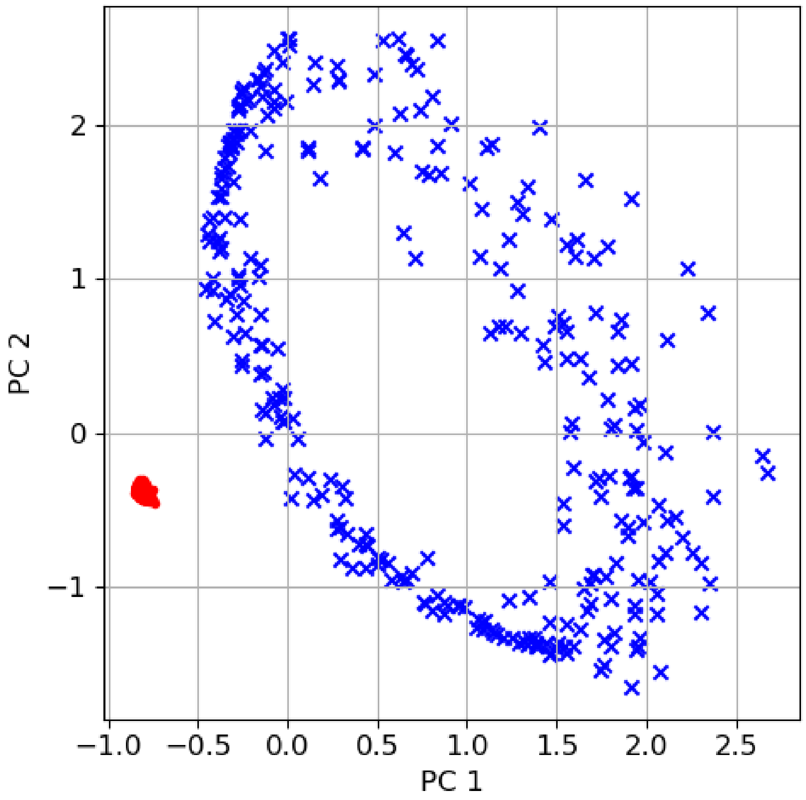


Kernel PCA – What it does

Apply PCA



Apply Kernel PCA and drop PC3



TUDelft

Practice

Given mean-centered data in 3D for which the covariance matrix is given by

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

Also given is a data transformation matrix

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix},$$

by which we can linearly transform every data vector x (taken as a column vector) to a new 3D column vector z through $z = Rx$.

We note that R is actually a rotation matrix that rotates in the second and third coordinate.

Also note that for its inverse, we have

$$R^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}.$$

Q1: What is the first principal component of the original data for which we have the covariance matrix $\boldsymbol{\Sigma}$?

Q2: Assume we transform all the data by the transformation matrix R, what does the covariance of the transformed data become?

Q3: What is the first principal component for the transformed data?

TU Delft

Practice

Given mean-centered data in 3D for which the covariance matrix is given by

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

Also given is a data transformation matrix

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix},$$

by which we can linearly transform every data vector x (taken as a column vector) to a new 3D column vector z through $z = Rx$.

Q2: $\boldsymbol{R\Sigma R}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{7}{2} & \frac{-\sqrt{3}}{2} \\ 0 & \frac{-\sqrt{3}}{2} & \frac{5}{2} \end{bmatrix}$

We note that R is actually a rotation matrix that rotates in the second and third coordinate.

Also note that for its inverse, we have

$$R^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}.$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Q1: What is the first principal component of the original data for which we have the covariance matrix $\boldsymbol{\Sigma}$?

Q2: Assume we transform all the data by the transformation matrix R, what does the covariance of the transformed data become?

Q3: What is the first principal component for the transformed data?

$\boldsymbol{Rv}_1$

TU Delft