Lecture 12: Data representations (cont'd)

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

9th May 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Administrative

Projects

- Great job with the projects so far!
- Remember that 7.5 credits means you signed up for a part-time job (20 hours per week)
  - 6 hours lectures/presentations
  - 14 hours for projects and reiteration of lecture material
- Keep in mind:
  - The projects are meant to be open and challenging to allow for multiple angles on the same topic
  - There is help if you need/want it: Office hours, mails, lecture pauses, ...
  - Tell me beforehand if you are going to miss a project presentation

Exam

- Exam will be distributed on 24th May **before the end of the course**. Why? So you have an easier time getting a hold of us for questions.
- You have three weeks for the exam, but it is not meant to take three weeks to write it
- **Hard deadline:** 14th June
- Two parts:
  1. Project reports: Determine pass or fail, i. e. 3 at Chalmers, G at GU
  2. Extra exercises to get higher grades, i. e. 4 or 5 at Chalmers, VG at GU
- You need to pass the project reports to pass the course. Extra exercises are optional but determine if you get a higher grade.

Project reports

- Project reports have to include the following
  1. If you realise during or after the presentations that you did some part wrong, you have to fix it for the exam
  2. Summarise the main take home messages of your topic (strengths/limitations of a method or challenges/solutions of a problem)
  3. Write conclusions/concrete thoughts on future work where you give a specific description of a path forward (i.e. what could you do, why, and what are expected results)
- Formalities
  - Reproduction of the presentation slides is not enough
  - We want to see that you gained understanding
  - One A4 page text per project
  - Figures/tables separate, but don't overdo it. Include what is meaningful

Back to NMF and dimension reduction

Recap: Best SVD approximation

Assume $\mathbf{X} \in \mathbb{R}^{p \times n}$. The **SVD** of $\mathbf{X}$ is

$$\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^T,$$

where $\mathbf{V} \in \mathbb{R}^{p \times p}$, $\mathbf{D} \in \mathbb{R}^{p \times p}$ and diagonal, and $\mathbf{U} \in \mathbb{R}^{n \times p}$.

Using only the first $q < \min(p, n)$ columns of $\mathbf{V}$ and $\mathbf{U}$, and the first q rows and columns of $\mathbf{D}$, leads to

$$\mathbf{X}_q = \mathbf{V}_q \mathbf{D}_q \mathbf{U}_q^T,$$

the **best rank-q-approximation** of $\mathbf{X}$.

This approximation is best in terms of the **Frobenius norm**, i.e.

$$\mathbf{X}_q = \underset{\text{rank}(\mathbf{M})=q}{\arg\min} \|\mathbf{X} - \mathbf{M}\|_F^2 = \sum_{i=1}^{p} \sum_{l=1}^{n} (X_{il} - M_{il})^2$$

Recap: Non-negative matrix factorization (NMF)

A **non-negative matrix factorisation** of $\mathbf{X}$ with rank q solves

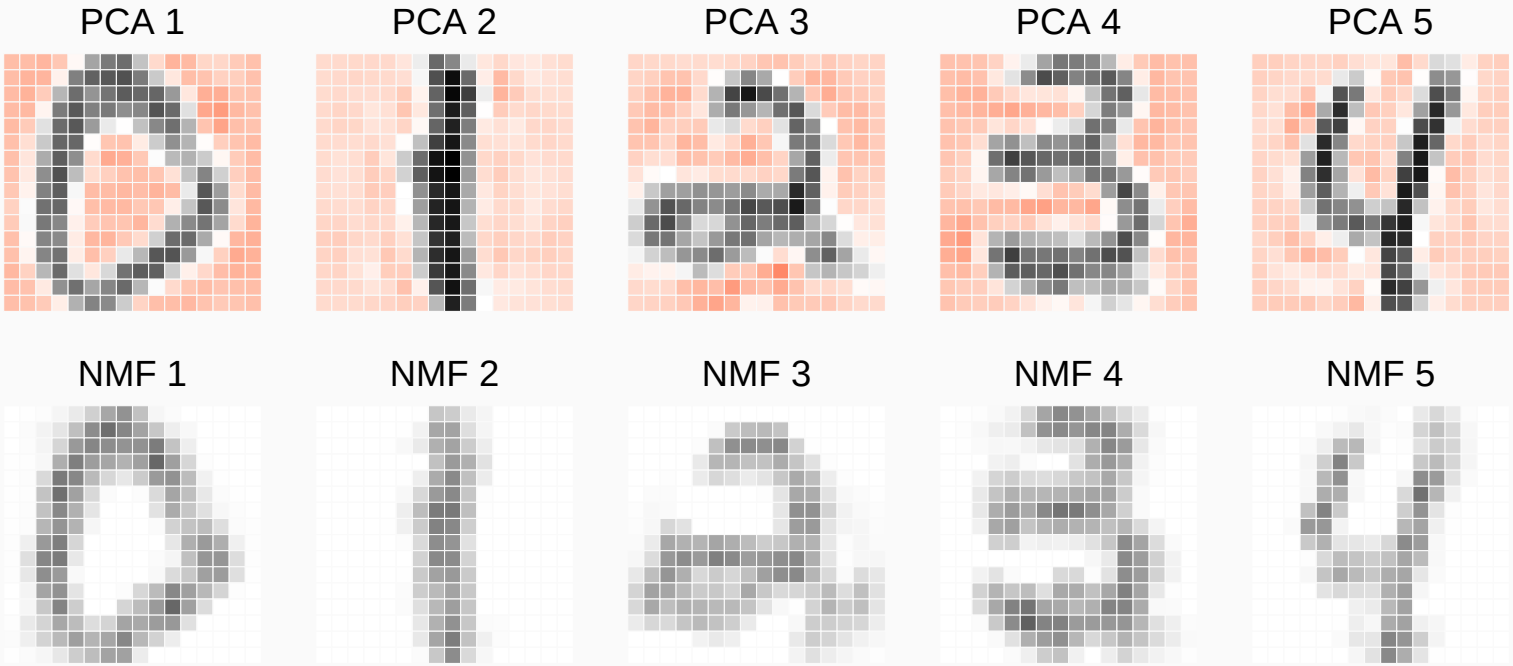
$$\underset{\mathbf{W}\in\mathbb{R}^{p\times q},\mathbf{H}\in\mathbb{R}^{q\times n}}{\arg\min} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 \quad \text{such that} \quad \mathbf{W}\geq 0, \mathbf{H}\geq 0$$

- **Sum of positive layers:** $\mathbf{X}\approx\sum_{j=1}^{q}\mathbf{W}_{\cdot j}\mathbf{H}_{\cdot j}^T$
- Non-negativity constraint leads to **sparsity in basis** (in $\mathbf{W}$) and **coefficients** (in $\mathbf{H}$) [example on next slides]
- NP-hard problem, i.e. no general algorithm exists

SVD vs NMF – Example: Reconstruction

MNIST-derived zip code digits ($n = 1000$, $p = 256$)

100 samples are drawn randomly from each class to keep the problem balanced.

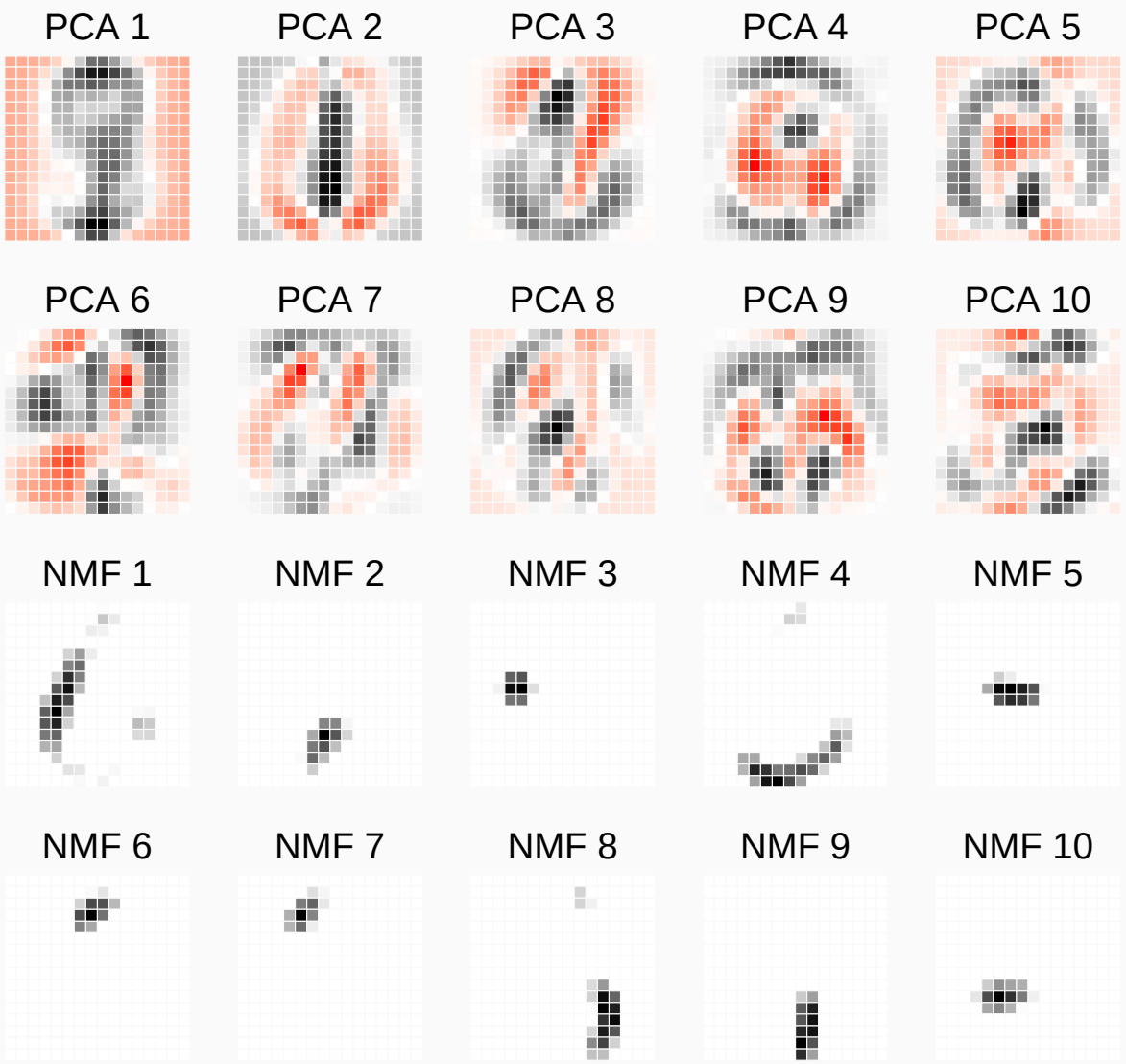


Red-ish colours are for negative values, white is around zero and dark stands for positive values

SVD vs NMF – Example: Basis Components

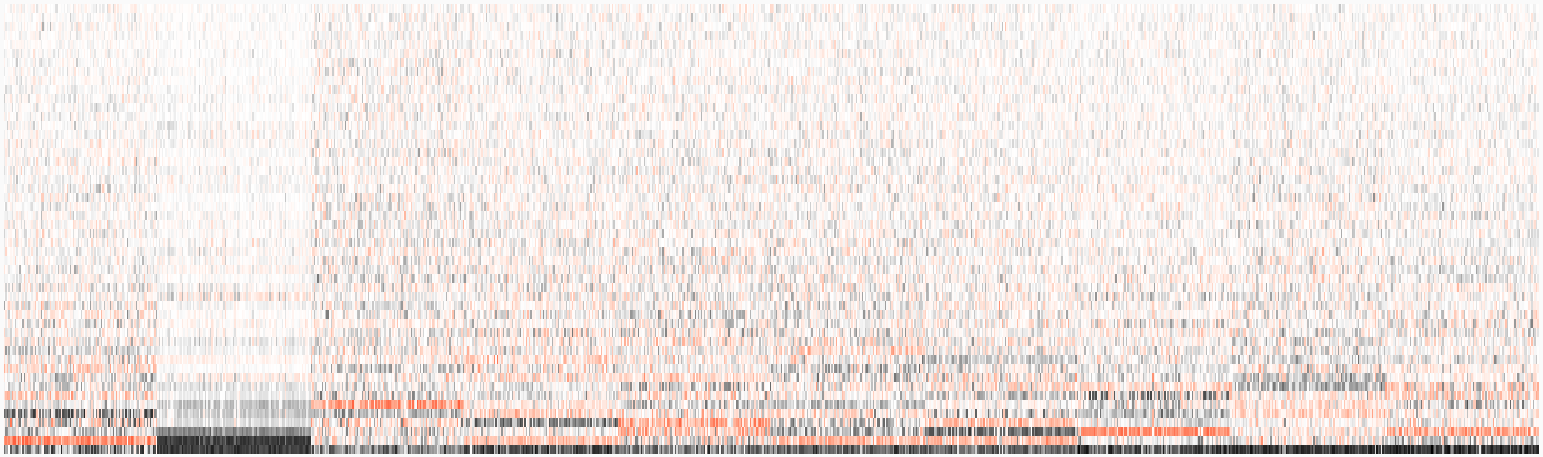
Large difference between SVD/PCA and NMF basis components

NMF captures **sparse characteristic parts** while PCA components capture more global features.

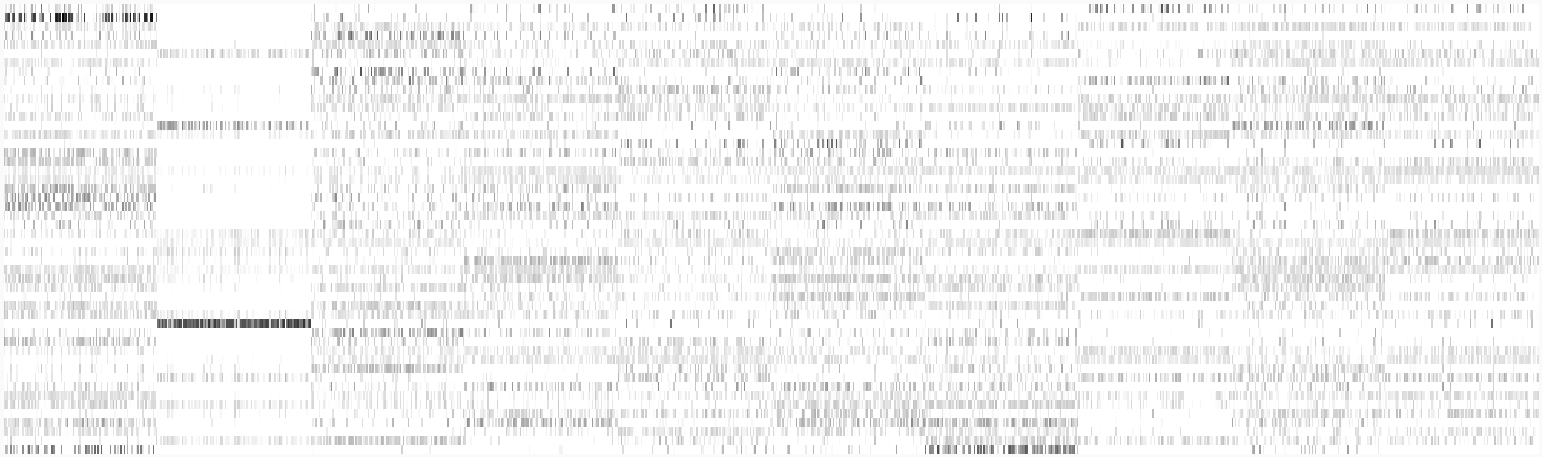


SVD vs NMF – Example: Coefficients

SVD coefficients



NMF coefficients



Note the additional **sparsity** in the NMF coefficients.

How to solve the NMF problem?

The NMF problem we considered was

$$\underset{\mathbf{W}\in\mathbb{R}^{p\times q},\mathbf{H}\in\mathbb{R}^{q\times n}}{\arg\min} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 \quad \text{such that} \quad \mathbf{W}\geq 0, \mathbf{H}\geq 0$$

Most algorithms use **two-block coordinate descent** and solve

$$\mathbf{W}^{(t)} = \underset{\mathbf{W}\geq 0}{\arg\min} \|\mathbf{X}-\mathbf{W}\mathbf{H}^{(t-1)}\|_F^2 \quad \text{and} \quad \mathbf{H}^{(t)} = \underset{\mathbf{H}\geq 0}{\arg\min} \|\mathbf{X}-\mathbf{W}^{(t)}\mathbf{H}\|_F^2$$

iteratively. It holds that

$$\begin{aligned}\|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 &= \sum_{i=1}^{p} \|\mathbf{X}_{i\cdot}-\mathbf{W}_{i\cdot}\mathbf{H}\|_2^2 \\ &= \sum_{i=1}^{p} \mathbf{W}_{i\cdot}(\mathbf{H}\mathbf{H}^T)\mathbf{W}_{i\cdot}^T - 2\mathbf{W}_{i\cdot}(\mathbf{H}\mathbf{X}_{i\cdot})^T + \|\mathbf{X}_{i\cdot}\|_2^2\end{aligned}$$

Therefore, the NMF problem in $\mathbf{W}$ (given $\mathbf{H}$) can be seen as p **independent non-negative least squares (NNLS)** problems.

Some notes on solving the NMF problem

- The problem is **symmetric in $\mathbf{W}$ and $\mathbf{H}$** since

$$||\mathbf{X} - \mathbf{WH}||_F^2 = ||\mathbf{X}^T - \mathbf{H}^T\mathbf{W}^T||_F^2$$

  No separate algorithms needed for $\mathbf{W}$ and $\mathbf{H}$.
- Algorithms have two crucial parts:
  1. **Initialisation** of $\mathbf{W}^{(0)}$ and $\mathbf{H}^{(0)}$
  2. **Update rule** for $\mathbf{W}^{(t)}$ given $\mathbf{H}^{(t-1)}$ and $\mathbf{H}^{(t)}$ given $\mathbf{W}^{(t)}$
- Other cost functions are possible.
  - **Note:** Cost functions determine the distribution of noise
  - Frobenius norm implies Gaussian distribution
  - An alternative for Poisson distributed data

$$D(\mathbf{X}||\mathbf{WH}) = \sum_{i=1}^{p}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{(\mathbf{WH})_{ij}} - X_{ij} + (\mathbf{WH})_{ij}\right)$$

  Resembles the **Kullback-Leibler divergence** and the **log-likelihood of Poisson-distributed data** with mean $(\mathbf{WH})_{ij}$ for X_{ij}.

Least squares updates for NMF

When using the Frobenius norm as a cost function, one possible update rule is **alternating least squares (ALS)**: Solve the unconstrained least squares problem

$$\mathbf{Z}^{(t)} = \underset{\mathbf{Z} \in \mathbb{R}^{p \times q}}{\arg\min} \|\mathbf{X} - \mathbf{Z}\mathbf{H}^{(t-1)}\|_F^2$$

and set elementwise $\mathbf{W}^{(t)} = \max(\mathbf{Z}^{(t)}, 0)$. Analogous for $\mathbf{H}^{(t)}$.

- The method is cheap but can have convergence issues.
- Can be useful for initialisation (some steps of ALS first, then another algorithm)
- **Alternating non-negative least squares (ANNLS)** is an alternative that solves the constrained least squares problem exactly. More stable, but much slower.

Multiplicative updates for NMF

Multiplicative updates (MU) have been popularized by Lee and Seung (1999). Their form depends on the cost function. In the following $\mathbf{A} \circ \mathbf{B}$ denotes elementwise multiplication of matrices and division is also meant elementwise.

1. Frobenius norm:

$$\mathbf{W} \leftarrow \mathbf{W} \circ \frac{\mathbf{X}\mathbf{H}^T}{\mathbf{W}\mathbf{H}\mathbf{H}^T} \quad \text{and} \quad \mathbf{H} \leftarrow \mathbf{H} \circ \frac{\mathbf{W}^T\mathbf{X}}{\mathbf{W}^T\mathbf{W}\mathbf{H}}$$

2. Divergence:

$$W_{ik} \leftarrow W_{ik} \frac{\sum_{l=1}^{n} H_{kl} X_{il} / (\mathbf{W}\mathbf{H})_{il}}{\sum_{l=1}^{n} H_{kl}} \quad \text{and}$$

$$H_{kl} \leftarrow H_{kl} \frac{\sum_{i=1}^{p} W_{ik} X_{il} / (\mathbf{W}\mathbf{H})_{il}}{\sum_{i=1}^{p} W_{ik}}$$

Multiplicative updates for NMF and gradient descent

Multiplicative updates are a special case of **gradient descent**.
Let $F(\mathbf{W}, \mathbf{H}) = ||\mathbf{X} - \mathbf{WH}||_F^2$ then

$$\nabla_{\mathbf{W}} F = \mathbf{WHH}^T - \mathbf{XH}^T$$
$$\nabla_{\mathbf{H}} F = \mathbf{W}^T\mathbf{WH} - \mathbf{W}^T\mathbf{X}$$

Gradient descent in $\mathbf{W}$ for step-length α performs

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \nabla_{\mathbf{W}} F$$

It can be shown that

$$\alpha = \frac{\mathbf{W}}{\mathbf{WHH}^T}$$

is an admissible step length and yields the MU for $\mathbf{W}$.

Note: Analogous results hold for the divergence.

Initialising NMF

NMF can be initialised in multiple ways

- **Random initialisation:** Entries in $\mathbf{W}$ and $\mathbf{H}$ are uniformly distributed in $[0, 1]$
- **Clustering techniques:** e.g. run k-means with q clusters on data, store cluster centroids in $\mathbf{W}$) and $H_{kl} \neq 0 \Leftrightarrow \mathbf{X}_{\cdot l}$ belongs to cluster k
- **SVD**: Determine best rank-q-approximation $\sum_{i=1}^{q} d_{ii}\mathbf{v}_i\mathbf{u}_i^T$, note that

$$\begin{aligned} d_{ii}\mathbf{u}_i\mathbf{v}_i^T = &([+d_{ii}\mathbf{u}_i]_+[+\mathbf{v}_i^T]_+ + [-d_{ii}\mathbf{u}_i]_+[-\mathbf{v}_i^T]_+) \\ &- ([+d_{ii}\mathbf{u}_i]_+[-\mathbf{v}_i^T]_+) + [-d_{ii}\mathbf{u}_i]_+[+\mathbf{v}_i^T]_+) \end{aligned}$$

  and initialize NMF by summing only the positive parts or the larger of the positive parts.

Kernel-methods

Kernels

A **kernel** is a function $k(\mathbf{x}, \mathbf{y}) \to \mathbb{R}$ that maps two elements of the feature space to a real number, such that

$$k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x}) \quad \text{and} \quad k(\mathbf{x}, \mathbf{y}) \geq 0$$

Can be seen as a (possibly non-linear) **generalized inner product** without bilinearity.

Note: This is similar but not exactly the same as the kernels used for **kernel density estimation**. There, $k(\mathbf{x}) \in \mathbb{R}$ with

$$k(\mathbf{x}) \geq 0, \quad \int k(\mathbf{x}) \, \mathrm{d}\mathbf{x} = 1,$$

$$k((x_1, \dots, x_p)^T) = k((-x_1, \dots, x_p)^T) = \dots = k((x_1, \dots, -x_p)^T)$$

$$k((x_1, \dots, x_p)^T) = k((x_{\sigma(1)}, \dots, x_{\sigma(p)})^T)$$

for any permutation σ of $\{1, \dots, p\}$.

Examples of kernels

- **Linear kernel** $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$
- **Polynomial kernel** $k(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^T \mathbf{y} + r)^m$
- **Radial basis function (RBF) kernel**
  $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma^2}\right)$
- **Laplacian kernel** $k(\mathbf{x}, \mathbf{y}) = \exp(-\alpha \|\mathbf{x} - \mathbf{y}\|_2^2)$

Note: There are kernels on more general spaces than $\mathbb{R}^p$, e.g. on strings.

Mercer/positive definite kernels

For a kernel $k(\mathbf{x}, \mathbf{y})$, and a set of features $\mathbf{x}_1, \ldots, \mathbf{x}_n$ define the so-called **Gram matrix**

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

If $\mathbf{K}$ is positive-definite for any n and all sets of features, then $k(\mathbf{x}, \mathbf{y})$ is called a **Mercer** or **positive definite kernel**.

Note: All kernels shown before are positive definite.

Importance of positive definite kernels

If the gram matrix is positive definite there is an orthogonal matrix $\mathbf{U} \in \mathbb{R}^{n \times n}$ and a diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{K} = \mathbf{U}^T \mathbf{\Lambda} \mathbf{U}.$$

Define $\boldsymbol{\phi}(\mathbf{x}_l) = \mathbf{\Lambda}^{1/2} \mathbf{U}_{.l}$, then

$$K_{lk} = \boldsymbol{\phi}(\mathbf{x}_l)^T \boldsymbol{\phi}(\mathbf{x}_k)$$

A result known as **Mercer's theorem** ensures that **for every positive definite kernel** $k(\mathbf{x}, \mathbf{y})$ there is a mapping $\boldsymbol{\phi}$ from the feature space to $\mathbb{R}^p$ (with $p = \infty$ allowed) such that

$$k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})$$

Example of Mercer's theorem

Consider the polynomial kernel for $\gamma = r = 1$ and $m = 2$ in a two-dimensional feature space

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^T \mathbf{y} + 1)^2 = (1 + x_1 y_1 + x_2 y_2)^2 \\ &= 1 + 2x_1 y_1 + 2x_2 y_2 + (x_1 y_1)^2 + (x_2 y_2)^2 + 2x_1 y_1 x_2 y_2 \end{aligned}$$

Define

$$\boldsymbol{\phi}(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)^T$$

then

$$k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})$$

Using this kernel is therefore equivalent to working in a **six-dimensional** feature space.

Advantages of using kernels

Summary

Using a positive definite kernel to measure the similarity between m-dimensional feature vectors is equivalent to

1. Using a (potentially non-linear) mapping to transform the feature vectors to $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^p$ with $p \geq m$
2. Using the Euclidean scalar product to measure similarity between transformed feature vectors $\boldsymbol{\phi}(\mathbf{x})$

The **kernel-trick** is to implicitly work in the higher-dimensional space of the $\boldsymbol{\phi}(\mathbf{x})$, but to only evaluate kernels in the original feature space and therefore avoid transformations to a high-dimensional space.

Recap: PCA

Recall: In PCA, the goal was to find the directions of maximum variance of the data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ by decomposing the covariance matrix (with $\mathbf{V} \in \mathbb{R}^{p \times p}$)

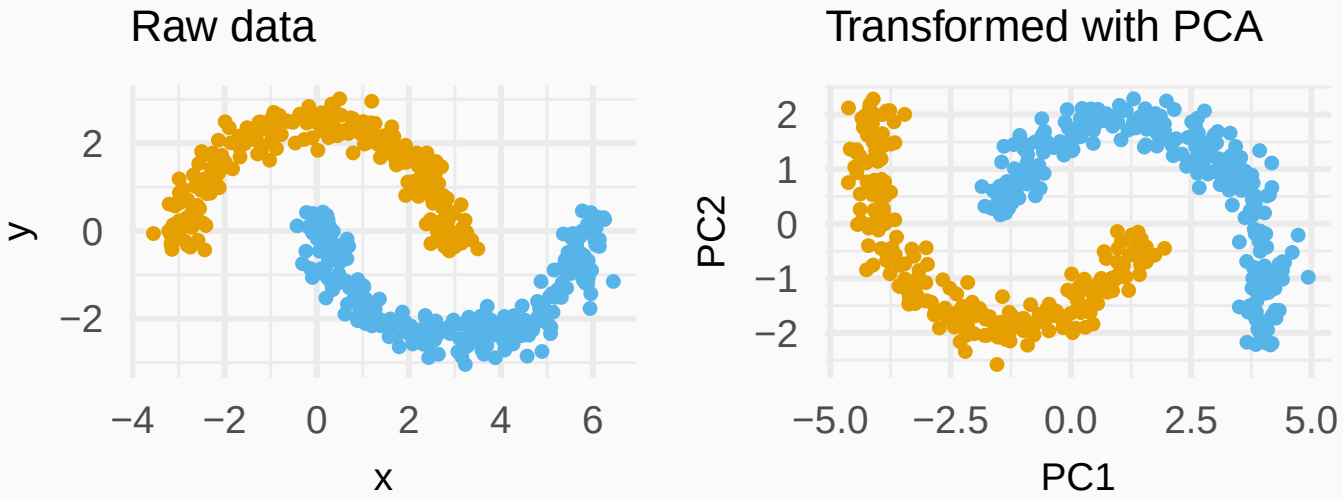
$$\mathbf{\Sigma} = \frac{\mathbf{X}^T \mathbf{X}}{n} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$$

Goals are

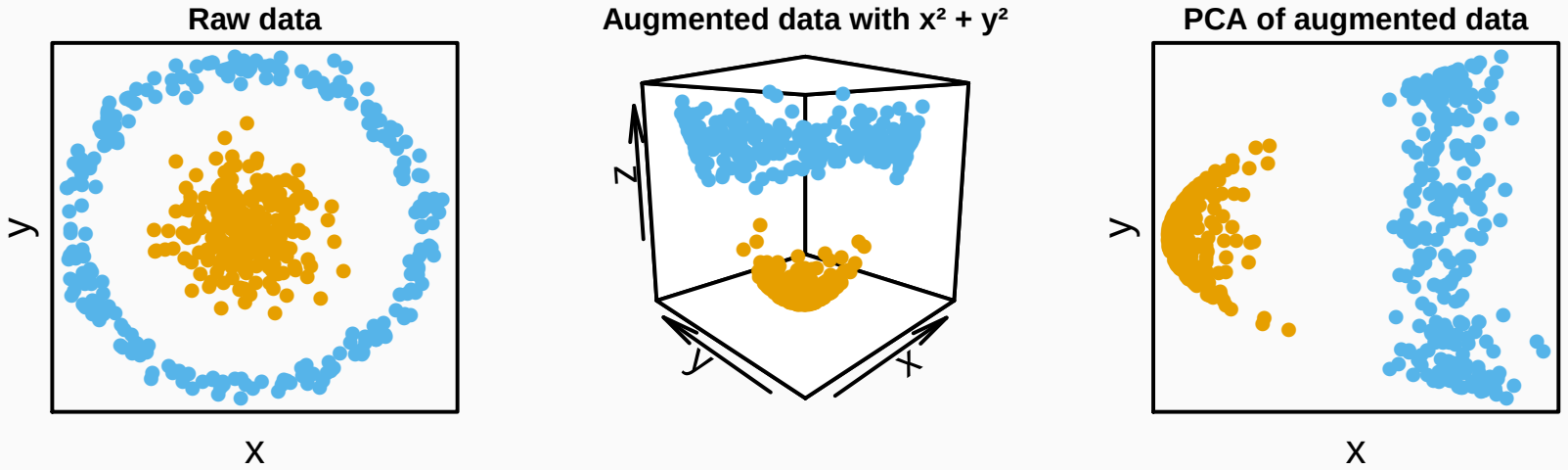
- Dimension-reduction (e.g. for visualisation)
- Finding important directions in the data relevant to e.g. classification or clustering

Limitations of PCA

PCA is **linear** and cannot uncover **non-linear structures**



Augmentation of features can help



Kernels and PCA (I)

Idea: Use the **kernel-trick** to define augmentations implicitly and keep computations manageable.

Given a positive definite kernel $k(\mathbf{x}, \mathbf{y})$, how can we perform PCA in the space of $\boldsymbol{\phi}(\mathbf{x})$?

Assume we have access to $\boldsymbol{\phi}(\mathbf{x}_l)$ for $l = 1, \dots, n$ and they are centred. Then we can perform PCA

$$\boldsymbol{\Sigma}^{\boldsymbol{\phi}} = \frac{1}{n} \sum_{l=1}^{n} \boldsymbol{\phi}(\mathbf{x}_l) \boldsymbol{\phi}(\mathbf{x}_l)^T = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T$$

where $\mathbf{u}_i$ are the principal component axes and λ_i the corresponding variances.

Kernels and PCA (II)

Note that

$$\boldsymbol{\Sigma}^{\boldsymbol{\phi}}\mathbf{u}_i = \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\phi}(\mathbf{x}_l)\boldsymbol{\phi}(\mathbf{x}_l)^T\mathbf{u}_i = \lambda_i\mathbf{u}_i$$

$$\Leftrightarrow \quad \mathbf{u}_i = \sum_{l=1}^{n}\frac{\boldsymbol{\phi}(\mathbf{x}_l)^T\mathbf{u}_i}{\lambda_i n}\boldsymbol{\phi}(\mathbf{x}_l) = \sum_{l=1}^{n} a_{il}\boldsymbol{\phi}(\mathbf{x}_l)$$

Using this representation of $\mathbf{u}_i$ in $\boldsymbol{\phi}(\mathbf{x}_k)^T\boldsymbol{\Sigma}^{\boldsymbol{\phi}}\mathbf{u}_i = \lambda_i\boldsymbol{\phi}(\mathbf{x}_k)^T\mathbf{u}_i$ leads to

$$\frac{1}{n}\sum_{l=1}^{n}\underbrace{\boldsymbol{\phi}(\mathbf{x}_k)^T\boldsymbol{\phi}(\mathbf{x}_l)}_{=k(\mathbf{x}_k,\mathbf{x}_l)}\sum_{j=1}^{n}a_{ij}\underbrace{\boldsymbol{\phi}(\mathbf{x}_l)^T\boldsymbol{\phi}(\mathbf{x}_j)}_{=k(\mathbf{x}_l,\mathbf{x}_j)} = \lambda_i\sum_{l=1}^{n}a_{il}\underbrace{\boldsymbol{\phi}(\mathbf{x}_k)^T\boldsymbol{\phi}(\mathbf{x}_l)}_{=k(\mathbf{x}_k,\mathbf{x}_l)}$$

Set $\mathbf{a}_i = (a_{ij})_j$ and use the Gram matrix $\mathbf{K} = (k(\mathbf{x}_i,\mathbf{x}_j))_{ij}$, then

$$\mathbf{K}^2\mathbf{a}_i = \lambda_i n\mathbf{K}\mathbf{a}_i \quad \Leftrightarrow \quad \mathbf{K}\mathbf{a}_i = \lambda_i n\mathbf{a}_i$$

Kernels and PCA (III)

The coefficients a_{ij} to determine the principal component directions $\mathbf{u}_i$ in the high-dim. space of the $\boldsymbol{\phi}(\mathbf{x}_i)$ can therefore be found by

- Solving the eigenvalue problem $\mathbf{K}\mathbf{a}_i = \lambda_i n \mathbf{a}_i$
- Requiring that

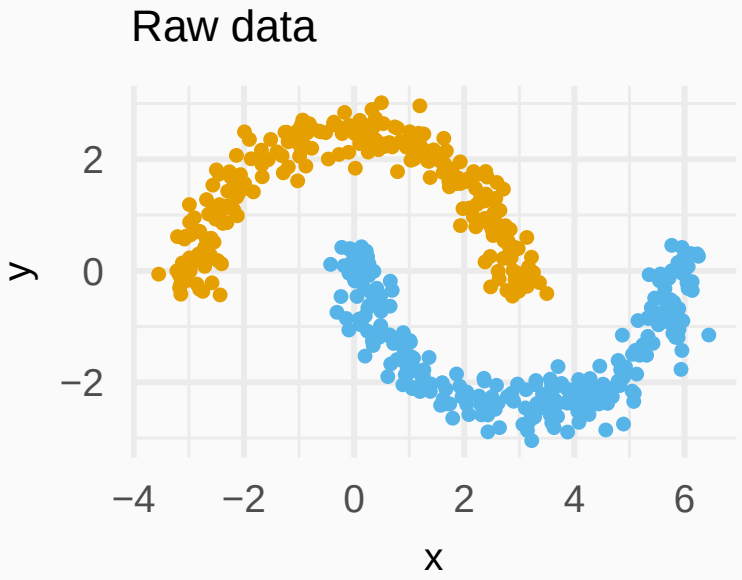
$$1 = \mathbf{u}_i^T \mathbf{u}_i = \sum_{l,k=1}^{n} a_{il} a_{ik} \boldsymbol{\phi}(\mathbf{x}_l)^T \boldsymbol{\phi}(\mathbf{x}_k) = \mathbf{a}_i^T \mathbf{K} \mathbf{a}_i$$

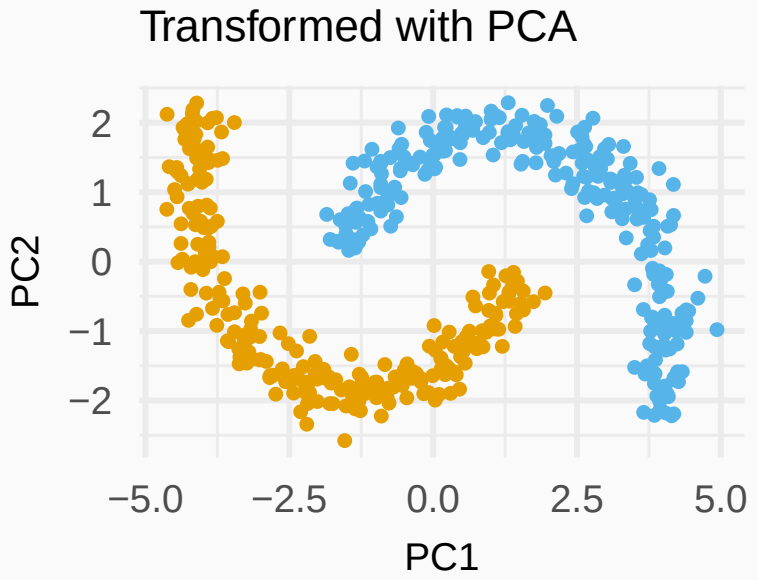
The i-th principal component projection of an arbitrary mapped feature vector $\boldsymbol{\phi}(\mathbf{x})$ is therefore

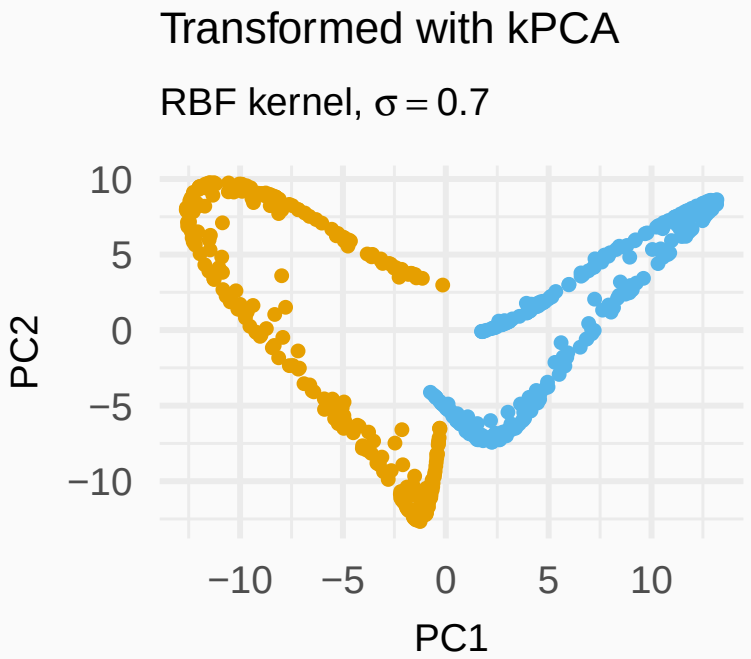
$$\boldsymbol{\phi}(\mathbf{x})^T \mathbf{u}_i = \sum_{l=1}^{n} a_{il} k(\mathbf{x}, \mathbf{x}_l)$$

This procedure is called **kernel-PCA (kPCA)**.

Example: kPCA







Take-home message

- NMF is a powerful matrix factorisation technique offering both sparsity and interpretability
- Kernels in combination with Mercer's theorem are a powerful tool to make high-dimensional computation manageable
- kPCA is a first example demonstrating the power of kernels