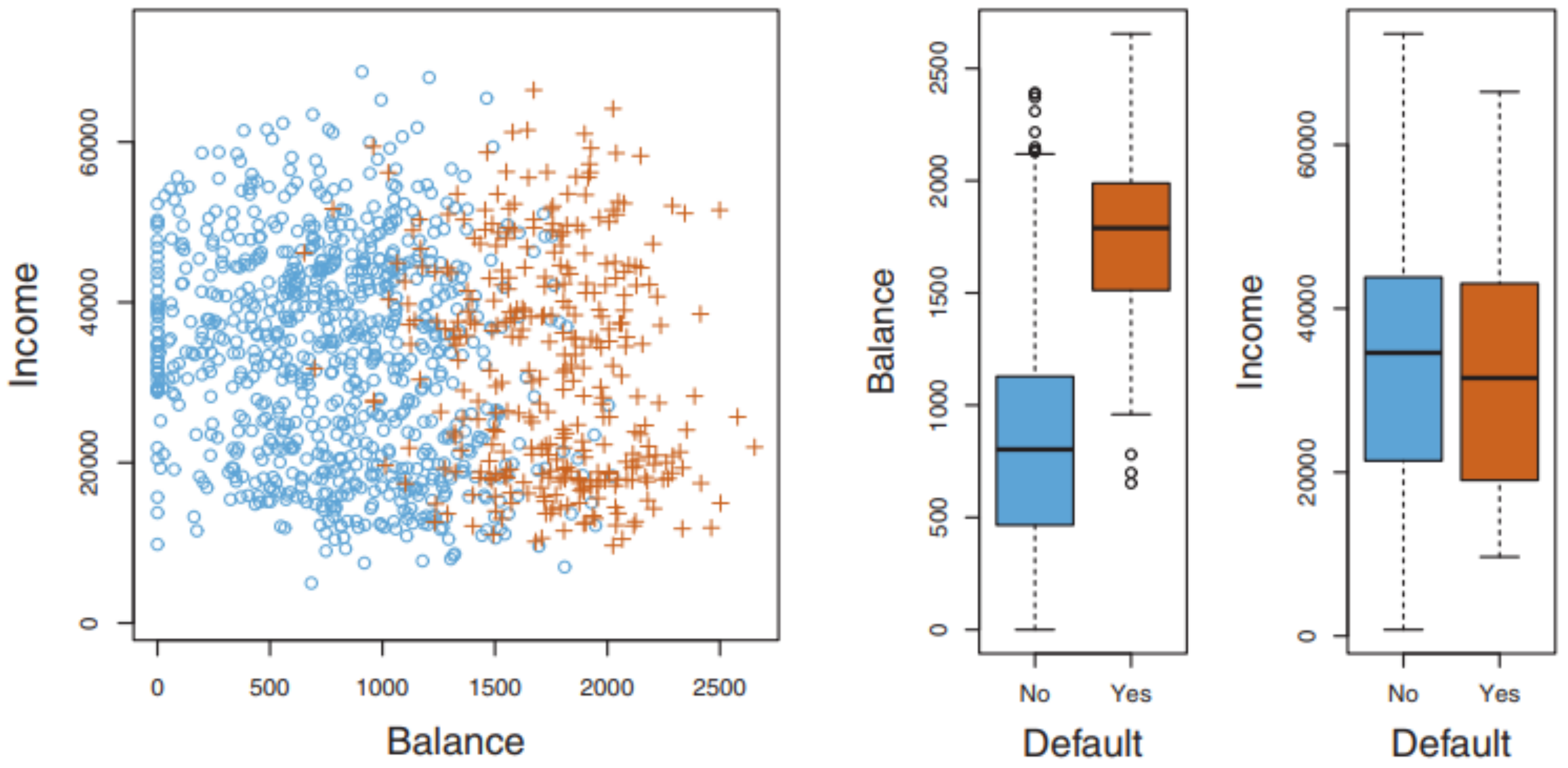
Dimensionality Reduction
Feature Selection & Extraction

Jing Sun

Is “Income” an informative feature?



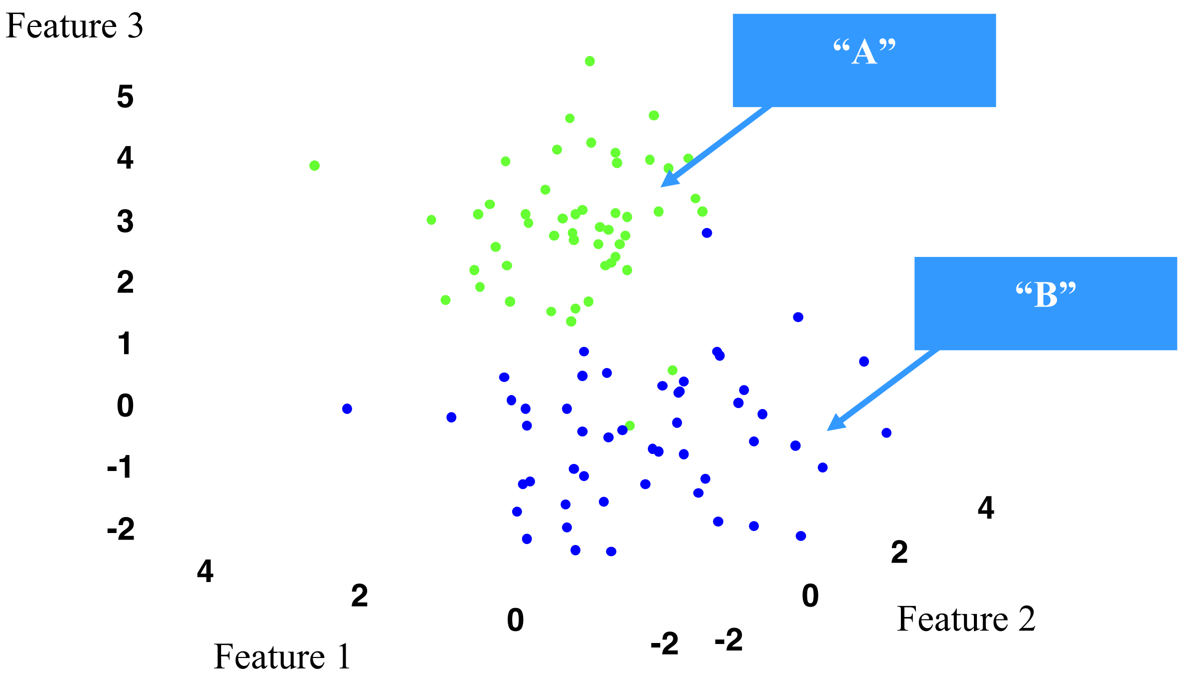
Feature Space

- A p-dimensional space,

in which each dimension is a feature

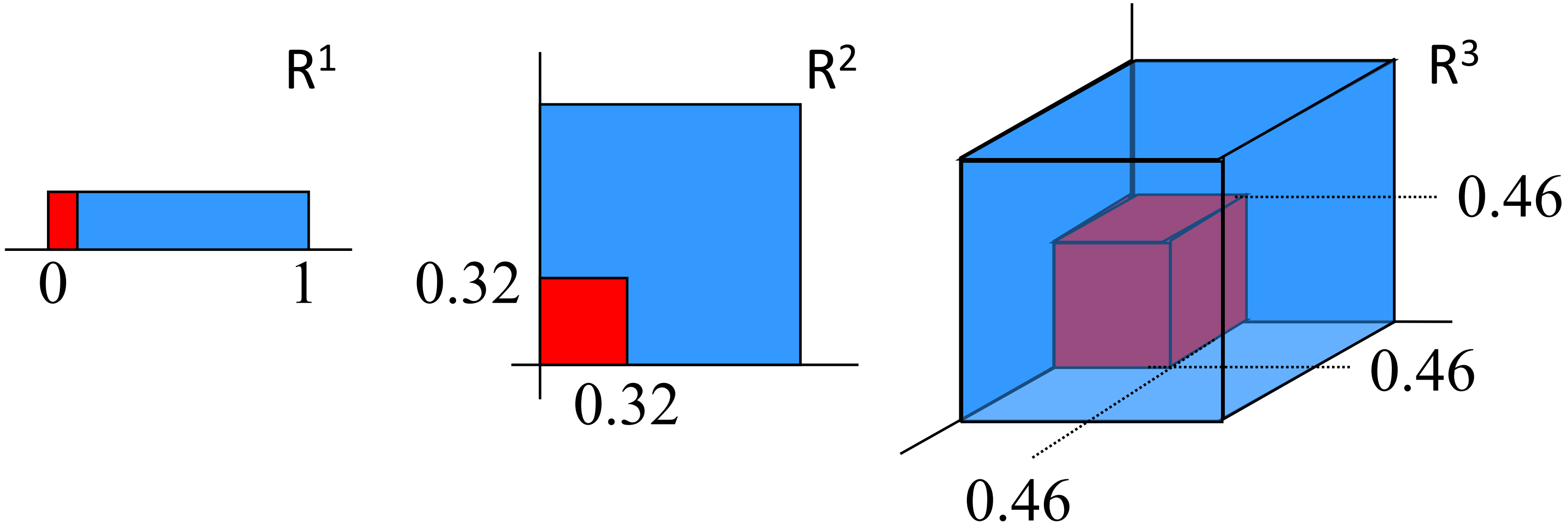
containing n [labeled] samples [objects]

- 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…

TUDelft

High-Dimensional Spaces

- Example: Boundary points ?

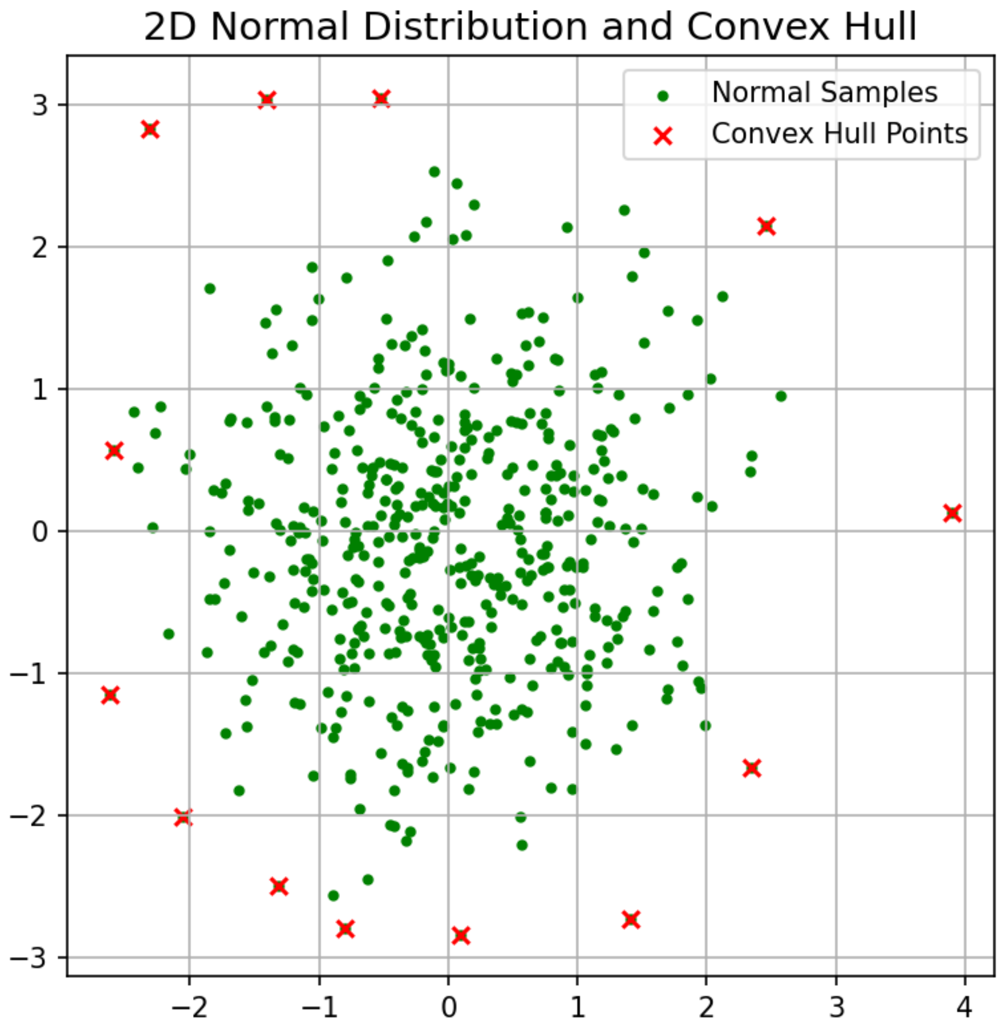
500 samples from normal distribution

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: Boundary points ?

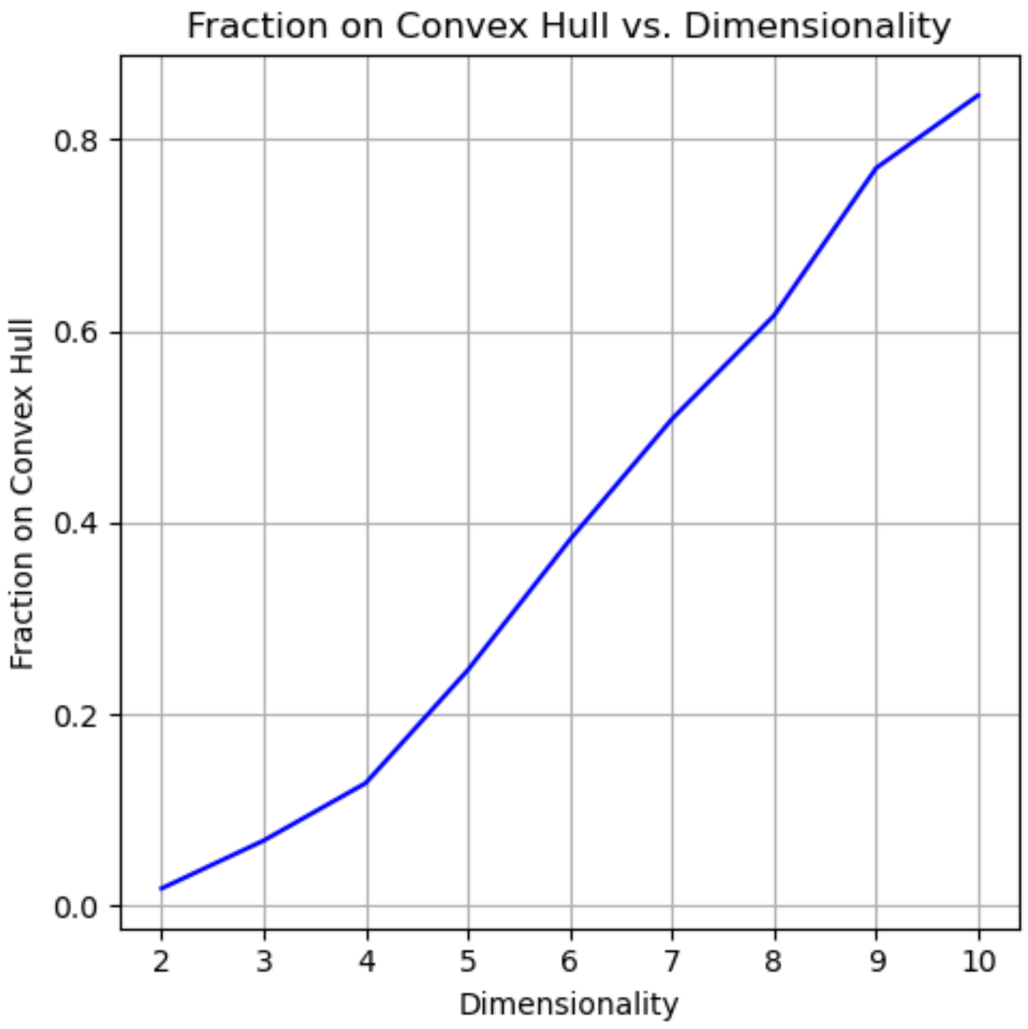
500 samples from normal distribution

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

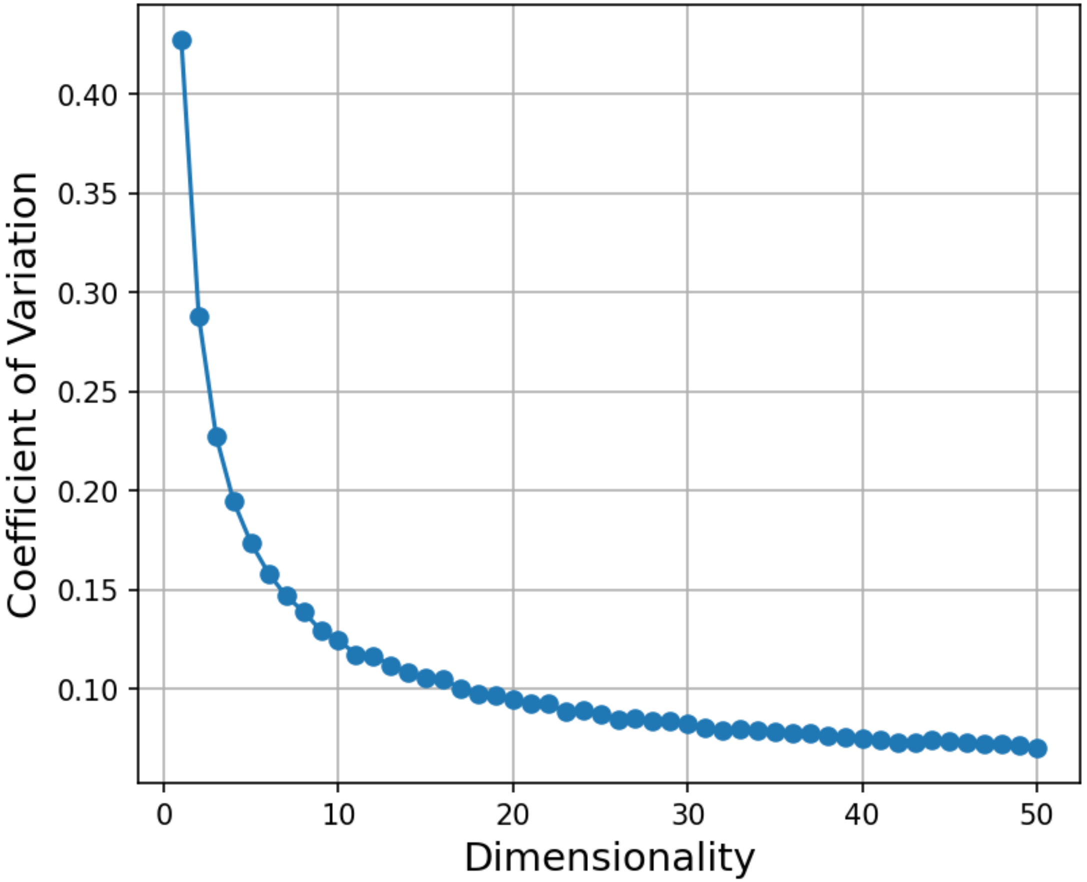
- Example: Points tend to have equal distances

200 samples from normal distribution

$N(2000, 8000)$

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

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



TU Delft

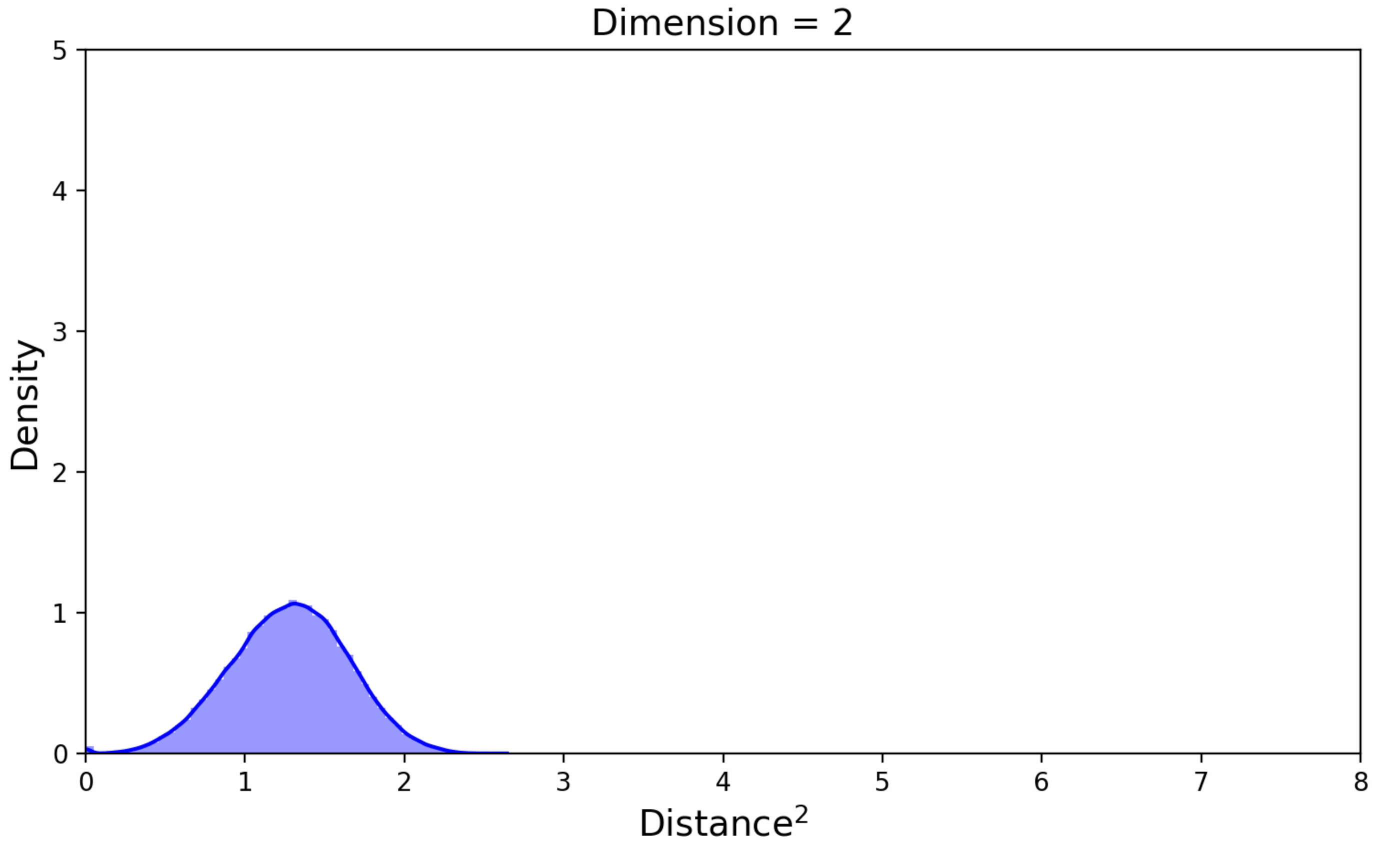
High-Dimensional Spaces

- 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



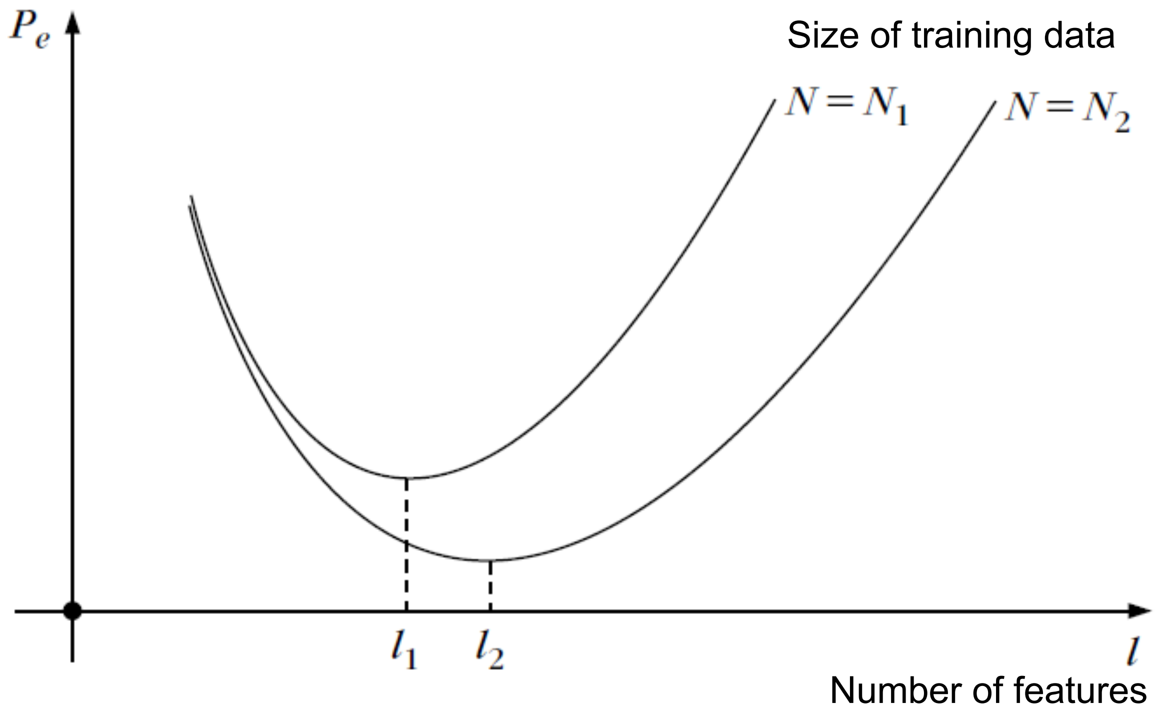
TU Delft

Dimensionality Reduction

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

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

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



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 by Selection or Extraction

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

- Criteria

  - Mahalanobis distance (vs Euclidean distance)

  - Scatter matrices (what are S_W, S_B, 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 S_W, S_B, S_T?)

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

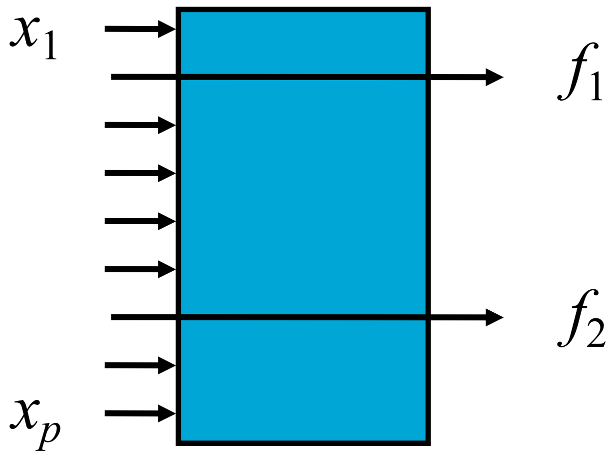
Feature Selection vs Extraction

- Feature selection :

  SELECT d **out of** p measurements

  Only a subset of the original features are selected.

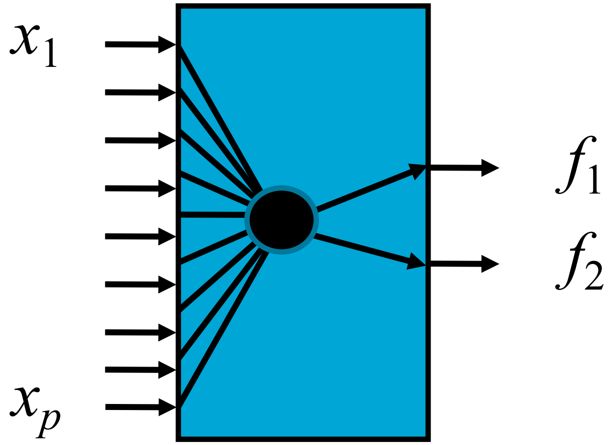
  There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ subsets.



- 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**

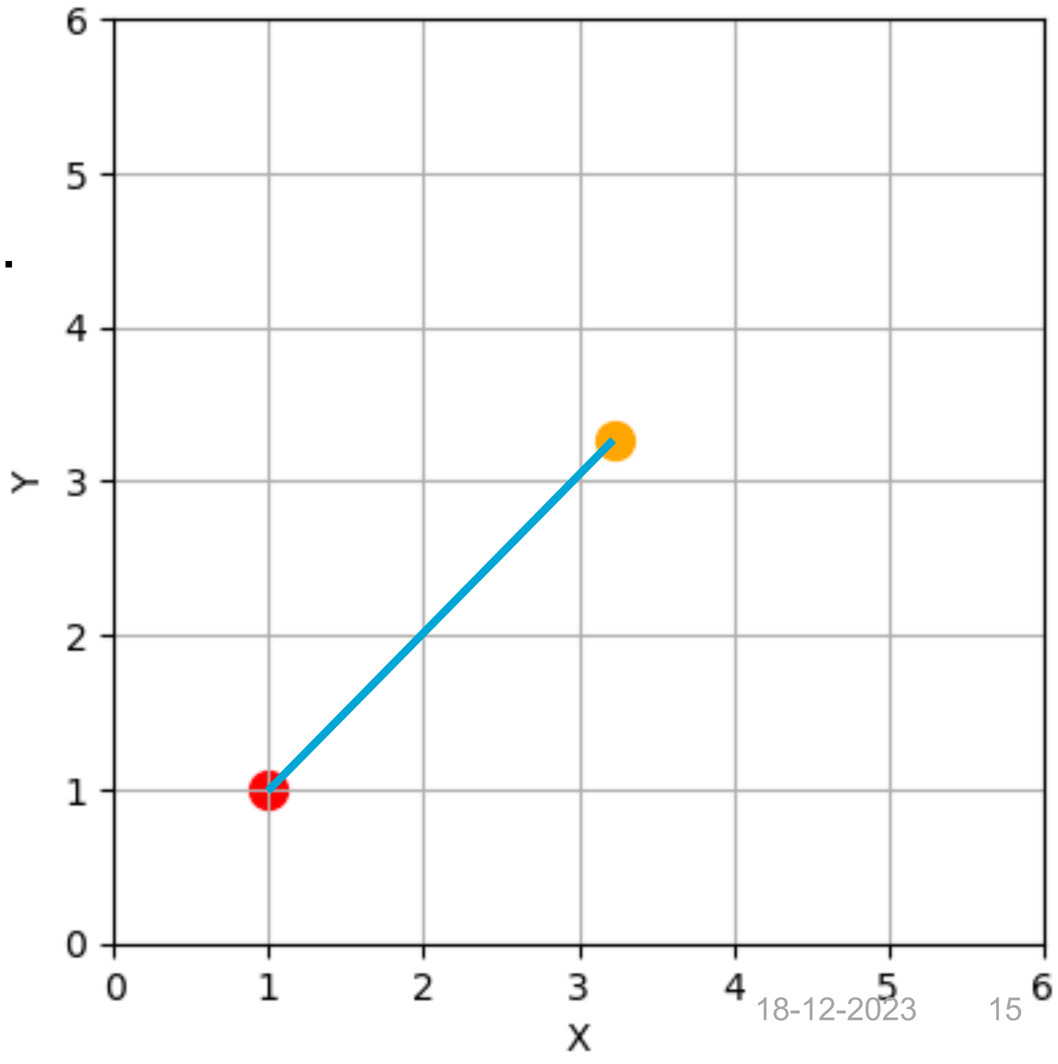
- Criteria
  - Mahalanobis distance (vs Euclidean distance)
  - Scatter matrices (what are S_W, S_B, S_T?)

- 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.

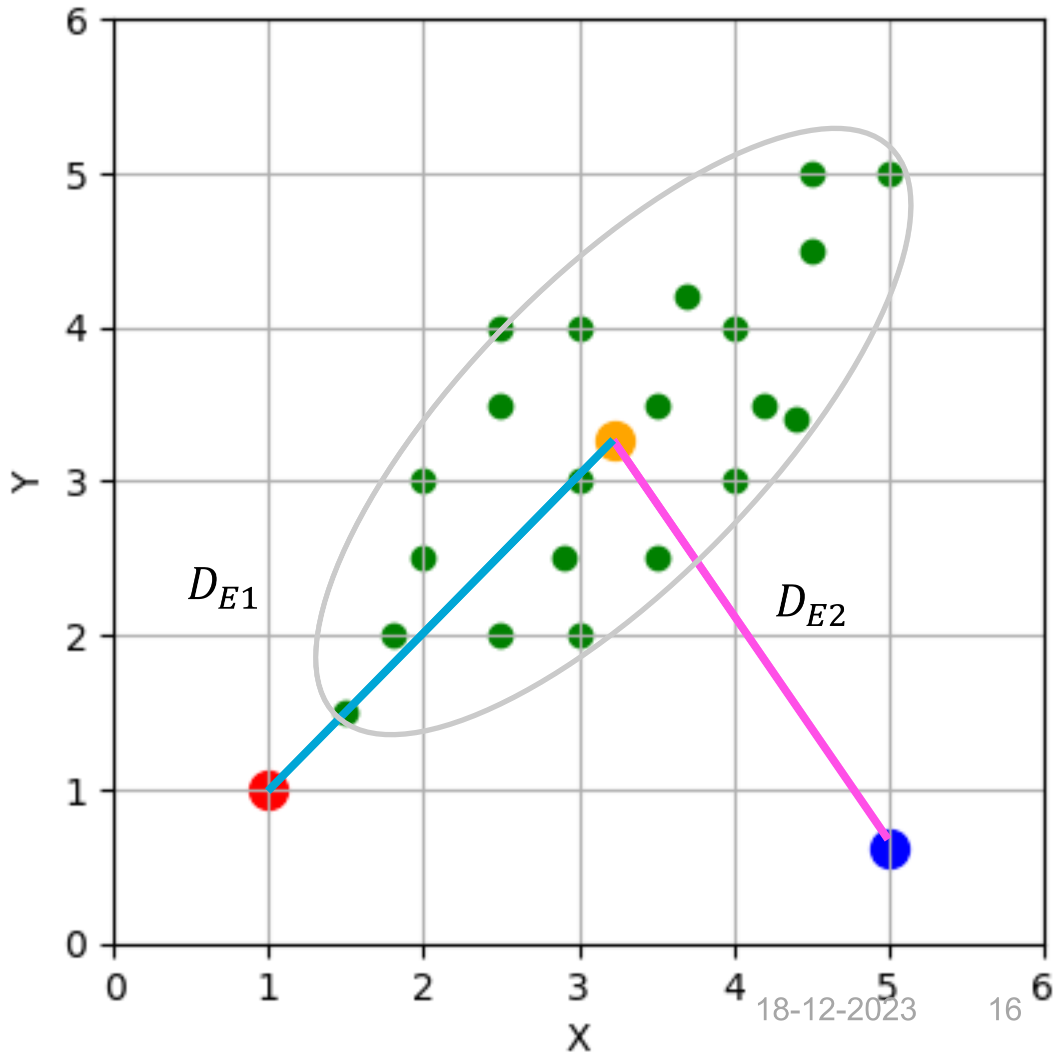
$$D_E = (x_{red} - x_{yellow})^2 + (y_{red} - y_{yellow})^2$$



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

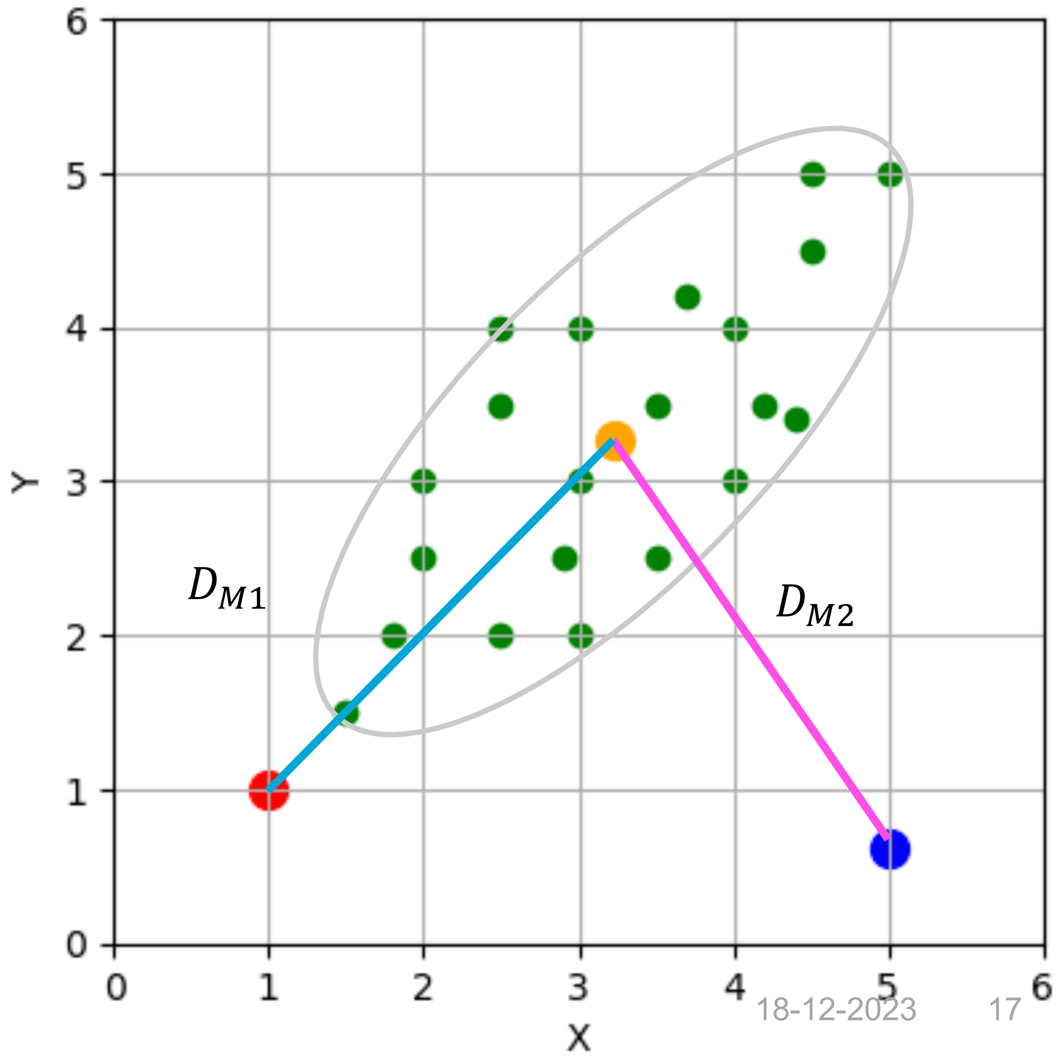
- Takes the **variance** into account.

- It is a distance measure
  between a point and a **distribution**.

- For red and blue points,

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

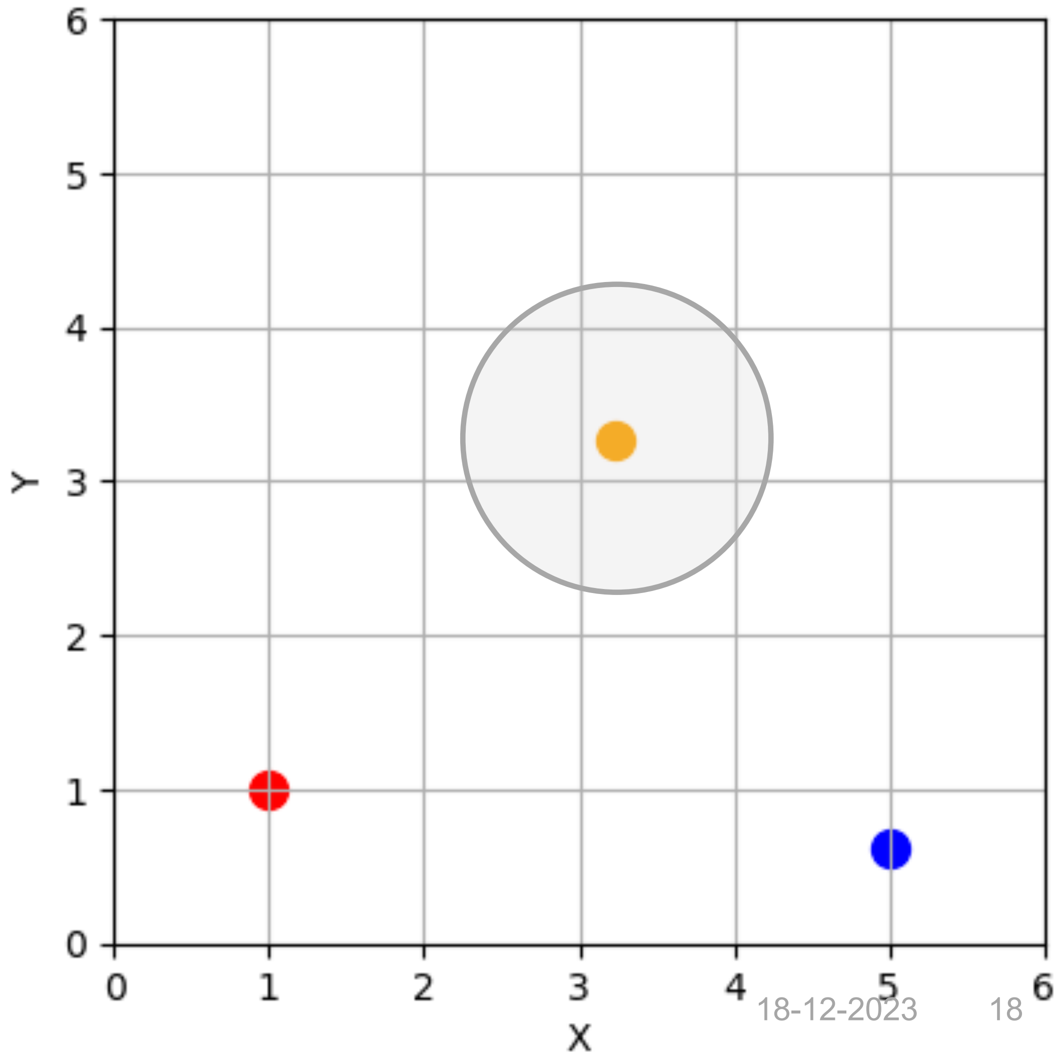
- You will see $D_{M2} > D_{M1}$



Mahalanobis distance

- Think about:
- What if Σ is an identity matrix?

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



Mahalanobis distance

- Mahalanobis distance between two classes:
  - Assumes Gaussian distributions with equal covariance matrix
  - $D_M = (\mu_1 - \mu_2)^T {S_W}^{-1} (\mu_1 - \mu_2)$
  - E.g., Exercise 6.21
  - What is this S_W?

Dimensionality Reduction by Selection or Extraction

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

- Criteria

  - Mahalanobis distance (vs Euclidean distance)

  - Scatter matrices (what are S_W, S_B, S_T?)

- Approaches

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

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

Scatter Matrices

- Within-class scatter matrix:

$$S_W = \sum_{i=1}^{M} \frac{n_i}{N} \Sigma_i \,,$$

Σ_i is the covariance matrix of class w_i; n_i is the number of samples in class w_i, out of a total of N samples.

- Between-class scatter matrix:

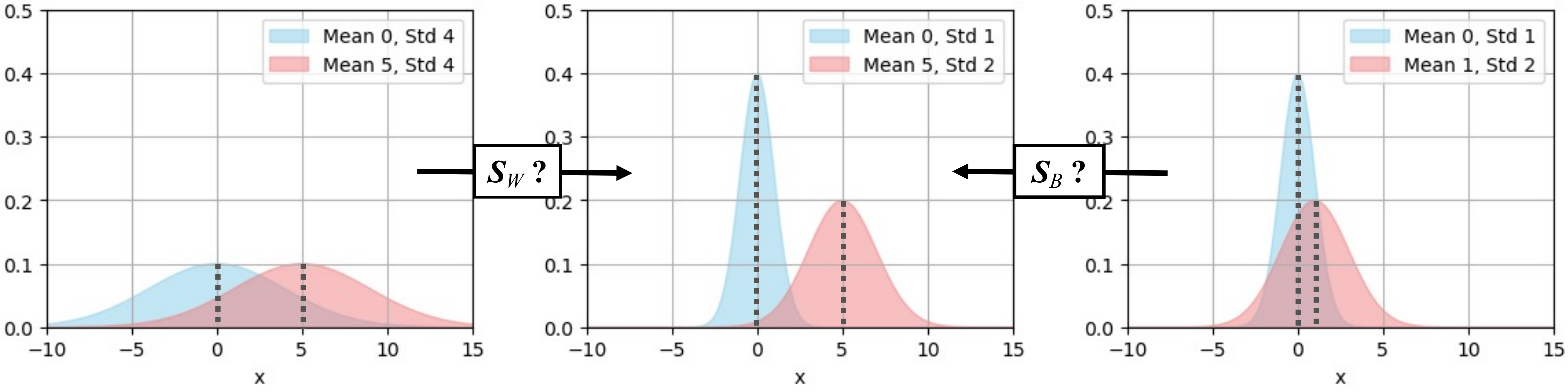
$$S_B = \sum_{i=1}^{M} \frac{n_i}{N} (\mu_i - \mu)(\mu_i - \mu)^T,$$

μ_i is the mean of class w_i, μ is the global mean.

- Total scatter matrix: $S_T = S_W + S_B$

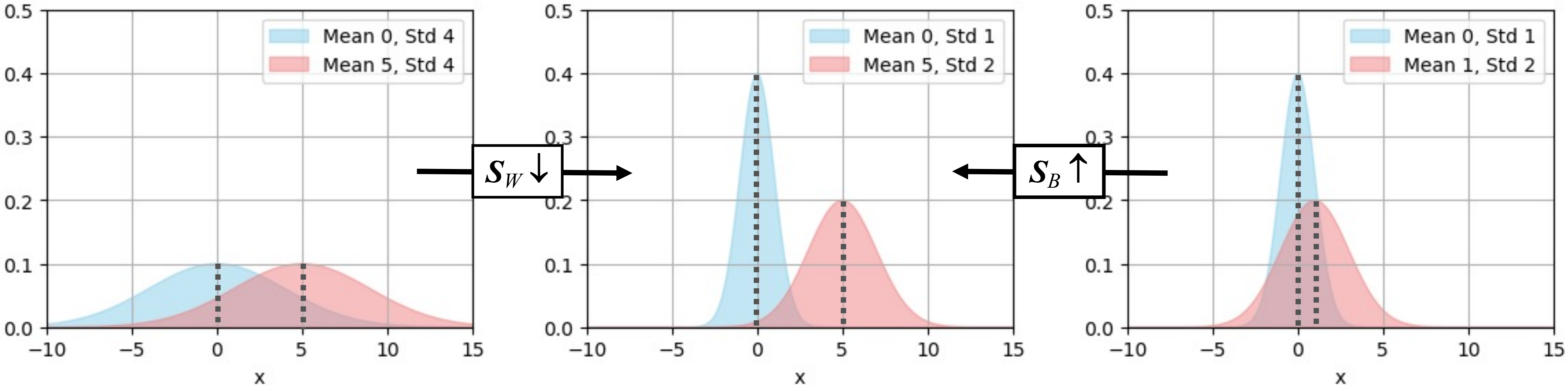
Scatter Matrices

- S_W = "average class width"; the smaller, the better
- S_B = "average distance between class means"; the larger, the better
- S_T = "overall width"



Scatter Matrices

- S_W = "average class width"; the smaller, the better
- S_B = "average distance between class means"; the larger, the better
- S_T = "overall width"



Scatter-based Criteria

- $J_1 = \dfrac{trace\{S_T\}}{trace\{S_W\}}$
- $J_2 = \dfrac{|S_T|}{|S_W|}$
- etc.
- by using various combinations of S_W, S_B, 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.

FDR: Fisher Discriminant Ratio

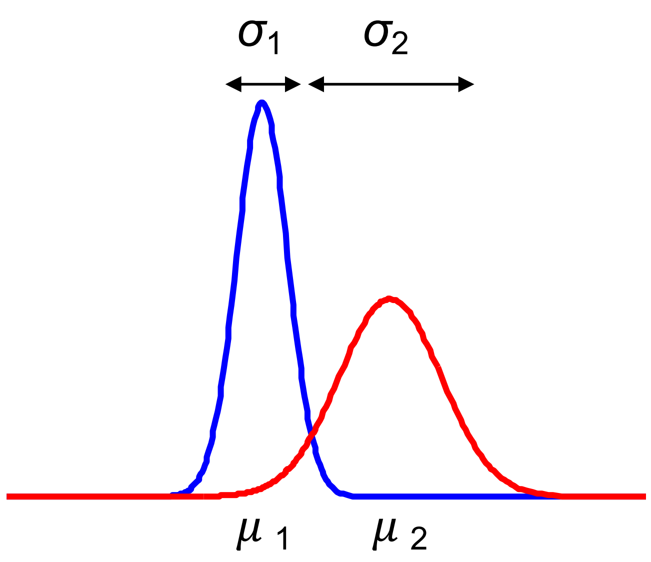
$$S_W = \sum_{i=1}^{M} \frac{n_i}{N} \Sigma_i \,,$$

$$S_B = \sum_{i=1}^{M} \frac{n_i}{N} (\mu_i - \mu)(\mu_i - \mu)^T,$$

- **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.

Dimensionality Reduction by Selection or Extraction

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

- Criteria

  - Mahalanobis distance (vs Euclidean distance)

  - Scatter matrices (what are S_W, S_B, S_T?)

- Approaches

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

  - Principal Component Analysis & Recall LDA (∈ linear **feature extraction**)

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
- **Exhaustive** feature selection
- It can be super Expensive! And Exhaustive!!
- Let's use **Sequential Feature Selection**!

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

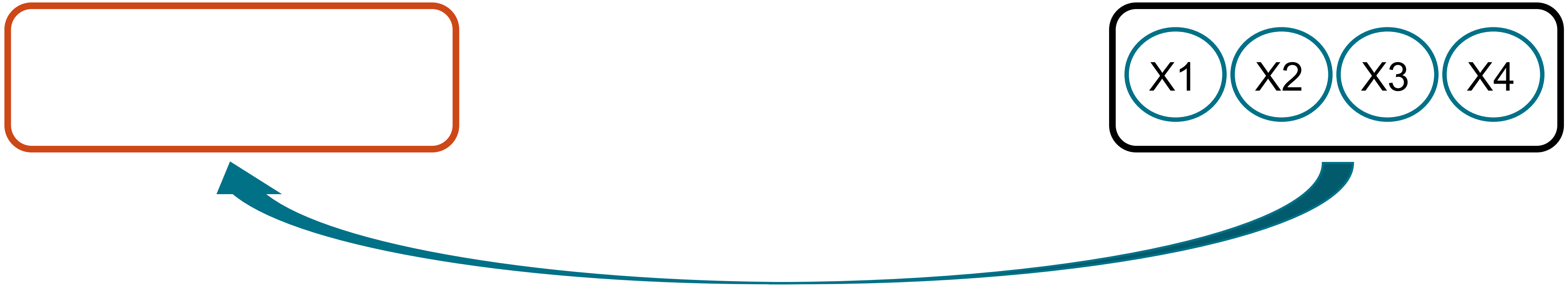
$$\sum_{i=1}^{p} \binom{p}{i}$$ combinations

Feature Selection Methods

- Filter approach
- Wrapper approach
  - *(Sequential Feature Selection)*
  - Individual Selection
  - **Forward Selection (FS)**
  - **Backward Selection (BS)**
  - *(combine the idea of forward and backward selections)*
  - Bidirectional Selection
  - Plus-L Take-away-R Selection
  - Floating Selection

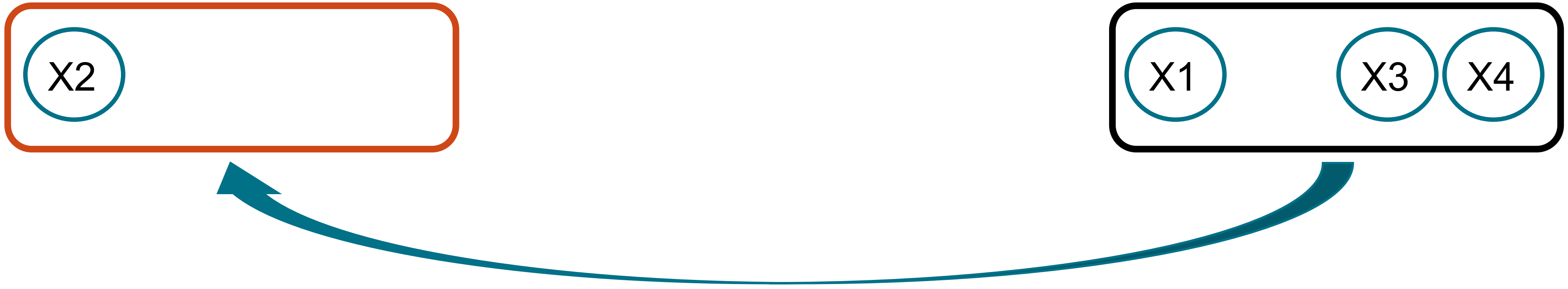
Forward Selection (FS)

- Start with **empty feature set**



Forward Selection (FS)

- Start with **empty feature set**

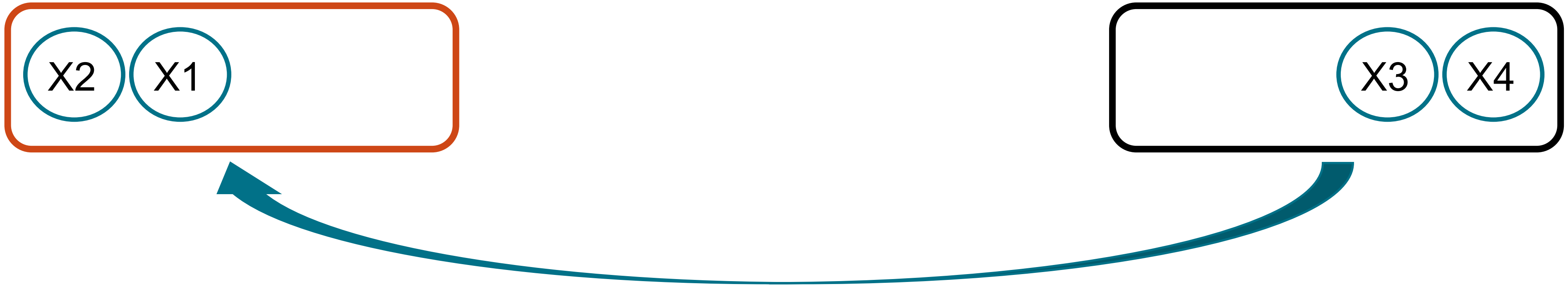


- Compute the criterion value for **each feature individually** and select the best one,

  X2 > X4 > X1 > X3 => X2

Forward Selection (FS)

- Start with **empty feature set**



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

  X2 > X4 > X1 > X3 => X2
- Keep the winner and compute the criterion for all two-feature combinations that include it.

  [X2, X1] > [X2, X4] > [X2, X3]
- … … until a predefined number of features are left.

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,

  [X1, X2, X4] > [X1, X2, X3] > [X2, X3, X4] > [X1, X3, X4]

  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,

  [X1, X2, X4] > [X1, X2, X3] > [X2, X3, X4] > [X1, X3, X4]

  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… …

  [X1, X2] > [X2, X4] > [X1, X4]

  … … until a predefined number of features are left.

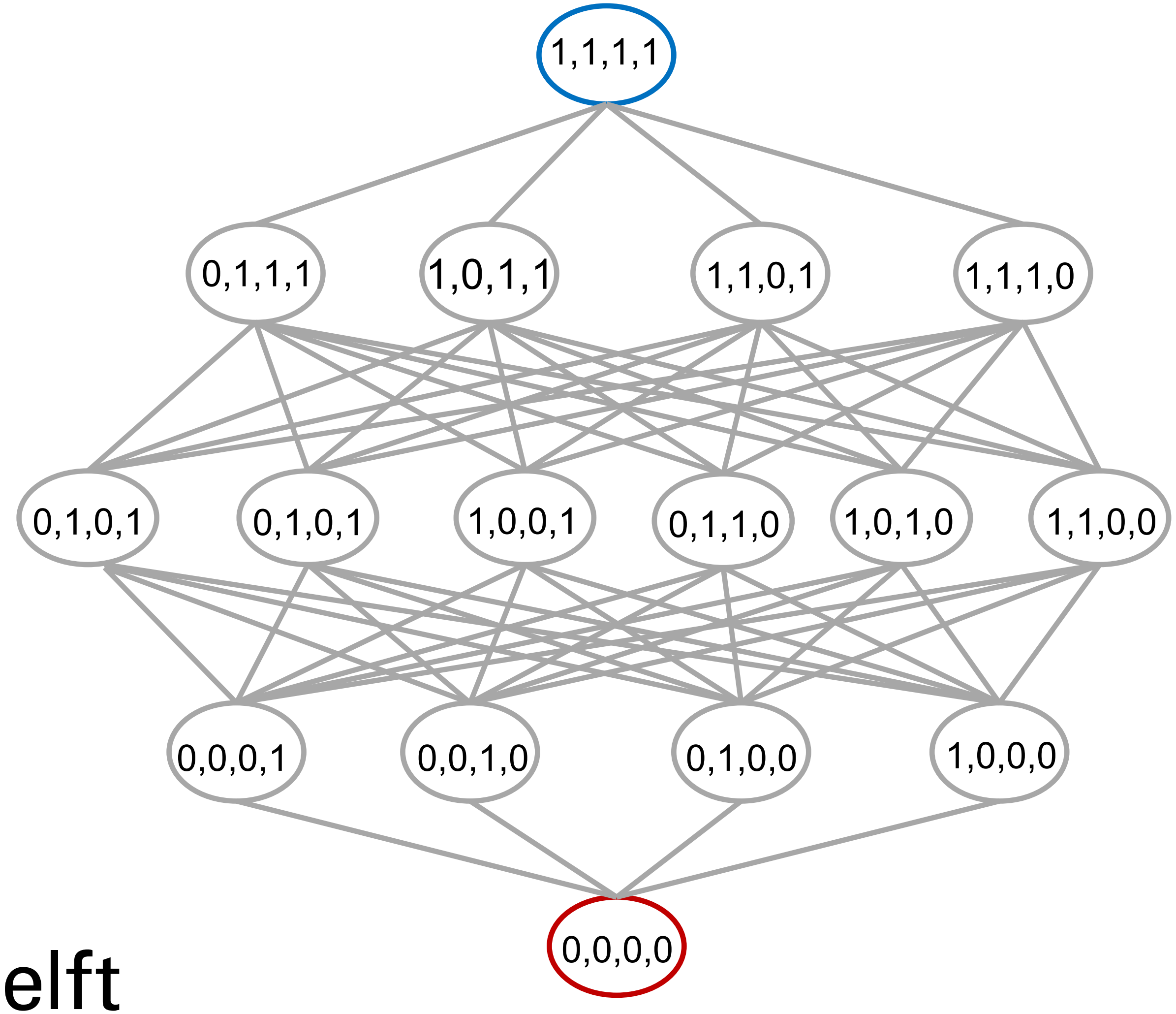
Bidirectional Selection

- 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.

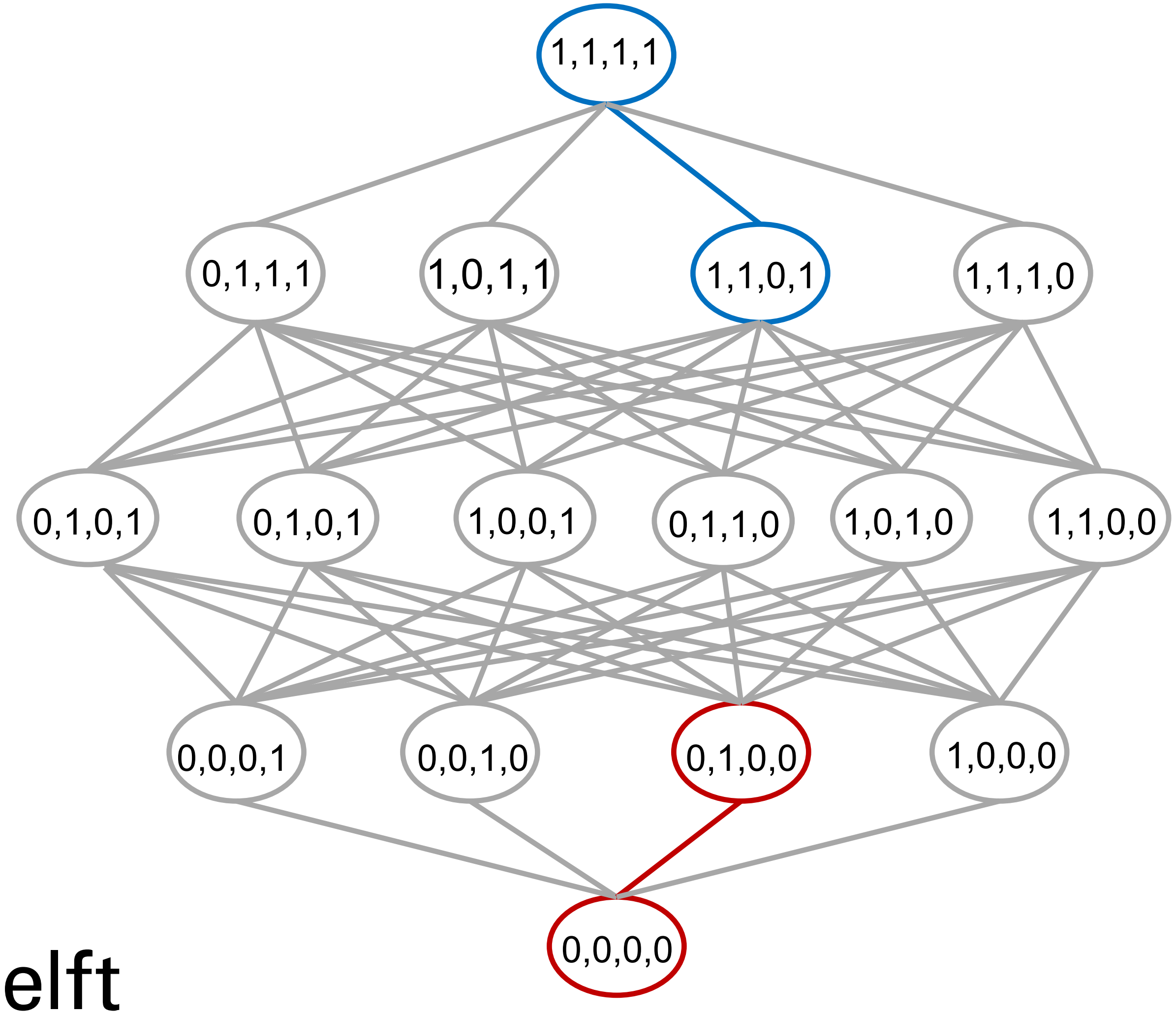


Full set of all originally available features

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.



Full set of all originally available features

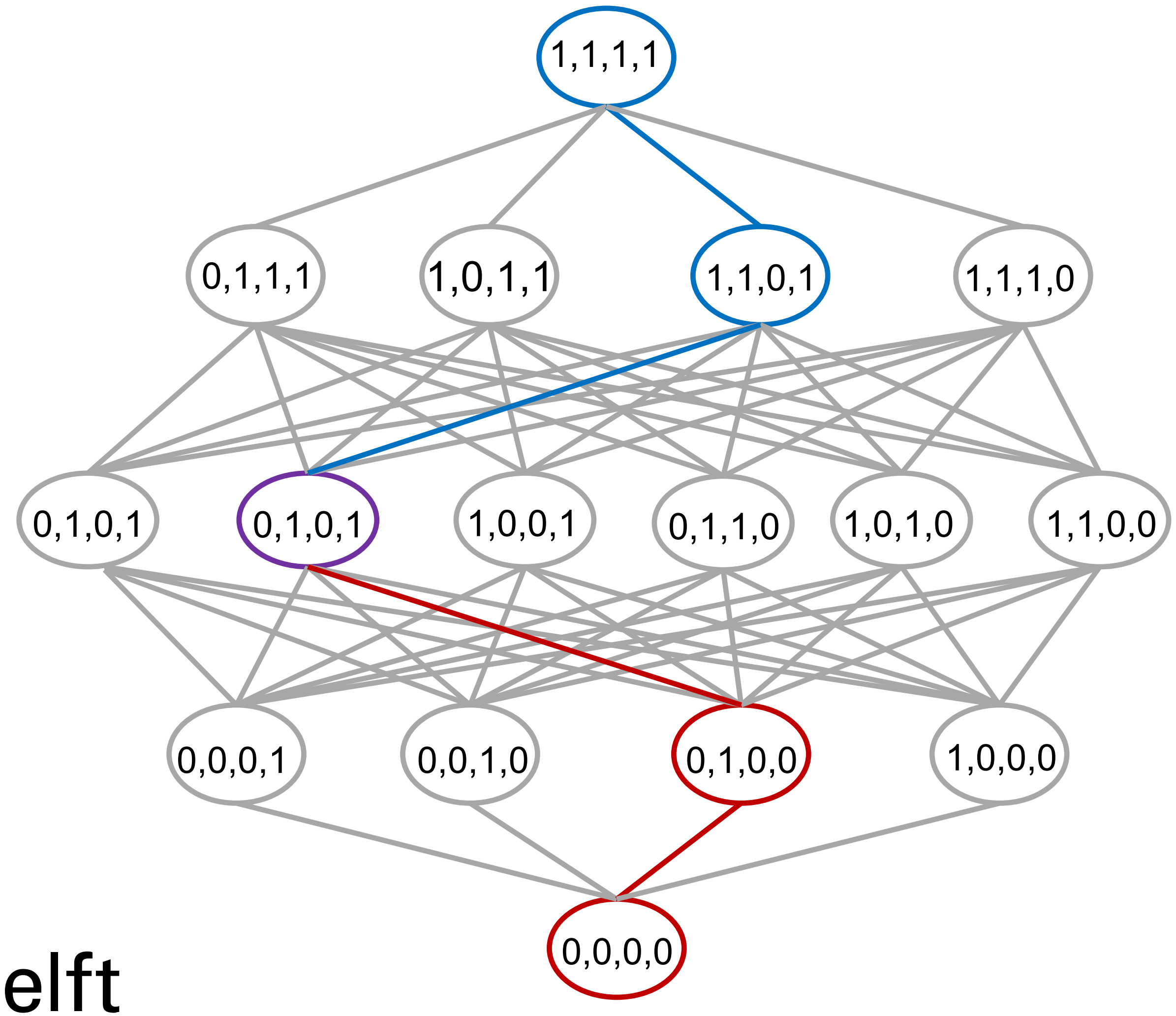
x_2, x_1, x_4

x_2

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.



Full set of all originally available features

x_1, x_2, x_4

x_2, x_4

x_2

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** feature 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. :-(

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 take away R.
- Floating Selection allows flexible backtracking:
  - The dimensionality of the subset during the search can be "floating" up and down.
- There are two floating methods:
  - Floating forward selection & Floating backward selection

Dimensionality Reduction by Selection or Extraction

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

- Criteria

  - Mahalanobis distance (vs Euclidean distance)

  - Scatter matrices (what are S_W, S_B, S_T?)

- Approaches

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

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

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:

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

PCA: offers different view of your data

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

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

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

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

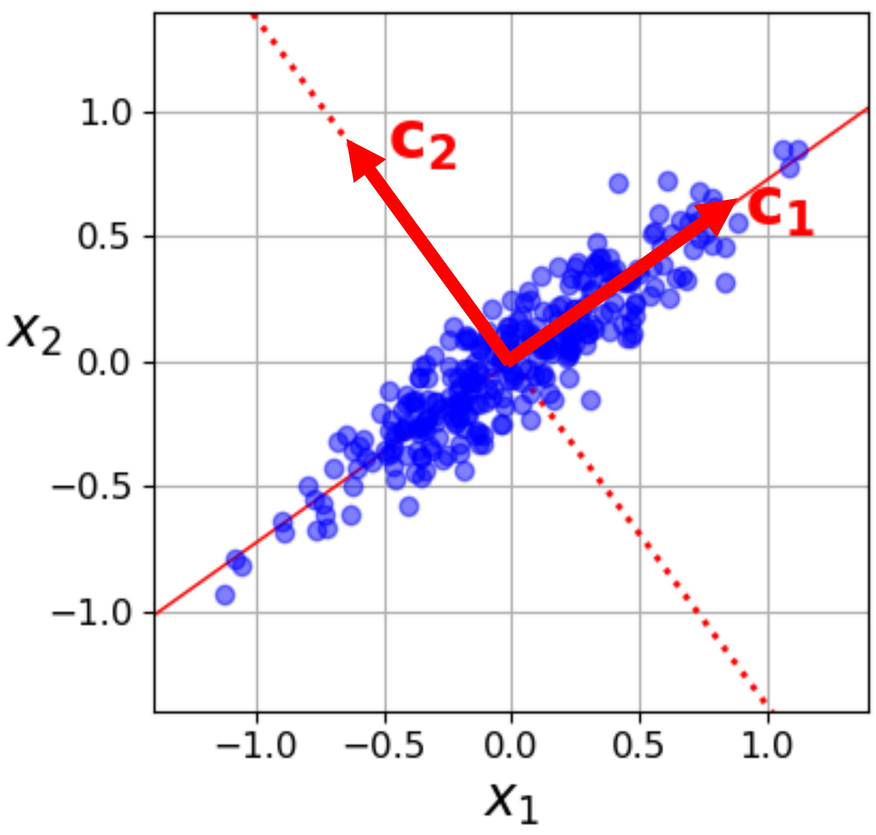
Each column of $\boldsymbol{w}$ is a principal component
ORDERED by the value of λ,
λ_1 is the largest eigenvalue

$$\boldsymbol{t} = \boldsymbol{w}^T\boldsymbol{x} = \begin{bmatrix} \boldsymbol{v}_1^T\boldsymbol{x} \\ \boldsymbol{v}_2^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{v}_k^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{v}_p^T\boldsymbol{x} \end{bmatrix}$$

Quiz:
What is the dimensionality of $\boldsymbol{t}$?
Same with $\boldsymbol{x}$?

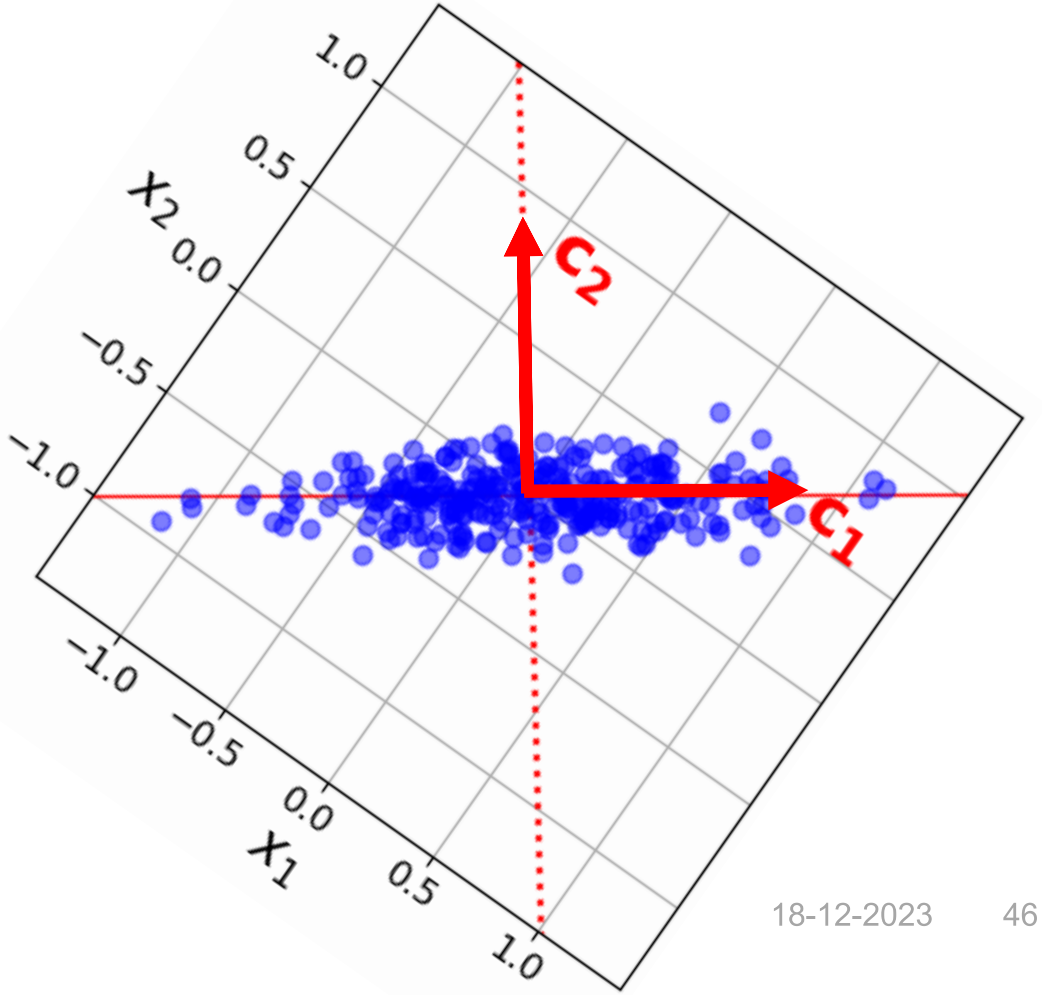
PCA: offers different view of your data

Original Space (2D)



$n = 300$

PCA Space (2D)



PCA: choose to reduce dimensionality

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

$$\boldsymbol{t} = \boldsymbol{w}^T \boldsymbol{x}$$

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

$$\boldsymbol{w} = [\underbrace{\boldsymbol{v}_1 \;\; \boldsymbol{v}_2 \ldots \; \boldsymbol{v}_k}_{\text{keep}} \;|\; \underbrace{\ldots \boldsymbol{v}_p}_{\text{drop}}]$$

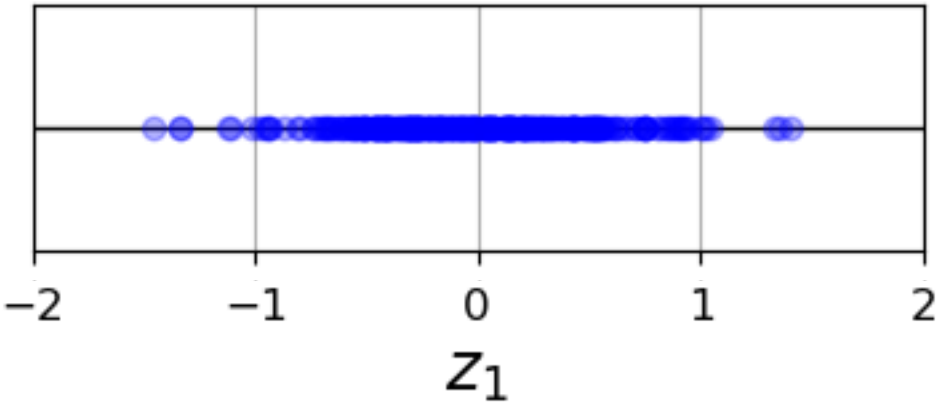
What is the dimensionality of $\boldsymbol{t}_k$?

$$\boldsymbol{t}_k = \boldsymbol{w}_k^T \boldsymbol{x}$$

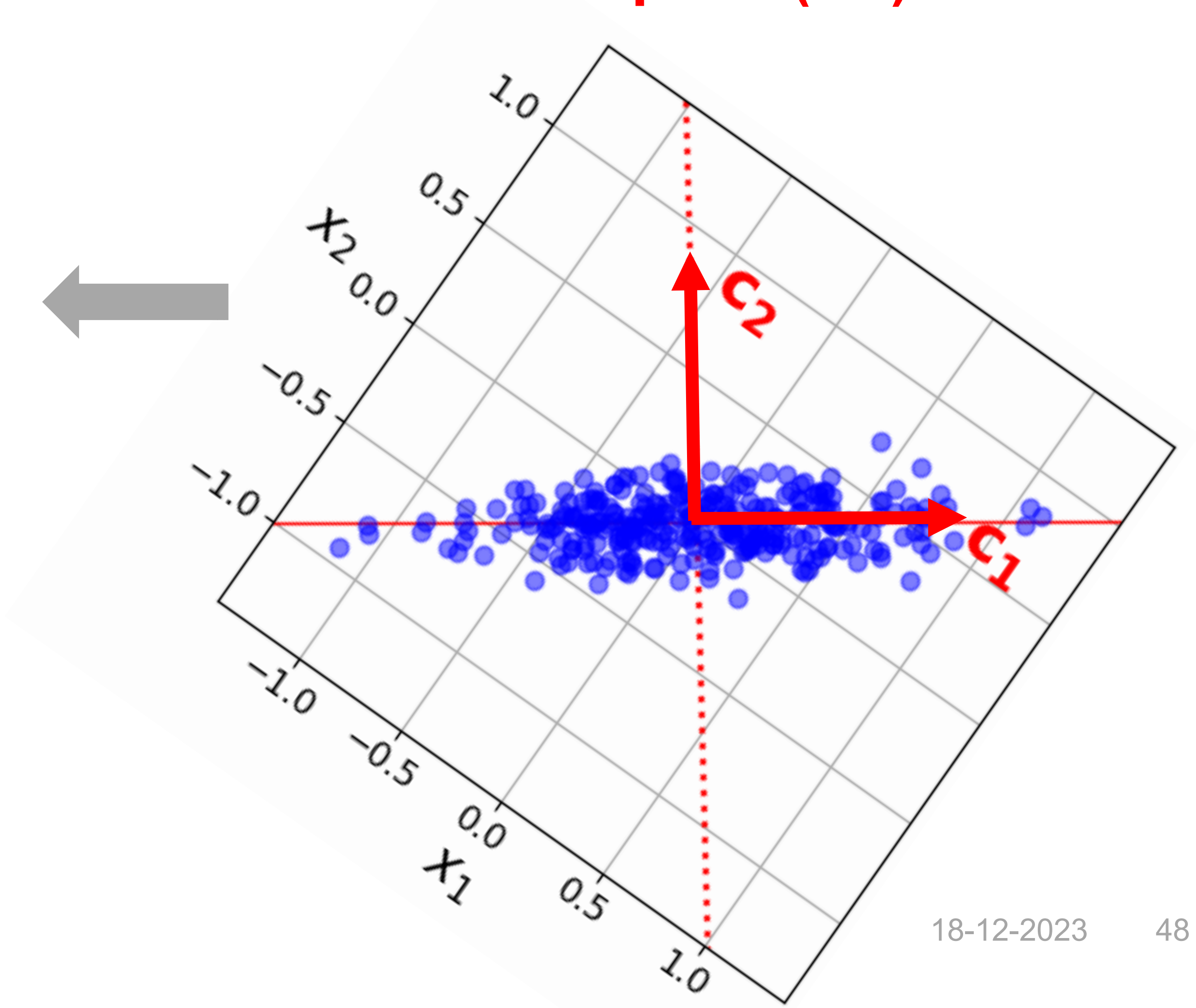
$$\underset{(k \times p)}{\boldsymbol{w}_k^T} = \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_k^T \end{bmatrix}$$

PCA: choose to reduce dimensionality

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



PCA Space (2D)



Quiz

$$\underset{(p \times p)}{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \dots & \boldsymbol{v}_k \dots \boldsymbol{v}_p \end{bmatrix} \qquad \underset{(k \le p)}{\boldsymbol{t}_k} = \boldsymbol{w}_k^T \boldsymbol{x} = \begin{bmatrix} \boldsymbol{v}_1^T \boldsymbol{x} \\ \boldsymbol{v}_2^T \boldsymbol{x} \\ \vdots \\ \boldsymbol{v}_k^T \boldsymbol{x} \end{bmatrix}$$

- 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 $\boldsymbol{v}_1^T \Sigma \boldsymbol{v}_1$ represent?

TUDelft

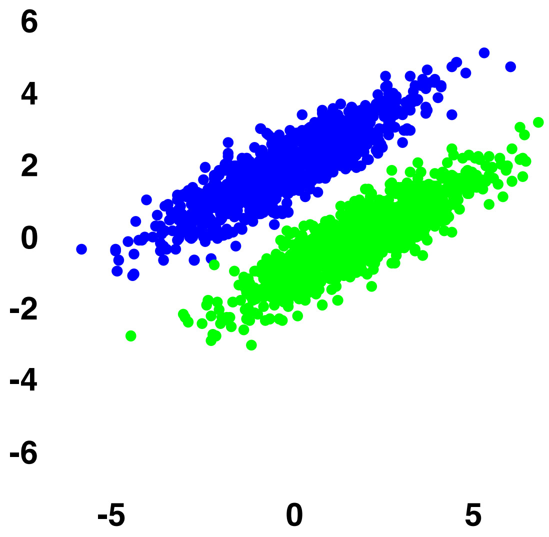
Two classical linear feature extractors

- Supervised:

  Linear Discriminant Analysis (Fisher Mapping) [LDA] / [fisherm]

  -- Capture the greatest separability

$$J(\boldsymbol{a}) = \frac{\boldsymbol{a}^T S_B \boldsymbol{a}}{\boldsymbol{a}^T S_W \boldsymbol{a}}$$



- Unsupervised:

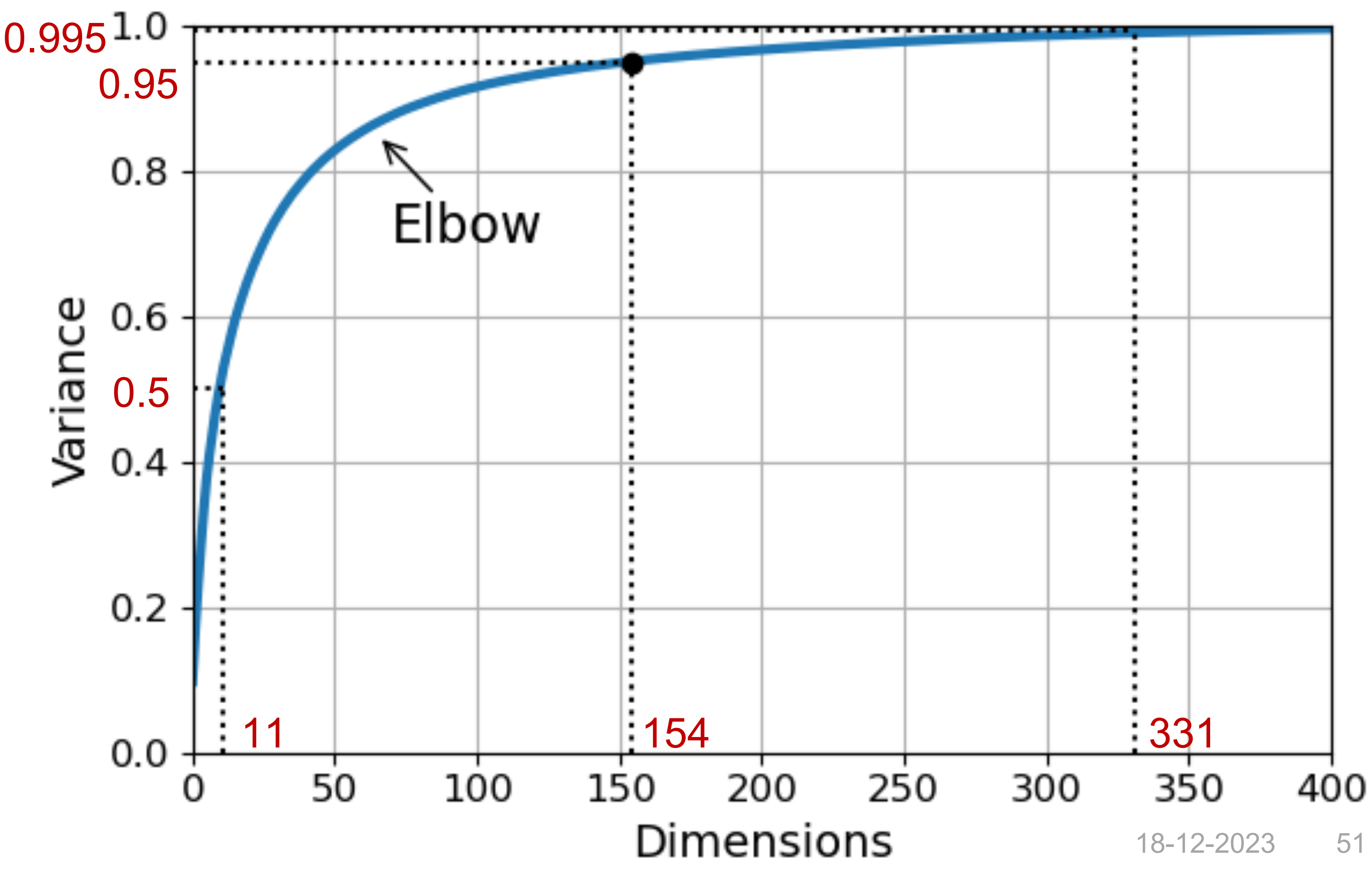
  Principal Component Analysis [PCA]

  -- Capture the greatest variance (global)

$$J(\boldsymbol{a}) = \boldsymbol{a}^T S_T \boldsymbol{a}$$

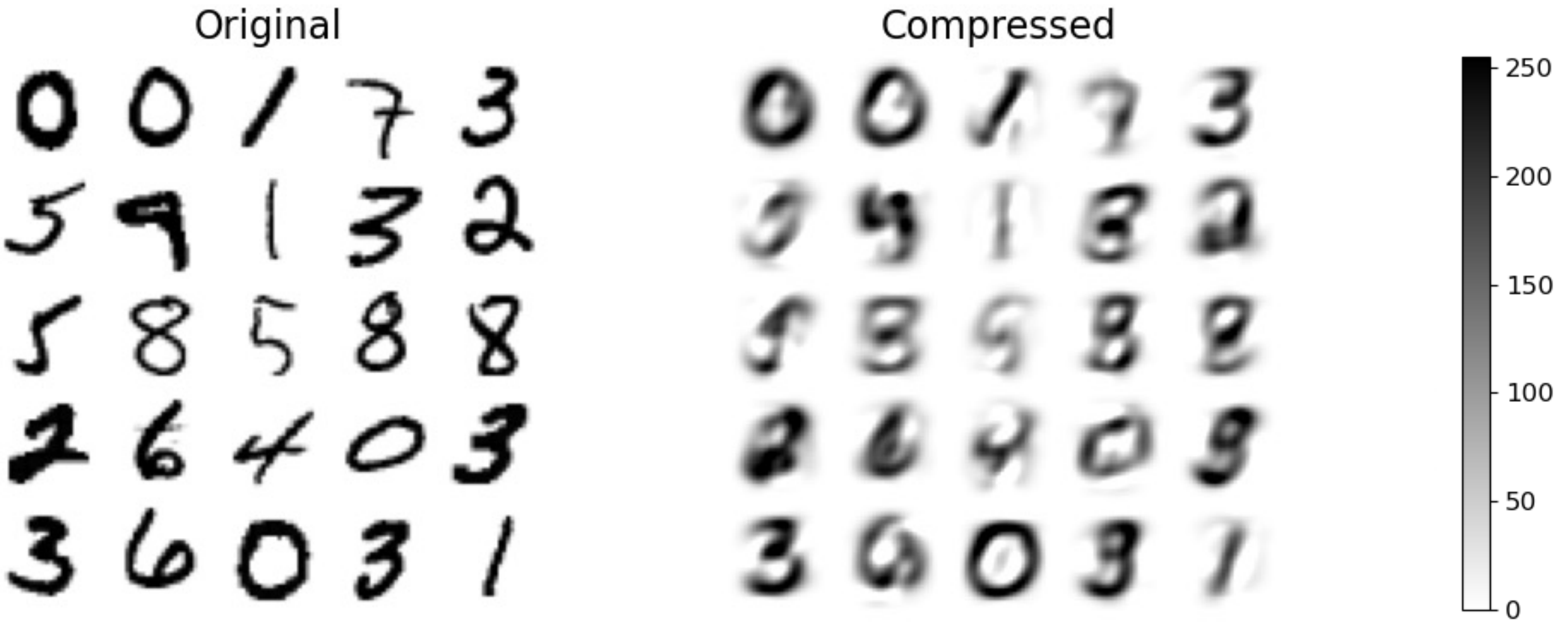
PCA on MNIST data

- PCA reconstructions
- Original space (784 D)



PCA on MNIST data

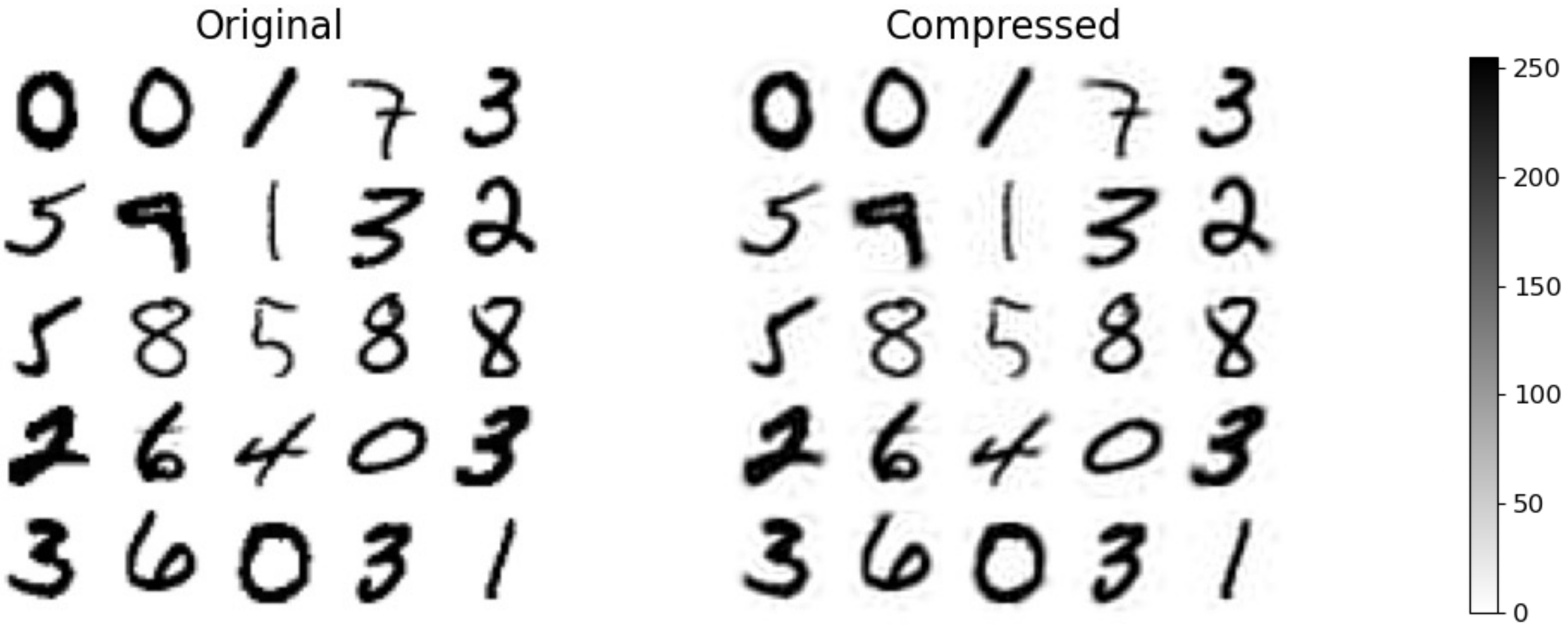
- 50% Variance: Dim = 11



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

PCA on MNIST data

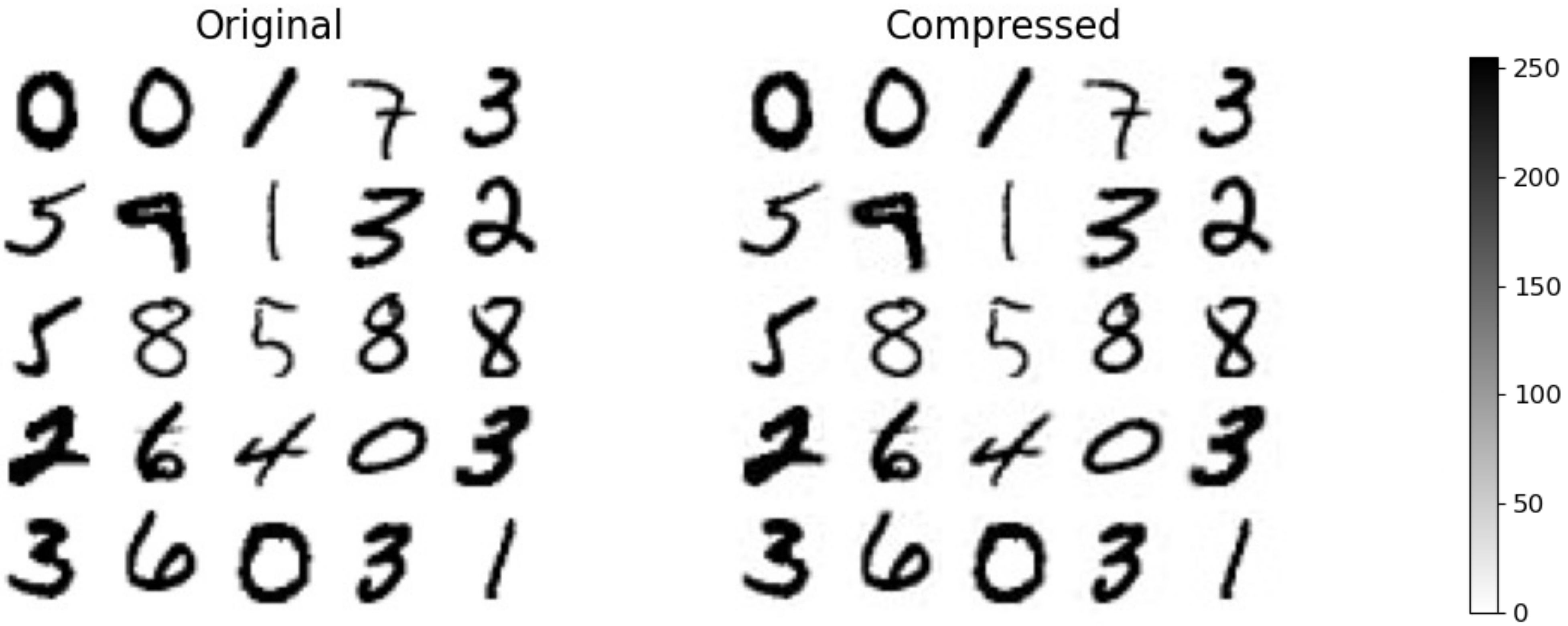
- 95% Variance: Dim = 154



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

PCA on MNIST data

- 99.5% Variance: Dim = 331

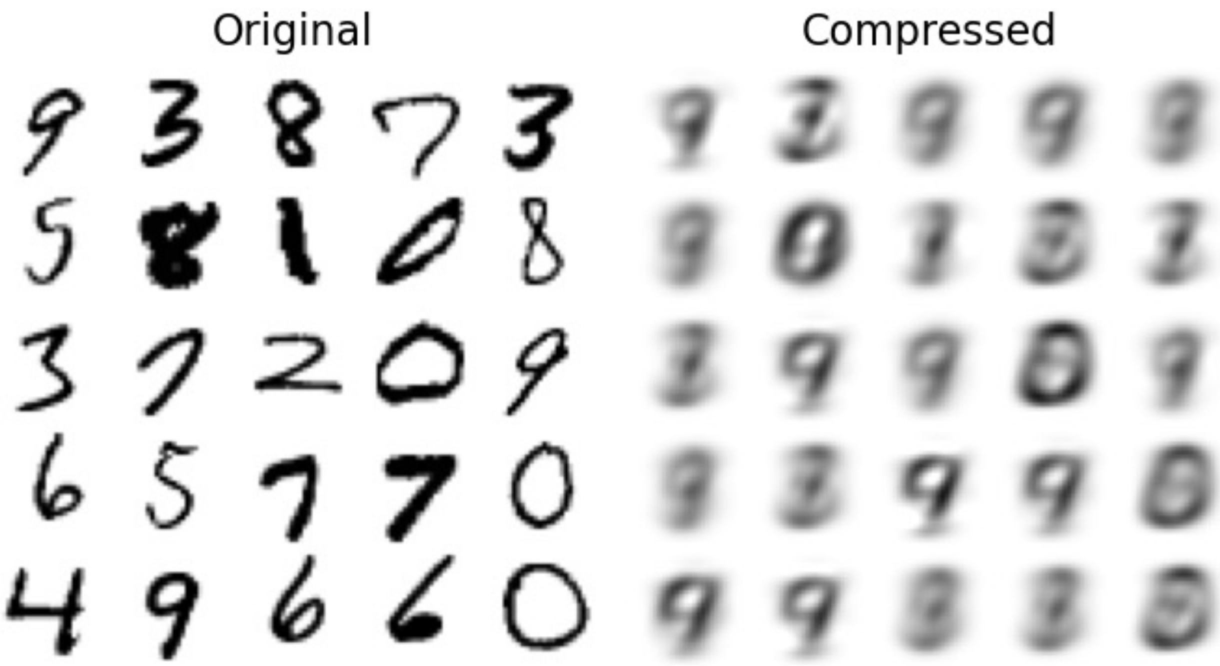


- The more PCs we retain, the smaller the reconstruction error becomes.

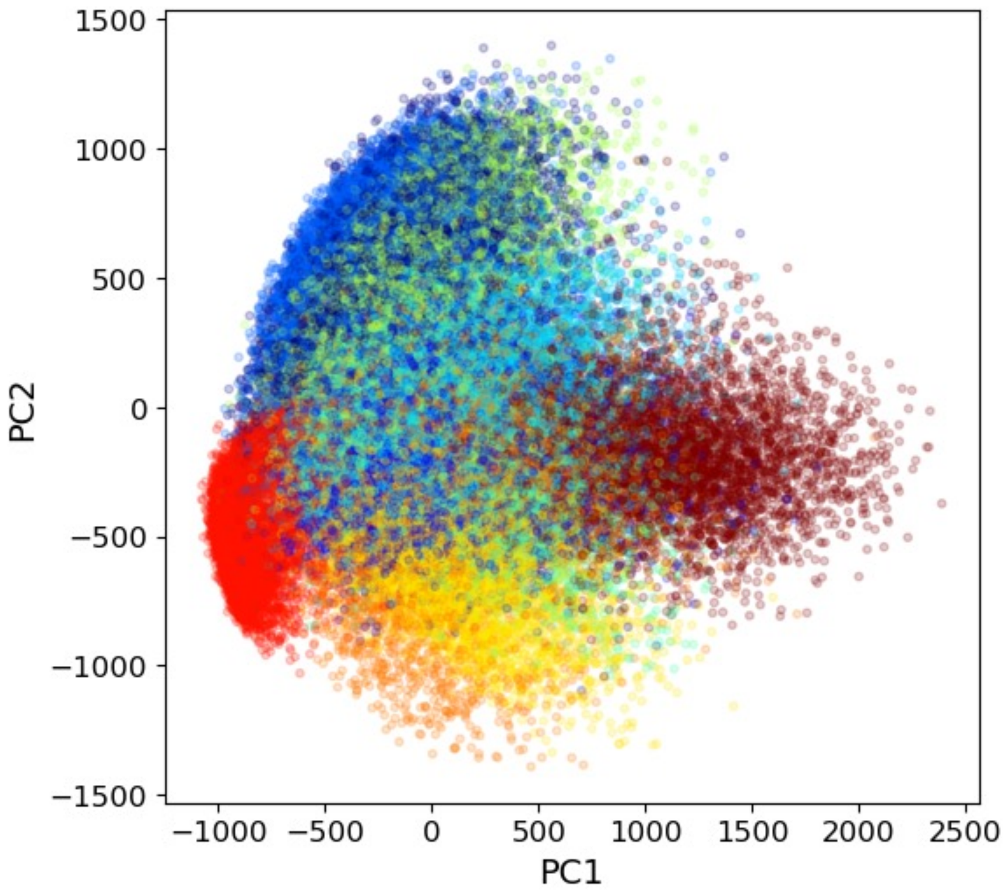
PCA on MNIST data

784 PCs

2 PCs



PCA SPACE

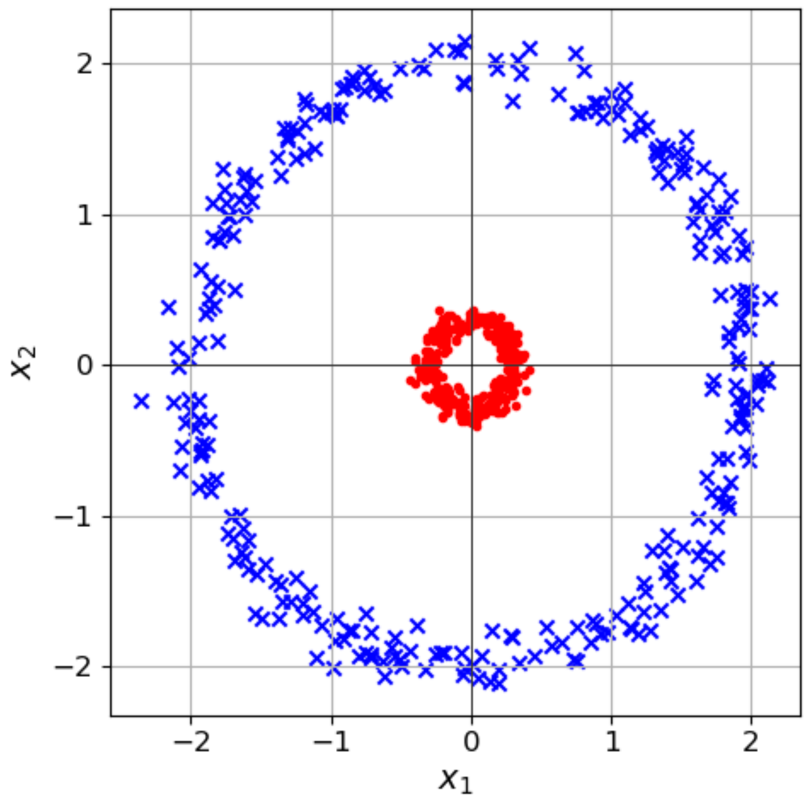


Colours indicate the class of the object

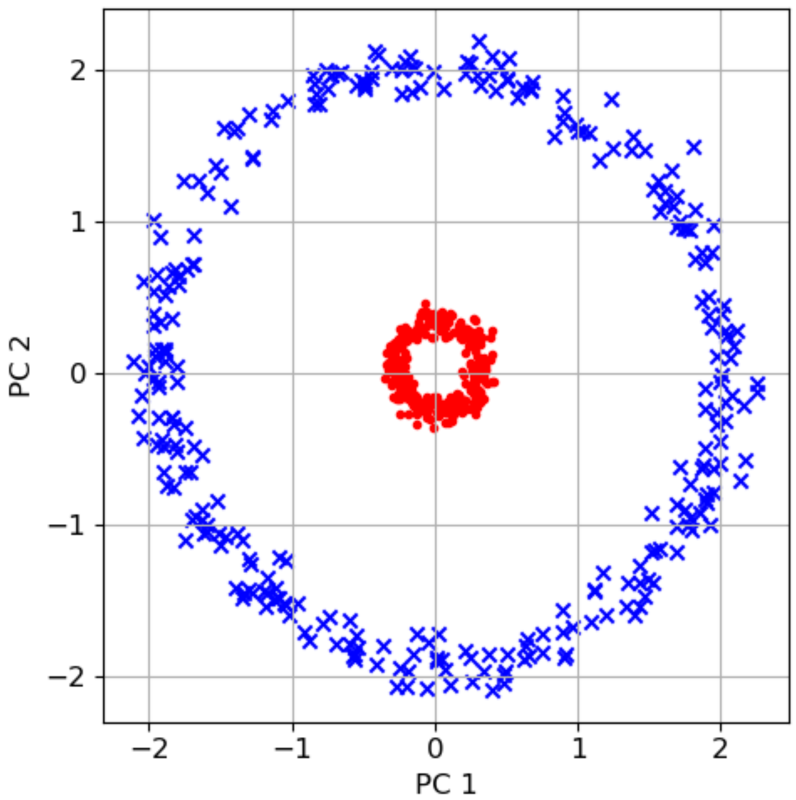
Kernel PCA - Motivation

- PCA is a linear method, i.e., particularly for clustering, it can only be applied to data that are linearly separable.
- However, in the case below, the data are not linearly separable in the original dimension.

Data in 2D space



Projection of the data using PCA

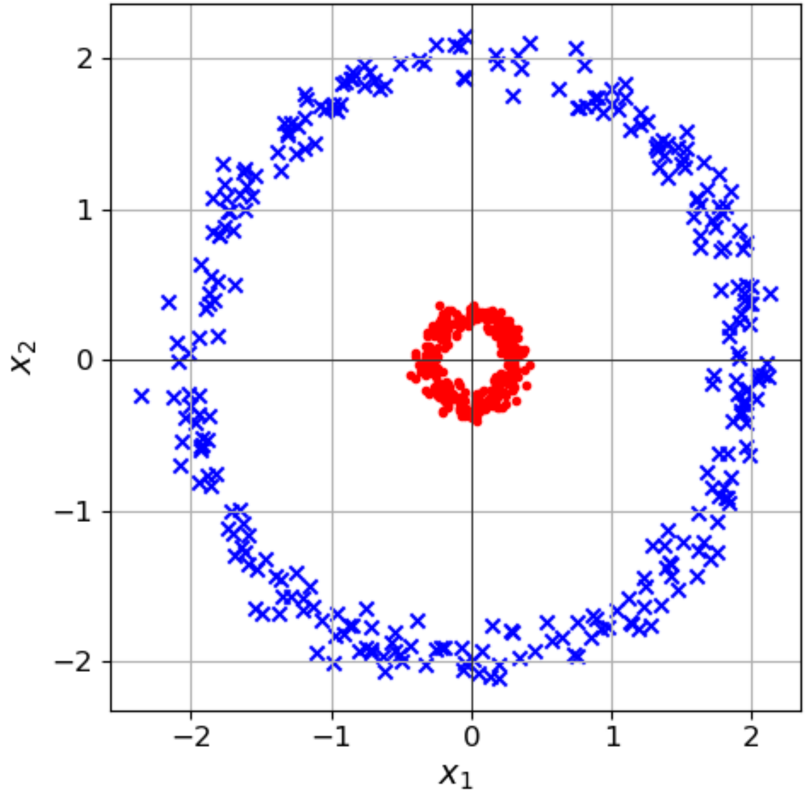


TU Delft

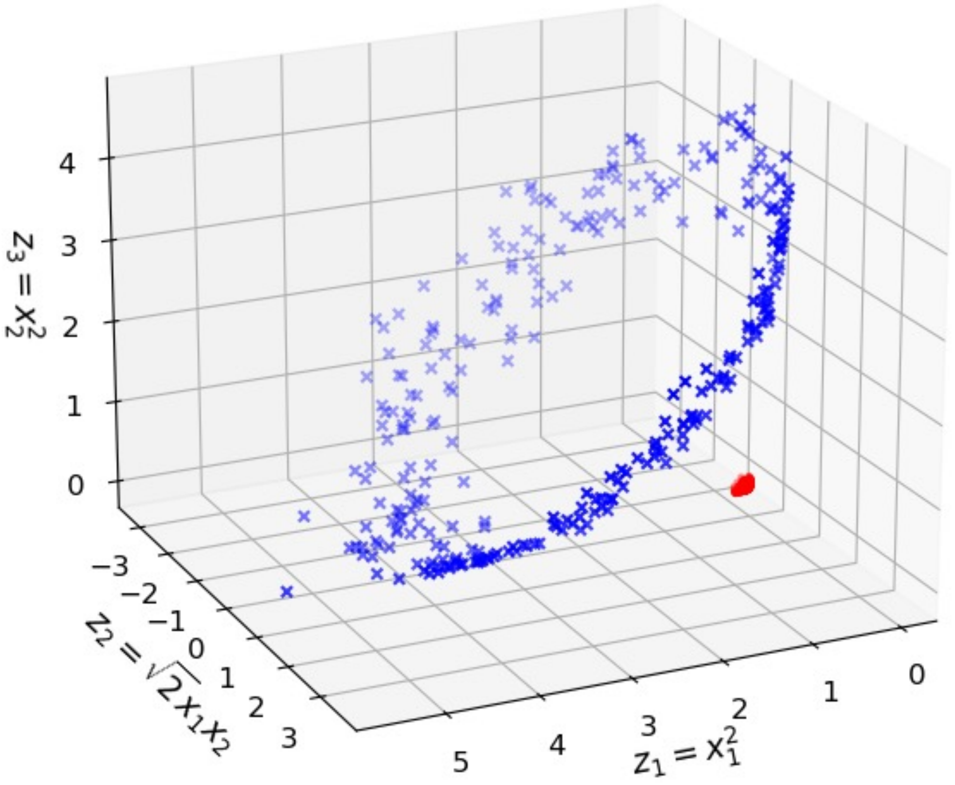
Kernel PCA – What it does

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

Data in 2D space



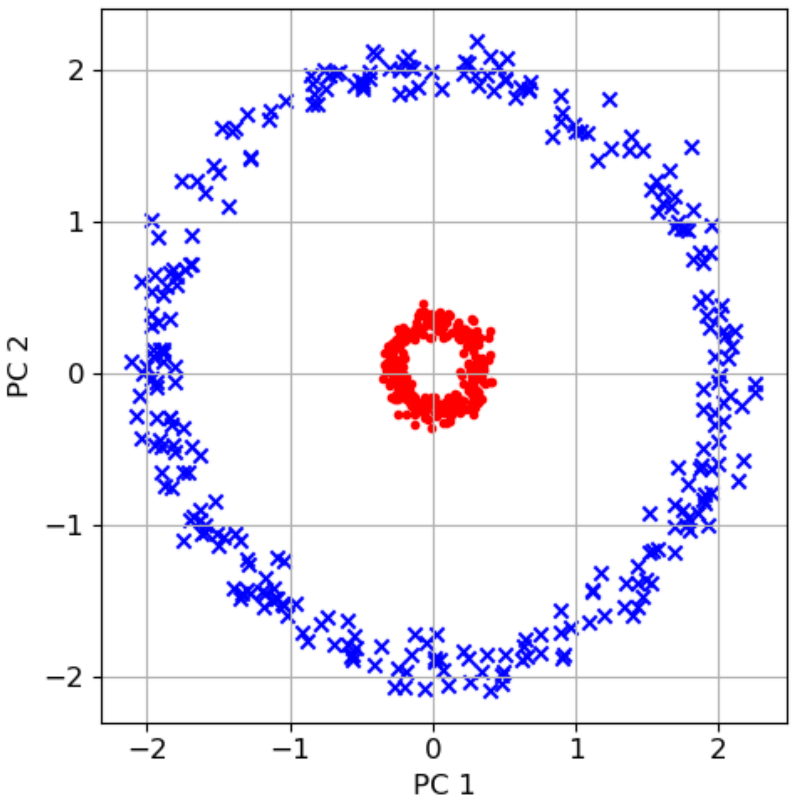
Data mapped to 3D space



Kernel PCA – What it does

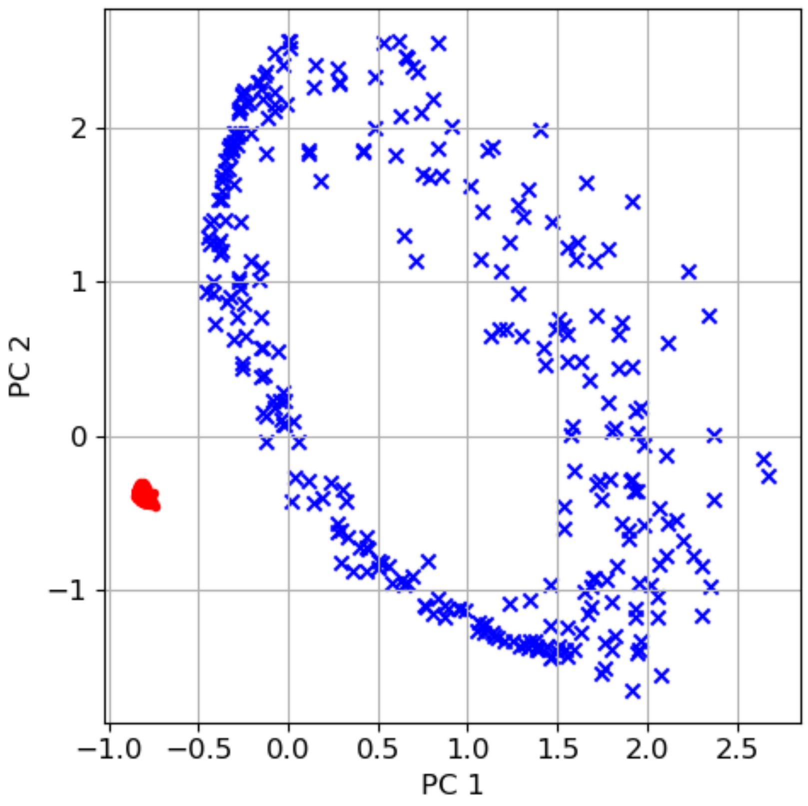
Apply PCA to the original data in 2D space

Projection of the data using PCA



Apply PCA to the data mapped to 3D space

Projection of the data using kernel PCA



Practice

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

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

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 C?

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?

Practice

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

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

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

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 C?

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?

$R\boldsymbol{v}_1$