Machine Learning and Intelligent Systems

Kernels (Part 2)

Maria A. Zuluaga

Dec 1, 2023

EURECOM - Data Science Department

Table of contents

Recap: Kernels So Far

Kernels

- We proposed to transform the input space to address the problem of non-linear separability
- We showed that thanks to the kernel trick it is possible to avoid the direct estimation of $\mathbf{w}$ by using inner products
- We showed that certain type of functions, the kernel functions, avoid the need to estimate inner products
- We introduced the kernel matrix
- Let's walk through it again (no proofs)

Learning without explicitly expressing $\mathbf{w}$

- We showed that $\mathbf{w}$ can be expressed as a linear combination of the training set $\mathcal{D}$:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \tag{1}$$

- This means we can perform gradient descent without expressing $\mathbf{w}$ explicitly
- We can also express the inner product of $\mathbf{w}$ with any $\mathbf{x}_j$:

$$\mathbf{w}^T \mathbf{x}_j = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{x}_j \tag{2}$$

Learning without explicitly expressing $\mathbf{w}$

- Thus, the loss can be reformulated as:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 \rightarrow \mathcal{L}(\alpha) = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \alpha_j \mathbf{x}_j^T \mathbf{x}_i - y_i \right)^2 \tag{3}$$

Learning without explicitly expressing $\mathbf{w}$

- Thus, the loss can be reformulated as:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 \to \mathcal{L}(\alpha) = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \alpha_j \mathbf{x}_j^T \mathbf{x}_i - y_i \right)^2 \tag{3}$$

- At test time, the hypothesis h will be used to make a prediction:

$$h(\mathbf{x}^*) = \hat{\mathbf{w}}^T \mathbf{x}^* = \sum_{j=1}^{N} \alpha_i \mathbf{x}_j^T \mathbf{x}^* \tag{4}$$

Input Transformation & Kernel Functions

- The formulations in Eqs. 1-4 holds also when we transform the inputs:

$$\mathbf{x} \rightarrow \phi(\mathbf{x})$$

- Transforming and estimating inner products can be a very expensive task
- There are certain functions, denoted **kernel functions**, that can be expressed as an inner product:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Kernel Matrix

- For a training set $\mathcal{D}$, the inner products can be pre-computed and stored in a **kernel matrix**:

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

- The stored kernel matrix $\mathbf{K}$ allows for fast computations during gradient descent. How?

Kernel Matrix

- For a training set $\mathcal{D}$, the inner products can be pre-computed and stored in a **kernel matrix**:

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

- The stored kernel matrix $\mathbf{K}$ allows for fast computations during gradient descent. How?
- In the transformed space we have

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$

Kernel Matrix

- For a training set $\mathcal{D}$, the inner products can be pre-computed and stored in a **kernel matrix**:
$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$
- The stored kernel matrix $\mathbf{K}$ allows for fast computations during gradient descent. How?
- In the transformed space we have
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$
- The term γ_i (see previous slide deck on kernels) can be reformulated:
$$\gamma_i = 2(\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)$$

Kernel Matrix

- For a training set $\mathcal{D}$, the inner products can be pre-computed and stored in a **kernel matrix**:

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

- The stored kernel matrix $\mathbf{K}$ allows for fast computations during gradient descent. How?
- In the transformed space we have

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$

- The term γ_i (see previous slide deck on kernels) can be reformulated:

$$\begin{aligned} \gamma_i &= 2(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i) \\ &= 2\left(\left(\sum_{j=1}^{N} \alpha_j \mathbf{K}_{ji}\right) - y_i\right) \end{aligned}$$

Gradient Descent & Predictions

- If we recall the update rule for α:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - \lambda\gamma_i^{(\tau)}$$

with λ the learning rate

Gradient Descent & Predictions

- If we recall the update rule for α:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - \lambda \gamma_i^{(\tau)}$$

  with λ the learning rate

- Using the redefinition of γ_i we obtain

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - 2\lambda \left(\left(\sum_{j=1}^{N} \alpha_j^{(\tau)} \mathbf{K}_{ji} \right) - y_i \right) \tag{5}$$

Gradient Descent & Predictions

- If we recall the update rule for α:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - \lambda\gamma_i^{(\tau)}$$

  with λ the learning rate

- Using the redefinition of γ_i we obtain

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - 2\lambda\left(\left(\sum_{j=1}^{N}\alpha_j^{(\tau)}\mathbf{K}_{ji}\right) - y_i\right) \tag{5}$$

- During testing,

$$h(\mathbf{x}^*) = \sum_{j=1}^{N}\alpha_j k(\mathbf{x}_j, \mathbf{x}^*) \tag{6}$$

- **Kernel trick:** We can perform learning and predictions in terms of inner products

General Kernels

Some Popular Kernels

Linear: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$ (same as linear classifier. Use?)

Polynomial: $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$

Radial Basis Function (RBF) or Gaussian Kernel: $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma^2}\right)$

Exponential: $k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{z}\|}{2\sigma^2}\right)$

Laplacian: $k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-|\mathbf{x} - \mathbf{z}|}{\sigma}\right)$

Sigmoid: $k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma \mathbf{x}^T \mathbf{z} + c)$

Designing Kernels

Kernel Functions

Not every function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ can be considered as a kernel.

The matrix $\mathbf{K}$ has to correspond to real inner-products after a transformation $\mathbf{x} \rightarrow \phi(\mathbf{x})$. This is the case if and only if $\mathbf{K}$ is a symmetric positive semi-definite matrix.

Kernel Functions

Not every function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be considered as a kernel.

The matrix $\mathbf{K}$ has to correspond to real inner-products after a transformation $\mathbf{x} \to \phi(\mathbf{x})$. This is the case if and only if $\mathbf{K}$ is a symmetric positive semi-definite matrix.

Symmetry: $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$

Kernel Functions

Not every function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be considered as a kernel.

The matrix $\mathbf{K}$ has to correspond to real inner-products after a transformation $\mathbf{x} \to \phi(\mathbf{x})$. This is the case if and only if $\mathbf{K}$ is a symmetric positive semi-definite matrix.

Symmetry: $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$

Positive semi-definite (PSD): A matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is positive semi-definite if and only if $\forall \mathbf{q} \in \mathbb{R}^N, \mathbf{q}^T \mathbf{A} \mathbf{q} \geq 0$.

Positive Semi-definite Matrix

Demonstrating that the kernel matrix $\mathbf{K}$ is PSD can be achieved in three different ways:

1. All eigenvalues of $\mathbf{K}$ are non-negative.
2. $\exists$ a real matrix $\mathbf{B}$ s.t. $\mathbf{K} = \mathbf{B}^T\mathbf{B}$.
3. $\forall$ real vector $\mathbf{q}, \mathbf{q}^T\mathbf{K}\mathbf{q} \geq 0$

$\mathbf{K}$: Gram matrix

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$, g a polynomial function with positive coefficient

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$, g a polynomial function with positive coefficient
5. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$, g a polynomial function with positive coefficient
5. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$
6. $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$, with f any function

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$, g a polynomial function with positive coefficient
5. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$
6. $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$, with f any function
7. $k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x}, \mathbf{z}))$

Well-defined Kernels

Well-defined kernels: Kernels built by recursively combining one or more of the following rules:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $c \geq 0$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, with k_1, k_2 well-defined kernels
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$, g a polynomial function with positive coefficient
5. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$
6. $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$, with f any function
7. $k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x}, \mathbf{z}))$
8. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{A} \mathbf{z}$, $\mathbf{A} \geq 0$ is PSD

Wrap-up

Wrap-up

- We reviewed the concept of kernel, kernel function and the kernel trick
- We presented a set of general and well-known kernels
- We introduced that a function needs to meet to be considered a kernel
- We presented some ways to design new kernels

Key Concepts

- Kernel trick
- Kernel function
- Kernel matrix
- Gram matrix
- General kernels
- Well-defined kernel

References

Further Reading and Useful Material

Source	Notes
Pattern Recognition and Machine Learning	Ch 6
The Elements of Statistical Learning	Ch 12