Lecture 12: Data representations - Kernel methods

Felix Held, Mathematical Sciences

MSA220/MVE441 Statistical Learning for Big Data

10th May 2021

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Kernel-methods

Kernels

A **kernel** is a function $k(\mathbf{x}, \mathbf{y}) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ 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.

Kernels measure **similarity** between features vectors.

Examples of kernels

- **Linear kernel** $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$
- **Polynomial kernel** $k(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^\top \mathbf{y} + r)^m$
- **Radial basis function (RBF) kernel** $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma ||\mathbf{x} - \mathbf{y}||_2^2\right)$
- **Laplacian kernel** $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma ||\mathbf{x} - \mathbf{y}||_1)$
- **Sigmoid kernel** $k(\mathbf{x}, \mathbf{y}) = \tanh(\alpha \mathbf{x}^\top \mathbf{y} + c)$

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

Note: All kernels shown on the last slide except for the sigmoid kernel are positive definite.

Importance of positive definite kernels

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

$$\mathbf{K} = \mathbf{V}^\top \mathbf{\Lambda} \mathbf{V}.$$

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

$$\mathbf{K}^{(l,k)} = \boldsymbol{\phi}(\mathbf{x}_l)^\top \boldsymbol{\phi}(\mathbf{x}_k)$$

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 some q-dimensional space (with $q = \infty$ allowed) such that

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

Example of Mercer's theorem

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

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

Define

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

then

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

Using this kernel to measure similarity between **two-dimensional** feature vectors 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 $\mathbf{x}$ to a q-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$
2. Using the Euclidean scalar product to measure similarity between transformed feature vectors $\boldsymbol{\phi}(\mathbf{x})$

Problem: $\boldsymbol{\phi}(\mathbf{x})$ might be hard to compute.

The **kernel-trick** is to replace scalar products with kernel evaluations. Computations are then done implicitly in the higher-dimensional space of the $\boldsymbol{\phi}(\mathbf{x})$, but all we need to do is evalute the kernel.

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

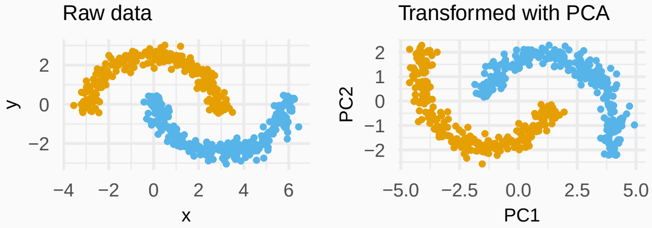
$$\widehat{\boldsymbol{\Sigma}} = \frac{\mathbf{X}^\top \mathbf{X}}{n-1} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$$

where $\mathbf{V} \in \mathbb{R}^{p \times p}$ is orthgonal and $\mathbf{D} \in \mathbb{R}^{p \times p}$ is diagonal. Goals are

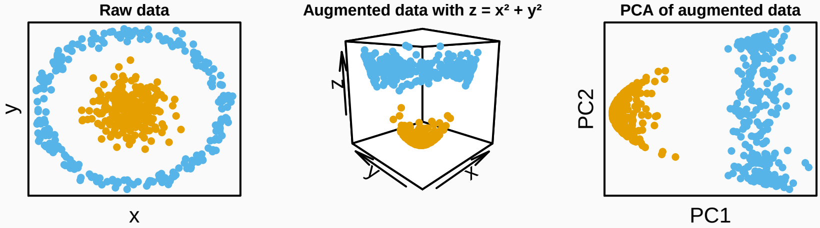
- Dimension-reduction (e.g. for visualisation)
- Finding important directions in the data relevant to e.g. classification or clustering

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 high-dimensional space of $\boldsymbol{\phi}(\mathbf{x})$?

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

$$\widehat{\boldsymbol{\Sigma}}^{\phi} = \frac{1}{n} \sum_{l=1}^{n} \boldsymbol{\phi}(\mathbf{x}_l) \boldsymbol{\phi}(\mathbf{x}_l)^\top = \mathbf{V}\mathbf{D}\mathbf{V}^\top$$

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

Kernels and PCA (II)

Note that

$$\widehat{\Sigma}^{\phi}\mathbf{v}_i = \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\phi}(\mathbf{x}_l)\boldsymbol{\phi}(\mathbf{x}_l)^\top\mathbf{v}_i = d_i\mathbf{v}_i$$

$$\Leftrightarrow \quad \mathbf{v}_i = \sum_{l=1}^{n}\frac{\boldsymbol{\phi}(\mathbf{x}_l)^\top\mathbf{v}_i}{d_i n}\boldsymbol{\phi}(\mathbf{x}_l) = \sum_{l=1}^{n}\mathbf{a}_i^{(l)}\boldsymbol{\phi}(\mathbf{x}_l)$$

Multiplying this presentation of $\mathbf{v}_i$ from the left on both sides with $\boldsymbol{\phi}(\mathbf{x}_k)^\top$ leads to (for all $k = 1, \ldots, n$)

$$d_i n\mathbf{a}_i^{(k)} = \boldsymbol{\phi}(\mathbf{x}_k)^\top\mathbf{v}_i = \sum_{l=1}^{n}\mathbf{a}_i^{(l)}\boldsymbol{\phi}(\mathbf{x}_k)^\top\boldsymbol{\phi}(\mathbf{x}_l) = \sum_{l=1}^{n}\mathbf{a}_i^{(l)}k(\mathbf{x}_k, \mathbf{x}_l)$$

In total, $\mathbf{a}_i$ is a solution to the eigenvalue problem

$$\mathbf{K}\mathbf{a}_i = d_i n\mathbf{a}_i$$

Kernels and PCA (III)

The coefficients $\mathbf{a}_i$ to determine the principal component directions $\mathbf{v}_i$ in the space of the $\boldsymbol{\phi}(\mathbf{x}_i)$ can therefore be found by

- Solving the eigenvalue problem for $\mathbf{K}\mathbf{a}_i = d_i n \mathbf{a}_i$ requiring that

$$1 = \mathbf{v}_i^\top \mathbf{v}_i = \sum_{l,k=1}^{n} \mathbf{a}_i^{(l)} \mathbf{a}_i^{(k)} \boldsymbol{\phi}(\mathbf{x}_l)^\top \boldsymbol{\phi}(\mathbf{x}_k) = \mathbf{a}_i^\top \mathbf{K} \mathbf{a}_i$$

- This is the Rayleigh quotient problem for the matrix K. Note that both $\mathbf{a}_i$ and d_i have to be determined.

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

$$\boldsymbol{\phi}(\mathbf{x})^\top \mathbf{v}_i = \sum_{l=1}^{n} \mathbf{a}_i^{(l)} k(\mathbf{x}, \mathbf{x}_l)$$

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

Centring and kernel PCA

- The derivation assumed that the implicitly defined feature vectors $\boldsymbol{\phi}(\mathbf{x}_l)$ were centred. **What if they are not?**
- In the derivation we look at scalar products $\boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_l)$. Centring in the implicit space leads to

$$\left(\boldsymbol{\phi}(\mathbf{x}_i) - \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\phi}(\mathbf{x}_j)\right)^\top \left(\boldsymbol{\phi}(\mathbf{x}_l) - \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\phi}(\mathbf{x}_j)\right) =$$

$$\mathbf{K}^{(i,l)} - \frac{1}{n}\sum_{j=1}^{n} \mathbf{K}^{(i,j)} - \frac{1}{n}\sum_{j=1}^{n} \mathbf{K}^{(j,l)} + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{m=1}^{n} \mathbf{K}^{(j,m)}$$

- Using the **centring matrix** $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, centring in the implicit space is equivalent to transforming $\mathbf{K}$ as

$$\mathbf{K}' = \mathbf{J}\mathbf{K}\mathbf{J}$$

General algorithm for kPCA

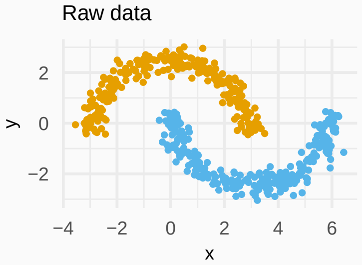
1. Choose a kernel $k(\cdot, \cdot)$ and possible hyper-parameters
2. Compute the Gram matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ for the data $\mathbf{x}_1, \dots, \mathbf{x}_n$
3. Centre $\mathbf{K}$ using $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ to get

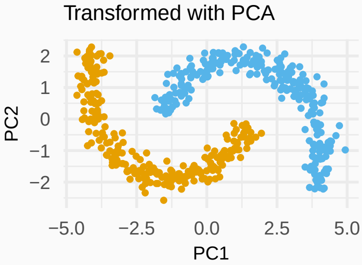
$$\mathbf{K}' = \mathbf{JKJ}$$

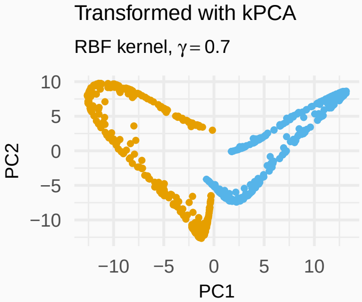
4. Perform a normal linear PCA on $\mathbf{K}' = \mathbf{A}\boldsymbol{\Lambda}\mathbf{A}^\top$.
5. The columns of $\mathbf{A}$ are the vectors $\mathbf{a}_i$ and set $d_i = \lambda_i / n$.
6. The projection of the l-th observation onto the i-th principal component axis is computed as

$$\eta_l^{(i)} = \mathbf{K}'^{(l,:)}\mathbf{a}_i \in \mathbb{R}$$

Example: kPCA







Kernel trick in other algorithms

Recap: Ridge regression

Ridge regression solves the problem

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

with analytical solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top\mathbf{y}.$$

The **kernel trick** requires scalar products between feature vectors. Note that

$$(\mathbf{X}\mathbf{X}^\top)^{(i,j)} = \mathbf{x}_i^\top\mathbf{x}_j$$

but here we have $\mathbf{X}^\top\mathbf{X}$.

Woodbury matrix identity

Assume that matrices $\mathbf{A} \in \mathbb{R}^{p\times p}$ and $\mathbf{C} \in \mathbb{R}^{n\times n}$ are invertible and let $\mathbf{U} \in \mathbb{R}^{p\times n}$ and $\mathbf{V} \in \mathbb{R}^{n\times p}$. The **Woodbury matrix identity** then holds

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$$

For a data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, let $\mathbf{U} = \mathbf{X}^\top$, $\mathbf{V} = \mathbf{X}$, $\mathbf{A} = \lambda\mathbf{I}_p$ for $\lambda > 0$, and $\mathbf{C} = \mathbf{I}_n$.

$$\begin{aligned}
\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p\right)^{-1}\mathbf{X}^\top &= \left(\frac{1}{\lambda}\mathbf{I}_p - \frac{1}{\lambda}\mathbf{I}_p\mathbf{X}^\top\left(\mathbf{I}_n + \mathbf{X}\frac{1}{\lambda}\mathbf{I}_p\mathbf{X}^\top\right)^{-1}\mathbf{X}\frac{1}{\lambda}\mathbf{I}_p\right)\mathbf{X}^\top \\
&= \frac{1}{\lambda}\mathbf{X}^\top\left(\mathbf{I}_n - \left(\lambda\mathbf{I}_n + \mathbf{XX}^\top\right)^{-1}\mathbf{XX}^\top\right) \\
&= \frac{1}{\lambda}\mathbf{X}^\top\left(\left(\lambda\mathbf{I}_n + \mathbf{XX}^\top\right)^{-1}(\lambda\mathbf{I}_n + \mathbf{XX}^\top) - \left(\lambda\mathbf{I}_n + \mathbf{XX}^\top\right)^{-1}\mathbf{XX}^\top\right) \\
&= \frac{1}{\lambda}\mathbf{X}^\top\left(\left(\lambda\mathbf{I}_n + \mathbf{XX}^\top\right)^{-1}(\lambda\mathbf{I}_n + \mathbf{XX}^\top - \mathbf{XX}^\top)\right) \\
&= \mathbf{X}^\top\left(\lambda\mathbf{I}_n + \mathbf{XX}^\top\right)^{-1}
\end{aligned}$$

Kernel ridge regression

Using the Woodbury matrix regression we get that

$$\hat{\beta} = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1}\mathbf{y}.$$

We can now replace $\mathbf{X}\mathbf{X}^\top$ with a **Gram matrix** $\mathbf{K}$ for an arbitrary kernel $k(\cdot,\cdot)$.

The variables $\hat{\beta}$ are called the **primal variables**. Define the **dual variables**

$$\widehat{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y} \quad \Rightarrow \quad \hat{\beta} = \mathbf{X}^\top \widehat{\alpha} = \sum_{l=1}^{n} \widehat{\alpha}^{(l)} \mathbf{x}_l.$$

Using the dual variables, computed with a chosen kernel, as weights for the observations to compute the primal variables is called **kernel ridge regression**.

Standard ridge regression is recovered when using the linear kernel

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}.$$

Prediction in kernel ridge regression

In normal ridge ression, we predict for unseen test data $\mathbf{x}$ as

$$\widehat{f}(\mathbf{x}) = \widehat{\boldsymbol{\beta}}^\top \mathbf{x} = \sum_{l=1}^{n} \widehat{\alpha}^{(l)} \mathbf{x}_l^\top \mathbf{x}$$

Using the **kernel trick** and replacing scalar products with kernel evaluations leads to

$$\widehat{f}(\mathbf{x}) = \sum_{l=1}^{n} \widehat{\alpha}^{(l)} k(\mathbf{x}_l, \mathbf{x})$$

for kernel ridge regression.

Take-home message

- 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
- The kernel trick can also be used in other established methods like ridge regression