FYS-2021 Machine Learning

Dimensionality reduction

Slides: Stine Hansen
Guest Lecture: Elisabeth Wetzer

UiT The Arctic University of Norway

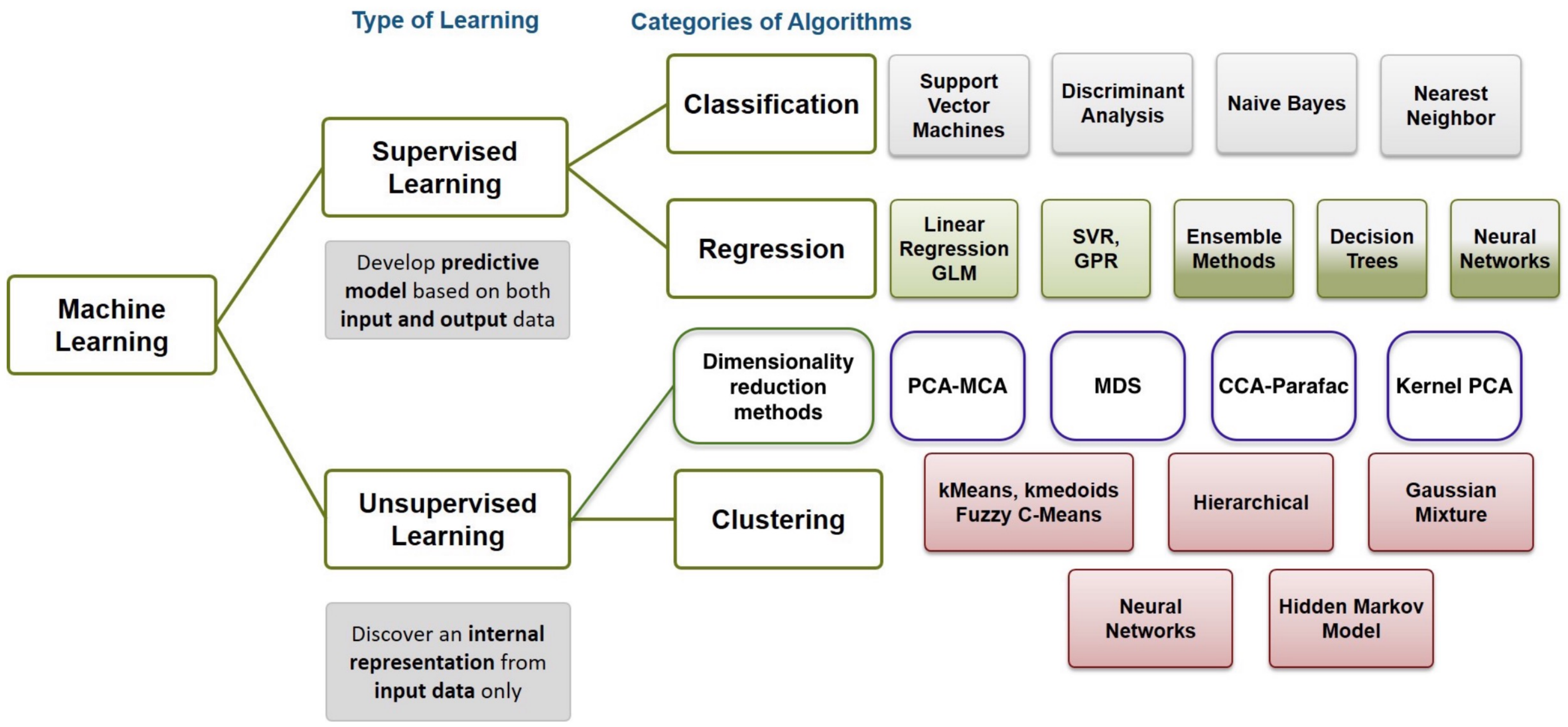
Roadmap

1. Dimensionality reduction: what and why
2. Inner product as similarity measure
3. Multidimensional scaling (from samples)
4. Practical example

Break

5. Multidimensional scaling (from distances)
6. Practical examples
7. Forward selection
8. Practical example

Machine Learning Taxonomy



Source:
https://jchiquet.github.io/ds4m/OverviewDimensionReduction.pdf

Dimensionality reduction

Given a set of n features, the goal in dimensionality reduction is to reduce the number of features to d features ($d < n$), while retaining a meaningful representation of the data.

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} X_q \\ X_r \\ \vdots \\ X_d \end{bmatrix}$$

Selects a subset of important features

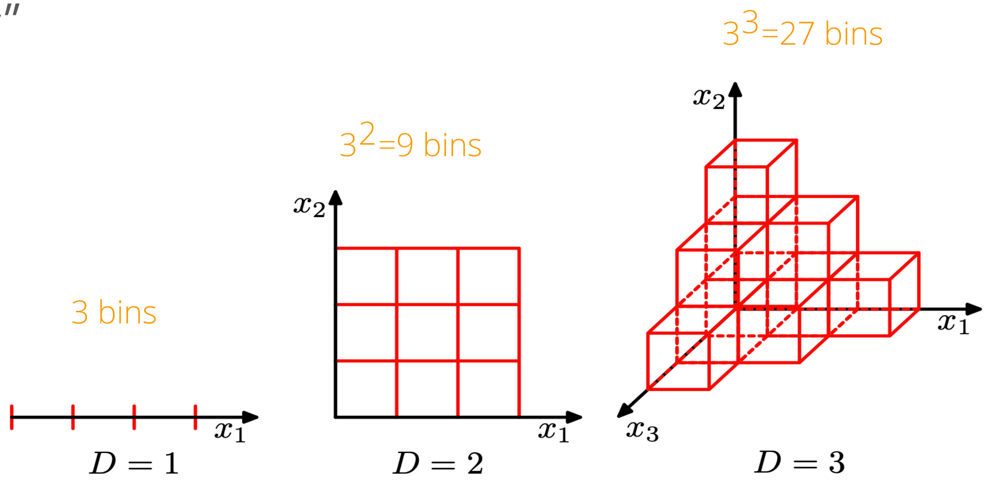
or

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_d \end{bmatrix} = f\left(\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}\right)$$

Extracts a set of new features as a function of the original features

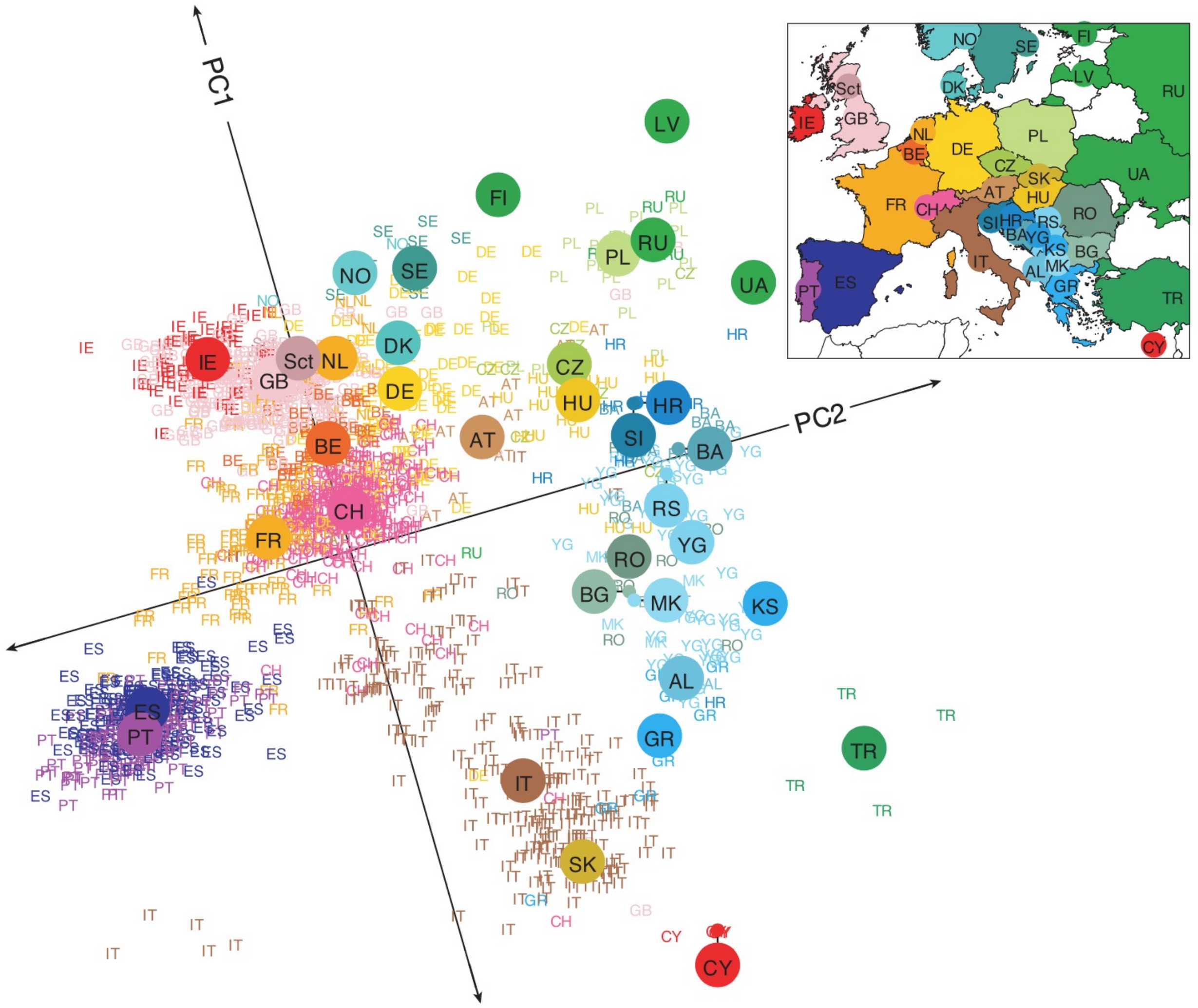
Why dimensionality reduction?

- Compression: Reduce computational complexity/memory usage
- Visualization: Enable visualization of high-dimensional data
- "Curse of dimensionality"



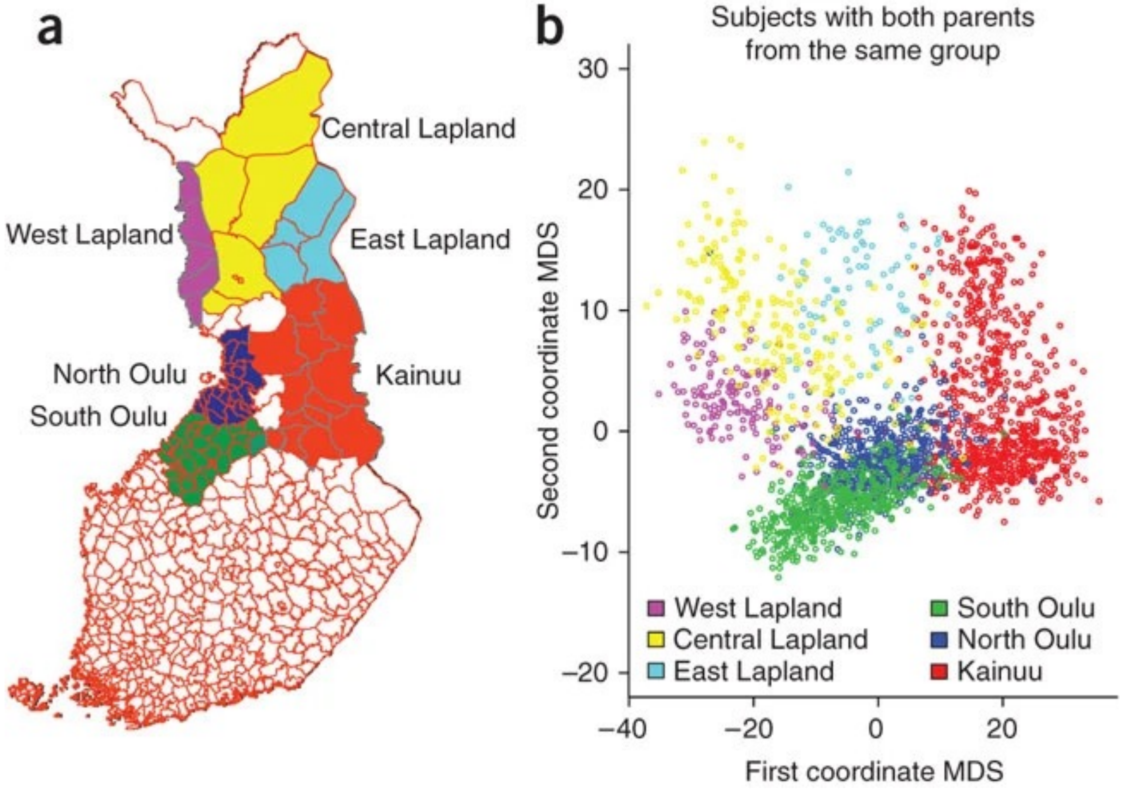
example: DNA

$$\begin{bmatrix} X_1 \\ X_2 \\ \\ \vdots \\ \\ X_{500\,000} \end{bmatrix} \longrightarrow \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$



Example from: Novembre, John, et al. "Genes mirror geography within Europe." Nature 456.7218 (2008): 98-101.

example: DNA



Example from: Sabatti, Chiara, et al. "Genome-wide association analysis of metabolic traits in a birth cohort from a founder population." *Nature genetics* 41.1 (2009): 35-46.

Dimensionality reduction methods

Feature extraction

- Principal component analysis (PCA)
- Linear discriminant analysis (LDA)
- Multidimensional scaling (MDS)
- t-distributed stochastic neighborhood embedding (t-SNE)
- Uniform manifold approximation and projection (UMAP)
- …

Feature selection

- Forward selection
- Backward elimination
- Variance threshold
- L1 regularization
- …

Dimensionality reduction methods

Feature extraction

- Principal component analysis (PCA)
- Linear discriminant analysis (LDA)
- Multidimensional scaling (MDS)
- t-distributed stochastic neighborhood embedding (t-SNE)
- Uniform manifold approximation and projection (UMAP)
- …

Will give abstract features which may be non-linear combination of the original features

Feature selection

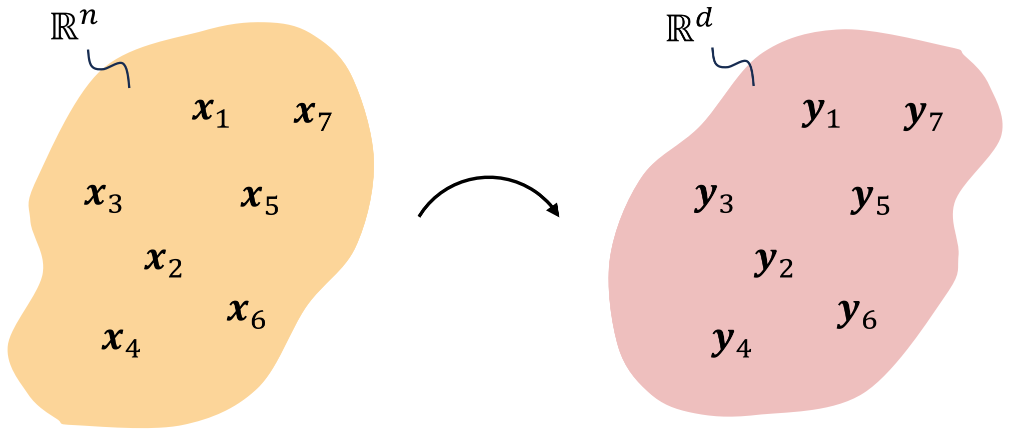
- Forward selection
- Backward elimination
- Variance threshold
- L1 regularization
- …

Selects a suitable subset of existing features

MDS: Multi-Dimensional Scaling

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \longrightarrow \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_d \end{bmatrix} = f\left(\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}\right)$$

is an unsupervised feature extraction method that aims to ***preserve inner products***.

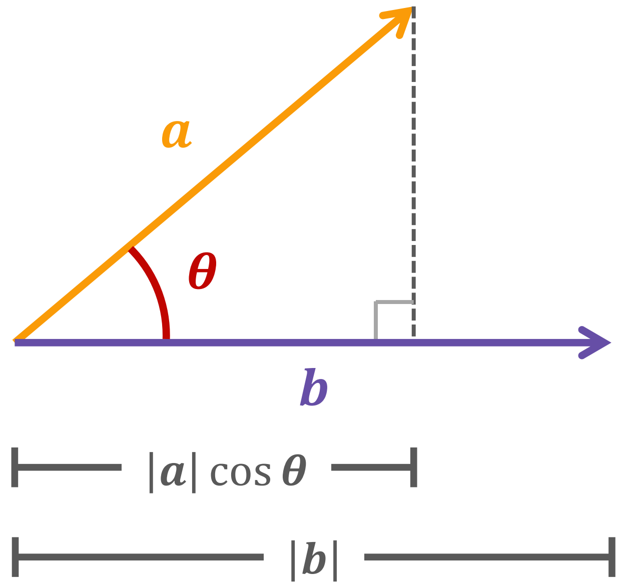


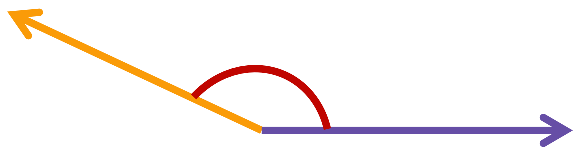
s.t. $\boldsymbol{x}_i\boldsymbol{x}_j^T \approx \boldsymbol{y}_i\boldsymbol{y}_j^T \quad \forall \; i,j$

Inner product: geometric interpretation

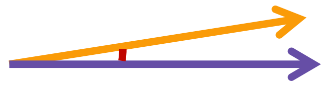
measure of the *similarity* between vectors

$$\begin{aligned} \boldsymbol{ab}^T &= \textstyle\sum_{i=1}^{d} a_i b_i \\ &= |\boldsymbol{a}||\boldsymbol{b}|\cos(\boldsymbol{\theta}) \end{aligned}$$

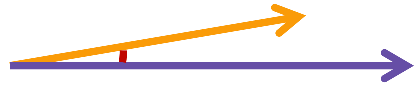




$\boldsymbol{ab}^T = 3 \cdot 3\cos\frac{5\pi}{6} \approx -7.8$



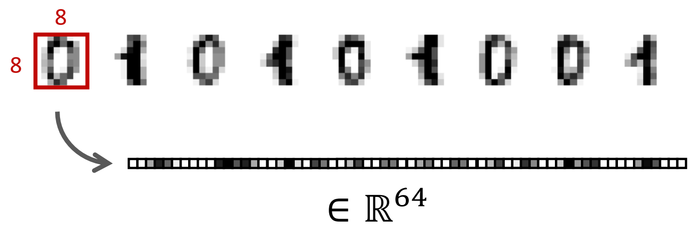
$\boldsymbol{ab}^T = 3 \cdot 3\cos\frac{\pi}{12} \approx 8.7$

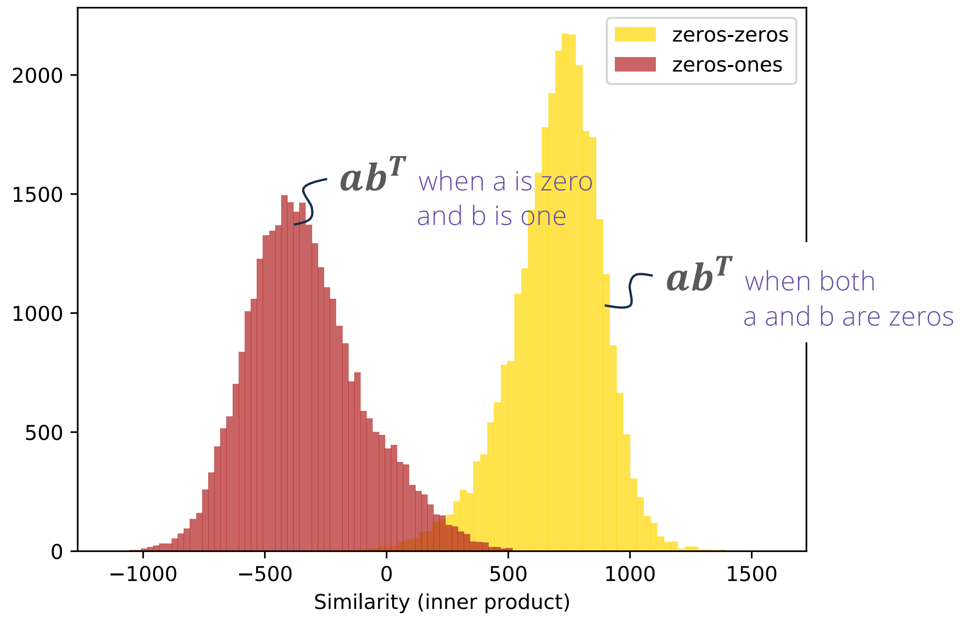


$\boldsymbol{ab}^T = 3 \cdot 4\cos\frac{\pi}{12} \approx 11.6$

example: inner product as similarity measure

vectorize images (size 8x8) of zeros and ones from the digits dataset[1] and compute inner products





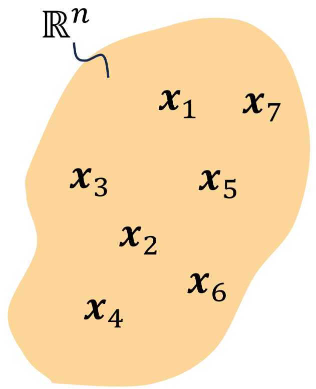
[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

The Gram matrix

Let

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^1 & \rightarrow & \\ \boldsymbol{x}^2 & \rightarrow & \\ & \vdots & \\ \boldsymbol{x}^N & \rightarrow & \end{bmatrix} \in \mathbb{R}^{N \times n}$$

the rows in X are the samples
(feature vectors of length n)



Then

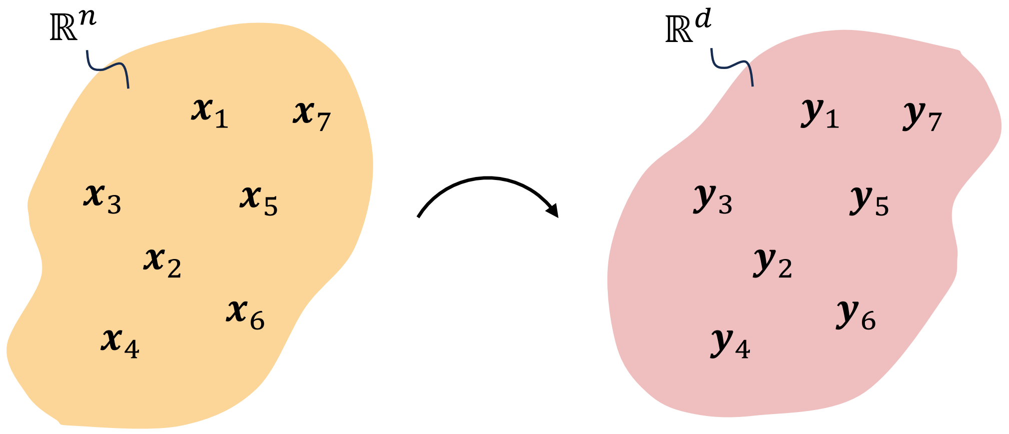
$$\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T = \begin{bmatrix} \boldsymbol{x}^1(\boldsymbol{x}^1)^T & \cdots & \boldsymbol{x}^1(\boldsymbol{x}^N)^T \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}^N(\boldsymbol{x}^1)^T & \cdots & \boldsymbol{x}^N(\boldsymbol{x}^N)^T \end{bmatrix} \in \mathbb{R}^{N \times N}$$

element ij in B is the inner product between x_i and x_j

is called the Gram matrix/inner product matrix and contains all possible inner products between the samples.

MDS: preserving inner products (similarities)

we want to find $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^N \in \mathbb{R}^d$ such that $\underset{\text{Gram matrix of Low dim. samples}}{\boldsymbol{Y}\boldsymbol{Y}^T} \approx \underset{\text{Gram matrix of high dim. samples}}{\boldsymbol{X}\boldsymbol{X}^T}$



Properties of the Gram matrix

The Gram matrix $\mathbf{B} = \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ is:

1. Real all elements are real numbers
2. Symmetric: $\boldsymbol{B}^T = \boldsymbol{B}$ remember: $(A \cdot B)^T = B^T A^T$

Which means that:

1. The eigenvalues of $\mathbf{B}$ are real
2. The eigenvectors of $\mathbf{B}$ are orthonormal

eigenvector of B

$$\mathbf{B}\boldsymbol{e} = \underset{\text{eigenvalue of B}}{\lambda}\boldsymbol{e}$$

Eigen-decomposition of the Gram matrix

Can rewrite the Gram matrix as:

$$\underset{N\times N}{\boldsymbol{B}} = \underset{N\times N}{\boldsymbol{E}^T}\ \underset{N\times N}{\boldsymbol{\Lambda}}\ \underset{N\times N}{\boldsymbol{E}} = \underset{N\times N}{\boldsymbol{E}^T}\ \underset{N\times N}{\boldsymbol{\Lambda}^{1/2}}\ \underset{N\times N}{\boldsymbol{\Lambda}^{1/2}}\ \underset{N\times N}{\boldsymbol{E}}$$

$$\begin{bmatrix} \boldsymbol{e}^1 & \boldsymbol{e}^2 & & \boldsymbol{e}^N \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix} \begin{bmatrix} \lambda^1 & & & \\ & \lambda^2 & & \\ & & \ddots & \\ & & & \lambda^N \end{bmatrix} \begin{bmatrix} \boldsymbol{e}^1 & \rightarrow \\ \boldsymbol{e}^2 & \rightarrow \\ \vdots & \\ \boldsymbol{e}^N & \rightarrow \end{bmatrix} \qquad \begin{bmatrix} \sqrt{\lambda^1} & & & \\ & \sqrt{\lambda^2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda^N} \end{bmatrix} \begin{bmatrix} \sqrt{\lambda^1} & & & \\ & \sqrt{\lambda^2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda^N} \end{bmatrix}$$

MDS: Putting it together

Given our original samples $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N \in \mathbb{R}^n$ MDS aims to find a lower dimensional representation of our data $\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^N \in \mathbb{R}^d$ $(d < n)$ while preserving the inner products (similarities) between samples, $\boldsymbol{YY^T} \approx \boldsymbol{XX^T}$.

Have that

$(A{\cdot}B)^T = B^T A^T$

$$\boldsymbol{XX^T} = \mathbf{B} = \left(\boldsymbol{E^T}\boldsymbol{\Lambda}^{1/2}\right)\left(\boldsymbol{\Lambda}^{1/2}\boldsymbol{E}\right) = \left(\boldsymbol{E^T}\boldsymbol{\Lambda}^{1/2}\right)\left(\boldsymbol{E^T}\boldsymbol{\Lambda}^{1/2}\right)^{\boldsymbol{T}}$$

If we now let $\mathbf{Y} = \boldsymbol{E^T}\boldsymbol{\Lambda}^{1/2}$, then $\boldsymbol{XX^T} = \boldsymbol{YY^T}$! Y as expression of eigenvectors and eigenvalues of XX^T

$$\mathbf{Y} = \underbrace{\begin{bmatrix} \boldsymbol{e}^1 & \boldsymbol{e}^2 & & \boldsymbol{e}^N \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix}}_{N\times N} \underbrace{\begin{bmatrix} \sqrt{\lambda^1} & & & \\ & \sqrt{\lambda^2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda^N} \end{bmatrix}}_{N\times N} = \begin{bmatrix} \sqrt{\lambda^1}\boldsymbol{e}^1 & \sqrt{\lambda^2}\boldsymbol{e}^2 & & \sqrt{\lambda^N}\boldsymbol{e}^N \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix} \in \mathbb{R}^{N\times N}$$

remember: we wanted to reduce the dimensionality to d (d<n). How?

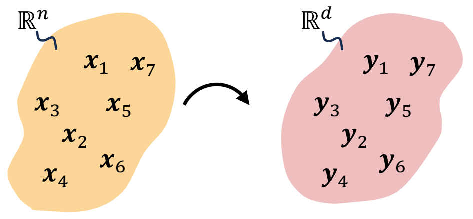
MDS: Putting it together

To obtain d dimensional representations for $\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^N$ such that $\boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}}$ is *as close as possible* to $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$, we can choose to set $\mathbf{Y} = \boldsymbol{E}_d^T \boldsymbol{\Lambda}_d^{1/2}$, where $\boldsymbol{E}_d^T$ is the matrix of eigenvectors corresponding to the d largest eigenvalues and $\boldsymbol{\Lambda}_d^{1/2}$ is the diagonal matrix with the d largest eigenvalues.

$$\mathbf{Y} = \underbrace{\begin{bmatrix} \boldsymbol{e}^1 & \boldsymbol{e}^2 & & \boldsymbol{e}^d \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix}}_{N\times d} \underbrace{\begin{bmatrix} \sqrt{\lambda^1} & & & \\ & \sqrt{\lambda^2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda^d} \end{bmatrix}}_{d\times d} = \begin{bmatrix} \sqrt{\lambda^1}\boldsymbol{e}^1 & \sqrt{\lambda^2}\boldsymbol{e}^2 & & \sqrt{\lambda^d}\boldsymbol{e}^d \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix} \in \mathbb{R}^{N\times d}$$

MDS: algorithm

Given our original samples $\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N \in \mathbb{R}^n$ MDS aims to find a lower dimensional representation of our data $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^N \in \mathbb{R}^d$ $(d < n)$ while preserving the inner products (similarities) between samples, $\boldsymbol{Y}\boldsymbol{Y}^T \approx \boldsymbol{X}\boldsymbol{X}^T$.



$\mathbb{R}^n$
$\boldsymbol{x}_1$ $\boldsymbol{x}_7$
$\boldsymbol{x}_3$ $\boldsymbol{x}_5$
$\boldsymbol{x}_2$
$\boldsymbol{x}_6$
$\boldsymbol{x}_4$
$\mathbb{R}^d$
$\boldsymbol{y}_1$ $\boldsymbol{y}_7$
$\boldsymbol{y}_3$ $\boldsymbol{y}_5$
$\boldsymbol{y}_2$
$\boldsymbol{y}_6$
$\boldsymbol{y}_4$

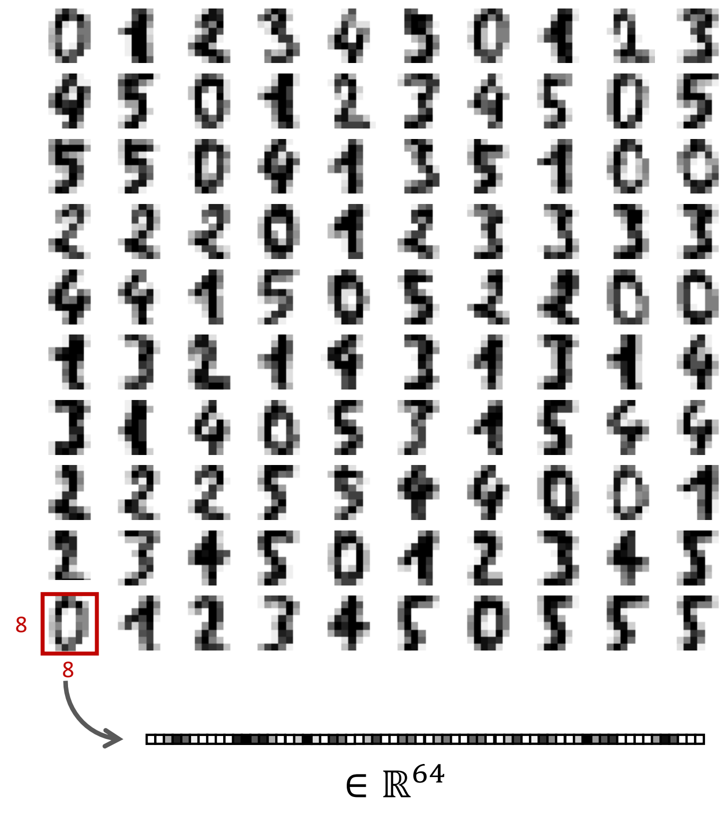
Note

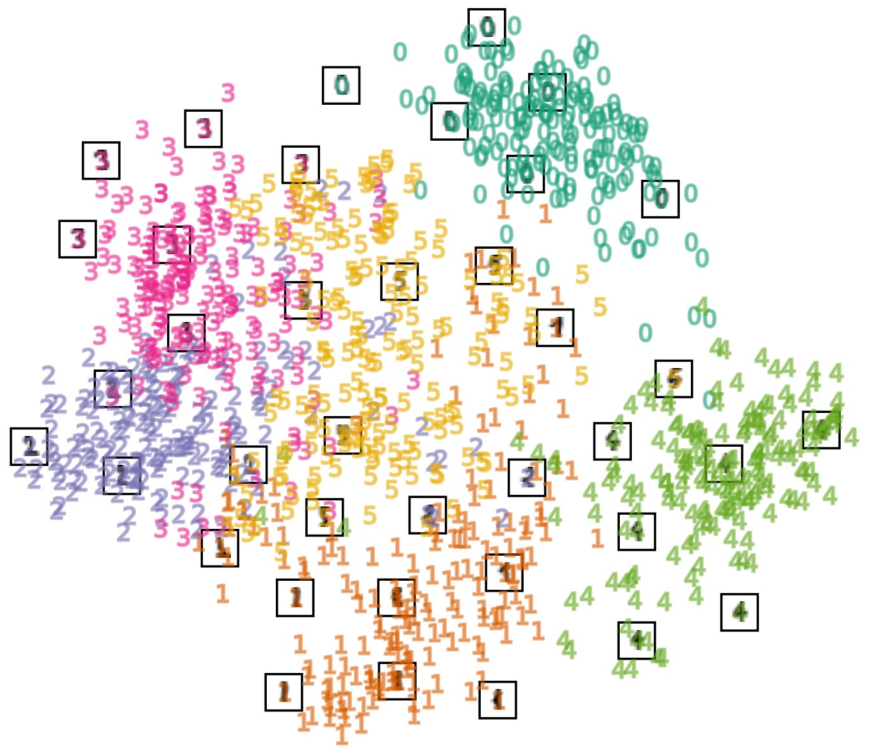
$[\lambda^1, \lambda^2, \dots, \lambda^N]$ E^T
eigenvalues, eigenvectors = np.linalg.eig(B)

column eigenvectors[:,i] is the eigenvector corresponding to the eigenvalue eigenvalues[i]

1. Compute the Gram matrix $\mathbf{B} = \boldsymbol{X}\boldsymbol{X}^T$
2. Compute eigenvectors and eigenvalues of $\mathbf{B}$
3. Sort eigenvalues and eigenvectors
4. Construct a matrix $\boldsymbol{\Lambda}_d$ with the d largest eigenvalues on the diagonal
5. Construct a matrix $\boldsymbol{E}_d^T$ where the columns are the eigenvectors corresponding to the d largest eigenvalues
6. Obtain $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^N$ as the rows of $\mathbf{Y} = \boldsymbol{E}_d^T \boldsymbol{\Lambda}_d^{1/2}$

example: image dataset





Roadmap

1. Dimensionality reduction: what and why
2. Inner product as similarity measure
3. Multidimensional scaling (from samples)
4. Practical example

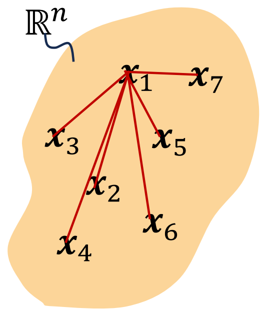
Break

5. Multidimensional scaling (from distances)
6. Practical examples
7. Forward selection
8. Practical example

Gram matrix from pairwise distances

can **not** compute inner products directly

What if we don't have access to the original samples $\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N \in \mathbb{R}^n$, but only to the matrix $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ containing all squared pairwise distances $d_{ij} = \left\|\boldsymbol{x}^i - \boldsymbol{x}^j\right\|^2$?



$$\boldsymbol{D} = \begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} & d_{17} \\ & 0 & d_{23} & d_{24} & d_{25} & d_{26} & d_{27} \\ & & 0 & d_{34} & d_{35} & d_{36} & d_{37} \\ & & & 0 & d_{45} & d_{46} & d_{47} \\ & & & & 0 & d_{56} & d_{57} \\ & & & & & 0 & d_{67} \\ & & & & & & 0 \end{bmatrix}$$

We can compute the Gram matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ consistent with these distances as a function of elements in $\boldsymbol{D}$!

$$\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}\boldsymbol{C}^T$$

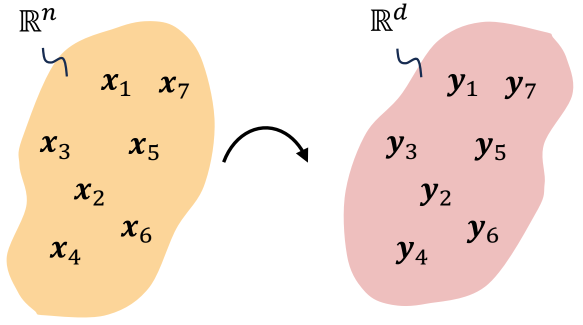
assumption: the data is centered around the origin ("double-centering")

where $\boldsymbol{C} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{J}_N$, and $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{J}_N$ is an $\boldsymbol{N x N}$ matrix of all ones. Thus use MDS to obtain a set of samples $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^N \in \mathbb{R}^d$ such that $\boldsymbol{Y}\boldsymbol{Y}^T \approx \boldsymbol{X}\boldsymbol{X}^T$.

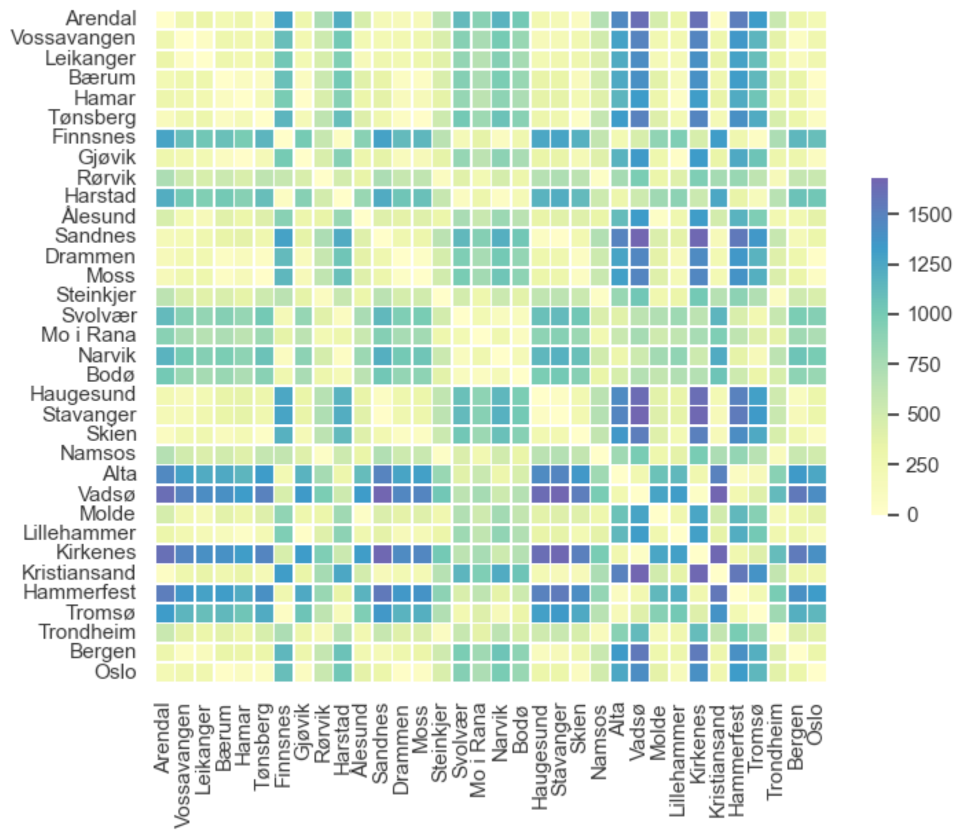
MDS: algorithm

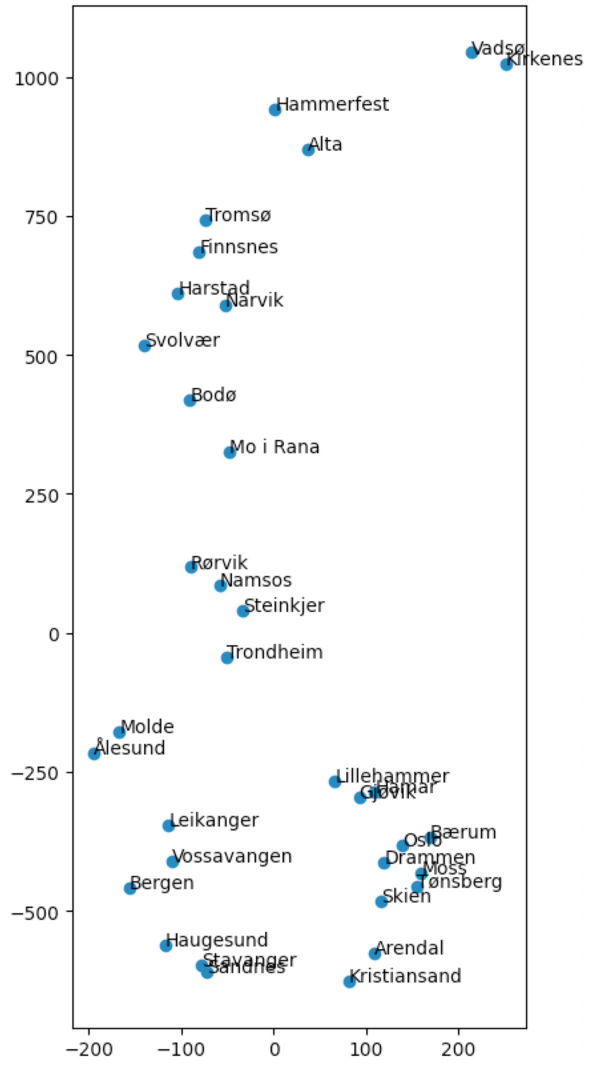
1. ***If** samples $\boldsymbol{X}$ are available:*
   Compute the Gram matrix $\mathbf{B} = \boldsymbol{X}\boldsymbol{X}^{T}$

   ***else if** only pairwise distances $\boldsymbol{D}$ are available:*
   Compute the Gram matrix $\mathbf{B} = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}\boldsymbol{C}^{T}$
2. Compute eigenvectors and eigenvalues of $\mathbf{B}$
3. Sort eigenvalues and eigenvectors
4. Construct a matrix $\boldsymbol{\Lambda}_d$ with the d largest eigenvalues on the diagonal
5. Construct a matrix $\boldsymbol{E}_d^T$ where the columns are the eigenvectors corresponding to the d largest eigenvalues
6. Obtain $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^N$ as the rows of $\mathbf{Y} = \boldsymbol{E}_d^T \boldsymbol{\Lambda}_d^{1/2}$



example: city distances





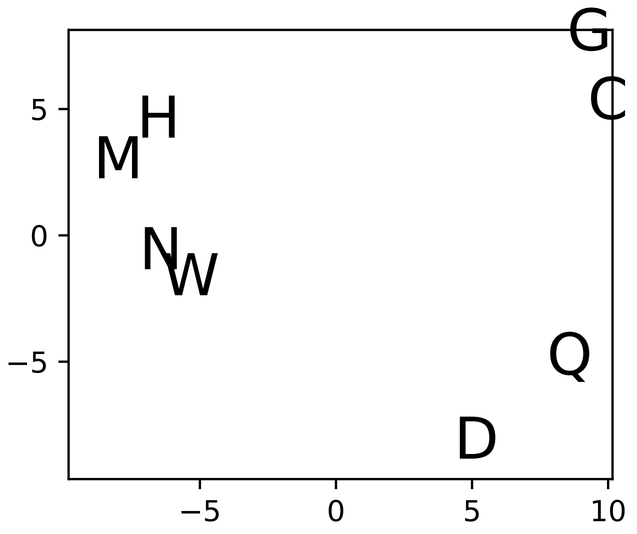
example: letter recognition

Letter	C	D	G	H	M	N	Q	W
C	–							
D	5	–						
G	12	2	–					
H	2	4	3	–				
M	2	3	2	19	–			
N	2	4	1	18	16	–		
Q	9	20	9	1	2	8	–	
W	1	5	2	5	18	13	4	–

Number of times "G" and "C" were confused

…these numbers represent similarities

$$d_{ij} = \begin{cases} c - sim_{ij}, & if\ i \neq j \\ 0, & if\ i = j \end{cases}$$



MDS can work with pairwise distances – they do **not** have to be metrics! Can be (dis-)similiarities!

Drawback of feature extraction

The new features are some combination of the original features, but what do they represent?

Depending on application, interpretability can be important!

$$\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix}Y_1\\Y_2\\\vdots\\Y_d\end{bmatrix} = f\left(\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix}\right)$$

Feature selection: Selects a subset of the original features.

the features still have an interpretation!

$$\begin{bmatrix}X_1\\X_2\\\vdots\\X_n\end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix}X_q\\X_r\\\vdots\\X_d\end{bmatrix}$$

Feature types

1. Relevant features: important for the predictive task — weight [kg], blood pressure, age, ...

2. Irrelevant features: unimportant for the predictive task — favorite movie

3. Redundant features: become irrelevant in the presence of other features — weight [g]

Feature selection algorithms aim to find the relevant features!

Dimensionality reduction methods

Feature extraction

- Principal component analysis (PCA)
- Linear discriminant analysis (LDA)
- Multidimensional scaling (MDS)
- t-distributed stochastic neighborhood embedding (t-SNE)
- Uniform manifold approximation and projection (UMAP)
- …

Feature selection

- Forward selection
- Backward elimination
- Variance threshold
- L1 regularization
- …

Forward selection

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \longrightarrow \begin{bmatrix} X_q \\ X_r \\ \vdots \\ X_d \end{bmatrix}$$

is a supervised feature selection method that aims to find the ***best subset of features***.

Best subset: the least number of features that most contribute to accuracy

forward selection is done in the context of a model and a prediction task!

Forward selection: algorithm

n number of features: $x^i_1, x^i_2, \ldots, x^i_n$

Aim: Given a model f and a labeled dataset $\{\boldsymbol{x}^i, y^i\}_{i=1}^N$, where $\boldsymbol{x}^i \in \mathbb{R}^n$, select the set of features $\mathcal{F}$ that "work best" together.

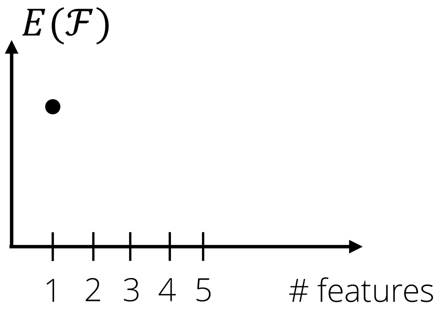
0. Shuffle the data and split into training, validation, and test
1. Init: $\mathcal{F} = \emptyset$ starting with an empty set
2. Repeat:
   - for each feature $x_i \notin \mathcal{F}$, train f on the set of features $\mathcal{F} \cup x_i$ (training data)
   - find the feature yielding the lowest validation error: $j = \arg\min_i E(\mathcal{F} \cup x_i)$
   - add x_j to $\mathcal{F}$ if $E(\mathcal{F} \cup x_j) < E(\mathcal{F})$

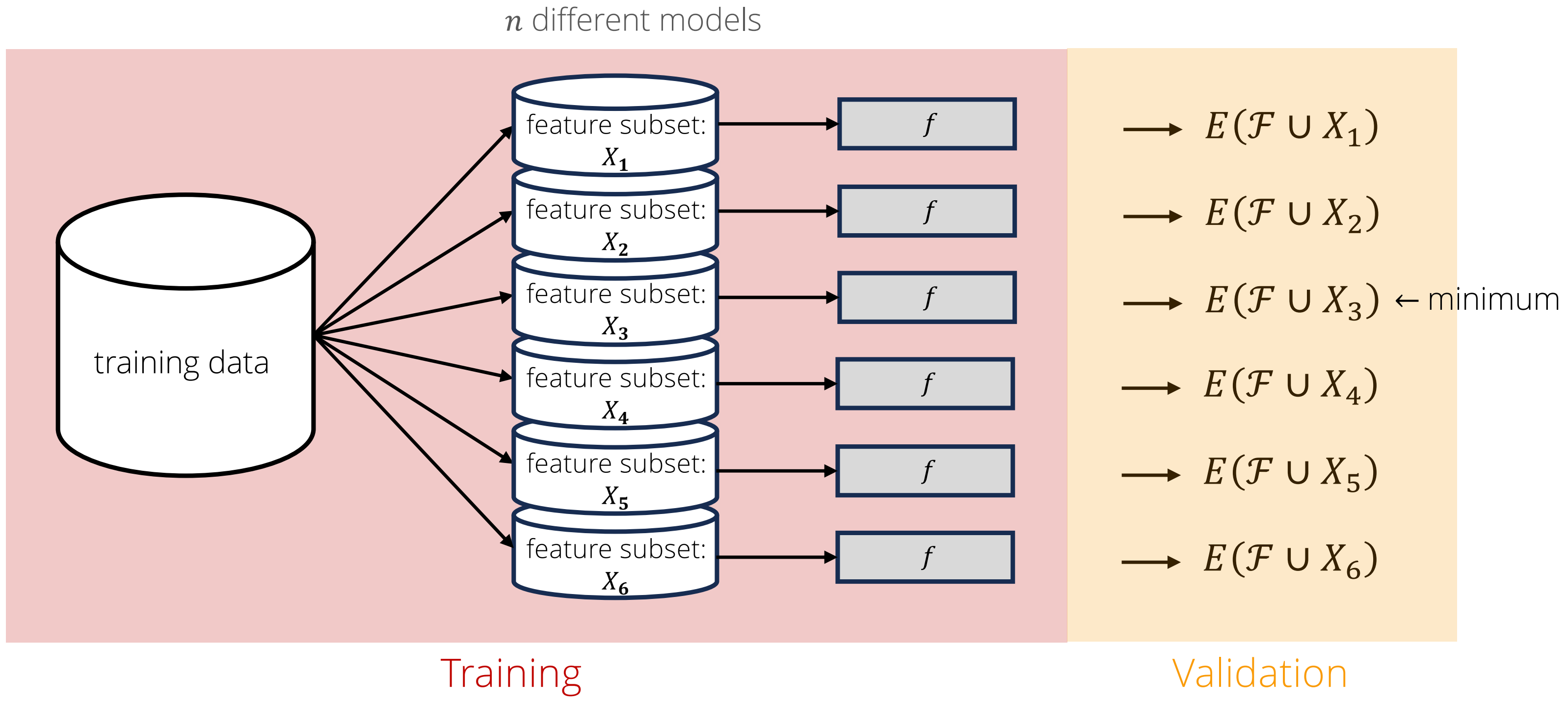
   Until: $E(\mathcal{F} \cup x_j) \geq E(\mathcal{F})$ stop when adding a feature does not decrease the validation error

Forward selection: example

$$\boldsymbol{X} = \begin{array}{l} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} \end{array} \begin{array}{l} \text{age} \\ \text{hair color} \\ \text{glucose level} \\ \text{BMI} \\ \text{height} \\ \text{weight} \end{array}$$

$\mathcal{F} = \{X_3\}$

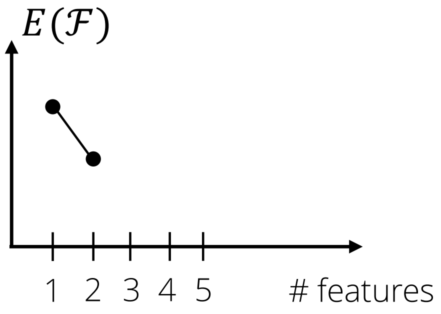


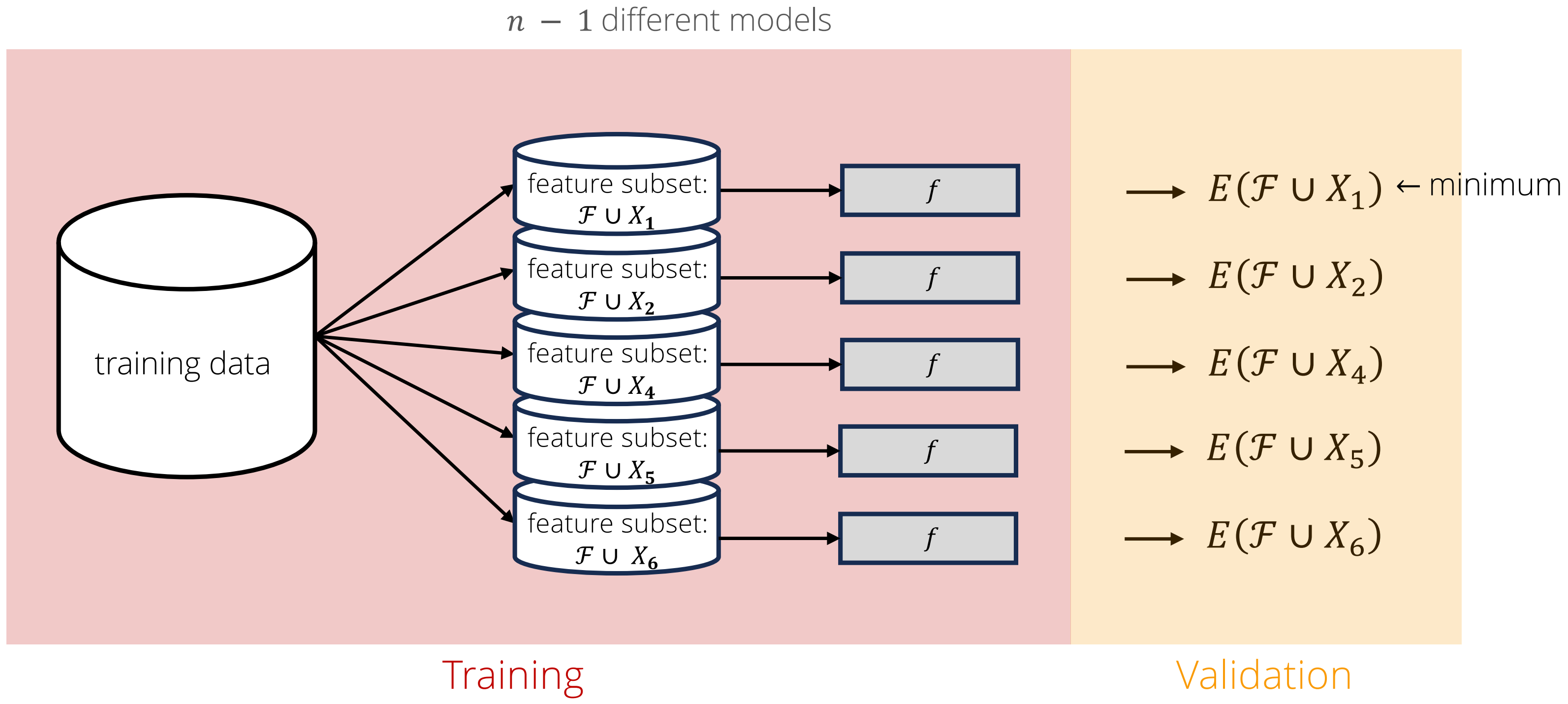


Forward selection: example

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} \begin{array}{l} \text{age} \\ \text{hair color} \\ \text{glucose level} \\ \text{BMI} \\ \text{height} \\ \text{weight} \end{array}$$

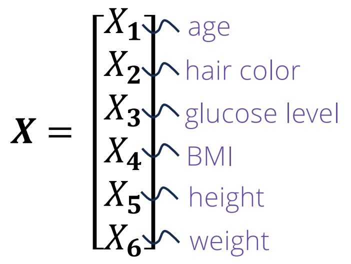
$\mathcal{F} = \{X_3, X_1\}$



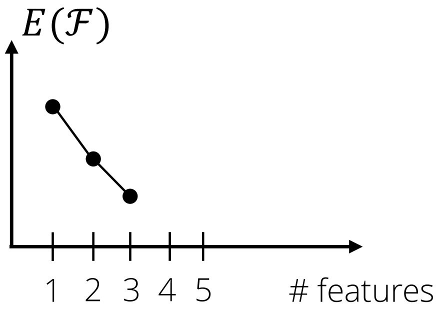


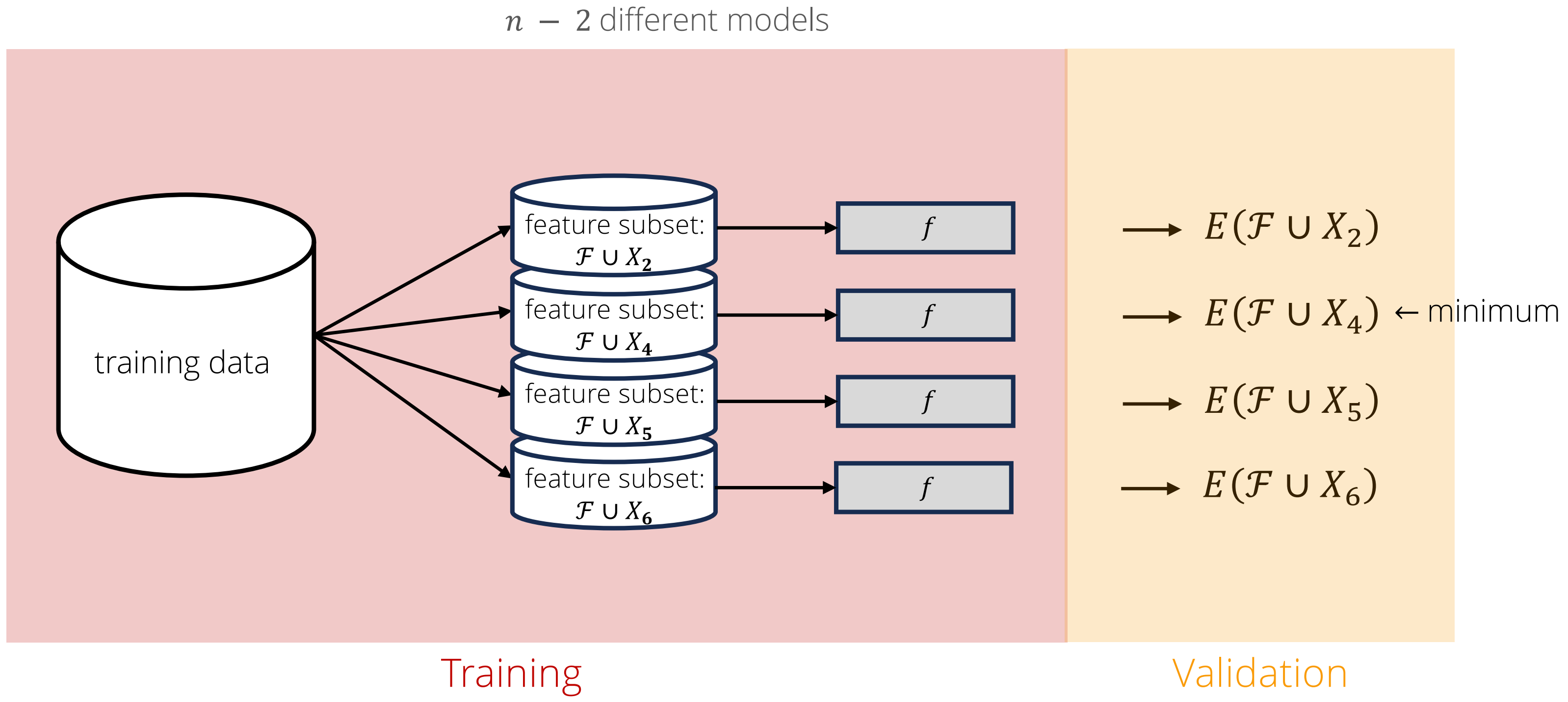
$n - 1$ different models
training data
feature subset: $\mathcal{F} \cup X_1$
feature subset: $\mathcal{F} \cup X_2$
feature subset: $\mathcal{F} \cup X_4$
feature subset: $\mathcal{F} \cup X_5$
feature subset: $\mathcal{F} \cup X_6$
f
f
f
f
f
$E(\mathcal{F} \cup X_1)$ ← minimum
$E(\mathcal{F} \cup X_2)$
$E(\mathcal{F} \cup X_4)$
$E(\mathcal{F} \cup X_5)$
$E(\mathcal{F} \cup X_6)$
Training
Validation

Forward selection: example

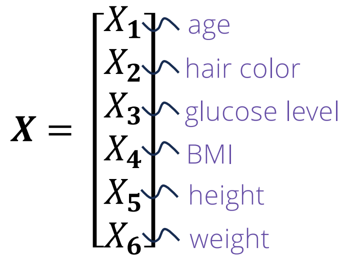

$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix}$
age
hair color
glucose level
BMI
height
weight

$\mathcal{F} = \{X_3, X_1, X_4\}$

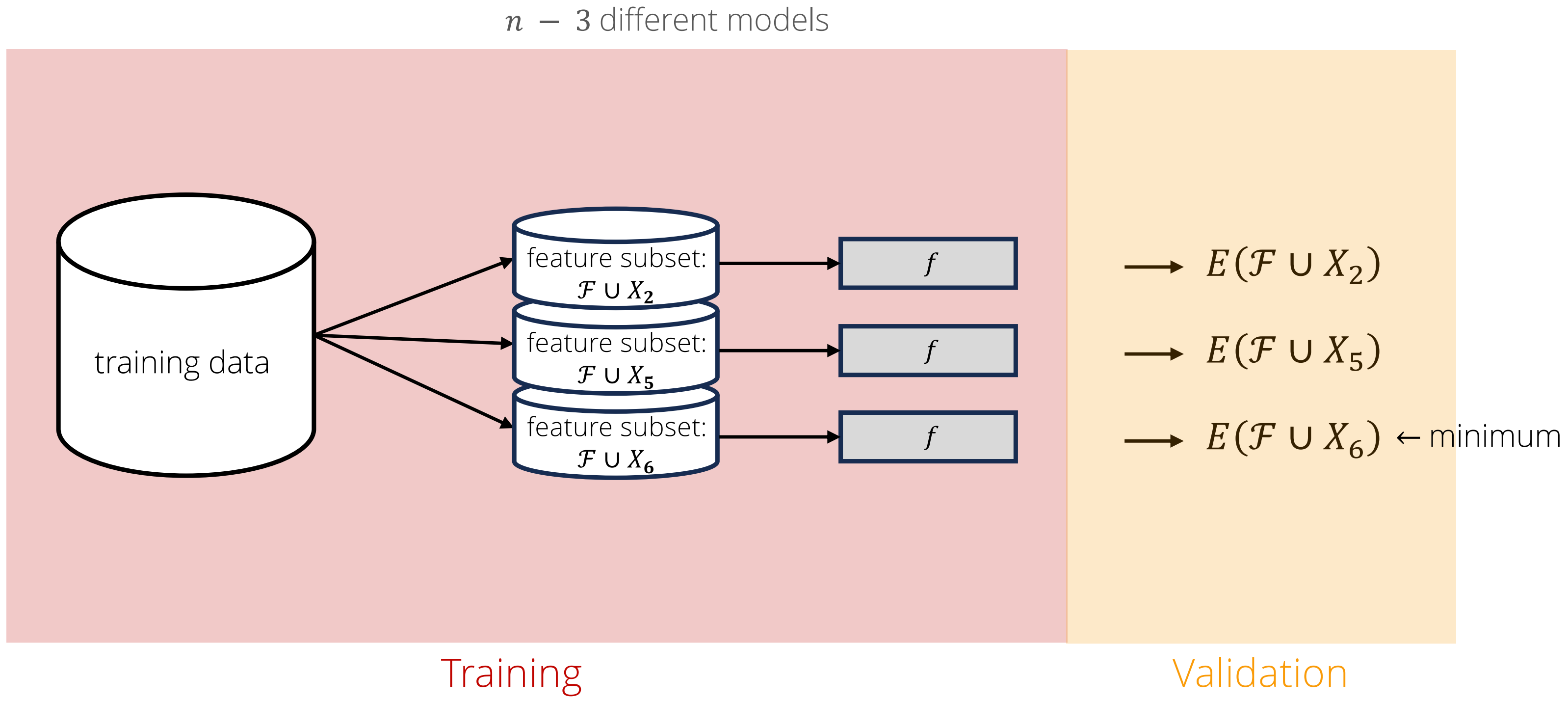



$n - 2$ different models
training data
feature subset: $\mathcal{F} \cup X_2$
feature subset: $\mathcal{F} \cup X_4$
feature subset: $\mathcal{F} \cup X_5$
feature subset: $\mathcal{F} \cup X_6$
f
f
f
f
$\longrightarrow E(\mathcal{F} \cup X_2)$
$\longrightarrow E(\mathcal{F} \cup X_4) \leftarrow$ minimum
$\longrightarrow E(\mathcal{F} \cup X_5)$
$\longrightarrow E(\mathcal{F} \cup X_6)$
Training
Validation

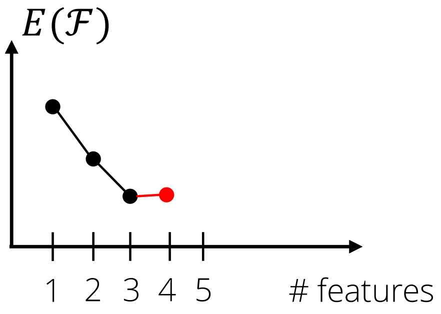
Forward selection: example



$\mathcal{F} = \{X_3, X_1, X_4\}$



$n - 3$ different models
training data
feature subset: $\mathcal{F} \cup X_2$
feature subset: $\mathcal{F} \cup X_5$
feature subset: $\mathcal{F} \cup X_6$
f
f
f
$E(\mathcal{F} \cup X_2)$
$E(\mathcal{F} \cup X_5)$
$E(\mathcal{F} \cup X_6)$ ← minimum
Training
Validation



Forward selection: remarks

- To go from n to d features, we need to train $n + (n-1) + (n-2) + \ldots + (n-d)$ models, which might become costly
- The selected features depend heavily on the model and the prediction task
- The selected features might depend on the train/validation split

improve robustness
through cross validation!

- Finding the optimal subset is **not** guaranteed (context/dependencies)

Conclusion

- Visited two methods for dimensionality reduction:
  - Unsupervised feature **extraction** method: Multidimensional scaling
  - Supervised feature **selection** method: Forward selection
- MDS from high dimensional data (samples)
- MDS from pairwise distances
- Forward selection (in the context of a model)
- Choice of method depends on the data and the problem setting